Modularity, antimodularity and explanation in complex systems

Luca Rivelli

Università degli Studi di Padova

FISPPA - Dipartimento di Filosofia, Sociologia, Pedagogia e Psicologia applicata

Scuola di dottorato di ricerca in filosofia

Ciclo XXVI

dottorato di ricerca in filosofia

Université Paris 1 Panthéon-Sorbonne

IHPST - Institut d'histoire et de philosophie des sciences et des techniques - UMR 8590

École doctorale 280 - philosophie

doctorat en philosophie

PhD Thesis (2015)

Modularity, Antimodularity and Explanation in Complex Systems

PhD Candidate:

Luca Rivelli

Supervisor (Padova):
Fabio Grigenti

Supervisor (Paris 1):
Philippe Huneman

Thesis Defended on: november 30th, 2015

To my father,

who first opened the doors
of my early curiosity

And to Gino Gerosa,

a real master in opening hearts
and new doors of life
when the old was closing

Modularity, Antimodularity and Explanation in Complex Systems

Luca Rivelli
2015

Università degli Studi di Padova, Department of Philosophy (FISPPA)

Université Paris 1 Panthéon–Sorbonne, Institute of History and Philosophy of Science and Technology (IHPST)

Abstract

Modularity, Antimodularity and Explanation in Complex Systems

This work is mainly concerned with the notion of *hierarchical modularity* and its use in explaining structure and dynamical behavior of complex systems by means of hierarchical modular models, as well as with a concept of my proposal, *antimodularity*, tied to the possibility of the *algorithmic detection* of hierarchical modularity. Specifically, I highlight the pragmatic bearing of hierarchical modularity on the possibility of *scientific explanation* of complex systems, that is, systems which, according to a chosen basic description, can be considered as composed of elementary, discrete, interrelated parts. I stress that hierarchical modularity is also required by the experimentation aimed to discover the structure of such systems. Algorithmic detection of hierarchical modularity turns out to be a task plagued by the demonstrated computational intractability of the search for the best hierarchical modular description, and by the high computational expensiveness of even approximated detection methods. *Antimodularity* consists in the lack of a modular description fitting the needs of the observer, a lack due either to absence of modularity in the system's chosen basic description, or to the impossibility, due to the excessive size of the system under assessment in relation to the computational cost of algorithmic methods, to algorithmically produce a valid hierarchical description. I stress that modularity and antimodularity depend on the *pragmatic* choice of a given basic description of the system, a choice made by the observer based on explanatory goals. I show how antimodularity hinders the possibility of applying at least three well-known types of explanation: mechanistic, deductive-nomological and computational. A fourth type, *topological explanation*, remains unaffected. I then assess the presence of modularity in biological systems, and evaluate the possible consequences, and the likelihood, of incurring in antimodularity in biology and other sciences, concluding that this eventuality is quite likely, at least in systems biology. I finally indulge in some metaphysical and historical speculations: metaphysically, antimodularity seems to suggest a possible position according to which natural kinds are *detected modules*, and as such, due to the computational hardness of the detection of the best hierarchical modular description, they are unlikely to be the best possible way to describe the world, because the modularity of natural kinds quite probably does not reflect the best possible modularity of the world. From an historical point of view, the growing use of computational methods for modularity detection or simulation of complex systems, especially in certain areas of scientific research, hints at the envisioning of a multiplicity of emerging scientific disciplines guided by a self-sustained, growing production of possibly human-unintelligible explanations. This, I suggest, would constitute an historical change in science, which, if has not already occurred, could well be on the verge of happening.

Résumé

Modularité, antimodularité, et explication dans les systemès complexes

Ce travail concerne principalement la notion de *modularité hiérarchique* et sa utilisation pour expliquer la structure et le comportement dynamique des systèmes complexes au moyen de modèles modulaires hiérarchiques, ainsi que un concept de ma proposition, l'*antimodularité*, relié à la possibilité de la *détection algorithmique* de la modularité hiérarchique. Plus précisément, je mets en èvidence la portée pragmatique de la modularité hiérarchique sur la possibilité de l'*explication scientifique* des systèmes complexes, c'est-à-dire, systèmes qui, selon une description de base choisie par l'observateur, peuvent être considérés comme composés de parties élémentaires discrètes interdépendantes. Je souligne que la modularité hiérarchique est essentielle même au cours de l'expérimentation visée à découvrir la structure de ces systèmes. Mais la détection algorithmique de la modularité hiérarchique se révèle être une tâche affectée par la démontrée intraitabilité computationnelle de la recherche de la meilleure description modulaire hiérarchique, et par l'excessive cherté computationnelle même des méthodes de détection approximatifs de la modulairité. L'*antimodularité* consiste en le manque d'une description modulaire aproprièe aux exigences de l'observateur, manque dû ou à l'absence de modularité dans la description basique choisie du système, ou à l'impossibilité de produire algorithmiquement une description hiérarchique valide, en raison de les dimensions excessives du système à évaluer en relation à la cherté computationnelle des méthodes algorithmiques. Je souligne, de plus, que la modularité et l'antimodularité dépendent du choix *pragmatique* d'une spécifique description de base du système, choix fait par l'observateur sur la base de ses objectifs explicatifs. Je montre comment l'antimodularité entrave la possibilité d'appliquer au moins trois types bien connus d'explication: mécanique, déductive-nomologique et computationnelle. Un quatrième type, l'*explication topologique*, reste par contre indemne. Ensuite j'evalue la présence de modularité dans les systèmes biologiques, avec ses possibles conséquences, et l'éventualité d'encourir dans l'antimodularité en biologie et en autres sciences: éventualité assez probable, au moins dans la biologie des systèmes. Je me permet enfin quelques spéculations métaphysiques et historiques plutôt libres. D'un point de vue métaphysique, l'antimodularité semble suggérer une position possible, selon laquelle les espèces naturelles sont *modules qui ont été détectés* et, en raison de l'intraitabilité computationnelle de la détection de la meilleure description modulaire hiérarchique, il est improbable qu'ils constituent la meilleure façon possible de décrire le monde, parce que la modularité des espèces naturelles assez probablement ne reflète pas la meilleure modularité possible du monde. D'un point de vue historique, l'utilisation croissante des méthodes computationnels pour la détection de la modularité ou pour la simulation de systèmes complexes, en particulier dans certains domaines de la recherche scientifique, suggère la possibilité d'imaginer une multiplicité de disciplines scientifiques émergentes, guidées par une production croissante et auto-alimentante d'explications potentiellement inintelligibles pour les capacités cognitives humaines. Cela, à mon avis, constituerait un changement historique dans la science, qui, s'il n'a pas déjà eu lieu, pourrait bien être sur le point de se produire.

Abstract

Modularità, antimodularità e spiegazione nei sistemi complessi

Questo lavoro riguarda principalmente il concetto di *modularità gerarchica* e il suo impiego nello spiegare la struttura e il comportamento dinamico di sistemi complessi mediante modelli modulari gerarchici, nonché un concetto di mia proposta, l'*antimodularità*, legato alla possibilità del *rilevamento algoritmico* di modularità gerarchica. Nello specifico, evidenzio la portata pragmatica della modularità gerarchica sulla possibilità di *spiegazione scientifica* dei sistemi complessi, cioè sistemi che, secondo una descrizione di base scelta dall'osservatore, possono essere considerati come composti da parti elementari discrete interrelate. Sottolineo che la modularità gerarchica è essenziale anche nel corso della sperimentazione volta a scoprire la struttura di tali sistemi. Il rilevamento algoritmico della modularità gerarchica si rivela essere un compito affetto dalla dimostrata intrattabilità computazionale della ricerca della migliore descrizione modulare gerarchica, e affetto dal comunque elevato costo computazionale anche dei metodi di rilevamento approssimati della modularità. L'*antimodularità* consiste nella mancanza di una descrizione modulare adatta alle esigenze dell'osservatore, mancanza dovuta o all'assenza di modularità nella descrizione di base del sistema scelta dall'osservatore, o all'impossibilità di produrre algoritmicamente una sua descrizione gerarchica valida, per le dimensioni eccessive del sistema da valutare in rapporto al costo computazionale dei metodi algoritmici. Sottolineo che modularità e antimodularità dipendono dalla scelta *pragmatica* di una certa descrizione di base del sistema, scelta fatta dall'osservatore sulla base di obiettivi esplicativi. Mostro poi come l'antimodularità ostacoli la possibilità di applicare almeno tre tipi noti di spiegazione: meccanicistica, deduttivo-nomologica e computazionale. Un quarto tipo di spiegazione, la *spiegazione topologica*, rimane sostanzialmente immune dalle conseguenze dell'antimodularità. Valuto quindi la presenza di modularità nei sistemi biologici, e le sue possibili conseguenze, nonché l'eventualità di incorrere nell'antimodularità in biologia e in altre scienze, concludendo che questa eventualità è abbastanza probabile, almeno in biologia dei sistemi. Infine, mi permetto alcune speculazioni metafisiche e storiche piuttosto libere. Dal punto di vista metafisico, l'antimodularità sembra suggerire una posizione possibile secondo cui i generi naturali sono *moduli che sono stati rilevati*, e in quanto tali, a causa dell'intrattabilità computazionale del rilevamento della migliore descrizione modulare gerarchica, è improbabile che essi siano il miglior modo possibile per descrivere il mondo, perché la modularità dei generi naturali molto probabilmente non rispecchia la migliore modularità possibile del mondo. Da un punto di vista storico, il crescente utilizzo di metodi computazionali per il rilevamento della modularità o per la simulazione di sistemi complessi, in particolare in alcuni settori della ricerca scientifica, suggerisce la possibilità di immaginare una molteplicità di discipline scientifiche emergenti, guidate dalla produzione di spiegazioni potenzialmente inintelligibili dal punto di vista cognitivo umano, produzione che potrebbe iniziare ad autoalimentarsi, portando potenzialmente ad una crescita inarrestabile. Suggerisco che questo scenario costituirebbe un cambiamento epocale nel campo della scienza, che, se non è già avvenuto, potrebbe benissimo essere sul punto di realizzarsi.

Mots clés:
modularité, modularité hiérarchique, antimodularité, émergence antimodulaire, émergence explicative, systèmes hiérarchiques, quasi-décomposabilité, systèmes quasi-décomposables, explication scientifique, intelligibilité des explications, intelligibilité de l'explication, explication, modèles d'explication, modèles de l'explication scientifique, explication mécanistique, déductive-nomologique, explication déductive-nomologique, explication computationnelle, explication topologique, relativité à l'observateur, systèmes complexes, détection de la modularité, détection de modularité, émergence faible, imprévisibilité des systèmes complexes, imprévisibilité, détection algorithmique de la modularité, computation, complexité computationnelle, difficulté calculatoire, simulation informatique, simulation par ordinateur, descriptions, description, niveaux de decription, niveau de description, mise en œuvre, implémentation, mécanismes, mécanisme, systèmes dynamiques discrets, automates cellulaires, espèces naturelles, biologie des systèmes, philosophie de la biologie, philosophie de la science, philosophie de la computation, systèmes complexes, science des réseaux, antiréalisme contraint, Herbert Simon, Herbert A. Simon, H. A. Simon, William Bechtel, Carl Craver, Cory Wright, Mark A. Bedau, Mark Bedau, Robert Cummins, Michelle Girvan, Mark E.J. Newman, Carl G. Hempel, Stuart A. Kauffman, Philippe Huneman

Keywords:
modularity, hierarchical modularity, antimodularity, antimodular emergence, explanatory emergence, hierarchical systems, near-decomposability, nearly-decomposable systems, scientific explanation, intelligibility of explanations, intelligibility of explanation, explanation, models of explanation, models of scientific explanation, mechanistic explanation, deductive-nomological, deductive-nomological explanation, computational explanation, topological explanation, models of scientific explanation, relativity to the observer, complex systems, modularity detection, weak emergence, unpredictability of complex systems, unpredictability, algorithmic detection of modularity, computation, computational complexity, computational hardness, computational intractability, intractability, computer simulation, descriptions, description, levels of decription, level of description, implementation, mechanism, discrete dynamical systems, cellular automata, natural kinds, systems biology, philosophy of biology, philosophy of science, philosophy of computation, philosophy of computing, complex systems, network science, constrained antirealism, Herbert Simon, Herbert A. Simon, H. A. Simon, William Bechtel, Carl Craver, Cory Wright, Mark A. Bedau, Mark Bedau, Robert Cummins, Michelle Girvan, Mark E.J. Newman, Carl G. Hempel, Stuart A. Kauffman, Philippe Huneman

IHPST - Institut d'histoire et de philosophie des sciences et des techniques
UMR 8590 Université Paris 1 Panthéon-Sorbonne / CNRS / ENS
13 rue Du Four, 75006 Paris, France

FISPPA - Dipartimento di Filosofia, Sociologia, Pedagogia e Psicologia applicata
Università degli Studi di Padova
Palazzo del Capitanio, Piazza Capitaniato 3, 35139 Padova, Italy

Un résumé substantiel en langue française est présent dans l'Appendice de cette thèse.

Un riassunto sostanziale in lingua italiana è presente nell'Appendice di quest'opera.

Contents

Part I

Introduction

Chapter 1

Modularity, Antimodularity, Explanation: an Introductory Tour

In this work, I am mainly concerned with the notion of *hierarchical modularity* in complex systems, its algorithmic detection and its use in *explaining* structure and dynamical behavior of such systems by means of hierarchical modular models. Specifically, I highlight the pragmatic bearing of hierarchical modularity on the possibility of *scientific explanation* of complex systems, that is, systems which can be described as composed of distinct elementary related parts. I stress that hierarchical modularity must be considered a *relative* notion, dependent on the choice, on the part of the observer, of a specific, basic, *preferred description* of the system, which consists of a representation of the system as a set of interrelated atomic parts. In such a kind of description, modularity basically manifests itself as the possibility of *decomposing* the system into recognizable, sufficiently defined and persistent subsystems (the modules) each one composed of parts which are more strongly related to each other than to parts belonging to other modules or the external environment. Actually, in my view, hierarchical modularity concerns not the real, physical system per se, but only its possible *descriptions* and *descriptions of descriptions*. These are theoretical models of a system, and I concentrate here on modular descriptions *of the models*, leaving mostly aside the thorny question of the relation between model and modeled phenomenon, namely, the relation between the empirical phenomenon and its first description: these problems would certainly deserve a thorough separate treatment which cannot be provided here, albeit in the end my proposal will in some way touch upon even that kind of questions.

After having considered the defining properties of hierarchical modularity, I concentrate on known algorithmic methods for its *detection*, that is, algorithms which, given a complex system (under the form of its *preferred description*, its description as a set of several interrelated parts), try to yield a hierarchical, modular re-description of the system. Once detected, hierarchical modularity appears to be the crucial feature of a system's description allowing for multi-level functional or mechanistic explanations of the system, which are important forms of explanation, widely used in science.

Along these lines, I subsequently focus on the opposite property, the absence of hierarchical modularity, which I call *antimodularity*, trying to draw the consequences of its possible manifestation in certain system descriptions. Antimodularity is a complex property, arising in a series of possible circumstances, whose main features are that of being, like modularity, dependent on the observer's choice of a preferred basic description of the system, but also, importantly, that of being dependent on some *computational constraints* of possible algorithms employed for modularity detection: most of these algorithms are highly computationally demanding, and there are

even theoretical results about the computational *hardness* of the search for an optimal modular description of a system. This computational complexity inescapably hinders the search for modularity in systems of scientific interest of a large enough size. I propose to call the effect of this hindrance *antimodular emergence*, by analogy with some known forms of computational emergence. I conclude that antimodular emergence entails (with some qualification) Mark Bedau's *weak emergence*, which is another form of computational emergence.

After having defined this new type of computational emergence, that is, antimodular emergence, which is due to the amount of computational complexity algorithms for modularity detection can manifest in certain cases, I try to draw some possible consequences of antimodular emergence on the possibility of *scientifically explaining* systems affected by it.

I take into consideration three classic models of scientific explanation: deductive-nomological, mechanistic, and computational explanation, plus a novel model, recently proposed by Philippe Huneman, topological explanation. I conclude that antimodular emergence affects the feasibility of all these types of explanation, albeit in different ways.

First and foremost, I claim that antimodularity negatively affects mechanistic explanation, a fundamental form of explanation in biological sciences. Taking side with Cory Wright and William Bechtel for an *epistemic* view of mechanistic explanation (as opposed to an *ontic* view), I show how antimodularity compels to single-level-only explanations, neglecting the need, essential for mechanistic explanations, of multi-level integration. The fact of limiting mechanistic explanation to the level of description representing the most elementary parts of the system, certainly hinders comprehension: for large enough systems, their mechanistic explanation at this level is too complex to be understood by human beings. And, understandability is a quality to be sought for in mechanistic explanation, at least according to William Bechtel and to other authors, too, as Petri Ylikoski, who considers "cognitive salience" one of the important features of explanations.

For what concerns classic *deductive-nomological* (*DN* henceforth) explanations, I show that, *antimodularity entailing weak emergence* in complex enough systems[1], deductive-nomological Hempel-style explanation for an antimodular system can not be recurred to, because, if it could, it would mean that the system is predictable by means of a law, and this is negated by the very definition of weak emergence, which, as said, is entailed by antimodularity. Thus, a complex antimodular system is not predictable, at least not predictable arbitrarily ahead of time by means of an analytical law, and, thus, it cannot be explained by a DN explanation. Anyway, if we take into consideration a specific type of complex dynamical system, that is, a Cellular automaton (*CA* henceforth), then an antimodular process generated by a CA can be explained, in a way, by producing a possibly very long list of deductions based on the initial condition and the CA rule (which, as a CA rule, for the Curtis-Hedlund-Lyndon theorem, has the same logical form of a scientific *law*), in a way resembling a long list of DN stepwise explanations. In this case as well, human comprehension is excluded by the possible length of the list, but, if we stick to the expectations of the post-neopositivistic advocates of the DN model of explanation, understanding is not required for a good explanation. So, in a way, antimodularity and, consequently, weak emergence, does not hinder DN explanation, at least in the case of CAs and other systems whose dynamics follow a *universal*, law-like rule, and as long as understandability is kept out of consideration.

I proceed to consider Philippe Huneman's topological explanation, a non-mechanistic type of explanation which is based on topological properties of certain abstract descriptions of a system.

[1] A qualification which is made clear in section 13.3.

I conclude that, being modularity itself, as well as its absence, topological properties, the presence or absence of modularity does not hinder, but *enables* the possibility of topological explanation.

I then focus on a third possible type of explanation: CAs and dynamic boolean networks can be considered *computational systems*. As such, they can be subject to *computational explanation*. I consider the case of trying to computationally explain a CA. To obtain a computational explanation, first the CA's behavior must be *seen* as a computation. I endorse an *intentional* view of computation, but subject to some mathematical constraints, and I try to delimit the range of system dynamics which can be seen as computational. Given that some CAs can indeed be seen as computational, I try to assess the possibility of their computational explanation. As it stands, to give such an explanation, the CA's behavior must be reverse-engineered in order to obtain a *specification* of the computation it is supposed to perform. But, this task of *specification mining* can be computationally hard, and so can fail. Even if a global specification is found, a good computational explanation amounts to a form of hierarchical modular functional analysis, and this is obtained by recursively mining specifications of parts of the code or subjecting the system to other methods for static or dynamic reverse-engineering. Should this process fail for reasons of computational hardness, or for lack of completeness of the found functional hierarchy, the system would end up being antimodular. In this case, antimodularity would hinder an understandable form of computational explanation, for the same reason it affects mechanistic explanation, with which computational explanation, which is a form of functional analysis, shows a strict affinity.

I subsequently stress the need for hierarchical and high-level modularity not only for *a posteriori* explanation of a known phenomenon, but also during the phase of scientific discovery, specifically, as already noted by James Woodward, during the search for causal relationships between parts of a mechanism both at low and at a higher level. Likewise I emphasize that multilevel modular explanation is also essential during the development of computer programs on the part of human programmers.

I subsume under the concept of *explanatory emergence* all the results about the unfeasibility of certain multilevel explanations and on the consequent fading of understandability due to occurrence of antimodularity, as well as any other case in which a system, for computational reasons, resists understandable explanation.

I then discuss, by examining some scientific literature, the likelihood for scientific research in certain areas to incur antimodular emergence, concluding that it is quite likely that some cases of antimodularity appear, especially in systems biology.

I dedicate the final part of this Introduction to more ample and, possibly, less supported and risky considerations. First, I sketch a possible metaphysical view that could stem from the considerations about antimodularity exposed before: I call this view *constrained antirealism*. It sees the empirical world we naturally perceive, as well as the world described by science, as the result of a process of modularity detection, in consequence of which the detected modules constitute what are commonly known as *natural kinds*. Given that modularity detection is constrained by factors of insurmountable computational complexity, and that for this reason the finding of the *best* modular description is precluded in principle, it is not likely that the world's subdivision in natural kinds corresponds to is best possible subdivision. Full evaluation of this metaphysical position requires however a wide discussion of a controversial hypothesis, *pancomputationalism*, and of variuos positions in philosophy of mathematics, a discussion which is better left to a subsequent work.

Finally, I take in this chapter some liberty in drawing the possible, alleged consequences on history of science of a recent and growing recourse to computational methods in science, starting

with simulation of complex systems: I reflect on the plausibility of simulations as explanations, especially in cases in which the system is antimodular, and consequently simulation can be executed, but the underlying dynamical model is unintelligible, because the system is simulated at a very low-level and a modular, high-level redescription is lacking. I then consider automated modularity detection, used to find structure in big datasets, based on real cases of data mining on a corpus of medico-biological literature, in which the automated system discovered important functional relations which had escaped human examination. Possibly indulging in drawing some extreme consequences, I conclude by suggesting that this growing use of computational methods in science could be on the verge of provoking a major paradigm shift in some discipline.

The subject matter of this thesis is multifaceted and not easy to label: being about the consequences of antimodularity, a property of certain systems, on possible scientific explanation of these systems, it is a work of philosophy of science. Given that the proposed property, antimodularity, depends on certain computational constraints affecting modularity detection, and that I recommend, in relation with the discussion on computational explanation, an intentional conception of what computation is, then this is a work of philosophy of computing, in the double sense of putting certain computational notions to philosophical use, and of proposing a philosophical reflection on the notion of computation itself. Considered that systems showing antimodularity are likely to be found amongst biological systems, the long-standing discussion in biology about modularity, and a host of examples I report from that discipline, it is even a work in philosophy of biology. Regarding explanation, I explicitly embrace an epistemic position, centered on the notion of *levels of description*, which are epistemic devices, so the present work has an epistemological facet. And, as probably every epistemological position, it has also a metaphysical import, which I try to sketch towards the end of this chapter. Finally, this dissertation makes use of all the above mentioned theoretical discussions to shed light on their possible consequences on the practice of science, broaching the possibility that a major, historical change, possibly a change of paradigm, is undergoing or on the verge of occurring in science. So, in a sense this is a thesis of history of science. Although still requiring observation and evidence, I think this historical hypothesis could give us a hint of the magnitude of the impact that the widespread adoption of computational methods has had or could be about to have on science as we know it.[2]

1.1 Modularity

I begin this clarification of the notion of modularity with a historical sketch: modularity appears to be a basic and widespread concept, that has probably been conceived more than one time, in partially independent and diverse theoretical and practical fields since long time. Nevertheless, modern philosophical reflection upon it began in the second half of twentieth century, with the especially relevant contribution of Herbert Simon. Working in the field of econometrics, he came to a conception of modularity under the form of hierarchical *nearly decomposable* systems[3], that is, systems that can be seen, at least as a first approximation, as recursively, hierarchically decomposable into sets of robust, partially independent, subsystem. This view on near-decomposability, which has subsequently influenced many other authors in different fields, is the basic idea which inspires my proposal on modularity.

[2] An in-depth analysis of the notion of hierarchical modularity and antimodularity, as well as a host of considerations regarding the epistemological implications I just mentioned, are conducted in the main chapters to follow. I am going to give here a shorter survey of the main contents of the work. I advise the reader that, in what follows, I will quite often make use of the terms "modularity" and "system" instead of "hierarchical modularity" and "complex system", leaving to the context the task of disambiguating their meaning.

[3] See the seminal Simon (1962).

In this work, I examine a possible conception of modularity in complex systems, and explore the consequences of the presence of modularity or of its absence (antimodularity) on the explanation of the behavior of such systems. Actually, I do not apply the concept of modularity to the actual, real systems, but to their *descriptions*, and to re-descriptions of descriptions, where a (re)description is understood, preferentially, as a computation taking a description and processing it in order to yield another description. Taking a widely epistemic stance, if not a fully antirealistic one (a position which will be better explained in section 1.5.1 of this introduction), along the lines of Cory Wright and William Bechtel's *epistemic* position on mechanistic explanations[4], I consider scientific explanations as *epistemic* devices, based on *descriptions* of phenomena, related to human *communication*, and requiring at least a minimum degree of cognitive *intelligibility*. Accordingly, I am interested in defining modularity as a feature *of descriptions*, which, if present, allows for certain comprehensive types of explanation. While section II is dedicated to a thorough exposition of modularity and other involved concepts, I give here a schematic explanation of what I want to propose.

1.1.1 Modularity in complex systems

Proceeding along the lines expressed above, I try to define the property of *modularity* in complex systems, as the possibility for a system of this kind to be described as a set of loosely related *modules*, that is, a set of well-defined, robust subsystems, with internal parts highly interconnected, each subsystem partially independent from the external context, being only weakly connected to other subsystems. I extend this view of modularity to that of the full *hierarchical* description of a system in terms of "higher" and "lower" levels of description, each of which is constituted by modules, and where, except for the lowest level, each of the modules at one level is a *macromodule*, that is, it can in turn be seen as internally characterized by a modular organization of *micromodules*, and so on recursively. As said, all of this concerns descriptions, not sets of real-world objects (this is in line with the essence of an epistemic view).

While the macro/micro modularity distinction is of course dependent on the choice of a particular level of description, the point to highlight here is that the whole hierarchical modular description turns out to depend, due to the definition itself of modularity, on the observer's *choice* of a specific significant *relation* between the elementary parts of the system, and this precisely because of the way the concept of *module* is defined: a module is a subset of the parts of a whole that are *related* to each other in a stronger way than how they are related to parts external to the module they are in. Recognition of a subset as a module requires thus that a *relation* between parts is taken into consideration first, and, depending on which specific relation is considered, the identifiable modular structure can change.

This definition of hierarchical modularity of course presupposes that a complex system is composed of distinguishable, related, elementary parts, and this in turn is due to the choice of an atomic, elementary description of the system: the choice of the set of *parts* and that of the *relation* holding between them amounts to the choice, on the part of the observer, according to her interests, of what I would call a *preferred description* of the system. Usually, there is a "natural" lowest-level description of a system in terms of elementary parts, often suggested by physical properties of the system combined with the researcher's interests: for example, in biology a tissue is naturally described as composed of cells, a cell is naturally described as a complex system composed mostly of interacting macromolecules, and in social sciences a society is naturally described as composed of individuals. The point to highlight is that hierarchical

[4] An *epistemic* position opposed to an *ontic* conception of causal explanations. See section 1.1.9, 1.4.3, as well as Bechtel & Abrahamsen (2005) and Wright (2012).

modularity is *relative* to such a choice, depending especially on the choice of the *relation* holding between the system's elementary parts, which usually is a less constrained choice than that of the parts themselves. For example, in a society we can consider affective ties between individuals, or, alternatively, we could choose relations of subordination. These two different descriptions of the system would most probably result in different hierarchical modular descriptions, because a module is defined as a subsystem of highly interconnected elements weakly connected with the surrounding environment, and this "connection" is precisely the relation between elementary parts considered in the chosen preferred description of the system: in the example cases, one relation is the relation of affective tie, the other that of power influence.

1.1.2 Modularity, decomposability, and economy of description

Modularity manifests as the possibility of *decomposing* a system[5] into recognizable, sufficiently defined subsystems, each one composed of parts which are more strongly related to each other than to parts belonging to other modules or to the external environment. It is the presence of these variations in strength of the relations holding between couples of parts of the system, which allows for the recognition of modularity: if all parts were fully connected to each other, modules would not appear, because a module is (informally) defined as a subsystem whose strength of connection with the rest of the system is *lower* (on average) than that of the connection between the module's internal parts. As noted above, the resulting modularity is relative to the specific relation between low-level parts which we are taking into consideration. This is a conception quite similar to the original one by Herbert Simon, that of *near decomposability*. Near decomposability allows the original system to be represented as a set of connected subsystems, and this decomposition can be reiterated until the obtainment of a full hierarchical description. The crucial point is that the original system, composed of its elementary parts, is thus describable in a high-level manner, under the form of *another* system whose parts correspond each one to one of the original system's modules. So, the high-level description turns out to be *simpler* than the low-level one, because, in the former, entire *groups* (the modules) of low-level parts are represented as *single* high-level parts, and so the parts of the higher level description are fewer in number than the low-level ones. If the system we are describing this way is static, like for example the list of the members composing the personnel of an organization, the high-level description appears usually more economical and perspicuous than the original list. The typical example is that of organizational charts. In an organizational chart, each group of persons working in the same office is represented by a single item, labeled with the name of the office. The office name represents the aggregate name of the group of persons working in the office.

1.1.3 Modules as repeated similar high-level parts

There is a further possible improvement in the economy of description of a complex system if it is possible to detect in it more subsystems which end up being identical or so similar to be possibly considered as the repetition of a single template. In this case, apart from the economy of description due to aggregation mentioned in the preceding section, even the *modular* description, which would comprise more than one identical modules, can be simplified by substituting every occurrence of these modules with a reference to the common template, which is then necessary to describe only once. This form of modularity is especially useful in engineering, and it is essentially at the basis of the design of complex artifacts, which are usually composed of identical or almost identical *standard* parts, occurring in multiple copies.

[5] Of course, with "system" here I mean a *description* of a system. In what follows, I will often use the term "system" simpliciter to mean its standard description, usually the "preferred" one.

1.1.4 Structural and dynamical modularity

It is easily conceivable that modularity can not only concern the *structure* of a system, but also its dynamical functioning: it is conceivable, for example, and even obvious, that modularity in the structure of a computer program (whose structure is a list of instructions) brings about modularity in its dynamical execution, because a computer program is not only a list of static instructions, but it is supposed to be *executed*, so modularity of the list should be reflected in the program's dynamical modularity.

Considering the relation between structural and dynamical modularity, this turns out to be not always a simple relation: the structural and dynamical aspects can be associated but also decoupled, albeit in most cases of dynamical systems their modular physical structure *induces* a form of modular dynamical functioning, given that in dynamical systems the dynamics is conducted *on* the system's predefined structure and it is thus constrained by it. The relation between structural and dynamical modularity is not however completely plain and in section 6 of this dissertation I will more deeply consider and discuss it.

1.1.5 Near-decomposability and Aggregability

A form of dynamical modularity first proposed in the early '60s by Herbert Simon and Albert Ando[6] derives from the system's near decomposability, and specifically from near-decomposability of the *mathematical model* describing the system's dynamics. This mathematical model is usually a recurrence relation, or a system of recurrence relations, in which the state of each elementary part of the system is represented by a variable: this equation represents an update function, with time as the independent variable, which determines how the state of the system's parts varies as time flows, and thus it is a mathematical model of the system's dynamics. In a system which is nearly decomposable in Simon's sense, the variables of this equation, which can be in great number because they represent the elementary interacting parts of the system, can be as well subdivided (modulo a certain approximation) into a partition of subsets of variables, each subset containing variables which influence variables inside other subsets only weakly: this corresponds to the fact that in a nearly-decomposable system, by definition, the interaction between certain groups of parts (that is, between the modules) are only weak. This way, each module's dynamics can be considered as evolving in time semi-independently from the dynamics of the other modules, and, accordingly, the equations describing these semi-independent dynamics turn out to be semi-independent one from the other. These equations governing semi-independent groups of variables can then be considered *functional modules*, a modular re-description of the original mathematical model describing the system's global dynamics. In nearly decomposable systems, their modularity determines also a kind of dynamical, or *process* modularity, under the form of a decoupling of temporal dynamics between parts of the system: dynamics *inside* modules is faster than the dynamic of interactions *between* modules.

Given the above conditions, in certain favorable cases which depend on the form of the modular equations, the system's global dynamics, originally described by the global update function, in which the state of every elementary part is described by a single variable, can be, modulo a certain amount of accepted approximation, further re-described under the form of another, *simpler* global update function. This update function is simpler than the original one, because in the new update function each variable represents an *aggregate value* of all the variables contained in each of the functional modules described above: the number of variables which must be taken into consideration to model the global dynamics of the system is thus reduced. When this condition

[6] Simon & Ando (1961).

holds (not every dynamical system is aggregable), the system is said to be *aggregable*, and this is evidently another form of *economy of description*, in this case economy of the mathematical model, allowed by the presence of modularity. The price to pay is an amount of approximation which depends on the fact that, in order to successfully aggregate the system's dynamics, certain interactions between parts of the system whose strength falls below a chosen threshold, are considered as null. The approximation could end up as being unacceptable in non-linear systems, where long-term behavior of the simplified description could diverge too greatly from the actual behavior of the system. The point to highlight is that, even here, choices on the part of the observer are involved: the choice of the preferred description (which, however, in many cases, is already given), and a choice about the approximation, acceptable or not depending on the observer's goals.

A quite important problem which affects aggregability is that the latter has turned out to be a computationally unfeasible task: there are proofs, in Kreinovich & Shpak (2006), and Kreinovich & Shpak (2008), that aggregability, and even *approximate* aggregability, already in *linear* systems, is *NP-hard*. This means[7] that there is no general algorithmic method that, applied to a mathematical model of the system's dynamics, can always produce in a feasible time a plausible aggregate, simplified version of the model, for models with a big enough number of variables. In other terms, this means that modularity detection in the dynamical model of a complex system is a computationally unworkable task, and so that *it is not to be expected, in general, that dynamical modularity can be found with a general method.*

Nevertheless, aggregation can in many cases be found more easily if we have some prior knowledge which can guide us in the partitioning of variables into semi-independent subsets. For example, in the case of genetic networks, we could know on empirical grounds that some group of genes always co-express, and so the variables representing these genes can be grouped together. This could simplify a lot the task of finding a good aggregation, a task which in principle is, as said, too demanding.

1.1.6 Modularity in discrete dynamical systems

There are complex cases in which the structural and the dynamical form of modularity are not easily separable, because some high-level structure of the system, itself "emerges"[8] from the complex low-level dynamics of the system. This is typical of certain complex *discrete dynamical systems*, such as certain boolean networks, or some cellular automata. While I dedicate some sections of chapter 5 to explain the basics of discrete dynamical systems, and more specifically of a subclass of them, the so-called *cellular automata* (*CAs* henceforth), a very short overview can be given here: such systems are composed of a number of simple parts, each of which, at any given time, happens to be in a particular *state*, chosen inside a finite set of possible distinct states. It is customary to think of each distinct state as a *symbol*, and to consider the set of possible symbols as an *alphabet* (think, in the simplest case, of the *0* and *1* symbols). Not only symbols are discrete, but so is time: in these discrete systems time proceeds by distinct *timesteps*, which we can call t_1, t_2, and so on. At any given time, the set of the states into which all the parts of the system happen to be, constitutes the global *configuration* of the system. The states of all the parts of the system are synchronously updated at each successive time step according to some deterministic rule, a rule which can be the same for all the parts of the system (as is the case in CAs), or different for each part. At a conventional initial moment, let's call it t_0, the

[7] See section 1.3.

[8] I use the term "emergence" here in an intuitive way, albeit this will be discussed briefly later on and, more deeply, in the following chapters.

system is in the *initial configuration*. The system's evolution is the sequence of successive global configurations it reaches as time goes, starting from the initial configuration. Typical classes of such systems are, as said, CAs, and a more comprehensive class, that of boolean networks. The dynamics of such evolution can, for certain systems, be extremely complex, in some cases demonstrably equivalent to the computational power of universal Turing machines, which are deemed to be[9] the most powerful class of computational systems. For this reason, the behavior in time of complex systems is, in general, quite difficult to predict, and, in the case of Turing-level capabilities, it is in principle *algorithmically undecidable* in general[10].

A form of modularity can be induced or appear in certain discrete dynamical systems, either by imposing on them a specific initial state, or, in some cases, by its spontaneous emergence in the system after a certain time along its evolution, regardless of the specific initial configuration: a phenomenon which is a form of *self-organization*. Modularity in this sense amounts to the fact that certain subsets of the global configuration of the system come to be partially, or totally, *frozen* after a certain time, that is, they come to constitute unchanging or little-changing parts of the configuration, this way partially isolating other subsets of the configuration, by hindering the spread of influence from each of these subsets to the others. This, implicitly, imposes a virtual high-level structure above the original low-level structure, a superstructure which can be seen as a set of dynamical modules (the unfrozen parts of the configuration) loosely connected with each other (by means of the residual connection paths which are not interrupted by the frozen parts). For an example of a discrete network with high-level modularity appearing during its evolution, see fig. 1.1.

In another, slightly different manner, self-organization can appear, especially in CAs, as the emergence of highly localized, partially robust, well delimited, only partially changing subconfigurations of the global configuration, the so-called *gliders*, which appear, as it were, to *move* across the system's configuration. An example is in fig. 1.2.

1.1.7 Modularity in computational systems

Being a form of discrete dynamical system, a computational system can of course exhibit modularity. Common real-world universal computers are highly modular machines already at the so-called "hardware" level. But another very important form of modularity concerns computer *programs*. A program is basically constituted by a lists of instructions which the computer hardware "executes" step by step. Of course, such a list can be devoid of apparent modularity, or it can instead be structured by the programmer in an obviously modular way, by subdividing it into disjoint sublists, each of which contains mostly instructions concerning only a limited set of variables internal to the sublist, except for an "input" and an "output" set of variables which are accessed also by instructions in other sublists. This way, each of these sublists can be considered a module, and the limited and controlled transfer of information between different modules is realized by the input/output variables, which are separate sets of variables that are the only ones to be accessed and manipulated by parts of the program external to the module: such a module can be considered a "black box" with a limited set of input and output lines. This way, the typical property of modules is realized: by considering as the chosen relation between parts of the list of instructions the relation between an instruction and a variable it acts upon, it can be easily seen that a sublist of instructions whose internal variables, those not included in the input and output sets, are of mostly internal use, and are less often or (better) never manipulated by external disjoint sublists, can be considered a module, endowed with internal

[9] Taken the Church-Turing thesis for granted. For an explanation, see the Appendix, section 17.3.
[10] As a consequence of the undecidability of the Halting problem. See section 17.2.6.

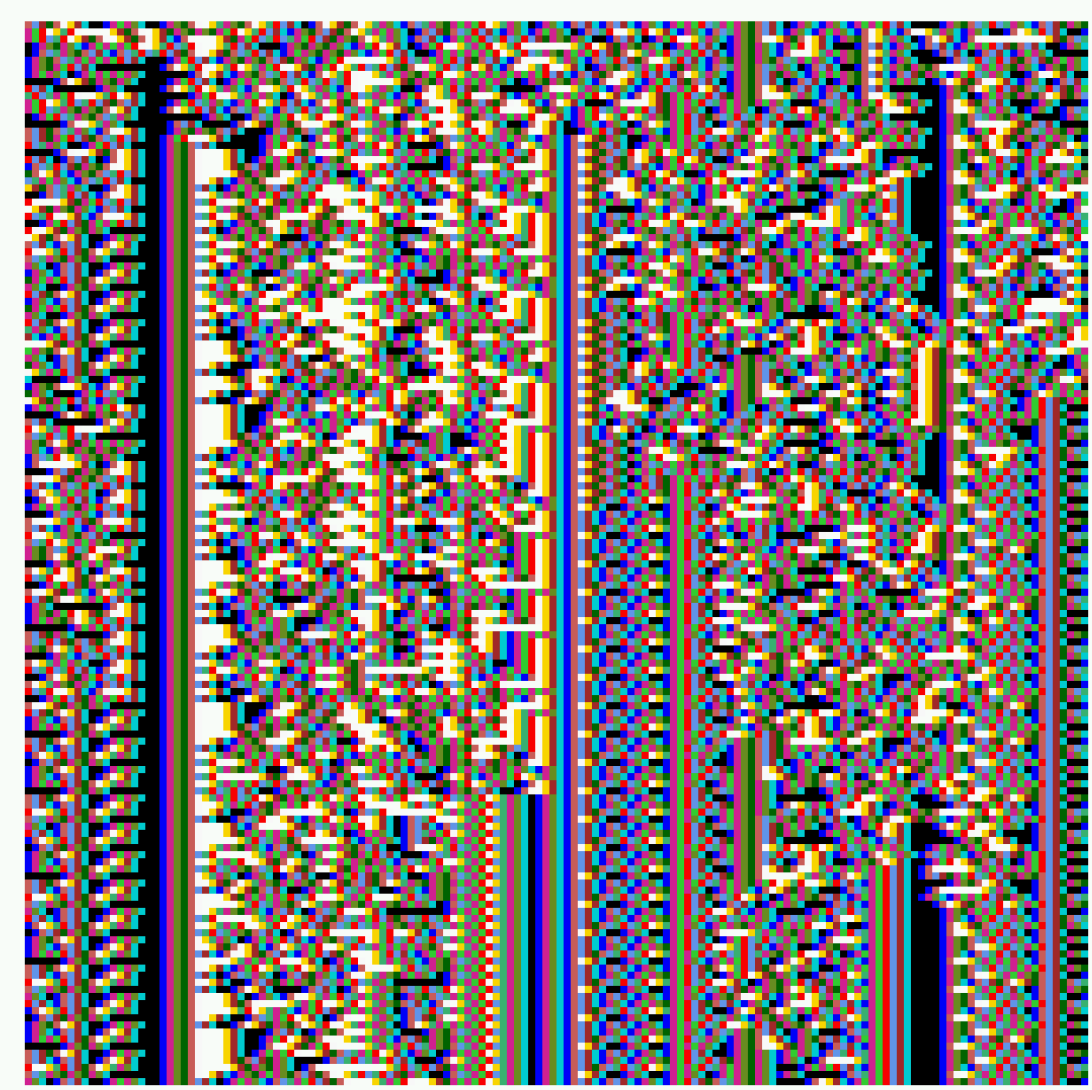

Figure 1.1: a partial evolution in time of a discrete network. Time flows from top to bottom, each row of pixels representing the configuration of the system at each timestep. Each pixel represents the state of one of the elementary parts of the network, its *nodes*. Vertical thick lines, black or patterned, which can be distinguished in the picture, are "frozen" subsets of the configuration. They induce a form of high-level modularity, by acting as more or less impenetrable "walls", this way rendering the system nearly-decomposable into several independent subsystems. (Image taken from Andrew Wuensche's DDLab Gallery, http://uncomp.uwe.ac.uk/wuensche/gallery/ddlab_gallery.html).

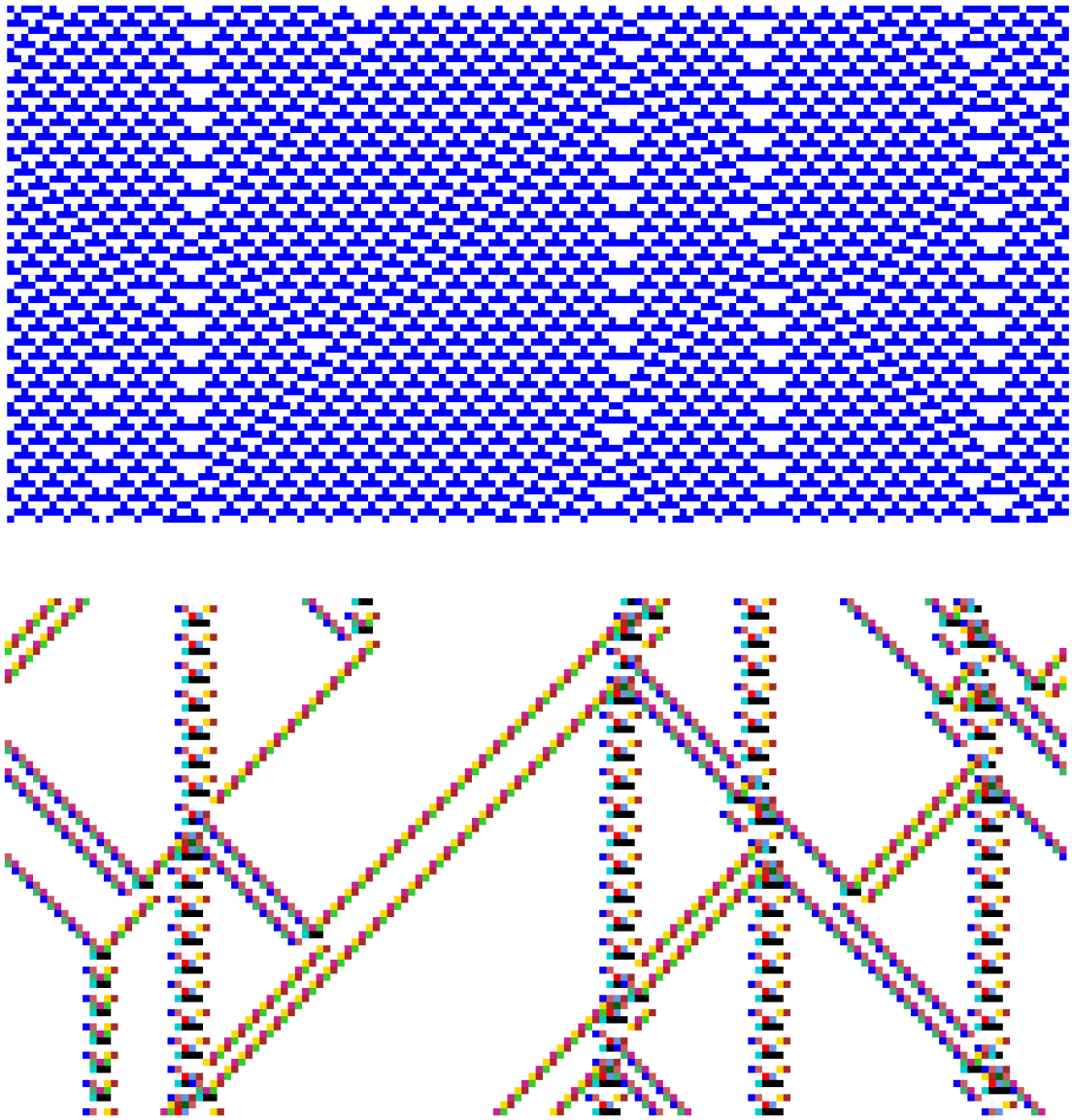

Figure 1.2: gliders in a CA, the so-called *Rule 54*, according to Stephen Wolfram's classification (see Wolfram 2002). Time flows from top to bottom, each row of pixels representing the global configuration of the system at each timestep. Each pixel represents the state of one of the elementary parts of the CA, its *cells*. Above: the CA sequence of states. Bottom: after having filtered out repeated background parts of the configuration, gliders appear more clearly, represented during time by straight lines, depicting the progressive relocation of these subconfigurations inside the global configuration (Image originally from Andrew Wuensche's DDLab Gallery, http://uncomp.uwe.ac.uk/wuensche/gallery/r54_filted.gif, modified).

cohesion and structurally quite independent from other modules. In a non-modular program, modification of the program on the part of the programmer can result quite difficult to carry out and control, because a change in a part of the program could affect potentially very distant parts. By contrast, given the limited connectivity between modules, especially in the case when only input and output variables are accessible to external modules, a change internal to the module, and affecting only internal variables, would not spread indiscriminately to other modules, and so its effects are easier to control. In general, in computer programming, what is sought for is *high cohesion* of modules and *low coupling* between them.

Modularity in the form of recurring similar or identical parts is at the basis of a related form of modularity of computer programs, compatible with the one sketched above: if more than one of the above described modules performs the same input/output function, instead of explicitly reiterating the module in the form of repeated copies of the sequence of its instructions inside the program's instruction list, the module can be written in the list only once, and can simply be invoked repeatedly in different parts of the program, by "calling" it each time with distinct input configurations, and by fetching possibly distinct output configurations as the module terminates its execution. Modules seen this way, as callable subprograms, are often called *subroutines* or *procedures*. This way of structuring programs can improve enormously their reliability, because testing of each procedure can then be done only once, as the complete system is assembled starting from the already tested modules.

1.1.8 Hierarchical modularity, levels, robustness and validity

I will try to clarify here the importance of a property of modules which I have only mentioned until now: *robustness*. Intuitively, for a modular description of a dynamical system, robustness means that a module at a higher level must endure a certain range of perturbations at the lower level, maintaining its distinct identity and persisting despite variation of state in its constituting components. Or, in certain cases, a robust module should remain the same despite possible substitutions of some of its low-level components. In modular descriptions of networks[11], which are *structural* modular descriptions, a module is considered robust when it does not change its identity despite the adding or subtracting of some links from the underlying low-level components[12].

In dynamical systems, persistence in the face of variations would of course make a module which can comprise *any* random set of lower level components a very robust module: it would not change despite any change at the lower level! But this would be a *trivial* kind of robustness. An explanatory adequate dynamically robust module should be not too evanescent, but neither should it be fixed in state: it should instead be able to follow, in a coarse-grained manner, the underlying dynamics. Adequate robustness of modules is essential when producing hierarchical descriptions of dynamical systems which we want to employ in scientific explanation, because a high-level module, to be explicative, must be supposed to faithfully track the low-level dynamics, albeit at a lower temporal and often spatial resolution, or at a lower precision. A module in the description of a dynamical system, to be useful for scientific explanation, should be robust to a degree which is chosen by the observer, according to her interests: this is another aspect of the relativity of hierarchical modularity to the observer's choice. In any case, a high-level module should not be too evanescent, otherwise its level of description would be unusable for explanations, nor should it be too robust, for in this case there would be no explanatory effect

[11] See section 1.2.1.

[12] Certain network modularity detection algorithms perform this kind of test in order to assess modular robustness. See section 3.2.1.2.

at all: the most robust module in absolute is a completely frozen module, one which stays in the same state for every possible combination of all the parts of the system, and such a module would simply constitute a "name" of the overall system (this could be useful in some cases, though, in order to identify phenomena for the first time).

But, high-level modular redescription in dynamical systems is a complicated matter: the modular high-level description must not *diverge* (at least not too much) in its dynamics with respect to the underlying low-level dynamical evolution of the system at the preferred description. A modular description must be *valid* (to use the terminology of scientific computer modeling), in order to be explanatory useful: it must track with a certain precision, albeit at a coarser-grained scale, the dynamical evolution of the system. A non-valid modular description is explanatory useless. This could be better understood with an example. Imagine we run a discrete dynamical system one time, obtaining a section of its evolution: refer to fig. 1.2. We could immediately devise a high-level modular representation of this dynamical run by simply taking each one of the diagonal lines representing a glider trajectory as a module, and then produce a high-level explanation of the given segment of evolution by mentioning only "glider trajectories" (along the lines of "the third glider collides with the second and destroys it, continuing toward the leftmost big vertical column", and so on...). Would that constitute a *valid* high-level description of the system's dynamics? Most probably not, because we would have based our detection of modularity only on a limited segment of the system's possible evolution: next time we start the system with an even slightly different initial configuration, it will with the highest probability produce a completely different segment of evolution, and the former description would be rendered useless. The point is, when redescribing such kinds of dynamical systems at a higher level, we must not seek to redescribe ephemeral, unique patterns of their evolution, but only adequately *robust* subsystems which appear in the system's dynamics with high frequency and regularity: only this way we could hope to obtain a valid high-level modular description[13]. In terms which will be clarified in section 1.4.3, we could likewise say that a module should ideally represent, at high-level, a *function*, performed in order to contribute to the overall behavior of the system. Functional decomposition is not arbitrary: not any arbitrary dissection of a system into random parts can be considered functional. An arbitrary decomposition, leading to an invalid description, would not be considered explicative in a functional or mechanistic way.

1.1.9 Modularity and explanation

It appears that modularity is linked with explanation in various and fundamental ways. Already Herbert Simon's first papers on nearly-decomposable systems highlight that the formula representing the aggregate dynamics[14] of a nearly-decomposable system is *simpler* than the formula of its original dynamics, and this means that aggregability produces economy of description. Since a scientific explanation of the system's dynamics (at least a deductive-nomological type of explanation[15]) would surely employ this formula, this achieves economy of explanation.

In general, modularity should allow for a form of *coarse-graining*, understood as the operation of taking a complex system represented as a set of many parts, partitioning this set in disjoint subsets, and considering, in place of the original system, another set in which each part corresponds to one of the disjoint subsets. This is basically the same operation, whether effected

[13] It can be objected that robustness of high-level modules can be obtained by mapping in a complex and non immediately evident manner heterogeneous sets of ephemeral low-level subconfigurations to high-level modules. This, I think, is not a trivial objection, especially given the importance I ascribe to the *relativity* of modularity. I will examine this question, which has profound implications, in section 14.5.2.

[14] See section 1.1.5.

[15] See section 14.2.

on sets of variables of an equation, as in aggregation, where it brings dynamical modularity, or on a network, where the original representation can be substituted with a network with fewer nodes, or in the case of functional and mechanistic explanations[16], where a group of interacting parts or actions can be seen as a whole function, or mechanism, and a group of mechanisms can be seen as a single super-mechanism, whose parts are the simpler single mechanisms. This in a way holds also for modularity in computational systems, where a list of instructions can be rewritten in a higher-level language in which each single high-level instruction corresponds to an entire sequence of lower-level ones: moreover, that of a high-level programming language is a very typical example of coarse-graining. In each of these cases economy of description is achieved, and arguably, understandability of the explanation is greatly eased.

Another form of economy of description is attained in certain modular descriptions when more copies of a single recurrent module can be substituted by a single citation of the general model of that module, a form which, in computer programs, corresponds to calling the same subroutine from different points of the program.

Moreover, considerations of economy or intelligibility aside, modularity is *necessary* to produce certain types of explanation. Robert Cummins' analytical explanatory strategy, that we will touch upon in section 1.4.3 along with mechanistic explanation, explicitly pleads for a hierarchical decomposition of the system's functioning, in order to explain it. Of course, this decomposition is possible just in the case some form of functional modularity is present in the system, that is, when the modules to be sought for can legitimately be considered functional modules. Similarly, the idea of a mechanistic explanation seems to require the finding of a coincidence between two kinds of hierarchies, that is between a structural and a functional description of the system, at least in the conception of Mechanism put forth by William Bechtel and his group: for these authors, which do not see mechanistic explanation as merely reductionistic, it is essential that the explanation be *multilevel*, and this corresponds to a hierarchical functional-mechanistic description of the system. Embracing an *epistemic* view of explanations[17], these same authors quite naturally highlight also the importance of the cognitive *intelligibility* of explanations, and this can be achieved by the modularity of the descriptions employed in explanations.

So, in the first place, it seems that at least explanations of a certain kind, namely mechanistic or functional explanations, *require* modularity, even when neglecting issues about the intelligibility of these explanations.

But, hierarchical modularity allows also for multilevel explanations which certainly enhance *comprehension*. Given an appropriate mechanistic hierarchical decomposition, a system can be described at any desired level of description, with different results on the intelligibility of the explanation: the more abstract, coarse-grained levels allow for a very simplified explanation, which usually induces better understanding, while the choice of proceeding down to lower, more detailed levels, enhances the information on the system conveyed by the explanation, possibly at the cost of understanding: the most detailed possible explanation is the one which describes the system in terms of the bottom-level entities[18], and, in many cases, the sheer amount of information contained in such a description could hinder its intelligibility.

[16] See section 1.4.3.
[17] See section 1.4.3.
[18] Bottoming out itself, which corresponds in my terminology to the reaching of the preferred description, is usually a matter of choice or convention, also according to Bechtel and his co-authors. See sections 1.4.7 and 11.1.5.

1.1.10 Some example applications of modularity in real-life scientific research

I will concentrate here on some brief considerations about the importance of modularity in biological thinking and researching, because biology is one of the fields in which modularity has more been at the focus of attention in recent times. An obvious observation is that organisms are undoubtedly modular at many levels: they are, in a biological view, roughly composed of systems, organs, cells, macromolecules. It is less obvious if modularity holds at certain intermediate levels which can be seen as complex systems, comprising many parts: for example, are the genome, the proteome, or the metabolic network modular?

So, a first question to pose is: does evolution produce modular architectures and dynamics in organisms? And, if this is the case, has modularity evolved by natural selection or for other reasons? Outside the possible empirical study of this problem, some a priori considerations have seemed capable of illuminating it, at least since Herbert Simon's times. There is a number of arguments pointing to the conclusion that natural selection should actually conduct to modular organization[19], all these arguments basically stemming from the following line of reasoning: in a non-modular, completely integrated organism, in which each part potentially affects each other, evolutionary change in one part could affect and possibly disrupt functions performed by the other parts, and, given this, the number of evolutionary attempts potentially needed to obtain a still functional organism after a change in one of its parts would be enormous, so it is arguable that, if this were the case, natural selection would not have had the time, despite the geological scale of the actual evolutionary times, to bring about the evolution of complex systems. This is more or less the general argument for evolution of modularity started in the '60s by Herbert Simon[20] and adopted, with variations, by many subsequent authors. Starting with works by Stuart Kauffman in the early '90s, an alternative (but in my opinion not so dissimilar, see section 7.1.2) argument has appeared, which, while affirming modularity of biological systems, denies its direct origin by natural selection: modularity is instead a self-emergent property of a certain class of dynamical complex systems, a property coinciding with the "freezing" of some of their dynamical subsystems (see above, section 1.1.6), which arises not by direct selection but in virtue of the intrinsic, mathematical features of these systems[21]. The genome (seen as a complex of interacting parts, that is the genetic regulatory network) of an organism can, according to Kauffman, be considered, with some approximation, belonging to this class of systems endowed with a tendency to make modularity emerge spontaneously, a class which turns out to be the class of most *evolvable* systems: natural selection's role would have been that of meta-selecting the *class* of evolvable systems on which, then, to operate its finer, analytical selective role, as classically conceived in Darwinism[22], and this class is the class of complex systems which, spontaneously, show some form of modularity.

Thus, it seems, all considered, that there are reasons for which evolved biological systems should have preferentially a modular organization. Many of these systems are so complex and composed of so many parts, that detection of their *functional* modularity, allowing their multi-level explanation, would be of great help also in *understanding* those systems.

[19] Arguments better analyzed in section 7.

[20] With the famous parable of the two watchmakers, see section 7.1.1 and, of course Simon (1962).

[21] This kind of explanation supplied by Kauffman can be seen as a form of *topological* explanation, along the lines of the model recently proposed by Philippe Huneman. See section 1.4.5 and Huneman (2010).

[22] I take the occasion here for a disclaimer: even when talking of natural selection in intentional terms, I am not advocating considering it as an intentional subject. This is only, obviously, a useful "façon de parler", widespread in philosophy and biology.

In biology, since the late '90s, some proposals on the possibility of seeing complex biological systems as composed of functional modules have been directly inspired by the engineer's viewpoint on artificial systems, especially electrical circuits: among the most prominent proposals about this view are McAdams & Shapiro (1995), and Hartwell, Hopfield, Leibler, & Murray (1999)[23]. This view has been applied to genetic and metabolic networks, where the high specificity of electrical connections between components of an electronic circuit is substituted by the specificity of the relation between a protein and its ligand, and the whole biological network is being represented as a *digital circuit* , which is equivalent (with some differences, taking into account signal propagation delays) to some boolean network. In these circuits, modules are the separate instantiations of *standard components*, connected by wiring, and the complete digital circuit can be viewed as a hierarchical structure, in which each level is describable as a circuit of interconnected, modular, repeatable parts, which enable high-level digital modules to realize virtually any digital circuit, even ones capable of being viewed as computational systems. See fig. 1.3 for an exemplification of the hierarchical view of an electronic digital circuit.

The modular parts can be, in the case of genetic networks, single genes and, at a higher level, complex of genes like the *operons* of bacterial genomes. Such higher level components would have a functional role, as is the case, for example, of an operon, which controls the production of a complex of enzymes carrying out a specific metabolic function. A possible schematic representation of a "genetic circuit" is reported in in fig. 1.4.

Hartwell et al. (1999) proposed that the linguistic terms ("amplification", "error correction", "coincidence detection", and so on..) corresponding to mid and high-level functions performed by modules at intermediate hierarchical levels, come to constitute a vocabulary of terms, essential for the functional description of biological systems.

1.2 Algorithmic detection of modularity

I examine here known algorithms for *modularity detection* in certain classes of complex systems, that is, algorithms which, given a complex system and a preferred elementary description of it, try to produce a hierarchical modular description of the system.

1.2.1 Modularity detection In networks and computational complexity

I take into consideration, specifically, algorithms for modularity detection in *networks*, because network models have emerged as one of the preferred ways of representing complex systems, especially biological systems, in recent research. A network can in general be seen as a set of parts, its *nodes*, connected to each other in various ways through *links* (a possible graphical representation of a network is fig. 1.5). There are two main, not incompatible, possible forms of modularity in networks: community structure and network motifs[24]. While the first is based on the typical conception of modularity as weakly connected robust subsystems, the second coincides with the idea of modules as repeatable standard parts.

In chapter 3 I make a detailed survey of the main proposed methods for detection of the two types of modularity, with a special attention to their computational feasibility: it turns out that most of the best algorithms for modularity detection in networks are computationally very demanding, and there is also a theoretically established limit on their accuracy. To sum up,

[23] For a meta-reflection about the methods of biological research, modularity, and the engineer's approach, see also Lazebnik (2002).

[24] See section 3.

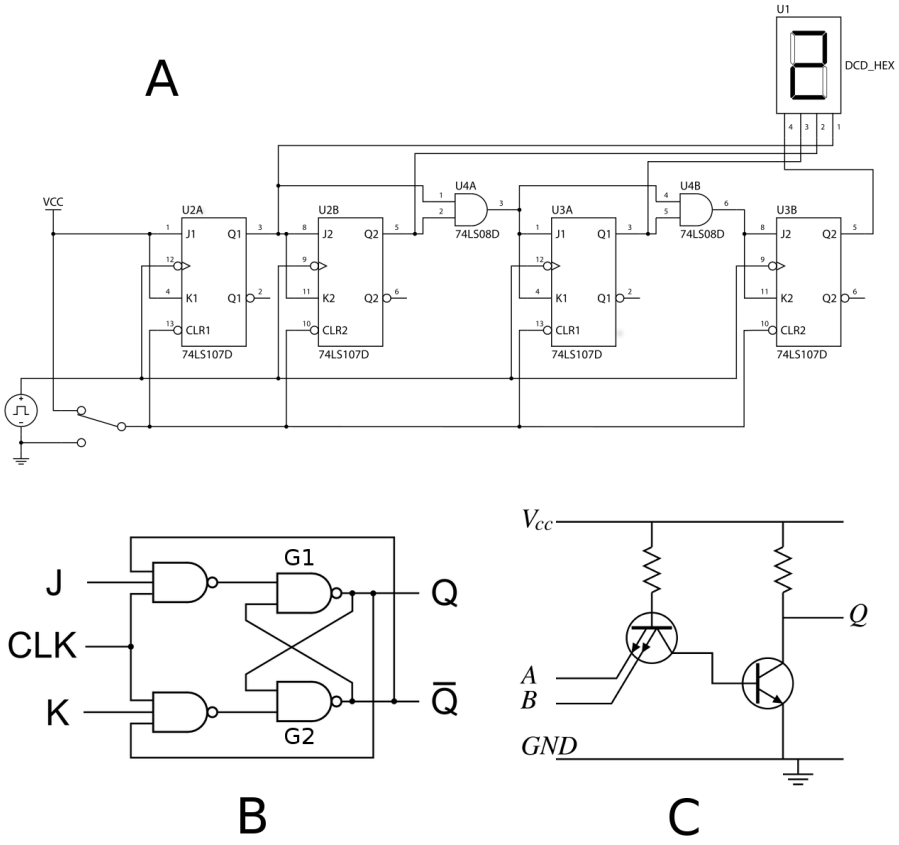

Figure 1.3: image A: a high-level diagram representing a digital circuit. Except for a few single logic gates ($U4A$ and $U4B$), most components are higher-level ones, and can be considered modules performing higher-level functions. In this case, each of the components labeled $U2A$, $U2B$, $U3A$, $U3B$ is a so-called *J-K flip-flop*, which is a type of 1-bit memory cell. Each flip-flop can be seen (image B) as internally composed of a certain number of simpler elements, namely NAND logic gates. Each of the two-input NAND gates labelled as $G1$ and $G2$ in image B are internally structured as a circuit composed of transistors and resistors, as in image C. Of course, a description at level higher than that of flip-flops is plausible: for example, the whole circuit of image A can be defined as a module performing the function of a single digit counter, which counts the impulses sent to its input line and displays the counted number in the display labeled DCD_HEX. As a module, this circuit can be employed as a standard part in other, larger circuits. (Images A, B and C taken from Wikipedia Commons, respectively at http://commons.wikimedia.org/wiki/File:4_bit_counter.svg, http://commons.wikimedia.org/wiki/File:JK-FlipFlop_(4-NAND).PNG and http://commons.wikimedia.org/wiki/File:TTL_npn_nand.svg).

it has been proved that the automated finding of the *best* modular description of a system is hindered by an insurmountable computational time complexity: the task is *NP-complete*[25]. Moreover, it turns out that most algorithms for simply *approximating* the optimal detection of modularity in networks are themselves highly computationally intensive. In general, it appears that the algorithmic detection of network modularity is affected by a trade-off between complexity of the task and dependability of the modular description produced, and for this reason the identification of approximate but acceptable hierarchical descriptions is algorithmically possible only for systems of limited size.

[25] See section 1.3 and the Appendix, section 17.4.

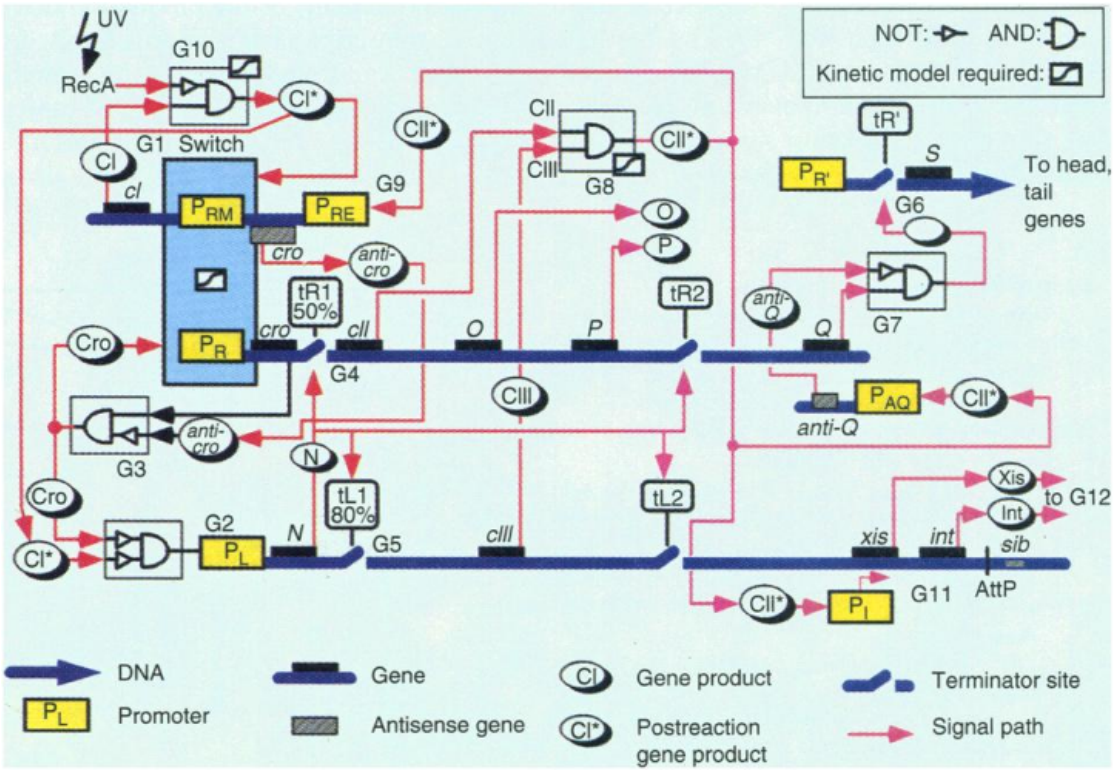

Figure 1.4: schematic representation of the genetic circuit generating the λ phage lysis-lysogeny dynamics. (The phage is a virus affecting bacteria. Image taken from McAdams & Shapiro 1995, p. 652).

1.2.2 Modularity detection In discrete dynamical systems and computational systems

In accordance with the considerations on dynamical modularity sketched above, I proceed to consider forms of dynamical modularity in some kinds of discrete, computational dynamical systems. The possibility of *detection* of dynamical and computational modularity in these kinds of systems, which can often be considered computational systems at the Turing machine level, turns out to be plagued by algorithmic undecidability or, at least, by hardness or high computational complexity. A first introductory discussion on this question is to be conducted in what follows, and more thoroughly in later chapters.

1.2.3 Some Applications of modularity detection in real-life researches

Given that functional and structural modularity, even if conceptually distinct, are often related, methods of automatic modularity detection in networks, which apply to the network *structure*, could, if applied to network representations of a biological system, yield an immediately functional modular description. The frequent coincidence between structural and functional organization in biological systems is confirmed by many works, notably, among others, by a series of researches by Zhou and Lipowsky[26], in which one of the best methods for modularity detection in networks is applied to the protein-protein interaction network of yeast, producing a modular description

[26] See Zhou & Lipowsky (2004) and Zhou & Lipowsky (2006).

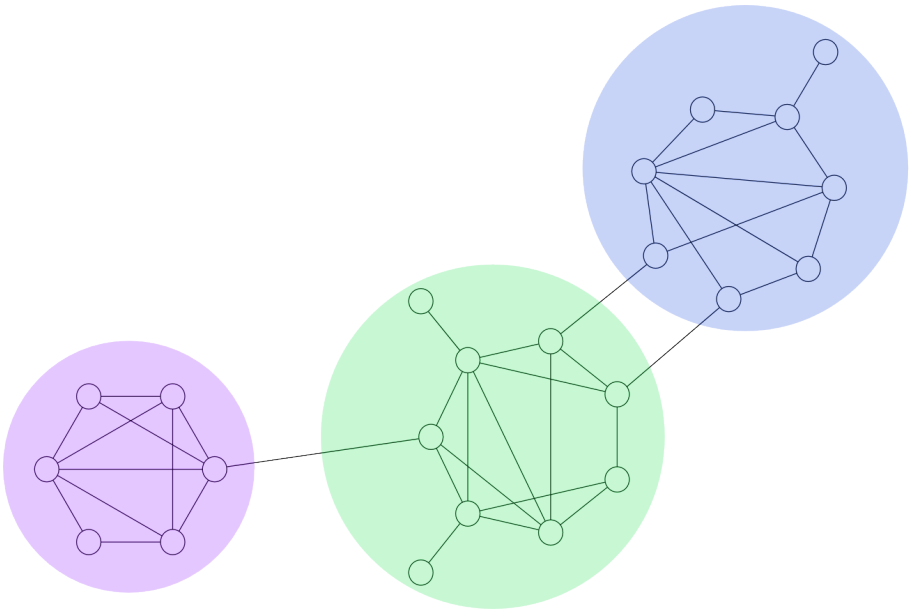

Figure 1.5: a network with community structure. In this picture, colored discs surround the communities, which show high density of intra-module links, while external, inter-module links, are more sparse.

comprising 449 modules, which turn out to correspond to already well-known functional subsystems, and which are components of an even higher-level modular description. Another important work highlighting coincidence between structural and functional modularity in biological systems is Guimerà & Amaral (2005b), which applies to metabolic networks an algorithm for modularity detection that identifies modules and then assigns them an alleged functional role based on structural intra and inter-module connectivity. The identified functional modules have roles which turn out to to be correlated quite well with the actual biological functions that the metabolites corresponding to each module actually fulfill in the whole metabolic network. See fig. 1.6.

1.3 Computational hardness

Computational hardness is a pragmatic limitation of certain computational tasks, which basically consists in the fact that they cannot possibly be brought to completion if the size of their input data exceeds certain limits[27]. This means that, in general, the computational task in question, while executable *in principle*, could be never brought to an end in human, or even astronomically feasible time, if the size of the input exceeds a certain magnitude. This is typical, for example, for problems which have execution time proportional to an exponential function of their size: even for small sizes of their input, their completion time can grow to unfeasible values, because exponential functions grow very steeply. So, even if computational hardness is not a limitation *in principle*, it certainly is an unsurmountable limitation from a pragmatic point of view. The most typical classes of computational complexity which can be considered hard are the classes of the so-called *NP-complete* and *NP-hard* algorithmic problems.

[27] This is *time complexity* of the program, which is not the only type of computational complexity. Other types of complexity and a better treatment of all the subject can be found in section 17.4.

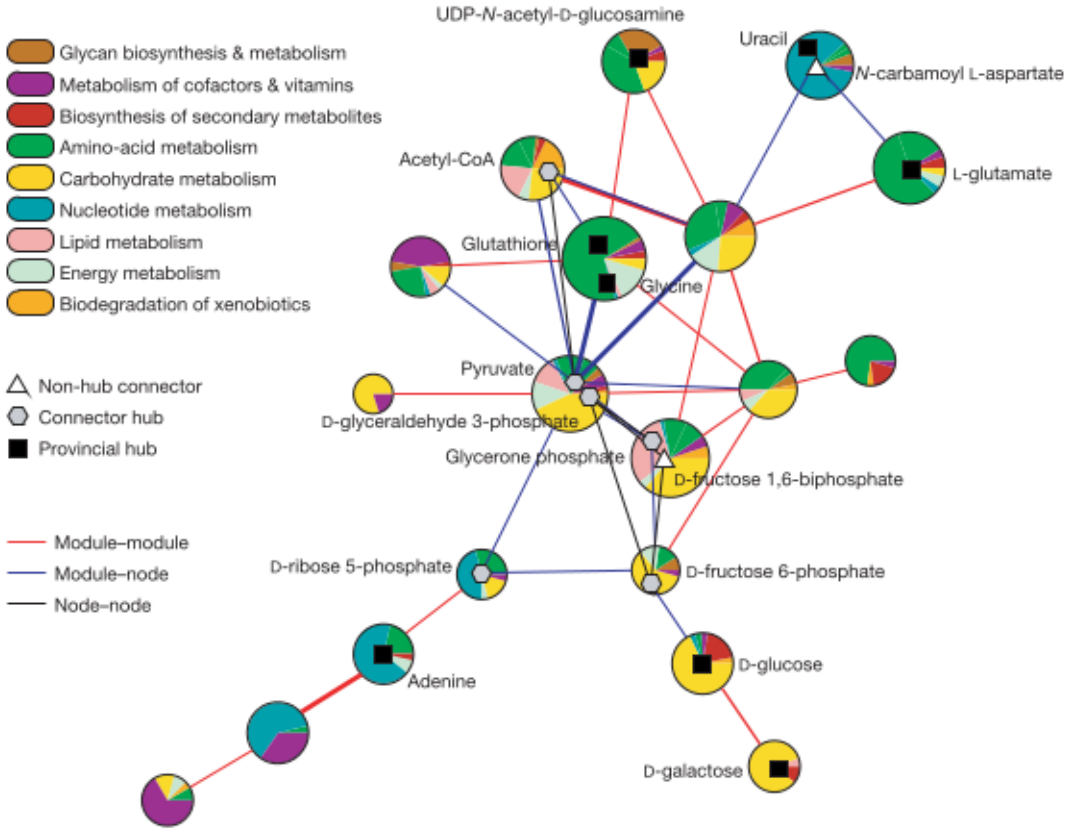

Figure 1.6: high-level modular representation of a metabolic network. (Image taken from Guimerà & Amaral 2005b).

Some algorithmic tasks which are not to be considered formally hard, can nevertheless be too computational demanding to be of practical applicability. This happens for example when a task requires a number of steps which is proportional to some integer power of the input size: for example, n^4, where n is the input size. In these cases, given a sufficiently big input, the program would take certainly too long to complete for it to be of any practical use.

Computational hardness and computational practical expensiveness are two basic notions upon which my proposal about antimodularity hinges, as we will see in next section.

1.4 Antimodularity

Given all the above results about computational hardness of algorithmic detection of modularity, I propose to define the property of *antimodularity* in general, as the impossibility of obtaining, *by means of algorithmic modularity detection*, a useful, valid, hierarchical modular description of a system. More precisely, a system shows antimodularity when its most feasible and faithful hierarchical description, yielded by algorithmic means, is too approximate to be a useful high-level description of the system anyhow, or it is even completely invalid. In these cases, the only possible hierarchical description comprises only the two trivial hierarchical levels: the level of the whole system and the level of its single, lowest-level parts: in other words, *antimodular* systems

are systems which, intuitively, can be explained by decomposition at *one* level only, the level of their elementary, finer parts.

Antimodularity is due to failure of the application of algorithmic methods for modularity detection, and this in turn can be possibly blamed on two conditions:

1. No intermediate-level modularity can be reasonably supposed in the system, given its preferred description. That is, roughy stated, the system so described is actually *not* modular. I call this case *intrinsic antimodularity*, meaning that antimodularity is intrinsic to the given preferred description, no matter how accurate the algorithm for its detection is. This situation can occur when the system's parts, according to the preferred description, are hyperconnected: for example, in a regular network each node is connected to *all* the other ones, and so no modularity can ever show up.

2. Regardless of the fact that an actual modular structure is present in the system's preferred description or not (like in point 1), antimodularity arises because, given the *high number of parts* composing the system's preferred description, the modularity-detection algorithm ends up being computationally too expensive to be brought to completion, either because it is computationally hard[28], or, although formally not hard, because it is too computationally expensive to be brought to an end anyway. I call this last reason simply *antimodularity* (of course, intrinsic antimodularity is a case of antimodularity).

The reason behind this antimodularity/intrinsic antimodularity distinction is that, while antimodularity could in some case be eliminated by improving the modularity-detection algorithm, intrinsic antimodularity would still hold in any case, being not due to the inaccuracy or to the computational cost of the algorithm employed, but to an objective feature of a system, where, due to the uniform distribution across the system of the strength of the relationship which relates parts, modularity, relative to that chosen relationship between parts, is actually, objectively, absent.

It could be useful, once in the presence of antimodularity, to have a method to tell if it is intrinsic antimodularity, or if modularity is present, but it cannot be detected. Being due to the statistical distribution of the relationship between parts, intrinsic antimodularity, at least structural intrinsic antimodularity, should be reasonably easy to detect, because intrinsic absence of modularity can be revealed by statistical surveys of the distribution of certain properties across a systems. So, it should be quite easy to tell antimodularity and intrinsic antimodularity apart, at least in certain cases. There are, however, exceptions which will be discussed in section 13.2.

In the light of what we have seen till now, it appears that modularity detection can, in sufficiently large systems, be actually prevented by problems of computational cost, or even computational hardness, so a system can be *pragmatically* considered antimodular, even if in principle it possesses some modularity, which, however, we are practically unable to automatically detect. An antimodular preferred description of a system does not possess, at least as far as we can know, any valid high-level modular description, that is, a description whose parts are endowed with a sufficient degree of robustness.

The pragmatic aspect of antimodularity, anyway, should not be downplayed as merely pragmatic: it is a pragmatic impossibility to bring to completion in a feasible time a computer program, but, especially when the computational hardness of an algorithm has been mathematically proved,

[28] See former section.

this pragmatic hindrance becomes something more compelling, assuming the cogency of a logical law: there cannot be any hope of rendering the algorithm for optimization of detected modularity, which is proved to be computationally hard, more computationally feasible. It does not matter how we try to improve a computationally hard algorithm, or improve the power of the system on which it runs: its execution time will, at least in certain cases, always overcome any possible improvement in speed. Optimization of modularity detection can be probably *approximated* in more reasonable times, but the trade off between speed and accuracy, which is typical of approximated algorithms for modularity detection, associated with the high number of parts of some complex systems, could make the found modularity excessively approximate or, conversely, make the detection time of a sufficiently precise modular description excessively high, even if the approximated algorithm is not, from a formal point of view, computationally hard.

So, antimodularity, at least for what concerns the search for the best modular description, which has been proved to be an NP-complete task, is a pragmatic but at the same time an *objective*, unavoidable property of a system, deriving from computational properties which do not depend on contingent constraints or on a choice made by the observer.

1.4.1 Antimodular emergence

I propose to call the occurrence of antimodularity in a system a case of *antimodular emergence*, and to consider it a form of *computational emergence*. Antimodularity appears indeed quite similar to a well-known form of emergence: *weak emergence*, a notion proposed by Mark Bedau since the mid-'90s. This is a notion of diachronic emergence related to certain properties of computational systems. I specifically compare my proposed notion of antimodular emergence to that of weak emergence, concluding that antimodularity entails, with some qualification, weak emergence, but that the converse does not hold for all systems: there are *modular* systems which are, at the same time, weakly emergent. I outline here the basic line of reasoning about the relationship between antimodularity and weak emergence, a discussion which will be expanded in section 13.3.

Mark Bedau (1997) introduces the notion of *weak emergence* (*WE* henceforth), which, in its original formulation, applies mostly to discrete dynamical systems:

> Macrostate P of S with microdynamic D is weakly emergent iff P can be derived from D and S's external conditions but only by simulation.[29]

Without dwelling here in an explanation of the terms employed in the above definition, it suffices here to say that I think Bedau's definition could be safely rephrased as:

> A macrostate is weakly emergent iff it can be derived given the preferred, low-level description of the system and the initial state of the system, but *only* by *microsimulation*, that is, by simulating the system's dynamics step-by step according to its lowest level description (that is, its preferred description).

It appears that, under most conditions, which will be better specified later on here and in section 13.3, antimodularity entails Bedau's weak emergence. The argument is, briefly, this: if a system is antimodular, then by definition this means that its *only* valid modular description is its preferred, lowest level description. This implies that the system is not *predictable* by means of a high-level,

[29] Bedau (1997), p.378.

modular simulation: because, if it were, that would mean that the high-level simulation, in that it is capable of predicting the system, represents a high-level *valid* modular description. But, in an antimodular system, this high-level modular valid description of the system is excluded by the definition of antimodularity. So, it can be concluded that the dynamics of an antimodular system is not susceptible to be forecast by any modular high-level simulation: if no other *non-modular* prediction method is applicable, then the only way to know how the system's behavior will evolve is by simulating the system at the level of its preferred description, that is by microsimulation. This last circumstance appears equivalent to the above rephrasing I made of Bedau's weak emergence definition. So, it seems that *antimodularity* → *weak emergence*. This implication is not absolutely sure, however, for it depends on the circumstance that an antimodular system, which is not predictable by any high-level *modular* simulation, be impossible to predict by any other non-modular means, too. In section 13.3 I will show how certain antimodular systems could indeed be predicted by non-modular high-level means, and so result being not weakly emergent, but I will argue that these systems are probably not very interesting in their behavior, and that in most interesting complex systems, like those computationally capable, antimodularity entails weak emergence.

The interesting thing is that the opposite implication does not hold: there are weakly emergent systems which at the same time are not antimodular, systems which have valid high-level modular descriptions. The system remains weakly emergent even in presence of these modular descriptions, because such high-level descriptions cannot be used to predict the system (a prediction that, if possible, would render the system *not* weakly emergent, by definition), but can be used only to *explain* the system. This can happen for two possible reasons. First, because such modular descriptions are too vague, too abstract, too high-level to be used to compute a dynamical simulation of the system: for example, flow-charts which simply summarily describe the functional role modules fulfill in the system, without providing enough details to allow for their implementation. These modular high-level descriptions cannot simulate dynamically the system, so they cannot anticipate in any way its dynamical outcomes, but can be used to *explain* the system, conveying a good explanation. Or, in another case, the reason for the system to be weakly emergent despite having possible high-level modular redescriptions, is that the system, even if functionally modular at high level, is intrinsically unpredictable, and this is the reason for its being weakly emergent. This can happen in computationally universal systems, which, as a consequence of the undecidability of the halting problem (the well-known property proved by Alan Turing along with his proposal of computational systems in 1936[30]), possess many dynamical properties which are intrinsically unpredictable. Real-world universal computers are usually systems of this kind: they are highly modular, but potentially unpredictable. So, they are modular and at the same time weakly emergent.

1.4.2 Antimodularity and models of explanation

After having defined this new type of computational emergence, which is due to the amount of computational hardness which can be manifested in certain cases by modularity detection algorithms, I try to draw some possible consequences of antimodular emergence on the possibility of *scientifically explaining* systems affected by it. I examine two well-known models of scientific explanation: the functional-mechanistic, and the deductive-nomological (*DN*, henceforth). I then evaluate a more debated model of explanation, *computational explanation*, and another type of explanation which has been object of recent scrutiny, the mathematical-topological type of explanation, a form of explanation adequate to explain certain features of complex dynamical

[30] As explained in section 17.2.6 of the Appendix.

systems. I conclude that antimodular emergence affects the feasibility of the first two types of explanation, as well as computational explanation, albeit differently, and leaves unaffected the possibility of topological explanation, even constituting an occasion for this kind of explanation.

1.4.3 Antimodularity and functional or mechanistic explanations

I claim that antimodularity negatively affects *mechanistic* explanation, a fundamental form of explanation in biological sciences. A brief detour is in order here to describe what this form of explanation amounts to.

The term *mechanistic explanation* usually refers nowadays in philosophy to a relatively recent model of scientific explanation, put forth since the '90s by several groups of philosophers of biology and of cognitive science working rather independently, the most prominent exponents of the two main lines of inquiry in this field being William Bechtel and his collaborators on one hand, and Carl Craver and his colleagues on the other[31]. Leaving for the moment aside the subtle differences between these two main conceptions of mechanistic explanation[32], I base here on Bechtel & Abrahamsen (2005), which is a standard text in the topic. Bechtel and Abrahamsen (*BA* henceforth) give a definition of *mechanism* as:

> A *mechanism* is a structure performing a function[33] in virtue of its component parts, component operations, and their organization. The orchestrated functioning of the mechanism is responsible for one or more phenomena[34].

The definition above defines a mechanism as what I have along this chapter called a *complex system*, that is, a system composed of interacting parts. The point to stress here is that there is a functional view involved: the global function, which represents the explanandum, is explained by describing the organization and interactions of the parts which, by means of their dynamical "orchestrated" functioning, produce the phenomenon. What is needed, according to BA, to explain a given phenomenon is then to first identify the parts and operations involved in its production. To this aim, the system as a whole must be subject to two operations, which BA call *structural decomposition* and *functional decomposition*: the first yields the set of elementary parts of the system, while the second, which in real-world science is often conducted separately from the first, identifies component operations. A third, desirable operation is *localization*, which consists in linking parts with the operations they perform. This way, a mechanistic explanation is given, according to BA. This low-level kind of explanation is not always the most desirable, and, as BA highlight, it is important that a hierarchy of mechanisms be considered, and that explanation be *multilevel*. According to BA, a mechanism may also involve *multiple levels of organization*, being often part of a higher-level, larger part of a larger mechanism: circumstances external to a given mechanism can be seen as larger overarching mechanisms, while components of a mechanism can be seen as mechanisms themselves, recursively composed of subparts.

It seems to me this whole conception of mechanisms could be easily rephrased in terms of modularity, along the lines of the view which I have sketched till now. The result of functional,

[31] The two corresponding seminal works are Bechtel & Richardson (1993) and Machamer, Darden, & Craver (2000).

[32] These differences, especially the most significant, between the so-called *epistemic* view, which I, along with William Bechtel, endorse, and the *ontic* view of mechanisms, supported by Carl Craver, will be discussed in chapter 10.

[33] See section 9.

[34] Bechtel & Abrahamsen (2005), p. 423.

structural decomposition and localization is what I have called the *preferred description* of the system: the identification of the basic, lowest level parts which the observer has chosen to identify. BA do not stress, as I do, the dependence of this description on a choice on the part of the observer, because they consider implicitly that there are natural preferred descriptions of some systems, and there undoubtedly are, for example in molecular biology, where the molecules (or, possibly, atoms) are the most natural elementary parts. The main difference with my view is then that my conception of hierarchical modularity is more general, comprehensive of non-physical, functional-only, forms of modularity, like that of computations.

That said, along with the definition of antimodularity, it is easy to show how *antimodularity compels to single-level-only explanations*, neglecting the need, essential for mechanistic explanations, of *multi-level* integration. Antimodularity would limit mechanistic explanation to the level of description representing the most elementary parts of the system, which is the most numerous and the most complicated level, and this fact certainly hinders comprehension as well: for large enough systems, their mechanistic explanation at this level is too complex to be understood by human beings, and understandability is a quality to be sought for in mechanistic explanation, according to some accounts, notably the ones by William bechtel and his collaborators. Others, too, deem intelligibility an essential feature of explanations, for example Petri Ylikoski, which considers "cognitive salience" an important feature of explanations.

It is evident that a mechanistic explanation tries to answer to "how" questions ("how a phenomenon is brought about?"), by showing the way the complex dynamical functioning of a set of interacting parts produces the phenomenon. The same question can be answered to, also just from the *functional* point of view, and this conception, mainly aimed at characterizing explanation in cognitive psychology, has been notoriously advanced by Robert Cummins. In a way similar to that of mechanistic decomposition, functional analysis begins with a characterization of the global phenomenon (the *disposition*; I will dedicate a discussion to this term of art in chapter 9) taken as the overall function to be explained in terms of its component subfunctions. This is a typical form of so-called *role functionalism*, in that the concept of function[35] is considered that of a partial role fulfilled by a subsystem in order to bring about the whole functioning of the overarching system. Seen from an explanatory point of view, the function of a subsystem is employed in explaining how the overall function, which is the explanandum, is performed by means of the organized contributions of its subfunctions, which execute their function in a programmed activity. This position is quite close to a computational view, and it is completely compatible with it. Actually, Cummins' functional analysis is the prototype of the typical explanation of cognitive psychology, which mostly consists of functional explanations, often in the form of *computational explanation*, that is, the exhibition of a computer program able to produce the cognitive phenomenon to be explained.

A more thorough characterization of Cummins' position is given in chapter 9, where the relationship between purely functional and mechanistic explanation is also better analyzed. What I would like to highlight here is that Cummins himself, since his earlier works, as in the seminal Cummins (1975), stresses that recursive functional decomposition until a full hierarchy is obtained is the strategy to seek for in scientific explanations, especially in biological ones. Antimodularity would completely hinder this goal, allowing for a two-level only explanation: the highest one, that of the explanandum itself, and, on the other end of the scale, the lowest level, that of the most elementary functions.

[35] The notion of function is to be examined in chapter 9.

1.4.4 Antimodularity and the deductive-nomological model

In the classic deductive-nomological (*DN*) view of explanation, stemming from the seminal work of Carl G. Hempel and Paul Oppenheim[36], explanation is seen as a *logical deduction* of the explanandum from the explanans, and what counts is validity and soundness of the deduction, with scarce attention directed to the *intelligibility* of the explanation: such a concern about understandability of the explanation would have been considered, in the historical post-neopositivistic milieu of the time, an inappropriate trespassing of philosophy of science into the territory of pragmatical, or worse, *psychological* aspects of scientific explanation. In such a view, all that matters for an explanation is that it is a correct deduction. Explanation is seen in this model as depending on the possibility of *prediction* of the phenomenon by means of a scientific *law*. The explanation itself amounts to the description of the logical derivation of the explanandum from a group of premises constituted by a scientific law and a set of clauses representing initial conditions of the system to explain.

For what concerns deductive-nomological explanations, I show that, antimodularity entailing weak emergence in complex enough systems, DN explanation for a complex antimodular system could not be recurred to, because, if it could, that would mean that the system is predictable, and this is negated by the definition of weak emergence itself. To clarify: it is excluded by the definition of a weakly emergent phenomenon that it can be predicted by means of a law which, given the initial state, determines in which state the system is going to be at any given time, and that this law has a mathematical expression which can be *analytically* solved. As said, this is excluded by the very definition of weak emergence, which basically states that a weakly emergent phenomenon (in a discrete dynamical system) is one that cannot be predicted, and that it can be reached only by performing the step-by-step microsimulation at the system's lowest level. Given that my notion of antimodularity, under the circumstances expressed in section 1.4.1, entails weak emergence, it turns out that a complex enough antimodular system cannot be predicted by an analytically solvable expression. So, no DN explanation of a complex antimodular system could be based on such an analytically solvable law.

Anyway, if we take into consideration a specific class of systems, namely cellular automata (*CAs*), then a weakly emergent process generated by a CA can in a way be explained by producing a possibly very long list of steps of its evolution, a list which can be seen as a list of *deductive* steps inside a formal logical system, in which the premises are constituted by the CA's initial configuration and the CA-rule, which gets repeatedly applied first to the initial configuration and then to the intermediate configuration obtained at each deductive step. Given that every CA-rule is, by the Curtis–Hedlund–Lyndon theorem[37], local and equally valid in any point of the CA's lattice, the form of a CA-rule can in this regard be assimilated to the form of a physical *law*, which, as a law, holds universally. Accordingly, by this analogy, the production of this list of consecutive states of the CA could in a way be assimilated to a long DN explanation, which must consist of a logical deduction of the explanandum starting from given initial conditions and a law. Even in this case, human comprehension would be hindered by the potential length of the list, but, according to the theoretical position of post-neopositivistic advocates of the DN model of explanation, understanding is an inessential feature of explanations, and it is not required for a good DN explanation. So, in a way, weak emergence and, consequently, antimodularity, does not hinder DN explanation, at least in the case of CAs and other systems whose dynamics follow a *universal* rule. From this considerations are excluded classes different from CAs, for example boolean networks in general, whose dynamics can follow locally changing rules which

[36] Hempel & Oppenheim (1948).
[37] See section 14.2.1.

are not universal. In these cases, the rule to be employed would be the global update rule, which, being nonlocal, is usually much more complex than a CA-rule, and, as a consequence, the list of deductions constituting the DN-style explanation of such systems would be even more unintelligible.

1.4.5 Antimodularity and topological explanations

I now consider consequences of antimodularity on the possibility of explaining a complex system by means of what Philippe Huneman calls *topological explanation*. Huneman describes topological explanation as "a kind of explanation that abstracts away from causal relations and interactions in a system, in order to pick up some sort of 'topological' properties of that system and draw from those properties mathematical consequences that explain the features of the system they target."[38]. Inspired from mathematical topology, the topological properties of a system are the properties concerning in a way its "shape" which are invariant under possible continuous deformations of the system. These structural properties must not belong to a material system, but can be parts of an abstract, mathematical space. In my terminology, I would say that these topological properties do not concern a system, but a description of the system. Now, topological explanation consists in explaining features of the system by appealing not to causal events between its parts, like mechanical explanation would do, but by pointing to some topological features of the system's representation in this abstract space. This kind of explanation is not mechanistic, because it does not specify the particular interactions between parts which give rise to the dynamical phenomenon: the explanation is specifically based on a *mathematical*, topological feature, which does all the explanatory work.

Topological explanation could also be based on the presence of *modular* structure: This can happen for example when a topological explanation of the robustness of a network's dynamics to local perturbations is given by mentioning that the network has a *community structure*[39]: this modular structure ensures that perturbations remain local or channeled, without spreading indiscriminately at the same speed on the whole network. On the contrary, intrinsic antimodularity could produce unbounded spread of perturbations across the network.

All in all, not only it seems that antimodular emergence does not hamper topological explanation, but it actually turns out that intrinsic antimodularity or its absence could indeed *allow* for certain topological explanations.

1.4.6 Explanation and prediction

The possibility, highlighted above in section 1.4.1, that there are systems which are functionally explainable and, at the same time, unpredictable, an example of which is the class of computationally universal systems, gives a quite remarkable indication, namely that prediction and explanation are disjoint endeavors: unpredictability does not render, per se, a system unexplainable. This is a curious result, because it proofs, in a way, that *prediction* is not necessary for explanation, and thus that the deductive-nomological model of explanation, even if it were free from other downsides, could not be the all-embracing model of explanation for science in general. In science certain explanations are acceptable even if they are not based on prediction: explanations which are functional, or mechanistic.

[38] Huneman (2010), p. 214.
[39] See section 3.2.1.

1.4.7 Computation and computational explanation

Before assessing, in the next section, the possible consequences of antimodularity on computational explanations, it is in order some reflection on what a *computational explanation* is. Specifically, we will be asking if and when a given system, most often a dynamical system, computes or not, in order to see if it can be given a computational explanation. To this goal, it seems inevitable to discuss the notion of *computation* itself.

Since the epochal work of Turing of 1936, which founded computer science, it can seem at a first glance that the concept of computation is absolutely clear: the work of Turing has indeed provided a touchstone against which to characterize what computation is. This is certainly correct: computation has been treated, since its inception with Turing, as an eminently *formal* question, deserving a *mathematical* approach. And this has surely been, I think, the *right* way to treat the question, for it has brought to light the essential features of computation: the properties of computability, its power and limits, which have been detailed in a thorough and rigorous way that no other approach to the problem of computing could have developed. As always, a mathematical approach has revealed itself as the most powerful way to highlight all the facets of an abstract phenomenon and to draw extremely important consequences and new ideas and models from this analysis: a look at the Appendix[40] will suffice to be convinced of that.

So, does a given system compute? In the wake of what stated above, this seems an easy question, decidable on formal grounds, but actually, from another point of view, it is not. To see this difficulty we could resort to imagining someone who launches a program on her computer, and that to our question "What does this program do?" answered that it *computes*, simpliciter. Would that answer make sense to us? Let's suppose, again, that some programmer wrote for fun a program by writing down a random list of instructions, and that, for a stroke of luck, the program, instead of crashing, went on and on, spitting out apparently random strings on the terminal. Would you count this program as computing? Yes, of course, in a manner of speaking, it computes... but computes *what*?!? Third case: we have a program running on a very barebone computer, which gives its outputs directly as binary strings: given the string: 00110101 it spits out 1111, given 01000011 it outputs 1100. Does it compute?

It seems to me that the above examples immediately suggest that recognition of the occurrence of a computation needs a sort of intentional attribution: it is the attribution of *computing*, to what otherwise could be seen as a mere physical transformation of tokens: to say that a given process computes, we need to *specify what* it computes. In order to do this, a condition must be fulfilled: a *mapping* must be established between the physical configurations of the machine, which the alleged computation acts upon, and meaningful signs. Only once we do this, are we in the position to at least try to guess *which* specific computation the system is performing.

The above mentioned mapping is an *interpretation*, which maps input and output configurations to meaningful signs of our language. If we apply this condition to the third example above and interpret the symbols of the apparently meaningless strings, we recognize that the system is probably performing multiplications: it suffices to substitute "3" for "0011", "5" for "0101", "15" for "1111" and "4" for "0100", "3" for "0011", "12" for "1100": this is, by the way, an obvious, base-2 numbering interpretation, but without an interpretation in terms of it, the computation the system performs would not have probably made much sense.

The interpretation is a *mapping*, which is itself an *algorithmic* operation, which, given certain configurations on input, produces other symbols. This is not something which concerns the prob-

[40] Section 17.

lem of intentionality understood as a philosophical problem. We operate intentional *attribution* on the set of configurations *resulting* from this mapping, which must have been chosen, in their formal properties, as able to *signify* something to us. Once the mapping is established, we can operate a more global intentional attribution, and say *which* computation the system on the whole is performing. In my view, only then the system can be seen as *computational*. Computation is *attribution* of computation, to systems which *per se*, are simply rule-governed discrete dynamical systems[41].

This "intentional", or "semantic" view of computation ("intentional" or "semantic" with all the disclaimers stated above) is not a new position, having been embraced by several authors, notably Jerry Fodor. Such a semantic view is opposed by some others, like Gualtiero Piccinini, who view computation as definable on purely mechanistic terms, without the need of recurring to any semantic attribution[42].

A problem arises here: given that the attribution of computation depends on the choice of a mapping, does this mean that *any* machine, provided a suitable mapping between meaningful symbols and its states, can be seen as performing *computations*?

Well, some conditions must be fulfilled: first,the machine must be digital: we are talking of *digital* computation here (and this is the dominant paradigm of computation at least since the '50s). So, to be considered computational, a machine must at least be considered digital, that is it must possess, and operate on, a finite set of possible stable distinct states. It is these states that the mapping will connect to symbols. But, in order for this to be possible, these states must be robust and distinguishable, and a deterministic rule must govern the transition between stable configurations of these elements. Not every system can fulfill such conditions, which are not trivial (think of distinguishing stable configurations in a turbulent fluid). But, inside the view I proposed here of modularity, it is easy to see that all that is required here is for the system to be a discrete dynamical system (or *DDS*), that is, a form of functionally modular system (see section 1.1.6).

The central point is this: to allow us to attribute to a system the execution of a computation, the following conditions must be satisfied:

a. an algorithmic mapping between linguistic symbols and possible input and output configurations of the system must be realized;

b. we must be able to say *which* is the particular function relating input to output configurations, that is which is the *specification* of the computation, in order to say what the system is computing.

Only in case the system is a DDS and this two more conditions are satisfied, the system can be seen as *computing*. Only then we will be in the position of saying that a system is performing a computation.

The order of conditions *a* and *b* above is inverted with respect to how a human programmer operates. In this case what is needed is not an interpretation, but an the establishing of a norm: in the case of programming, first the specification (point *b*) is arbitrarily *chosen*, it is

[41] Of course it can be raised a problem here: if the system is considered as already rule-governed, that means that an original intentional attribution has already been done. It is outside the scope of this work to tackle here this and other similar thorny questions, analogous to the infamous "kripkensteinian" rule-following problem.

[42] See for example Piccinini (2008).

considered the norm according to which the program will have to operate, and *on the basis of the specification*, the programmer chooses the *mapping* (point *a*) from symbols to input and output configurations which he deems best, in order to proceed to the *implementation* of the program, that is, the specification of the parts and the structure of the mechanism (the program) which will, at the end, be able to realize the chosen specification according to the chosen mapping. Thus, the choice of the mapping determines the choice of the *specific* structure of the *program*. All this series of operations constitutes the *implementation* of the chosen specification.

Speaking of *implementation* along the lines of Galton (1993), and Partridge & Galton (1995), I consider the relation specification-implementation a very universal one: an *implementation* is the specifying of a method to "realize"[43] a given overall specification. When considering a program, there is not, however, a *unique* overarching specification and a single level of implementation, for the two notions are *relative*, exactly like those of "higher" and "lower" description level, and that of function, which[44] is the partial role something fulfills relative to the scope of a global function. *Relative* in this case means that something which is the implementation of a specification, can in turn be considered a lower-level specification to be implemented at an even lower level. In other words, given a specification there is the need to find one possible implementation of it, and in the style of structured or modular programming, such an implementation will be decomposable itself into modules. Each module, being a specific input-output function[45] constitutes itself, in turn, a specification, which will be implemented at a lower level, and so on.

It seems sensible to think that the same multi-level abstract *hierarchy* in which each macro-component is multiply realizable by sub-components, and so on, underlies the notions of structured programming, functional decomposition, and hierarchical modular levels of descriptions.

As said, a programmer starts with a specification and tries to implement it. But it is possible to start with an initially *uninterpreted* discrete process, aiming to *discover* what computation, if any, and how, the process performs. This is a bottom-up path, and is typical of *reverse engineering* a computation. The top-down path of starting with a specification of a computation and trying to recursively decompose the specification, which is a function, into subfunctions, and these in turn into simpler subfunctions, and so on, in order to say *how* the specification is brought about, is instead the path of the *computational explanation*, which is the functional explanation typical of cognitive psychology, where the specification, which is the cognitive faculty (the equivalence cognition/computation is the basic tenet of cognitive psychology), is explained in terms of a hierarchical functional representation.

1.4.8 Antimodularity, cellular automata and computational explanations

Now we can turn our attention to the problem of the consequences of antimodularity on *computational explanation*. As said, this is the typical kind of explanation employed in the cognitive sciences. However, I will not try here to evaluate what impact could antimodularity have on these sciences, but only how it could impact a "toy model": I take as a model of computationally capable system a cellular automaton (a *CA*), in order to see how its dynamical behavior could be explained as something which performs a computation. The idea is that, if the CA is antimodular in its dynamical behavior, the task could be hampered, or even made impossible. If this is the case, this means that antimodularity could have impact on the possibility of computational explanation.

[43] In a sense closely akin to that of the property of *realization* in philosophy of mind. I will not scrutinize the notion here.

[44] Recall section 1.4.3.

[45] Function in a mathematical sense, see. section 9.

Let's thus consider the case of trying to computationally explain a CA. Two basic questions can be asked here:

- is a CA a computational system?
- if it is, how can we explain the computation it performs?

Regarding the first point, of course a CA, being a discrete dynamical system, respects the conditions stated in the preceding section of this chapter, and can be safely considered a computationally capable system. But to say that it *computes*, we must be able to say, in some way, *what* it computes: no computational explanation can be applied to it before a mapping is established between its configurations and symbols meaningful to us. We could try to map the state of its cell to some sensible symbol, for example we could map those states to "black" and "white", but this would lead to "computational" explanations of this sort: "according to the repeated application of its rule, the CA produces a progressive variation of the state of its cells, which can, under various conditions, change from white to black". That does not seem a very useful explanation. A complete explanation requires a perspicuous specification, that is, the ability to say what the system computes. Here, the specification is too vague: "progressive variation of the state of its cells under various conditions". The exact specification of the CA, seen as an input/output function, is given by the repeated application of its rule, so a sensible specification could be given in terms of the rule of the CA. Now, there are some problems arising here: usually, most CA rules are described in terms of a"lookup table", which is an extensional listing of how the rule determines the value of a cell at the next time step based on the values of the neighbor cells. Such a table becomes uncontrollably large as the neighborhood of a CA expands. So, for some CAs, the exhibition of this table would be impossible, or anyway would render a computational explanation containing it completely meaningless. In two-valued CAs, their rule can be seen as specifying a boolean expression. So, we could think of simplifying its description into the form of a boolean expression. This, however, is quite certainly a computationally hard task, so, it cannot be guaranteed to succeed in all cases.

All the above suggests us that we should try to find *higher-level* explanations, possibly *multilevel* computational explanations, in order to have a more useful explanation. In other words, we must devise a way to recognize a CA as a computationally-capable machine at a level which is higher than that of its elementary cells. To obtain that, we could try to see if the CA's dynamics is able to produce certain types of dynamical modules, that is, persistent enough high-level structures, whose behavior at the higher level can be seen as rule-governed, in order to fulfill the condition of seeing the CA dynamics at this higher level as another discrete dynamical system, a higher-level one, which can be seen as different from the DDS constituted by the CA and its rule. In other terms, in order to obtain a useful computational explanation of a CA, a first condition is (*i*) that a form of dynamical high-level *modularity* can be reliably detected in the global dynamics of the CA. Another condition must hold: (*ii*) the high-level modular dynamics must successfully *track* the low-level dynamics of the CA, without diverging from it. This condition of *validity* (to use the terminology of scientific computer modeling) is a quite complex one and is better specified in sections 2.2.1 and 6.6.8 of this work, but it basically amounts to this: that the dynamics of the high-level description must not diverge in time from the correspondent dynamics of the CA at its lowest level of description, that of cells.

It turns out that certain CAs are actually endowed with such a form of higher-level robust modularity: as we have seen, some CAs can generate *gliders* (see fig. 1.2), which end up realizing,

in many cases, predictable interactions one with the other, as in the case of Rule 54[46], and these predictable interactions can be seen as the high-level implementations of boolean functions, with gliders acting as traveling "bits". This interpretation in terms of boolean functions could then allow to build up progressively a multi-level explanation in terms of sensible computations, much in the way in which computer programs can be described by progressively higher-level programming languages. This way we would have built a part of the conditions required to explain by means of a *computational* explanation a CA.

However, this interpretation in terms of gliders is not always possible: there are certain "chaotic" CAs, like rule 30 (see fig. 1.7), which *never* show subconfigurations robust enough to be considered dynamical modules able to render the high-level representation computationally capable[47].

A point must be highlighted here: this impossibility to individuate stable dynamical modules in a CA, like in the case above or Rule 30, can be seen as a form of *intrinsic antimodularity* of the high-level description of the CA. So we can say that antimodularity, in this form, already prevents the first step, step (i) above, required to provide a computational explanation, a step which consists in viewing the CA as computationally capable at a high level. So, it seems that, at least in this form, *intrinsic antimodularity actually prevents computational explanation.*

However, it is sure that, for certain CAs, their high-level interpretation as computing systems is possible: there is a complex mapping, devised by Matthew Cook[48], with which he has been able to prove that rule 110, another elementary CA, can be seen as a computational system on the level of the universal Turing machine. Also the most famous CA, John Conway's *Game of Life*, has been proved to be Turing complete[49]. So, it is a proved fact that, under certain interpretations, some CAs can be seen as computing: this first condition can be seen as established, at least for certain CAs.

But, if we want to give a *computational explanation* of a system, another condition must be satisfied: that the system is actually *computing*, and not just that is computationally *capable*. And, to this aim, we should first be able to say *what* it is computing: that is, we must be able to express its input/output relationship, its *specification*. We must note that we are working here in the *reverse-engineering* field: we have a machine, the CA, which we know that is computationally capable, and we should, in order to computationally explain it, *produce* its specification.

Now, the task of reverse engineering program specifications, apparently is tough. Basically, it is a matter of producing all the inputs of a program and to observe all the correspondent outputs: it requires a form of *induction*. Leaving the finer details to another occasion, I simply note here that there is a host of problems tied to the fact that Turing machine-level computationally capable systems are affected by the halting problem, and this renders the above task very difficult if not impossible: due to the fact that the number of possible input/output couples to be observed grows exponentially with the maximum size of the input, it is at least a computationally hard task (see sections 14.5.2 and 17.2.6 for details). There are approximate algorithms for this purpose of inferring specifications (algorithms for "specification mining", see section 4.3.1.1) which however give often too approximated result and cannot reverse-engineer specifications of Turing-machine level computations.

[46] See for instance Martínez, Adamatzky, & McIntosh (2014).

[47] Why can't we contrive a mapping from sets of chaotic configurations to meaningful symbols, this way rendering even a chaotic CA computationally capable at high-level? An answer implies a discussion on the complexity of the mapping between system configurations and symbols, a discussion which is developed in section 14.5.2.

[48] See Cook (2004).

[49] See Rendell (2002).

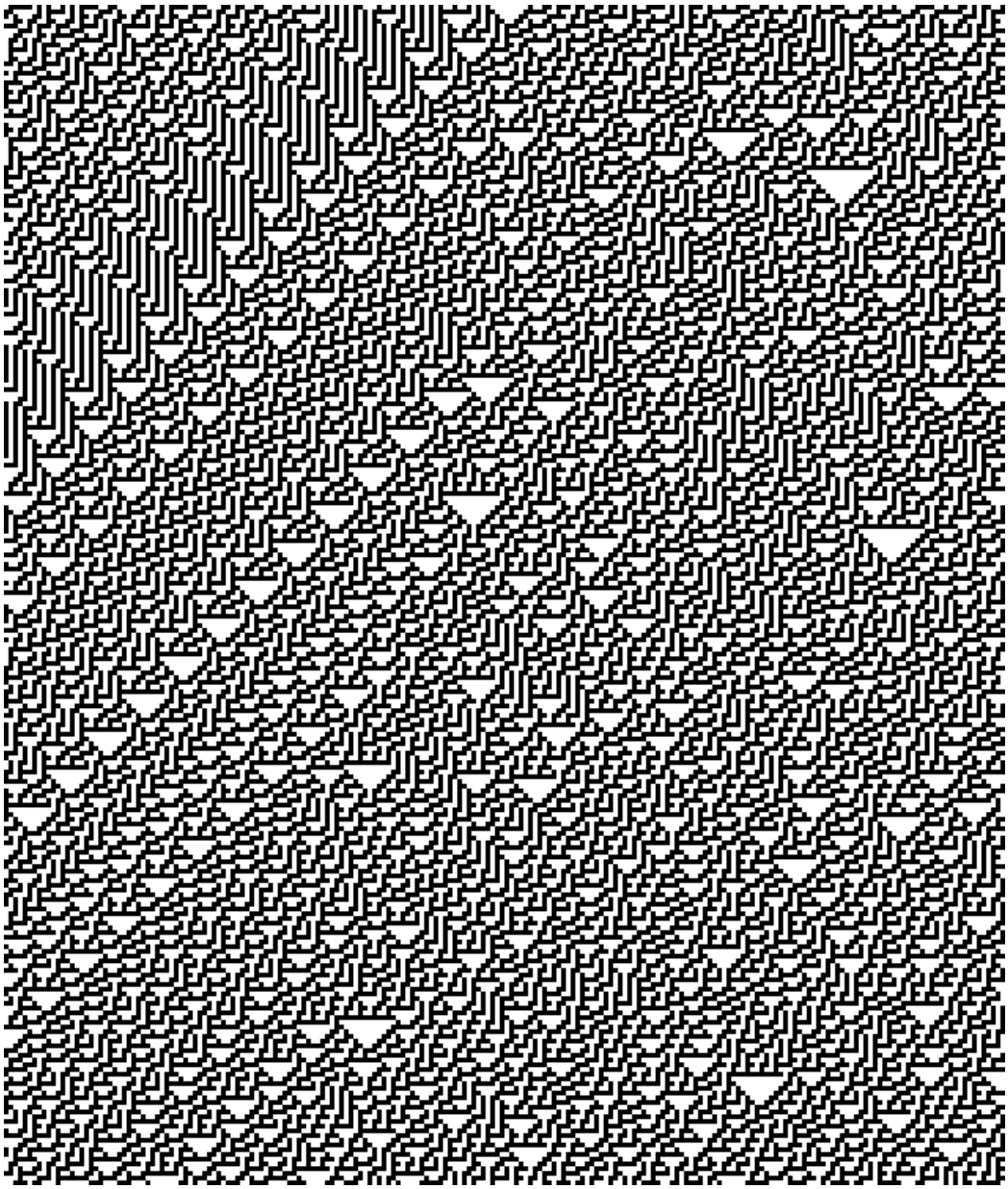

Figure 1.7: a chaotic segment of evolution of the elementary CA Rule 30. Time flows from top to bottom, each row of pixels representing the global configuration of the system at each timestep. Each pixel represents the state of one of the elementary parts of the CA, its *cells*.

But we needed the program's specification in order to computationally explain it. And this specification is very hard to infer.

However, a specification in terms of the mere input/output function is not the only way a specification can be given, and, even if it is the most precise, it is neither the most perspicuous way to begin with, because a list of input/output couples can be meaningless. So, another, more perspicuous form in which a specification can be given, is in an *aggregate* form: a more or less synthetic way to sum up the whole input/output function. One way to do that is giving the specification in terms of its decomposition in subfunctions, which is a form of hierarchical decomposition.

Now, this is the interesting point: if a *hierarchical*, *modular* representation of the computation could be devised by some means, it would be possible to test each module *separately* to seek for the specification of *only* that module, a task which would most likely turn out being more feasible *by order of magnitudes* than that of submitting every possible input to the whole program in order to directly infer the global specification, because a module is identifiable by the very fact that it should be only loosely or sparsely connected to the other modules, and this translates in a probable reduction in the number of possible inputs to the module, and a consequent easier exploration of that module's input space[50].

The fact that it has been possible to find the single specification of each module due to the system's decomposability, hopefully allows, if the specification of each module is not too complicated, for a form of *aggregation*, as discussed in section 4.1.5.1: if we are able to further abstract the module specification by "naming" it in a meaningful succinct way, giving the module a name which is representative and explanatory of the function it performs (as for example when we say that a module performs the "multiplication" operation), then each module's specification can be substituted by this more concise definition of *what* function the module performs. Then a global specification of the whole system can be given *in terms* of a description (usually in the graphical form of a *flow chart*) of the modular structure of the system as a directed network of connected modules, where modules are seen as nodes labeled with their succinct "names" representing their specification, and their input and output connections are the directed links between nodes.

So, this type of explanation seems possible, after all. But *it requires that a functional modularity of the computation can be found* , and this, in turns requires two conditions: first, that the system is actually computationally capable at a high level, and this is not guaranteed: *intrinsically antimodular* systems, like CA rule 30, hinted at above, are not even susceptible to be seen as computationally capable at high level. Second, another condition is that, even if the system is computationally capable, and actually possesses dynamical modularity, this modularity *can be actually found*. This could be hindered by some factors anyway, due to the high computational cost of the algorithms for modularity detection, or the excessive approximation of the results they obtain, when they can be obtained in a reasonable time: the obtained hierarchical representations could be not faithful enough to the actual functional modular organization of the system to be considered *valid* modular descriptions, able to sufficiently well characterize the computation performed.

It seems, thus, that *antimodularity can hinder or prevent also the computational kind of explanation.*

But, could *partial* reconstructions of the program's functional hierarchy still be used in explanations? Well, it seems, intuitively that the functional models so produced would be very constrained by *ceteris paribus* clauses, in order to keep them inside the range of *known* input/output

[50] Even if this is not *guaranteed*. See section 4.3.1 for a better discussion.

relations, and, among the known input/output relations, in the range of those which do not diverge too much, for lack of validity of the modular model, from the actual observed behavior of the system. So, it seems that an explanation based on them should also be so constrained in its applicability. It could possibly appear as an acceptable explanation, but it would in a way be only a *post-hoc* explanation of the range of behavior of the system actually observed during the process of modularity detection, and not of all possible behavior.

It may well be that, in computational cognitive science, such a kind of limited explanation could be accepted, and, moreover, it is likely that there are *only* explanations of this kind in some subfields of cognitive psychology. In that science, the task of finding the specification and the functional relations between modules, is left to human experimentation, and that is most probably a slower process than automated algorithmic ones.

To conclude this section, I think that also this failure in finding computational explanations of certain systems can be considered a form of *antimodular emergence*. This computational emergence regarding computational explanations can be seen as due to two forms of antimodularity: first, *intrinsic* antimodularity affecting chaotic systems, which can not even be considered computationally *capable* at a higher level of description, and second, antimodularity due to the excessive computational cost or excessive approximation of modularity detection algorithms.

The consequence of this form of antimodular emergence is that the system affected by it is only explainable at its lowest level of description, and this would in general not constitute an intelligible form of explanation, in sufficiently complicated systems. It is to be expected that antimodular emergence could affect also computational explanation in cognitive and neurosciences, and this is worth, it seems to me, of further investigation.

1.4.9 High-level modularity as a condition for programming and scientific research

At least for a computer programmer, the claim by Robert Cummins, which we will encounter in section 9.2, that functional analysis has an explanatory capacity, is nothing new: a programmer, at least implicitly, is continually developing partial *explanations* of how the program under construction works by means of the organized execution of its instructions, and, at a higher level, of its subroutines or even higher modules. The very act itself of programming starts from the *specification* of the whole program (e.g, being "a word processor"), and development proceeds by analyzing, in a Cummins-like sense, this global function, which is the specification, into smaller subfunctions which together make up the implementation. In turn, each subfunction gets decomposed, if possible, in simpler subfunctions, and so on. The actual writing of the program, the act of writing the sequences of instructions composing each subroutine, usually bouncing up and down across hierarchical levels, analyzing global functions into smaller subroutines, implementing them and going back to decompose other high-level functions, or of composing them starting from the simplest subroutines and going up in the hierarchical levels, is almost unfeasible without a former, at least implicit, *explanation*, on the part of the programmer herself, of the whole system in hierarchical terms. This can be seen as a form of *functional explanation* the way Cummins understands it. It seems, then, that functional hierarchical explanation is necessary for *computer programming*, in computer science.

In empirical science, functional, multilevel explanation is probably essential not only *after* a theory or a model of a phenomenon has been devised, that is when explaining an already known phenomenon, but also *in the making* of a theory. *Interventionistic* accounts of causation like

that of James Woodward[51] see a causal relationship as holding between two entities when an hypothetical variation of the state of one entity, purposefully induced by an experimenter, that is, an *intervention*, systematically produces a variation in the state of the other entity: when this hypothetical circumstance holds, we can say that the two entities are causally related. A mechanistic account of explanation requires during scientific research the advancement and refining of the mechanical description of a mechanism by progressively discovering all the causal relations holding between its parts. To do this, the experimenter proceeds by intervening on each part separately, and seeing if some consequent variation occurs on other parts. But, to properly identify causal links, intervention on the state of a part requires a temporary, at least virtual, disruption of the structure of causal links going from other parts of the mechanism towards the part on which we are intervening: interventions on mechanisms require that the mechanism is temporarily *modified* by eliminating some of the connections between its parts. Woodward claims that the set of equations representing correctly a causal system must be *modular*, because otherwise, given that detection of causal relationships requires intervention on a part of the system by temporarily disrupting only the causal influence which bears on that certain part, were the system completely not modular, this precise disconnection of one causal path would disrupt not only the part of the equation interested by the intervention, but also other parts of the system. So, dynamical modularity is always present in a mechanism, at least at the lowest level, that of the preferred description.

But, the point is, if we want to redescribe a system mechanistically at higher-levels, we could certainly construe relations between high-level parts as prima facie "high level" causal relations. In that case, in order to proceed by intervention Woodward-style, modularity is needed also in the equations representing the system's dynamics at these *higher* levels. All considered, this condition holds if the update function's[52] structure is *hierarchically* modular, and this in turn represents the fact that the system is functionally, and probably also, dynamically, and, quite likely, structurally, *hierarchically modular*.

The same utility of the presence and of the possible detection of modular structure shows up in the phase of discovery of complex networks, especially in cases in which the discovery of links between nodes requires a complex experimental work, like in gene regulatory networks and other networks of biological interest. Certain recently proposed algorithmic methods, like that in Clauset, Moore, & Newman (2008)[53], could be of great aid in this kind of task because, based on detected hierarchical modularity in the already discovered partial network, they can probabilistically produce, with a good chance of success, a prediction about where in the network missing links should probably show up with further observation, thereby fruitfully guiding subsequent experimentation.

Thus, it seems that hierarchical modularity is important or even essential in the phase of scientific research and experimental *discovery*, besides being almost essential, as we have seen in the former sections, for explanation of an already studied phenomenon.

1.4.10 Explanatory emergence

Given that the lack of understanding due to the presence of antimodular emergence in a system can seemingly affect most kind of explanations, I propose to generalize the notion with the following definition:

[51] See Woodward (2003), and section 6.9.
[52] The function governing the dynamics of a DDS: see section 5.1.
[53] Described in section 6.9.

> *explanatory emergence* is a property of systems or descriptions of them that consists in the fact that, for absolute or pragmatic computational reasons, they resist *understandable* explanations.

This is a more general definition than that of antimodularity, a definition comprising other possible effects of computational constraints on the explanation of complex systems. In section 14.6, I will explain how I understand the above definition, which does not apply necessarily to computational tasks, but also to human operated tasks which bear on explanations, and the possible usefulness of this definition in the current scientific landscape.

1.4.11 Is it likely to encounter antimodular systems in science?

Antimodularity appears to depend on the *choice* of a relation, which is specified in the preferred description, between the elementary parts of the system. Antimodularity can occur when, given this chosen relation, modularity detection according to it turns out to be too computational demanding to be brought to completion in a feasible time, or when, although modularity detection is successfully completed by means of an approximate algorithm, the produced modular description appears too approximate to be capable of validly representing the original system.

What is the likelihood that either of these two circumstances can be encountered during scientific research? It must be stressed that computational complexity of modularity detection concerns algorithms for detection of modularity which do not employ any other informations about the systems than those included in their preferred description, that is, the level of their elementary parts and their relations. By adding constraints on how the elementary parts can be grouped into modules, the task can be highly simplified. This is equivalent to devising ad hoc algorithms for modularity detection, and ad hoc algorithms could end up being less computationally demanding than a general one. Actually, in many cases, this seems exactly what science does: it searches for empirical constraints to help us choose among the possible theories of the world. This raises the chance that scientific method can produce modular, intelligible descriptions of phenomena.

But, we must ask if new developments in science can shift the focus on systems of such a complexity that even the known, empirically found constraints about them could end up being too few to allow the successful completion of modularity detection on such systems.

In the case of biological systems, we can be actually be reasonably sure that they are modular, at least at certain levels. There are many arguments, empirical and theoretical, which favor this conclusion, which are to be treated in section 7.1. Nevertheless, there can be significant biological systems, like for example interaction networks in the cell metabolism, which can end up be so huge to possibly produce effects of explanatory or antimodular emergence due to the high computational cost of the algorithm for data mining or modularity detection in relation to the size of the system. Another type of situation in which this antimodular emergence can be expected to show up, is in data mining for modular information in the already existent *literature* on certain scientific topics, an example of which is presented in section 1.5.2.

1.5 Some additional reflections on modularity, metaphysics, computing, history of science

In the above sections, I have outlined the main structure of this work, in which I try to reflect on the notion of modularity, its relation with description, explanation, computation, understanding, and to outline the conditions under which modularity manifests itself or, on the contrary, cannot

be detected. My general aim is to describe the importance of what we could call a "modular way of thinking" for human knowledge of the world, and specifically the great importance of modularity on much of scientific conceptualization, especially in the so-called "special sciences", which mostly recur to types of explanation, *mechanistic* or *functional* explanation, which are the modular kinds of explanation *par excellence*: to explain in these two ways, it is necessary to find a way to describe a system as modular.

I then focused my attention specifically on ways to detect modularity in large complex systems, stressing the fact that, unfortunately for science, these ways are algorithmically complex and for this reason they are not guaranteed to give useful results. When a failure in giving modular descriptions manifests, this is what I call a case of *antimodular emergence*. This fact hinders, and possibly prevents in some cases, the possibility of scientific explanation and comprehension of complex systems large and complicated enough to escape a proper modular explanation, systems which, due to their sheer size, cannot be grasped cognitively in their non-modular, low level description, which is their only remaining possible description. I called this condition "explanatory emergence". This is the main goal of this work, which I envision as a work of philosophy of science, focusing especially on disciplines of biological derivation and their methods, and, at the same time, on computation, seen as a method for research in these disciplines, but, first of all, as a promising theoretical framework in the light of which to try to rethink some classically philosophical conundrums, an attempt which I make when trying to reformulate as computationally feasible redescriptions what has been traditionally conceived as explanations, and as a *specification/implementation* relationship a relationship between levels of organization which has been traditionally seen as a relation of constitution.

The above questions, constituting the backbone of my proposal in philosophy of science, are more deeply discussed, as expected, in the rest of the chapters to come. I will however dedicate the next two last sections of this chapter to reflections which touch upon some questions which in a way fall outside the main scope of this work as I have just outlined it.

The first (section 1.5.1) is a reflection on the possible consequences of metaphysical nature deriving from my view about the nature of explanations, which is an eminently epistemic stance. I say that this is outside the main scope of this work because I do not want to deeply embed my discourse about philosophy of explanation in science into a metaphysical context: I am not a metaphysician, nor my stance is completely antimetaphysical, but I probably have an inclination toward being always cautious when dealing with strongly metaphysical questions. Along the lines of Kant, I cannot help but be a little suspicious about strong metaphysical claims, because they could have very serious consequences, at times, and it is not usually very clear when they can be considered adequately supported. For this reason, I planned to keep the philosophical reflection on science which constitutes the main aim of this thesis more or less free from explicit metaphysical interferences (although I guess this could be judged a highly questionable planning, and probably one of dubious feasibility).

Nevertheless, some metaphysical consequences of the general approach I adopted and of the results about computational limits on modular description which I outlined, open, it seems to me, the door to some fascinating if risky possibility of delineating a peculiar metaphysical stance, and I do not want to miss the chance to try to briefly envision this stance. I dedicate the next section to this "experimental" purpose. The sketch presented here is at best rough. What is not well thought-out here will be better left for a future occasion.

In the section which follows (section 1.5.2) I will indulge in some other possibly rash and unwise endeavor, that of trying to foresee possible long-term historical consequences of the picture of computation applied to scientific explanation which I outline in the main line of this work. In this

case, I want to leave this reflection out of the philosophical-scientific backbone of the thesis, but not because I think historical reflection is out of place in a work of philosophy of science. Quite the contrary, actually: frequently, in many parts of the main chapters, I follow, where possible, lines of historical reconstruction of the most known recent debates which center around the main questions at stake. I think, indeed, that at least a chronological, if not historical, exposition of ideas and discussions is eminently important for philosophical writing and reflection, even in a mostly analytically-informed discipline like philosophy of science. The reason I left these historical considerations out of the main scope of the present work are thus not a disdain for history of ideas, philosophy, or science, but the inherently intellectually risky nature of these reflections, which will become evident in the body of the dedicated section. Nevertheless, I value such reflections as endowed with a potential of further, probably more rigorous investigation, which will be better not ignored, at least by my future research.

1.5.1 A metaphysical attempt: Modularity as ontology? Constrained antirealism

As we have seen, an important and necessary feature of a module is its robustness: a module is something which must endure a range of perturbations, and on some timescale it must persist a certain amount of time. Another defining trait of modules is the fact that they enjoy some amount of independence from the rest of the system and from other modules, and this is due to an at least partial *isolation* of the module, to its possessing some form of recognizable *boundary*. These are properties that modules share with *entities*, or with *objects*, that is, with what can conceivably be considered the basic ontological components of the world. I would suggest that this is not a mere coincidence. I think it is plausible to say that our perceptive system operates a *modularity detection* on the raw data which impinges on our bodies, in order to yield a description of the world in terms of entities or objects: objects that we perceive are the modules produced by this process of module detection. It is as well plausible, I think, to consider the limits of this modularity detection operated by each organism, in the light of the limits we have seen as affecting modularity detection algorithms. Of course, there are some major differences here: first, the perceptive systems are *not* serial algorithmic computations, like those implemented on standard computers, and could accordingly be exempted from exhibiting the same computational complexity. However, if we take for granted a form of physical determinism (at least macroscopical) and the finiteness of perceptual biological systems, the computational equivalence of those perceptual processes with some algorithm should be quite guaranteed: perceptual systems can be seen as computational systems. We must then consider that algorithms in certain complexity classes are *inherently* intractable, regardless on the computational architecture on which they are implemented: for example, even if highly parallel architectures are used, like those of neural networks, EXPTIME problems cannot be successfully tackled, because their required time for completion grows exponentially, while the amount of parallelism can grow at most linearly[54], and the exponential function grows incomparably faster than any polynomial one. A further objection can nevertheless be raised: perceptual systems do *not* need to *discover* a good modular structure in an initially unstructured bunch of stimuli, because they are already fine-tuned to detect a more or less well-known modular structure in a world which is more or less stable: an organism is adapted to detect some kinds of objects around it, and its perceptual system is already structured and *biased* toward detection of *those* kinds of modules, that is, those kinds of objects in the world. This fine-tuning has been, in a classic darwinistic view, produced by natural selection. Such a biased modularity detection process could well be much *less* computa-

[54] Or, possibly, quadratically, or even cubically, if we imagine some futuristic three-dimensional computational "growing cube".

tionally complex than the raw process of finding an initially *unknown* good modular structure in a big set of unstructured data, which requires the performing of a NP-complete task, the task of optimizing the modularity measure Q[55]. This is, at least, quite likely: perceptual processes are probably not so computationally hard. There are, however, two rejoinders. The first is that perceptual processes are not less constrained by computational complexity, at least indirectly, than modularity optimization is constrained: perceptual processes as they are now have been achieved through natural selection (at least on darwinistic accounts), and it is *this* process, the process of natural selection, which has borne the burden of trying to optimize the modularity measure Q on an initially perceptually unstructured world. NP-completeness is in some cases so hard that not even natural selection can be deemed having had enough time to perform an exhaustive search among the phenotypic space of possible perceptual systems, so it can be argued that the actual perceptual systems coming out of evolution have *indirectly been affected* by these same computational constraints, due to the excessive computational complexity of the task of the search for the best modular hierarchical description, even for this search *during the course of evolution*. The second answer to the former objection stems from this last consideration: quite possibly because of their lower computational complexity, perceptual processes are *less accurate* than algorithmic Q optimization, and this is another way to say that optimal modularity detection on data coming from the empirical world is *not* what perceptual systems yield: perception is indirectly constrained by the computational hardness of modularity detection, in the sense that it has been rendered more or less unreliable by this computational complexity.

Another hint in this direction is the fact that on an interventionistic account of causality such as that of James Woodward, which has been adopted as a standard position by some advocates of mechanistic explanation, for example Carl Craver, modularity of the dynamical equations governing the system is necessary for separating the variables onto which to intervene during experimentation in order to discriminate single causes. Along these lines, the mechanistic decomposition of a phenomenon depends on the possible modularization of its global states. Of course, identification of the mechanism's parts, or entities, which are the causally active elements of a mechanism, depends on the successful bringing about of this piecemeal causal assessment. So, it seems that the very ontology of a mechanism, the set of its *parts*, depends upon modularizability of its dynamics. This last consideration, in the light of the computational hardness of modularity optimization, tells us that, quite likely, experimental science, is not and will not be able to yield the *most* plausible ontology of the world, not only if conducted step-by-experimental step by human subjects, but not even if fully automatized: computational hardness of Q optimization or of aggregability of variables in dynamical models[56] is impossible to overcome. It seems quite likely then, that the ontology of mechanisms discovered by science is not the best possible ontology.

One may ask where this odd view of modularity as ontology positions itself along the realism-antirealism line, and not only in the standard sense of antirealism about unobservable entities posited by scientific theories, but in a wider, all-encompassing sense questioning the *reality* of even macroscopic, mid-sized object. Moreover, if our carving of the world into sensible pieces is an effect of the possibility of modularity detection, and if this possibility depends in turn on computational constraints, it is legitimate to ask whence these constraints derive. However, it is not in the scope of this thesis to delve into deep metaphysical discussions. So, the most we can do here is to admit that if we identify modules which are feasibly detectable, given constraints deriving from computational complexity, then we could say this view is a form or what I would

[55] This is a measure of the quality of the modularity detected by an algorithm. Optimizing it means to choose the best of all the possible hierarchical descriptions of a system. See section 3.2.1.2.

[56] See sections 1.1.5 and 2.2.1.

call *constrained antirealism*: this is a position which can be certainly considered antirealistic, because, according to it, it is not known which entities are *real* per se. Better, in this view, the question could not even make much sense: the reality of entities, that is their being endowed with robustness and boundaries, is a *result* of the modularity detection process, which itself is constrained by computational complexity. The reality of entities, in this view, is not an *intrinsic* feature of entities, but it derives more from the objectiveness of computational constraints on modularity detection. So, this is certainly a form of antirealism. It is however, antirealism endowed with *objectivity*, because the computational constraints it is subject to are objective, insurmountable[57] and, albeit their consequences are of a pragmatical nature, the limitation they pose are insuperable[58]. In a sense, we could even deem constrained antirealism a form of *realism*, because the absolute objectivity of computational constraints could depose in favor of their *reality*, in the sense of independent existence. Constrained antirealism could then be considered a form of *weak realism*, although a quite different kind of realism from the typical ones, even the platonic kind. In a way, I think this could be considered a peculiar kind of *kantism*, in which the transcendental conditions take the form of computational constraints.

That said, the question about the *nature* of these computational constraints remains open: the discussion which would unfold is very thorny, and touches upon a very actual and problematic thesis, which is commonly known as the thesis of *pancomputationalism*. This is the thesis that the fundamental nature of the world is computational, or informational, and that all the so-called physical reality consists in the effects of this fundamental computational process. Debates around this issue are very complex and long-standing, calling into question philosophy of physics and also philosophy of mathematics. As said, this is not the appropriate place to examine these problems deeply. To sum up, I can say that my metaphysical position, in contrast with the choice of most proponents of mechanistic explanation, is not that of scientific realism. This is the reason I sketched above a more liberal form of explanation than strict mechanism as the elective model of explanation for complex and computational phenomena, that is, the multilevel functional explanation based on the recursive specification/implementation relation.

1.5.2 Computational methods in scientific research: a possible historical turning point?

All the above considerations could suggest an historical thesis. But I suspect that this thesis, which I would like to propose, could be conceivably accused of being not well supported. The main reason of my suspicion is that I'm not sure it can be viewed as a *historical* thesis, concerning facts of the past: it could well be a thesis about *future*, incipient historical developments in science. In any case, it is my impression that there are interesting possible facts related to computational methods, modularity and antimodularity, very near in time to the present moment, on the verge of happening, if they have not already occurred.

So, the historical thesis I am about to suggest here is *still* probably not very supported, yet it is undoubtedly a quite strong thesis. I am aware that this combination of high impact and low support is a quite dangerous one. What I would like to do is then, at least for the moment, to give only a suggestion, or some mere hint, toward the possibility that a major change of paradigm in science has just happened or could be about to happen. This change of paradigm has occurred or is about to occur, due to the availability and use, for several purposes, of powerful computing

[57] Unless, of course, P=NP. But this seems very unlikely at the moment. For an explanation, see section 17.4.1.3.3.

[58] It is not clear if some forms of hardness could be overcome by quantum computing, should quantum computing machines become feasible.

machines and algorithms in several aspects of the practice of scientific research. The uses of computers in science whose consequences I would like to consider here are two uses, belonging to two different phases of scientific research.

Let's start with the more evident: *computer simulations*. Something, I argue, has or is about to change in science from the moment when complex computer simulations have been or will come to have been accepted as proper *scientific explanations*. The point is: since its inception as Galilean physics until very recent times, modern science has, quite plausibly, taken into consideration systems which are explainable in relatively simple terms, or which are susceptible to be described by approximate models which are faithful enough (given the observer's goals) to the real empirical phenomenon. In the course of the thesis, I have tried to show that certain systems, for reasons of *computational complexity* (reasons which in a sense are pragmatic but from another point of view are objective) cannot be described in a *modular* manner. This *antimodularity* has the consequence of rendering those systems susceptible of only very low-level descriptions. The problem is that usually such descriptions don't allow for a high degree of understandability, due to the sheer amount of detail they carry. However, in many cases, *computer simulation* can nevertheless dynamically model such complex non-modular systems, making bare *prediction* feasible, at least prediction via step-by step simulation covering a certain finite temporal range: such systems are in a way dynamically predictable, but their behavior can not be *explained* in an understandable manner. Is this a real problem? Until now, it seems to have been the case that most scientific explanations *have* ended up being human-understandable. Most explanations of natural complex systems have until now, in some measure, been of this kind. The mechanistic model, of widespread use in biology, has always highlighted the necessity of a *multilevel* explanation, which brings with it a potential high degree of understandability. But, mechanistic models which are affected by antimodularity would not permit multilevel explanation, and this would render them very difficult to understand, albeit still a possible basis for computer simulation. Should such kind of low-level very complex mechanistic models be accepted as proper explanations themselves, without expecting any further improvement in their understandability?

There is a similar case in recent history affecting not empirical science, but mathematics: the case of *computer-assisted proofs*: mathematical proofs carried out automatically, at least in part, by means of computer programs. Such proofs are potentially too long and complex to be verified by humans. Consequently, since their first appearance, with the demonstration of the four-color theorem[59], computer generated proofs have made a part of the mathematical community raise objections to their acceptability: one of the possible answers is the proposal to accept as proof not the proof itself, with its excessively long list of deductive steps, but *the program* which generates it. But now the burden of the request for exactness shifts from the generated proof to the program. A program listing, if not excessively complex, is usually understandable: this property of programs comes from the fact that programs are generated by human programmers which (as I highlighted in section 1.4.9), in order to keep control of the developing program are compelled to build it in a modular way. Thus, human-generated computer programs are most often modular, and so potentially understandable. This modular understandability does not warrant the exactness of the program, that is, its conformance to its specification, but certainly eases a lot its *formal verification* (the exact proof of the exactness of the program), which, however, due to the undecidability of the halting problem, cannot be always guaranteed. So, in certain cases, the computer must be *trusted*, albeit non demonstrably so, about having generated a proof without errors. And no human could do better than the machine. Should mathematicians accept the automated proof, in these cases? The debate is still quite open.

[59] Appel & Haken (1976).

This question dealing with methodology of mathematics is similar but not identical to the one I raised above in methodology of science about the acceptability of computer simulations as *scientific explanations*: the main difference is that in mathematics there is no need for *explanation*, but only for *proof*. Understandability of a proof serves only the goal of proof *verification*. In empirical science, however, there is a need for *explanation*. What is the explanation of a computer simulated phenomenon, which, due to its being antimodular, cannot be redescribed in a human understandable manner? Is it the computer program performing the simulation, that constitutes the *explanation*?

As we have seen, the very concept of explanation itself is subject to a variety of points of view, one of which conflates explanation and logical *proof*. This is what happens in the Hempel-Oppenheim's deductive-nomological view of explanation, where explanation is seen as *logical deduction* of the explanandum from the explanans, and little to no attention is directed to the intelligibility of the explanation: a concern about understandability would have been considered improper, in the post-neopositivistic climate in which the model was proposed, carrying with it the risk of tainting scientific explanation with pragmatical, or, worse, *psychological*, whimsical, concerns. From such a stance, all that should matter for an explanation is that it is a correct deduction.

So, what about the explanation of a simulated phenomenon? If a discrete dynamical model of a phenomenon which is employed for its simulation[60] ends up being antimodular, this means that it is susceptible to only a strictly low-level description, and quite probably a very complicated one. Such a description could however appear as being generable by the reiteration of a very simple, albeit non-analytically solvable, law or rule (the update rule, see section 1.1.6). Is this an explanation *proper* of the model? If the phenomenon, as described by the given model, is antimodular, we already know by definition that no explanation of it which is more coarse-grained than this reiteration of a simple law can be given: if we don't change the model, we must content ourselves to explain the phenomenon by means of the citation of the system's initial configuration, of the update rule and by starting the program on this configuration. The program acts by repeated, potentially infinite application of the simple rule on the system's configuration in order to modify it. If we "unwind" the running of such a simulation, we obtain a list which reports first the system's initial state, and then proceeds to cite a long series of reiterations of the simple rule on the configuration, with each corresponding new configuration as the result of each application of the rule. This list would, in a way, constitute a form of DN explanation of the simulation, for it can be assimilated to a *formal deduction* starting from an initial statement, which is represented by the initial configuration, and a universal statement, in this case the simulation's update rule, which is law-like. So, can we content ourselves with such an explanation? In the thesis, I advocate for an *epistemic* view of explanation, epistemic in the ample sense employed by Cory Wright and William Bechtel[61], which entails that to explain is basically a communicative and cognitive act, and that it requires on the part of the receiver the possibility of *understanding*. The above explanation, constituted by a long list of changes of state, would not certainly be very understandable, by this standard, but this aspect could be neglected by a supporter of the DN view (let's leave aside here the DN model's own well known flaws, which could render it unacceptable).

In the case of complex biological mechanisms, composed of thousands or even orders of magnitude more parts, if most modular detection algorithmic methods failed in detecting modularity in such systems due to their size (antimodular emergence), all we could do is to produce a strictly low

[60] I mainly restrict inquiry in this work to *discrete* systems and precesses.
[61] See for example Wright (2012).

level description of the system, and proceed to simulate its dynamical behavior according to the various activities the elementary parts of the system are supposed to perform. Today, this is a quite common method employed in the simulation of complex biological networks, be them genetic, proteic, metabolic or neural ones, with an already substantial, and ever growing, mass of researches. What constitutes the *explanation* of the phenomenon, in these cases? Is it the program which runs the simulation? But the program, which is usually modular because human-written, and as such subject to be potentially understood, if the simulated system is really antimodular, by definition of antimodularity can not constitute a *high level* modular description of the phenomenon: the program simulating an antimodular system could be understood *in its own terms* as a hierarchical modular program, but not as a *description* of the simulated system. It would certainly be an understandable program, but what we would understand of it would be the fact that it simulates the complex phenomenon by reiterating an enormous number of times some simple operations, correspondingly precisely to the simple activities of the simple parts of the system which it is simulating. So, it does not seem to me that the *program* could be taken as an explanation: being isomorphic to the low-level dynamical description of the system, it would not constitute a multilevel *computational explanation*.

So, what should we view as the explanation of an antimodular computer-simulated phenomenon? I think it is *the whole dynamical simulation* which must be taken as explanation, and, given that the phenomenon is antimodular, the simulation can only be looked at and watched, but not *understood*. At least understood in a functional or mechanistic multi-level way. It could nevertheless be significantly explained in a *topological* way (see section 1.4.5): by taking into consideration some general features of the network constituting the model of the complex system, some conclusion could probably be drawn about some features of the dynamics which occurs, by simulation, on that model.

So, we return to the question: should low-level very complex mechanistic simulations of such a kind be accepted as proper explanations themselves, without expecting any further improvement in their understandability? If the answer is yes, then science has undergone a great historical change: science could eventually approach systems which, being too complex and interconnected to be the objects of modularized descriptions, should have been left out of scientific research before the advent of computer simulation. This has already happened in part, at least since three decades: just think of all the literature on simulation of complex and chaotic systems which has flourished since the '80s. It must be noted that, being most of those simulated systems, at least in certain regions of their phase spaces, antimodular, the typical explanations employed in texts about that subject (like, for example, Stuart Kauffman's works), are statistical or *topological* explanations (in the sense employed by Philippe Huneman, see section 1.4.5), a form of explanation which, as we have seen, is still allowed in antimodular systems.

Another possible way in which computers could revolutionize scientific research, a way of which a specific form is modularity detection, is in aiding to *find* a theoretical model. In this case an algorithm comes to partly substitute the researcher, not in the collection of raw data, but in the task of devising a theoretical model fitting *already* available data. And yet, this is not the whole story, because sometimes it happens that the collection of raw data itself can be automatically performed. This is especially true in cases in which the object of study is itself a *digital* object: study of texts, literature, internet content, internet structure, and so on. In any case, after the phase of data collection, there comes the need for a theoretical model able to subsume all the collected data: usually, the model is devised by the human researcher. But what about cases in which the experimenter is not able to devise such a model? What about cases in which the amount of data is so enormous that it is not to be expected that any human will be able to discern a pattern in it, in order to elaborate a theoretical model? Here, too (except in cases when the

data, albeit complex, can be *aggregated* in a way which allows for a simple description, like in statistical mechanics), it is a matter of discerning some structure in the data, some modular structure of any type, for the data to be susceptible of human-understandable modeling and explanation. Performing what is called "data mining", computers have been able in some cases to validly supplement humans in this task.

I would like to mention here a particularly surprising case. Using a method for community structure detection (that is, modularity detection[62]) in networks, Wilkinson & Huberman (2004) have been able to algorithmically analyze the published research literature about colon cancer, and automatically find modules of genes involved in colon cancer in the human genetic regulatory network, without even having to previously supply the program with the raw data describing the genetic network: all the necessary data have been automatically "mined" from the *literature*. This case is particular because, here, the data themselves are already stored in a computer-processable format, and they are not ad-hoc structured, even if, ultimately, the data (the academic literature) actually come from the work of human researchers. But another seriously surprising result is that Wilkinson and Huberman's system was able to find parts of the genetic network involved in colon cancer which had *eluded* the attention (the limited span of attention) of human researchers: the machine found some new, and probably unattainable by humans, theoretical model of a phenomenon! Now, in this case, the obtained result is still likely expressible in human understandable form, precisely because it is a *modular* description of the genetic system under observation. But what if a program, by analyzing clinical literature, found a modular model which joins into modules data of a heterogeneous nature, in a way it is unlikely any human could come to spontaneously devise? For example, by producing a modular model in which modules are composed both of genes and of proteins of the proteome, but which are only correlated in a not simple way? Could this model be still understandable by human researchers? Otherwise, what if the model, albeit modular, is composed of hundreds of mid-level modules with no higher-level modular description able to group some of them together? Consider that, due of its computational complexity, Wilkinson and Huberman's algorithm is unable to process networks with more than a few thousand genes. Because of the excessive computational complexity of the high-level modularity detection algorithm, thus, we should resort to a model of the phenomenon which is not modular at a higher level. Such a model, if valid, could be feasibly used to perform some later *simulation* of the observed phenomenon. But the phenomenon will not be easy to explain by means of the model, since its functional decomposition has been impossible, and, as it stands, it will be too complex to be understood.

But, let's speculate further: could it even be possible that the *phenomenon* itself, uncovered by the algorithmic data mining, ends up being neither a *known* phenomenon, nor an easily *understandable* one? What about a complex phenomenon which no human would have plausibly even considered, and which it is difficult for us to even plausibly describe, or *name*? Even if beyond human spontaneous intuition or even human comprehension, it can, and most probably will, be the case that computer programs come to discover such kind of phenomena. But, what will it be of science then? I think this could go potentially deeper: the task of modularity detection needs a *relation between the parts* of the system, in order to asses its modularity, and this relation and the parts themselves are given together in what I have called the *preferred description*: as I argued above[63], it is this preferred description, along with the computational constraints on modularity detection, to determine the "ontology" of the system under observation. Now, what about the possibility of *changing* the preferred description from the typical, "natural" one, to a contrived one? Given the possibility of algorithmic description of complex relations

[62] See section 3.2.1.
[63] Section 1.5.1.

between parts, then even complex, apparently *non-natural kinds* could potentially be detected as modules and be made object of science, by letting algorithms associate heterogeneous sets of real-world properties into parts and relations of a high-level description. A bizarre, completely different "world" could come out of that description. Such a non-standard description could even be hierarchically decomposed so as to be understandable, at least in principle. Or, it could give rise to a such complex hierarchical structure to be useful for explanation only in principle: due to remapping, a new set of "unnatural kinds" could emerge, and with them, new disciplines. Of course this modularization must be able to detect sufficiently *robust* modules, otherwise that would not constitute a *valid* modular description of the world. It could, moreover, be objected that the causal, common natural description of the world populated by causally cohesive objects is the only possible robust one. The question is open, and I suspect some not obvious, surprising, *valid* modularization is possible, and has already been done, namely by the quantum-level physical description[64].

All things considered, pervasive resort to this kind of artificial remapping, or to more familiar simulations like those described above, could certainly bring about in some scientific areas such a series of innovations in method and criteria as to constitute a change of paradigm in science, with the potential to see the rise of new scientific disciplines. The downside is that we will have to abandon the perspective of science as a path towards a better and better understanding of the world: the trend would be towards an unprecedented form of scientific "automated explanation", possibly unintelligible.

The above is a possible formulation of the historical thesis I wanted to argue for. Maybe it requires excessive stretch of imagination. And, it is quite clear I did not produce in this work a sufficiently strong support for this claim. That is because I aimed lower, in this dissertation. I only tried to clear the ground by proposing a series of definitions and arguing first for a property, *antimodularity*, which, if and when occurring in real phenomena, could bring about problems for certain models of scientific explanation and the need for computer simulation. The conditional above lacks support for the premise: it is likely that antimodularity occurs especially in certain complex phenomena, but I have not showed that such a kind of phenomena are so central and widespread in the scientific literature today. It will have to be seen if this is the case.

But, as a second point, I think it has to be highlighted that the growing need of resorting, when modularity can be actually found, to finding it via *algorithmic* means, and so to recur to a kind of explanation favored by advancements in computational power, could *itself* favor interest in particularly complex phenomena, or, in certain cases, bring even about a chance to "see" the existence of phenomena which could have completely escaped the attention of non-computational scientific research: automatic discovery of modular structure, where feasible, could produce prima facie unintelligible modular descriptions, if the machine is able to group parts into modules by considering contrived, unnatural but possibly significant relations between parts which had been previously invisible to human understanding. And, from that point on, a trend towards a more computational, possibly less humanly understandable science, is a trend fueled by positive feedback: such a science, if these explanations are also used to guide further research, could advance in ways that are obscure to us.[65]

[64] Probably quantum physical description is not immediately a discrete *modularization* as the ones which I consider, which are more compatible with a mechanistic view of systems. Nevertheless, quantum physics constitutes an alternative valid description of the world, respect to the commonsense one, or to that of classical physics.

[65] In order to keep the discussion inside the topics of philosophy and history of science, I do not dare here to call this future, possibly human-unintelligible, mostly-computational science, a "posthuman" science, albeit what I said could probably evoke some legitimate use of that term.

Part II

Modularity

Preamble

This major section treats the question of *modularity*, by first trying, in chapter 2, to give an informal definition of this notion and to delineate a brief history of the recent philosophical and scientific reflection on this idea.

It is then highlighted (chapter 3, section 4.2 and chapter 5) the use of the notion of modularity in three areas: network science, computer programs and discrete dynamical systems. In particular, focus is posed on the methods for automatic detection of modular structure in these kinds of systems.

After this review of modularity, in chapter 6 I will try to give some more analytic, even if not properly formal, general definition of modularity and related concepts, especially hierarchical descriptions.

In the final chapter of the section (chapter 7) I will discuss the occurrence of modularity in certain biological systems from a theoretical point of view, along with some example applications of modularity and related concepts in an empirical special science, namely biology.

Chapter 2

A first look at modularity

Modularity is a basic general notion which, upon a little reflection, appears almost universal in its applicability: at least implicitly, this idea has permeated human knowledge and human practice since very ancient times.

Typically, a *module* is seen as a more or less defined unit, possessing an identity as a whole, which can in some way be connected to other units. *Modularity* is the property a system possesses when it is susceptible to be described as a set of interconnected modules.

Examples of modularity abound, both in nature and among man-made objects and organizations. In architecture, a building is almost always constituted of modules, be them bricks, or window panes, or prefabricated macro-elements. In an organization, each office can be seen as a module, interconnected to other offices by some kind of communication channel. In turn, an office can be seen as a complex system composed by modules in the form of its workers. As a biological example, an organ is composed of cells, each of which is composed of different macromolecules, (such as proteins, DNA, RNA) which are polymers, each of which is composed of momomers, each of which is composed of atoms, each of which is composed of particles, and so (possibly) on.

Depending on the field of human knowledge in which modularity manifests itself, we could discern two broad conceptions of system modularity: one, which pertains to the construction of man-made systems and artefacts, is the view that a complex system is *built* starting from a set of basic modules, which are to be connected together and possibly recombined in various ways. Often, modularity in this sense implies the idea that a module is a standard part that can be employed in many identical or similar copies to constitute a more complex composite object. This view is typical, for instance, of architecture, of industrial design, and in general of production processes in which many simple standardized parts (the modules) are assembled together to give rise to the finished product. An obvious example is that of a building constructed by connecting together several pre-built concrete part. The same concept applies to many artifacts, as exemplified in fig. 2.1.

The other conception is that of modularity as a way to *describe* a system: a modular system can be described as *prima facie* decomposable into mostly independent parts.

In describing a system, the property of modularity consisting in the possibility of seeing modules as similar repeated parts, is sometimes secondary to the other property of modular systems, that of their susceptibility to be described as sets of semi-independent parts: it would be perfectly sensible to describe a system as composed of partially independent parts, even if these parts are all different. Nevertheless, if some of the modules which can be seen as components of a system,

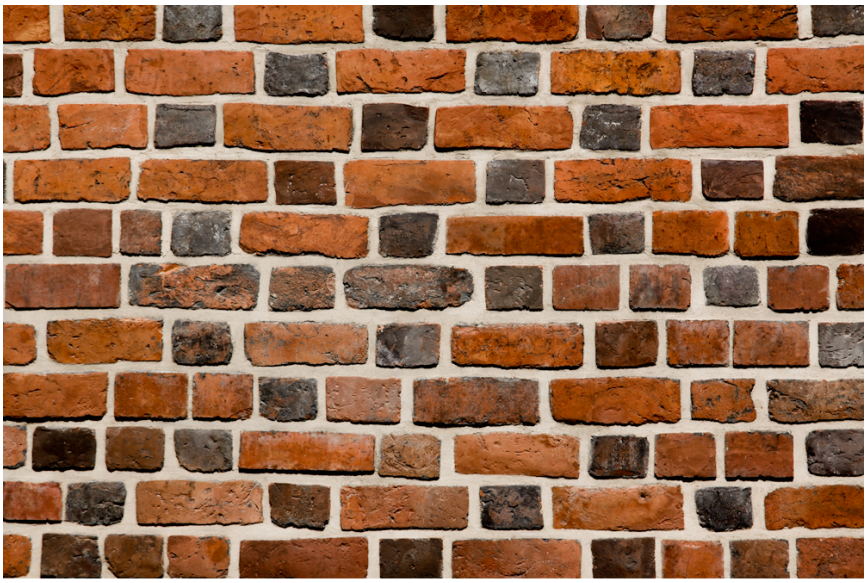

Figure 2.1: examples of modular artifacts.

appear to be sufficiently similar, this can lead to an economy of description of the system: it is sufficient to describe in detail the standard "model" of which these parts are copies or variants only once, and simply *cite* the recurrence of its copies in various parts of the system, specifying where necessary only their difference with respect to their model. Repeatability of similar parts can make this coarse-grained description of the system simpler than a fully detailed one.

In natural modular systems, especially in biological systems, these aspects of description and construction, and decomposability and repetition of parts, are most often intertwined, at various levels[1]: modules which we employ in describing an organism often coincide with modules which have arisen during the organism's development, as a modular way of "constructing" it: a trivial example is the description of the body of a myriapod (see fig. 2.2) as composed of "metamers", that is, modules roughly identifiable as recurring similar parts constituting together the trunk of the animal, each module supporting a pair of legs, with all the muscular and neural required subsystems. The similarity of these modules most probably reflects some underlying modularity of the developmental *process*, which in turn is due to some form of *functional* modularity in the genetic regulatory network underlying ontogeny. This modularity of the genetic network could also in many cases be seen as the existence of evolutionary modules, groups of genes which have undergone changes during phylogeny independently from other parts of the genome. The fact that partially independent parts of an organism can be seen as similar could reflect the economy of description cited above: the DNA specifies only the general template determining the structure of a module, and only differences between the variants have to be specified where necessary. This is typically exemplified by the presence of cellular types: each cellular type is only a variant of the totipotent original cell, variants specified by the activation of certain groups of genes. From another point of view, each cellular type is in turn the "template" on the basis of which a multitude of almost equal cells can be generated, and come to constitute a tissue.

2.1 An informal definition of modularity

From these examples, some properties of modularity appear evident: a module can be either (*i*) a single atomic element, or (*ii*) a subset of the system's elements. In the latter case, its members appear more closely and durably related to each other than to elements belonging to other modules.

It emerges from these intuitive features of modularity, that modularity is a property of *wholes constituted by distinct interrelated parts*. It can be argued that this idea of an organized composite whole is precisely what is captured by the term *system*: even in its common use, the idea of system, however vague, presupposes the possibility of distinguishing *parts* in the whole, and the fact that these parts are *interrelated* in some way. This is confirmed by dictionary definitions[2]. Another, typical feature of most, even if not all systems, is that they can be seen as dynamical, that is, characterized by configurations of their parts, or of the condition of their parts, which change in time.

Modularity can be seen as an *intrinsic* feature of systems so understood: the elementary parts of a system are modules themselves, according to the intuitions on modularity sketched above. But modularity can also appear in the possibility of seeing the system as decomposable in subsystems, that is proper subsets of the system, each of which, as a subsystem, possesses a form of individual identity, deriving from the fact that its internal cohesion is high with respect to how strongly its

[1] Most of the following will be discussed in later chapters, especially section 7.

[2] For example, a system is "an organized or connected group of things" according to the Oxford English Dictionary, and "a group of related parts that move or work together" according to Merriam-Webster Dictionary.

Figure 2.2: a centipede, *Lithobius forficatus*, shows a sort of modularity in the repetition of similar metamers.

internal parts are connected to parts external to the subsystem: each subsystem is thus a module, and the whole system, in addition to be constituted by its elementary parts, can also be seen, at a "higher level", as constituted by a set of interrelated modules. Speaking of "higher-level", it immediately appears, and it will be stressed all along this works, that the idea of modularity is related to the idea of *hierchical* levels of description of a system.

We could then envision some informal definitions of modularity and related concepts:

- A*module* must possess the following characteristics:

  i. it must have a sufficiently well-defined boundary, that is, a degree of *isolation* from the external context
  ii. it must be able to retain its unity and identity in a range of external conditions. This second clause means that a module must possess some sort of *robustness* in the face of external perturbations.

- *Repetition of identical or similar modules* is also a feature which, although not necessary to speak of modularity, occurs quite often in modular systems: the module can be seen in this case as a template for the identification or the production of instances of identical or similar copies of a given structure or element, which recur at various locations inside the system. The module becomes a *standard part*.

- *Modularity* is the fact that a system can be seen as composed of modules, in some way related or connected one to the other. But the very notion of modularity intuitively entails

that a modular system can be quite easily decomposed in the set of its separated modules: it seems, in general, that the connection *between* modules must in a way be *weaker*[3] than the cohesion of the the module, that is, than the strength of the bonds between the module's internal components. Such kind of system could be to a first approximation separated into various parts, coinciding with the modules, without affecting the integrity of each module. This seems to be a defining characteristic of modularity: even in biology, although an organism is to be certainly considered a completely integrated system, it stands to reason that what appear to us as "modules" at a macrosopic level, that is, internal and external organs, or even the segments of an arthropod's body[4], are, in a way, "deconnectable".

- *Hierarchical structure* is another typical property of many modular systems: in many cases, a system can be seen as composed of a set of modules, each of which in turn can be considered as comprising other sub-modules, and so on, until only individual atomic elements are taken into consideration. In its basic occurrence, modularity reveals itself at least at the lowest level, that of the system's elementary, atomic, parts[5], and at the highest level, that of the whole: given the informal notion of module under consideration here, it seems clear that also the system itself can be considered a module, in that it possesses a more or less stable identity and a more or less defined boundary which circumscribes it, isolating the system from the external context: this context can often be seen in turn as a supersystem of which the system under consideration is an internal module.

A general view of modularity reflecting some or all of the above intuitive characteristics is widespread, from natural sciences, to art, to most human creations. As I will try to argue later, the presence, in an object under observation, of modularity understood in this wide sense, is probably a necessary condition for any cognitively feasible scientific endeavor.

2.2 Early concepts related to modularity

Modularity is an ancient concept: modular patterns in art and architecture date back millennia. The idea of a modular organization has probably been conceived more than once along human history, in partially independent and diverse theoretical and practical fields. Nevertheless, in philosophy and in science, an explicit treatment of the question of modularity is more recent, and at least until even more recent times, the idea of modularity has probably only been taken for granted, rather than explicitly considered.

A first theoretical reflection on modularity and affine concepts, a reflection which still informs current conceptions of modularity, can be dated to the beginning of the 1960s, when a mathematical treatment, rather than a simply qualitative analysis of the concept of modularity, began to be developed. A series of questions which would lead to theorization about this notion began to arise in the field of economic sciences first, namely in econometrics.[6] A 1953 paper by Ta-chung

[3] Weaker by some kind of measure: in most cases strength, conductivity, ease of use, probability of occurring, temporal frequency of use.

[4] See for example fig. 2.2.

[5] As clarified in section 6.6 and 1.5.1, I do not see this hierarchical decomposability as ontological, at least in the commonly accepted sense of "ontological", but pertaining only *descriptions* of a system. As a consequence, as many philosophers concerned with mechanistic explanation, I do not consider the existence of an absolute bottom level, composed of atoms proper: bottoming-out is relative to the interest of the observer. See also chapter 10.

[6] Econometrics is a discipline devoted to the quantitative study, modeling and prediction of macroeconomic phenomena by means of mathematical, statistical and computational tools. For an introduction see Tinbergen (1951) and Baltagi (2011).

Liu, an economist of the International Monetary Fund[7] had focused attention on limitations in mathematical modeling of macroeconomic phenomena. The paper dealt with the construction of a mathematical model for gross national product forecasting. It highlighted the impossibility of separating, on the basis of observational data, the effects produced by some variable among the myriad ones that in a real economic system actually influence the dependent variable under observation, with the consequent impossibility of building mathematical models representing such indiscernible variables as distinct terms in their equations. Such limitation would lead to the construction of inevitably simplified mathematical models, comprising only a few *aggregate* variables: usually, a model for macroeconomic forecasting is constituted of a system of equations describing the change in time of the values of a set of variables representing aggregate economic quantities, that is, quantities which are sums, or, in general, combinations, of different kinds of quantities (for example, consumption of goods and consumption of services).

Although the terminology of the time rarely makes use of the term "modular" or any of its cognates, the concept of modularity came actually to be implicitly treated in a series of studies at the beginning of the 1960s, starting with the seminal works of Herbert Simon and his collaborators.

Simon & Ando (1961) tries to formalize the concept of *aggregation* and *near-decomposability*. In what follows, I will attempt a mostly non-formal explanation of the two concepts, and a clarification of their relation to a general concept of modularity.

2.2.1 Aggregation in dynamical systems

Let's consider a *dynamical system*, that is, a system which, at any given time, is in a certain state or configuration, that is, in a certain assignment of values to a set V of variables.

If we consider time as proceeding in discrete steps[8], the *next configuration*, that is, the configuration which the system will enter at the next time step, follows a certain *dynamics* D, that is, the next configuration is a function D of the current configuration, where *current configuration* stands for the current values of the variables of V.

We then consider a subdivision S of the set V of original variables. That is, we divide the variables into several *possibly overlapping* groups.

Applying a certain *aggregation function*, we obtain for each group a single *aggregate value* that represents an aggregate of the values of that group's variables. Thus, each group can be represented by its aggregate value, computed according to the aggregation function.

Now, we consider the *aggregate configuration*, which is the set of all the aggregate values[9].

The aggregate configuration will have *its own* dynamics, which is given by a certain function D'. This function, applied to the current aggregate configuration, gives the next aggregate configuration.

This way, we have performed on the dynamical system what can be called an operation of *aggregation*.

For an operation of *aggregation* to be valid, a condition, which I will call here the *aggregation condition*, must hold: the condition that the *corresponding aggregate configuration* of the next

[7] Liu (1955).

[8] Mostly, in the rest of this thesis I will take into consideration systems which are discrete in time and space. A better specification of this question is carried on in section 5.1.1.

[9] Let's remember that the *configuration* is the set of values of all variables of V, instead.

configuration[10] be equal to D' applied to the current aggregate configuration. In other words, the condition states that if we applied the dynamics D' to the current aggregate configuration we would obtain another configuration which is equal to the corresponding aggregate configuration of the next configuration[11].

For example: if the aggregation function is the mean of the values it applies to, the condition stated above means that the evolution of the configurations of the means (of the subsets in the subdivision S of the set V of non-aggregate variables) must be equal to the mean of the original non-aggregate evolution.

We could rephrase the above example like this:

1. We take the current configuration, that is the configuration at the current time step.
2. We apply its *dynamics* to it.

[10] The *next configuration* is the set of values of the original (non-aggregate) variables of V at next time step, a configuration obtained by applying D to those variables' current values. The *corresponding aggregate configuration* of the next configuration is obtained by applying to it the subdivision S, and then calculating each aggregate value for each of the sets in this subdivision.

[11] The above mostly informal exposition could be expressed in a more formalized way:

We consider dynamical systems, that is: systems which, at any given time t, are in a certain *state* or *configuration* $c_{(t)}$, which is a certain assignment of values to the elements of a set v of variables:

$$v = \{x_1, ...x_n\}$$

Given a configuration $c_{(t)}$ in which the system happens to be at time t, that is, the values at time t of the elements of the set v, the *next configuration* $c_{(t+1)}$ in which the system will be (that is, the configuration of the system at time $t+1$) is a certain function D (the *dynamics*) of the configuration at time t:

$$c_{(t+1)} = D(c_{(t)})$$

Since c is a set of values of the variables $x_1, ...x_n$, we could write

$$c_{(t)} = \{x_{1(t)}, ...x_{n(t)}\}$$

where $x_{i(k)}$ is the value of variable x_i at time k . Substituting the values of the variables at time $t+1$ to $c_{(t+1)}$, we get

$$\{x_{1(t+1)}, ...x_{n(t+1)}\} = D(x_{1(t)}, ...x_{n(t)})$$

We then consider a subdivision of the set of original variables. That is, we divide the variables into m possibly overlapping groups.

Applying a certain *aggregation function* A to the variables of each group, we obtain for each group a *single* value that represents an *aggregate* of the values of that group's variables. Thus, each group can be represented by its aggregate value X_i, computed according to the aggregation function A.

Now, we consider the *aggregate configuration* C, which is the set of values of all the aggregate variables X_i. The aggregate configuration will have its dynamics D' :

$$C_{(t+1)} = D'(C_{(t)})$$

The following equation, which I call the *aggregation condition* must hold, for the aggregation to be valid:

$$A(c_{(t+1)}) = D'(C_{(t)})$$

that means that, if we take the aggregate configuration at time t and apply to it the aggregate dynamics function D', we get the same result that we would get if we made the original non-aggregate system evolve according to its dynamic D, and then, afterwards, we applied the aggregation function to the whole original set of variables $\{x_{1(t+1)}, ...x_{n(t+1)}\}$.

3. We obtain the *next configuration*, that is the configuration at the following time step.
4. We take the current configuration, that is the configuration at the current time step.
5. We subdivide it in several subsets according to a division D.
6. We obtain the *aggregate configuration*, a set of values each of which is the mean of the elements of one of the subsets obtained at point 5.
7. We apply another dynamics to the aggregate configuration.
8. We obtain the *next aggregate configuration*, that is the aggregate configuration at the following time step.
9. We take the *next configuration* obtained at point 3.
10. We subdivide it in several subsets according to the division D.
11. We calculate every mean of the elements of this subdivision, obtaining an aggregate configuration.
12. The aggregation condition holds if this last configuration is equal to the configuration obtained at point 8.

2.2.1.1 Approximate aggregation

The fact that the aggregation condition holds means that the evolution through time of the aggregate system faithfully copies the aggregate version of the original evolution (where with "aggregate version of the original evolution" I mean the evolution obtained by calculating each subsequent configuration of the original system according to its dynamics, and, *afterwards*, by aggregating the values of the last of these configurations).

The aggregation condition could in many cases be relaxed: we could demand the aggregated system's dynamics to just track the original system's more or less faithfully, up to a certain acceptable maximum amount of error, established according to our aims and goals. In other words, we could demand just an *approximated* aggregation, which reproduces well enough for our purposes the behavior of the original system by means of a simplified version of its theoretical model.

The evolution of the aggregate configurations through the dynamics D' can also be seen as *another* dynamical system (let's call it the *aggregate system*), different from the original non-aggregate one. We could then see the aggregation condition as stating the fact that the evolutions of these two systems must not (in a way) *diverge* with time: More precisely: that the evolution of the original system, once processed by the aggregation function[12], coincides up to a maximum error with the evolution of the aggregate system. This circumstance can also be rephrased by saying that the aggregate dynamics is *valid*. The approximate version of the aggregate condition could however not hold in non-linear systems, where the evolution of their global configuration can in time diverge exponentially from the aggregated one.

2.2.1.2 Aggregation is computationally hard

The fact that the aggregation condition holds or not, of course hinges on the choice of an appropriate subdivision of the set of the dynamical system's variables, of an aggregation function and, depending in turn on these, on the finding of an appropriate, valid, aggregate dynamics.

Unfortunately, the search for a suitable aggregation of variables requires a complete enumeration of all possible subdivisions of the set of variables, in order to find the one which yields the

[12] That is, once gone through a subdivision according to S of each of its configurations, and once each of the obtained subdivided configurations are processed into *aggregate configurations*.

smallest error[13] when employed in an aggregated model of the system's dynamics. This search for an optimal subdivision of variables is an extremely demanding task, from a computational standpoint. Actually, the task has turned out being *NP-complete*, as proved in Winker (1992).

More recently, Kreinovich & Shpak (2006), and Kreinovich & Shpak (2008), have also proved that even *approximate* aggregability is *NP-hard*, already in *linear* systems.

This means[14] that we cannot hope to find a general algorithmic method that, applied to a mathematical model of the system's dynamics, can always produce in feasible time a valid aggregate, simplified, and even approximate version of the model.

This does not mean that aggregated dynamical models can never be found: the problem of aggregation can in many cases be more easily solved if we have some prior knowledge about the system which can guide us in partitioning the system's variables into sensible semi-independent subsets. An example classically proposed by Herbert Simon, which will be cited in section 2.2.3, the example of the office cubicles, shows that a plausible aggregation of a dynamical model can be produced based on prior knowledge of the system's structure and of dynamical properties of this structure.

2.2.2 Decomposability

When to a system's configuration is applied a division which produces a *partition*, that is a complete subdivision of the original system into non-overlapping subsets, then we can say that the system is being *decomposed*.

In a completely decomposed system, every subsystem is independent from each other: it changes in time[15], but the fact that the system is *decomposed*, that is, *partitioned*, means that no influence can occur *between* different subsystems.

We can apply an *aggregation* to such a *decomposed* system, by substituting to each subsystem an aggregate value of the values of its internal variables.

If a decomposed system is aggregated this way, *and* the aggregation condition holds, the system is said to be *decomposable*.

The fact that a system is decomposable means that the original system can be seen as a group of *non-interacting*, completely independent subsystems, each of which undergoes its evolution according to its own internal rules. In these cases, the whole dynamical system is equivalent to another one in which each one of the independent subsystems *is substituted by a single variable* which represents the aggregate value of the variables of that subsystem.

It's easy to see that this notion of a decomposable system quite directly fulfills the intuitive features of modularity: the fact that the system is composed of completely independent subsystems, each of which maintains its own cohesion (in the sense of being independent from all the others and as such not interfering with, nor being influenced by them), recalls the idea that a module can be seen as a set of elements, possessing a sufficiently well-defined boundary (in this case the "boundaries" of the partitions), endowed with some robustness (the fact that, for the aggregation condition to hold during evolution, the partition of the system must endure). In particular, it is trivially fulfilled here the property of a modular system to be composed of modules which are weakly connected each other: the subsystems are in fact *completely* independent.

[13] See sections 2.2.1, 2.2.1.1 and 2.4.

[14] See section 17.4.1.1.

[15] A subsystem changes according to the part of the dynamics' function which mentions *only* its own internal variables.

Given that in the case of a decomposable system, to each subsystem, which is composed of more than one variable[16], is substituted a single variable in the aggregate system, this last system is composed of *less* variables than the original one.

In cases of decomposable systems, then, it is sufficient, for the purpose of calculating certain aggregate values of the system in the future, to perform the calculation not the system's original dynamics (which should take into account *all* the variables of the original dynamical system), but on the dynamics of the aggregate system. The advantage of calculating the dynamics of the aggregate system is that its number m of (aggregate) variables is *smaller* than the number n of the original variables, and so the computation can be easier than in the original case.[17]

2.2.3 Simon-Ando near-decomposability

The problem of validly decomposing a system is that of finding an adequate partition of the variables into a number of groups smaller than the number of variables, an adequate aggregation function, and, given those, an adequate aggregate dynamics D'. It turns out that only in a few cases these requirements can be satisfied: namely only when a system is actually composed of completely independent subsystems. This seems indeed to be a quite special case, because, usually, interesting systems are *not* a jumble of different unrelated systems.

But, it turns out that there are many more cases in which a system is *nearly-decomposable*. This is the original terminology introduced by Herbert Simon and Albert Ando in 1961 in a paper, *Aggregation of Variables in Dynamic Systems*[18], which can be probably considered the ancestor of all the contemporary literature on modularity.

A nearly-decomposable system is a system which can be seen, *with some approximation*, as a decomposable system. In order to better explicate this property, I point here to a typical example, which is made in both Simon & Ando (1961) and Simon (1962):

> Let me provide a very concrete simple example of a nearly decomposable system. Consider a building whose outside walls provide perfect thermal insulation from the environment. We shall take these walls as the boundary of our system. The building is divided into a large number of rooms, the walls between them being good, but not perfect, insulators. The walls between rooms are the boundaries of our major subsystems. Each room is divided by partitions into a number of cubicles, but the partitions are poor insulators. A thermometer hangs in each cubicle. Suppose that at the time of our first observation of the system there is a wide variation in temperature from cubicle to cubicle and from room to room – the various cubicles within the building are in a state of thermal disequilibrium. When we take new temperature readings several hours later, what shall we find? There will be very little variation in temperature among the cubicles within each single room, but there may still be large temperature variations among rooms. When we take readings again several days later, we find an almost uniform temperature throughout the building; the temperature differences among rooms have virtually disappeared.[19]

[16] Excluding the trivial case of a partitioning in which each partition contains a single variable.

[17] The fact that the computation of the aggregate dynamics is easier that that of the original one is not guaranteed, even in the case of completely decomposable systems: this depends on the actual dynamics inside each module, and could not hold in the case of highly nonlinear dynamics internal to modules. In that case, even if the aggregate dynamics contains a lower number of distinct variables than the original dynamics, the function representing the aggregate dynamics could happen be at least as complex as the original dynamics. See also next section.

[18] Simon & Ando (1961).

[19] Simon (1962), p. 474.

For a depiction of this example, see fig. 2.3. Simon numerically illustrates the example with a table of data similar to table 2.1.

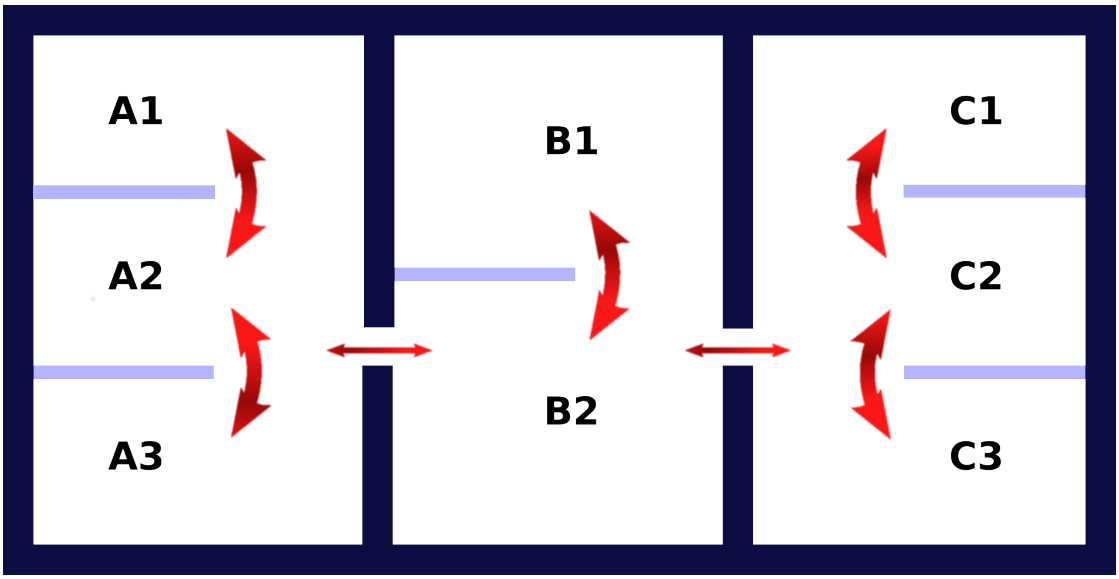

Figure 2.3: a map showing the office in Simon's example: the double arrows represent the heat exchange rates between rooms or their subdivision (the cubicles): thicker arrows mean higher rate of heat exchange between cubicles in the same room, thinner arrows lower heat exchange rate between different rooms.

	A1	A2	*A3*	*B1*	*B2*	*C1*	*C2*	*C3*
A1		100		2				
A2	100		100	1	1			
A3		100			2			
B1	2	1			100	2	1	
B2		1	2	100			1	2
C1				2			100	
C2				1	1	100		100
C3					2		100	

Table 2.1: values of heath exchange rates in the near-decomposable system in Simon's example. In this and the following table, different colors indicate the different *modules*: in the example's case, the different rooms.

In the table, Ai are the cubicles of room A, Bi are those of room B and Ci the cubicles of room C. Each point in the table, located at (X, Y) reports the heat exchange rate between cubicle X and Y. Heat exchange rates among cubicles of the same room are located, due to how the table is constructed, *along the diagonal* of the table. Heat exchanges between cubicles belonging to *different* rooms are *external to the diagonal*.

By looking at the values of Simon's example as represented in table 2.1, it appears immediately evident that the most intense heat exchange is along the diagonal, that is, among cubicles of the same room, and that heat exchange between cubicles of different rooms is *order of magnitudes* lower. If we decide to *approximate* the system by considering these lesser rates as *null* when they

fall under a certain threshold which we have conventionally postulated, we obtain a table similar to table 2.2.

	A1	A2	A3	B1	B2	C1	C2	C3
A1		100						
A2	100		100					
A3		100						
B1					100			
B2				100				
C1							100	
C2						100		100
C3							100	

Table 2.2: a table representing the *decomposable* system in Simon's example, obtained from the original system by ignoring the values of heath exchange rates which fall below a certain threshold.

In this second table, only heat exchange rates along the diagonal are not-null, which means that there's *no* heat exchange between cubicle of different rooms, that is, that the rooms are *completely independent subsystems*. A system composed of independent subsystems is a *decomposable* system, in that its aggregate evolution can be *exactly* described by taking into consideration only one aggregate value for each of their subcomponents (the *modules*, in this case represented by the rooms). The fact that only the diagonal presents non null values is the signature of a decomposable system, if represented by this kind of tables.

Of course, the second table would exactly describe the system *only* in case this is exactly decomposable, and, as such, constituted of completely independent subsytems. We know this is *not* the case for the example of the rooms, in which *some* heath exchange occurs between different rooms. But, if this inter-module exchange is sufficiently smaller than the exchanges occurring among cubicles of the same room, as in the example, we could, at the cost of an acceptably small error in describing the system's dynamics, treat the system *as if it were* decomposable. That is, we could consider the system a *nearly decomposable* system.

It must be noted that this presupposition of a near-equivalence between the approximate model and the original may not hold for *nonlinear* systems, in which a small approximation in the equation describing their dynamics could lead to massive divergence of the actual dynamical evolution of the system from the predicted one. In fact, Simon and Ando explicitly target only *linear* systems in the cited papers.

Thus, an extension of this simplification technique to *nonlinear* systems seems at first problematic, for these systems can, and probably will, greatly diverge in their asymptotic long-run dynamical behavior compared to the simplified model. However, some particular classes of nonlinear systems have been shown to be nearly-decomposable. Fisher (1963a) and Fisher (1963b) proves near-decomposability conditions for certain nonlinear systems used in econometrics. Cale, O'Neill, & Gardner (1983) proofs that, in modeling certain ecological systems, there are quite stringent condition a nonlinear system must respect to be considered near-decomposable and, as such, be treated with an acceptable approximation by means of an aggregate model. In the same field of ecology, two works, Iwasa, Andreasen, & Levin (1987) and Iwasa, Levin, & Andreasen (1989) state the conditions for perfect and approximate aggregation in nonlinear dynamical systems. They find that conditions for perfect aggregation or aggregation with acceptable approximation are dependent on many factors, including the desired time-scale at which we want to observe the system's behavior, and the choice of the features of the system the observer is

interested in. In any case, here too, there is a trade-off between aggregation degree[20], and the precision obtained.

However, it seems defensible to say that some nonlinear systems can not allow for any kind of approximation effected by ignoring weak links between some of their parts, because the approximated model would always diverge from the exact one even in the short run. This could be the case when, even if the intermodular structural links are weak, the connected modules can influence each other in a nonlinear way: this can produce a very strong short-term reaction as a consequence of even a faint variation in the intensity of the interaction, thus making it so that the presence of the link cannot be ignored: in this case, the system can not be simplified. Prediction of the dynamical evolution of systems of this kind would almost certainly have to be effected by using the non-approximated, and as such non aggregated, model, with all its computational cost in terms of the potentially high number of variables to be considered.

2.2.4 Timescales and decomposition in nearly-decomposable systems

Regarding the original idea of near-decomposability, in their 1961 paper[21] Simon and Ando proved two theorems, which state that:

> (a) in a nearly decomposable system the short-run behavior of each of the component subsystems is approximately independent of the short-run behavior of the other components;
> (b) in the long run the behavior of any one of the components depends in only an aggregate way on the behavior of the other components.[22]

I think the two above properties could be rephrased more perspicuously as:

i. in a nearly decomposable system the fine time-scale internal evolution of each of the modules can be modeled by ignoring any influences originating form the module's external environment, that is, as if the system were completely decomposable;
ii. at a coarser time-scale, the evolution of the systems can be modeled by considering only its aggregate dynamics, obtained by aggregating the evolution of each module into a single value.

These properties of a nearly-decomposable system allow for the following treatment of the system's dynamics: when the observer is interested in the behavior of the system at a fine timescale, the subsystems can be treated as independent modules, and each module studied separately from the others. When interested in the long run behavior of the system, it is sufficient for the observer to substitute the system with an aggregate version of it, that is with *another* dynamical system in which a single variable represents the aggregate state of each subsystem of the original system.

In nearly-decomposable systems, there is an evident decoupling between the timescale of the interesting intra-module interaction and that of the interesting inter-modules interactions. This

[20] With "aggregation degree" I mean here the degree of computational simplification of the system's dynamical model obtained by simplifying it by considering it decomposable and by aggregating the decomposed system's dynamics.
[21] Simon & Ando (1961).
[22] Simon (1962), p. 474. Those cited here are less formal statements of the same two theorems, as phrased by Herbert Simon.

is an important characteristic of modular systems, and it will be discussed in next section and further on[23]

2.3 Hierarchical modularity

It seems clear that, in many cases, the description of a complex as composed of modules loosely connected together can be applied recursively to the modules themselves, giving rise to the view of a hierarchical modular system. This seems a view naturally applicable to many kinds of systems of some interest: biological, sociological, economical, and many others.

In Simon's case, a decomposable or nearly decomposable system can be naturally and intuitively seen as a three-levels[24] hierarchy: the whole system, which represents the "highest" level, is composed of its elementary parts, which constitute the "lowest"level. But, being nearly decomposable, the system can also be seen at an intermediate level of description as constituted of nearly independent subsystems, the subsystems into which it can be sensibly decomposed or nearly-decomposed. This kind of decomposition can in certain cases be applied to each subsystem in turn, and so on, until a reasonable set of "bottom level" elements is found.

Simon (1962), Tries first to characterize hierarchical systems in a more general way. According to him, hierarchical organization is typical of what he calls "complex systems", loosely defined as follows:

> Roughly, by a complex system.[25] I mean one made up of a large number of parts that interact in a nonsimple way.[26]

Simon writes:

> By a hierarchic system, or hierarchy, I mean a system that is composed of interrelated subsystems, each of the latter being in turn hierarchic in structure until we reach some lowest level of elementary subsystem. In most systems in nature it is somewhat arbitrary as to where we leave off the partitioning and what subsystems we take as elementary. [...] For certain purposes of astronomy whole stars, or even galaxies, can be regarded as elementary subsystems. In one kind of biological research a cell may be treated as an elementary subsystem; in another, a protein molecule; in still another, an amino acid residue.[27]

[23] Especially in section 6.7, 3.2.4.1 and chapter 6.

[24] I use here the term "level" in an informal and intuitive way, but I will try to elaborate on the question of levels in section 6.6.

[25] It must be noted that the way Herbert Simon uses the expression "complex system" runs against a well-established tradition, preceding and following Simon's proposals, namely, the tradition to see as *complex* mainly those systems which can be loosely defined as "chaotic": that is, systems composed of a usually large number of parts interacting in nonlinear ways. This are circumstances which impede approaching the system's behavior in a "piecemeal" fashion by subdividing it in near-independent subsystems, each one more tractable than the whole. "Complex system" understood in this "holistic" fashion seems to amount to exactly the *opposite* of "complex system" in Simon's sense. This must be explicitly stated to avoid some misunderstanding in analyzing Simon's work.

[26] Simon (1962), p. 468. Simon's text continues this way: "In such systems, the whole is more than the sum of the parts, not in an ultimate, metaphysical sense, but in the important pragmatic sense that, given the properties of the parts and the laws of their interaction, it is not a trivial matter to infer the properties of the whole. In the face of complexity, an in-principle reductionist may be at the same time a pragmatic holist." (p. 468). I think the position hinted at here is a very interesting attempt to avoid the typical metaphysical conundrums of the reductionism/antireductionism debate. I will try to propose a somewhat analogue position, based on a certain kind of pragmatical considerations, in section 14, regarding the *explanation* of complex systems.

[27] Simon (1962), p. 468.

It has to be noted that this arbitrariness in stating the "bottom" level is quite parallel to the analogous question about deciding which "bottom entities" to consider in mechanistic explanations. In fact, it can be argued that the whole conception of mechanistic explanation is a form of individuation of a hierarchical structure in Simon's sense and of use of this structure to explain the functioning of the system. This is not only implicit in the "new mechanistic" line of thought, an important line of contemporary philosophical reflection which will be expounded in chapter 10: this analogy between mechanistic explanation and Simon's view is explicitly acknowledged by some authors, for example by Bechtel & Richardson (1993).

According to Simon, the recursive division of the system into subsystems to form a hierarchy can be effected, in many cases, by treating the system as nearly-decomposable[28] at each hierarchical level: each level can be seen as constituted of elements, which can be grouped into subsystems whose internal elements appear more strongly interrelated one to the other than to parts external to the subsystem. Near-decomposability turns to decomposability by treating the system as if it were completely decomposable, a simplification which is realized by ignoring the relational links among elements whose intensity falls under a specified value[29].

For example, we could take a population, which at the lowest level of description is composed of separated individuals. By subdividing the population into subsets individuated by area of residence, we could obtain an intermediate level of description. At this intermediate level, the parts are the modules, that is, the local communities, whose internal members present a stronger affinity between them than members of different communities. These modules, the local communities, which at this intermediate level can be considered the elementary parts, can be in turn grouped to form regional communities, each of which is internally constituted *of local communities*. And so on, until the whole population is viewed as a single "elementary" part: this is the highest hierarchcal level.

It is important to highlight that, as a result of this procedure, the detected hierarchy can in this case be seen as dependent on two choices: (*i*) the *specific relationship* among the parts we want to consider, and (*ii*) the chosen *threshold* under which a relation is to be considered null if its intensity falls below it.

2.4 Generic near-decomposability.

It must be considered that, upon examination, the original notion of near decomposability proposed by Simon and Ando turns out to be a form of modularity *of the equations* describing the dynamics: exchange rates are properties of the *mathematical* expression of the system's dynamics, that is of its dynamical *model*. The fact that in certain cases low exchange rate coefficients can be considered null is a property of the *mathematical* model. I would call this form of modularity *dynamical model modularity*, and will better analyze it in section 6.5. But the notion of modules as internally strongly connected structures more loosely interrelated one with the other is more general, and should be applicable also to systems not explicitly modeled by a system of equations, or to statical hierarchical systems. Actually, a similar criterion can be easily devised for static modular hierarchies: namely, it is sufficient to substitute a metric of static proximity or affinity between elements to that of exchange rate[30].

[28] See section 2.2.3.
[29] See section 2.2.3.
[30] That modularity detection requires the choice of a particular metric is an essential property of modularity, which is to be discussed in section 2.5 and 6.3.

I would thus like to propose a principle of *generic near-decomposability*, allowing for the simplification of a system by considering very weak links between its elements as null, a principle that can be applied to any kind of system which shows variations in magnitude of some relation whatsoever between its elements. That the hierarchical decomposition effected according to this criterion turns out to make sense or not, will depend on the relation taken into consideration, and on the chosen threshold of the relation's magnitude under which the relation has to be considered null.

Thus, the idea, highlighted above, of arbitrariness in individuating the bottom level, could seem to extend to arbitrariness in individuating the levels of the system's hierarchical structure, when this structure is detected by considering a nearly-decomposable system as decomposable: this impression is due to the apparently arbitrary choices of the kind of relation among the system's parts, and of a threshold level. But, given that, when considering interesting systems, it is *science* to be charged with the decision about which relation to take into consideration and which threshold can be considered useful for an acceptable approximation of the system, it could be argued that the arbitrariness here is constrained: after all, science necessarily seeks a certain degree of precision in modeling a system, a degree which varies according to the contextual scientific purposes. So, the choice of the simplified model is not completely arbitrary: it is always based on the trade-off between acceptable precision in science and the computational effort needed to simulate or explain the system's dynamics at the highest possible precision (its non-simplified dynamics) or to fully describe the system at the highest possible level of detail, in the case of a static system.

Importantly, as we will see, in the case of *explanations* two other conditions must be highlighted: (*i*) it must be taken into consideration also the computational effort needed *to find out* the hierarchical description to be used in the explanation, and (*ii*) it is my contention that we should also consider our *cognitive limits* in handling too complex explanations. these two conditions together raise a whole host of problems and questions which constitute one of the main concerns of this work, as we will see in later chapters.

Regarding acceptable simplification and the occurrence of near decomposability in systems studied by science, Simon notes:

> In the more general sense, justifications for approximation must be related to the decisions that depend on the approximating – if the decisions based on the approximate model are not much "worse" than the decisions based on the more elaborate model according to some criteria, then we may be justified in using the approximate, simpler model. This consideration is strengthened if, while the improvement of the final decision is very slight, the cost of working with a larger model is very much greater than that of working with an approximate, simpler model[31].

> [...] near decomposability is a rather strong property for a matrix to possess, and the matrices that have this property will describe very special dynamic systems – vanishingly few systems out of all those that are thinkable. How few they will be depends of course on how good an approximation we insist upon. If we demand that epsilon[32] be very small, correspondingly few dynamic systems will fit the definition.[33]

[31] Simon & Ando (1961), p. 112-113.
[32] Epsilon is the maximum approximation error [my note].
[33] Simon (1962), p. 475.

Although, among randomly generated systems, the odds of finding nearly decomposable ones are "vanishingly small", Simon notes that, in actuality, many systems studied by science are in some way modular:

> But we have already seen that in the natural world nearly decomposable systems are far from rare. On the contrary, systems in which each variable is linked with almost equal strength with almost all other parts of the system are far rarer and less typical.[34]

As we will see in section 7.1.1, Simon has other independent reasons, mainly based on abstract evolutionary considerations, to believe that natural systems, especially biological ones, manifest modularity, and that non-modular systems are fairly rare. Other reasons can be *empirical*: the physical forces in atoms and molecules vary by orders of magnitude at different scales[35]. Accordingly , time scales are different: atomic vibration is of a much higher frequency than molecular one. In this case the choice of the correct hierarchical description is not arbitrary.

Regarding hierarchies, Simon informally introduces the notions of *flat hierarchy* and *span*. According to him, a hierarchy as usually understood, is the division of a system in a *small or moderate* number of subsystems: such a number is the *span*. A hierarchical system is *flat* at a certain hierarchical level when it has a *wide* span[36]. Systems with flat hierarchical levels are not very easily modeled, because even their modular descriptions are constituted by high numbers of parts, parts which must all be considered when explaining or simulating the system.

2.5 Modularity is relative to the choice of a metric

It emerges quite clearly fro what has been said until here that a fundamental property of modularity is that of being relative to a relationship holding between parts of a system: modules are precisely those sets of elements of the system whose internal connections are stronger than those with the external context. But the nature of these "connections" can change in countless ways: any relation holding between at least couples of elements can be chosen, and can be considered a metric: for example, in a social group a first metric can be that of genealogy; a different one can be that of love relationships, a third one that of party membership, or that of the difference between two person's heights. Without loss of generality, we could even conceive ad-hoc all-or-nothing strange relationships, or even properties of single elements, instead of n-ary relationships, like fro example that of "being an american actor or being older than one of his uncles": such a feature is a property of single elements, but it has an abvious corresponding discrete metric of proximity between elements, which is that of sharing the property or not.

Of course, given a certain system under observation, changing the relationship taken into consideration the possible corresponding modular structure which can be detected in the system changes accordingly, often drastically. In many cases, modularity can appear or disappear as the metric changes.

But, how to choose an adequate metric? In the former section, I claimed that, given that it is science that we are dealing with, this choice is made according to some criteria of scientific relevance and opportunity relative to the aims and purposes of the discipline under consideration. In many cases, a well-established science has a set of intended relationships to base on when

[34] *ibid.*

[35] From stronger to weaker: covalent bonds, ionic bonds, hydrogen bonds, intermolecular bonds.

[36] I will make use of a slightly revised notion of *flat hierarchy* in section 6.6, and in section 13.2 when introducing the definition of *antimodularity*.

considering modularity of the systems it considers[37]. Albeit affected by these constrains, the point to highlight is that a specific choice of the relevant relationship or metric is always necessary, albeit this choice can be implicit. It is this *pragmatic* dimension bearing on the development of scientific theories which deserves attention here. In section 13.3 I will argue that there are other, inescapable constraints, of a computational nature, which bear on the choice of a metric used to assess modularity.

2.6 Summary of the chapter and outlook

In this chapter, I gradually outlined a notion of modularity starting from intuitive observations and seminal works in the literature of the past fifty years which started to touch upon concepts related to modularity. Based on that, I tried to delineate a core set of features which characterize many forms of modularity, and which will be of use in all the rest of this work: *aggregability*, *near-decomposability*, *hierarchical structure* and *temporal decoupling* between dynamics at different hierarchical levels. Most of these notion will be further discussed and, where possible, generalized, in chapter 6.

In the next three chapters, I will highlight the importance of the generalized notions of modularity based on near-decomposability, and of hierarchical descriptions, in three fields of theoretical research: networks, discrete dynamical systems and computer programs.

[37] See section 6.6.9 and the notion of *preferred description*, a notion which has already been touched upon in section 1.1.1 of the Introduction.

Chapter 3

Modularity and networks

The concept of modularity, as sketched in the preceding chapter, is readily applicable to *networks*. In this section, after introducing the notion of network, which is an abstract device essentially composed of nodes linked together, and some of the interesting features of networks, I will proceed to specify how network modularity can be understood.

Intuitively, a modular network can be seen, based on on a general conception of modularity like that introduced in section 2.1, as a network in which it is possible to identify subsets of nodes whose elements are more connected to each other than to nodes external to the subset. Although Simon's view of near-decomposable systems is inspired by the same intuition, his approach applies to systems quite different from networks generally intended (even if, as we will see[1], the systems Simon takes into consideration could be considered specific cases of networks), and as a result not all of the properties of those systems apply to modularity in *networks*. On the other side, modularity detection in networks has usually focused (with some exception, as we will see) on determining modularity in the network *structure*, and not necessarily in the networks's *dynamic functioning*. Static structure and network dynamic functioning are certainly related, but not always in a simple way, as discussed in chapter 6. In this section I will take for granted an intuitive understanding of the fact that a network's structure can support a dynamics on it: just think of an electric network and the current flowing on it. For a proper discussion on the relation between network structure and network dynamics, refer to section 6.2.

While modularity can be intuitively spotted at first sight in the graphical representation of simple networks, this is no more the case when networks coprise hundreds or thousands of nodes: their graphical representation would look in most cases as an intricate mess. fortunately, since the early 2000s, some algorithms for automatic detection of modularity incomplex networks have been devised. The main concern of this chapter is to describe the features of the best known of these algorithms, and to highlight their downsides. Resorting to these computational methods has revealed to be necessary in many special sciences dealing with enormous networks, such as molecular and cell biology, which, with the advent of automatic methods of experimental discovery of the molecular networks which make up cells and organisms, have to face enormous amounts of raw data about the structure of these networks: detection of modularity in these networks would certainly ease their comprehension and further research.

[1] In section 6.2.

3.1 Networks and network science

Abstractly conceived, a *network* is a set of items, called *vertices* or *nodes*, with connections between them, called *edges*.[2] The number of edges linked to a node is the *degree* of that node. If the edges have a *direction* (as for example in the case of causal connections), the network is called a *directed* network, *undirected* otherwise.

Network models have been employed in the last few decades for modeling a multitude of systems of heterogeneous type, ranging from physical complex systems, to computer and communication networks, to social, organizational, ecological, economical, neural or other biological networks (mainly genetic, or metabolic, or protein networks).

Depending on the chosen network, which is usually seen as a *model* of some empirical phenomenon, edges can represent any kind of relation between elements of the system: be it causal influence, information transmission, parenthood, acquaintance, and so on. Depending on the type of relation represented, edges can possess a direction, in which case the network is a directed one.

As said, interest in the properties of networks and their application as models of real phenomena has gradually increased over the last five decades, under a multitude of approaches, in a varied spectrum of disciplines, from mathematics (graph theory) to biology, to sociology. All this heterogeneous literature, according to some, like Börner, Sanyal, & Vespignani (2007), expose the need for an interdisciplinary but explicitly defined research field: *network science*, understood as the study of abstract properties of networks applicable to disparate real-world situations. In the words of these authors,

> Today, the computational ability to sample and the scientific need to understand large-scale networks call for a truly interdisciplinary approach to network science. Measurement, modeling, or visualization algorithms developed in one area of research, say physics, might well increase our understanding of biological or social networks. Datasets collected in biology, social science, information science, and other fields are used by physicists to identify universal laws. For example, unexpected similarities between systems as disparate as social networks and the Internet have been discovered [...]. These findings suggest that generic organizing principles and growth mechanisms may give rise to the structures of many existing networks[3].

In what follows I propose a partial reconstruction of the main chronological line of theoretical achievements in this emerging discipline. I definitely don't aim here at giving a thorough exposition of all the relevant results, especially the more recent ones, in a field which, albeit young, is characterized by an already cospicuous and fast-growing literature. I will make a survey of the main questions and problems, with the aim of highlighting some properties and limitations which will serve my main considerations and proposals in section 13.

3.1.1 Random and regular networks

Starting in the '60s, models of *random networks* have been studied by mathematicians, with Erdős & Rényi (1960) as the founding paper. Random networks are networks in which the edges between nodes are distributed *at random*. These networks serve as a benchmark against which to compare a given network in order to detect some structure in it: if the network under observation

[2] See Newman (2003). Mathematically, networks are *graphs*: the corresponding theoretical branch is *graph theory*.

[3] Börner et al. (2007), p. 539.

reveals statistical properties which are different from those of a random network, then it possesses some *structure*[4].

During the following two decades, much attention has been posed to random networks as described by the Erdős-Rényi model, and also to *regular networks*: networks in which the degree of connectivity between nodes is roughly *uniform* across all network (for example, as in a lattice of nodes, each of which is connected to all its neighbors).

Random networks and regular networks show fundamentally different statistical properties. Random networks possess a low average path length between any two connected nodes: this means that the number of nodes interposing between any two nodes along the path which connects them is relatively small, on average. However, the average amount of local connectedness[5] between any two nodes is low: many neighborhoods appear only sparsely connected.

By contrast, in regular networks, there is high average local connectedness, but also high mean path length between nodes: this is because any node is directly connected only to its neighbors, so, to reach a given node from a distant one, one has to traverse many intermediate nodes, which are in turn the neighbors of some other node (see fig. 3.1).

3.1.2 Small-world networks

A more recent seminal paper, Watts & Strogatz (1998), highlights the fact that, previously, network models had usually been supposed to possess either a completely regular topology or a completely random one, but that, as models of actual world phenomena, many interesting networks are not expected to be completely regular: they show a *quasi-regular* structure characterized by some amount of disorder. Studying mathematical models of networks with these properties, Watts & Strogatz came to the conclusion that, under appropriate conditions, systems of this kind show a topology which reveals itself as apt to perform computations.

Specifically, in such non-completely regular networks, nodes are densely linked to each other locally, and, in virtue of a few long-range "extra" links between some couples of distant elements, they come to constitute a "small world" (a concept similar to what in pop culture is known as "6 degrees of separation"): inside the system, communication between nodes (or causal influence, or in general some type of efficient relation holding between nodes) is easy, because the extra, longer links, make distant elements appear closer: these longer links decrease the average distance between nodes, while at the same time the amount of local connectedness remains high as in a regular network (fig. 3.2).

This kind of topology combines features which in random or regular networks appear conflicting. Small-world architectures like these seem able to support forms of *computation*, and have been found to occur in extremely diverse real cases: for example, continental-wide electrical power grids, or the nervous system of a simple organism like *Caenorhabditis Elegans*[6].

We will see in section 3.2.6 that the property of being a small-world network is not incompatible with the property of being modular.

[4] See Caldarelli & Catanzaro (2012), pp. 29–30 and Alon (2006), p. 29.

[5] Actually, what I here call "local connectedness" is the *clustering coefficient*. It measures the connectedness of the neighborhoods of a given node, that is the amount of edges between the nodes of the neighborhood. For example, given a person, and two of her friends, if the two friends are also friends of each other, the clustering coefficient is higher than in the case in which the two friends are not friends to each other, but only friends to the person in question. The coefficient measures, in other words, the "cliquishness" of a circle of close nodes. *Cliquishness* is a technical term in graph theory. See Watts & Strogatz (1998), p. 441, fig. 2. See also section 3.2.1.5

[6] *C. Elegans* is a nematode worm which has become a model organism in biology.

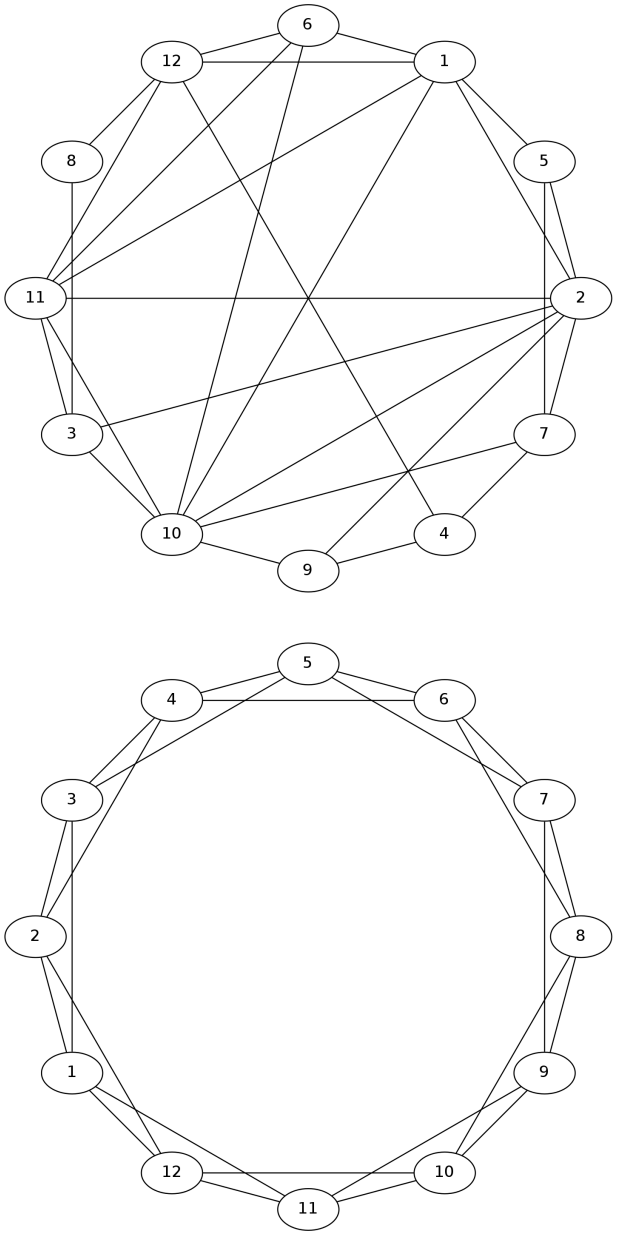

Figure 3.1: a random network (top) and a regular network (bottom).

3.1.3 Scale-free networks

Another feature typical of certain small-world networks is highlighted in a seminal paper by Barabási & Albert (1999), who construct a model for networks of a generic nature which, in contrast to former models, which are mostly static, *grow* with the addition of new vertices according to *preferential attachment*: this is a dynamics of growth of the network's structure in which new nodes tend, with a certain probability, to be connected to the nodes with already high

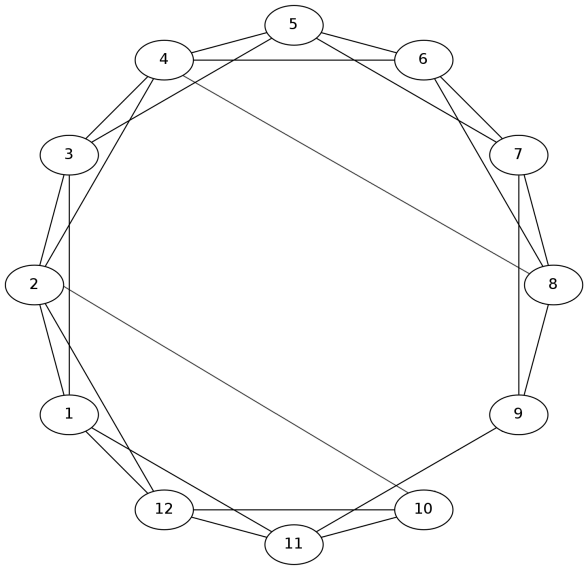

Figure 3.2: a small-world network, which combines short average distance and high local connectedness. In this example, links between node 8 and 4 and between 10 and 2 are the "extra" long range links which allow for the *small-world* phenomenon by non-linearly reducing the average path length, while the rest of the network roughly resembles that of a regular network (see fig. 3.1), retaining a high local connectedness.

number of connections, in what could be called a "rich get richer" schema[7]. With time, some nodes of such networks emerge as *hubs*, that is, poles of attraction which get linked-to by many other nodes. This way, hubs come to assume a *degree*[8] that is much higher than the average degree of nodes in random or regular networks. This can be seen as a global property of such kind of networks, that of being *scale-free*, so called for the reason these networks don't have a typical scale. This means that in such networks there is not a *typical* degree a node can have: degree varies wildly among nodes, and it is not a more or less uniform parameter across the network. In other words, contrary to what holds in regular or in most random networks, there is a certain amount of nodes, the hubs, with a degree much higher than the majority of the other nodes[9].

Scale-free networks are also small-world networks, because they have high local connectedness, due to the fact that many nodes attach to hubs, and small average path length due to the fact that nodes attached to the same hub are at most at 2 edges of distance one to the other[10].

In interesting, *directed* real-world networks, such as many genetic regulatory networks, there are hubs which have a high *out*-degree towards other nodes: these hubs are genes which regulate many other ones. The converse may not hold: each gene is regulated by a small number of other genes. So, the network is scale-free with reference only to its out-degree distribution[11].

[7] This type of preferential attachment is a dynamics typical of some real-world networks, such as the citation network of scientific papers: some papers come to get more cited than others, and keep on being cited preferentially, because their already high number of citation makes them appear more prominent than most other papers.

[8] which is the number of edges linked to a given node.

[9] Technically, *scale-free* means that the distribution of degrees in nodes does not follow a gaussian curve, but a *power law* of the form $p(k) \sim k^{-\gamma}$, where the value of γ stands usually between 2 and 3.

[10] See Amaral, Scala, Barthélémy, & Stanley (2000).

[11] See Caldarelli & Catanzaro (2012), p. 81.

The property of being scale-free endows a network with a certain amount of *tolerance* to failure or disruption of a part of its nodes, albeit in scale-free networks this tolerance is obtained at the cost of increased sensitivity to *targeted* disruption of certain specific parts of the networks, such as the hubs[12]: while targeted destruction of a few hubs would disintegrate the network into disconnected parts, destruction of nodes chosen at random has a high probability of leaving the network global structure mostly intact, as showed in fig. 3.3. This kind of *resilience* is arguably an essential feature of computing or biological systems, such as the nervous systems or genetic networks.

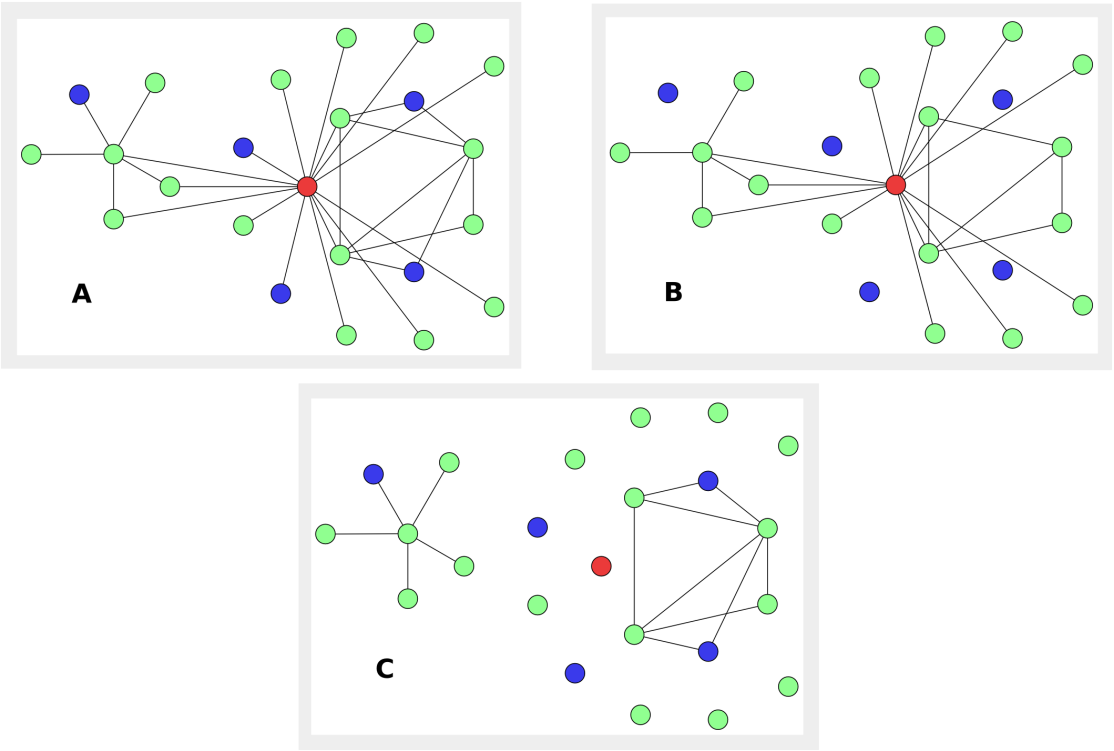

Figure 3.3: resilience of a scale-free network to random attacks. Frame *A*: a scale-free network, with a main hub highlighted in red and five non-hub nodes colored blue. Frame *B*: effect of a random attack which ends up hitting only the five blue non-hub nodes. Frame *C*: effects of an attack specifically targeting the red hub.

3.2 Modularity in networks

Local connectedness and average path length, as well as the property of being scale-free, are *statistical* properties which refer to short or long-range properties of the nodes, or of local aggregates of them, holding, on average, across all the network, but which do no tell much about the *specific structure* of the network under consideration, when applied to networks which are neither completely random nor completely regular: in particular, these properties do not tell us much about if and how the network is structured into distinguishable or even discrete subnetworks. *Modularity* instead is a property which describes the occurrence or absence of this kind of structure in the network.

[12] See Albert, Jeong, & Barabási (2000).

3.2.1 Community and hierarchical structure detection

A terminology which has been used in network science to express a property analogous to the notion of modularity which I exposed in section 6, is that of *community structure*, which is "the division of network nodes into groups within which the network connections are dense, but between which they are sparser"[13]. The *communities* are, in other words, "groups of vertices that have a high density of edges within them, with a lower density of edges between groups"[14]. The concept of community structure has been proposed by Michelle Girvan and Mark Newman, starting from the seminal Girvan & Newman (2002), and subsequently in many other papers.

As can be easily seen, networks endowed with community structure show a property which coincides quite well with a general conception of modularity, as depicted in section 6: the property of having subsets of elements, the *modules*, whose internal elements are more intensely related to each other than to external ones. In this case, the modules are the communities: the whole network presents certain subsets of nodes where nodes belonging to a subset are more densely connected to one another than to nodes belonging to other subsets. The peculiarity here lies in the relation taken into consideration to assess modularity, which in the case of network communities is that of the *density of interconnections* between nodes of the network, while in the general case can be a relation whatsoever between elements of the system. An example of network with a community structure is depicted in fig. 3.4.

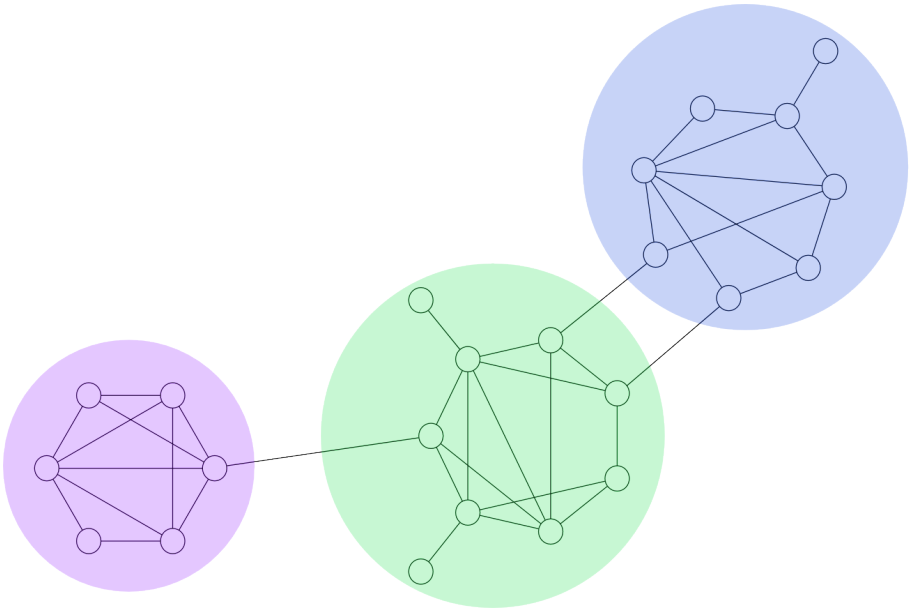

Figure 3.4: a network with community structure of the type proposed by Newman and Girvan. In this picture, colored discs surround the communities, which show high density of intra-module links, while external, inter-module links, are more sparse.

It is certainly interesting, given a network, to assess if it shows any degree of community structure, by means of some method for community detection. For reasonably large networks, automatic detection is undoubtedly required, and several algorithms have been proposed for this purpose,

[13] Newman & Girvan (2004), p. 193.
[14] Newman (2003), p. 193.

starting from the classic algorithms proposed in Girvan & Newman (2002) and Newman & Girvan (2004).

We are not dealing here with detection of only a single set of modules in a given network: in full accord with Herbert Simon's conception of *hierarchical systems*[15], the algorithms devised by Newman and Girvan are able to find a full *hierarchical*, tree-like, structure of communities and subcommunities in a network.[16], an example of which is in fig. 3.5.

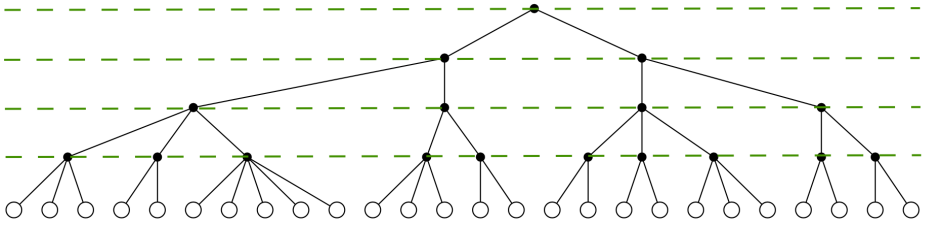

Figure 3.5: a hierarchical structure of communities in a network. White circles are the network's nodes. Black dots represent communities, higher-level modules composed of sets of nodes or of other communities, in turn. Green dashed lines represent "levels" in the hierarchy. The highest one locates the level whose community is the whole network.

Newman and Girvan's algorithms start by identifying the highest, *coarser* level[17] of the hierarchy, composed of a single community coinciding with the whole network, to proceed detecting lower, progressively *finer* levels, composed of progressively smaller communities. The lowest, finest level, is obviously composed of only single-node communities: as highlighted in section 6, a single atomic element has intuitively all the properties of a module, and can be considered as such. Thus, the lowest level, that of single-node communities, can be considered a *modular* level as well. Together with the highest level, constituted by a single module comprising by the whole network, these two levels can be considered what I would call the *trivial hierarchical levels*, always present in any modular system, and they constitute a flat hierarchy[18]. The number of detected hierarchical levels depends on a choice about when to stop the detection: potentially, all the possible hierarchical levels can be detected this way, but the algorithm can be stopped at a certain point of its processing, yielding a partial hierarchical description which focuses on a certain set of modular levels, while ignoring other finer hierarchical levels.

3.2.1.1 The Modularity measure Q

There is a problem with the methods by Newman and Girvan mentioned above, which has been soon acknowledged by the authors: the algorithms proceed by proposing a succession of

[15] See section 2.3.

[16] Historically, other methods for detecting so-called *hierarchical clustering* in data had formerly been developed, especially by sociologists for social networks, but the method by Girvan and Newman is the first to be presented as applicable to networks in general, and works on different principles than those on which the former methods are based: specifically, while cluster detection in sociology usually works by starting with a set on unrelated nodes and by proceeding to progressively add links to them (a so-called *agglomerative* method), the algorithm by Girvan and Newman proceeds by progressively *subtracting* links from the complete network (a *divisive* method). This method overcomes some drawbacks of the agglomerative ones, which, for example, tend to see nodes which are peripheral to a cluster as *single* nodes, each constituting a module, while in actuality these nodes can be considered as naturally belonging to the cluster. See Newman & Girvan (2004), pp. 1–2.

[17] I'm using the expression "level" only intuitively here, while I will try to give it a more thorough analysis in section 6.6.

[18] See also section 6.6

progressively finer splits of the network (the hierarchical "levels") into progressively smaller communities, but they do not give any indication to *which* of these splits are most representative of an actual, sensible hierarchical structure of the network[19], given that the algorithm, per se, would produce some division of the network anyway, even when applied to random networks.

To avoid this drawback, starting from Newman & Girvan (2004), these authors propose a measure of the degree of modularity of a network, which they call Q: this is the quality measure on which their evaluation of modularity detection is based.

The Q measure has more or less become the standard of reference in network modularity detection. Ranging continuously from 0 to 1, Q measures the probability that a candidate, alleged community structure, already detected by an algorithm (be it the algorithm by Newman and Girvan, or another one) is or is not *actually* present in the network. This probability is measured by comparing the density of intra-module links in the network under observation (links between nodes belonging to the same community) to the expected density of links between those same nodes, but in a random network[20], that is, a network which has been purposely constructed by completely reconnecting the nodes of the original network, but this time *randomly*[21]. The tacit assumption is that random networks should not display, usually, a significant community structure, that is, they should not be expected to display a significantly higher density of links in some subregions than in others. So, the more the amount of intra-module links in a given modular description of a network proposed by the algorithm is higher than the same amount in the corresponding random network, and thus the Q value is high, the more this candidate modular description reflects a genuine community structure present in the actual network under analysis. In other words, a modular description with a high Q is a *good* modular description, which more reliably reflects the real modular structure of the observed network. Q measures the probability that a community structure detected in the network (not necessarily by Newman and Girvan's algorithms, but also by other means) is not an *apparent* feature of a network: this is the case when the network, although appearing in some way structured, does not in actuality differ significantly from a random network. Q is a necessary measure, given that most algorithms for community detection, because of their internal logic, tend to find *some* community structure even in what are known to be random networks.

Given a modular description of a network, Q is a measure of the quality of this candidate modular description of a network. But from another point of view, we could consider, given a network, the highest possible value of Q for that network, which is the value of Q for the best possible modular description of that network, that is, for the modular description, among all the possible ones, which best reflects a significant modularity present in the network. If we consider this virtual maximum value of Q for a given network, this value becomes a measure of the *amount* of actual community structure in a network: that is, the probability that the given network is different from a random network in its distribution of links between nodes: if there is community structure in the network under observation, then the average density of links between its nodes should significantly vary in different parts of the network.

[19] See Newman (2004a), p. 6. and Newman & Girvan (2004), p. 7.

[20] See section 3.1.1.

[21] In the words of its proponents, the quantity Q "[...] measures the fraction of the edges in the network that connect vertices of the same type (i.e., within-community edges) minus the expected value of the same quantity in a network with the same community divisions but random connections between the vertices. If the number of within-community edges is no better than random, we will get $Q = 0$. Values approaching $Q = 1$, which is the maximum, indicate networks with strong community structure. (newman:2004evaluating, p. 3. The paper also includes a formal definition of the Q measure.)

It seems clear that, in a way, being Q based on how much the communities are internally more densely interconnected than how an average random network is connected, or, in other words, how significantly higher is the density of intra-community connections with respect to the inter-community ones[22], this measure reflects quite well a criterion similar to that used in Herbert Simon's original near-decomposability notion[23], but applied to the structure of *non-weighted* networks, where near-decomposability of the network into communities is allowed by low *density of inter-module connections*. In my view, this corroborates the idea that Newman's and Girvan's Q is an acceptable measure of the general amount of modularity possessed by a network, and so that the quality assessment of a given modular representation of a network can be quite *plausibly* based on Q[24].

Thus, Q actually helps quite well in identifying the hierarchical structure best describing a given network: as said above, when trying to detect a hierarchical modular structure in a network, Newman and Girvan's algorithms start by detecting the presumed highest hierarchical level of modularity, to then proceed downward, in order to discover finer and finer hierarchical levels. The modularity measure Q is employed during this process, to assess the plausibility of the detected modularity at each hierarchical level: the hierarchical description which best matches the actual (if there is any) hierarchical structure of the network under observation, is that whose hierarchical levels coincide with *peaks* (local maximums) of the Q measure. This is exemplified in fig. 3.6, which depicts the hierarchical tree of a social network obtained by the algorithm in Newman & Girvan (2004), where the modular levels in the hierarchy in correspondence of which Q reaches local peaks are highlighted with dotted red lines: these are the two topmost levels. As Q goes progressively down after the second peak, the other, finer levels are supposed to be less and less representative of an actual, substantial modularity present in the original network.

Besides those by Newman and Girvan, many other algorithms for community detection have been proposed, and I will touch upon some of them in the following sections. A useful, albeit partial, survey and performance comparison of such algorithmic approaches is Danon et al. (2005).

3.2.1.2 Reliability of the detected modular structure and computational hardness of Q optimization

As we have seen, a typical problem with hierarchical structure detection is that of the reliability of the hierarchy obtained by means of some algorithm: how can we be sure that this modular structure reflects some modularity actually present in the network under observation, and is not simply an *apparent* modular structure imposed on a network which, per se, is not modular, or which has actually a different hierarchical structure?

As mentioned in the previous section, a modularity measure Q has been adopted for the assessment of the quality of the modular structure detected in a network: plausible modular levels in the hierarchy are those at which Q reaches a peak, which means that that level of modular description reflects quite plausibly an actual modular structure in the network. This gives, in a way, a form of guarantee that the detected modularity reflects an actual organization present in the newtwork. Nevertheless, a problem is raised by the fact that most classic algorithms for hierarchy detection, like that of Girvan & Newman (2002), directly produce only *a single* hierarchy, dependent on the specific algorithm they use, completely neglecting to produce any possible

[22] The latter being on average equal to the expected density of links in a random network, taken as reference.
[23] See section 2.2.3.
[24] Actually, Q can be in certain circumstances problematic. For example, it can assume negative values when in a network each node is considered as a module. For such reasons, in some cases alternative measures have beeen proposed. See for example Danon, Díaz-Guilera, Duch, & Arenas (2005), pp. 4–5 and Massen & Doye (2005), cited in Danon et al. (2005), p. 3.

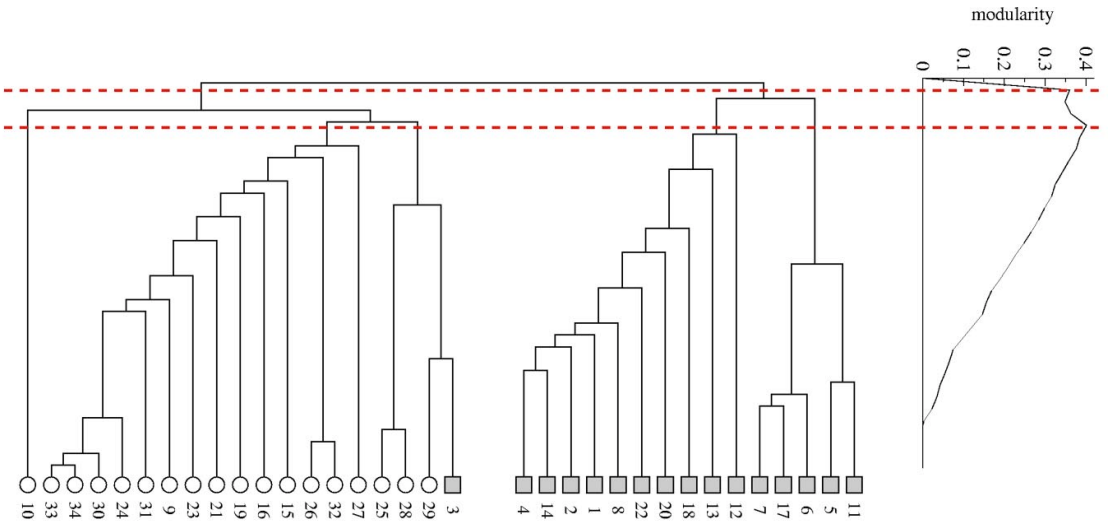

Figure 3.6: the hierarchical tree of a social network obtained by Newman and Girvan's algorithm (the topmost, trivial level comprising the whole network is not shown in the picture). The two dotted red lines highlight the two modular levels in correspondence of which the modularity measure Q (graph on the right) reaches local peaks. Image adapted from Newman & Girvan (2004).

alternative hierarchical organization in the network under observation, which could present a better, higher Q value at every level: it is a fact that, by changing the *metric* for modularity, that is, the measure according to which two nodes can be considered more or less strongly connected, or by changing the algorithm for community detection, completely different hierarchies can come out: compare for example fig. 3.6 with fig. 3.7: the two hierarchies are produced by different algorithms applied to the same social network. Unlike the first, the second, low-quality algorithm[25] detects a hierarchical modular structure which is not plausibly present int the actual network. In this case Q is, on average, much lower than in the case of fig. 3.6.

So, while, *given a certain algorithm* for community detection, the best hierarchical levels *which that algorithm can detect* are those that *locally* maximize Q, in order to find the best *possible* hierarchical representation of the observed system, we should instead choose, among all the *possible* hierarchical structures detected *by every possible algorithm* for community detection, the one that on average maximizes the value of Q. That is, while when keeping fixed the algorithm for community detection we only need to search for local maximums of Q in order to chose the best modular representations that that specific algorithm can supply, when we want the best possible hierarchy ever, we need to find the one which makes the Q value reach not its local, but its *average* maximum. This process is called *optimization of* Q. It implies that any possible hierarchical description be evaluated, and that the one with the Q curve maintaining on average the highest values be chosen.

As expected given its nature, it recently turned out, as proved by Brandes et al. (2008), that the task of optimizing Q is an *NP-complete* task, and thus that it is quite certainly computationally intractable[26].

[25] See Newman & Girvan (2004), p. 9.

[26] I will treat the computational complexity of this task and of modularity detection algorithms in section 3.3.1. For the notion of computational complexity in general, see sections 17.4 and 17.4.3.1.

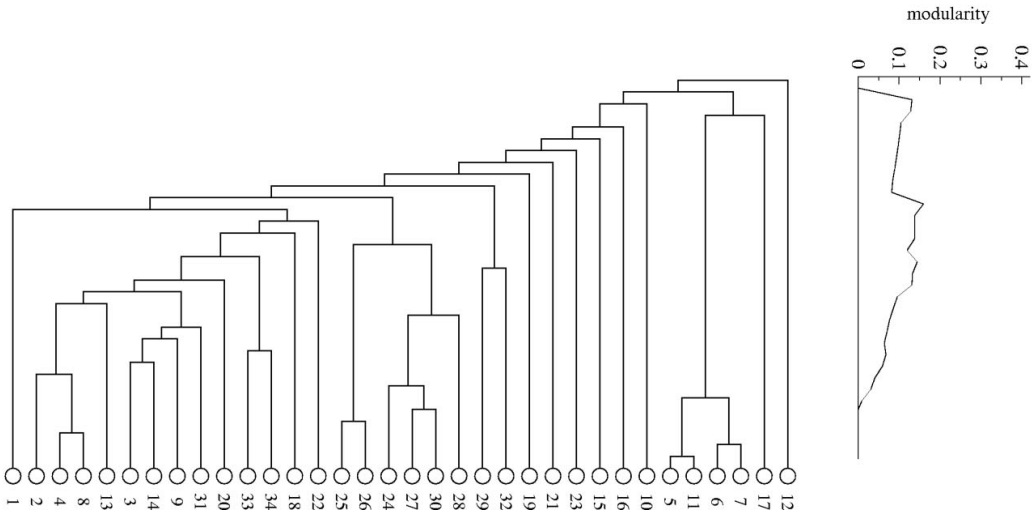

Figure 3.7: the hierarchical tree of a social network obtained by Newman and Girvan's with a purposely chosen non optimal algorithm. The graph on the right shows the modularity measure Q. Image adapted from Newman & Girvan (2004).

This means that the task of producing the *best* modular hierarchical representation of a given network, that is, the task of determining the hierarchical modular description which best represents the *genuine* modular structure of the network, is too computationally intensive to be feasible, because it would involve finding the hierarchical description which optimizes Q, and this is a NP-complete problem. So, it is not to be expected that the *optimal* modular description of networks larger than a few nodes can be found. This in turn calls for research of approximate methods.

Actually, most of the algorithms which work by non-optimizing Q, starting with the ones in Girvan & Newman (2002) and Newman & Girvan (2004), are actually, in a sense, a way to obtain an approximate solution to the problem of detecting the actual hierarchical structure of a network. But it would be interesting to try to approximate Q optimization itself, or to find alternatives to the use of this measure of modularity detection quality.

Questions of this kind have been raised in numerous recent papers, and some of them have put forth solutions based on quality measures different from the classic modularity measure Q.

A method which directly tries to optimize Q in an approximate manner[27] is the one proposed by Newman (2004b), which is in $O(n^2)$, and as such quite fast. A more refined version of the algorithm, put forth by the same Mark Newman, with Aaron Clauset and Cristopher Moore and other authors in Clauset, Newman, & Moore (2004), runs even faster, in $O(md \log n)$, where m is the number of edges, n the number of nodes and d the number of levels in the full hierarchy detected. With some plausible assumption about certain structural properties which frequently occur in networks which are typical candidate for examination, the algorithm can be considered as running in $O(n \log^2 n)$. This is essentially a *linear* running time, and, as the authors highlight, this constitutes quite a breakthrough in performance, allowing for the detection of modularity in networks with millions of nodes. However, these two methods work by considering only local

[27] Approximate otimization is performed by following a *greedy* algorithm. A greedy algorithm accepts at any given time the *local* optimum. This heuristic is based on the hope that, this way, the global optimum will sooner or later be found among these local ones. Of course this is not guaranteed at all.

information on individual communities, and lose in precision against the more classic, but way slower, method of Newman & Girvan (2004), which makes its detection by basing on non-local information: as highlighted in Newman (2004b) p.3, community structure is a nonlocal quantity, or at least this holds for the type of community structure detected by Newman and Girvan's algorithm. This stems from the fact that their algorithm detects community structure of the basis of the *edge betweenness* measure, a measure which gives also information about distant, and thus non-local, connected nodes[28]. As we will see in section 3.3.1.2, this trade-off between accuracy and speed is often present in the choice of an algorithm for community detection.

Clauset, Moore, & Newman (2007) proposed solution is an algorithm which performs the exploration of a *statistical sample* of the space of every possible hierarchical structure which could be present in a given network, and which proceeds, by means of a bayesian inference, to identify the hierarchical structure which best fits the given network. A quite similar, slightly more sophisticated approach to extract a sensible full hierarchical structure from a network has been put forth in Sales-Pardo, Guimerà, Moreira, & Amaral (2007).

Algorithms based on simulated annealing, which is an optimization procedure based on stochastic sampling[29], are proposed in Guimerà, Sales-Pardo, & Amaral (2004), Guimerà & Amaral (2005a) and a similar approach in Massen & Doye (2005). They work by employing simulated annealing to approximately optimize the value of the Q modularity quality of candidate modular descriptions. Community structure detection by means of such algorithms ends up being very accurate across a range of typical networks. The downside is that the time complexity is quite high[30]

Fortunato, Latora, & Marchiori (2004) propose a method fro community detection based not on the same metric employed by Newman and Girvan, that is edge betweenness, but on a measure, related to the idealized flow of information inside a network, called *information centrality*. This measure as a high value for edges connecting different modules and a low value for intra-module edges. Fortunato and colleagues claim that their method, although slower, is of precision comparable to that of the best method by Newman and Girvan.

Another algorithmic method has been suggested by Lancichinetti, Fortunato, & Kertész (2009). It is an interesting method for, as the authors highlight, it was, at the time of its proposal, the first family of algorithms for the simultaneous detection of the hierarchical *and* of the *overlapping* community structure of a network. Moreover, they claim that the method can be easily extended to *weighted* and *directed* networks. I touch upon the possibility of detecting overlapping communities in section 3.2.1.4, and on the question of weighted and directed networks in section 3.2.7. What interests us here is the fact that this method is supposed to reliably find a *genuine* hierarchical structure in observed networks. It works by varying a continuous parameter, called α, which specifies the grain of the hierarchical level to put under observation. In other words, α determines the *scale* at which modularity is searched for in the observed network, by determining the size of the communities which are to be taken into consideration: the larger the

[28] About the notion of edge betweenness: formerly in the text, to avoid too much technicalities, I slightly simplified the matter by stating that, in Newman and colleagues' papers, it is the property of *density of interconnections* to be taken into consideration, in order to detect modularity. Actually, they employ the similar but not identical property of *edge betweenness*: this is, roughly, a measure of how many nodes a given edge directly of indirectly gets to connect. In the authors' words, edge betweenness is "some measure that *favors edges that lie between communities* and disfavors those that lie inside communities" (Newman & Girvan, 2004, p. 3) Thus, in a sense, this measure is the *opposite* of the modules' internal connectedness, but this is justified by the "divisive" method Newman and Girvan algorithms' employ. For this and other technicalities, consult the original paper.

[29] See Kirkpatrick (1984).

[30] See section 3.3.1.

value of α, the smaller the communities possibly detected[31]. By tuning α to different intermediate values, by way of this "zooming" in and out, the algorithm can produce different-grained candidates for the levels of a possible hierarchy describing the network under observation. The paper proposes a way to pinpoint the most "natural" ones among these candidate levels, that is, the levels which would come to compose a modular hierarchical structure reflecting some kind of *genuine* modularity actually present in the network. The basic idea is that a detected candidate modular description reflects genuine modularity in the network if its detection is *stable* against variations in the value of the α parameter, that is, if the same exact candidate modular description gets detected within a large enough *range* of α values, and not only at a single precise value of the parameter. In other words, a significant hierarchical level community structure should not "vanish" only by slightly zooming in or out of it. This seems plausible, but it is based on the assumption that modularity is a quite "local" property, as acknowledged by the authors, and that this assumption is plausible is a debatable question, for, as we have seen, according to some, like Mark Newman, community structure is *non-local*.

A different concern is that the detected modularity and/or hierarchical structure, being based on a modularity measure, could reveal itself not as a *significant* feature of the observed network, but simply as the effect of *chance*: as highlighted in Karrer, Levina, & Newman (2008), it has been showed that in sufficiently large *random* networks, the likelihood that they are susceptible to a highly modular description rises[32]. Karrer et al. (2008) addresses this problem by proposing a new definition of the quality of a modular description, based not on Q, but on the *robustness* of the alleged modular structure to small perturbations of the network structure: a modular partition of the network is not simply due to chance when it persists while the network structure is gradually altered by adding or removing links. This method seems to reflect quite directly the idea, already highlighted in section 6, that a structure, to be considered a module, must possess some form of *robustness* to perturbations. Karrer and associates give an algorithm for assessing the *significance* of the community structure detected in networks by using this method. This is especially useful when trying to assess modularity in networks representing data obtained by empirical observation and measurement, where there's a high likelyhood of having obtained data contaminated by noise: some nodes of the network could be the spurious result of noise. If a detected network modularity shows low robustness to perturbations, it could be argued that it is too sensitive to noise, and thus could end up not representing faithfully enough an actual modularity present in the observed phenomenon.

3.2.1.3 Modularity detection in weighted networks

The main difference between the typical approach to modularity in networks and the general conception of modularity, is that, quite often, the relation taken into consideration for modularity detection in networks is *density of connections between nodes*, a relation according to which modules amount to communities of *densely* interconnected nodes, with sparser connections between different communities. In this case, the *strength* of a connection is considered to be an *all-or-nothing* property: a connection can only be either present *or* absent between two nodes. In other words, only *non weighted* networks are treated. By contrast, a more general conception of modularity in networks should concede to consider, for evaluation of modularity, any possible

[31] As obvious, extreme values of α yield the trivial hierarchical levels: a large enough α value yields a modular structure composed of each node as a module, while a small enough value makes the algorithm view the entire network as a single module.

[32] This is due to the fact that the number of possible subdivisions of a network rise extremely fast as a function of the number of network nodes, so, however unlikely it can be, a subdivisions for which the measure of modularity is high is actually likely to appear, sooner or later, in the vastness of the space of possible subdivisions, given a sufficiently large random network. See Karrer et al. (2008).

relation between nodes, even continuous-valued relations, and modularity detection should possibly be based not only on the *density* of non-weighted connections, but also on their individual *strength*.

The oversimplification of network models effected by considering them as non-weighted networks is often done on purpose, for the reason that in many cases the oversimplified network model still allows for sufficiently good *qualitative* analysis, prediction or explanation of the original phenomenon, while being more computationally tractable than a more realistic model.

Thus, although there are cases in which weighted models are taken into consideration, often discrete models are prevalent. Indeed, the pioneer papers about community structure detection, Girvan & Newman (2002) and Newman & Girvan (2004), focus on networks with *undirected*, *unweighted* edges and, in order to asses community structure, they consider a property of edges they call edge betweenness, which does not consider the weights of edges, but only the presence or absence of edges between nodes.

However, algorithms for community detection in *weighted* networks have been put forth, starting with the seminal Newman (2004a). In this paper, a proposal is made to consider weighted networks as non-weighted *multigraphs*, that is networks in which *more* than one edge can connect the *same* two nodes. The basic idea is that a link with a given integer weight W between two nodes of the network can be represented on the multigraph as W different non-weighted links between the same two nodes (a simple extension allows for an analogous representation of non integer values). This way, the *weight of an edge* between two nodes in the network gets immediately converted to *density of edges* between the same nodes in the multigraph. For an example, see fig. 3.8.

The interesting point in Newman's idea is that once a weighted network gets converted into a *non-weighted* multigraph, and Newman and Girvan's[33], well-known algorithms for community detection can be applied, with minimal modification, *to this multigraph*, since it is, in a way, a form of non-weighted network. Thus, the scope of applicability of the algorithms classically used for hierarchical modularity detection in non-weighted network is extended to the weighted network case.

A quite thorough, completely different method for community detection in weighted networks has been proposed and progressively refined by Haijun Zhou in a series of papers starting in 2003[34], a work which culminated in Zhou & Lipowsky (2004). The starting idea, introduced around 2000 by other authors[35], is that of a virtual particle stochastically jumping from node n_1 to a linked node n_2 of the network, with probability proportional to the weight of the link connecting n_1 and n_2 and to the number of common nearest neighbor nodes of the two nodes, in what is called, in analogy with the physical phenomenon, *network brownian motion*. This way, the particle measures a kind of distance between two nodes, which is inversely related to the weight of the link connecting them and to the density of the neighborhood to which the nodes belong. This metric can be used to distinguish communities as subnets whose internal links are "shorter" than links connecting to external communities. Zhou's method is agglomerative[36], that is, it starts by considering the network as constituted of one-node wide modules, and proceeds aggregating progressively more and more nodes into modules. The choice on which nodes to aggregate is based on a value which reflects the global structure of the network, which the authors call the *dissimilarity index* of each link coupling interconnected nodes, measured according to the above

[33] See 3.2.1 and following sections.
[34] Zhou (2003a), Zhou (2003b).
[35] See references in Zhou & Lipowsky (2004).
[36] while the classic method by Girvan and Newman is *divisive*, see section 3.2.1.

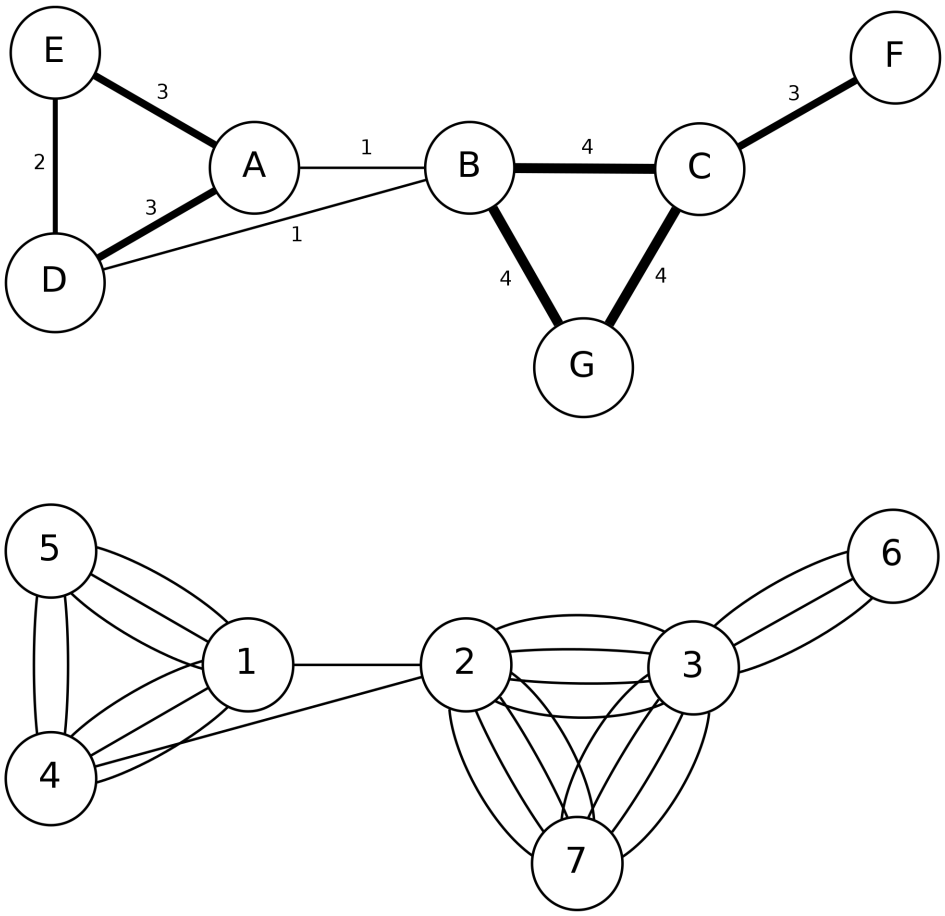

Figure 3.8: top: a weighted network. Number along edges represent the weights. Bottom: the same network represented as a non-weighted multigraph.

mentioned form of virtual distance between nodes measured by the brownian motion of the particle. The dissimilarity index of *communities* of nodes is the average of the dissimilarity indexes of the links connecting its internal nodes. Aggregation proceeds by merging at each step the two less dissimilar communities[37], obtaining this way a dendrogram which represents the hierarchical modular structure of the network. The dendrogram is then modified in order to try to optimize the Q modularity value. Each detected module is finally tested for *robustness* against perturbation of the network structure, and is assigned a robustness value, *integrity*, together with another value, *affinity* which measures the strength of the intra-module interaction with respect to inter-module interaction. This robustness evaluation is important, for it allows to put confidence in the actual presence of the detected community structure in the network under observation. The algorithm, which, according to the author, outperforms Girvan and Newman's classic method, has a time complexity of $O(n^3)$ (see section 3.3.1).

Other algorithms for modularity detection in weighted networks have been devised, such as the

[37] For the first step, each single node is seen as a community.

ones in Arenas, Fernández, Fortunato, & Gómez (2008), Brandes et al. (2008), and Lancichinetti et al. (2009).[38].

3.2.1.4 The problem of overlapping communities

A limitation of classic community detection methods is their incapacity to detect *overlapping* communities. As highlighted by Palla, Derényi, Farkas, & Vicsek (2005), many actual real-world networks are composed of overlapping communities, where a node can be a member of more than one community: for example, in social networks, the same person can belong to different family, work and friendship communities, which thus partly overlap.

Many typical algorithms for community detection are capable of revealing only *non-overlapping* communities. This limitation affects for example Newman's and Girvan's algorithms, and most of the methods surveyed in Danon et al. (2005). More recently, there have been numerous attempts to overcome this restriction, and some promising results have been achieved, such as the pioneering proposal of Palla et al. (2005). The algorithm described there is able to detect overlapping communities, but it has the downside of running in *exponential time*[39].

A family of algorithms capable of revealing in a more reasonable time a full hierarchy of overlapping communities is proposed in Lancichinetti et al. (2009): this is a method which I already described in section 3.2.1.2[40]. The general algorithm runs in $O(n^2 \log n)$, and, as said, according to the authors, it is capable of being easily extended to weighted networks and to directed ones[41].

3.2.1.5 community structure and scale-free networks

As highlighted in Ravasz, Somera, Mongru, Oltvai, & Barabási (2002), scale-free networks, which grow according to preferential attachment in a stochastic manner, should not in general manifest community structure, because, as the network grows, hubs come to connect many other nodes in a way that renders the existence of highly separated communities problematic. To be more precise, we note that Guimerà et al. (2004) observed a relation between modularity in typical scale-free networks and the parameters of the growth by preferential attachment through which they are generated: they tested the amount of modularity in the networks so generated[42], and found that, for networks of fixed size, modularity decreases linearly with the number of links preferentially attached to nodes at each step of the network's growth. in other words, the higher the number of nodes connected to major hubs in the scale-free network, the more connected are all parts of the network, and the less likely it is that modules (in the form of communities), which require isolation, could emerge.

Ravasz et al. (2002) note a tension between the fact that scale-free networks are not in general expected to show modularity and a host of empirical findings which show that biological complex networks, such as metabolic networks, are scale-free networks and and the same time show structural and functional modularity. Along these lines, Ravasz, Barabási and colleagues explored the possibility for scale-free networks to also be modular. They posed their attention on the *clustering coefficient*, which is, intuitively, a measure of the amount of interconnectedness between

[38] See sections 3.2.1.2, 3.3.1 and 3.3.2.

[39] I touch the question of computational complexity of community detection in section 3.3.1.

[40] As noted by the authors, the concept itself of overlapping communities seems at odd with the idea of fitting them in a hierarchical organization. They respond by giving a slightly relaxed definition of hierarchy. See Lancichinetti et al. (2009), p. 6.

[41] For the relation between non-weighted and weighted networks, see section 3.2.7.

[42] They measured the amount of modularity by assessing the best possible Q, optimized by simulated annealing. See section 3.3.1.

the nodes belonging to the neighborhood of a given node: if all neighboring nodes of a given node are connected to each other, then the clustering coefficient is 1, and it goes toward zero as interconnectedness between neighboring nodes decreases. Ravasz and Barabási consider $C(N)$ the average value of the clustering coefficient in a network composed of N nodes a global measure of the network's potential modularity. As shown by Albert and Barbàsi, in general the average clustering coefficient value in scale-free networks is quite small, and decreases with the increase of the network size[43]. On the contrary, Ravasz et al. (2002), studying the metabolic networks of 43 different organism, observed that in all these cases the average clustering coefficient is quite high and remains stable as the network's size increases. Nevertheless, Jeong, Tombor, Albert, Oltvai, & Barabási (2000) and Wagner & Fell (2001) found that in these biological networks degree distribution follows a power-law, and thus they are scale-free networks. To solve this paradox, Ravasz, Barabási and co-workers found a method[44] to artificially create scale-free networks which are also for certain hierarchically modular, and that thus show at the same time high clustering coefficient and power-law distribution degree. An exemplification of this method is reported in fig. 3.9: the scale-free network is obtained by iteratively copying certain basic patterns of connected nodes, as explained in the image caption. The result is a quasi-fractal, self-similar, hierarchically modular structure which, while growing by a process which is different from the preferential attachment process typical of scale-free networks, nevertheless turns out to exhibit the characteristic power-law distribution of node degrees, and thus is actually a scale-free network[45]. At the same time the network is hierarchically modular, and, going up the hierarchy, modules at each hierarchical level are progressively less connected internally, that is, their clustering coefficient lowers progressively. It appears that in networks of such a kind there is a particular distribution of the clustering coefficient C which ends up being a form of *signature* for this kinds of networks: C is roughly inversely proportional to the degree of a node[46]. In other terms, nodes connected to highly "popular" hubs are less connected to each other than nodes connected to less important hubs: this is obvious, given the way in which the network has been constructed: highly popular hubs are the central nodes of high-level modules, and such nodes connect different lower-level modules, where the nodes internal to these lower-level modules are not directly connected to nodes internal to other, different modules at the same hierarchical level. This situation is compatible with a form of modularity where important hubs connect *different* communities, a properties of hubs which can also be interpreted in terms of *roles* they fulfill, in this case "connector" roles (as we will see in section 3.2.3). Ravasz et al. (2002) observe a distribution of the clustering coefficient C in the metabolic networks of the 43 organism they consider which is very similar to the inverse proportionality to the degree of a node found in the artificially constructed networks exposed above, a fact which leads these authors to conclude that the observed biological networks show also roughly the same type of scale-free, hierarchical modular structure these constructed networks possess. By closely analyzing the modular structure of the metabolic network of *Escherichia coli*, they also conclude that, at least in this organism, the detectable hierarchical structural modularity is strongly correlated with the functional modularity of its metabolic process, although the coincidence is not perfect: certain protein synthesis processes cross boundaries of the found structural communities. The authors also speculate, on the basis of observational data reported in the literature, that the kind of modular hierarchical structure they proposed is present in the majority of biological network, and that the construction by multiplication of copies of preexisting modules could be implemented

[43] See Albert & Barabási (2002), p. 75.
[44] Exposed in Ravasz et al. (2002) and Ravasz & Barabási (2003).
[45] While in the example the network is generated by a deterministic iterative process, Ravasz et al. (2002) also propose a stochastic generative method, which gives similar results in network properties.
[46] $C(k) \sim k^{-1}$ where k is the degree of a given node.

in real biological systems by the fact that evolution often proceeds by gene duplication. I will treat these and related questions in section 7.

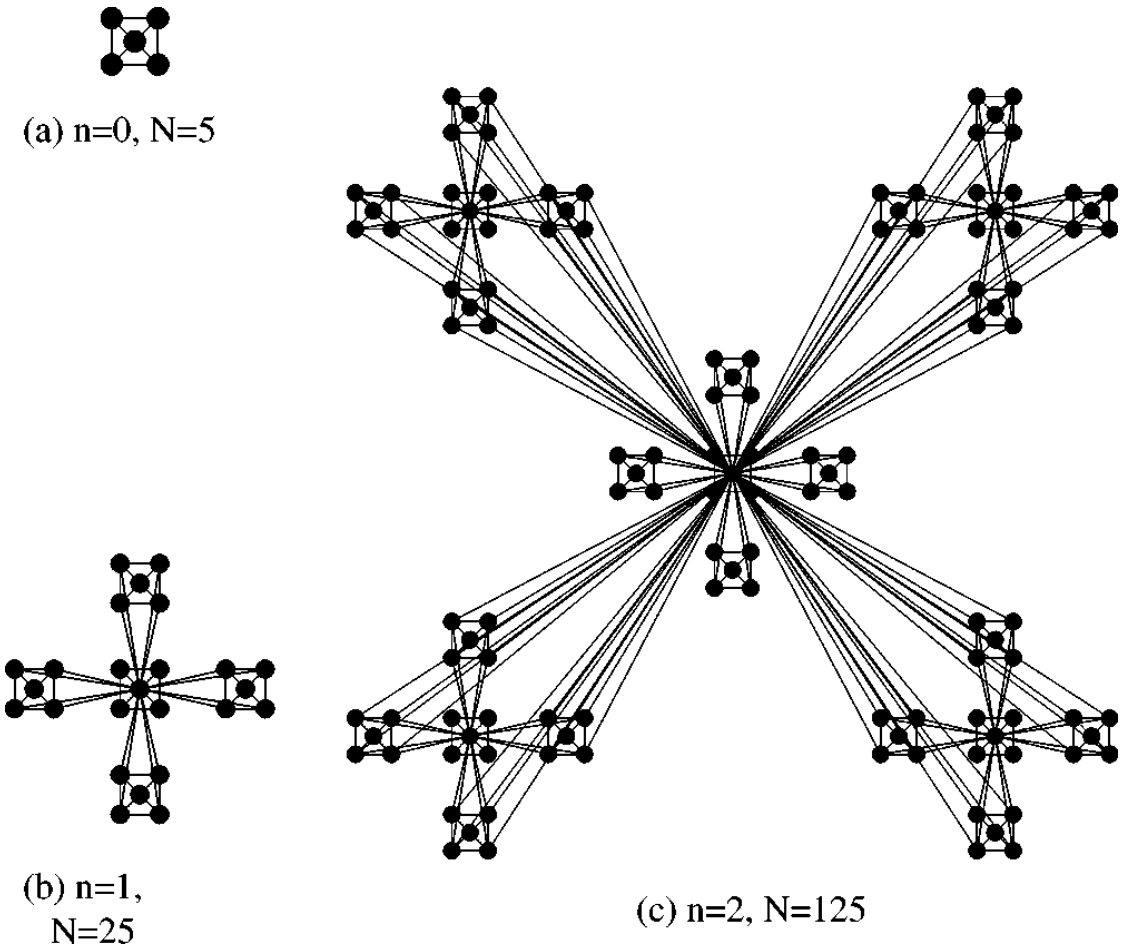

Figure 3.9: exemplification of the method, devised by Ravasz and Barabási, for the construction of a network which is both scale-free and hierarchically modular. We start (image *a*) with a basic, internally highly connected, module composed of five nodes. The second step (image *b*) consists in producing four copies of this module and to connect each peripheral node of each copy to the central node of the original module: the composite pattern so obtained can be considered a module at a higher hierarchical level. In *c* the process gets iterated to an even "higher level", by copying the pattern obtained in *b* and by connecting peripheral nodes of each copy to the central node of the *b* pattern. The process can be repeated indefinitely. This way, a network which is at the same time scale-free and hierarchically modular is obtained. Such a network has a clustering coefficient following a distribution which is about inversely proportional to the degree of a node, and each modular hierarchical level is composed of progressively less connected modules, that is, modules with a progressively lower clustering coefficient. (Image taken from Ravasz & Barabási 2003).

3.2.2 Network motifs and network themes

Another property of modular systems, distinct from near-decomposability although perfectly compatible with it, is the property of comprising repeated similar elements[47]. For a network,

[47] See section 6.

this property is that of showing *network motifs*, that is, subgraphs that recur more than one time in different parts of the network, and whose presence in the network is not due to chance, that is is not due to the network's random structure.

The concept of *network motif* has been introduced in two seminal research papers, partly by the same authors: Uri Alon, Shai Shen-Orr and Ron Milo of the Weizmann Institute of Science. The first paper, Shen-Orr, Milo, Mangan, & Alon (2002), is explicitly inspired by a former work, Hartwell et al. (1999)[48], which advocates the search for basic building blocks of biological systems in the form of recurring functional elementary subsystems, such as negative or positive feedback loops, or amplifiers, or similar simple functions. Shen-Orr et al. (2002) is the first work to introduce the term *network motif* to denote a such a kind of elementary module recurring in gene regulation networks. A subsequent paper, Milo et al. (2002), generalizes the former work to any *directed* network and puts the notion of network motif to test against several types of networks, from biochemistry, to ecology, to neurobiology.

Network motifs are defined as "patterns of interconnections that recur in many different parts of a network at frequencies much higher than those found in randomized networks"[49].

Detection of network motifs is performed by taking, for each integer n[50] all possible n-node subgraphs of the network and by comparing the number of their occurrences in the network under observation to the number of the occurrences of the same subgraphs in a sample of random networks: the types of subgraphs which occur significantly more often in the observed network with respect to their occurrence in the sample of random networks, are considered network motifs (we see here a criterion somewhat similar to that employed in the Q measure for evaluating the quality of a proposed community structure, described in section 3.2.1.1).

Network motifs are simple types of structural modules which recur more than one time in the network. In directed networks, the directional nature of edges increases the odds that network motifs identified by purely structural methods like the one cited above are actually *functional* units in the network.

Motifs realize the property of modularity which consists in the possibility of reducing the amount of information needed to specify the whole network: by reducing the specification of the entire network to a description of the repeated occurrences of the same simple types of building blocks and their interrelations (in addition to interrelations to other building blocks which do not recur, but appear only once), a *lossless compression of information* is achieved, with respect to a verbatim description of the network which does not distinguish recurring modules: in the first case, the internal structure of each network motif type is required to be described only *once*, regardless of the number of its occurrences in the network. In the second, *every* occurrence of the same substructure must be described in its internal details.

Network motifs can also be seen as *functional* basic "building blocks" of more complex functionality: typical examples of network motifs found in biological networks are *feed-forward* loops and *feedback* loops (see fig. 3.10), oscillators, or also elementary graphs performing simple *boolean functions*[51]. In other words, as Levy & Bechtel (2013) highlights, even if in their formal definition network motifs are seen as *structural* patterns, the real interest behind their introduction

[48] See section 7.

[49] Shen-Orr et al. (2002), p. 64.

[50] Usually, in these studies, n does not exceed 5, for reasons which will be elucidated in what follows.

[51] It must be noted that, quite often, research on genetic or other biological networks is conducted on *boolean* simplified models of the actual genetic network, which in itself is not boolean, but admits of intermediate values of regulation. Usually, the boolean approximation is considered sufficient to assess some qualitative functional properties of the network. See Alon (2006).

lies in seeing them as performing functions (typically, biological functions): they can be seen as functions in the sense[52] that they play a *role* in bringing on the whole functioning of the dynamics occurring on the network. Along these lines it seems then plausible[53] to consider network motifs as elementary computational functions which can come to constitute building blocks of more complex computational networks. Computation-capable networks, understood in a general sense, are certainly ubiquitous in the living world: genetic, neural, proteic, metabolic networks can all be considered, under an appropriate interpretation[54], computational systems which process informations[55].

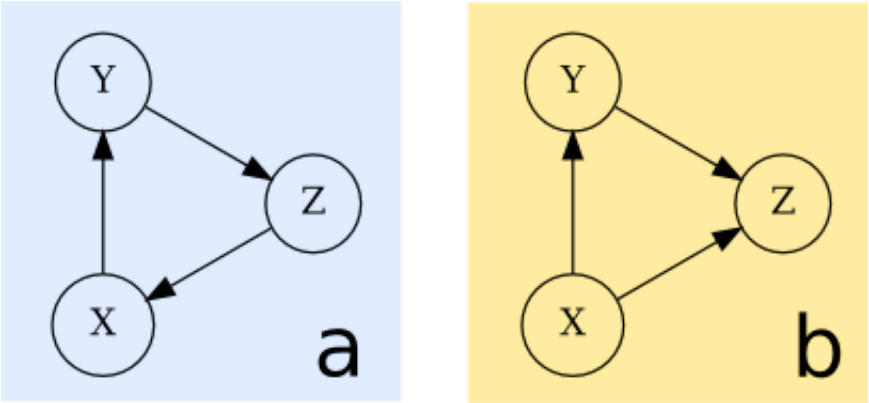

Figure 3.10: two typical network motifs. a: feedback loop; b: feed-forward loop.

It turns out that the same repertoire of networks motifs is often present in networks with similar functionality: for example, Milo et al. (2002) found the same two types of network motifs, the so-called *feed-forward loop* and *bi-fan*, in the genetic networks of both *Saccharomyces cerevisae* and *Escherichia coli*. Two similar types of motif were found in the neuronal network of *Caenorhabditis Elegans*[56]. This similarity in the motif types of the genetic and neuronal network could, according to the authors, suggest that similar constraints hold in networks with the same basic function: in this case, the function of performing some kind of information processing. Both the examined network types, the genetic one and the neuronal one, get information from sensory input, in this case biochemical signals or sensory neurons, and process it in order to transmit the processed result to effectors, that is, motor neurons or structural genes. This hypothesis seems corroborated by the fact that in other types of network with different functions, for example, ecological food webs, other, different types of motifs recur. On this bases, it has been proposed in Milo et al. (2004) that the types of motifs present in a network could be used for classification of network types.

[52] See section 9.1.

[53] See for example Shen-Orr et al. (2002) and Lee et al. (2002).

[54] It is the opinion of many that not every process can be considered, *per se*, a computation. See sections 14.5.1 and 14.5.2 for a view on the conditions for computation.

[55] The idea that many biological networks can be considered computational is widespread. For example, Hartwell et al. (1999) claims that an organisms, and even a cell, can be seen as a computational system in which the inputs are environmental measurements and the outputs are behavioral responses: "Indeed, the history of life can be described as the evolution of systems that manipulate one set of symbols representing inputs into another set of symbols that represent outputs." (p. C49). See also section 7.2 and 7.3

[56] *C. Elegans*, a small nematode worm, is one of the few organisms whose nervous system's wiring has been completely determined. It consists of 302 neurons, connected by about 5000 synapses. See White, Southgate, Thomson, & Brenner (1986).

Network motifs are recurring modules of usually *small* size, typically 3 or 4-nodes subnetworks. There is a relationship between the property of being a *small-world* network[57] and the recurrence in that network of network motifs: due to the fact that being small-world entails high neighborhood clustering, it is to be expected that 3-nodes motifs abound in such networks[58].

Actually, search for motifs with more than 5 nodes is usually considered pointless, for the reason that there are 1,530,843[59] different 6-nodes motifs, and thus that their classification would be hopeless: network motifs are interesting, and can be used as simple, standard building blocks, precisely because there is a *small* numbers of types of motifs.

Nevertheless, it is to be expected that higher-level modularity concerning identical recurring modules could show up at different hierarchical levels in certain networks, so as to constitute a full hierarchical modular structure: it is possible that, going up in scale, *higher-level* subnetworks, *larger* than network motifs, show up as higher level modules *composed* of strictly interrelated, often overlapping, network motifs.

Subnetworks of immediately higher order than network motifs are what Kashtan, Itzkovitz, Milo, & Alon (2004a) call *motifs generalizations*. This kind of subnetwork can be seen as an extension of a basic network motif, obtained by adding more nodes in parallel to those constituting the original motif. Motifs generalizations appear to perform specific *information processing* functions, as confirmed by the analysis of motifs generalizations recurring in gene transcription networks of *E. Coli* and *S Cerevisiae*, and in the neuronal network of *C. Elegans*. The type of motifs generalizations which have been found are similar in both genetic networks, but different in the neuronal one. This seems to corroborate the idea, also proposed by Milo et al. (2004), that motifs or their generalization can be useful for classifying the type of network and its nature.

Other structures of higher order than network motifs have been defined *network themes* in Zhang et al. (2005). They often can be seen as representing integrated complexes which perform a certain type of *function*[60] in the network. Often, they can be seen as *modules* in the sense of clusters densely interconnected internally, and with sparser connections toward the external world.

Gulbahce & Lehmann (2008) explicitly considers motifs as only the smaller type of composite modules, putting them in a hierarchy that goes from single nodes, to motifs, composed of a few nodes, to communities, composed of motifs. Thus, although motif modularity represents a particular feature of modularity in general, that of repeating occurrences of a single type of module, analysis in term of motifs can be seen as simply a quite low modular level in the hierarchical modular description of a network.

3.2.3 Network roles

As discussed in the preceding section, and more in general in chapter 6, structural modularity in networks appears often related to functional modularity, and functional analysis, as we will see in section 9.2, consists in identifying the role a functional module fulfills inside the whole functioning of a system.

Modularity detection can reveal the modular structure of a network, and, since structural modularity can in most cases be considered as at least partially determining a corresponding functional

[57] See section 3.1.2.
[58] Zhang et al. (2005).
[59] See Itzhack, Mogilevski, & Louzoun (2007).
[60] See section 9.1.

modularity[61], each detected structural module can be, at least prima facie, suspected to perform one of the subfunctions which a functional analysis of the system would individuate. The interesting question is how to individuate the structural modular description which best matches the functional modularity of the dynamic process running on the network: while community structure emphasizes the relative independence and isolation of each module, which can be seen as a partially independent unit, identification of roles depends on the relationships which hold *between* functional modules, and thus a functional modular description should be based on the individuation of communities *and* of their roles. A role-based description can be more or less coincident with the community-based one: often, as we will see, in the role-based descriptions, there are *single* nodes which act as modules, especially modules which relay communication between other modules.

While the specific input-output function performed by a module, which can be seen as its role, can be explicitly characterized by further modular decomposition of the module into functional subparts, such as network themes and network motifs, oftentimes it is useful to recognize the high-level function fulfilled by the module, that is, in other words, the *type* of functional role it performs inside the overall functioning of the network, in order to give a coarse-grained functional characterization of the network functioning[62]. The basic idea is that the type of role depends on how the module is connected to other modules: for example, intuitively, a module which is internally simple and stands between two information-processing modules, can be considered a relay module (or a so-called "connector"), a module which basically conveys information between two other modules of the networks.

A search for methods of detection of this kind of structural-based functional modularity in networks was started already in the '70s in sociological research, well before the idea of modularity in networks were to be examined in general[63]. Studying social networks, Lorrain & White (1971) introduced the notion of *structural equivalence* between nodes: two nodes are structurally equivalent when they are replaceable inside the network structure, and this happens when they are connected to the same elements (see fig. 3.11). The idea behind this notion is that a network can be partitioned into subnetworks, each of which is constituted by an equivalence class of structural equivalent nodes, and that such a modular partition could capture, based on this *structural* equivalence, an idea of *functional* equivalence.

It must be noted that this form of detected modularity does not usually coincide with *community structure*[64], which is defined according to the metric of connectedness between nodes: partitioning a network by structural equivalence does not, as fig. 3.11 clearly shows, constitute a classical case of community structure modularity, because it is not based on the classical metric of betweenness of edges or connectedness of nodes. Nevertheless, the partitioning found according to structural equivalence constitutes a form of modularity in accord with the general idea of modularity: it constitutes a modularity detected precisely according to a metric of structural equivalence between nodes: modules are groups of nodes which are more structurally equivalent than how they are equivalent to nodes outside the group.

There is, however, a problem with Lorrain & White (1971)'s original formulation of structural equivalence: the condition it states, that two nodes are structurally equivalent when they are connected to the *same* nodes, turns out to be too strong and not really useful, because perfectly structural equivalent nodes are too rare in real networks.

[61] See discussion in section 6.

[62] This is more or less what functional explanations tries to accomplish in cognitive psychology: being multiply realizable, specific functional roles are subsumed under *types* of roles.

[63] As we have seen, this line of research started in the first 2000s with the works of Newman ans Girvan.

[64] See section 3.2.1.

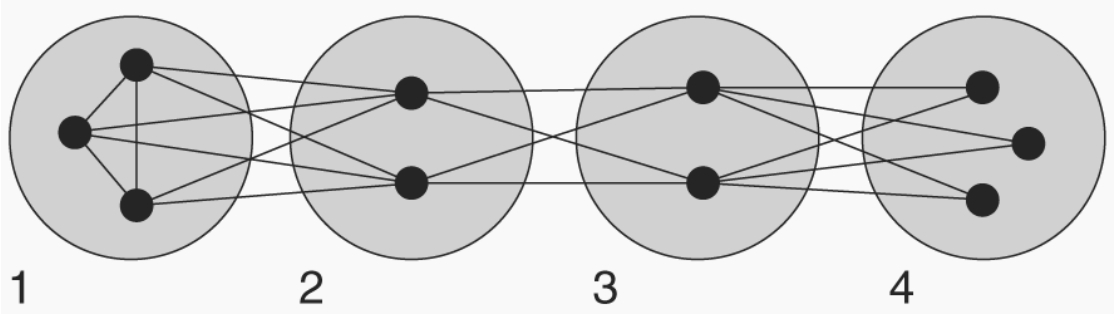

Figure 3.11: example of *structural equivalence*. A network is partitioned into modules (labeled *1*, *2*, *3*, *4*), each of which groups together nodes that are structurally equivalent according to Lorrain & White (1971)'s definition. For example, module 4 contains nodes which are structurally equivalent because each of them is connected exactly to the same nodes in module 3. (Image adapted from Guimerà & Amaral 2005a).

The notion of structural equivalence has been accordingly relaxed into various forms of weaker equivalence that Faust (1988) subsumes under the notion of *general equivalence*. In these forms, equivalence is equivalence among *types* of roles, and this gives a more synthetic functional description than that based on specific non-recurring roles. Nodes in a network which perform the same role type are to be considered generally equivalent. As an example, when considering the hierarchical structure of a military organization, commanders, who only give orders to others, can be considered generally equivalent, and the same holds for soldiers who receive orders to execute. In this cases the role types are that of "commander" and "soldier".

One kind of general equivalence called *regular equivalence*, widely used in sociology, was proposed by White & Reitz (1983), who deem it to be the most natural way of detecting social roles: regular equivalence is *the relation that holds among nodes which are connected to nodes belonging to the same class of regularly equivalent nodes*. In other words, whereas only the nodes which are tied to the same other nodes are considered *structurally* equivalent, for two nodes to be considered *regularly* equivalent, it is sufficient that they are tied to nodes which are regularly equivalent themselves. For example, two commanders, commanding two disjoint groups of soldiers, are regularly equivalent, because each commander is tied only to *soldiers*, while the same commanders can not be considered structurally equivalent, for they are connected to *different* soldiers. At the same time, the equivalence class of soldiers is defined with respect to the equivalence class of commanders.

All these attempts to individuate the roles of nodes and modules of the network are attempts to give a functional analysis of the network. Less strict notions of roles, such as that of regular equivalence, give a more "high-level", or coarse-grained modular view of the network structure. This is apparent in fig. 3.12, which compares functional descriptions of the same network, based on structural equivalence and regular equivalence: the regular equivalence-based one is a higher-level description than the one based on structural equivalence. In other words, by relaxing the requirement for equivalence, the function types so detected are more *abstract* and more *multiply realizable* than the corresponding stricter ones[65].

Guimerà & Amaral (2005a) argue that, despite the apparent plausibility of regular equivalence, this notion is not apt to capture functionally interesting roles related to the modular structure of the network. It is true that, by definition, regular equivalence allows the individuation of artificial, contrived roles which do not capture any type of actual, plausible functional dynamical

[65] Notions of abstraction and multiple realizability are to be better examined in section 6.6.

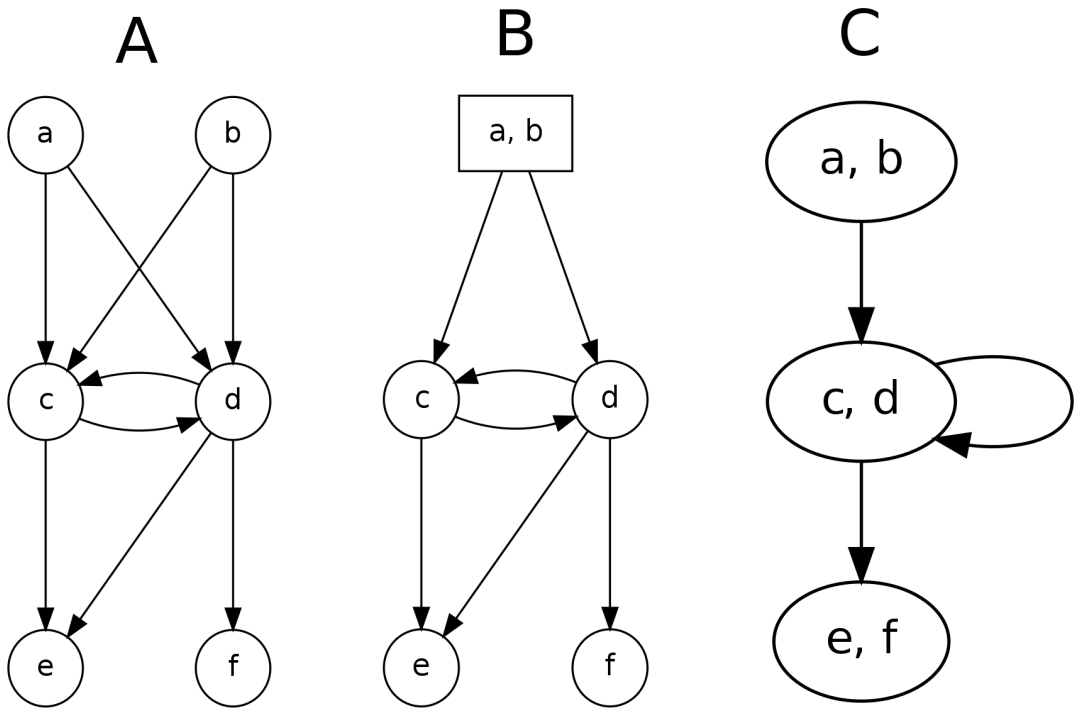

Figure 3.12: progressively coarser-grained functional descriptions of a network. Graph A: the original network, depicted at its lowest level, the level of single nodes. Graph B: nodes a and b of the original network are structurally equivalent, and as such are subsumed under a single module. Graph C: regularly equivalent nodes are subsumed under single modules. It is clear that regular equivalence yields a more abstract view than structural equivalence.

modularity: this could be easily seen in the example network of fig. 3.13, which shows that roles involved in communication and information processing and information flow in networks are neglected by regular equivalence, when this equivalence is based on a metric concerning non-functional properties of nodes, such as color similarity.

Starting from this alleged[66] incapability of regular equivalence to capture the functional modularity of networks, Guimerà & Amaral (2005a) propose an interesting, different definition of network roles which directly ties roles with the modular structure of a network understood as its community structure: the idea is that structural network modularity in the form of community structure actually reflects a form of functional modularity, and that functional roles can be attributed to nodes based on their topological properties, that is on how they are connected with nodes inside and outside *their community*. Proposed measures of these topological propoerties of nodes are *participation coefficient* and *within-module degree*, and node roles are classified according to the relative values of these two measures: informally, the participation coefficient measures how much a node belonging to a module is connected to nodes of *other* modules, and the within-module degree is a measure of how each node is well connected (so to speak) to the

[66] The example produced by Guimerà and Amaral does not seem to me to show a problem which can be blamed specifically on the notion of regular equivalence: given that *any* notion of modularity is relative to a metric, the problem lies here with the metric. In the example given by Guimerà & Amaral (2005a), the chosen metric, that of equivalence by color, is clearly not relevant in a functional description of the dynamics which a network can express, and thus it appears an ad-hoc choice aimed at forcedly producing a counterexample.

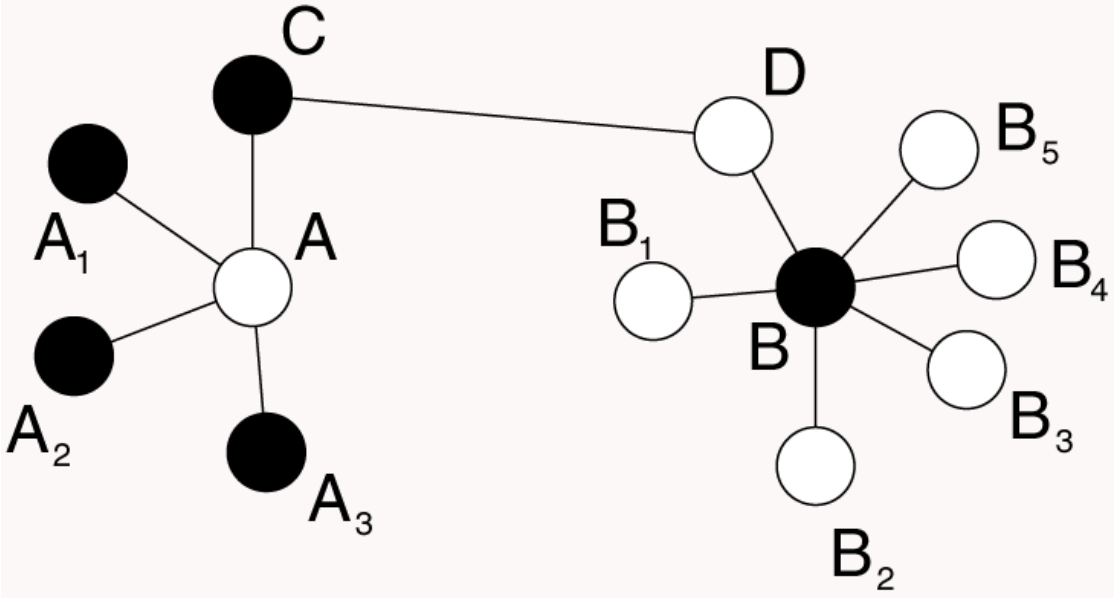

Figure 3.13: an example showing that *regular equivalence*, as conceived in White & Reitz (1983), does not capture certain types of functional modularity in networks: in the example, by definition of regular equivalence, white nodes turn out being regularly equivalent because they are connected only to black nodes, and black nodes are regularly equivalent for they are connected only to white ones, but the fact of being a white of a black node does not capture any relevant functional role: interesting functional roles are for example that of being a hub, to which many other nodes are connected, or to be a relay node, which mediates communication between to larger modules. The role of hub is fulfilled in the example network by nodes A and B, while the role of relay belongs to nodes C and D. Regular equivalence, reflected in the color of nodes, does not capture these functional roles. (Image adapted from Guimerà & Amaral 2005a).

other nodes in *its* module. Various combination of values of these two parameters allow for the identification of several types of module roles or node roles: Guimerà and Amaral propose to distinguish seven main types of functional roles, which they call "universal roles". First, a main distinction is operated along the dimension of *within-module degree*: *hubs* are nodes which score higher than a certain threshold on this measure, and non-hubs are nodes which score lower than this threshold. Further, finer distinctions inside each of these two categories can be made, based on the value of the *participation coefficient*. Three type of hubs are individuated, with growing participation coefficient: *provincial hubs*, *connector hubs* and *kinless hubs*. While provincial hubs connect other nodes of their module, and connector hubs devote about half of their connections to communication toward external modules, kinless hubs are not clearly belonging to specific modules, but rather act as external relay points of communication between different modules. Non-hub nodes can be finely classified into for categories, ranging from ultra-peripheral nodes, which are connected only to nodes belonging to their module, to connectors, which connect intra and extra-module node, to kinless nodes, external to modules, but which are not real hubs because of the limited number of their connections.

The important point to highlight in the proposal by Guimerà and Amaral is that their method of role classification directly connects the typical notion of *structural* modularity in networks, that of community structure, with a notion of *functional* modularity based on roles, because roles are defined *relative to* the detected community structure: as said, according to their definition, roles are characterized by the values of two parameters, *participation coefficient* and *within-module*

degree, which are values computed on the basis of the connections of a given node to other nodes lying within and without its *community*. Accordingly, Guimerà and Amaral's method is actually accomplished in two phases: (i) community detection is operated on the network[67]; (ii) roles are assigned to nodes and modules based on measures referring to their connectivity patterns within their modules and toward the external context. Guimerà and Amaral call this composite method the production of a "cartographic representation of complex networks", by analogy with cartographic representations, in which not only cities and roads connecting them are reported (where cities are modules and roads their connections), but also classes of relative dimension and importance of roads and cities (parameters corresponding to roles) are represented in the map by the size and color of elements: see for example fig.3.14. In this view, functional roles are relative to the modular structure of the network, and to the dynamics which can be implemented on such a modular structure: in other words, the network's structural modularity in the form of community structure *induces* a corresponding functional modularity in the form of certain fulfilled roles, which can explain the networks' dynamical behavior.

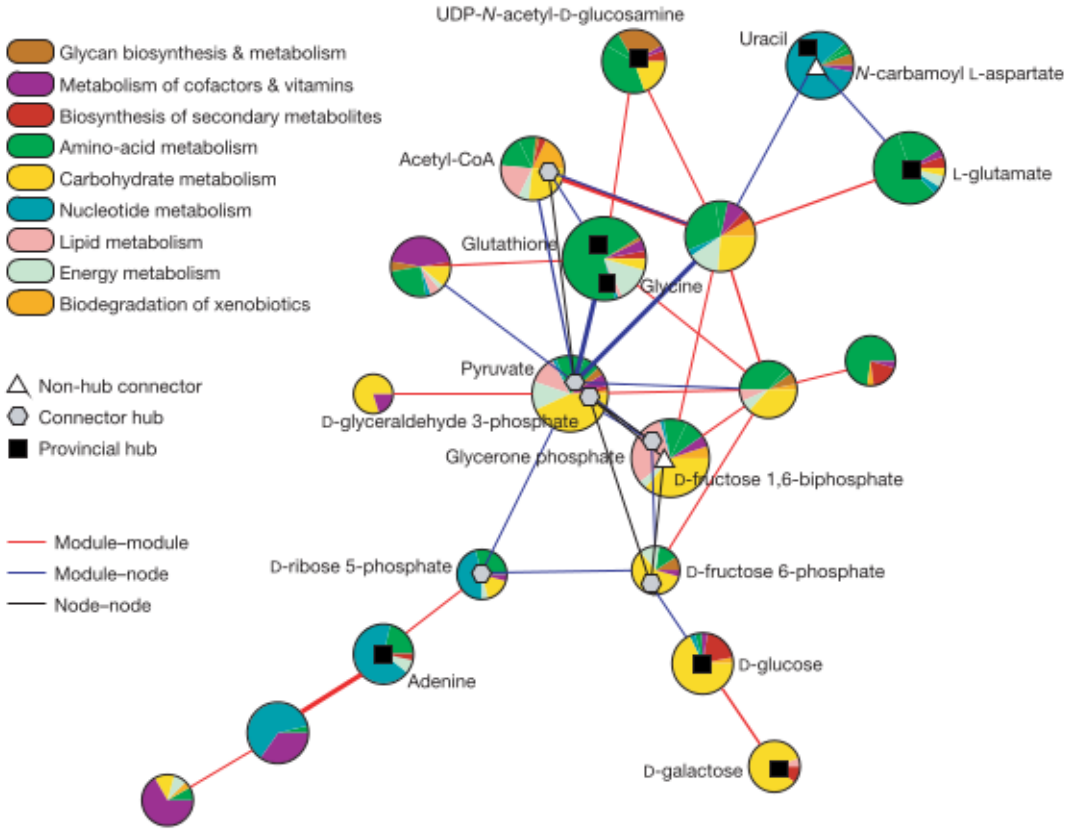

Figure 3.14: "functional cartography" of a metabolic network. Image taken from Guimerà & Amaral (2005b).

That the correlation between community structure and functional modularity proposed by Guimerà and Amaral can be useful in explaining the network's dynamics, is a point to be verified, of course. The authors put this idea to the test in Guimerà & Amaral (2005a) and Guimerà

[67] The authors propose a custom method for community detection, based on simulated annealing (and as such quite computationally intensive), which, according to them, gives particularly good results. See section 3.2.1.2.

& Amaral (2005b), by applying it to real-world networks. On the whole, they consider three artificial networks and biological networks in twelve organisms, comprising bacteria, archaea and eucaryotes, and find a good correlation between the functional decomposition determined according to their proposed role types, and the already known functional modularity of these networks.

Building on the classical notions of structural and regular equivalence Reichardt & White (2007) propose a generalization of the concept of role, which sees a description in terms of roles as any way to partition the nodes of a given network into equivalence classes based on their connection patterns, and to construct, based on the detected roles, simplified networks representing abstract functional models of the original network (an example of an abstract model is depicted in fig. 3.12, graph C). From the point of view of this generalized paradigm, the classical notions of structural equivalence, regular equivalence and also of community structure turn out to be *particular* ways of partitioning the set of nodes of a network: for example, community detection is a grouping of nodes realized according to the density of connection[68] between them. Reichardt and White propose an algorithm for finding the best abstract simplified model of a given network, which makes use of simulated annealing to find the optimal model, and is thus quite computationally intensive[69]. The proposed general conception of roles fits well with the general idea of modularity, according to which modularity is always relative to a chosen metric, and applies this general idea to networks: while community detection makes use of the specific metric of connection density, many other metrics are envisionable, each of which gives rise to different forms of modularity. In this framework, network modularity can thus be seen as quite a general property[70].

3.2.4 Functional typology of hubs: *party* hubs and *date* hubs

In functional modular descriptions of networks like the ones expounded above, much importance is given, as expected, to hubs which connect different modules. A well-known, if debated, differentiation of hubs into two main types, especially notable in scale-free networks[71] of biological significance, the distinction between *date hubs* and *party hubs*, was proposed first in Han et al. (2004), inspired by observed properties of the protein-protein interaction network of yeast: while so-called "party" hubs interact with all their connected nodes in a more or less simultaneous manner, "date" hubs interact with different connected nodes in different times. Han and colleagues, following the idea, discussed in sections 6.7 and 3.2.4.1, that there should be a relevant decoupling of timescales in modular hierarchical systems, speculated that this distinction in temporal activation between the two types of hubs reflects their topological patterns of connection inside the network. The idea is that intramodule interactions can be expected to be much more frequent than inter-module ones, and can be seen as more or less "synchronous" when considered from the slower time scale of a higher-level point of view. This is reflected in the observation that party hubs in the yeast protein interactome interact with the nodes connected to them more or less simultaneously, and so they can be considered as *intramodule* hubs. On the other hand, inter-module interactions, while viewed at a high-level, show their specific temporal sequences, and this is reflected in the fact that date hubs interact with different partners at different times: date hubs can thus be considered *inter-module* hubs, that is "higher level" connectors between

[68] The formal measure is usually edge betweenness. See section 3.2.1.2.

[69] They don't explicitly specify the time complexity of the algorithm, but state that an exhaustive search for the best model, instead of the approximate search by simulated annealing, would require exponential time: this is expected, given that community detection, whose optimization task is, as highlighted in section 3.2.1.2, NP-complete, is actually a particular case of the general method proposed by Reichardt ad White.

[70] See also section 3.2.7 for further discussion.

[71] See section 3.1.3.

different modules. The hypothesis that this distinction between two classes of hubs based on properties of their temporal activation can coincide with different *functional roles* of the hubs in a functional modular description of the network, was validated in Han et al. (2004) by comparing the modules which are supposed to be connected by date hubs with already known functional modules in the yeast's protein interaction network.

While the proposed theoretical distinction between date and party hubs has since spread in the literature, it is a distinction which has also been contested: according to Batada et al. (2006) and subsequent works[72], the date/party distinction based on activation rates does not reflect, at least in the yeast protein interactome, an actual distinction of roles in the functional modular structure of the network, but is simply due to a sampling bias in the original work by Han and colleagues. They claim that, contrary to the hypothesis which sees the modular structure of biological networks as composed of highly segregated modules loosely connected by means of date hubs, the actual topological structure of these networks resembles more a group of partially overlapping, highly connected modules.

3.2.4.1 Timescale decoupling and dynamical methods for community detection

Due to the high computational cost of many of the algorithms for hierarchical detection based on network structure[73], different methods based on the *dynamical behavior*[74] of networks have been proposed. The idea stems from the original discussion by Herbert Simon on the different timescales of the dynamics in different parts of a system, which show up in a hierarchical modular structure, as hinted at in section 6.7: intra-modular dynamic interactions between nodes occur in general at a faster rate than the inter-module ones. Based on this intuition, some methods for community detection in networks have been presented: the basic idea is that *structural* modularity of a network, that is its community structure, bears on the dynamics implemented on the network, and that, for this reason, it is possible to infer community structure by observing the network's dynamics. It must be noted that this approach based on *dynamical* properties of the network, is able to make use of information which is not available at all by observing solely the static structure of the network: dynamics can show properties not directly reducible to those of the structure on which they run. Consequently, some of these methods produce results which are more faithful to the modular dynamics that take place in the network, and this can be quite useful, considered the fact that usually a network model is supposed to represent the structure of some dynamical system.

A typical approach derives from ideas in the seminal Kuramoto (2003), a work in which a model for the dynamical interaction of a set of connected oscillators is described. Several works have built on this approach. Starting from this work, Alex Arenas, Albert Díaz-Guilera and their staff, have conducted, since the seminal Arenas, Díaz-Guilera, & Pérez-Vicente (2006), studies aimed at evaluating the relationship between a network's structure and the dynamics implemented on it, and at modularity detection in networks by means of timescale decoupling. Their method is best exposed in Díaz-Guilera (2008), a review of these former works. Community structure is detected by implementing a dynamics on a network by means of oscillators, each oscillator corresponding to a node, influencing and influenced by the other oscillators linked to it. The theoretical model employed by the authors, which is directly derived from Kuramoto's model, is too complex to be described here in detail, but it can be summarized: the oscillators are identical, with the same frequency, with different phases at startup, randomly assigned. When

[72] Further discussed in section 7.
[73] See the preceding sections.
[74] For the relation between structure and dynamics, see section 6.2.

able to fully influence each other, two oscillators tend, due to the nature of the interaction, to become synchronized: that is, the difference between their phases, in a certain time, is reduced to zero. If the reciprocal influence of the oscillators is weak, synchronization is less easily achieved. Synchronization between a group of nodes is proportional to the number of links connecting them. Thus, in a modular network, intra-module synchronization, where link density is high, will be achieved first, and then, in a sequential manner, synchronization will become to spread from each module to its external context, until, in the end, all the nodes of the whole network came up synchronized. This is a manifestation of hierarchical temporal decoupling similar to what we have already seen in Herbert Simon's office room example. A similar form of temporal decoupling can be seen in the temporal activity of different classes of hub nodes, as already explained in section 3.2.4. "Party" hubs, which are supposed to be inter-module, interact in a more or less synchronous manner with their connected nodes, while "date" hubs manifest a slower-rate pattern of interaction with the nodes connected to them, with interactions with different nodes occurring at different times. This is consisten with the idea that date hubs are supposed to connect different *modules*.

Pan & Sinha (2009) found that in real-world modular weighted networks the inter-module connections tend to be weaker than the intramodule ones, and, accordingly, a timescale decoupling of the synchronization of oscillators can be observed, a decoupling which tends to zero as the relative strength of inter-module connections becomes of magnitude similar to that of intra-modules links. This is a very interesting, in intuitive, result, which shows that timescale decoupling is not a necessary feature of modular networks, and that it can be overcome by certain properties of the network connections. For example, even if a network is structurally modular, if the dynamics implemented on it are strongly non-linear, a simple timescale decoupling is not to be expected, because the non-linearity of inter-module connections can compensate for their sparseness[75].

Building on the aforementioned dynamical methods of modularity detection, which as a downside do not seem to yield a very high precision, Boccaletti, Ivanchenko, Latora, Pluchino, & Rapisarda (2007) propose an algorithm which improves accuracy of community detection: it works by individuating a set of possible modular partitions of the network based on the timescale decoupling of synchronization of oscillators, proceeds by evaluating the Q modularity quality of each candidate modular subdivision so obtained, and chooses the subdivision with the highest Q value. This way, in a time complexity of only $O(n^2)$, the algorithm yields results of accuracy comparable to those of the best classical methods of community detection, and inferior only to algorithms based on simulated annealing, which are much more computationally expensive.

3.2.5 Coarse-graining of networks with community structure or recurring modules

Once high-level modularity is found in a complex network[76], it is to be expected that the network description could be simplified by resorting to a *coarse-grained* model of the original network: coarse-graining amounts to constructing a new network, different from the original one, whose nodes correspond to the modules identified in the original network, and whose edges correspond to the connections between those modules. In a manner in some way reminiscent of the aggregation of variables,[77] it is this way obtained a network with less nodes and edges than the original.

[75] I discuss these problems also in section 6

[76] that is, once communities, or network motifs or themes, have been detected, and possibly even higher structures, such as communities of communities, and so on. I'm still using here the notion of "high level" informally. I carry out a deeper discussion on levels of descriptions in section 6.6.

[77] See section 2.2.1.

As explained in section 6.8, the simplification obtained by representing the network as a hierarchical modular system can ease understanding of its structure, understanding which could be severely limited in the case of a complex network which does not exhibit any obvious modular structure: a non modular network with a high number of nodes could overload our perceptive capacities. By looking at a coarse-grained representation of network, in which each module of the original network is substituted by a single node, the observer can certainly more easily catch the network structure.

In the preceding sections I introduced two basic forms of network modularity: community structure and modularity by recurrence of similar substructures, which, in the case of networks, take the form of network motifs or network themes. Accordingly, coarse-graining of networks can be performed by basing it on either kind of modularity.

A typical example of coarse-graining by *community structure* identification can be found in Newman & Girvan (2004), a seminal paper about network modularity. The authors take into consideration a social network which represents article coauthorship between a set of physicists: in this network they identify, by means of their algorithm[78], the communities, and thereby come up with a coarse-grained version of the network, as represented in fig. 3.15. Network structure appears certainly more understandable by looking at the coarse-grained version.

A more complex, important and more informative form of coarse-graining takes into account not only communities, but also their function, as well as the structure of the connections between them, by representing also node-level functional information: inter-module nodes which connect different modules are identified and functionally classified according to their roles, as already seen in section 3.2.3[79].

Another form of coarse-graining is allowed by the identification of *recurring similar modules*. Coarse-graining based on this kind of modularity can be exemplified by the method reported in Itzkovitz et al. (2005), which consists in:

1. identifying *network motifs* and superstructures composed of motifs, and considering them as module types, which the authors call *CGUs* (Coarse-Graining Units); the set of all CGUs, that is, the set of the types of recurring modules, is called the *CGU dictionary*;
2. constructing a new network (different from the original) in which each node is a CGU and edges stand between CGUs. This is the *coarse-grained* version of the original network.

This is a kind of modularity which differs from the simple recognition of communities: while community-based coarse-graining allows for a better understanding of the network structure, a coarse-grained description based on the detection of recurring modules, such as motifs, or network themes, or higher-level similar structures[80], also allows for a form of *lossless information compression* of the detailed description of the network. I would like to stress this difference: while coarse-graining by community detection allows for a simplified representation of the network at the coarse-grained level, this coarse-grained representation contains *less* information than the original network: the detailed structure network of the *inside* of communities gets lost in the coarse-grained representation[81]. To take such level of detail into consideration, we must resort to the original, fine-grained representation of the network. On the contrary, when we obtain a

[78] See section 3.2.1 and 3.2.1.1.
[79] See also section 6.4.
[80] See section 3.2.2.
[81] This is similar to what happens with aggregation of variables in near-decomposable systems, as described in sections 2.2.3 and 2.4.

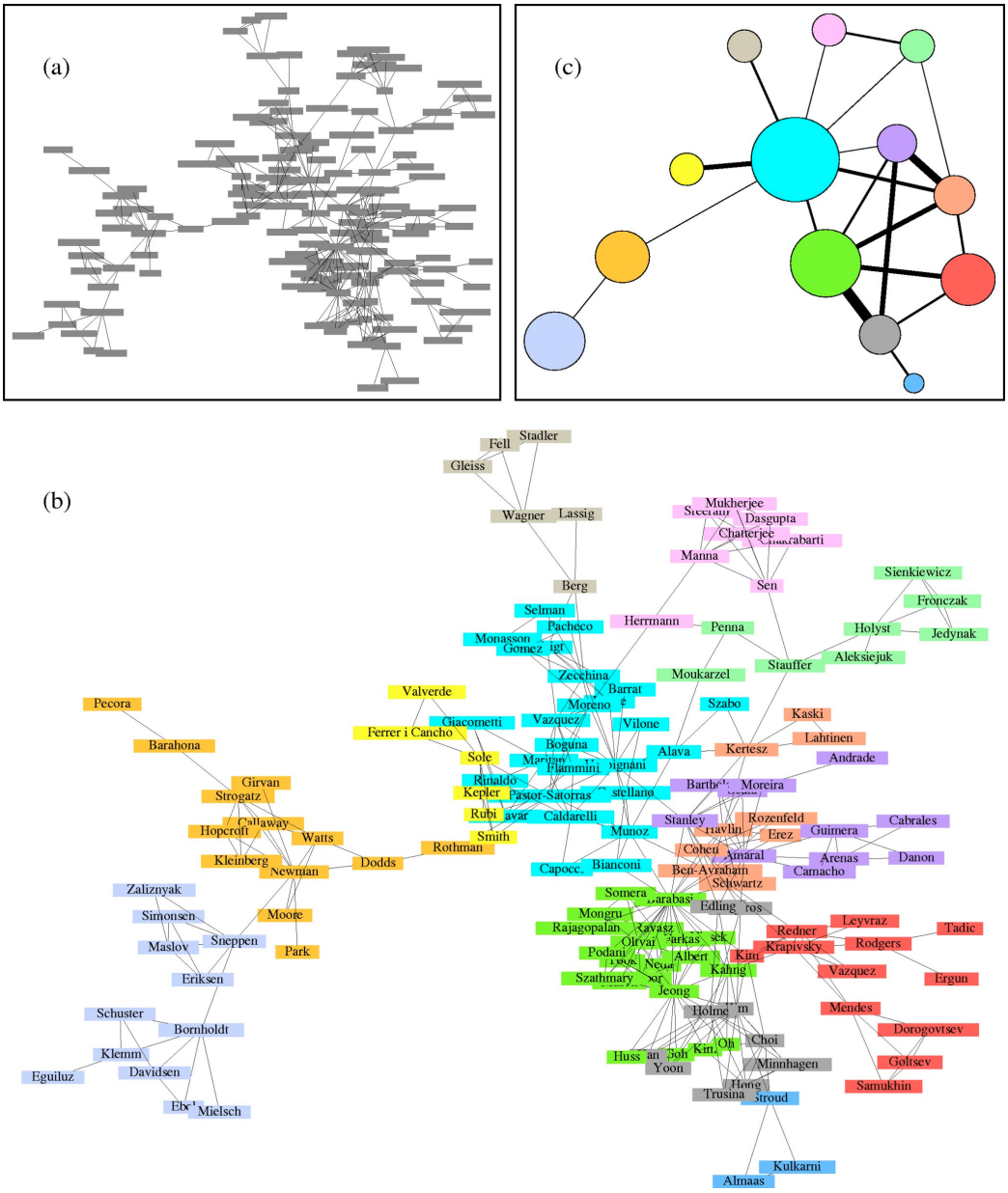

Figure 3.15: (a) the original network of coauthorship between physicists; (b) detected communities in the original networks, identified by colors; (c) the coarse-grained representation of the network: colors indicate which community of the original network each node in the coarse-grained network represents. Image taken from Newman & Girvan (2004).

coarse-grained description of the network based on the identification of recurring similar modules, we could in principle come to know the internal details of each module not by looking at the original fine-grained representation of the whole network, but simply by looking at the internals of *that* module's corresponding type in the CGU dictionary. This means that the coarse-grained representation virtually contains, in a compressed form, all the information contained in the original fine-grained representation of the network.

Itzkovitz and colleagues explicitly make an analogy between network modularity detection and the way[82] digital electronic circuits can be *reversed engineered* in order to view them as composed of functional blocks, in turn these blocks can be seen as composed of logic gates implementing boolean functions, and the gates as composed of transistors and other simpler components. In the same way the coarse-grained network can be seen as constituting a higher-level model of the original network: in the coarse-grained network, each node, corresponding to a module (a community of nodes) in the original network, is seen as a *black box*, equipped with a set of input and output links. This way, every node can be considered as a unit performing a given computational function, which transforms in a given way input signals into outputs.

The fact that these modules can be seen as performing computational functions plausibly renders them, in many cases, *modules* also in the general sense of being sets whose internal elements are more densely interconnected to each other than to elements of other modules. The reason is that it is expectable that to perform an interesting, not completely trivial computation, the internal complexity of the module, measured in terms of the interconnections between its internal elements, that is, in terms of variables connected by simple functions, should be higher than the complexity of its input and output lines: arguably, a module whose internal complexity is on the same level of that of its external connections, probably amounts to a trivial *repeater*, a module which simply copies or its inputs to its outputs, or at most scrambles them, or a module which reduces information by outputting a lossy representation of its input. The very fact that, to perform a computation, a function requires to get information from an orderly, structured, set of inputs in order to produce the processed configuration in a structured set of outputs, in some way implicitly presupposes that, if the processed information needs not to get lost or to deteriorate, the set of outputs need not be indiscriminately connected to every other module. Thus, the density of interconnections *between modules* has to be limited by channeling outputs towards specific inputs, along well defined routes. This limited connectivity between modules, together with the fact that modules perform computational functions, renders quite likely that most modules show up as being more intensely connected internally than toward other modules. And this matches the notion which I sketched in section 6, the notion of modules as structures which are internally strongly connected and more loosely interrelated one with the other.

As said, in Itzkovitz et al. (2005) the authors seek to identify, for a given network, the set of the *types* of modules which can usefully coarse-grain the network. These types should ideally be few in number, with each module internally simple, and such that the coarse-grained description obtained by substituting parts of the original networks with these modules ends up being as simple as possible, and, hopefully, simpler than the original network. These required properties are partly pulling in opposite directions, and for this reason an algorithm is performed to search for the set of modules which *optimize* their combination[83]. Once the set of module types is found, a coarse-grained representation of the original network is produced, in which each node

[82] This is a typical analogy, see also sections 7.2 and 4.3.

[83] Time complexity of the algorithm is not explicitly stated in the paper. It is an algorithm making use of simulated annealing and Monte Carlo procedures, so it makes uses of statistical sampling to simplify the search inside an otherwise huge search space, and thus it is probably not expected to give absolutely optimal results in the choice of the module types set.

represents a module of some type. The paper highlights the fact that, usually, not all parts of the original network can be replaced by coarse-grained modules, and some nodes of the coarse-grained network still come to represent some original nodes.

A quite interesting feature of the algorithm employed by Itzkovitz et al. (2005), is that it can be recursively applied *to the coarse-grained representation*, in order to possibly obtain a *full hierarchy* of representations of progressively higher level of abstraction based on types of repeated modules.

The paper reports the application of this algorithm to two real world cases, in order to test its functionality: coarse graining of an electronic circuit, and of a biological *signal transduction network*, a kind of protein-based intracellular information processing network.

The electronic circuit, which originally had 516 nodes and 686 edges, got modularized in terms of transistors at the base level, of several types of logical gates at a higher one, and in terms of more complex computational objects[84] at two even higher levels, thereby obtaining a coarse-grained equivalent version composed only of 42 nodes and 56 edges, a higher-level version with complexity lower than that of the original system by an order of magnitude.

The protein interaction network examined had 94 nodes and 209 edges, and got coarse-grained by mainly a single, very simple type of network motif[85]. See fig. 3.16. The paper does not declare of how many nodes the coarse-grained network is composed, but it appears that the biological network taken into consideration reveal itself as less modular than the electronic circuit. Nevertheless, it has to be highlighted that in its coarse-grained representation three signaling channels can be easily distinguished, and that they turn out to correspond to three homologous links in the real biological network, which were already known. This approach to modular hierarchical can thus be considered pretty promising. Other examples of fruitful coarse-graining of networks can be found in section 7.2.

Community-based coarse-graining and coarse-graining based on detection of recurring modules are the two possible basic kinds of coarse graining, but mixed approaches, which take the best of both worlds, are possible: for example, by first performing a coarse-graining based on motifs, and then by *coarse-graining the coarse-grained description*, by community structure detection.

3.2.6 Modularity, small-world networks and information processing

A very interesting feature of modular networks showed by Pan & Sinha (2009) is the fact that, at a moderate degree of modularity, such networks *also* show the property of being *small world* networks[86]. Having been noted that small-world networks facilitate information processing, this result seems to corroborate the possibility that natural systems which are known to perform information processing, or at least systems which can be *seen* in this way[87], can turn out to be also hierarchical modular systems. Pan and Sinha highlight that, besides being small-world, these modular networks also possess a feature which, as we have seen in sections 6.7 and 3.2.4.1, is a well-known signature of hierarchical modular organization: that of manifesting a dynamics with multiple discrete time-scales characterizing local and system-wide interactions.

[84] Such as *counters*, at the highest level.

[85] The algorithm had been slightly modified to take into account that we don't have complete data about the actual network, and that recurring motifs in actual networks could turn out not being exactly identical. Modifications were thus directed towards detection of network motifs of different size approximately sharing a common network structure.

[86] See section 3.1.2.

[87] See section 14.5.1.

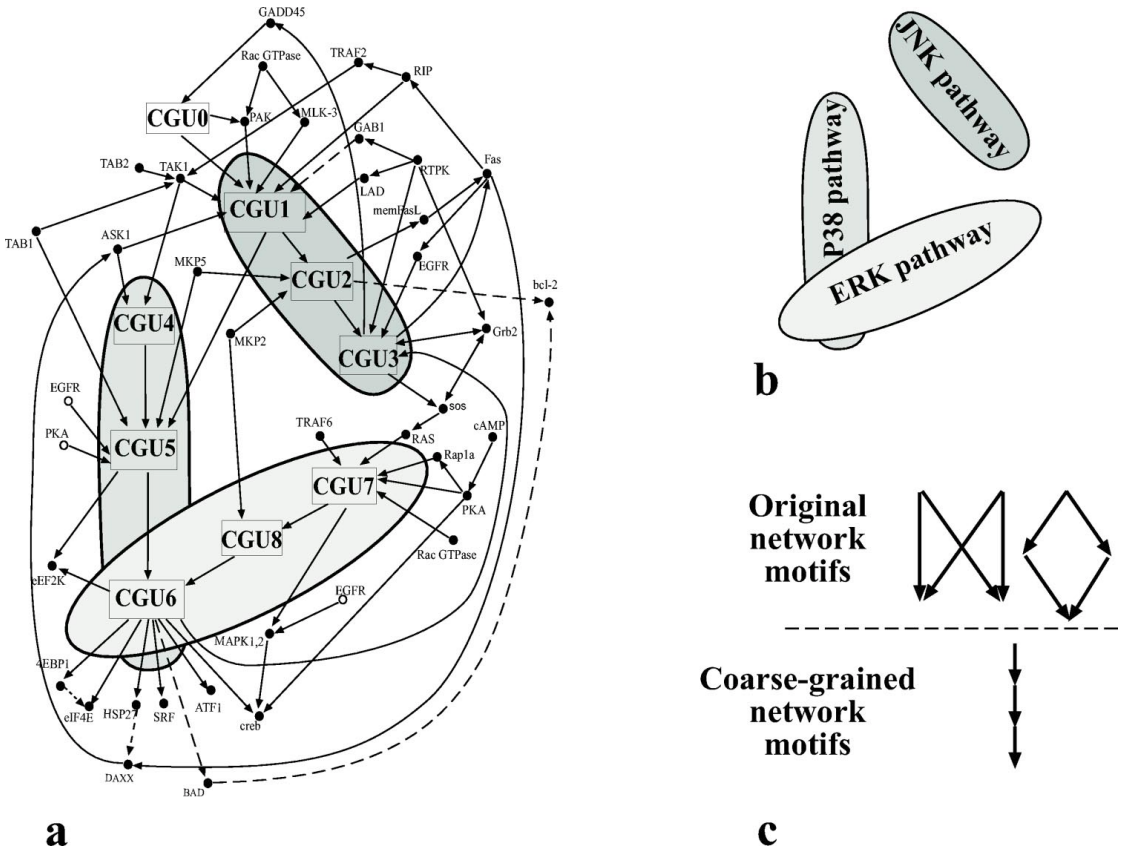

Figure 3.16: (a) coarse-grained version of the signal-transduction network; (b) the three already known signaling channels; (c) the set of network motifs employed. Image taken from Itzkovitz et al. (2005).

3.2.7 Differences between modularity in networks and general modularity

In section 6 we have informally analyzed a general notion of modularity, while in section 2.2.3 Herbert Simon's view of the matter has been exposed. There are some notable and often subtle differences between these conceptions of modularity on the one hand and the idea of modularity in networks on the other hand.

First, community structure, the most common form of modularity in networks, is detected according to a metric of *edge betweenness*[88] which can be informally interpreted as the density of connections between nodes. Thus this form of modularity is based on a *fixed* metric. But, as we know, modularity in general can be based on any metric between elements of the system. This possibility is raised in networks by Reichardt & White (2007), which propose a more generalized view of network modularity, based on metrics different from simple density of connections. Their method of modularity detection consists in a simulated annealing procedure aimed to optimize a measure of modularity quality, which can be seen as a generalized version of the modularity quality metric Q[89], and community structure is only one type of modularity their general method can detect. In general, their method allows for detection of *roles*[90] of nodes, where each detected

[88] See section 3.2.1.2.
[89] See section 3.2.1.1.
[90] See section 3.2.3.

module corresponds to a role. This framework allows thus for a generalization of the notion of modularity in networks, rendering it more akin to a general notion of modularity.

But a more basic, notable difference between modularity in networks and the general notion of modularity, is that in networks the typical metric used to detect modularity is a measure of some property of *nodes* and *edges*, while, in the general case, it should be possible to asses modularity on the basis of any given property whatsoever of any of the system's elements or sets of elements. This peculiarity of network modularity of being typically based on properties of nodes or edges stems from the fact that the obvious interesting property of a network is that it is a *network*, and a network is, essentially, a set of connected nodes and nothing more. However, nothing prevents us to use edges between nodes in a network to represent a non-purely network relationship: say, the degree of similarity between certain phenotypic traits in a group of insects species. In this case, the resulting representation is a network with weighted edges, where nodes represent species, and edges represent the strength of their phenotypic similarity. On this representation of a taxonomic zoological network, an algorithm for modularity detection can be applied, and the found modularity will be a form of modularity defined according to the relation (similarity of traits, in this case) *represented* as the edges. But the algorithm will have worked by searching for modularity according to its standard built-in metric, that is the purely *network* property about edges and nodes properties: usually, edges strength, or connection density. By grouping nodes in modules according to this structural, *network* property, the algorithm finds a modular structure that can be viewed as reflecting the modular structure *of the represented system*.

This shows that *network* modularity is after all an *inclusive* notion, if we allow for weighted networks, and it is not much less general than modularity as formulated as in the general definition of section 2.1: the relation represented as the edges between nodes, besides being a structural network metric proper, can more in general be chosen to represent any given relation whatsoever between elements of the system. Let's take a system composed of a number of elements, a system which is not necessarily a network, and a certain relation R of whatsoever nature between these elements. In order to perform a detection of the system's modularity according to this relation R, a network must be produced in which the relation R is represented by some standard network relation between its nodes, such as node density or edge betweennes. This network representation is to be produced by some algorithmic method. By changing which relation of the original system is chosen to be represented in the network, the structure of the network representation can change accordingly, and so its modular structure can change as well, or disappear: this is normal, since modularity depends on the choice of a specific metric. Once this network representation of the relation between elements of the systems is obtained, then a network modularity detection algorithm, which works according to some typical *network* metric, is applied to the network representation, and the modularity detected in the network by this algorithm, which constitutes network modularity detected *according to the network metric in the network representation*, will mirror a corresponding modularity present in the represented system, evaluated *according to the chosen relation R holding between elements of the system*.

For a very general example, the relation between nodes represented in a network can be the fact that nodes in the system share or not a certain property. This is exemplified in fig. 3.17 and fig. 3.18.

3.3 Limitations of algorithmic detection of modularity in networks

There are many possible algorithms for the automatic detection of modularity in complex networks, which consider modularity under the form of network motifs, of community structure or

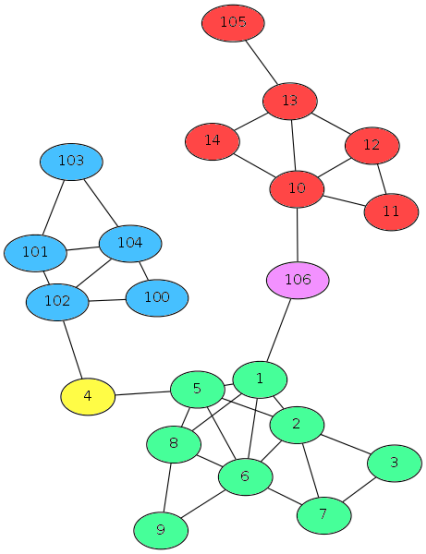

Figure 3.17: a bare network, which is modular according to the density of reciprocal links between the nodes. Distinct modules have different colors.

of hierarchical organization. Many of the known algorithms exhibit some limitations, such as a dubious reliability, a certain amount of computational complexity or a trade-off between speed and accuracy. The following sections are dedicated to a discussion of these problems.

3.3.1 Time complexity of community structure and hierarchy detection[91]

As seen in section 3.2.1.2, Brandes et al. (2008) has proved that the task of optimizing the measure Q of modularity[92] is an *NP-complete* task, and thus quite certainly a computationally intractable one[93]. As we have seen[94], Q evaluation is necessary in order to asses the reliability of a detected modular structure, and its *optimization* is therefore needed to choose the best modular partition amongst all the possible ones, or, in other words, to infer the most plausible hierarchical description of a network. Significantly, this limitation of the measure of modularity due to computational complexity holds for an extension of this measure to *weighted* networks[95].

This renders the idea to find the absolutely best possible hierarchical description of a given network quite hopeless, and thus we probably should, in many occasions, make do with a method for modularity detection which does not try to optimize Q, or which does not rely on this measure of modular detection quality and its optimization. Many of such algorithmic methods have been examined in sections 3.2.1, 3.2.1.1, 3.2.1.2, and 3.2.1.4. I will now try to assess their usefulness and feasibility in terms of the time they require for their execution. In my view, this is an important problem, and I will assess its consequences for scientific research and explanation in later chapters.

[91] The algorithms cited here are discussed in sections 3.2.1 and 3.2.1.2 and 6.2.
[92] The measure proposed by Newman and Girvan, see section 3.2.1.
[93] See section 17.4.3.1.
[94] In section 3.2.1.1.
[95] Brandes et al. (2008), p. 178. In the case of weighted networks, the definition of Q, instead of employing the number of edges, is based on the sum of edge weights.

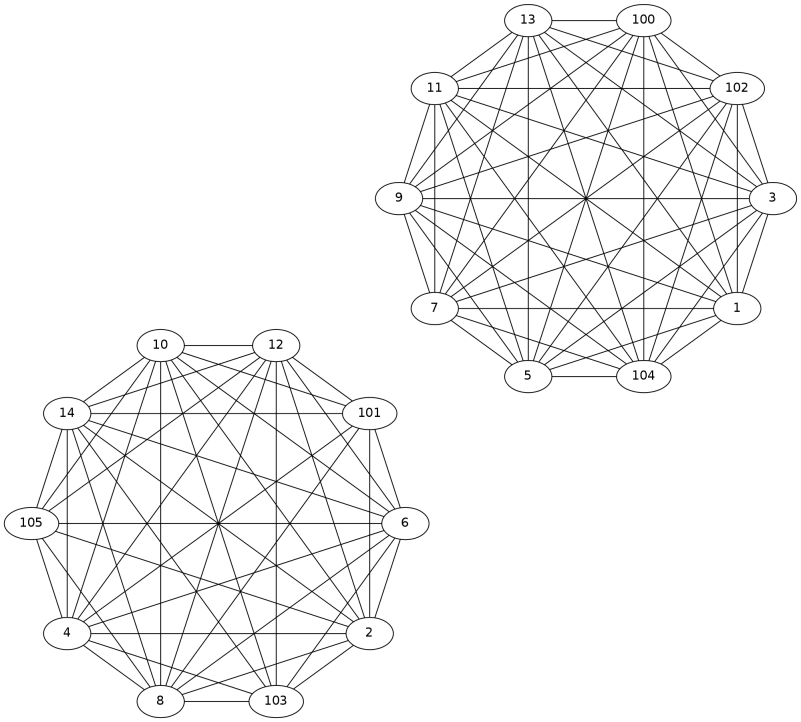

Figure 3.18: a network with the same nodes of the network in fig. 3.17, but which represents, with its connection structure, a complex property (the property is of being an even number lower than 20, or an odd number higher than 100), and which is, accordingly, modular. Here, distinct modules are the subset of nodes sharing the property, and its complement.

Numerous approximate algorithms have been devised for community structure detection. Danon et al. (2005) make a survey of many algorithms known at the time for the detection of community structure in complex networks. Each algorithm presents specific strong and weak spots.

Many algorithms for detection of community structure are affected by quite a high *time complexity*[96]. The original Newman and Girvan algorithm has a time complexity of $O(n^3)$ (with n the number of nodes in the network under analysis) for sparsely connected graphs, or $O(m^2n)$ in the worst cases, when m, the number of links between nodes, is significantly larger than n. Most of the other algorithms examined in Danon et al. (2005) are of polynomial complexity, but some of them are $O(n^3)$ and even $O(n^4)$, like the method proposed by Fortunato et al. (2004). This complexity, being polynomial, is theoretically considered a *tractable* one[97]. However, it is a quite *high degree* polynomial, and networks with more than 10^4 nodes tend to result hardly tractable.

Although not explicitly stated by the authors in their papers, the method employing simulated annealing proposed by Guimerà and Amaral[98], which is deemed the most accurate for community detection, has quite high time complexity: this can be inferred by some comments in Danon et al. (2005), where it is stated that the simulated annealing method is the most accurate of the examined ones but that it is slower, and that for this reason it cannot (at the time) be run on a

[96] See section 17.4.1.1.
[97] See section 17.4.
[98] Guimerà & Amaral (2005a), mentioned in section 3.2.1.2.

networks with more than 10^5 nodes, with a runtime of several month to be expected when this method is run on network of about 370.000 nodes. Consider that the other algorithms to which this method is compared in the paper comprise the ones by Newman and Girvan together with other ones, all with a time complexity of $O(n^3)$, so its time complexity is quite probably higher than $O(n^3)$.

In general, algorithms making use of simulated annealing, for their very nature, are quite intensive from a computational standpoint, even if they are a more feasible alternative to a brute force full search of the space of the problem, a search which usually requires exponential time.

The algorithm described in Karrer et al. (2008)[99] for measuring the significance of an already detected community structure has a time complexity dependent on the community detection algorithm adopted. In Karrer and colleagues' paper, the algorithm taken into consideration for community detection is the one proposed in Newman (2006), which runs in $O(n^2 \log n)$. The significance assessment meta-algorithm runs slower that that, for it must run Newman's algorithm several times, applying a different perturbation to the network structure before each run. Based on the authors' qualitative considerations[100], it seems plausible that at best only networks with less than a few million nodes could realistically be so analyzed in the foreseeable future.

The thorough method for detecting full hierarchies described in Lancichinetti et al. (2009)[101] runs in $O(n^2 \log n)$. This is a better runtime than many of the aforementioned algorithms, but for large enough networks, where 10^6 or more nodes are to be expected, even this method could become potentially problematic, unless one resorts to a massively parallel implementation, which, according to the authors, is quite feasible anyway. This algorithm can be easily extended to *weighted networks*. Another, different algorithm for native hierarchy detection in weighted networks is proposed by Zhou & Lipowsky (2004), and it runs in $O(n^3)$.

The method put forth by Palla et al. (2005) for the detection of overlapping communities[102] runs in exponential time, and it is therefore computationally hard.

The algorithm for evaluation of the best hierarchy fitting a network proposed by Clauset et al. (2007), is, according to its authors, definitely more computationally intensive than the classic ones, and realistically applicable only to networks with a number of nodes of the order of a few thousands[103].

Sales-Pardo et al. (2007) describes a quite similar algorithm with the same purpose, which, according to the authors, has a computational cost which "limits network sizes to $\sim 10,000$"[104].

Among the algorithms for detecting communities in *directed* networks, the one by Leicht & Newman (2008) is an extension to directed networks of the original method by Girvan & Newman (2002), and has the same time complexity, that is $O(n^3)$.

[99] See section 3.2.1.2.

[100] They claim that calculation for a network with 5000 nodes takes about a day on a standard (as of 2008) desktop computer. They also claim that the algorithm is trivially parallelizable, and so could be linearly sped up.

[101] See section 3.2.1.4.

[102] described in section 3.2.1.4.

[103] Clauset et al. (2007), p. 3 and 7. In the paper, there's no better explicit estimation of the time complexity of the proposed method . It is noted that an *exact* algorithm would involve a search through the super-exponential space of possible hierarchies, which has a cardinality $(2n-3)!!$, where !! is a double factorial and n the number of nodes in the network. This most assuredly makes the exact algorithm intractable. The proposed one, instead, makes use of a Monte Carlo method, which examines only a statistical sample of this space, ending up being only a probabilistic estimate of the best hierarchy, but definitely more computationally tractable than the exact one, albeit still computationally heavier than classic hierarchy detection algorithms.

[104] Sales-Pardo et al. (2007), p. 15227. The estimation matches quite well that of Clauset et al. (2007), which is not surprising, given that the algorithms work in a very similar way.

An algorithm for community detection in *directed weighted* networks, based on an alternative definition of Q, has been put forth in Arenas et al. (2008). In this method, Q_M if the *motif modularity*, defined as the density of *motifs* inside a community compared to the density of the same motifs in a random network with the same node characteristics. To assess this modularity measure, it must be specified which kind of motifs we are interested in: for example, by specifying as a criterion the density of triangular motifs, the algorithm detects as modules subnetworks whose nodes are densely connected by triangular, possibly overlapping, motifs. By specifying as a criterion the betweenness, that is the standard metric for modularity detection, this method obtains the same modules which would be detected by an algorithm based on Q[105]. Time complexity of this method, based on *motif modularity* is arguably very high, at least if what is sought-after is the *best* modular description of the system. This is because optimization of motif modularity is likely as computationally hard as classic Q modularity, which is NP-complete[106]. To the best of my knowledge, the hardness of motif modularity optimization has not been proved yet, but it seems plausible nevertheless, given the similarity of the two measures as defined in Arenas et al. (2008).

3.3.1.1 Accuracy of community structure detection

As we have seen[107], a problem which affects approximated algorithms like those mentioned in the preceding sections, is that, as Brandes et al. (2008) proved[108], they are not guaranteed to perform accurately enough to detect the optimal modular representations, because the approximation degree can vary wildly. This means that we can never be sure that the hierarchical modular structure found by an approximated algorithm is the best one, at least as long as we base the assessment of the quality of the modular decomposition on the measure Q, and the network is not trivially small. But, as we have seen, by its very definition it seems that Q is able to represent quite naturally the amount of what has since Simon (1962) been considered modularity, so it can be argued that the assessment of the reliability of the detected modularity should be based precisely on this measure.

However, appropriateness of the modularity measure Q itself has been put in doubt: it has been showed, by Fortunato & Barthélemy (2007), that even if optimization of Q *were* possible (and *it is not*, for it is NP-complete), the resulting detected hierarchy, in many cases, would not take into account small modules as independent communities, and would instead consider them as fused into bigger ones. In other words, the algorithmically detected modules could actually comprise other in principle well-defined submodules, too small to be caught by the algorithmic detection.

This *resolution limit* in community detection seems to be an intrinsic limitation of algorithms which work by trying to optimize Q, both precisely and by approximation. The only solution Fortunato ad Barthélemy envision, is to revise every modular description found, by examining every smaller module found, to see if it comprises even smaller proper modules, or to resort to a measure of modularity alternative to Q, on which, however, at the time of their paper, there was no clear consensus[109].

[105] This way, the algorithm by Arenas et al. (2008) implements modularity detection at different "levels of abstraction" (to anticipate a terminology which I will introduce in section 6.6).
[106] as proved in Brandes et al. (2008).
[107] Sections 3.2.1.1, 3.2.1.2, 3.3.2.
[108]. See also section 3.3.1.
[109] The authors even predict that other measures of modularity could manifest their own resolution limits,too. This has indeed turned up to be the case with the alternative measure proposed by Arenas et al. (2008).

3.3.1.2 Trade-off between accuracy and speed in community structure detection

In general, there is a trade-off between efficiency and precision, affecting approximated algorithms for finding community structure in networks. This seems not completely surprising, given that the task of Q optimization has turned out being NP-complete: if we adopt Q as a reliable measure of the plausibility of a detected community structure type modularity (and, as I have argued in section 3.2.1.1, this seems actually quite plausible, at least if we stick with the conception of general modularity I tried to reconstruct), then we will suffer the computational limitation in optimizing Q. We could certainly resort to approximated methods for this optimization, but of course those methods will most likely yield approximate results. Given that the approximated methods are usually chosen for their being computationally tractable, and thus much faster, it is to be expected that they will fall shorter on other criteria, and so it is not surprising there will be a trade-off between speed and detection accuracy. This trade-off could not be an absolute law, for newer, more efficient algorithms are continuously researched. Another solution would be to resort to a method which optimizes parameters different from Q, and whose optimization is intrinsically faster.

Faster, approximated algorithms than the classic Q-based ones by Newman and Girvan cited in the former sections have been found, which run much faster, as for example Newman (2004b) and the even more efficient Clauset et al. (2004), which runs in almost linear time[110], but, as said, the trade-off between speed and accuracy means that quite often these algorithms are less precise, in identifying actual community structure, than the slower ones: the fast algorithms are more likely to choose not the most optimized modular partition of the network, and at times they fail in distinguishing actual modularity from random fluctuation: it is typical that many algorithms find community structure even in random networks.

Another very fast method, proposed by Wu & Huberman (2004), runs in *linear* time with respect to network size, but is not capable of finding a full hierarchical structure, like the classic algorithms by Girvan and Newman: it detects only the smallest modules around nodes, and requires the number of communities to be known in advance, a property which renders it not really useful in many cases of modeling of real phenomena, when rarely the number of communities is already known.

In section 3.2.4.1 I have described some algorithms for community detection based on the network dynamical properties. The most accurate of these methods, proposed in Boccaletti et al. (2007) is in $O(n^2)$ complexity class, thus less demanding than many classical structural methods, and producing similar or even better results in terms of accuracy. This is probably, at the moment, the most promising method with respect to the complexity/accuracy trade-off.

3.3.2 Time complexity of network motifs detection[111]

The original algorithm for detecting network motifs, as devised by Milo et al. (2002) is extremely time consuming. The reason is that it involves counting all the occurrences of *all* 3 or 4-nodes subnetworks of a given network: this number can be as large as the number of *simple combinations* of k elements out of the set of all n nodes in the network, which is equal to $\frac{n!}{k!(n-k)!}$, and grows faster than $(\frac{n}{k})^k$. It is plausible that this limitation could not allow this algorithm to be run

[110] These algorithms try to approximate Q optimization by *greedy optimization*. See section 3.2.1.2 for a better explanation.

[111] The algorithms cited here have been exposed in section 3.2.2.

fruitfully on networks of more than 10^4 nodes[112], or even smaller networks, if what is searched for are motifs of more than 4 nodes.

As a remedy, algorithms that count motifs in a *sample* of the total network, in order to *estimate* the actual total number of motifs and their distribution, have been proposed, starting with Kashtan, Itzkovitz, Milo, & Alon (2004b).

However, the *exact* counting of the actual number of motifs in the network would be preferable. Better feasibility of this task has been achieved with an algorithm proposed by Itzhack et al. (2007), which is optimized to the point of being 2000 times faster than the classic algorithm for motifs of 3 or 4 nodes and networks of not less than 50.000 nodes, even faster than the sampling-based algorithms, while at the same time allowing for accurate analysis, with precise counting of the motifs, in networks of previously intractable size.

Another algorithm, devised by Grochow & Kellis (2007), takes into account motif symmetry during the enumeration of motifs, showing an *exponential speedup* with respect to the original algorithm proposed in Milo et al. (2002), and even to one of its improved versions. This new algorithm was capable of discovering a complex 15-nodes motif which occurs several thousand times in the protein-protein interaction network of *Saccharomyces cerevisae*. This constitutes an interesting finding, which results from a performance seemingly unattainable with previous methods.

3.3.3 Time complexity of network roles detection[113]

The papers cited in this work which propose algorithms for discovery of network roles do not explicitly assess the time complexity of these algorithms. However, we know that, as discussed in section 3.3.1, algorithms making use of simulated annealing are quite intensive from a computational standpoint, and most of the methods proposed for discovery of network roles make use of this stochastic techinque: specifically, both Reichardt and White's method for finding the best abstract model of a network, and Guimerà and Amaral's algorithm for community detection, which is part of their method to produce a functional cartographic representation of a network, recur to simulated annealing, and can thus deemed to be highly computationally intensive.

3.4 Summary of the survey on modularity detection algorithms

In the last sections we have examined many algorithmic methods for detecting various forms of modularity in complex networks, ranging from network motifs, to communities, to hierarchies of motifs and/or communities.

The precision limits of the different proposed algorithms and their computational time complexity have been highlighted. It turns out that this time complexity level, while being in most cases polynomial, and as such not theoretically intractable, is often high enough to hinder modularity detection in networks larger than a certain size. Actually, there are networks which, albeit quite big, in many cases are of great interest for many scientific disciplines: for example, it is normal to examine networks with millions of nodes when studying the World Wide Web, or social networks. In worse cases, depending on the precision of the chosen algorithm, even the modularity degree of networks with a number of nodes of the order of 10^4 can turn out to be difficult to analyze. This limitation could raise some difficulties in systems biology, where networks of such a magnitude

[112] See Itzhack et al. (2007).
[113] The algorithms cited here have been exposed in section 3.2.3.

(gene transcription networks, metabolic ones, proteic network, or a combination of different network types) are to be expected.

Nevertheless, it all depends on the precision in modularity detection *required*: many approximate algorithms are orders of magnitude faster than the most precise ones, and speed can in some cases be preferable to precision. Besides, the search for more efficient methods is going on at a fast pace.

For what concerns *community structure* detection, even if optimization of the modularity measure Q has been proved to be an NP-complete task[114], there have been attempts to devise algorithms for its approximate optimization, or to carry out modularity detection based on measures alternative to Q, such as the dynamical methods: the verdict about the actual feasibility of community and hierarchy detection on very large networks is still open.

Approaches based on *network motifs* have showed up as being quite computationally intensive as well, limiting feasible analysis to networks of the order of 10^4 nodes, and the sought-for motifs to no more than 5-nodes ones. Here too, the alternative is to accept a speed/accuracy trade-off by resorting to *sampling* methods, which are less accurate than exhaustive ones.

Nevertheless, recent proposals such as that put forth in Grochow & Kellis (2007) have exponentially raised the maximum allowable size of analyzable networks, permitting the search of functionally meaningful motifs composed of 10-20 nodes. This innovation seems an important milestone, and here too the question of computational feasibility of a modular functional description in terms of motifs of large complex networks remains open.

[114] See section 3.3.1.

Chapter 4

Modularity of computer programs

This chapter highlights the importance of modularity and related notions for computer programs. In order to clarify some basic concepts concerning computer programming and programs, a quite substantial introductory section (4.1) has to precede section 4.2, which is the part specifically dedicated to program modularity.

4.1 Computer programming

Differently from the more theoretical questions regarding the foundations of computation typical of computer science, whose treatment is dominated by mathematical rigor, in fields affected also by technical concerns, such as that of information technology, less philosophical attention is usually directed towards giving rigorous definitions of basic concepts. As a typical consequence, in these fields there is a lack of widespread consensus about the definition of certain fundamental notions. Some of the notions I will touch upon in this section, like those of *program* and *specification*, belong to this last category. However, I am not about to pursue here a complete philosophical examination of these subjects, but only to anticipate a series of concepts and questions which will be of some use in further theoretical parts of this work.

4.1.1 Computer programs

The intuitive notion of a computer *program* has naturally emerged almost since the original 1936 paper by Alan Turing[1] which started computer science by introducing the idea of Turing machine (*TM*, henceforth): as we have seen, a universal Turing machine (*UTM*[2]) is a TM which accepts as input the description of another TM and some data for this TM to act upon. It then proceeds to execute the same operations this same TM would perform on the given data. The symbolic description of a TM given as input to the UTM is a complete description of the TM to be emulated, that is, a complete description of its transition table. The transition table can be seen as a set of directives which instruct the TM on how to act according to the possible given circumstances (where a *circumstance* consists of each particular *configuration* of the machine, that is the <*current state*, *current symbol read*> couple). In other words, given the read/write ability of the TM, the transition table constitutes a *list of instructions* on how to operate on data.

[1] Turing (1936). See section 17.

[2] For TMs and UTMs, see section 17.2.5.

A *computer program* is a generalization of this concept of a list of instructions, applicable to any universal computational architecture. Computing architectures different from that of the original UTM have been conceived and physically realized since the late 1940s. All these hardware architectures have been proved to be computationally equivalent to the UTM, but have usually the advantage of being faster and easier to program than the Turing machine.

4.1.2 The Von neumann architecture

A typical architecture used in modern computers is the so-called *Von Neumann architecture*, named after its origin in seminal works by John Von Neumann and others in the '40s[3]. It is a form of *stored-program* computer, in which the list of instructions to execute is stored in main memory alongside the data on which they are supposed to operate. A *Central Processing Unit*, or *CPU*, which, in a way, takes in these machines the place of the Turing machine's "head", has read and write access to a memory that is constituted by an array of cells, each holding a symbol taken from a fixed alphabet. Usually, the basic alphabet with which the machine works is binary, comprising only two symbols, typically represented as 0 and 1. This choice of a binary alphabet has a practical side: it makes it easy to represent those two symbols in an electronic device by means of a coarse electrical property like the absence or presence of a certain minimum voltage between two conductors. It has also the advantage, being immediately interpretable as a form of base 2 numerical representation, of allowing for an easy implementation of boolean logical functions or arithmetical functions. In modern machines, more than one *binary digit* (*bit*) is usually contained in a single memory cell, so the alphabet from which the symbol contained in a cell is taken is wider than the typical alphabet of 2-3 symbols used in TM exemplifications: in a typical modern personal computer, a memory cell contains a string of 8 bits (a *byte*), which can represent 256 symbols[4]. Each cell has a unique numerical *address*, which identifies it. There is a sequential ordering of cells derived from this numerical addressing schema. But, unlike the original TM's memory (the "tape"), which allows only sequential access, this is a *Random Access Memory* (*RAM*): any memory cell can be accessed at any given moment with a single-step operation by specifying its address, regardless of the particular position of the cell that the CPU was scanning at the previous computational step. This is in striking contrast to the TM, where, to reach a particular cell, the head must move step-by step through the cells that stand between the current cell and the cell the head is directed to. As it is easy to guess, a Von Neumann architecture, in virtue of random access memory, is in general faster than the original UTM by orders of magnitude.

The *instruction set*, that is, the set of operations the CPU can perform, is fixed by the hardware specific architecture. Each instruction is represented in memory by a symbol or a string of symbols. A program is typically a sequence of symbols representing these instruction, interspersed with symbols representing the data on which the instructions are supposed to operate. The program is stored in memory in a block of sequentially disposed cells. The instruction set comprises instructions to fetch data from memory and store them on the CPU internal *registers*, as well as instructions to write data stored in registers to a specified memory location. Registers are limited blocks of memory separated from RAM and internal to the CPU, on whose content the CPU directly performs operations. The instructions performing these operations are typically a limited set of logical and arithmetical instructions.

To "run" a program, the CPU, starting initially on the first cell of the sequence of memory cells containing the program, examines each instruction of the program at a different timestep, and

[3] Von Neumann (1945). See also Priestley (2011), §6.1.
[4] More symbols can be obtained by combining two or more adjacent cells.

performs the instruction on the data stored in the cell or cells which sequentially follow the cell in which the instruction under consideration is located. After having completed each instruction, a special numerical counter, called the *program counter*, is incremented by a number of steps sufficient to make it point to the memory address holding the next instruction to be executed. As a consequence of this increment of the program counter, the CPU comes to virtually "position" itself over the cell pointed at by the counter (just like the TM head positions over an adjacent cell on the tape after each operation), and proceeds to execute the instruction contained therein. The process goes on until some instruction to halt the program is encountered. In absence of specific instructions in the program manipulating it, the program counter always increments sequentially at each computational step in order to point to the following next adjacent instruction, so the program's representation in memory is scanned and executed sequentially. There are, however, instructions for manipulating the program counter itself, instructions which thus make the CPU "jump", at the next computational step, to the location of any specified non adjacent instruction. Starting from that cell, the program will go on executing the next computational step. These *jump* instructions can be unconditional or conditional: *unconditional jumps* make the program execution jump to a certain RAM cell regardless of any other circumstances, while in *conditional jumps*, the jump is effected or not depending on the result of some prior logical or arithmetical operation acting on some data. These conditional jumps, or *branches*, are operations which influence the *control flow* of the program, that is the order in which its instructions are executed. It is this conditional jump capability the crucial property which gives a computing architecture the TM-level computational power.

As an example of computer program, the listing reported in fig. 4.1 is a simple commented program written for a Von Neumann-like architecture. The program is here reported not in its binary form, but in a symbolic notation, more readable than binary code, which is a form of *assembly language*[5].

```
0000 LDA #$00   Load 0 into the A register
0002 LDY #$06   Load 6 into the Y register, which is used as a counter
0004 DEY        Decrement counter  Y
0005 STA ($1000),Y  Fill memory location at address 1000+Y with the value stored in A
0008 CPY #$00   Compare the value stored in Y with 0
000A BNE 0004   If the two values are different, jump to location 0004
```

Figure 4.1: the listing of a computer program written in the assembly language of the *MOS Technology 6502* microprocessor.

In the listing in fig. 4.1, the first vertical column shows the number of the memory location containing each instruction[6], the second column the symbolic name of the instruction, the third the *arguments* of the instruction, that is, the data on which it has to act upon.[7]. The fourth column contains an explanatory comment, which is not part of the program.

The following would be the *binary* representation of this program in RAM[8], a representation which closely matches the actual physical representation of the above program in a computer:

[5] For other informations on assembly languages, see section 4.1.4.1. The assembly language in this example is specific to the *MOS Technology 6502*, a microprocessor used in many widespread personal computers till the late '80s.

[6] Cell addresses are not consecutive: each instruction would fill a memory cell (at least in the processor architecture used in the example), but some instructions are necessarily followed by the data they are supposed to use, and in these cases the data takes another cell or two.

[7] The numerical values in the listing are in *hexadecimal* code, which is a form of numerical coding base 16 typically used in information technology, more apt than decimal to represent values stored in *bytes*.

[8] In the left colum the memory addresses are reported.

```
0000    10101001
0001    00000000
0002    10100000
0003    00000110
0004    10001000
0005    10011001
0006    00000000
0007    00010000
0008    11000000
0009    00000000
000A    11010000
000B    00000100
000C    00000000
```

4.1.3 What a program is

I have given some examples of a program, but, as somehow expected, it seems there is not a universally standard accepted definition of *computer program*.

The Encyclopædia Britannica online gives the following one:

> *computer program*, detailed plan or procedure for solving a problem with a computer; more specifically, an unambiguous, ordered sequence of computational instructions necessary to achieve such a solution. The distinction between computer programs and equipment is often made by referring to the former as software and the latter as hardware[9].

According to Piccinini (2008):

> In computer science, a program is a list of instructions implemented by a concrete string of digits; 'executing a program' means responding to each instruction by performing the relevant operation on the relevant data[10].

An aspect highlighted in the Encyclopædia Britannica's definition, is that a program is supposed to be a *procedure* for *solving a problem*. This view of a procedure with a specific purpose can be considered as stemming from the foundational questions about computation itself, and namely from the Entscheidungsproblem, which demanded a *solution to a problem* in the form of a *mechanical procedure*: Turing gave the definition, with his abstract machine, of what a *mechanical procedure* is, and a *program* can be seen as a list of instructions which describes the *procedure* that has to be performed by the machine to *solve a given problem*.

To sum up, It seems that a program is, under one aspect a list, under another a procedure, and that it is also supposed to *solve a problem*. So, by its nature, a program is not the execution of a random list of instructions, but it has the feature of being capable, with its execution, to solve a *specific* problem: this property can also be rephrased by saying that a program has a specific *function*. I think this is its most problematic feature, and it will deserve further analysis in section 4.1.5.

[9] http://www.britannica.com/EBchecked/topic/130654/computer-program
[10] p. 314.

But, stemming from the very idea of the UTM, there is another facet of the notion of *program*, one which is related to *programmability*: the program, that is, the description of the TM which is to be emulated by the UTM, is supplied to the UTM as a string of symbols on the tape: for a universal computing machine, the program to execute is just *data*. As data, the program itself can be easily modified to be adapted to different necessities, while the actual mechanism of the UTM remains fixed. This is a basic distinction which afterwards has come to be known as the distinction between *hardware*, that is, the fixed mechanism of the universal machine executing the program, and *software*: the program to be executed[11]

4.1.4 Programming languages

The list of instructions which constitutes the program is expressed as a list of symbols. Apart from the specific arithmetical coding which was employed by Turing in his 1936 paper to represent the instructions of the UTM, any coding whatsoever into symbols can be established to this purpose. Typically, each instruction in the program is represented by a string of alphanumeric symbols.

In computer science, a set of strings constitutes a *language*[12]. A *program* is expressed as a sequence of elements of a language, that is as a sequence of strings. The language employed in the description of the program is a *programming language*.

There is a unbounded number of possible programming languages, but, given a specific computing machine hardware, there is usually a native language, the so called *machine language*, or *machine code*, which corresponds to the set of elementary instructions directly executable by the hardware in virtue of its constitution. Machine code is expressed as binary strings[13].

4.1.4.1 Low-level languages

Machine code is considered the *lowest-level* representation of a computer program, and it is relative to the specific hardware architecture of the machine in question. In general, the class of *low-level languages* comprises machine code languages and their slightly more abstract representation, the so-called *assembly languages*. These are languages constituted by symbolic alphanumeric names which directly stand for the actual machine code instructions, plus some other strings standing for constants, positions in the program, and other auxiliary structures. The relation between the assembly code and the corresponding machine code is usually one-to-one, so the level of detail to be specified when programming in assembly is the same that would be requested by programming directly in machine code, but the assembly listing of the program, being a sequence of alphanumeric strings, is more readable than machine code, which is usually expressed in binary numerical notation[14].

4.1.4.2 High-level languages

Actually, a computing machine can execute *only* programs expressed in its machine code. To facilitate programming, since the early times of information technology, many *high-level languages*

[11] Usually, with "hardware" we refer to any piece of physical equipment making up a computing machine or an extended system composed of the computer and its peripherals, and oppose the term to "software", which stands for "immaterial", symbolic data. I think the very difference between the two terms does not stand in their physical realization, but in the mutability of software with respect to the hardware. I will return on this point in section 6.6.

[12] As explained in the Appendix, section 17.1.

[13] At least in computers based on a digital two-valued architecture, which are the vast majority of existing machines. See section 4.1.2.

[14] For an example of an assembly program listing, see example in section 4.1.2.

have been devised. "High-level" means that such a language differs from machine code, in being a representation of coarser granularity[15]: a single instruction in a high level language usually corresponds to more than one machine code instructions.

Given that a computing machine can execute only its *machine code*, when programs are written in high-level languages the instructions expressed in the high-level language, the so-called *source code*, must be *translated* to machine code before being executed. There are two main methods to accomplish this translation:

- compilation;
- interpretation.

Compilation amounts to substituting (usually by means of a program, the *compiler*) to each high-level instruction in the source code a corresponding sequence of machine code instructions. This way, the obtained list of instructions is immediately a machine code list, and as such constitutes a program directly executable by the hardware. This program obtained by compilation runs as fast as an identical program written natively in machine code.

Interpretation is instead a more indirect translation: there is a program, the *interpreter*, which takes as input the high-level program listing, and, for each high-level instruction it encounters in the given order, it executes in real-time a predefined program which performs a sequence of machine code instructions equivalent to the prescribed high-level instruction. This continued real-time interpretation procedure ends up being orders of magnitude slower than the direct execution of an equivalent compiled program.

A high level program can be roughly seen as composed of: (*i*) a set of *variables*: labeled memory storage locations which can contain data; (*ii*) statements which assign data to variables by evaluating formulas comprising variables and logical or arithmetical operators; (*iii*) statements which prescribe some action to perform; (*iiii*) statements which read data from variables; (*iiiii*) *conditional statements*, which modify the control flow of the program (for example, jumping to another instruction) on the basis of the value of some variable or formula.

As an example, a program written in Javascript, which is a modern high-level language, can look like this:

```
var num1 = 660;
var num2 = 33;
while (num1 > 0)
{
    var tmpnum = num1;
    num1 = num2 % num1;
    num2 = tmpnum;
}
console.log(num2);
```

This program calculates the greatest common divisor between the numbers stored in variables *num1* and *num2*. The code between the two curly braces is a *block* of code. The *while* statement

[15] A high-level language can be considered a representation at a higher level of abstraction. I will discuss the notion of level of abstraction in section 6.6.

evaluates a condition (in this case the condition is true if the value of the variable *num1* is positive) and, if this condition holds, it repeatedly executes the following code block until the condition ceases to be satisfied. Statements like *while* are called *control flow statements*, for they influence the control flow. Another typical control flow statement in high-level languages is the *for loop* construct, which makes the subsequent block of code loop continuously, while a counter variable increases or decreases inside a prespecified range of values.

4.1.4.3 Syntax and semantics of programming languages

Informally, the *syntax* of a language is the set of strings which can be accepted as well formed statements and formulas in that language. Formally, the syntax of a language can be expressed as a formal grammar[16]. Most high level languages are context-free languages: languages which can be generated by a context-free grammar or accepted by a pushdown automaton[17].

Informally, the *semantics* of a high-level language is the sequence of low-level instructions corresponding to the translation into machine code (by means of a compiler or an interpreter) of the high-level statements and functions of that language.

The semantics of machine language resides in the *actual operation* performed by the *hardware* as a consequence of executing each machine language instruction. Being related to a specific hardware architecture, machine language needs specification of the hardware it runs onto, to give it a semantics.

4.1.4.4 Program semantics and flow charts

Intuitively, a general notion of semantics for a *program* is this: the meaning of a program, seen as a list of instructions, consists in the computations that a machine can perform when executing that program.

Flow diagrams, introduced by Goldstin and von Neumann in the late '40s[18], are a graphical means of representing the sequence of operations a computer carries on when executing a given program. They are thus a form of high or low-level representation of the program semantics. Nowadays, they are better known as *flowcharts*. For an example, see fig. 4.2.

4.1.5 Program specification and program implementation

A program is, as we have seen in section 4.1.3, a procedure aimed at solving a problem. When undertaking the task of writing of a program, the programmer starts with a first consideration of the problem the program is supposed to solve. This problem can be expressed in a more or less detailed way. The description of this problem, or, in other words, of the *function* the program will have to fulfill, is called the *specification* of the program.

This notion of *specification* is to be remarked here. Informally, a *program specification* is to be understood as simply some description of a program's functionality, that is, a description of what the program is supposed to do. This can be expressed, as said, as the problem the program is supposed to solve, or, in a less abstract way, as a description of the precise relationship between the program's inputs and its outputs. In different occasions, the specification is given in a variety of more or less abstract forms.

[16] For formal grammars, see section 17.2.10 of the Appendix.
[17] See section 17.2 of the Appendix.
[18] See Priestley (2011), §7.9.

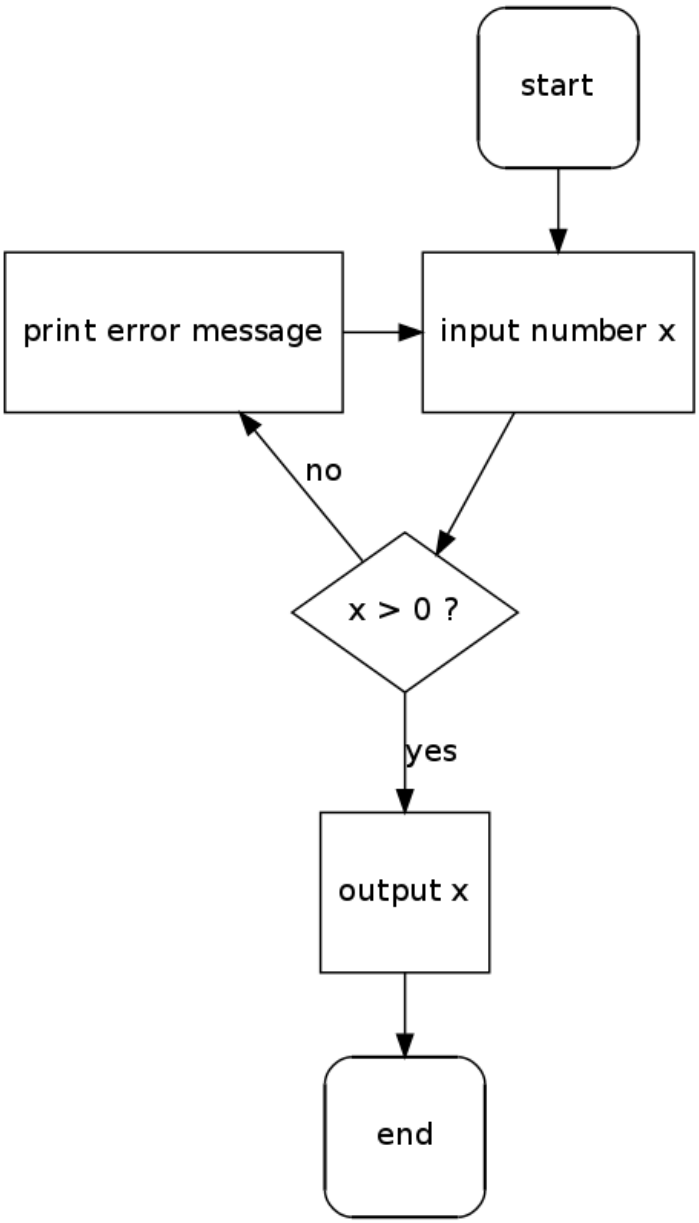

Figure 4.2: a flowchart describing an algorithm which accepts non-null positive input numbers.

A specification does not enter into the details of *how* the specified requirements are to be fulfilled: it is the *program* which specifies how to accomplish the required task.[19]

Regarding what a specification is, Galton (1993) writes:

> In computer science a specification is a more or less precise or exact statement of how a software system is required to behave. It provides a criterion against which the success or correctness of the software system can be judged. Minimally, a specification is a functional specification, that is, it specifies the system as a 'black box' in terms of what output it will deliver when activated by any given input. Nothing is said of, for example, how long it will take to deliver the output, and the specification is pure in the sense that nothing is said about how the generation of output from input is to be achieved.

A highly detailed specification can thus be seen as the description of a *function*, in the mathematical sense of a *mapping* between elements of two sets: in this case, between possible inputs and desired outputs implemented by the program.

An *implementation* of a given specification is a program satisfying the specification[20]. Of course, a given specification can be satisfied by *more than one program*: the same specification can be *implemented* in more ways than one. Implementation is the complementary concept of specification: for a specification there is an implementation, and an implementation, as such, is the implementation of a specification.

As said, a specification does not enter into implementational details. According to Galton (1993), though, this two level (implementation-specification) framework is too rigid: a specification is a specification with respect to an implementation, and an implementation can be itself seen as a specification with respect to another, more detailed implementation. Thus, the specification and implementation levels[21] are not absolute, but *relative*. This stems from the fact that, usually, during the development process of a program, the so-called implementation of a given specification does *not* initially implement all the details of the final program[22], but leaves some details aside, supplying in their place sub-specifications which in turn will have to be implemented. An example will clarify this point[23]: a given specification requires a certain list of strings to be ordered alphabetically (a classic *sorting* task). The implementation can be realized by more than one possible algorithm. Well known algorithms for this task are the so-called *insertion sort* and *merge sort*. Typically, such algorithms will be at first supplied as the descriptions, in a natural language, of certain procedures. This kind of description usually does not specify every detail: it can mention operations in the described procedure, such as "the element gets inserted in a list" or "the two lists are merged". Such very high-level descriptions of algorithms, despite being part of this first proposed implementation, can be seen in turn as specifications *themselves*, in turn in need of an implementation. The specification/implementation relation can then be applied recursively: quite often, further implementation of such lists of operations as the above descriptions of algorithms, is realized in a high-level language. And even a high level language can in turn be seen as a set of specifications (the high level instructions) in need of implementation, implementation which will be realized by a lower-level language (usually assembly language[24]).

[19] See Colburn (2004).
[20] See Bird & Wadler (1988), p. 13.
[21] My view on the nature of these "levels" is to be better specified in section 6.6.
[22] That is, all the details of the program which will eventually be executed in order to perform the wanted algorithm, implementing the original specification.
[23] The example is taken from Galton (1993).
[24] See section 4.1.4.1.

The last, lowest level implementation is usually considered simply an implementation and nothing else. This bottom level is not fixed and established, and depends on a pragmatic *choice* of the language (high or low-level) with which to implement the original general specification[25].

4.1.5.1 Specification, abstraction and naming

The above considerations suggest that we could envision an entire hierarchy of "levels" each of which is the *specification* of the immediately lower level, which constitutes the *implementation* of this specification. In turn this lower level is the specification of its underlying level, and so on, until the "bottom" is reached.

An important point is that a specification is always more *abstract* than its implementation: it is a description of a possible implementation which neglects some of the features of the implementation. More precisely, a specification is a representation which is abstract enough to be *multiply realizable*: by ignoring some of the features of its implementation, it leaves open the possibility that the *other* facets of the implementation, the ones that it has neglected, vary freely, and this leaves space to possible different versions of its implementation, versions differing in the features that are neglected by the specification. A specification ends up representing only the *common* features of its possible implementations.

In other words, a specification is basically an *abstraction*, where the concept of abstraction is to be understood as a representation which represents only *some* aspect of an object of observation, not all its aspects[26]. Any abstraction can, per se, be a specification: the very fact that it is an abstraction makes it susceptible to be *implemented* in more possible particular ways. Of course, the more abstract a specification is, the higher is the number of its possible implementations. A specification can then be more or less *strict* as a function of its degree of abstraction: a very strict specification does not leave room for many different possible implementations[27].

When taken to a high degree of abstraction, that is when ignoring most aspects of its implementation, the specification can turn out to be so synthetic as to be useful as a simple "name" for the function realized by its implementation. Abstractions of this kind are "addition", "printer", or similar cases in which a very short expression, which abstracts a crucial feature of a system, is adequate to "name" the system in an "aggregate" manner: in a way, by naming a crucial feature of the system, it summarizes in this feature all the complex of activities which constitute the whole functioning of the system.

There is some exception, though, to this idea that a specification is an abstraction. For example, this happens when a specification understood as the *name* of the input/output function of a program, is constituted by an an arbitrary chosen verbal expression assigned to the program, as its name, purely *by convention*. To make an easy example, take Google, the well-known search engine. What is "Google"? It is the name of a specific program, but this name is not an abstraction of the input/output function that the program performs, which can legitimately represent it in an aggregate way: it is a name applied to it by pure convention. On the contrary, a

[25] This situation of a choice determining the bottoming out of a recursive hierarchical descent recalls quite closely (most likely not by chance) analogous situations regarding mechanistic explanation (section 11.1.5) and modularity detection (section 2.3), and is strictly related to some of the considerations on levels of description that will be carried out in section 6.6.

[26] I will try to clarify the notion of abstraction in section 6.6.

[27] I'm speaking here on an informal, intuitive level of the *number* of possible implementations: it can be suspected that the number of possible features of any object is infinite (but what kind of infinity?), and on this basis it can be argued that, for any specification, however abstract, the set of its possible implementations has always the same cardinality. However, I'm not even trying to embark here in this kind of discussion, which is well outside the scope of the present work.

very high level abstraction of the function Google performs would banally be the following way of naming it: "search engine". This expression could legitimately (and probably does) constitute a valid aggregate representation of what Google does. The other way of naming Google's function, that is, naming this function by calling it "Google" is purely a verbal label attached to the actual Google specification, with specification intended here in the sense of Google's input/output function. Curiously, (and, probably, interestingly), at least in Google's case this mere "labeling" of the program specification has (for social reasons, in this case) come gradually to constitute a specification of Google's function also in the sense of becoming an *abstraction* proper of this function: to see this, it is sufficient to consider that, these days, as an answer to the question "what are you doing on the web?" one is socially legitimated to answer "I'm googling for...". In this case, it seems that the verb "to google" has assumed a functional meaning which represents a very high abstraction of what the program referred to by the name actually does: "googling" is now undoubtedly a synthetic way to say something on the lines of "searching the web with a certain search algorithm, which indexes the text of the web and searches in the indexed data a set of strings the user has specified..., [and so possibly on until the full functional specification is given]". But, this "reversion" from the purely *conventional* label to a form of abstraction occurs only for social reasons. When the use of a conventional label becomes widespread, the word, per se apparently devoid of meaning, inherits, legitimately from a social standpoint, due to its constant association to the function it labels, the functional meaning of this labeled function: initially, "Google" did not mean anything, now it is a verb and means "searching the web". This is an abstraction of the actual function of Google's program, while originally the same word stood only for a conventional label.

4.1.5.2 Kinds of specification

Based on considerations and reflections like the ones above about the possible senses of the notion of specification (input/output function, purpose of the program, "name" of the program or of a function), and on the most usual ways *names* get attributed by programmers to functions and program they write, I have come to a tentative classification of the meaning of "specification" and "name" of computer programs, which in my opinion can help avoid some misunderstandings about this notion. I classify specifications into four main "kinds", as explained in the next paragraphs.

In the first three of these classes, specifications or names are seen as *abstractions* of the actual input/output functions characterizing the set of all their possible implementations. The fourth class sees a name as something which is actually only a verbal label attached to a set of implementations. This fourth kind of function and specification naming is a degenerate one, albeit a quite frequent one in information technology practice, and does not help very much (except by allowing for some economy of description of programs) in analyzing, understanding and explaining the function of programs, an activity which is very important for programmers, as we will see in section 4.3.

It must be noted that some of the distinctions below prefigure or constitute the "computational" equivalent of certain notions of functions belonging to explanation in empirical sciences, and which are studied by philosophy of science, notions which will be discussed in section 9.

4.1.5.2.1 Kind B: bare specification

- B: *bare specification*: specification as a *mathematical* function holding between possible inputs and the corresponding outputs, provided extensionally as a set of input/output couples.

4.1.5.2.2 Kind A: the aggregate kind

- A: *aggregate name*: an abstract representation of the specification understood as in B. If this abstraction is of sufficiently high degree, then it can be seen as the "name" of the specification understood in sense B, and it can act as an aggregate placeholder instead of the bare specification. There are two ways to obtain this kind of abstraction:
  - A1. *Autogenous*: the name is an abstract way to name the input/output function. This can be obtained by devising a proper "aggregate value" of the function, for example by approximating it, or by taking an already at hand name for a well known function, such as in the case of calling "addition" the function which adds number, or as in the case of "absolute value".
  - A2. *Contextual*, or *relational*: this is a way of naming a function understood as in point B by citing the *role* which it fulfills inside a larger context, a context which can be implicit or explicit. In this case, the specification so understood is indeed an abstraction, but it is not an abstraction which picks out some feature of the specification as function of point B: it is an abstraction which picks out a *relational* feature of its implementation, that is the *role* that at least some of its implementations fulfill inside a larger system into which they can be in certain circumstances inserted. This is not an *intrinsic* feature of the input/output relation. For example, saying that something is a heart is giving the specification of any system which fulfills the role of pumping blood in some organism: this way of naming such a kind of system is an abstraction picking up this *relational* role of the system.

4.1.5.2.3 Kind M: the modular kind

- M. *modular specification.* This type of specification is an abstraction in the form of a modular aggregate representation of the class of its underlying implementations. it can be of varying degree of abstraction, and, accordingly, is classified into two classes:
  - M1. *composite name*: these specifications are modular abstractions, but their expression is synthetic enough to allow for their use as aggregate "names" of the classes of their implementations. Examples: "search engine", "merge sort".
  - M2. *aggregate modular*: these specifications are modular abstractions, more complex and articulated than M1 class and for this reason they cannot be used as names. They can be of any degree of abstraction. Typical examples are most of the intermediate levels of hierarchical modular description of a system. This kind of specification can show any degree of detail, often appearing more naturally as an implementation than a specification: for example, a program written in a high-level language is a specification in this sense, with respect to its machine-language implementation, but it is usually seen already as an implementation of the more global specification of the program.

It is important to note that both M1 and M2 types can in general be seen as possible implementations of A1 or A2 specifications. Examples: "if negative, then change sign, if positive leave unchanged" is an M2 specification which can be seen as an implementation of the "absolute value" A1 specification. "search engine" is an example of a M1 specification which can be seen as the implementation of an A2 specification. This shows that the A kind is the more abstract kind of specification.

Another point is that both M1 and M2 kinds can be used as substitute specifications of B specifications. This happens in the case of analytic formulas standing for mathematical functions: for example $y = 3x+2$. In this case, an M2 specification is used in place of the B equivalent (which cannot be given in a finite form). This is a very important case, which shows the fruitfulness of a modular approach to the expression of specifications, a fruitfulness which will be highlighted in the rest of this chapter.

4.1.5.2.4 Kind C: the kind by convention

- C: *conventional name*: this type of specification acts as a "name" for the class of its implementations, but it is not an actual aggregate name for it, because it is *not* an abstraction of the members of this class in any way, in the sense that it does not pick any feature of its implementing systems. It can however in part act as a placeholder to be used in a coarse-grained, apparently more abstract description of a system, even though is does not possess any explanatory power, because, being simply an arbitrary linguistic expression which gets associated to the class of its implementations *by convention*, it does not convey informations on the features of its implementations. This use of conventional labels can be recurred to in three different circumstances:
  - C1. *lazy attribution*: a functional module has a perfectly comprehensible specification, but this is a bit too "complicated" to be summarized in a word or two: the programmer or the observer renounces for lack of imagination to recur to an A1, A2 or M1 specification, and, for lack of a better way to name the function by abstraction, simply attaches it a label whatsoever. Examples: "Firefox" and "Chrome" as names of web browsers, or "Dolphin" as the name of a file manager.
  - C2 *forced attribution*: the function performed by the module under consideration is *very* complicated, and it is not plausible to expect that, with some effort, a sensible way of summarizing it can be given. So, the observer is *compelled* to give the function a purely conventional, concise name. Example: "Julia set", a fractal named after its discoverer, without any reference to features of the named function. The set appears as too complicated to allow for its synthetic description or naming (see fig. 4.3).
  - C3: *pointless attribution*: the attribution of an arbitrary label to a specification is made for pointless reasons, on a voluntary base, even if a significant abstraction could have been easily found. Example: "Google" as the name of a search engine. As noted above with the example of Google, even such a silly use of a name could, for social reasons, acquire with time a meaning which is in some way explicative of the function which the implementations of such a specification realize.

4.1.6 A common definition of computer program

The considerations reported above are only a few possible examples taken from a spectrum of diversified and contrasting theoretical positions which can be found in the literature of philosophy of computing, about the nature of computer programs and program specifications.

However, it seems in general that a minimum consensual nucleus of the concept of computer program comes to comprise the properties of *list of instructions*, *programmability*, *function* and *specification*. I am going to adhere here to this general conception, by proposing to adopt this basic definition:

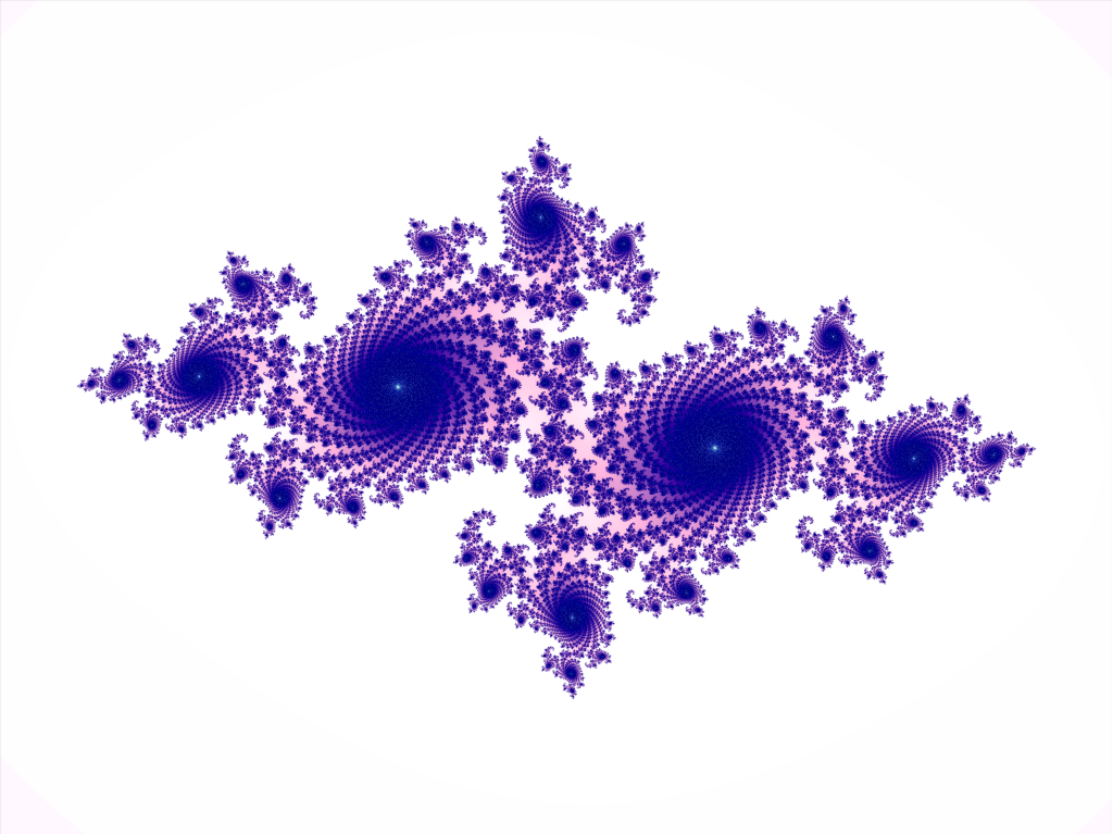

Figure 4.3: a representation of the *Julia set* fractal.

> a *computer program* is a modifiable list of instructions, to be executed by a given type of computing machine, which performs a function described by its specification[28].

4.2 Program modularity

4.2.1 Subroutines

Since the beginning of the '60s, computer programmers started to realize that often, in large enough programs, there are very similar subsequences of instructions which occur in different parts of the program. This seems to constitute a waste of memory space and a waste of programming time. A first solution was realized with the introduction of instructions which allow for the definition of *subroutines*[29], parts of the program which can be *called* from anywhere else in the program: a *call* to a subroutine makes the control flow[30] jump to the beginning of the list of instructions constituting the subroutine; then, the subroutine gets executed, and when its

[28] I'm not going to start in this section a philosophical analysis of the notion of function. This issue will be treated in section 9. Here, *function* can be understood both informally, as something the program is supposed to do, and more formally as the specification of possible input/output couples.

[29] I use here the word "subroutine" as a generic term to indicate what in many high-level languages can also be called "procedure" of "function".

[30] See section 4.1.2.

execution terminates, the control flow of the program returns to the step immediately following the call instruction. The subroutine can be called any number of times from any part of the program[31], so this constitutes a significant saving of memory and programming time, because it avoids the need to repeat the writing of the same sequence of instructions in different parts of the program. A subroutine can be intuitively seen as a kind of "module", which constitutes a subprogram implementing a specific functionality independently from the rest of the main program. In section 2, I touched upon the concept of modularity in general, trying to make explicit the properties of what can or can not constitute a *module*: it can be immediately recognized that, at least for its being a substantially independent section of code which can be re-used in different parts of the program, a subroutine comes to constitute a module according to the general definition given in section 2.1.

4.2.2 Structured programming

Starting in the late sixties, awareness of a potential problem with the high level programming languages used at the time began to become widespread among computer programmers. Conditional and unconditional jumps in the program sequence risk to make complex programs an intricate web of references to other, often distant, parts of the program (a form of so-called "spaghetti code", in hacker terminology). This has the consequence of making the process of understanding a program, of discovering errors in it (the so-called "bugs"), modifying the program to bring about improvements or extensions, and in general maintaining the program, a very difficult task: the spread of crisscross references from various parts of the program to other ones entails that an alteration of one part of the program often requires an alteration of one or more other parts. The same difficulties arise with comprehension of the program's structure, and with developing of complex programs of this kind when they are not written by a single programmer.

A famous 1968 letter from Edsger Dijkstra to *Communications of the ACM*[32] is considered the official beginning of the *structured programming* paradigm, a programming style which stands in opposition to the style described above, dominant till then, a style criticized for making indiscriminate use of jumps. Dijkstra states that the use of the *GOTO* statement (the instruction which usually stands for a jump in high-level programs) is harmful, and that programmers should structure their programs in another way. Even the extensive use of subroutines in *low-level* programming would probably not free the programmer from the need of resorting to jumps: the only way to execute a cycle in most machine codes was by the use of a conditional jump based on the value of a register interpreted as a counter.[33] *High-level* languages, however, usually possess certain constructs, such as the so-called *while loop*[34] that do not require, when programming *in the high-level language*, to recur explicitly to GOTO jumps in order to program a loop (albeit such a high-level construct when interpreted or compiled gets actually translated into a machine code loop making use of jumps). The point raised by Dijkstra was that at the time, even with the expanding use of high-level languages which had already started in the '60s, the problem of GOTO was still affecting large-scale programs written in such languages, because, for reasons tied to acquired bad habits, programmers continued to make extensive use of that statement. Structured programming instead requires that the programmer makes exclusive use of the high-level looping statements for the required control flow, as well as extensive use of subroutines whenever possible. This way, any kind of program can be written without recurring to GOTO

[31] A subroutine can even be called from inside itself. This constitutes an example of the so-called *recursive programming* style.
[32] Dijkstra (1968).
[33] See the low-level example program in section 4.1.2.
[34] See section 4.1.4.2.

statements. That the approach proposed by Dijkstra can be universally applied, was guaranteed on the basis of the Böhm-Jacopini theorem, proved in 1966[35], which states that any flowchart, that is a graph representing the control flow of a program, is equivalent to another flowchart constructed by making use only of conditional statements, subroutines calls and loop structures: namely, the program ends up being made of sequences of non-control instructions, conditional statements (the so-called *if-then* constructs), iterations of blocks of instructions by means of *for* or *while* loops, subroutine definitions and calls to subroutines.

Such structured programs show a kind of *modular structure* due to the separability of blocks of the program performing different functions: the avoidance of the GOTO statement guarantees that there are no "links" inextricably tying together distant parts of the program, a circumstance which, if present, would not allow for this separation of the program code into partially independent blocks.

4.2.3 Object-oriented programming

In the structured programming sytle, the program's structure can be virtually separated into blocks, and this constitutes a form of program modularity. But each block can still manipulate variables recurring in other parts of the program, and so the avoidance of GOTO statements does not guarantee the complete functional separability of blocks of instructions. Program modularity has continued to improve in different ways in the late '70s and in the '80s, with the advent of the *object oriented* paradigm. Object oriented programming prescribes to subdivide the program in a series of modules (*objects*), each of which *encapsulates* (the *data encapsulation* principle) the variables which it makes internal use of, and *hides* (the *information hiding* principle) those same variables from influences coming from parts of the program external to the object[36]. This way, the autonomy of modules increases, as well as the integration between their internal constituent. This is the realization of what Dijkstra (1982) calls a "separation of concerns", a circumstance which renders the program more easily analyzable and modifiable.

It is typical of a software object to implement information hiding by *denying* direct manipulation and access of its internal variables to parts of the program external to the object: an external part of the program can read and set values of these variables only by calling specific subroutines *of the object*, which are exposed to "public" use: calling one of the publicly exposed subroutines of an object constitutes the way to provide *inputs* to the object, and in turn the called subroutine of the object provides, after having completed its execution, an output to the part of the program which had called it. We can then view an object as a "black box" with an input channel and an output channel, where what stands between the inputs and the outputs (that is, the internal parts of the box) is not directly linked to other parts of the program: this way, an object can be considered a separate module, linked to other modules only by its input and output connections. This can be viewed as realizing the property of the general conception of modularity (as expounded in section 2.1), which consists in considering inter-modules ties as basically *weaker* than links between the modules' internal parts: in most program modules performing interesting computations, it is arguable that the number of internal interconnections (in the form of instructions accessing variables in order to read, set, and manipulate the variable's values) is higher than the number of input lines. Taking as metric the metric of density of interconnections, the program module would thus show weaker connectivity toward the external context. Even when this condition does not hold, it is arguable that a minimally interesting processing of the input on the part of a module would take a certain amount of time steps, during which the *state* of the input and output

[35] Böhm & Jacopini (1966).
[36] Information hiding was first proposed in a seminal work by David Lorge Parnas (Parnas 1972).

lines, which are not involved in the internal computation, remains stable. Only when the module has finished its computation, it produces a new output and returns control to other parts of the program: during the execution of the module's internal procedure the state of the rest of the system remains stable. This makes the case that, at least for a minimally complex computation performed by the module, on average its input and output variables[37] change less frequently than the module's internal states change. If we take as a metric of strength of connection the *frequency* of information exchange, then the input and output external connections of the module end up being, on average, more weak than its internal ones. Another way to put it, is that, at least in classical computer architectures like the von Neumann model[38], where computation is performed by the *sequential* execution of computing steps, it is most likely that the subroutine calls, which represent *inter-module communication*, occur during the program execution definitely less frequently than the execution of sequences of other instructions, which represent *intra-module interactions* between instructions and variables. This can be easily seen as the manifestation, as also highlighted in sections 2.2.4 and 6.7, of the timescale decoupling between intra-module and intermodule dynamics, which is a typical signature of modular systems.

This way, the general definition of module is clearly fulfilled by software objects. To assess software modularity, we could also take into consideration the relation that associates instructions with the variables they manipulate: in a modular program constructed in accordance to the object oriented paradigm, variables employed by a module are accessed exclusively from inside that module, so instructions manipulating the *same* variables are certainly instructions internal to the *same* module. This is the metric which is used in program slicing, a technique for detection of program modularity, which is described in section 4.3.4.

4.2.4 Program modularity, coupling and cohesion

It seems that in general an increase of modularity means a decrease of the structural dependencies *between* modules, together with an increase of *internal* module cohesion. In the information technology jargon, these two properties are described as *coupling* and *cohesion* respectively, two terms introduced in the seminal book Yourdon & Constantine (1979)[39]. Thus, an increase in modularity means a decrease in coupling together with a simultaneous increase in cohesion. According to the modular programming paradigm, *high cohesion and low coupling* is precisely what is sought for when designing program architectures. All in all, this conception of modularity employed in software engineering appears quite close to the more general view of the notion developed by Herbert Simon of modularity as near-decomposability: near decomposability means that a system is, up to some approximation, decomposable (low-coupling) into highly cohesive structures.

4.3 Reverse Engineering and modularity detection in computer programs

As we have seen in section 4.2, usually computer programs are *by design* structured in a modular fashion. There are, however, some techniques which aim to detect structural and functional modularity of *already written* computer programs, without taking into consideration any information about the modularity which the programmers could have given to the program structure during development. Quite often, the reason for this is that such informations have been lost, or have

[37] That is, the module's variables which are accessible via its publicly exposed subroutines.
[38] See 4.1.2.
[39] See also Laplante (2007).

been hidden from public knowledge: the examiner is confronted with a working but mysterious program, possibly not equipped with readable source code but provided only in the machine code form, and has to infer its internal functioning and, in some cases, even its more general specification[40]. There is an established discipline which deals with such daunting tasks: reverse engineering.

Reverse engineering can in general be seen as the practice of taking an already built technological device of some sort, and of applying to it a series of analytical techniques aimed at understanding the nature and functioning of the device, usually in order to reproduce it.

As of today, an organic corpus of literature on the subject in general does not seem to exist. This is due to various factors, not the least of which is the fact that reverse engineering techniques have traditionally been employed in military applications, and as such have been often surrounded by secrecy. Nevertheless, in computer science the subject of reverse engineering of *software* is more developed, stemming naturally as an evolution of the problem of *debugging*, the practice of finding errors (*bugs*) in a program, a task which can be seen itself as a form of reverse-engineering.

There has however been some attempt to define the subject matter of reverse engineering in general: Rekoff (1985), in the context of reverse-engineering of computer *hardware*, gives the following definition:

> Reverse engineering is defined here as the act of creating a set of specifications for a piece of hardware by someone other than the original designers, primarily based upon analyzing and dimensioning a specimen or collection of specimens[41].

Here, *specification* is to be understood as a description of what the piece of hardware does. But, the process can be also applied to software. In this case the goal is, as Galton (1993) puts it, "to give an exhaustive characterization of the input/output relation defined by the program"[42].

A process of reverse-engineering turns out, unsurprisingly, being the exact *reverse* of the process of programming (which is "forward" engineering): while in computer programming one tries to correctly implement an already given specification, which is usually a very high-level description of what the program is supposed to do[43], or, less often, a formal description of the input/output function that the program will have to produce[44], in reverse engineering the inverse route is followed: based on the static observation of the code of a program, or on the behavior of the program consequent to certain inputs applied to it, an attempt is done to infer the specification that the program implements. But, as explained in section 4.1.5, as noted by Galton, the status of specification or implementation is *relative*: a given description can be seen as a specification, which has to be implemented, but the implementation itself can be seen as a specification with respect to a possible even more detailed description which constitutes its implementation. The relation specification/implementation turns then out being a relation between levels of description[45], and as such is relative to the choice of those levels. It is then conceivable that reverse

[40] That is, even the *purpose* of the program must at times be inferred, in the form of an at least minimally useful abstract or partial specification of its input-output function.

[41] p. 244.

[42] p. 115.

[43] As more thoroughly explained in in section 4.1.5.

[44] It can be sensibly objected that a full I/O specification is almost never supplied to the programmer, in the real-world of software development: often, specifications given to the developer by their directors are so vague that it is difficult to test the resulting program's compliance with them. I am treating here idealized cases.

[45] A more rigorous definition of what I call here "levels of description" will be attempted in section 6.6.

engineering can be facilitated by performing it first in a bottom-up direction, across *progressively ascending* levels of abstraction, in a reverse fashion with respect to functional analysis or mechanistic functional decomposition[46]. This way, one of the potential hierarchical structures describing the program is gradually found.

With regard to reverse engineering of *software*, Chikofsky & Cross (1990) states:

> Reverse engineering is the process of analyzing a subject system to
>
> - identify the system's components and their interrelationships and
> - create representations of the system in another form or at a higher level of abstraction.[47]

According to Beck & Eichmann (1993), reverse engineering of software subdivides in:

1. *Algorithm recognition*: extraction of behavior from the implementation.
2. *Design recognition*: extraction from the algorithm of the rationale for behavior.

The first point consists in inferring from the program low-level implementation (be it in the form of the source code, the machine language code, or its dynamical behavior) the algorithm, which is a representation of what the program does, described at a first more abstract form than the actual machine code. The second point infers from the algorithm its specification, that is the actual global input/output function of the program.

As expected, this description of reverse engineering suggests that this is a practice similar in some way to the search of a hierarchical description of a system, a form of hierarchical modularity detection: what is sought for is to find progressively higher levels of description of the program. One could ask why the process of reverse-engineering should follow only this bottom-up direction. In actuality, reverse engineering can also be applied to systems of which the global specification is already known, with the purpose of understanding *how* they work, in order to modify them: what is sought for in this case of reverse-engineering is in other words an *explanation* of the system[48], and explanation is, incidentally, what is sought for in science as well. In that case, the task to carry out is a progressive decomposition of the global function, which is already known, into a set of subfunctions which, in their coordinated interactions, produce that global function. This top-down route is precisely what a Cummins-style functional explanation, a type of explanation which will be expounded in section 9, amounts to: Robert Cummins' functional analysis is a way to explain *how* the system's global specification (which explains *what* the system does) is fulfilled in terms of the organized composition of a number of "less problematic" (as it were) specifications (functions), which are the specifications of its submodules. When, however, as in the views on reverse engineering exposed above, the global specification of the system is not known in advance, then reverse engineering must proceed in a *bottom-up* fashion in order to obtain that specification, that is a description of what the system globally does. The global specification can be given at various, chosen levels of abstraction and degrees of approximation, as already discussed in section 4.1.5.1, but in its least approximate form it amounts to the full specification of the input-output function of the program. For a program with an unbounded

[46] For these two kinds of analysis employed in mechanistic and functional explanation, see sections 10 and 9.
[47] p. 15.
[48] This would be a form of functional, and specifically *computational* explanation. These kinds of explanations are treated in sections 9.2 and section 14.5.1.

input size, and for which therefore the exhaustive enumeration of all the input/output couples constituting this specification is in principle impossible, the problem of inferring its specification is in its essence the problem of induction, and this summons a constellation of related thorny problems, the most prominent of which are the "kripkenstein" rule-following paradox introduced in Kripke (1982) and the problem of the "grue" predicate proposed by Nelson Goodman. This cluster of riddles could indeed raise a host of questions related to the subject matter of this work, but discussing them in detail lies outside the limited scope of this treatise, so a thorough discussion of the problem is left to a better occasion.

4.3.1 Reverse-engineering of program specifications in modular programs

When the observer lacks the overall specification of a given program, in order to obtain this specification one could think that a *direct* bottom-up way could be attempted, a way which works by trying to infer the specification directly from observation of the whole program code or of the program's behavior.

Let's now evaluate this possibility. First, we could try to infer the program specification by statically examining the program's code *structure*, if it is available. For serial von-neumann-like architectures, this listing is constituted by the source code of the program, possibly expressed in assembly code. In the worst cases, the actual executable program code, in the form of machine code, will always be available, and this should in general be easily convertible into assembly code[49]. It must be highlighted that reverse engineering of a program's specification based on the analysis of its source or assembly *code*, the so-called *static analysis*, is hindered by the unsolvability of the halting problem[50]: should it be always possible to infer, following some well-specified method, the *complete* specification of any program, that is, its complete input/output mapping, by just examining the program's instructions listing, then we would be able to discriminate non-halting machines from halting ones, because non-halting machines are the ones with an only partial input/output function: a function whose value is not defined for certain input values (the values for which the machine never halts, and for which, thus, it does not produce any output). But, since isolating the set of non halting Turing machines by looking at their code (the machine representation) has proved to be an impossible task, then *in general* reverse engineering based exclusively on code examination is in principle bound to fail. In other words, it cannot exist an algorithmic method which, applied to a given program whatsoever, guarantees to find its complete specification by looking at its code, although of course, in many *specific* cases, with ad-hoc methods, finding the specification should turn out to be feasible.

A potentially more promising way of trying to infer the program's global specification is by subjecting it to *dynamic analysis*, that is, the observation of the program's behavior (its output), as a function of the inputs supplied to it. But, even in this case, we already face a difficulty: given that some programs certainly never halt when fed with specific inputs, because they enter infinite loops, then how can the observer come to know that the observed program has not entered an infinite loop? The possible circumstance that the program is not yielding any output after an

[49] See section 4.1.4.1. In principle, static analysis could be based not only on assembly or high-level source code of the program, but on the binary string constituting the machine-language executable representation of the program. This is usually unnecessary, given that, based on the specification of the CPU architecture, (see section 4.1.2) which usually is already known, an assembly listing of the program can be straightforwardly obtained starting from the machine language executable. If not even the specification of the processor on which the program is supposed to run is known, then static software reverse engineering is hopeless: a machine language executable of the program cannot be given any semantics without knowing which computer architecture it is supposed to run on. In these cases, a reverse engineering of the hardware must precede the software one.

[50] See section 17.2.6.

inordinate amount of time since supplying it with a certain input, could mean that the program has indeed entered an infinite loop, but it could very well mean that the computation the program is performing on that input takes a due (even if inordinate) amount of time. We already know that this doubt cannot be guaranteed to be dissolved in advance for any observed algorithm by conducting a static code analysis, because of the undecidability of the halting problem. But, even when observing a *running* program, how can the observer know if the program will ever come to a halt, when limiting the observation only to the program's inputs and outputs without looking at the *internals* of the program? The only hope the observer has to solve this doubt is indeed by means of observation of the values of the internal variables of the program and its internal control flow in order to infer if the program's internal state is progressing in a non-periodic way toward a possible end, or if it is cycling indefinitely through a set of states. But, even in this case, it would be practically impossible to assess the entering of a cyclic behavior by observation of the program's internal values, in case the period of the loop the program has entered is enormously long. And this circumstance could not be excluded at all, given that the maximum period of a looping program is proportional to the number of its possible internal states, which itself is an exponential function of the number of internal variables employed in the program. Overall, the one highlighted here is a drastic hindrance to the obtaining of a complete specification by only looking at the program's input/output behavior.

But, let's say that we come to *know* in some way, possibly by observation of the program code's structure, that the specific program under observation is guaranteed to always halt, for any input[51]. How could we in this case infer its specification, that is, its complete input/output function? The simplest idea is again that of subjecting the program to all possible input values, and of observing the corresponding outputs, in order to infer the complete input/output specification. In this case, though, another difficulty must be faced, which is, in its essence, the same difficulty of scientific induction: the set of all possible inputs can be infinite, or, if finite, this set is often too vast to let us hope that every possible input gets supplied to the program in order to allow for an inference of the complete input/output program specification: even in finite input sets, the number of possible input combinations grows exponentially with the number of variables constituting the input of the program. For example, if the input is constituted by a binary representation, let's say in the form of binary strings of a fixed length, it is a basic result that the cardinality of the set of binary strings of n bits length is equal to 2^n. Except with respect to programs which can accept only a very limited string length in their input, exploration of the entire input space is impossible, for it is a $O(n^2)$ problem, and thus an intractable one[52]

However, If a *hierarchical*, *modular* representation of the program could be devised by some means, it would be possible to *decompose* the program in its modules, and test each module *separately* to seek for the specification of only that module. This task would most likely turn out being more feasible (by order of magnitudes) than that of submitting every possible input to the whole program in order to directly infer the global specification, which is, as we have seen, an intractable task. The reason for this simplification is quite obvious: it is known, by the

[51] This is not an unreasonable expectation: in many cases examination of a program structure's can allow for the proof that the program ends for any input. This is not forbidden by the undecidability of the halting problem. What Turing's theorem proves is that it is impossible to expect to find a general method always able, in any case, to came to the conclusion, by simply observing the code of a given program whatsoever, that the given program will halt on any input. But, if we study a specific program, we could possibly be able in many cases to decide if it will ever fail to halt: for a trivial example, if we observe that the program's code does not comprise any conditional or unconditional branch, then we can be sure that the program will always halt, because the only circumstance in which a program does not halt, is when it enters a loop, and a loop needs a jump.

[52] For example, exploring the space of 64-bit strings already requires the observation of $1,84467440737e+19$ cases.

definition of generic modularity[53], that a module is identifiable by the very fact that it should be only loosely or sparsely connected to the other modules. This translates in a probable reduction in the number of possible inputs to the module, and a consequent easier exploration of that module's input space.

The fact that it has been possible to find the single specification of each module due to the system's decomposability, hopefully allows, if the specification of each module is not too complicated, for a form of *aggregation*, as discussed in section 4.1.5.1. Let's clarify this scenario. After having tested a module, we have arrived at inferring its input/output specification. From that moment on, the module can be treated as a black box whose input-output relation respects this specification. If we are able to further abstract the module specification by "naming" it in a succinct way along the lines already highlighted in section 4.1.5.1, giving the module a name which is representative and explanatory of the function it performs (as for example when we say that a module performs the "multiplication" operation), then each module's specification can be substituted by this more concise definition of *what* function the module performs. This concise label which gets attached to a module constitutes in a way the "aggregate value" of the module: it summarizes under a single description all the operations internal to the modules. If it is possible to find these kinds of descriptions for all or most modules of the program, then a global specification of the whole system can be given *in terms* of a description (usually in the graphical form of a *flow chart*) of the modular structure of the system as a directed network of connected modules, where modules are seen as nodes labeled with their succinct "names" representing their specification, and their input and output connections are the directed links between nodes. As better explained in section 4.1.5.1, a module's *name* is a form of *aggregate* representation of the module's input-output function, which synthetically represents in some abstract way the function that the module fulfills, and so the complete description of the modular structure comes to represent in an aggregate way the overall global specification of the system, as a modular structural *composition* of the sub-specifications of each module. Of course, this reconstruction can be applied in a hierarchical fashion, with multiple hierarchical levels, each of which is an aggregate description of the underlying one, composed in a structured way by the set of subspecifications, which are the specifications of each module of the underlying level. The relation between any two levels is always that between a higher-level specification and a lower-level implementation of it: any specification at a certain level is an abstraction, in the form of an aggregate modular description, of the underlying description.

The interesting point of what described above is that the modular structure of the software has allowed for a succinct and computationally treatable reconstruction of the global specification of the program. This specification is given not in its explicit, extensional form, that is, the list of all its possible input/output couples, but in the form of a hierarchical functional *explanation*. We were dealing in this section with reverse-engineering of software, but this has given us the occasion of describing a notion which is exactly similar to the notion of *functional explanation* typical of certain scientific disciplines, a notion which will be better expounded in section 9.

It must be noted that the circumstances described above presuppose several conditions:

1. The program has a modular structure.
2. It must be possible to *find* its modular structure.
3. The single lowest-level modules must be simple enough to render the reverse-engineering of their single specifications a feasible task.

[53] See section 2.4.

Let's analyze the above conditions in detail.

- Condition 1 is often fulfilled, by humanly-written programs, at least if the program has been designed and developed in a systematic, well-planned way: first, as of today, except in special cases, almost every programmer writes programs in high-level languages. These are languages that impose a minimal modular structure, in the form of high-level constructs acting on program blocks[54] even if the program is not purposely written in a modular fashion. This feature of high-level languages always ensures a minimum level of modularity which is represented by the two-tier hierarchy of the classical distinction between low-level language (that is, machine language) and high-level language. A more rich hierarchical structure is aquired by the program if it has been programmed in a purposely modular way, especially making correct use of object-oriented languages. It must be noted, however, that not all object-oriented languages inherently enforce information hiding, and information hiding is the crucial feature which defines the modular structure of a program: information hiding ensures that modules are only connected by dedicated, sparse, ordered input-output channels, allowing to easily *isolate* each module, both structurally and functionally, from the rest of the system. And, as highlighted in section 2.1, isolation is one of the defining properties of modularity. Given that information hiding is not in many cases automatically enforced by the programming language, the risk is that in poorly maintained programs, especially after a long maintenance history, the determination to enforce information hiding is gradually given up by the programmers, ending up with programs constituted by heaps of cross-referenced, disorganized, entangled code. In other words, encapsulation and information hiding is violated by the diffuse access of variables originally internal to a module from outside the module. If this external violation of the module boundaries becomes too widespread, the modular structure of the program becomes difficult to detect.

- Condition 2 is that of the feasibility of the detection of the program's modularity: even if modularity is present overall, it could be difficult to detect it, and this for various reasons. The first, which has been hinted at above, is that modularity can be not so clear-cut in a given program: a modular structure must be *significantly* distinguishable from a randomly connected one. This is easily understandable by analogy with network modularity[55]: as we have seen in section 3.2.1.1, the amount of modularity of a network is measurable by a quantity, the so-called Q, which measures in what degree the network's modular structure differentiates from the non-modular structure of a randomly connected network. Another reason for which the modular structure of a program can end up being unidentifiable, is that the program is too large, and so it is literally impossible to examine it in its entirety in order to obtain a modular description of it, even by algorithmic methods. As noted by Tzerpos & Holt (1998), certain algorithms for modularity detection in reverse engineering run in $O(n^3)$, or $O(n^2)$, execution times which, although polynomial, could render the analysis of programs constituted by millions of lines barely tractable, and actual cases of poorly maintained, ill conceived, tangled programs of this size can exist in the industry in some cases. Of course, for such huge software systems, an idea of their modular structure can be obtained by other means than algorithmic modularity detection, but if their modular structure had degraded in time beyond a certain degree (which is not an

[54] See section 4.1.4.2.

[55] Actually, an abstraction of the modular structure of a program could be immediately represented by a directed network, where nodes are the program's modules and the edges are the input and output inter-module links connecting them.

unlikely condition for large software projects) their actual functional modularity could have faded in such a way as to render the approximate modular description which the system still allows an overly approximated explanation of the actual program.

- Condition 3 is in some sense the more crucial of the three: it is precisely the possibility that modules have a more limited span of input combinations than the full program, and thus that the reverse-engineering of the full input/output specification of each single module turns out to be an inherently feasible task. This is the circumstance that enables us, once a modular structure is found, to substitute the low-level original description of the program with an aggregated high-level functional explanation: if condition 3 fails, that means that it has been impossible to infer the input/output specification of each module, and thus that it has been impossible to obtain, as explained in section 4.1.5.1, an appropriate "label" representing in an aggregate way the function of the module, a label to be employed in the high-level aggregate modular description of the overall system.
  The fact that condition 3 holds is, in actuality, not guaranteed. It can happen, for example, that the program is composed of many modules, but that each of them takes the *full* input of the program in order to process it, each module in a peculiar way, in order to proceed to a final combination of the results of these separate computations: in this case, the space of the input configuration of a module has the same size of the space of possible input configurations of the overall program. If a module has a too wide input channel as in this case, it is advisable to proceed to a better analysis in order to see if the module itself is in turn decomposable into *smaller* modules each of which, it can be hoped, will have a smaller input channel. But, even in this second iteration, the result is not guaranteed. In general, it seems that at least the smallest modules, the most low-level ones, must possess a small set of possible input configurations in order to give us the chance to exploit the ease in inferring their specifications to the purpose of hierarchically explain the program in an aggregated way. If this condition of ease in finding the specification of the smaller modules holds, then it is feasible to produce the specifications of higher-level super-modules composed of these elementary modules. Specification of the super-modules would be produced not in the form of their extensional specifications expressed as lists of input/output couples, but as functional explanations, often expressed as flowcharts. The bottom-up reconstruction of a full hierarchical modular structure of the program can then progressively proceed, until in the end the whole structure of the program is given as a (possibly graphical) hierarchical representation of its functional modularity: this would not amount to explicitly describing the overall program specification under the form of its complete input/output mathematical function, but the modular hierarchical functional representation would probably, on the contrary, constitute a *better* representation of the overall specification of the program with respect to an extensional exhaustive listing of its input/output complete relation, because, contrary to the extensional input/output representation, this modular hierarchy gives important insights into *how* the global specification is brought about by the coordinated functional interactions of the subsystems of the whole program. The extensional specification would have only showed in a "brute" way *what* the program does, and not much more. An example of hierarchical representation of a software system is given in fig. 4.4.

4.3.1.1 Specification mining

I describe here another technique to dynamically infer the specification of a program by observation of a sample of the program's execution traces, that is, dynamical records of the program's

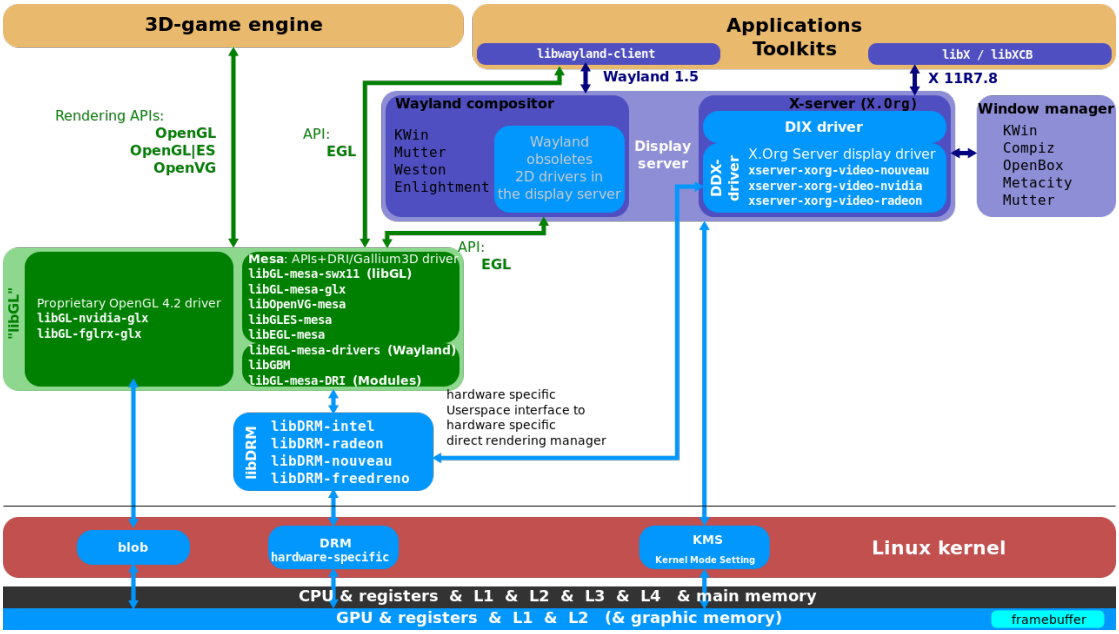

Figure 4.4: a quite high-level hierarchical representation of a software system: the Linux graphics stack. (Image attributed to Shmuel Csaba Otto Traian, taken from https://commons.wikimedia.org/wiki/File:Linux_Graphics_Stack_2013.svg)

executions. This method makes use of *data mining* techniques. Data mining is a subfield of *machine learning*, and consists in the practice of computationally searching for previously unknown hidden regularities in big amounts of data.

A seminal work, Ammons, Bodík, & Larus (2002) uses machine learning to infer (this is a form of *inductive* inference) a probabilistic finite automaton[56] (or *PFA*) which constitutes the specification. In a later simplification phase, the obtained PFA is scrutinized for edges of its graph with a below-threshold probability, and such edges are pruned, in order to obtain a deterministic FSA. This simplification phase is precisely a way of obtaining a modularization by ignoring weak links in a nearly-decomposable system. This way, a more economic and abstract specification is obtained.

Specification mining is affected by some serious problems.

First, the specification is produced by a FSA, so specifications realizable by Turing machine-level algorithms are excluded by definition.

A second point is that it is possible that some algorithms for mining the specifications end up being computationally expensive, or hard, because some of the tasks involved in the proposed algorithm are in principle intractable (exponential time on the number of variables considered in the execution traces), although it seems that the worst cases in which the algorithm takes too much time are rare, according to Ammons et al. (2002).

The automaton produced by machine learning is moreover not exact, for it is a *simplified*, *approximate*, *modular* form of the hypothetical exact one, so it can eventually, if applied to a non-linear

[56] See section17.2.3 in the appendix.

system, *diverge* from the program's behavior, in the long run. Nevertheless, it is the *simplified* automaton which is more useful in a possible computational explanation, due to its simpler structure and consequent high intelligibility. But this simplification of the automaton is possible only when the specification specifies a computational task which is in principle describable in a modular way, and it is not highly non linear.

All in all, the method proposed by Ammons et al. (2002) is interesting, given the computational cost of a full reverse engineering of a specification, but its many limitations make it seem susceptible of sporadic applicability.

4.3.2 Program modularity favors program development

It appears that program modularity, especially when it is actively planned during development, greatly favors the feasibility of reverse engineering of the program. Needless to say that program modularity enhances the ease of development of programs as well: at least mentally, if not in a written and possibly systematic way, the human programmer who is about to start writing a program, or who is already in the midst of writing it, continually proceeds, by alternating between bottom-up aggregation and top down decomposition, to produce schemata and diverse representations of the program's hierarchical structure, to mentally travel across the hierarchical levels of the program and between program modules she is creating in order to constantly clarify herself what she is about to produce, the relationship with the higher-level specifications and with the lower-level implementations of the parts of the program on which she is working, how to refine the modules, test their local functionality, and how to organize the structures at various levels in order to proceed in the best way to the program construction, including how and when to proceed to its restructuring, a practice which is sometimes needed.

In other words, the production of a functional hierarchical representation of the program is not needed only by the reverse engineer who is seeking the best way to explain the functioning of an initially unknown program, but it is also deeply needed by the *creator* of the program, who continually *explains* to herself the structure and function of what she is creating, while in the very act of creating it. This undeniable fact, that explanations are important not only *a posteriori*, but also during the development process, has an analogy in science, where explanation is needed not only to explain already happened phenomena, but also in research, during the making of a scientific model. This circumstance will be better highlighted in section 6.9.

4.3.3 Inherently antimodular programs

If the program is not modularizable because one or more of the conditions 1-3 reported in section 4.3.1 fails to hold, then we must face, in reverse engineering, the bleak outlook of having to try to inductively reconstruct the global specification of the program in a direct bottom-up way, a task which, as we have seen, is in most cases unfeasible. It is true that there are techniques of *machine learning* which allow for the *probable* inference of an *approximate* version of the program specification on the basis of a number of observations which is only a polynomial function (not an exponential one) of the possible input size, like the Probably Approximately Correct (*PAC*) learning framework devised by Leslie Valiant. But this method is generally capable of inferring only quite simple specifications, and cannot work for reverse engineering of programs implementing complex algorithms[57].

[57] For PAC learning, see Valiant (2013). I am not going to treat the complex and multifaceted subject of machine learning and the limitations of its methods in this work.

So, inability of describing the modular structure of a program severely hinders the discovery of its specification. As explained above, reasons for this inability are multiple. As we have seen, programs written in a high-level language are inherently endowed with a limited form of modularity, the modularity imposed by the structure of the high-level language. But, can a program be completely devoid of significant modularity at a higher-level than that? Can a program be inherently *antimodular*?[58] Of course, it is perfectly possible to purposely produce contrived examples of programs which, being completely antimodular, implement nevertheless sensible specifications. There is even a form of diversion in the hacker community, the so-called art of *obfuscated programming* which consists precisely in deliberately devising sensible programs which are very hard to reverse engineer, or, in other words, which are very difficult to understand, in absence of explanations on the part of the creator of the program[59]: in most cases, obfuscation techniques produce inherently antimodular programs, where many parts of the program are reciprocally cross-accessed by other parts. In these cases, condition 1 in the list of section 4.3.1 gets negated.

But, apart from contrived examples, can a *sensible* program be completely devoid of modularity? There are certainly programs which, though written in an apparently linear and ordered way, are not easily decomposable into modules. The main reason can be that a program of this kind is too *short*, when expressed in a preferred programming language, to allow for a further partitioning of its few instructions into subgroups with a functional autonomous significance, that is, into *modules*. In these cases, though, one could suspect that the conciseness of the program could allow for an easy reconstruction of its specification by observation of its code: after all, a short program should also result simple to understand in its behavior. Sadly, this is not always the case: there are programs which are short, not easily modularizable, and with a very complicated behavior. In these cases, inference of their specification is quite hard to be obtained without exhaustively producing all of their input/output couples. But this is unfeasible, as we have seen. For an example of such a kind of program, consider the following listing (written in a "pseudocode"):

```
while length of string > 1:
    begin loop
        print string
        if string[1] is "a" then string = string[2:] + "bc"
        if string[1] is "b" then string = string[2:] + "a"
        if string[1] is "c" then string = string[2:] + "aaa"
    end loop
```

The above code[60] takes as input a string s composed of "a", "b", or "c" characters, and outputs sequences of strings such as the one showed in fig. 4.5, which have an apparently unpredictable behavior:

[58] I introduce here informally the term "antimodular", but I will more explicitly propose and advocate the introduction of *antimodularity* in section 13, as a term standing for a complex notion consisting in general in failure of the detection of modularity in complex systems. This is one of the main notions which I want to put forth in this work.

[59] Other than recreational use, obfuscated programming can also have commercial application, aimed to hinder possible attemps to reverse-engineer a commercial program. There are also algorithms for producing forms of obfuscated code.

[60] In this notation, `string[n]` stands for the n-th character in the string, starting from the left, and `string[n:]` stands for the string, minus its n leftmost characters. The statements `begin loop` and `end loop` are the delimiters of the interposed block of code, which gets repeatedly executed while the condition above it holds, that is, while the length of the string is greater than one character. The `+` sign concatenates two strings.

```
aaaaaaaabc
aaaaaabcbc
aaaabcbcbc
aabcbcbcbc
bcbcbcbca
bcbcbcaa
bcbcaaa
bcaaaa
aaaaa
aaabc
abcbc
cbcaaa
caaaaaa
aaaaaaaa
aaaaaabc
aaaabcbc
aabcbcbc
bcbcbca
bcbcaa
bcaaa
aaaa
aabc
bca
aa
```

Figure 4.5: an example of the output of a string-manipulation program corresponding to the Collatz conjecture.

Almost certainly, the above program cannot be rephrased in a meaningful "aggregate" way, because it is evidently too short to allow for a sensible functional decomposition which differs substantially from the natural one, that is, the modular description already provided by its being expressed in a high-level language, a language which is already a form of "aggregate" description with respect to machine language[61]. We will have then, in order to describe what the program does, to try to infer its specification, that is, precisely to try to understand what the program does for any input it receives. But the brevity of the program in this case does not help: the length of its input is in principle unbounded, because it is constituted of strings of any possible length. So, even an exhaustive enumeration of all the possible input/output couples is precluded, and we will have to "guess" its specification in other, finite ways. Sadly, though, this particular program has been proved equivalent to a mathematical procedure whose general outcome is the object of the *Collatz conjecture*, an unproved conjecture in number theory, which has resisted proof till now, and could even end up being undecidable (not provable nor disprovable inside standard arithmetic). Finding the complete specification of this program would be equivalent to *solving* the conjecture, so it is not likely that we could succeed. Even understanding in some sufficiently complete way the specification, that is, what the program does, would possibly amount to gaining some insight toward a proof of the conjecture to which it is connected, and, given that this conjecture has resisted till now any attempt to prove it, the eventuality of understanding this program seems unlikely.

It seems, then, that reverse-engineering can be severely hindered by the absence of modularity in programs.

[61] See section 4.1.4.2 for a clarification.

4.3.4 Modularization of computer programs by program slicing

As we have seen, modularity is typically imposed on the program structure by the programmer during development, by following certain guidelines of supposedly good program design. I will use here the term *modularization* to refer to the reverse-engineering operation of subdividing the code of an already written program into a set of more or less independent sequences of code statements. Clearly, a program whose code has been structured in modules during programming is afterwards naturally modularizable by subdividing it in the set of its modules. But by making use of some techniques, often even of a program which has not been already explicitly structured into a modular form by its programmers can be attempted a modularization into more or less independent functional modules.

There is an enormous literature on detection of modularity in computer programs, and here I will only explain, as an example, a single technique: the method of *program slicing*.

In a couple of seminal works, Weiser (1981) and Weiser (1982)[62], the author observed programmers while debugging unknown programs, and noticed that they routinely subdivide programs under examination into pieces which do not represent *contiguous* pieces of code (as usually occurs while programming, when applying modularity often means structuring contiguous code in modules), but which represent the *data flow*: these non contiguous pieces of code are the parts of the code that influence the same particular set of variables. Weiser calls these parts of program *program slices*. In his 1981 paper, Weiser defines *program slicing* as "a decomposition [of the program] based on data flow and control flow analysis"[63], as opposed to traditional program decomposition into procedures and abstract types. This method allows, by examination of the source code, for the detection of parts of code, the program slices, each of which amounts to a module: each slice is identified by a position in the program code listing, and by a certain set of variables. A given slice contains code which makes use of and influences *only* the specified variables. And, no other parts of code outside a given slice modifies or makes use of the variables pertaining to that slice. This condition coincides with what is required for a module according to certain conceptions of program modularity, such as object-oriented programming, as highlighted in section 4.2.3.

In program slicing, basically the code listing of the whole program is "sliced" into a set of non-overlapping sequences of code instructions, which operate on different sets of correspondingly non-overlapping variables. This, according to a general definition stemming from the original Herbert Simon's intuition about decomposability[64], constitutes precisely a form of modularity: all the statements in the code of a given slide are more closely related to one another (by virtue of focusing on the same variables) than how any of them is related with any other statement in the code outside the slice. This condition is perfectly compatible with the general definition of modularity given in section 2.1.

[62] The second paper actually predates the first.
[63] Weiser (1981), p.352.
[64] See section 2.2.3.

Chapter 5

Modularity in discrete dynamical systems

Simply stated, a dynamical system is a system whose state changes in time according to a certain update rule.

We have already described a specific form of modularity of dynamical systems, *aggregability*, in section 2.2.1. Aggregability holds for continuous and discrete dynamical systems, and it is a form of modularity of the update rule, as will be also clarified in section 6.5. I refer to those sections for a deeper analysis of aggregability in dynamical systems. In this section I am concerned with different forms of modularity which can affect dynamical systems, and specifically discrete dynamical systems: as already stated, this is the class of dynamical systems, taken as paradigmatic, which I will mainly deal with in this work.

Discrete dynamical systems can also show forms of dynamical modularity different from aggregation, even in the absence of aggregability. I would like to take here in consideration a form of *process modularity*, that is, modularity manifesting as the presence of portions of the space-time evolution of the dynamical system which show some kind of persistent identity and independence from other portions of the same configuration, and which can therefore be considered modules. As we will see, this form of modularity can be used to implement a high-level computational capacity in certain dynamical systems.

In what follows, a discussion on modularity in discrete dynamical systems will be preceded by some introductory sections on generic discrete dynamical systems and on a subclass of them, *cellular automata*.

5.1 Discrete Dynamical Systems

I consider here *discrete* dynamical systems, or *DDS*s, that is, dynamical systems in which both time and the possible states of the system are to be considered discrete. The defining characteristic of a DDS is simply that it possesses some kind of global state (which can be atomic or a configuration of atomic states), a state that can change in time at each time step (or *timestep*) according to a deterministic *update rule*, that states which state must follow the current state at the next timestep.

DDSs can assume many forms: the most obvious types of DDS are the classical computational architectures, like the Turing machine or the von Neumann architecture, on which common

computers are based. These computing architectures are mainly *serial* in their functioning, but there is an unbounded variety of possible DDSs, many of them with a much more parallel or distributed structure: the defining characteristic of a DDS is simply that it possesses some kind of global state, that can change in time at each time step.

5.1.1 Modular/digital and DDSs

Discrete dynamical systems can show different forms of modularity, but the basic one stems from their being discrete, or *digital* systems: it is, I think, arguable that being digital is a form of modularity.

The notion of *digital*, despite being apparently clear, can be deemed to be in need of a definition anyway. We could start with a definition by John Haugeland:

> A *digital system* is a set of positive and reliable techniques (methods, devices) for producing and reidentifying tokens, or configurations of tokens, from some prespecified set of types.[1]

It stems from this definition that digital systems deal with distinct, discrete elements (the tokens) which are endowed with the ability to stably maintain their identity in time, in order to allow their possible re-identification by means of a "positive" and "reliable" set of techniques. These last qualifications, as Haugeland highlights, mean that the method for production and reading of these tokens is a method which "can succeed absolutely, totally, and without qualification"[2]. This makes such kind of tokens immediately qualify as modules, at least according to the definition of module proposed in section 2.1, because tokens of this kind turn out to be well delimited, independent, and robust entities, where robustness is what enables them to maintain their identity without risk of weakening in time, and this is the feature that allows for their positive and reliable re-identification: re-identification implies that some amount of time has passed since a first identification of the token, and the fact that reidentification can be positive and reliable entails that the token has not changed overly during this time lapse. Thus, according to the above definition, digital systems are *modular* systems.

In addition, it appears from the above definition that a digital system is not only a set of tokens, but some sort of *machine* which produces, reidentifies, and that presumably, albeit not explicitly stated, destroys configurations of tokens. We could add another condition, not explicitly stated in Haugeland's definition, which aims to be more general, that is, the condition that the machine manipulating tokens acts not in a continuous manner, but only in discrete successive steps: this way, the "machine" constituting a digital system, albeit physically realizable, corresponds obviously to the implementation of a computational abstract machine, a concept which is indeed at the basis of the notion of digital computation[3]. These machines are the object of computer science, and are kinds of discrete dynamical systems.

In other words, explicitly extended this way with the condition of discrete time, the above definition of digital system becomes the definition of a DDS: if we see a configuration of tokens as the current global state of a system, and destroy that configuration to substitute it with another, different one, we can say that the system has changed of state. If this happens only at discrete timesteps, and according to some rule, then we have realized a DDS. And, given that,

[1] Haugeland (1989), p. 53.
[2] *ibid.*.
[3] A basic exposition of the theory of digital computation is the Appendix, section 17.

as argued above, digital systems are modular systems, a DDS is a modular discrete dynamical system.

This kind of "digital" modularity is common to all DDSs, and it is not particularly interesting. It is the form of modularity pertaining to the DDS's *preferred description*, which is the most natural description of a system, a concept which I touched upon in section 1.1.1 of the Introduction, and will be better clarified in section 6.6.

5.1.2 A general definition of DDS

We could reformulate as below all the above considerations in a series of basic definitions of notions pertaining to discrete dynamical systems[4].

- An *alphabet* is a finite set of *symbols*. It is beyond the scope of this work to analyze in detail the notion of *symbol*. Suffice to say here that a symbol is a *type*, constituted by token symbols which can be considered belonging to the same type according to some easily applicable criterion. Each token symbol is a particular finite configuration of some matter which happens to be easily recognized and produced, and which tends to remain unchanged in time, if not influenced by external forces.
- A *state* is the particular condition, constituted by a certain configuration of token symbols, in which a DDS is in, at any given moment, and that can change at the next timestep.
- An *update rule* (or simply a *rule*, or *dynamics*) is a function from states to states: it determines, based on the system's current state, which state it must change into, at the next timestep.
- A *discrete dynamical system* is a system which, at any given time, is in a certain state. Its state changes at each timestep according to an update rule.

In more formal terms:

a DDS is a system which is, at any given time t, in a certain state or configuration $c_{(t)}$, that is a certain assignment of values (the instantiation of token symbols) to the elements of a set v of variables:

$$v = \{x_1, ...x_n\}$$

Given a configuration $c_{(t)}$ in which the system happens to be at time t, that is, the values at time t of the elements of the set v, the *next configuration* $c_{(t+1)}$ in which the system will be at the next timestep (that is, the configuration of the system at time $t+1$) is a certain function D (the update rule, or *dynamics*) of the configuration at time t:

$$c_{(t+1)} = D(c_{(t)})$$

Since c is a set of values of the variables $x_1, ...x_n$, we could write

[4] Equivalent definitions are reported also in section 17.1 of the Appendix, which treats computational machines.

$$c_{(t)} = \{x_{1(t)}, ...x_{n(t)}\}$$

where $x_{i(k)}$ is the value of variable x_i at time k . Substituting the values of the variables at time $t + 1$ to $c_{(t+1)}$, we get

$$\{x_{1(t+1)}, ...x_{n(t+1)}\} = D(x_{1(t)}, ...x_{n(t)})$$

5.2 Cellular automata

Cellular automata (*CA*s henceforth) are a class of discrete dynamical systems which share certain common characteristics. They are widely studied for their structural simplicity associated with the surprising variety and complexity of their dynamical behavior. There is an enormous literature on cellular automata, which highlights their properties under different standpoints. The two main approaches consist in considering CAs as digital systems, making use of computer science results to study their computational features, or in considering them as dynamical systems, resorting to the mathematical tools provided by chaos theory, a theory which studies the dynamical behavior of discrete and continuous non-linear systems.

It is impossible here to expound the subject of cellular automata in all its facets, so I will limit the following exposition to a handful of basic properties of CAs which will be of use in the rest of this work[5].

A *cellular automaton* is a DDS constituted by an array, finite or infinite, of elements, its *cells*. The array can be one-dimensional, representable as a row of adjacent cells, or can have any higher-order dimensionality, and in this case it can be represented as a grid of cells. Most studied CAs are 1-D or 2-D.

Each cell of the array can be, at any given time, in a possible *state* (usually called a *color*), constituted by a symbol taken from a finite alphabet. The overall state of a set of cells at a given timestep is constituted by the specific configuration of all the single states of each cell in the set, and as such is called a *configuration*.

The update rule of a CA is constituted by the repeated application, for each cell, of a *local* rule (the *CA rule*), which determines the state that the cell will assume at the next timestep (the cell's *next state*) on the basis of the state of all the cells included in a finite *neighborhood* of the cell under consideration.

A *neighborhood* of a given cell is a finite set of cells which are placed at fixed distances from the reference cell, usually including this cell as well. A neighborhood is characterized by its

[5] Cellular automata are, from a theoretical point of view, potentially unbounded in the size of their internal configurations, but their physical implementations (or simulations on physical conventional computers) are of course finite in size. While theoretical finite models of cellular automata have slightly different properties than those of potentially infinite ones, I will not mention these differences in what follows. The situation is similar to that of most treatises on abstract computational machines, like the Turing machine: while Turing machines have different properties than the linear bounded automata, which are their finite counterparts, these differences are usually neglected in introductory texts. In general, where for potentially infinite machines there is uncomputability, for their finite counterparts there is only computational intractability (this does not always hold, anyway: there are properties of finite machines which are uncomputable as well).

radius, that is, the maximum distance from the reference cell at which a cell belonging to the neighborhood can be placed. Fig. 5.1 depicts a neighborhood of radius 1 in a 1-D cellular automaton. A typical neighborhood for 2-dimensional CAs is the so-called *Moore neighborhood*, represented in fig. 5.2, which has radius 1 and comprises all the adjacent cells of the reference cell.

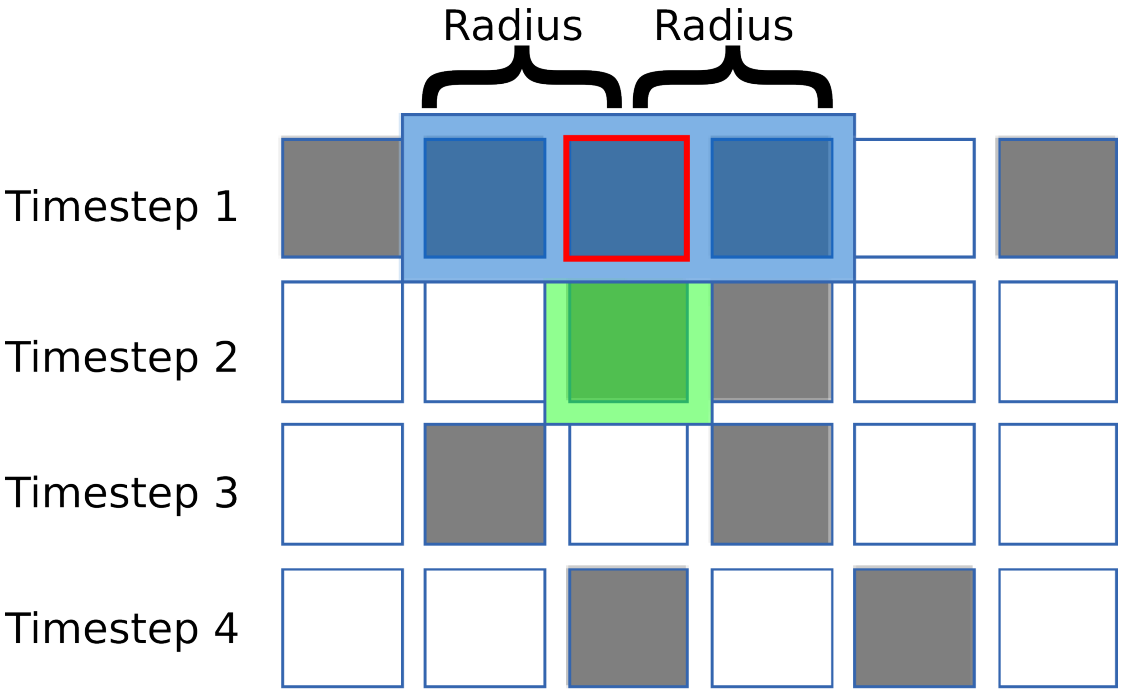

Figure 5.1: part of a temporal evolution of a 1-D cellular automaton. Time flows from top to bottom. Each row of cells represents the state of the automaton at a given timestep, with cells in different states represented by their being white or colored. The blue rectangle highlights the neighborhood, which in this case has radius 1, of the cell surrounded by the red frame. The green square highlights the same central cell at the following timestep: the value of the cell at this step is determined only by the previous values of the cells of its neighborhood, that is by the values of the cells in the blue rectangle.

The CA rule is a function which maps the configuration of all the cells in the neighborhood of a given cell to the next state of that cell. Given that a neighborhood is an area of finite radius which surrounds the cell under consideration, the CA rule is a *local* rule, which considers only the state of a limited set of cells of the CA around the reference cell, ignoring distant cells, and it acts on the value of the reference cell only. This contrasts with the generic conception of update rule for DDSs stated above[6], where the update rule maps *global* states of the DDS to other global states. Of course, being a DDS, a CA has itself a global update rule, which is constituted by the synchronous application of the CA rule to all the cells of the Ca's array. It is by this synchronous composition of local rules that all the cells in the CA simultaneously change state at each timestep, and thus that the global state of the CA changes.

A CA is completely determined by its CA-rule. The rule is a function from configurations of the neighborood to cell values. The complexity of the rule is thus dependent on the extension of

[6] Section 5.1.2.

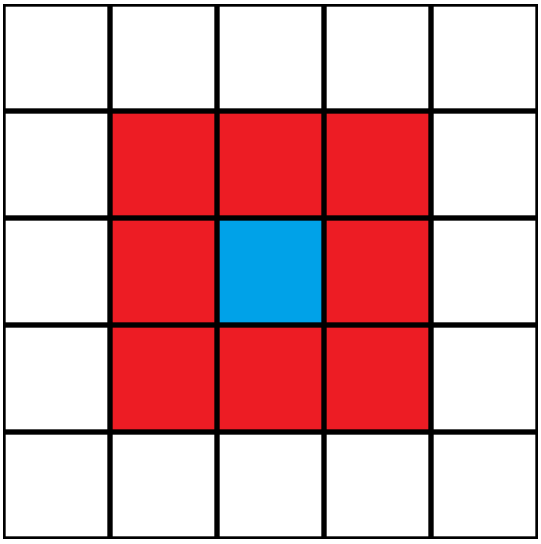

Figure 5.2: the Moore neighborhood for 2-D cellular automata. The reference cell is at the center, colored in blue.

the neighborhood, other than on the number of colors: the rule must map each of the possible configurations of the neighborhood to different values of the reference cell. The number of these configuration-value couples which are necessary for the specification of the rule grows exponentially with the length of the radius, so the update rule can result highly complex even for simple CAs[7], and CAs with radius larger than a certain size can be computationally intractable because of the space necessary to hold the rule's data.

In the case of two-color CAs, a rule could in principle be represented as a logical operation which combines variables, representing the cells of the neighborhood, in order to yield the next value of the reference cell. This operation, represented as a logical expression, would certainly be a more synthetic representation of the rule than its extensional representation in the form of an explicit list of all its input-output couples. The problem here lies in the fact that, if the logical expression representing a rule is not already known, it is not easy to obtain such an expression starting form the extensional representation of the rule: this extensional representation can be seen as a *truth table*, and the best known algorithms for converting a truth-table into an analytic expression, such as the *Quine–McCluskey* method, work in exponential time depending on the number of bits in their input (in this case on the number of cells in the neighborhood), so this conversion is probably computationally unfeasible in CAs with large enough neighborhoods.

The space of all the possible CA rules of a certain radius is huge: there are $2^{(2^{(2r+1)})}$ rules for two-color CAs of radius r. Because of this vastness, and of the possible size of rules with even a not very big radius, the study of CAs has been mainly conducted on automata of very small radius: basically, only 1-D automata of maximum radius 2 and 2-D CAs of radius 1 have been considered for detailed study.

The most famous 2-D CA, and also one of the first to be studied is the so-called *Game of Life* (or *GOL*) CA, invented by John Horton Conway in 1970, first presented in a Scientific Ameri-

[7] For example, for a two-color 1-d CA with symmetric neighborhood of radius r, the number of couples needed in order to specifiy the rule is $2^{(2r+1)}$.

can article (Gardner 1970). GOL has revealed to be capable of an enormously varied possible dynamical behavior, a behavior which changes wildly depending on the initial configuration.

Among 1-D CAs, those of radius 1 and two colors represent a sufficiently restricted class to be the object of a detailed study: there are only 256 rules with these features, and each rule has only 8 couples of values. This class has been called by Stephen Wolfram the class of *elementary* cellular automata, or *ECA*s. Each ECA is represented by a number from 0 to 255 which corresponds to a particular base-2 numerical representation of its rule. ECAs are usually individuated by means of this numerical code.

5.2.1 Stephen Wolfram's classification of CAs

A CA is a DDS, and as such, during its functioning, it undergoes an *evolution* in time. An evolution always starts with the automaton in a certain global state, which is its *initial configuration*. The specific evolution the CA will follow depends exclusively on the initial configuration. In many CAs, even slightly different configurations can give rise to completely different evolutions. This sensitivity to initial conditions is a manifestation of the non-linearity of most of these dynamical systems. As we will see, there are, however, some CAs which do not show this kind of sensitivity, and which converge toward a typical constant or simply periodic behavior after a certain number of steps, regardless of the specific initial configuration. Certain other CAs show instead the same kind of seemingly chaotic behavior independently of the choice of the initial configuration chosen. In other cases, starting from certain configurations, after a period of seemingly chaotic behavior, the CA settles on a complex, partially ordered behavior in which certain significant, ordered dynamical structures "emerge" from the preceding chaos. This is a form of "self organization" which manifests in some CAs. Certain CAs are very flexible and can show any of the above sketched dynamical regimes, depending on the choice of the initial configuration. In what follows we will encounter a specific classification of these kinds of of behavior.

Studying ECAs, Stephen Wolfram has famously proposed (in Wolfram 1984) a scheme aimed at qualitatively classifying the possible dynamical behavior of cellular automata. This well-known scheme is composed of four classes which, according to Wolfram, can summarize all possible behavioral types of CAs in general (not only ECAs). Wolfram's classification is largely based on intuitions initially deriving from a huge number of "experimental" observations Wolfram had conducted on simulation of CAs in the course of years. The proposed classes do not have formal definitions, but are differentiated according to humanly-observed global properties of the CAs' behavior. Besides, the generalization of this classification to any possible CA is purely inductive. Nevertheless, this criterion has encountered a huge success in the research field of DDSs, and numerous formal specifications have been attempted. Many authors believe that the qualitative classes of behavior described by Wolfram catch every significant qualitative distinction in the behavior of any complex system. The classification reflects more or less the types of evolution which I have hinted at above, and can be expressed as follows, closely following Wolfram (2002)[8]:

- class I: CAs with simple behavior, in which almost all initial conditions lead to very similar uniform and constant states[9];

[8] pp. 231-235.

[9] What are viewed here as constant states, are actually cycles of period 1: the CA continues its evolution, but the reached state does continually repeat, so it appears as a stable state which does not change anymore. It must be noted that, per se, a CA does not have a final state, like automata studied by classic computational theory have (as explained in section 17.2): a possible criterion to judge a CA has come to a final state is that of cyclic behavior of period 1 or small period.

- class II: CAs with many different possible final states which in general consist of a certain limited set of simple structures which are constant or cyclic with short period;
- class III: more complicated behavior, seemingly random and chaotic in most respects, although some ephemeral, small-scale structures are always visible at some scale;
- class IV: a mixture of order and randomness: localized moving dynamical structures appear, which albeit being simple, interact with each other in very complicated ways.

Fig. 5.3 reports some sample behavior of CAs belonging to the above four classes.

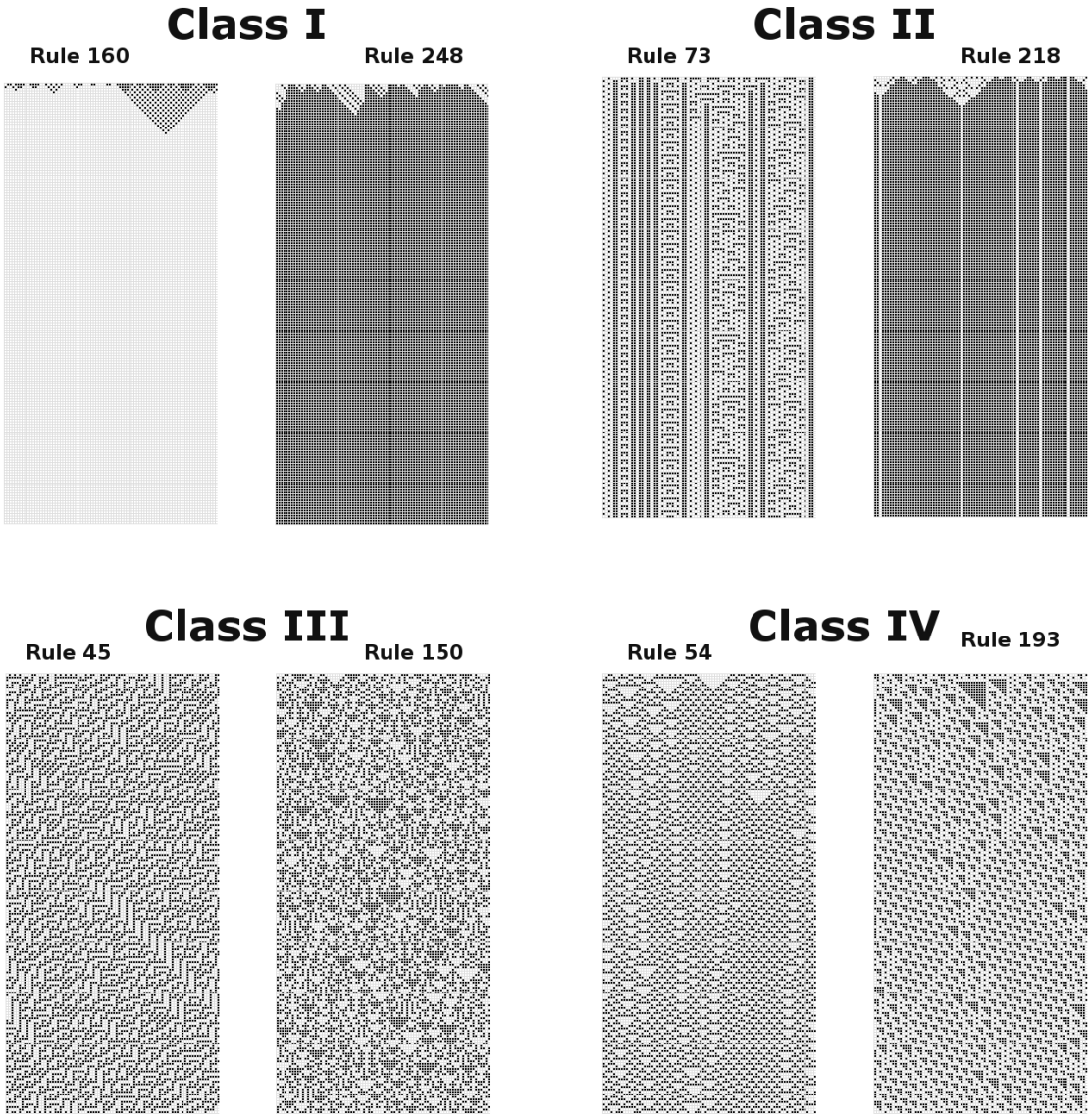

Figure 5.3: examples of dynamical behavior classified according to Wolfram's qualitative criterion. For each class two representative ECAs of that class are shown. CAs are 1-D. Time flows from top to bottom. (images generated with the web app at http://systems-sciences.uni-graz.at/etextbook/sw2/ca_1d.html).

Wolfram's classes intuitively follow a growing complexity: class IV is the class of systems manifesting the most complex behavior, while class I systems are mostly trivial. Class II systems can apparently show more varied evolutions, but these are still quite predictable. Class III systems are completely chaotic, and, while seeming the most complex, it is not possible to discern in the pattern they produce any sign of sensible order. This type of complexity resembles that of pure randomness, and as such can be suspected to be a form not of complexity but mere confusion.

It is class IV that which comprises the most interesting CAs. These systems manifest a behavior which is seemingly capable of maintaining a form of order while undergoing a very complex dynamics. Systems belonging to this class have been suspected to be able to perform complex computations, and indeed for some of them it has been *proved* that they are Turing-complete, that is, that they are capable to perform computations on the level of universal Turing machines[10], which constitute the most powerful class of known computational systems.

Among the well studied CAs, which certainly belong to class IV, there is Conway's GOL, which has been proved Turing-complete in Rendell (2002). Among class IV 1-D CAs, rule 110 has been demonstrated capable of universal computation by Matthew cook, in Cook (2004). Under which circumstances CAs can be considered computationally capable is not a simple question, and I dedicate section 14.5.1 to a reflection on this theme. However, in the next sections of this chapter some preliminary hint on this problem will be provided.

5.2.2 Process modularity in CAs

CAs belonging to class I, II and IV possess a form of *process modularity*[11], which consists in the following condition: certain localized and partially isolated subconfigurations of cells of the CA lattice can appear and persist for a certain amount of time steps without disintegrating, maintaining during this time a certain distinguishing identity and cohesion. In some cases, some of these configurations can be so robust as to persist indefinitely in absence of external perturbations, that is, if isolated from other dynamically changing structures. Some of these robust structures, especially in class IV CAs, can even in certain circumstances endure perturbations coming from the external context, reconstituting their identity after certain limited interactions. In other words, these structures appear to possess some form of robustness, both spatial and temporal. This robustness, together with their recognizable identity and their being localized, immediately qualifies these subconfigurations of the CA lattice as *modules*, according to the defining properties of modules brought forth in section 2.1.

Simple modules, present in most class I , class II and class IV CAs, are "frozen" structures[12] and oscillators. An example is in fig. 5.4. A frozen structure, called also a *still life*, is a subconfiguration of the CA lattice which does not change in time. Degree of robustness can vary: certain frozen structures are very robust, and can stand perturbations coming from the external context without losing their integrity. This way, they act as "impenetrable walls" which spatially limit the spread of certain perturbations inside the CA lattice. Oscillators do change, but staying in place and following an oscillatory behavior of limited period.

By looking at fig. 5.4, it appears clearly that certain vertical structures, like in this case the frozen structure and the oscillator, are endowed with a substantial spatio-temporal uniformity, which can be seen, at least intuitively, as signaling their enduring identity: precisely because of this robust identity, these structures can be considered modules. Other structures represented in

[10] For Turing machines, see section 17.2.5 of the Appendix.
[11] This notion will be better discussed in section 6.2.
[12] See the discussion on frozen structures based on Stuart Kauffman's ideas in section 7.1.2.

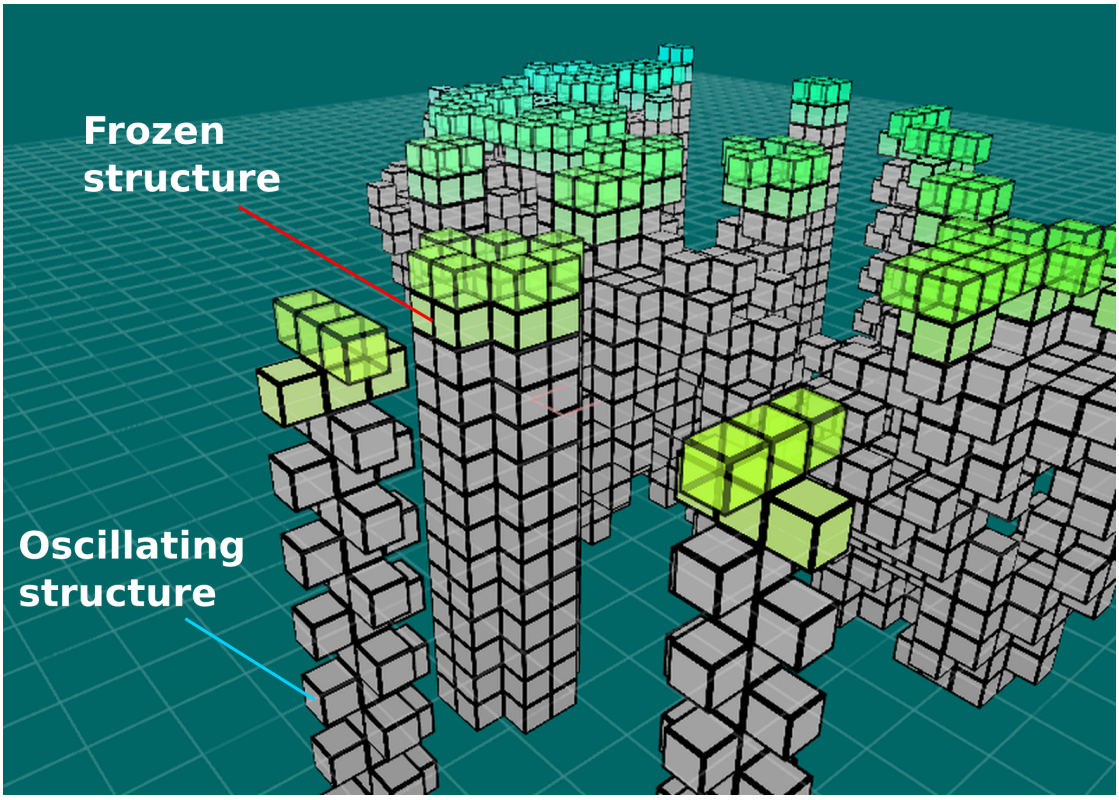

Figure 5.4: a 3-D view of a segment of evolution of the 2-D *Conway's Game Of Life* CA. Time flows from bottom to top in the figure: each horizontal plane containing one of the stacked layers of cells represents the 2-D global configuration of the CA at a given timestep. (Image modified from the original at http://xerol.org/h/img/1246, attributed to Xerol).

the figure change more drastically in time, and do not, at least at first sight, appear as modules. Of course recognition of a module's identity is a *relative* question, as will be highlighted shortly in what follows.

In other cases, persisting recognizable subconfigurations of the CA evolution, that is, process modules, change dynamically during their life, and their enduring identity can be recognized *despite the changes they undergo*: in other words, their identity is *multiply realizable* by a number of specific configurations. Typically, these changing modular configurations virtually "move" across the lattice as they change. Classic cases of this kind of modules are the traveling modular structures which can appear in the evolution of class IV systems, such as the "gliders" (examples of which are represented in fig. 1.2 of the Introduction, figg. 5.5 and 5.6) or more complex "spaceships", such as the spaceship of Conway's GOL depicted in fig. 5.7.

As an aside, it must be noted that, in each CA, there is a *speed of light*, which is the maximum speed that a "traveling" structure can reach. The speed of light is related to the width of the neighborhood taken in consideration by the rule, and is equal to the maximum radius of this neighborhood (for rules with symmetric neighborhood there is only one radius value). In Elementary CAs, which have rules of radius 1, the speed of light is equal to one cell per timestep. No structure can travel faster than this speed, because this is the maximum distance any possible influence between cells can reach in a timestep. It is possible, however, that moving structures

move at a speed well below the speed of light. For example, the spaceship in GOL hinted at above travels at 2 cells for 5 timesteps, while the gliders travel at 1 cell every 5 timesteps.

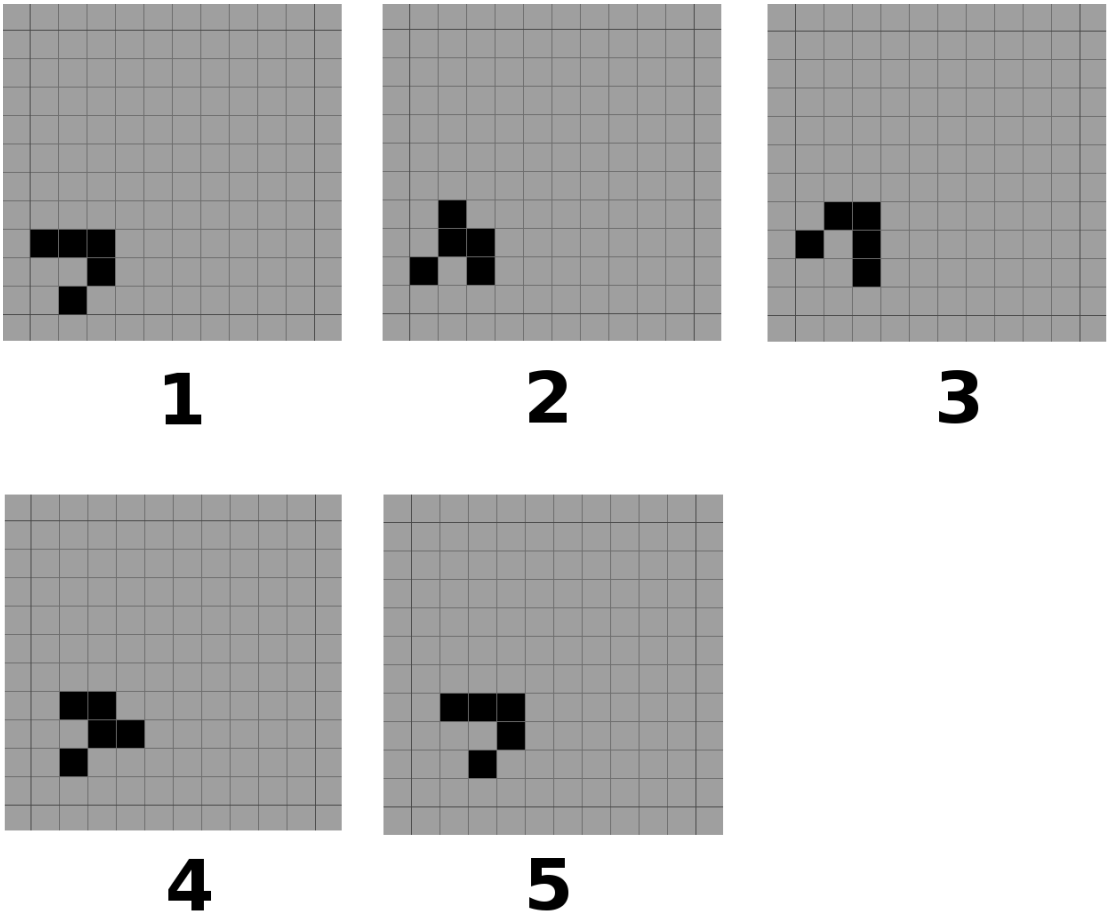

Figure 5.5: a part of a temporal evolution of the *Conway's Game Of Life* 2-D CA depicting the basic cycle of a *glider*, a modular dynamical structure which, if undisturbed, travels across the CA's array. Time flows from image 1 to image 5. The glider cycles through four possible configurations (1 to 4) before returning to the initial configuration (step 5). During this cycle the glider travels diagonally 1 cell.

The form of process modularity described above can also be seen, from a certain standpoint, as a peculiar form of *aggregability*: we must note that, in a truly general conception of modularity, a process of aggregation can consist of any computable function which maps a configuration of the module's subparts to the module's aggregate state, not only of classical arithmetical functions like those mentioned in Simon's classic examples. The act of recognizing a changing subconfiguration of a CA lattice as a persisting module is a form of aggregation: the aggregate state is simply a placeholder for the presence of the module, when this is actually present: the aggregation function is a *module detector* which, when a process module is present in the CA's evolution, outputs a certain value, let's say the position of the central point of the configuration of cells constituting the module. This position constitutes the aggregate value of the process module, a value which represents the module in its totality.

Let's take a look at the spaceship depicted in fig. 5.7: its robustness consists in the possibility

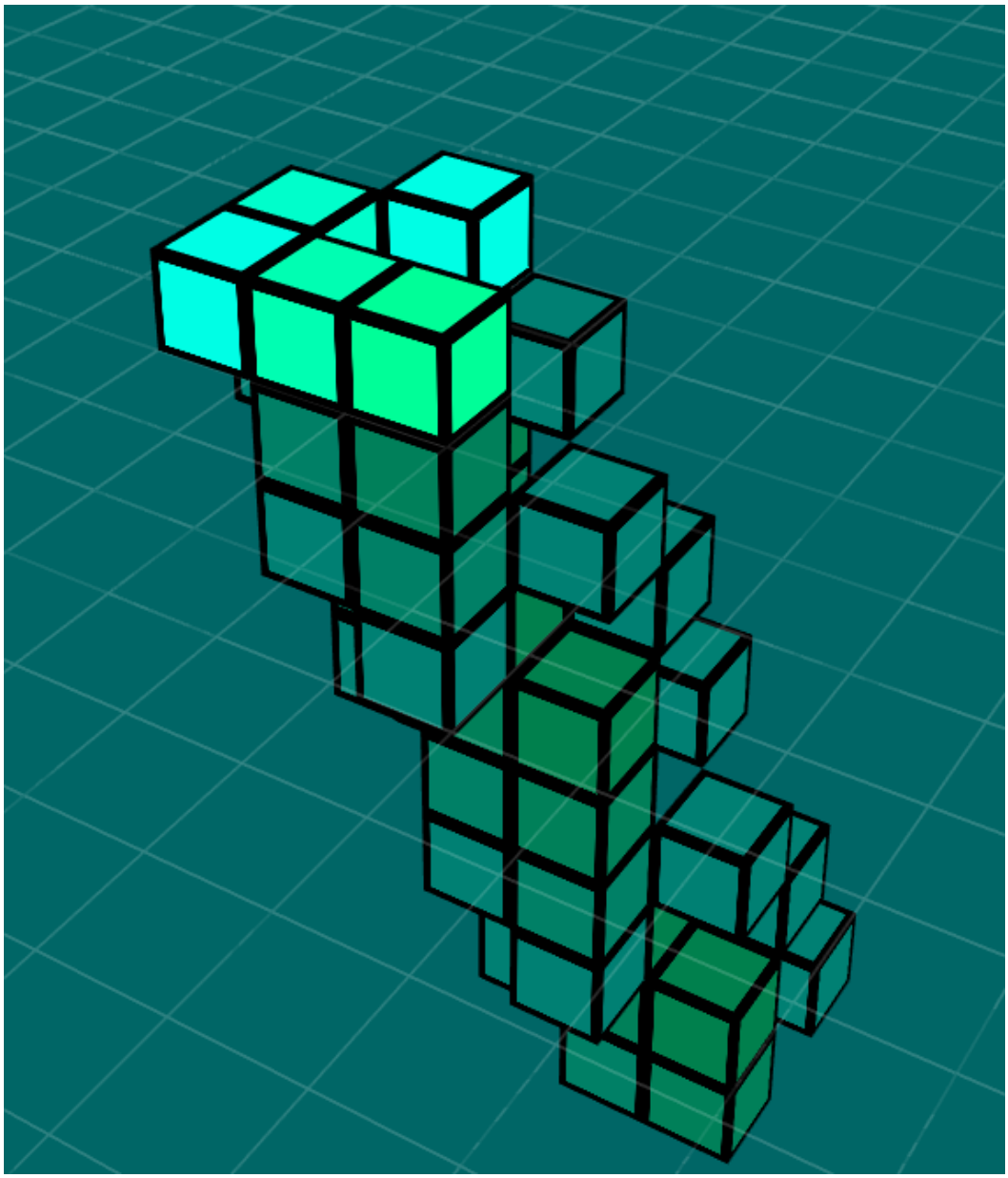

Figure 5.6: a 3-D view of a glider of *Conway's Game Of Life*, a 2-D cellular automaton. Time flows from bottom to top in the figure: each horizontal plane containing one of the stacked layers of cells represents the global 2-D configuration of the CA at a given timestep. The glider's displacement during time is clearly indicated by the inclination of the trail, corresponding to a diagonal line passing through the implicit centroid of the glider (Image taken from Wikimedia Commons: https://commons.wikimedia.org/wiki/File:Glider_trail.png, attributed to Xerol at en.wikipedia).

of remaining recognizable as the same spaceship despite the continuous changes of its microcomponents (the CA's cells which compose it), and in keeping its cohesion in spite of these high-frequency internal interactions. In this case the aggregation function is any kind of "spaceship detector", a function which maps all the possible five internal configurations which correspond to the given spaceship to the position of its central point in the matrix of another DDS: see fig. 5.8 for a graphic explanation. The new DDS so obtained is a *coarser-grained* or, in other words, *aggregated* version of the first (the CA), and it is the analogous, in this (discrete) case of the aggregate systems treated by Simon. If the hierarchy individuated in the CA corresponds to an actual dynamical hierarchical subdivision of the system, then the DDS at the level of "spaceships" can be studied independently from the other levels[13].

It is very important to notice that robustness of a process module of this kind is evidently dependent on the ability of the module detector to recognize the module in spite of the repeated changes in time of the module's micro configurations: one can suspect that, after all, robustness is "in the eye of the beholder", instead that in the module. This suggests some crucial considerations which, although anticipated here, will deserve a more thorough treatment in section 14.5.2. First, it must be stressed that, as already pointed out in section 2.5 and in many other parts of the current work, modularity is *relative* to the *choice* of a metric between elements of the system. In this case, the case of process modularity in CAs, the chosen metric is a complex metric which "groups" together the configurations of the cells which come to actually constitute the module. For lack of a better way to describe it, we could say that in the case of the example above, this complex metric is precisely the algorithm which performs the spaceship detection: according to the relative view of modularity, we could choose another, different algorithm to detect simple gliders instead of spaceships. The crucial question is: are we free to choose any possible algorithm for detection of process modules? This, will be argued in section 14.5.2, is not a completely free choice, depending on three factors: (i) the computational power of the detection algorithm with respect to the power of the computation we want to attribute to the modular DDS (a complex question which is not appropriate to summarize here, and which will be treated in section 14.5.1); (ii) the computational feasibility of the detection algorithm, which must not be too computational demanding; (iii) the possibility to effectively *find* the detection algorithm.

Figure 5.7: a "spaceship" of period 5 in Conway's Game of Life. This macroconfiguration changes cyclically at each time step through microconfigurations 1 to 5, while moving two pixels to the right for each complete cycle. If we subsume all the 5 configurations under the same "spaceship" concept, we can see the "spaceship" as a "robust" configuration.

[13] Even in CA's which produce quite robust macrostructures, this is not always the case: the fact that a "glider regime" can be maintained throughout the system's evolution, often depends on the choice of particular initial conditions. See section 5.2.3.

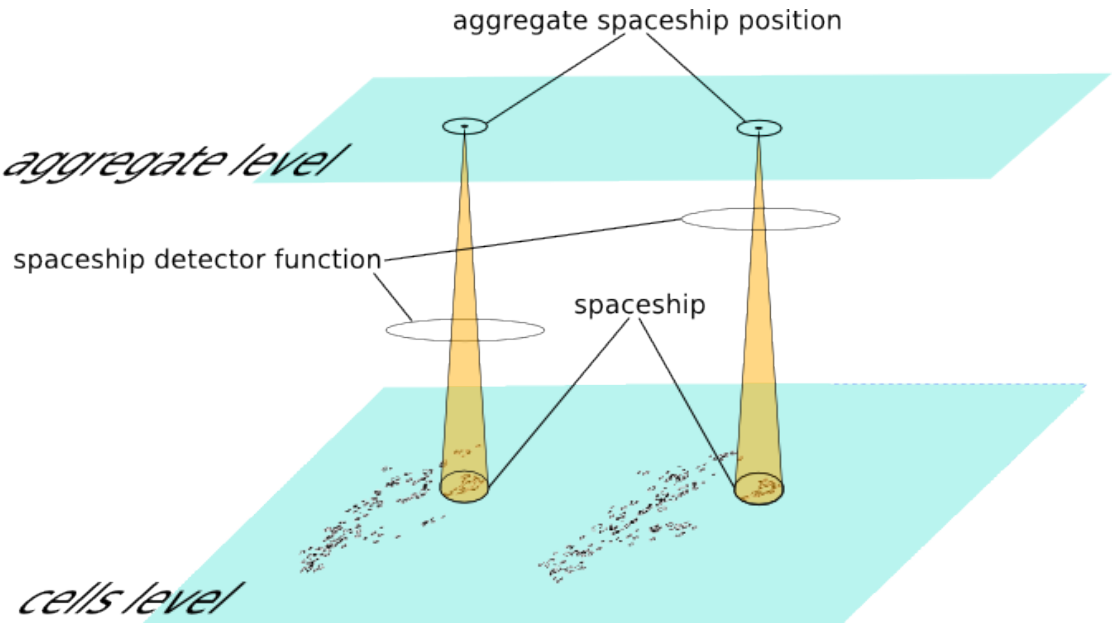

Figure 5.8: the spaceship detector maps any one of the five internal configurations of the period 5 spaceship to a cell of the "aggregate" system, another CA in which each cell represents a spaceship. The position of this cell is the "aggregate" position of the subsystem of the original CA's dynamical evolution constituting the "spaceship". The higher level's dynamics is decoupled from that of the original CA, because the aggregate position of the spaceship advances of 2 pixels every 5 timesteps of the original CA.

Besides gliders and spaceships, other important types of modular dynamical structures in certain class IV CAs have been discovered. Among the most interesting are the so-called *glider guns*, which are structures cyclically "firing" new gliders in certain directions. There are also glider "eaters", which annihilate gliders colliding with them, and glider reflectors, which make gliders change direction after a collision.

5.2.3 Self-organization in CAs

Evolutions of CAs of class IV are highly dependent on the specific initial conditions: started from certain configurations they can display a highly ordered behavior, manly characterized by the diffuse presence of gliders. Starting from other configurations, their behavior can proceed in a much more chaotic manner. Fig. 5.9 shows the very different regimes that can be obtained in the same CA, starting from different initial configurations: the top part of the image depicts a seemingly chaotic dynamical process, while the lower part shows a very ordered process based mainly on gliders and their auxiliary structures, like glider guns and eaters: this kind of dynamical regime is usually called a *glider regime*.

Certain CAs (it is not clear if Conway's GOL is among them[14]), show a particular disposition to enter spontaneously a glider regime of some sort after a certain period of time: regardless

[14] The study of the general properties of the long-term evolution of 2-D CAs is hindered by the enormity of their state space, with respect to 1-D CAs: even for finite CAs, the number of possible initial states is exponential on the number of cells, and in 2-D CAs, the number of cells is quadratically higher than that of 1-D CAs of the same radius, so the cardinality of the state space of fixed radius 2-D CAs is enormously higher than that of the state space of 1-D CAs with the same radius. As a consequence, it is not easy to survey the possible initial states of a 2-D CA. It is possible that GOL shows self-organization, but this is not known in general.

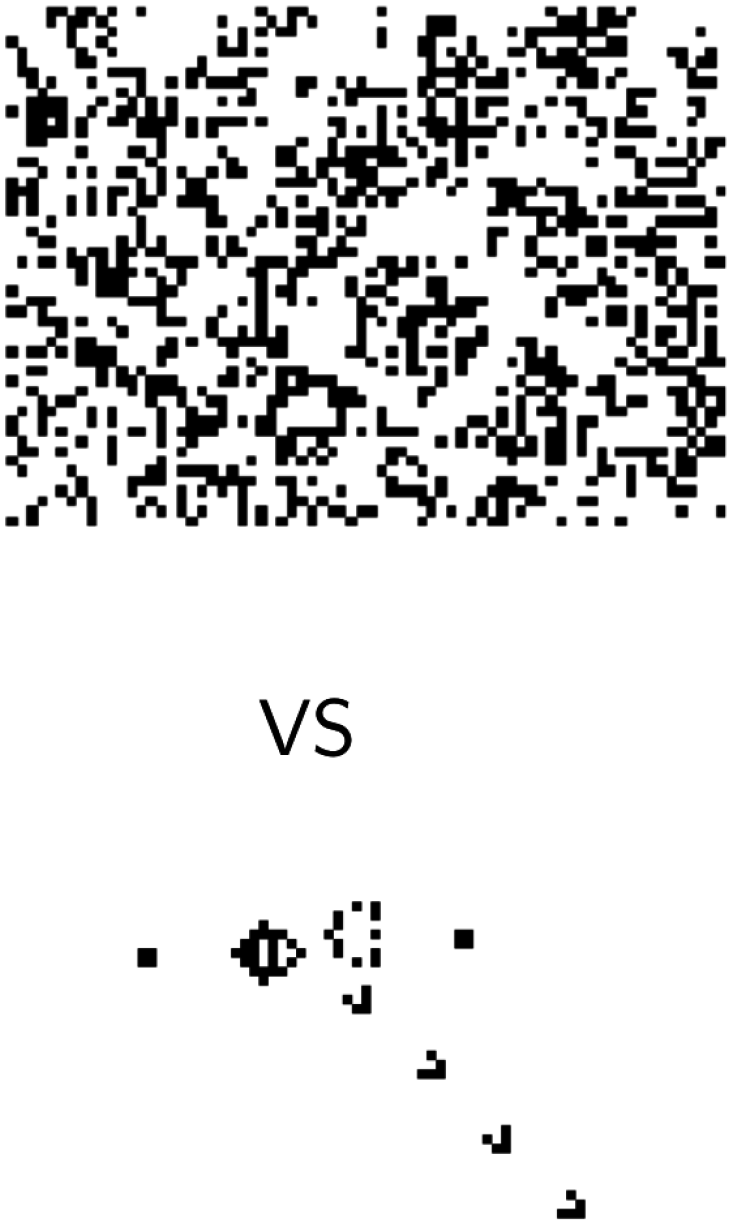

Figure 5.9: two very different dynamical regimes generated in Conway's GOL by different initial configurations. Top: chaotic regime. Bottom: glider regime.

of the initial conditions, in most cases, after an initial transient seemingly chaotic phase[15], the CA's evolution enters some form of glider regime. This is an interesting phenomenon, which reveals that some complex systems possess the intrinsic capacity to self-organize, that is, to produce a complex but ordered dynamical structure without external guidance. This is a form of what can be intuitively described as "computational emergence", in the sense that some form of dynamic order *emerges* spontaneously. Self-organization is a very important phenomenon, for it guarantees that certain CAs will spontaneously enter regimes which can enable them to perform computations, as we will see in the next section.

5.2.4 Higher-level modularity in CAs

All the modular dynamical structures appearing in certain CAs described above suggest the possibility that some form of complex, coordinated, structured process can be realized by a CA at this modular dynamical level of description[16]. And *computations* are typically structured, dynamical complex phenomena, so it is conceivable that, under certain circumstances, CAs can be seen as performing computations. Many studies have indeed highlighted the possibility for gliders and similar dynamical modular structures in CAs to be employed as components of information-processing higher-level configurations, or, in other words, to be employed as components of computational virtual machines implemented *inside* the CA lattice, by means of certain special configurations which can, under certain circumstances, act as machines performing computations. As hinted at in section 5.2.1, it has indeed been proved that certain CAs (namely, ECA Rule 100 and Conway's GOL), are capable of implementing *any* conceivable type of computation with these means.

But, a first problem is immediately raised: in what circumstances in general, and how, a given CA (or a DDS in general) can perform computations? A first answer could be that *any* DDS inherently performs computations, because it is a digital dynamical system, which takes an input configuration of elements (its initial configuration) and, by proceeding step-by-step, transforms this configuration into successive different configurations. If, by some criterion, we deem this information processing to have been concluded (typically, when the CA reaches a loop of period one, that is, a configuration that does not change anymore), then we can take this final configuration and consider it the *output* data of the computational process. Here lies a big problem: certain circumstances must hold, for a processing of discrete configurations to be considered a *sensible* computation: not any processing of tokens can be considered a *computation*. This problem is very thorny and, although having been touched upon in section 1.4.7 of the Introduction, it will be better dealt with in a dedicated section, 14.5.1. Suffice here to say, that, in an intuitive way, to say that a system is computing is to *attribute* to that system the execution of a computation: the same system or process can be considered as computing or not computing at all, or as performing different computations, depending on how we, in a sense, "interpret", or "view" it. Specifically, a mapping between the system's initial (the input) and final (the output) configurations on one side, and a set of meaningful symbols on the other, must be established for us to be in condition to say that the system has performed a certain computation acting on these meaningful symbols (as said, a more thorough discussion of this problem, which seemingly involves the notion of intentionality, is to be carried out in chapter 14.5.1).

Along these lines, it is conceivable that even the evolution of a CA can be interpreted as executing certain computations. In order to do that, a minimum meaningful interpretation must be given

[15] See Hanson & Crutchfield (1997).

[16] I'm using the expression "level of description" on an intuitive basis. I will try to give it a more complete, computational characterization in section 6.6.

of its configurations. For CAs exhibiting glider regimes, as those in Wolfram class IV, the most natural idea is that of viewing the presence or the absence of a glider at a certain point in time at a specific location as the presence of a binary *1* or *0* value, and collisions between gliders as the implementation of logical operations on these values. In other words, the crucial idea is that of seeing the CA as performing a computation, but a computation at a *higher level* with respect to the computation trivially executed by any CA at the level of the application of its rule to cells. I'm employing here the notion of high-level in the sense intended in computer science when speaking of high-level and low-level language, which we have already encountered in section 4.1.4.2[17]. In this view, the high-level computation is implemented by the low-level one starting from particular initial conditions which represent, in the form of gliders and other modular dynamical structures, the data to be processed and the high-level program to be executed (this relation between the initial conditions and the computation performed is a complex matter, and will deserve a separate treatment in section 14.5.1).

To be in condition of considering gliders as bits and their interactions as simple basic computations, a study of the laws governing glider interaction must be conducted. As has been observed, in certain class IV CAs, glider interaction is indeed predictable, because it is affected by law-like regularities simple enough to be synthetically described. But the study of these regularities is not easy anyway because of many reasons. First, in many cases gliders are not simple to detect: while the glider in GOL is a well-defined not very complex isolated dynamical structure, in other cases, like in the examples of class IV rules depicted in fig. 5.3, gliders are embedded in regular complex backgrounds which require that they be first algorithmically "filtered out" by the glider detector in order to obtain the glider as an isolated structure. Often gliders have a complex periodicity, and this filtering requires an adequately complex glider detector. Second, certain CAs display a wide variety of complex gliders, and the number of their possible interactions in slightly different circumstances is high, while these different circumstances of interaction can lead to completely different effects: the outcome of an interaction between two complex gliders can depend crucially on the relationship between the phases of the colliding gliders, and, for gliders with long cyclic period, the number of collision circumstances to consider in order to exhaustively classify all their possible interaction is high. Study of glider interactions can thus be a daunting task: actually, this is a case of mostly inductive inference, because we must infer the set of possible outcomes of an interaction between gliders as a function of the circumstances occurring before the collision. In general, class IV CAs exhibit such a complexity of behavior as to hinder the possibility of predicting, just by examining the CA rule, the outcome of still unobserved interactions between gliders: this unpredictability is a form of "emergence" occurring in these systems, as we will better see in section 13.3. As a consequence, all possible combinations between the circumstances occurring before and after glider collision must be actually observed during runs of simulations of the CA under consideration, in order to give a complete theoretical description of the "laws" governing the glider regime. The number of these combinations being quite high in many cases, the task is daunting.

Nevertheless, classifications of this kind have in some cases been attempted. General observations on process modularity in DDS had been conducted since early works by Stuart Kauffman[18] in the late '60s, substantially improved in the '80s. Pioneering works on the specific analysis of glider behavior have later been those of James Crutchfield and his staff at the Santa Fe institute, who proposed algorithmic ways to filter gliders out of their background to study the laws governing their interactions, an exercise of what they called, by analogy with physics, *computational mechanics*, a discipline devoted to study the laws and structures emerging in the space-time

[17] A deeper analysis of the notion of levels is to be conducted in section 6.6
[18] See section 7.1.2.

behavior of complex DDS. In Hanson & Crutchfield (1997), a nearly complete classification of gliders in ECA *Rule 54* is put forth.

Another researcher at the Santa Fe Institute, Andrew Wuensche, developed since the '90s general methods for the automatic detection of CA rules able to produce process modular evolutions, and for the automatic filtering of gliders and other dynamical modules in CAs, even without previous knowledge of the existence of these structures, as exemplified in fig. 1.2 of the Introduction. In Wuensche (1999), he stresses that, once a CA has entered a glider regime, its behavior can be described at a higher level[19], exclusively in terms of interaction between gliders and related modular structures, without any mention of the underlying micro-dynamics that occurs at the level of single cells and of the application of the CA rule on them: Wuensche stresses that this change of level of description is analogous to how certain sciences, like chemistry, explain phenomena at their own level, without mentioning the underlying subatomic phenomena. The point is that of explaining the behavior of the system making use of *modular*, "high-level" descriptions. As wil be better explained in section 6.6, this can be seen as a form of aggregation, analogous to the original idea of aggregation by Herbert Simon: the CAs dynamics, when inside the glider regime, can be seen as nearly decomposable into dynamical subconfigurations (the gliders and similar modules) which allow for an aggregated representation of the system's dynamics.

This aggregated representation can be naturally thought, as highlighted above, as a *computational* description. Andrew Adamatzky and Genaro Juárez Martínez, bringing on at the University of the West of England a long line of research started by Adamatzky in the '90s, have recently given a *complete* characterization of the gliders' behavior in the *Rule 54* elementary CA, as reported in Martínez et al. (2014). They had already proposed in Martínez, Adamatzky, & McIntosh (2006) to interpret glider-glider collisions as the performing of boolean operations on bits, as exemplified in fig. 5.10. Once having implemented in this way[20] logical functions on a high level description of the CA, other, subsequent forms of higher-level modularity come naturally, analogously to computer programs modularity.

Actually, once the level of boolean functions gets implemented in a computational system, it is easy to proceed, by combination of these logical modules, to the construction of more computationally complex super-modules, and so on. As I try to clarify in section 4.3, however, while it is quite easy to purposely *construct* a modular computation, or computer program, it is less easy to *detect* modularity in an already existent program or computational process, of which the modular structure is unknown. That is, while the practice of modular *programming* is relatively easy, the inverse problem, that of modular detection of computational processes, beyond the level of simple boolean functions, is a practice of *reverse-engineering* and is less obvious. This has been better highlighted in section 4.3 and will be more deeply pondered in sections 14.5.1 and 14.5.2.

[19] This naive idea of level of description will be better analyzed in section 6.6.

[20] The notion of implementation, introduced in section 4.1.5, like many other regarding computation is, I think, in need of clarification: I attempt such a feat in the section dedicated to computation, 4.1.5.2.

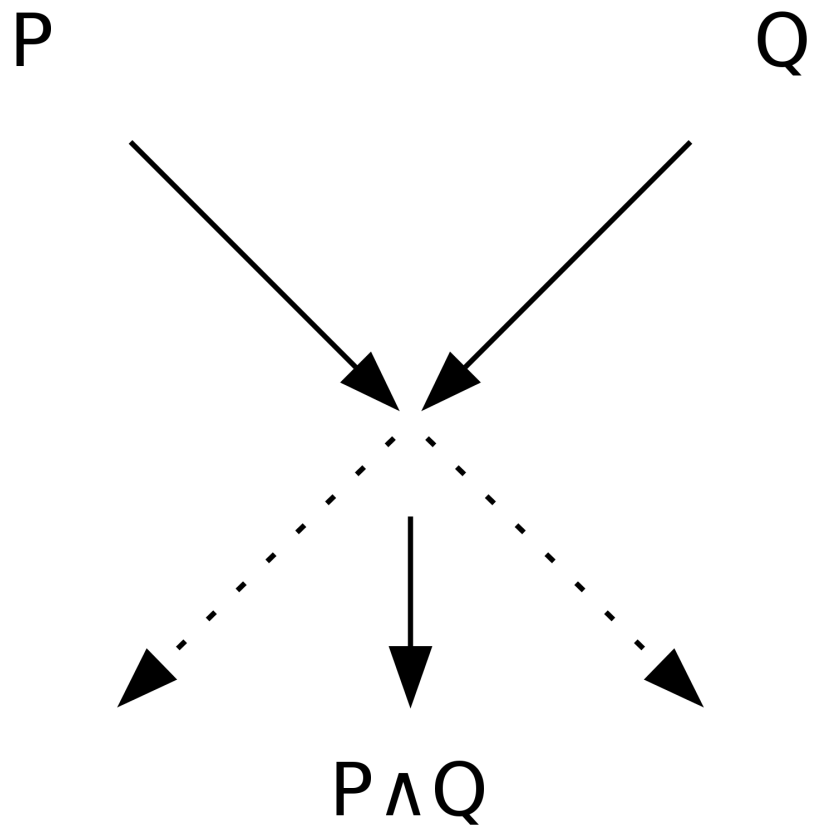

Figure 5.10: a logical AND function implemented by glider collision in the ECA Rule 54, as proposed by Martínez et al. (2006). Presence and absence of a glider are banally interpreted as the *1* and *0* boolean values. The diagonal lines are two gliders on the way of collision. The vertical line represents a third glider, of a different type, which emerges as effect of the collision and annihilation of the other two gliders: presence or absence of this glider implements the logical AND, because this glider appears if and only if the other two were present.

Chapter 6

Thinking about modularity

In this chapter I will proceed to some further speculations about modularity, specifically involving the relation between forms of modularity (structural, dynamical, functional), in order to proceed to a tentative definition of several concepts related to hierarchical modularity and levels of description, and to an exposition of how I intend their epistemological and ontological status: I view modularity as referring to *descriptions* of a system, and as such as pertaining to *representations* or theoretical *models*, not to the *actual* ontology of a system. I will then highlight in general the importance of modularity for explanation, prediction and experimentation, which are the most important aspects of scientific research.

6.1 Modularity and its properties: summing up

From all that has been said in the former sections of this work, we can recapitulate some properties of modularity which appear evident.

A module can be:

- a single atomic element
- a subset of a system's elements

In the latter case, elements internal to the module appear more closely and durably related to each other than to elements belonging to other modules.

Intuitively, a *module* has the following characteristics:

- it has a sufficiently well-defined boundary;
- it has to be able to retain its unity in a range of external conditions:
  - that is, it possesses some sort of *robustness* in the face of external perturbations;
- it can occur in multiple *copies*;

Tipically, a *system* is *modular* when it has the following characteristics:

- a number of its subsytems can be seen as modules;
- the interconnections *between modules* are weaker than the interconnections *between the elements composing a module*;

6.2 Structural and dynamical modularity

It is my opinion that Simon's original exposition of near-decomposability is not so clear-cut, and that some analytical reflections are now in order, considered how much Simon's approach has informed all the subsequent literature on modularity. In particular, some sort of confusion can possibly arise about modularity of a structure and modularity of a dynamical process.

In his seminal 1962 writing on complex systems, Simon states a general principle of modularity: a system is hierarchically modular if it is composed of subsytems which are more intensely connected on the inside, than one another. In his words:

> In hierarchic systems we can distinguish between the interactions among subsystems, on the one hand, and the interactions within subsystems – that is, among the parts of those subsystems – on the other. The interactions at the different levels may be, and often will be, of different orders of magnitude.[1]

Then Simon proceeds to explain *near-decomposability*. The example he proposes, that of an office subdivided in rooms and cubicles, could perhaps give rise to some misunderstanding: at first, it seems that modularity here is reflected in the *physical* structure and organization of the office: cubicles in the same room are more interrelated to each other than to cubicles external to the room, because of their spatial location and because the subdivisions internal to a room are thinner than walls separating different rooms. This, at first sight, seems to be a case of *structural* modularity, based on the relation "being near or separated by a thin diaphragm": this relation is strong between cubicles of the same room, and weak between cubicles of different rooms. Structural modularity is a *static* concepts: it holds at any time.

But then, the thermal *dynamics* of the systems are taken into consideration: precisely due to the structural modularity highlighted above, a *temporal scale differentiation* occurs in the systems' dynamic process: because thinner walls are less thermally insulating, intra-room thermal exchange turns out being more intense than inter-room exchange, leading to a *faster* evolution of the temperature distribution in each room towards thermal equilibrium, faster with respect to the evolution of the thermal distribution across different rooms. This makes the cubicles in a single room appear more interrelated one another *temporally*: the higher speed of evolution of the room's subsystem is due to a higher rate or frequency of thermal exchanges between cubicles[2]. The relation taken into consideration here is *temporal rate* of thermal exchange. This is a form of what I would call *process modularity*, or *dynamical modularity*. In the example by Simon, process modularity turns out to be related to the structural, static modularity of the system, which in turn is due to the physical structure of the walls and disposition of the rooms.

I will try to sketch here a quite rough, tentative distinction between structural and dynamical modularity. This analysis could appear to some insufficiently thought-out from a philosophical standpoint, but it is to be intended here as only a first provisional analysis, based on loose intuition, attempting to kickstart a series of considerations. It is my hope that these considerations could shed eventually some light on a convoluted group of concepts regarding modularity.

6.2.1 Structure and process

Let's start with the difference between *structure* and *process*.

[1] Simon (1962), p. 473-474.

[2] What at the aggregate level of thermodynamics appears as a flux rate, at the molecular scale corresponds to a higher frequency of exchange of fast moving molecules.

An object has a *structure* (let's say a *shape*) which does not change in time: this structure defines the object's identity and allows us to recognize the object as precisely that object: should its structure change significantly enough, we would be in presence of a *different* object.

An object or an entity can be in different *states* at different times. A *change of state* is something which does not significantly change the *structure* of the object.

A *process* is the dynamics of some consecutive changes of state of an object occurring in time. Processes can be continuous or discrete. To keep things simple, in this work I will mostly treat discrete processes, that is processes which occur at discrete timesteps, and in which the space of possible states is discrete as well. Given that a process does not, by definition of *change of state*, alter the object's structure, we could say that the object's structure *supports* the process, or that, in other words, a process occurs *on* the object's structure. To give some intuitive example, it will suffice to think of an electric circuit: the *structure* of the circuit is given by the electrical conductive connections between its components, (e.g., metal wires running between switches, lamps, and so on. . .). This structure, often represented as a schematic diagram, does not change. But, the circuit itself can be, at different moments in time, in different states. At least, a circuit can be in one of two different states: a state of electric current flowing (the state "on") or the state in which the circuit is not supplied with current, and it is "off". Or, in other cases, the circuit can be in one of a continuous set of possible states, in which a state is identified by the amount of current which is flowing in the circuit. A *process* occurring on this circuit is the sequence of changes of state occurring to it.

Some objects can have a structure which shows some degree of modularity: this is a form of *structural modularity*, and as such it is usually static (except in the case in which the object itself is changing). But such objects can often undertake changes of state, that is, they can support processes. *Process modularity* can often take place as a spatio-temporal modularity of the process, that is, of the dynamics of the changes of state occurring on the structure.

Here is the important point: in general, structural modularity does not always coincide with process modularity, but their interrelation seems intuitively likely: parts more closely interrelated *structurally* influence more easily one another, or more rapidly.

An obvious form of structural modularity is modularity based on the *geometrical* shape of the system and the *spatial* relation between its parts. But, attention must be paid here: *spatial* and *structural* modularity do not necessarily coincide: this is quite clear in the case of *networks*, for example networks of interconnected computers. In the case of a network, its *structure* is an abstract, non-spatial, fixed configuration, in which physically distant nodes (e.g. servers residing in different continents) can be directly linked structurally, by means of a direct connection between them without intermediaries: in the abstract structure of the network, these servers would appear adjacent, even if they are physically far apart. In these cases, it is not the *spatial* disposition of the parts which counts, but the *topology*, that is the abstract structure of connectivity of the system, which is indeed represented by a network, undestood as a mathematical object. A network, in general, is an abstract object basically conceivable as a set of elements, the nodes, linked together. Networks are mathematically studied in graph theory and, in a somewhat more experimental fashion, in network science[3].

Even if not necessarily based on spatial relations[4], topological modularity is nevertheless a form

[3] Modularity in networks has been extensively treated in chapter 3.

[4] Although it *could* be based on spatial relations. Actually, the spatial structural modularity of an object is due to the topological structure of the system as it is represented in the common euclidean space, in which the common metric of euclidean *distance* is taken as the relation between elements of the system on which to apply the generic criterion for near-decomposability when assessing modularity.

of *structural* modularity, because a *fixed* topological structure is usually assumed[5].

What about process modularity *in networks*? A difference between Simon's approach and modularity in networks is that, as we have seen, with his notion of near-decomposability Simon considers *dynamical* systems: systems in which evolution in time of their parameters is studied. Networks, on the other hand, are usually seen as *static* structures, and search for modularity is a search for *structural* modularity, performed by analyzing the network's fixed structure. For this reason, since networks, as mathematical objects, lack a dynamical aspect, is seems the main considerations about *timescales* in modular hierarchical systems are not immediately transferable to hierarchical modular networks.

Nevertheless, dynamics can be implemented *on* a network: in most cases, a network is to be considered the basic fixed infrastructure *on which* certain dynamical events take place: this situation applies, for example, to transport networks, where the dynamics is that of quantities transported over it, electric power networks, where the dynamics it that of the amounts of current circulating in the network, genetic networks, whose dynamics is the sequence of activation/deactivation and transcription of genes, or social networks, which have a dynamics of human interactions.

We could even consider a "topological" representation of the classic near-decomposability example by Herbert Simon[6] by means of a weighted *network*: this network representation obtains if we substitute, as in fig.6.1, a node for each cubicle, and a weighted link for the heat exchange rate between couples of cubicles[7]: this way, a *network* comes to represent the structural modularity of the system in Simon's example.

Given that a processual dynamics can be easily implemented on a network structure, we could ask ourselves if networks can show some form of process modularity, or dynamical modularity over the structural one, and to what extent modules identified according to structural properties can also be dynamical modules. Dynamical modules are modules whose dynamics derives from, and affects more strictly and in a more temporally coincident manner, the nodes inside the module than nodes external to it, giving rise, as in Simon's example, to dynamics at *different timescales* according to their level in the hierarchy: slower timescale at the higher level, faster timescale at the lower level, the level of its subsystems. As we have seen, in Simon's example structural modularity does indeed induce process modularity in the system, and this seems quite likely to occur in general.

In networks, links between nodes can be abstractly and generally conceived as communication channels: they channel influences of some kind between nodes. In the case of *weighted* networks, the channel itself possesses a degree of "conductivity" of these influences, a degree represented by its weight, which can vary between different links. *Structural* modules in the network are then identified as those subsets of nodes whose elements are linked together by links which are more conductive than the links the same internal nodes of the subset entertain with nodes of other subsets. This higher conductivity would certainly correspond to a higher *dynamical* influence between nodes of the same *structural* module, giving rise to a *dynamical* modularity which will probably turn out to correspond to the structural modularity found in the system, albeit

[5] There have been studies on networks modifying in time: as reported in section 3.1.3, an important one is Barabási & Albert (1999), which considers a particular modality of *growth* for networks. However, the typical community detection algorithms try to detect modularity in a network's *static* structure. Besides, many kinds of networks are by their nature static or with a very slow changing rate: for example, organizational charts, or, in biology, genetic regulatory networks.

[6] Presented by Simon as in fig. 2.3.

[7] Actually, as explained in section 3.2.1.3, a weighted network is not strictly needed, for we could use, with some approximation, a non-weighted multigraph, in which high heat exchange rate is represented by *more* edges connecting *the same nodes*, and low rate by less edges.

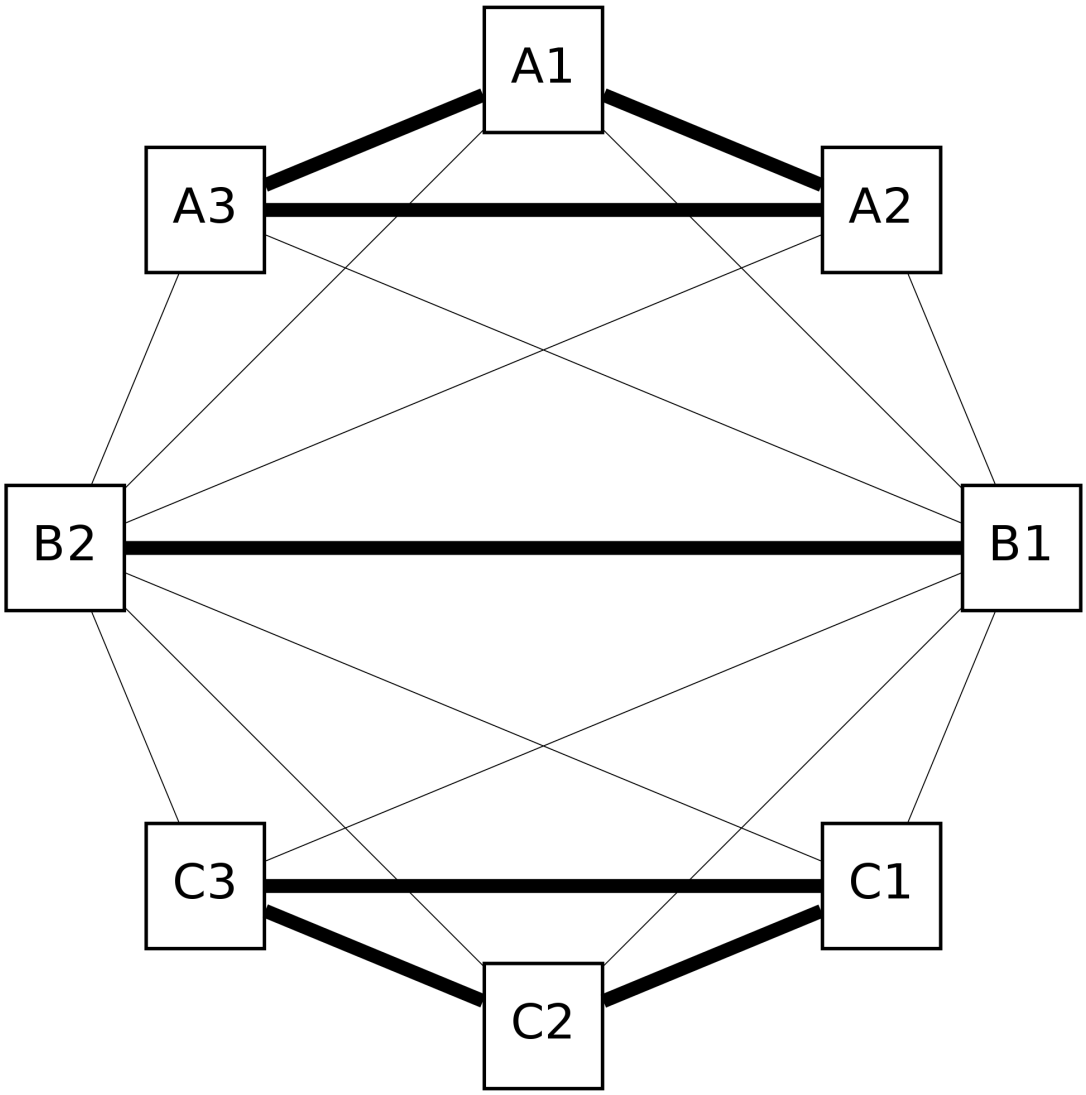

Figure 6.1: topology, rendered as a weighted network, of the system used as an example for Simon's near-decomposability (see section 2.2.3).

with varying degree of coincidence between the two, due to the possibility that non-linearity in the dynamical behavior of nodes could compensate for difference in link weight. Even in non-weighted networks, where structural modularity is detected by measuring not the weight of intra-modular links but their *density*, higher density of connections between nodes in the same module should usually mean higher frequency or probability of dynamical influence between them than between any of them and any external node. Although this is not guaranteed to always be the case (because non-linearity can interfere), the correlation between a situation of high intra-module and low intermodule connectedness and the decoupling of dynamical timescales has been observed in modular networks, as explained in section 3.2.4.1.

Thus, it stands to reason that, in weighted networks, the weaker (compared to those internal to modules) connections *between* modules, regardless of the specific nature of the connection, could lead to an inter-module coordination of the dynamics of different modules which turns out to be *slower* than the intra-module dynamics, where certainly communication or reciprocal influence between nodes in the same module is to be expected as more frequent.

This could be expected also for the reason that a modular network has presumably evolved or been created with modular structure not by chance, but to allow an optimized performance: it would not probably make much sense to give weak connections to elements which have to perform at the same time scale and with high-frequency of interaction: accordingly, it is to be expected that nodes connected by strong links are supposed to communicate, or influence each other with a higher frequency of interaction than nodes connected by weaker links. This could extend even to non-weighted networks with information-processing capabilities, where intermodule connection are supposed to be used for information transmission at a lower *frequency* than intramodule connections. This could allow the extension of Simon's statements about differences in time scale according to hierarchical levels, also to *boolean networks* performing information processing or computations[8].

Pan & Sinha (2009) supports this view. The authors first note that often, in natural information processing systems like brains, it is desirable that local dynamics occur at a finer time-scale than global ones, in order to maintain synchrony between local areas processing specific stimuli, while large-scale synchrony occurs in pathological states, like epilepsy, and it is *not* a desirable feature of normal functioning. Pan and Sinha's study makes then use of a model consisting of a moderately modular network which also possesses the small-world property[9], to show that the network's structural modular configuration makes coordination within local clusters occur much more rapidly than global coordination, thus making the network's dynamics exhibit the diversity of timescales typical of hierarchical modular functional organization.

6.3 Modularity is relative

Changing the metric, a system can show a different modularization, or not appear modular at all. Some examples:

- fig. 6.2 shows a network, which is modular according to the metric measuring density of reciprocal links between the nodes (this is the typical metric for algorithmic community detection in networks. See section 3.2.1);

[8] Some very important questions can be raised about computation and modularity, questions which i will touch upon in section 14.5.1.

[9] See section 3.1.2.

- fig. 6.3 shows a network which is modular according to a metric which measures the difference of the numbers naming the nodes.

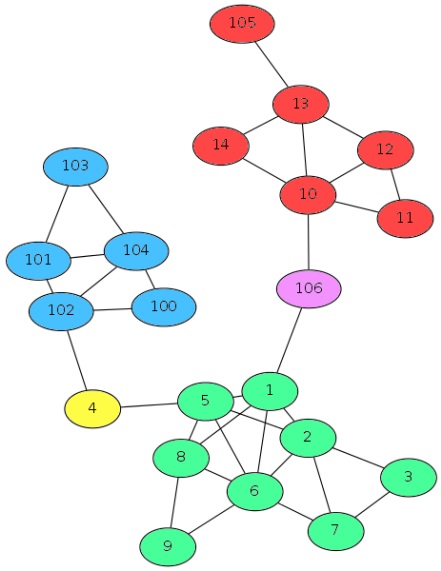

Figure 6.2: a network, which is modular according to the density of reciprocal links between the nodes. Distinct modules have different colors.

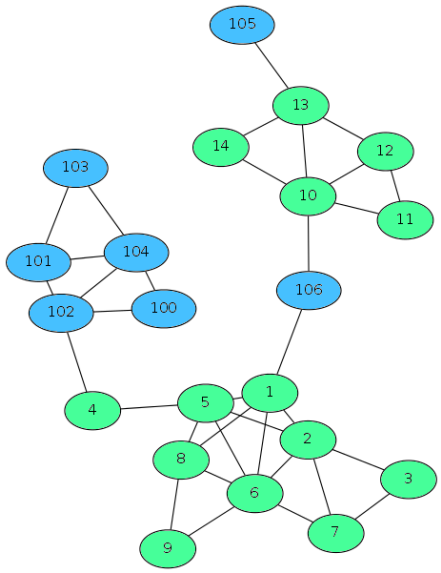

Figure 6.3: a network, which is modular according to the metric which mesures the numerical distance of the nodes' names. Distinct modules have different colors.

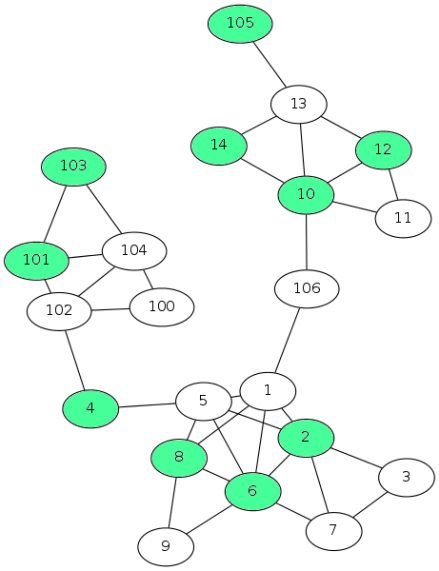

Figure 6.4: a network which is modular according to a complex property. Distinct modules have different colors.

Besides a metric proper, more in general a *relation* between elements of the system can be taken as reference, in order to assess its modularity. Changing relation, a system's modular structure can change, or disappear.

A relation can also be the sharing of a *property*, as in fig. 6.4, where the property is: *even numbers lower than 20, or odd numbers higher than 100.*

As can be easily seen, in these three examples, the network has a fixed structure, but in each case modularity is assessed based on different metrics, and the corresponding modular structures end up being different.

6.4 Forms of functional modularity

So far, we have treated *structural* and *dynamical* (or *process*) modularity, trying to see if and when they are related. What about *functional* modularity? Is it present in complex systems? And, is it related to structural and process modularity? I will proceed to distinguish several cases of functional modularity. Some of these distinctions come from the different ways in which the notion of function can be understood, as highlighted in section 9.

Identification of modules in a given system is an operation which can be effected by way of *structural* or *functional* considerations: for example, a module in a physical mechanism can be seen as a structural unit in virtue of its being spatially cohesive, in the sense of being composed of parts which are in some degree spatially grouped together, or as a *functional* unit in virtue of its capacity to accomplish a specific function, despite it being possibly constituted of parts spatially spread over a large area and not pertaining, apparently, to the same substructure. In general, these two kinds of decomposition depend on the type of *relation* we take into consideration between elements of the system in order to proceed to the identification of modules. But it all

depends also in an essential manner on what conception of "function" we are sticking to when we talk of "functional modularity".

Probably, the simplest notion of function is the mathematical or computational one: a function is a relation between two sets, according to which to each element of the first set corresponds one element of the second set. It seems natural to claim that for a module to support this form of function, the system must be dynamical, in that the performing of the function cannot occur in an exclusively static system[10]. For a module to perform a function in this mathematical sense, it must be possible to recognize a set of possible inbound interactions affecting the module, which are to be considered the *inputs*, and another set of possible outbound interactions with and toward other modules, which are to be considered the *outputs* of the module. It is easier to visualize this in the case of networks, where the module performing the function must be equipped with a set of input links and a set of output links. The module's dynamical performing of a *function* consists in this case in the fact that, based on a certain configuration of interactions of variable magnitude (interactions of whatever nature) applied as an input, the module produces a certain configuration of interactions toward external modules which constitute the output corresponding to the given input: this way, it performs its input-output function. This function can often be considered a computational function, especially in discrete system. More in general, in analogue systems the module performs a mathematical function.

In *networks*, mathematical or computational functional modularity appears in its simplest form in the case of *network motifs* (see section 3.2.2), which have been seen, since their introduction, as functional modules performing information processing or control functions[11].

From the *structural* point of view, network motifs manifest in the form of simple, recurring subnetworks, of wider directed networks: in these networks, because of the directionality, *structural identity* between recurring copies of a subnetwork warrants the likelihood that also *functional* identity occurs between them, for the functionality is presumed to depend on directionality: as said, a functional module is to be seen as an entity with input and output links, and the internal "information processing" is obviously dependent on the direction of information transfer between the internal nodes.

Functional modularity of motifs can also be seen as a case of Cummins-type functional modularity: typical network motifs mentioned in papers which analyze biological or computer networks have often been immediately identified as functional modules: for example, the feed-forward and the feedback 3-nodes loops[12] correspond to typical, long-known functional modules in control theory. I think this can be characterized as functional modularity as intended by Robert Cummins (a form of functionality discussed in section 9), because here the function is not the isolated module's input-output mapping, but a function which explains certain features of the contextual higher-level system, that is, a function having a role in the explanation of the overall function of the global system: for example, certain network motifs contribute to explaining the feature of maintaining certain properties of the whole system stable by means of control systems based on negative feedback, as in homeostatis.

[10] I don't see how consideration of modules as mathematical functions in a *static* structural-only modular system could make sense, at least for interesting systems outside the abstract, platonic realm of mathematics. Perhaps, we could consider such possibility in some very particular cases, such as when in a containment hierarchy, let's say a tree of "smaller boxes into larger boxes", a certain branch of the static tree subdivides recursively according to a certain mathematical function, such as 2^n (in the case of a binary tree-shaped branch, with n corresponding to the hierarchical level inside the subtree). But, I'm not interested in such a kind of static "functions" here, and I will not take them into consideration. I'm looking here for functions that must be *worked out*, or computed, and not which present themselves as already deployed.

[11] See for example Milo et al. (2002) and Alon (2006).

[12] See section 3.2.2.

In their explanatory role, network motifs realize the property of modularity which consists in the possibility of reducing the amount of information needed to specify the whole network: by reducing the specification of the entire network to a description of the repeated occurrences of the same simple types of building blocks and their interrelations (in addition to interrelations to other building blocks which do not recur, but appear only once), a *lossless compression* of information is achieved in the system's explanation and description.

It is arguable that the same network motifs can also allow for economy of *process* description. This would be realized if *structural* network motifs showed up to coincide with *dynamical* simple modules. There are, as we have seen, a priori reasons to expect that this coincidence holds, but only empirical research can confirm it in real-world networks.

Several works seem to have confirmed functional-structural coincidence for network motifs. For example, Mangan & Alon (2003) confirmed that certain types of motifs, namely certain feed-forward loops, circuits well known in control theory whose function is that of delaying response to inputs (*coherent* feed-forward loops) or to accelerate it (*incoherent* feed-forward loops), actually perform the theoretically predicted function in models of gene transcription networks constructed from data reported in known genetic databases, such as of those of *Escherichia Coli* and *Saccharomyces Cerevisiae*. Mangan, Zaslaver, & Alon (2003) also confirms *experimentally* this kind of coincidence, showing that the feed-forward loop performs a processing function on chemical input signals in real-world *E. coli*.

An important empirical confirmation of functional-structural coincidence for network motifs is highlighted in Grochow & Kellis (2007): using an innovative and much faster algorithm than the traditional ones[13], these researchers were capable of discovering a complex 15-nodes motif occurring more than 27,000 times in the protein-protein interaction network of *Saccharomyces cerevisae*, and whose occurrences are often overlapping. This motif is overlapping over a limited set of 29 nodes, a set which turns out to correspond to the core of the transcriptional machinery of *S. cerevisae*: in a way, as the authors claim, the algorithm re-discovered the gene transcription cellular machinery (a Cummins-style *functional* complex) based solely on the *structural* information provided by the network representation of the cell's proteins interaction.

At a larger scale than that of motifs, it turns out that motif generalizations and network themes or other superstructures made of motifs could also plausibly represent kinds of modules fulfilling specific functions. Going further upward in scale, it is therefore conceivable that the network *communities* found by algorithms for community detection[14] can be expected to be composed of, or at least comprise, among other structures, network motifs or higher superstructures. Such communities should reveal themselves as *functional* modules, besides being structural ones. This structural-functional coincidence seems more likely in the case of directed networks, for in this case the structural module can be seen as a functional module with a specific set of inputs and outputs. Typical algorithms for finding community structure, however, act on *undirected* networks.

Algorithms to detect community modularity in *directed* networks have been recently proposed[15]. An example is the method put forth in Leicht & Newman (2008), which can be considered an extension to directed networks of the classic algorithm from Girvan & Newman (2002), based on a modified definition of the modularity measure Q.

[13] See also section 3.3.2.
[14] As described and discussed at length in sections 3.2.1 and 3.2.5.
[15] See section 3.3.1.

The fact that the algorithm for community detection in *directed weighted* networks by Arenas et al. (2008) already described in section 3.3.1 is based on the density of *motifs* inside *communities*, makes it a natural extension of the modular description by means of motifs to descriptions based on communities which are *made of* motifs, thereby pointing in the direction of a completely modular description of the system, both in the sense of modules understood as robust internally densely connected substructures weakly interconnected with each other, and in the sense of modules as repeated "building blocks" of the system. In such a kind of description, individual nodes and links are substituted by a hierarchy of high-level communities linked one to the other by a few links, where each community is composed by smaller communities, and so on, going down in description scale, to sub-communities, to motifs and, eventually, to the single nodes of the original network and links between them. This kind of complete hierarchical structure, as we will see in section 6.8, can allow for a comprehensive multilevel kind of mechanistic explanation, which is what is usually sought for in many scientific disciplines. It is in general plausible that in such a hierarchy each community could be seen as a complex performing a high-level specific function: in a directed network, if a community or subcommunity is composed of *network motifs*, which are detected by a criterion similar to that of *motif modularity* described above, structural-functional coincidence in the hierarchy is guaranteed by the fact that motifs are simple *functional* building blocks, at least in the sense of a mathematical or logical function, understood as a relation holding between input and output. A higher-level community composed of motifs can then be seen as a "black box" with a limited group of links pointing toward it, which represent the "inputs" and other links coming from it, which would be the "outputs". Given a *structural* decomposition of such a kind, it seems feasible and convenient to give a *functional* description of the system in terms of it.

If what we seek is an explanation in terms of the theoretical notions of a given discipline like biology, which sees functions as *etiological*[16] functions, however, it is not *guaranteed* that each of these identifiable functions in the organism turns out to correspond to one of the particular communities in a hierarchy obtained by *structural* decomposition of the system: a *function* in this etiological sense is, quite vaguely, a function which has revealed as capable of increasing the organism's fitness in past generations, and it could very well be a function distributed among nodes belonging to different and distant *structural* communities. This could happen, for example, for neural systems, which often show the "small network" property[17], which allow distant parts of the system to be closely connected, allowing their concerted functioning in order to fulfill a function. In such cases, a specific functional description in terms of a *structural* decomposition obtained by a modularity detection algorithm would end up being not a good explanation, because the structural decomposition would not coincide with the functional one: mathematical or computational functions, like those performed by motifs or communities identified with this method could appear not very "natural", according to previous knowledge about the observed system and the current scientific theory involved. For example, according to Mazurie, Bottani, & Vergassola (2005), based on observations of an integrated network of transcriptional and protein-protein interaction in *Saccharomyces cerevisae*, it is not the case that, in general, network motifs undergo a special evolutionary pressure with respect to other non-recurring subnetworks, so it seems that they are not functional modules in the sense of complexes fulfilling evolutionary selected functions. Based on considerations by Mazurie and collaborators, it can be argued that network motifs are not *modules*, in the sense that often they cannot be seen as basic blocks of a nearly decomposable systems, for they are "embedded in larger structures and entangled with the rest of the network"[18]. There are however opposite stances: for example, Conant & Wagner

[16] See section 9.
[17] See section 3.1.2.
[18] Mazurie et al. (2005), p. 9.

(2003) found that many of the same motifs in the genetic network have evolved repeatedly in *Escherichia coli* and *S. cerevisae*, hinting to their being the product of selective pressure toward their appearance, and thus that these motifs should represent biologically significant functions.

The settlement of this kind of questions should be referred to further empirical observation and considerations about compatibility of the candidate modular decomposition with the rest of the discipline's knowledge. However, it already seems that in many cases the structural/functional (in the functionalistic, or Cummins-like sense) decoupling is not very likely or widespread: there are apparently valid theoretical and empirical reasons in support of the idea that structural and functional modularity go often together, especially in biological systems. While it is indeed plausible that, in a certain percentage of cases, the majority of very simple network motifs can not be seen as subsets of the network representing functions in the functionalistic sense, functionalistic modularity appears in general more likely when larger subnetworks are taken into consideration, as in the case of communities, especially in directed networks. A corroboration of this hypothesis comes from findings like that of Guimerà & Amaral (2005b)[19]. In this work, coincidence of community modularity and functional modularity is assessed and confirmed for metabolic networks in twelve organisms, comprising procaryotes and eucaryotes. The method employed is innovative enough to deserve a short digression: first, *community structure* in the metabolic network is detected by a custom algorithm. Then, another algorithm gives each *node* of the network a *role*, based on its linkage inside its module, and its linkage relative to the rest of the other modules. This two-phase method connects structural properties of nodes relative to the *communities* in which they are located with their *functional roles* in the network's dynamical functioning. Two main kinds of identified roles are: the role of peripheral nodes, which are mostly linked to nodes inside their module, and the role of connector hubs, that is, nodes which realize inter-module connections. Using these categories, Guimerà and Amaral construct what they call a *cartographic representation* of complex networks[20], a type of representation which conveys more information with respect to a typical representation based solely on community structure. An example of such a representation is in fig. 3.14 of section 3.2.3. In this "cartographic" representation, information is given about module size, connection strength between modules, and about the role certain particular nodes fulfill in the general network connectivity: for example, singular *connector nodes* are highlighted, nodes through which important inter-module connections pass, and other characteristics of certain other node roles. By comparing this representation with well-known experimental data about the metabolic role of each node in the complex metabolic chains of the organism under observation, it appears that the structural "cartographic" representation quite well matches the known metabolic *functional* subdivision. It seems, then, that in many cases, a node's *role* as detected by the algorithm is quite well related to an actual functional biological role that the metabolite corresponding to the node actually fulfills in the whole metabolic network. This correlation can bring to the hypothesis that important connector nodes, whose elimination would disrupt communication between entire modules, are actively conserved across species by natural selection. This turns out to be the case, as a further analysis confirms: different roles are subject to different evolutionary pressures.

To sum up, the method employed by Guimerà and Amaral gives a strong contribution to the idea that modularity detected by *structural* observation can coincide with *functional* modularity.

[19] Better described in section 3.2.3.
[20] See section 3.2.3.

6.5 Modularity of the dynamical model and prediction

In dynamical systems, which can be modeled by a set of equations expressing a certain dynamics, prediction of the system requires solving this formula analytically, or simulating the system by means of the formula used as a computational model. In the first case, the state of the system can be calculated at any arbitrary point in time of its evolution, by means of the analytical solutions to the equations describing its dynamics. However, recourse to simulation is very frequent because most interesting dynamical systems have a non-linear dynamics which cannot be usually analytically solved, but which allows only for the calculation of the system's global state at the next timestep, starting from a current (initial) condition[21].

Prediction of the system can be effected in these cases by a repeated application of the dynamics' formula for each timestep until the exact time is reached at which we want to predict the state of the system: prediction at n successive timesteps requires iterated application of the formula n times. This renders prediction at many timesteps in the future a computationally expensive task.

For a system composed by an enormous number of elementary parts, this non-aggregate formula would have to cite the same number of variables, each one tracking the state of a single elementary part. Such a formula could be difficult to produce, or, if found (possibly by automatic means), prediction of the system's behavior in the long run could be difficult to simulate, for the high number of variables could require a too high computational effort, if we have to calculate the function for a high enough number of steps.

Fortunately, in certain cases the system turns out to be nearly decomposable, and functional modularity in the sense of near-decomposability often allows for a more efficient representation of an acceptable approximation of its dynamics.

The very example made by Herbert Simon of the office room system[22] offers a simple illustration of this property. In that case, near decomposability entails that each single room reaches thermal equilibrium very rapidly internally (for its internal cubicles are highly thermally coupled), in such a way that, when considered at the timescale of the whole system, each room can be represented, with an acceptable degree of approximation, by a single variable which represents its average temperature. Thus, in place of many variables (each one representing the temperature of one cubicle), a single one is substituted, by aggregation. This results in a simplification of the aggregated *formula* expressing the dynamics of the system, albeit at the cost of a loss of "resolution": the new simplified formula does not permit to calculate the temperature of the specific sub-room cubicles. This is a form of modularity of the mathematical expression of the dynamical model of a system. We could call this form of modularity *dynamical model modularity*.

The main point is that, if we are interested in determining some more general parameter of the system at some point in the future, this prediction can be effected by means of the *simplified*, modular, aggregated formula, thus with a more economical computation than the calculation of the original, non-simplified formula. This saving of computational resources should result in a saving of the computation's run time, allowing for a longer-time prediction than when using the original formula.

That the aggregated formula of a nearly decomposable system is computationally simpler than the original non-aggregated one, is, however, not guaranteed: in certain cases of nearly-decomposable systems, the update function representing the dynamics of the aggregate system turns out being

[21] See sections 2.2.1 and 5.1.
[22] See section 2.2.3.

of complexity comparable to that of the original update function representing the dynamics of the non-aggregate system. Economy of an aggregated model is reached only when we take as an aggregated model a *simplified* model of the original system, a model which *approximates* the original system's dynamics, allowing the aggregated model's behavior to be faithful to the behavior of the original system *up to an acceptable error*. But, in significantly non-linear systems, any simplified model will diverge in time in an unbounded way with respect to the behavior of the original system, and thus any chosen threshold of maximum acceptable error would be crossed, should we let the simplified simulation run for a long enough time. So, in the case of non-linear systems, dynamical model modularity has a limited scope or timespan of application.

6.6 Hierarchical levels of descriptions

In this section I will sketch, through a series of informal definitions, a framework for the theoretical analysis and systematization of the intuitive notions of *description* and *level of description*, a framework specifically aimed at simplifying reasoning about hierarchical modular descriptions. While non-formal definitions will be used, these definitions put forth a technical use of the aforementioned terms about descriptions, often inspired by computational notions, a use that is in some way different from their typical ordinary language meaning.

6.6.1 Abstractions

As is easy to understand, any description requires unavoidably a process of *abstraction*: it is simply impossible to describe an object, or a process, in *all* its aspects, because the number of aspects is unbounded, even for a simple object[23]. Each specific description must thus focus necessarily only on a finite and often small set of aspects of the object or phenomenon to describe, a set of aspects which is each time chosen according to the aims and purposes of the observer. In my view, there is always this pragmatic component of descriptions, which intervenes in every case, also in scientific research, where the relevant aspects to observe and to explain are chosen in accordance with the observer's specific needs and with the current expectations of the scientific discipline in question.

Abstraction is thus a way of considering a phenomenon under some of its aspects, neglecting the others. An abstraction can be supposed to be a computable procedure: an empirical phenomenon, constituting raw data, can undergo an algorithmic process of abstraction which operates by selectively processing only some aspects of these data, yielding an *abstract representation* of the phenomenon.

Of course, this could seem a too naive view, because there is probably no such thing as a purely empirical phenomenon: many well-known philosophical positions, from Kant to a significant part of current philosophy of science, deny that senses, let alone scientific instruments of observation, can unbiasedly report real-world phenomena as they are in themselves. Raw data obtained with scientific observation is already a *representation*, produced by some form of primary abstraction: *measure* itself is such an abstraction. But for a process of abstraction to be seen as a computational procedure (and this is indeed my proposal), it must already act on *representations*: as discussed in section 14.5.1, computations act on representations.

But what is a representation? It is, I propose, itself the result of an abstraction: the output of a procedure (the abstraction) which, taking as input some data, produces an abstract represen-

[23] At least for observable macroscopic phenomena or objects. Probably, this would not hold for theoretical entities, like elementary particles, which are however unobservable. I do not wish to enter into this problem here.

tation of these data, that is a description which focuses on some of their aspects neglecting all the others. An *abstraction* is then a computation, acting on abstract representations, yielding other abstract representations. It seems there is the risk of some infinite regress here: the *first* representation, the "bottom level" one, from which all the other are obtained, must not be the product of an abstraction itself, but must be some form of "raw" data. This is a hard question. An answer to it would involve defending some specific position on the realism-antirealism axis. It is not, however, in the scope of this work to embark on such basic questions on the relation between real, actual objects or processes and their primary, phenomenic representations[24]. So, I will take here for granted that, when we speak of *empirical* phenomena, we are actually *already* speaking of *representations* of some sort.

What I advocate here is an *epistemic* stance[25] about scientific descriptions and explanations, which sets aside questions regarding the relationship between the real world and its representations. What interests me here is to reflect on representations and their transformations, which produce other representations, and on the relationships between properties of the original representation, the transformation, and the resulting representation. As better maintained elsewhere in this work,[26] I see science as something dealing with representations in this sense, and specifically scientific explanation as an eminently communicative and cognitive question[27] dealing with representations.

How an empirical phenomenon is represented, however, is not dependent on a completely free choice[28]: each scientific discipline has a basic *ontology* and its theories deal with the objects of this ontology and their properties. In other words, each science has its *natural kinds*. It is this basic ontology, which is usually already given (at least for well-established sciences[29]), that can be further abstracted. But, as stated above, I think this basic ontology must be considered already a *representation*. This representation can vary, across specific sub-disciplines of a science, or across sciences: for example, particle physics deals with subatomic particles, while chemistry deals with atoms and molecules. In general, each science, or also each specific line of inquiry, has its basic ontology of elementary entities, properties and operations on them.

Given that it is my intention to consider the transformation of representations as a computational task, in the digital sense of computation considered by computer science, I will specifically consider only *digital*, or in general *discrete* representations. Reducing the scope of the following considerations to discrete dynamical processes and systems could be judged to be an oversimplification. This could well be the case, but a discussion of this aspect would probably require entering the very vast debate about the nature of the discrete and the continuous in philosophy of mathematics, or possibly quarrels just as complex and vast (if not more) about the nature of computation and the ultimate nature of physical reality[30]. There is not enough space here to touch upon these questions. As will become clearer in what follows, I want to argue for the

[24] Although, as we said in section 1.5.1, my theoretical position bears in some way on this problem, too.

[25] in the sense of the word employed by, among others, Cory Wright and William Bechtel, as explained in section 10.

[26] Section 1.5.1.

[27] This also contrasts with an "ontic" view of explanation, as highlighted in section 10.

[28] For a debate, see, again, section 1.5.1.

[29] Of course, in newly emerging sciences or scientific paradigms, one of the first duties is precisely that of discovering (or "inventing", according to a less realistic vision) the basic ontology of that scientific branch. And it is obvious that a scientific ontology is almost never fixed and complete, although in well-established sciences, like chemistry or molecular biology, the basic ontologies are quite well established. In certain cases, such as psychology, sometimes the basic ontology is apparently already provided by common-sense notions.

[30] Questions such as: is computation constrained by *physical* limits, or is it a purely *mathematical* question? Is physical reality continuous, or at very small scales it shows a basic discreteness? Is reality the result itself of a computation, as claimed by *pancomputationalism*?

fact that scientific theories and explanations tend to assume a modular form. In accord with the epistemic stance announced above, in what follows we will deal only with *theoretical models* and their transformations, not with "real" phenomena, and we will assess questions regarding the modularity of these systems. As I tried to show in section 5.1.1, discreteness is trivially a form of modularity. Specifically, we will consider here only *discrete* models, taking them as paradigmatic cases of modular systems by means of which to try to shed light on the notion of modularity in general. This "digital" simplification will characterize the rest of this work.

6.6.2 Preferred languages

Every discipline has not only a basic ontology, but also a basic *language*, understood as a *vocabulary* constituted of terms referring to the kinds of the basic ontology: terms for types of elementary entities, their properties, and for operations on them. I call this language the *preferred language* of a discipline[31].

For example, very roughly stated, the preferred language of chemistry is that of atoms, ions, molecules as the entities, valency, atomic and molecular weight and other properties of them, and reactions between the entities; the preferred language of molecular biology is that of biological macromolecules, their properties, the various kinds of interactions they entertain and the transformations they undergo, and interactions and operations affecting the complexes constituted by these molecules, up to the cellular level; the language of cognitive psychology is that of mental representations, certain kinds of relations between them and operations on these representations, understood as computations.

6.6.3 Abstraction, aggregation and multiple realizability

As intended here in a semi-technical sense, an *abstraction* is a computable procedure acting on a representation. This definition is quite liberal, and does not pose per se constraints on what computable transformations the initial representation can undergo. But, in its ordinary language sense, and even philosophical, traditional sense, the word "abstraction" certainly involves the idea of something present in the observed representation being neglected, in order to obtain a more coarse-grained, less detailed representation. This neglect of information can be effected by simply discarding certain aspects of the observed representation, or by partitioning the range of their possible values, and by taking an aggregate value as representing whole subsets of the partition, or, in general, by transforming the set of the observed representation's aspects into a smaller set of aspects, or by aggregating groups of entities of a representation: the point is that the *abstract* representation obtained from the original one has *less* properties than the original one. This entails that more, different representations described in the detailed, non-abstract way, can end up, under some abstraction, being represented as the *same* abstract representation: an abstract representation represents the *set* of the representations which differ precisely in the properties which the abstraction has neglected, while having in common the set of properties that the abstraction has considered. In other words, an abstraction entails *multirealizability*: the same abstract representation can be obtained starting from a set of different basic representations, which are its realizers. Thus, an operation of abstraction entails that more than one of the

[31] A language can also be seen, in a somewhat more formal way, as a collection of types, types which can be instantiated by tokens. This reminds of similar definitions which were given in sections 5.1.1 and 5.1.2, definitions pertaining to the concept of "digital", or "discrete". We can then see a language as a collection of discrete types which can be instantiated by certain tokens. In computer science (see section 17.1 of the Appendix) a *language*, understood in the discipline's technical sense is precisely a collection of types: it is a collection of *strings*, which are its types (strings which, in turn, can be seen as decomposable in ordered collections of symbols, which are again types).

original representations end up being mapped by the abstraction function into a single "abstract" representation: an abstraction in the philosophical sense is a function *many-to-one*[32]. So, if we apply an abstraction function to an entire set of representations, the new set of abstract representations ends up having a smaller cardinality than the original one (at least in the case of finite sets[33]).

I propose to call *proper abstractions* what I described above, that is computable functions which map representations to representations in a many-to-one fashion.

I want also to include into the category of abstractions what I would call *lossless abstractions*: functions which produce elements which are not multiply realizable, that is, functions which map representations to representations *one-to-one*[34]. Examples are the identity function, or computable functions which produce "abstractions" which are simply different representations of an object at the same level of detail of the original representation.

It results from the definitions above that an abstraction always produces a set of representations whose elements have *at most* (in the case of lossless abstractions) the same degree of detail of the original representations, and usually *less* detail, when a proper abstraction is used[35].

6.6.4 Transformation of languages by abstraction

A very important point is that, from a given *language*, another one can be obtained by a process of abstraction, abstraction understood as a computable procedure which acts on the constituents of the original language. For example, the language which describes society as a system of interrelated individuals can, by abstraction, produce another language which describes *groups* of individuals, and relations between *groups*.

A *programming language*, which to all intents and purposes is a language in the sense intended here (comprising entities, the *variables*, and instructions to manipulate the variables), can, by abstraction, be translated into another, usually higher-level[36] computer language. A higher-level language, as expected, given that it derives from the original one by abstraction, has *single* instructions which correspond to *sequences* of instructions of the original, low-level language. It is more "abstract" (in the common sense) and coarse-grained than the original one. It is, also, multi-realizable by more than one lower-level language. For example, a program written in a high-level language like C, can be compiled, that is translated into, an unbounded set of different machine-level languages, each corresponding to a different CPU architecture[37].

In the technical sense which has been discussed in section 4.1.5, each instruction of a high-level language constitutes a *specification* of a more or less elementary computation which can be implemented in an unbounded number of ways by other languages. As an instance of the use of a programming language, a specific *program* written in a high-level language constitutes a specification which can be implemented in an unbounded number of ways by programs written in other languages.

---

[32] That is, it is a *non-injective* function.

[33] Of course, in the case of infinite sets this is not guaranteed: it is a basic fact of set theory that the cardinality of an infinite set obtained from another infinite one by removing a part of its elements can still have the same cardinality of the original set: for example, if we perform an "abstraction" by ignoring the sign of an integer number (function "absolute value"), we obtain the set of natural numbers starting from the set of the integers, but the two sets have the same cardinality.

[34] That is, *injective* functions.

[35] Of course, I do not consider as *abstractions* transformations which "add" detail to a given representation (whatever that could mean).

[36] See section 4.1.4.2.

[37] See section 4.1.4.2.

6.6.5 Descriptions and simulations

I consider here the notion of *description*. We have already introduced the notion of *language*, a collection of types of basic entities, their possible properties and possible operations on them.

I propose to view a *description* as the *dynamical* description of a phenomenon, expressed in terms of a given language: what I call a description can supply a *theory* of the described phenomenon, in the form of a *dynamical model*, namely a *discrete* dynamical model (or *DDS*), which allows for some (possibly limited) dynamical reproduction, or *simulation*, of some aspect of the phenomenon's dynamics.

A *description* deals dynamically with entities of a language, where "language" is to be understood in the sense described in the preceding sections: a description acts on tokens instantiating the vocabulary of the basic ontology provided by a certain language, and evaluates and transforms these tokens according to functions and operations chosen among the operations specified by this same language. From this standpoint, a description coincides with what in section 5.1.1, following a definition by John Haugeland, was called "digital system", and basically is a kind of computationally capable system.

A clarifying example comes immediately to mind: a description so understood is equivalent to a *computer program* written in some *programming language*. Computer programs are a form of dynamical system, and the "language" in which they are written is formally a language in the computer science sense recalled above. My definition of *description* is thus akin to a definition of computer program, but I understand it as more general, including discrete dynamical systems which must not necessarily be imagined in the form of the classic computer architectures. In other words, a description is a specific discrete dynamical system of some sort: it can be a distributed parallel system like a cellular automaton, or a boolean network, but also a more classic serial computer, like a specific Turing machine or a specific computer program running on a von Neumann architecture.

From a more formal, computational standpoint[38], a *description* is a *machine* acting on strings of a *language*, a language produced by a certain abstraction of the original phenomenon.

I call *simulation* the *execution* of a run of the machine constituting the description, starting from a certain initial state. For example, in a CA, the initial state is the global configuration of the cells of the CA, and a simulation is the evolution of the CA starting from that initial configuration.

6.6.6 Languages and levels of description

I propose to tie the notion of *language* in which a description is expressed, to the notion of *level of description*, by means of some definition.

A *level of description* is a set of languages.

Let's examine some cases relating languages and levels of descriptions.

Two descriptions expressed in the same language are certainly *at the same level* of description. But, as we have seen, languages can be transformed by abstraction into other languages. A *lossless* abstraction[39], which does not reduce the detail of the original language, that is, that maps, in some computable way, the original language's terms to the terms of the new language

[38] See section 17.1 of the Appendix.
[39] A type of abstraction admitted by my definition, as specified in section 6.6.4.

in a *one-to-one* way, produces two languages that are at the *same* level of description. Intuitively, this means that, at least in the case of finite languages, the two languages have the same number of elements. This seems to me to adequately capture the idea of level of description: a level of description is intuitively a *level of detail* in which something is described. A language produced by an abstraction which does not discard possible distinctions expressible in the original language, maintains the level of detail of that language. Thus, a level of description is an equivalence class of equally detailed languages.

A *proper abstraction*, which takes a language and maps its elements many-to-one into another language, produces a *higher-level language*, that is, a language which is at a higher level of description with respect to the original language. The latter can then be seen as the *lower-level language*, belonging to a lower *level of description*. Of course[40], a higher level language turns out being multiply realizable by lower-level ones. This seems to me to capture in a quite faithful way the intuitive idea of "more abstract" as "higher level", and "less abstract" as "lower-level".

Given that the relation between levels is transitive, a *hierarchy* of levels of description quite naturally derives from the possibility to iteratively transform languages into other languages by the repeated application of abstraction functions.

All the above can be extended from languages to *descriptions*:

- two descriptions are *at the same level* when they are expressed in the same language, or in languages at the same level, that is, languages that can be obtained one from the other by means of a *lossless* abstraction.
- a description is *higher-level* (or *at a higher level*) than the original one when the language in which it is expressed can be obtained, by means of a *proper* abstraction, from the language in which the original description is expressed.

6.6.7 Redescriptions

A *description*, understood as a machine plus the entities it acts upon, can be in turn *re-described*: that is, the description can undergo, in order to yield *another description*, a set of computable transformations which can act upon *both* the language in which it is described, usually by means of some kind of abstraction function, *and* on the structure of the *machine* which manipulates the entities.

I thus propose to call a *redescription* any computable function which, fed with a description (in the form of a certain machine expressed in a certain language), produces another description, in the form of another machine and another language. The redescription can change the structure of the machine, the language, or both.

In order to simplify a bit, in what follows I will often call "redescription" not only the computation which *transforms* a description into another, but the newly obtained *description* as well, leaving to the context the charge of disambiguating between the two uses.

6.6.8 Validity of a redescription

Given that we are dealing with dynamical systems, a redescription must be checked for its *validity*: this is a term of art in the field of computer simulation of dynamical systems, which pertains

[40] See section 6.6.3.

to the following question: is the simulation accurate in following the dynamics of the simulated system? For this to be the case, it is necessary that the simulated result does not *diverge* in time from the actual dynamical behavior of the simulated system: the *error*, that is, the difference between the simulated behavior and the behavior of the actual system must stay inside a limited range, as time flows.

In our case, we start with a dynamical model, a *description*, which gets *redescribed*. The redescription is *valid* if the obtained description's dynamical behavior does not diverge in time (up to a chosen maximum error) with respect to the dynamical behavior of the original description. This condition must be enforced because, given the liberal definition of redescription stated above, a redescription, in producing a derivate description, could in countless ways, by way of ignoring functional and relational aspects of the description to redescribe, produce a derived description which only barely resembles the original one. The point of redescribing is precisely this: if we are interested in only *some* aspect among all the aspects of the dynamical behavior of the original description, we could redescribe it in a way that takes into consideration only that aspect alone, or that aspect together with some other one which is necessary to consider in order to obtain a valid redescription.

In general, a redescription ends up being not valid when it simplifies too much, or neglects altogether, the aspects of the original description which are *relevant* for the aspect of its behavior the observer has chosen to focus on. If we were to take a realistic stance toward the observed original description (an *ontic* stance[41]), we could say that these are the *causally relevant* aspects of the original phenomenon: check for validity ensures that the redescription is *causally relevant* for the phenomenon to be explained. This condition of causal relevance is a condition which must be assured when producing mechanistic explanations of a phenomenon, mechanistic explanations which constitute re-descriptions of the observed phenomenon, and that, to be explanatory, must of course be valid redescriptions[42].

The validity condition for redescriptions resembles what I call the *aggregation condition* for nearly decomposable systems[43]. This should not come as a surprise, for aggregation of a nearly decomposable system is precisely a *redescription* of the dynamical system. It is carried on in a way that takes for granted that the system can be treated as if it were decomposable. This assumption of nearly-decomposability, as seen, introduces an error which must be checked to remain within certain accepted bounds. When this happens, the aggregation condition, which is a form of validity condition, holds. An aggregate description of a nearly decomposable dynamical system is what we will call a *modular* redescription in section 6.6.10.

6.6.9 Preferred descriptions

As we have seen in section 6.6.2, every discipline has a *preferred language* with which it formulates its theoretical descriptions and explanations. These are *descriptions*, in the sense expounded above, and can be called the *preferred descriptions* of a discipline.

Every science or line of inquiry provides such a set of basic descriptions, expressed in the discipline's *preferred language*: in every science there is a basic theory, or set of theories, which in principle describe the dynamical interaction of the entities of the basic ontology. Of course, not every science has a *complete* basic theory: this is true, if ever, only for long-established sciences, when not in a phase of paradigm change. But, the goal of any science is precisely that of finding

[41] See section 10.
[42] The question of mechanistic explanation is treated in section 10.
[43] See section 2.2.1.

such a kind of basic theory, and in many sciences a basic theory is in the making, albeit sometimes almost permanently so. Besides, there must not necessarily be a single basic theory, nor a unifying one: according to many, for example the advocates of the so-called "new mechanistic" view[44], special sciences such as molecular biology do not seek for a single theory composed of law-like all-encompassing generalizations, but for the discovery of *specific* mechanisms: in this case there would not be a basic theory, but a set of preferred descriptions, the descriptions of the mechanisms. Nevertheless, it can be argued, a form of common basic theory is still present in these sciences as well: the basic vocabulary of molecular biology is indeed that of organic macromolecules, and a basic description of the molecules' interactions and behavior is shared in the discipline, albeit in chemico-physical terms. I would call a basic description like this, understood as kinds of entities plus a theory of their interaction, the *preferred description of a discipline*.

I propose also to use the term "preferred description" to refer to *types* of descriptions, relying on the context to disambiguate: for example, we could say that in network biology a description in terms of linked nodes is a preferred description of a system such as the genetic regulatory network of an organism.

But a preferred description can be a *specific* preferred description, pertaining to specific cases: I would call for example the genetic regulative network *of yeast*, when represented as a network of linked nodes, the preferred description of a single case, from the standpoint of network or systems biology.

Thus, given a certain specific phenomenon, I would call a *preferred description of the phenomenon* any description (description understood as a dynamical machine) making use of the preferred language *of the discipline* inside which the phenomenon is taken into consideration: while some disciplines have all-encompassing preferred descriptions, any specific description can be a preferred description of *some* phenomenon.

It is important to note that the notion of preferred description is *relative* to a discipline: the same system can be described by different preferred descriptions, when considered from the standpoint of different disciplines. For example, the same social group is to be described differently by economics and sociology. As we will see in what follows, the preferred description can also vary *within* a discipline according to the hierarchical level of description taken into consideration, where hierarchical descriptions are feasible.

To sum up, the preferred description of a phenomenon from the standpoint of a certain discipline, is easy to determine: it is the kind of description which is typically employed in that discipline. Certain disciplines even appear and define themselves based on the proposal of giving some new type of description of already studied phenomena.

Some examples of preferred descriptions: in neurosciences, preferred descriptions are certainly descriptions in terms of nervous systems seen as networks of neurons, possibly hierarchically organized, whose dynamical functioning produces the observable behavior of an organism. So, preferred descriptions of specific phenomena will be given in neuroscience as descriptions of *neural mechanisms*. From a different standpoint, the standpoint of behaviorism, the same phenomenon of animal behavior is described in terms of the whole organism interacting with the environment, and specific preferred descriptions will be patterns of sensory-motor interactions between the organism and the surrounding environment. To consider a non-empirical science, the preferred description of a cellular automaton[45] is certainly its description in terms of cells and of the local

[44] See section 10.
[45] See section 5.2.

CA rule acting on them, albeit in certain cases other descriptions can be given of the system (for example, in terms of gliders).

The purpose of this work is to investigate, taking into consideration the requirements of scientific explanations, the relation between preferred descriptions and their transformations yielding possible re-descriptions. To this aim, I will stick to a highly-simplified view of science: I will take into consideration a science which observes phenomena which are already representations, and moreover, that are discrete, computable representations, representations which are themselves the product of a computation and can be subject to further computable transformations.

Inside this simplification, scientific disciplines produce *preferred descriptions* in the form of discrete dynamical models of these observed phenomena, preferred descriptions which are, then *simulable*. Of course, incomplete basic theories, or specific mechanistic models, which are the only theories or models provided in actuality in many fields of real-world special sciences, are effectively simulable only in a limited range of conditions. But, the presupposition involved here is only that of an *in principle* simulability.

6.6.10 Modular redescriptions, aggregated redescriptions, explanatory redescriptions, robustness and validity

Generalizing the original intuition of near-decomposability and aggregation, we can consider an *aggregate redescription* a description, derived from the original description, which makes use of entities, or variables, each of which represents some form of aggregate behavior of a *set* of variables of the original description. While aggregation in Herbert Simon's examples consists in aggregating sets of variables by means of a simple arithmetic function, such as the arithmetic mean, the aggregation operation can conceivably be effected according to any computable function on the set of variables to aggregate. Actually, any computable *proper abstraction* can be seen as an aggregate value, given that a proper abstraction takes a single facet, or a subset of aspects of an object or phenomenon, and cites this single aspect or this set of aspects as the distinguishing facet of the phenomenon: if the abstraction is rightly chosen, and it reflects the relevant aspect (the aspect relevant to the phenomenon's dynamical behavior[46], or other prominent features, according to the observer's interests) of the phenomenon, then the abstraction can be considered an aggregated representation of the phenomenon, because its selected aspects come, in a way, to "stand for" all the other aspects of the phenomenon.

In computational systems, even the subdivision of the system into computational modules, each module considered as a black-box between input ad output channels, is the product of a form of *aggregation*, in the sense that to the detailed description of all the variables involved in the internal functioning of the module, like in the original description, is substituted a single "variable", which is the "name" of the box: this "name" is the *specification* of the program module: as already highlighted in section 4.1.5.1, specifications of programs can often be seen as "aggregate" values of the program they specify. This holds also from the *dynamical* standpoint: in a program module seen as a black box, to the dynamical behavior of each variable inside the module is substituted a single aggregate behavior, which is the function that holds between the inputs and the outputs of the box: that is, its *program specification*, which is a way to "name" in an aggregate way the overall dynamical behavior of the module (however, as I explained in section 4.1.5.1, I think not all types of program specification can be seen as "aggregate names").

Any redescription whose machine acts on entities representing aggregates of the entities of the original description can be seen as an aggregate description. But, usually, we need an aggregate

[46] This reflects the concept of *validity*, which is treated in section 6.6.8.

description which tracks *significant* features of the original descriptions. That is, the aggregate entities must reflect some "robust" aggregate properties of the corresponding original entities, relative to the researcher's interests. For example, describing peoples as family groups, when interested in a genealogical description, is describing them in genealogically robust terms, while aggregating them on the basis of their preferred food would not be a robust aggregated description, because it does not reflect any genealogically robust metric (though it would be interesting from a gastronomical or sociological point of view, probably): aggregated values so measured would not persist across the analysis of different samples of people, while the family relationships would. These aggregate values can be seen as representing robust subsets of the set of entities of the original description.

In general, an aggregate representation of a given representation (I remind that we are always talking here of operations on representations, never of operations on "real" objects) is a transformation of a language into another one by means of a proper abstraction: in this case, the aggregation function. A language obtained by proper abstraction from an original one is always, by definition[47] a *high-level* language with respect to the original one.

So, an important point to note is that an aggregate redescription is always situated at a *higher level of description* with respect to the level or the original description. This stems from the fact that an aggregate redescription is precisely expressed in a language obtained by means of a *proper* abstraction, the aggregation, on the original description's language, and by definition a language so obtained is a *high-level* language with respect to the language of the original description.

As we have seen, according to the general definition given in section 2.1, a *module* is a partially isolated *robust* subset of a system possessing a well-defined boundary. Robustness of a module is its robustness relative to the chosen relation or metric used for the module detection, that is, its resilience to perturbations of the context in which it is immersed, perturbations measurable by the chosen metric. A module can be seen as a group of entities possessing, by way of its robustness, a sort of identity as a group. The module's identity can be represented by a single placeholder for the whole module, which can be considered the module's *name*. This is what is done when, for example, a network gets coarse-grained, as described in section 3.2.5: in the coarse-grained network, a single node corresponds to an entire *module* of the original network. We could say that a single node of the coarse-grained network description "names" a module of the original network description. This name of the module needs not be conventional, but can, better, explicitly reflect some property which is common to all the elements constituting the module, usually a common property related to the metric employed in the modularity detection phase. The module's name can in any case be considered a robust *aggregate* value of the elements constituting the module[48]. So, a modular coarse-grained redescription can be considered an *aggregate* robust redescription.

In *dynamical* systems, this robustness of the aggregate redescription is even more important: if we want (as we usually want) to make the aggregate system match, or *track* in some significant way the dynamical behavior of the original system, without diverging from it too much in time, that is, if we want the aggregate redescription to be *valid*, robustness must affect also the aggregate *dynamics*: each aggregate entity must be spatio-temporally robust, that is, it must have caught a functional subdivision of the original description's processual dynamics into *process modules*, where the term functional is to be understood as describing the fact that a part of a system's processual dynamics performs a defined role which contributes to the global functioning of the overall system in a recognizable way (this conforms to the "explanatory role" conception of

[47] See section 6.6.4.

[48] Because even a purely conventional name reflects the common property of the module's entities which consists in belonging to the same module.

functions, as explained in section 9). This role comes to constitute the "name" of the module, which is an aggregate representation of the function it fulfills in the overall functioning of the system. Functions so understood constitute a certain machine manipulation on a part of the original description's entities: the single function can be seen as a sub-*description* of the original description constituted by a specific machine (let's call it a sub-machine) which operates on a subset of the original description's entities. The function this sub-machine operates on this subset of entities can, as said, be named by the role it fulfills in the overall functioning of the description to which it belongs, but can also be seen as an input/output function, taking as input a subset of entities, each one in a certain state at a certain time, and producing a different configuration of states of these entities at a later time as the output. The whole original description can then be seen as constituted of a structured set of subdescriptions interacting by means of groups of entities which they manipulate in turn, and that can be seen as realizing input and output connections between the functional sub-descriptions[49].

When this condition holds, that is, when the original description can be redescribed as a structured system of sub-descriptions interconnected by dedicated channels, each subdescription understood as a particular machine fulfilling a certain function on a subset of the description's entities, the description is *nearly-decomposable*. In a nearly decomposable description, there is *functional modularity*. The fact that the description is nearly decomposable, means that it is so redescribable with a certain approximation. This can happen when the functional modularity detected is not perfect, that is, when isolation between the subdescriptions is not complete, and some subdescription can interfere with the internal processing of another subdescription by manipulating some of its internal entities. There is, otherwise, perfect decomposability and perfect functional modularity when communication between subdescriptions happens only along the dedicated input-output channels. To make use of computer terminology we have already introduced in section 4.2.3, in this case there is perfect data encapsulation and information hiding.

All the above can be better visualized in the case of a description constituted by a computer program. In this case, we detect its modularity by individuating parts of the code which act on only a subset of the whole set of variables of the program (this detection of program modularity can be automated by *program slicing*, as seen in section 4.3.4). Once these parts are individuated, the program can be represented as a system constituted of these subroutines interconnected by well defined input-output channels.

A nearly-decomposable description can be immediately redescribed in an aggregate way: to each functional sub-description which comprises it (the subroutines in the computer program), can be assigned in the high-level redescription a *single operation* of the high-level machine (a high-level function or instruction in a high-level programming language, for example). To the entities of the low-level language we can make correspond aggregate entities in the high-level redescription, or, in certain cases, also the same non aggregated entities. This depends on the degree of validity we want to attain: when redescribing computer programs in a high-level language, we usually do *not* want to introduce approximations in the program's behavior, but simply to redscribe the same exact program in a more humanly-understandable and manageable way. In this case the same detail of data is to be manipulated by the low-level machine and by the high-level one. In case of descriptions of other kinds of system, this perfect validity is not strictly required, and we can get by with some acceptable degree of approximation. In this case (the example par excellence is Simon's example of the office rooms) we can employ aggregated entities at the higher-level redescription.

[49] This whole process of decomposition into sub-descriptions corresponds to functional analysis (treated in section 9.2) or functional decomposition as a phase of mechanistic explanation, as expounded in section 10.

An important point to make is that, when the higher-level redescription is intended *for explanatory purposes only*, even in the case of computational systems or other systems which in principle would require perfect validity, we can make use of aggregated entities or aggregated variables in the high-level description. This is possible because, provided that the functional modularity represented in the high-level modular redescription is in principle valid, for explanatory purposes which exclude the actual *running* of a simulation (of the program), we can abstract from the actual fine details of the data to be processed, and limit ourselves to only cite, in an aggregate way, the identity of modules of data. For example, when explaining by means of a flow-chart, we could indicate the input of an image-processing high-level instruction as "image" or "image data", while in an actual running of the same program written in the same high-level language, the same image processing instruction has to be provided with the actual pixel-by-pixel data constituting the image to be processed. But inside the epistemic conception of explanations which I put forth, an explanation is to be considered as connected to a communicative act, and this act does not require, at least in certain circumstances, the convey of all the details which constitute the actual phenomenon to be explained. So, in the explanatory use of the same program, it is not necessary to specify the same amount of data, and aggregate values can be used, without losing in precision. On the contrary, when aggregate data are used inside the *simulation* operated by the dynamical model, loss of precision is often consequent.

Even in the exclusively explanatory use, however it is necessary that the proposed high-level redescription, or, better, its high-level *machine*, acting eventually on non-aggregated variables, be *valid* with respect to the original one. Otherwise, we would not be proposing an adequate *explanation* of the original phenomenon (that is, of the original description): an explanation should exhibit a description which reflects the relevant features of the dynamical description which has to be explained, and this is ensured precisely by the *validity* of the redescription: if this condition does not hold, it means that the redescription does not reflect in a sufficiently accurate way (or does not reflect at all) the relevant features of the dynamical functioning of the original description. In a mechanism, these are the causally relevant features, the features which have to be outlined in order to say that the phenomenon has been explained. A description which does not catch these features is explanatory irrelevant: it is not an explanation at all.

Thus, the operation of finding a valid *modular* redescription amounts to the finding of a valid aggregate redescription, that is, a suitable subdivision of the variables of the original description into *modules*, and to the finding of a suitable *modular update function* (that is the machine acting on the modular level) which gives as a result a *valid* redescription, where validity, as said, is to be understood as explained in section 6.6.8, and means that the new modular redescription must track in an acceptably faithful way the aggregated dynamical behavior of the original description. The acceptable degree of approximation of this tracking is, as always, dependent on the purposes and aims of the researcher.

In the case of aggregate redescriptions, the obtained description can result *invalid* if our supposition of nearly-decomposability, or modularity of the original description, turns out to be false: this could happen if the redescription we employed to turn the original description into a modular description is not accurate enough to catch the *actual* modular structure or modular dynamics of the original description with sufficient precision, or if this redescription ascribes to the original description a modular structure which is not there at all.

For example, in the case of networks, as we have seen, the fastest algorithms for community detection are not very precise, and they could produce modular descriptions which do not reflect any sensible modularity in the network: most algorithms for community detection, in many cases, actually "see" modularity even in completely random networks. The obtained modular

description would most probably be deceptive, because if a dynamics were to be implemented on the structure of the original network, in most cases its dynamical behavior would diverge in time from the behavior of the modular description, for the reason that it is unlikely that such an approximate and fortuitous redescription has caught the relevant causal relations of the original description.

In certain cases, redescriptions which are valid only in a limited range of time with respect to the starting time of the original description of a dynamical system, or which in general are valid only in a limited range of certain values of the variables of the original description, can be accepted, but this must be explicitly stated. In some cases, these partially valid redescriptions are the only possible ones, as we will see.

A modular redescription, as highlighted above, aggregates in some way more entities of the original description in order to make them correspond to a single entity in the modular redescription. In a purely structural modular redescription, aggregation is simply a matter of finding the communities of low-level entities which will, by aggregation, come to correspond to single entities in the modular redescription.

In a discrete dynamical system (*DDS*), however, any entity at any given time is in a possible *state*,[50] and the aggregation function must take care of this dynamical aspect as well: in a DDS, all the single *states* of the variables constituting a module come to be combined in a single aggregate state, which is the state of the single variable which in the redescription represents the module. Also, in a discrete dynamical system, the entities in the modular redescription must be processed by a machine in order to track, inside this redescription, the dynamical behavior of the original description. In a modular redescription, the *dynamical model* of the original description gets itself decomposed into modules, which come to constitute the machine of the modular redescription. This usually means that, as in the case of programming languages, each module identified in the machine of the original description is a *functional* part of the program, a part that gets treated as an isolated box with inputs and outputs in the machine of the modular redescription. For example, as we have already seen in section 6.6.10, an entire subroutine of the original description can be represented by a single high-level instruction in the modular high-level redescription.

In general, with aggregation, to more entities and/or more single values of the original description comes to correspond a *single* entity and/or a single aggregate value in the redescription. It is clear then, that the operation of aggregation reduces the *number* of single elements to be taken into consideration. If the aggregated redescription is adequately valid, then its complexity in terms of number of involved parts or values is lower than the complexity of the original description. It must be noted that reduced complexity in the number of parts does not necessarily correspond to a reduced complexity of the overall *description*, understood as above as the combination of variables (that is, symbols representing each one a part) plus a machine acting on them. This is because a valid redescription could end up dealing with less variables than the original description, but it is possible that these aggregated variables must undergo, for the redescription to be valid, a computation at the redescription level which is more complex than the computation operated on the more numerous parts of the original description by the machine of that description. In this case, however, the reduction in complexity of the redescription concerns at least the number of variables.

It is arguable that, especially in redescribing non-linear systems, in the high-level aggregated redescription there is a trade-off between the complexity of the aggregate machine and the detail

[50] As explained in the introduction to DDSs in section 5.1.

in the data on which it must act: the less detailed the data, the possibly more complex the machine must be in order for the whole redescription to be valid.

For purely *explanatory* purposes, however, we can be confident that a given valid redescription can be further aggregated both for what concerns its machine's modular description, and the data the machine manipulates: it suffices to give a sensible name to each module of the machine and a sensible name to the type of data it acts on. As in the example above, these names can be "image processor" and "picture data". These names can be considered aggregate values, and a name of a functional module of the machine is a form of *specification* of the computation which has to be effected by that module. Different forms of specifications (as described in section 4.1.5.2) can give rise to different explanations with different explanatory value: for example, an explanation in which a specification of each functional module in a flow-chart is given by means of a conventional meaningless name (e.g. "FIFO"), and the meanings of these names are not better specified, ends up being not a very perspicuous explanation, while if the meanings are more significant (as in "output memory buffer"), the whole explanation acquires more intelligibility.

However, even a more perspicuous naming like the latter, would not probably suffice to render the modular redescription a *runnable simulation*: when redescribing computer programs or in general computational systems, the redescription can be required for two different purposes, not necessarily separated: simulation and explanation. Simulation means that the description must be able to *run*. For example, a modular redescription of a given program in the form of a translation of the original program in a higher-level language, is supposed to be runnable, to be able to perform a simulation. This is guaranteed by the fact that the higher-level language possesses in turn an obvious lower-level implementation, provided by its transformation by means of a compiler or an interpreter (see section 4.1.4.2), which is its lower-level redescription, and which is the description which is *actually* run. Also, the original program possesses a similar, isomorphic, lower-level implementation able to be run. Thus, the modular redescription of the original program in terms of the higher level language is able to be *run* as well, and can be used for *simulation* purposes. It could well be used also for *explanatory* purposes, for example when the higher-level language is more perspicuous to the observer than the language into which the original description is written. There are, however, other cases in which a high-level modular redescription is useless for simulation and useful only for explanatory purposes. This can happen when the language of the redescription is so abstract that it cannot be implemented without recurring to informations external to the redescription itself. For example, a modular diagram like that of fig. 6.5 is such a case. Here, and in all the cases of redescription useful only for explanatory purposes, the high-level language does not have an already known typical implementation, as a high-level computer language has: the high-level language provides too little information to convey an obvious way to implement it. This can be due, for example, to the use of meaningless labels as the names of the high-level modules, as highlighted above. But also other cases of more meaningful specifications can be difficult to be correctly implemented, like the case of fig. 6.5, where the names of the modules give a functional specification, which is however too vague to allow for an immediate implementation: the abstraction by means of which the redescription has been obtained has produced a too big loss of information. Can a simulation be run on the description represented in the image? Not directly on *this description*: we would first have to implement the description (that is, translate it, seen as a specification, into another, lower-level description which implements it) as a real computer hardware, and in order to do this, we would need further non-obvious informations, external to those provided by the depicted description.

I propose to call a modular redescription useful only for explanatory purposes, an *explanatory redescription*.

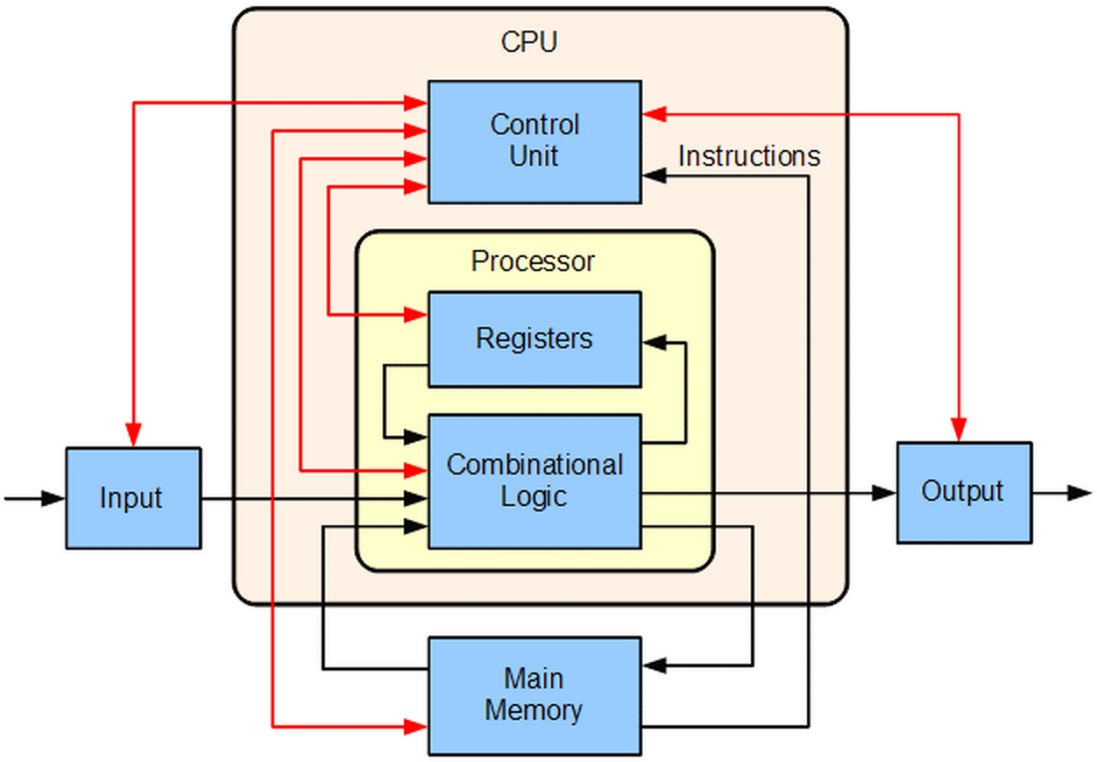

Figure 6.5: a diagram representing the modular high-level structure of a typical computer (von Neumann architecture). (Image by Lambtron, taken from https://commons.wikimedia.org/wiki/File:ABasicComputer.gif).

In general, the finding of a valid aggregate redescription, as originally intended by Herbert Simon[51], involves two phases:

i. aggregation of variables;
ii. finding a valid update function describing the aggregate variables' dynamic behavior.

Step (i) consists in finding a suitable aggregate abstraction of the entities of the original description. Step (ii) consists, mainly, in finding a dynamic update function for the aggregate system, or, in other words, a *machine* which, for each aggregate entiy, describes the dynamical behavior *of the aggregate* system. This machine acts at the level[52] of the aggregate description.

As highlighted above, finding a a valid modular redescription amounts to finding a certain valid aggregate redescription: in this case, step (i) consists in finding a modular description of a given system, by means of some algorithm for modularity detection, as for example those described in section 3 for what concerns network modularity.

But, this two-phase search is not simple: as we have seen in section 2.2.1.2, phase (i) is in general computationally intractable, while, at least for *linear* systems, point (ii) is more easy to

[51] As described in sections 2.2.1, 2.2.1.1 and 2.2.1.2.
[52] "level" understood as explained in section 6.6.6.

fulfill. And, actually, as we have seen, also in the case of optimal modularity detection, which corresponds to phase (i), there is computational intractability concerning optimization of the Q value of community modularity, as described in section 3.2.1.2.

Of course, that does not mean that modularity detection or aggregation is always unfeasible, because this intractability affects general algorithms for finding optimal modularity based only on certain features of the observed system and a chosen metric on these features, in absence of any further information or constraints about the modular structure of the system. Traditionally, in empirical science, point (i) is instead conducted not in a general algorithmic manner, but with the help of some specific information, derived from observation, which can provide some bias and some constraints which constitute hints as to how to suitably decompose the system into modules. Nevertheless, for sufficiently large systems, as we will see in section 13.3, this computational hardness in the search for modularity could hinder scientific research.

6.6.11 High-level modularity and macrodescriptions

I propose to call a *modular redescription* any *valid* redescription which consists in a modular representation of the original description (that is, a description where each element corresponds to a module of elements of the original description, and where relations between elements correspond to relations between the aforementioned modules), together with a machine acting on this modular representation in order to validly model the dynamical behavior of the original description, that is to produce a dynamical behavior of the modular representation which does not diverge more than an accepted (and chosen) error degree from the dynamical behavior of the original description, seen in its aggregate (or modularized) form.

This condition of validity holds when the higher-level redescription captures the *functional* modularity of the original description, and not only its structural modularity, or a modularity which is not functionally relevant to the dynamical features of interest for the observer.

It is important to make some observations and give some new definitions here:

1. *Every description is modular*: in general, *any* description of which we are talking in this work is *modular*. This stems from the fact that, by the definition of description employed here[53], a description is a discrete dynamical system, and discrete dynamical systems are modular, given that, as highlighted in section 5.1.1, DDS are digital systems, and digital systems are modular systems.

2. *Preferred descriptions are modular*: accordingly, even preferred descriptions, in that they are discrete dynamical systems, are modular: in a preferred description the modules are the single, atomic entities of the basic ontology of the discipline according to which the description in question is a preferred description.

3. *Trivial modularity*: given that any description is intrinsically modular, this modularity is a *trivial* modularity *for that specific description*. Inside a given discipline, the modularity of the preferred description of that discipline is thus trivial *for that discipline*: this trivial modularity represents the lowest modular description level of that discipline, the "bottom" level[54]. There is also another trivial level of modular description, always present in any description: this is the *highest level*, the one which considers the whole system (the whole

[53] See section 6.6.5.

[54] Of course this bottom level is dependent on the discipline taken into consideration, as examples made above have clarified.

description) as a single module. These two levels, the lowest and the highest, can be considered together the *trivial hierarchical levels*, which are always present in any modular system, and they together constitute a *flat hierarchy*.

4. *Modular redescriptions are higher-level*: as implied by the definitions given in the former sections, any modular redescription is situated at a *higher level* than the original one, because its ontology is obtained from the original one by a *proper* abstraction, and a proper abstraction transforms a language into a higher-level language. Thus, the modular redescription is at a higher level than that of the original description, and each entity in the ontology of the modular redescription represents a set of entities of the original system: it is a macro-entity composed of micro-entities. I thus propose to call a modular redescription a *macrodescription*, and its modularity *macromodularity*.

5. *Higher-level modularity is non-trivial*: the non-trivial thing about modularity of descriptions lies in their being susceptible of modular *re*-description: given any description, it is not guaranteed that it can be redescribed in a modular way. If this obtains, then we can say that the original description shows a form of *high-level modularity* (in addition to its already present low-level, trivial modularity). The non-trivial question is precisely to ask if some description shows any high-level modularity.
   High level modularity, as we will see at length in section 6.8, is important because it allows for certain types of *explanation* of the original description, types of explanation which are precluded in absence of high-level modularity.

6. *High-level modularity is relative*: of course, like modularity in general, high-level modularity is *relative* to a chosen *relation* between entities of the original description, or chosen properties of them. We can in general talk of the choice of a *metric* on the original description, a metric on the base of which, high-level modularity is assessed: different chosen metrics give rise to different modular high-level redescriptions. Some metrics do not even allow for any modular redescription of certain original descriptions. It could be also possible to talk of abstractions, instead of *metrics*: in general, modularity can be assessed relative to a criterion which consists in a computable function which acts as "module detector" (for an example, see section 5.2.2). We can conceive this function as an abstraction, which takes groups of entities of the preferred description and subsumes them under a single placeholder. Such a kind of function constitutes a module detector when the groups that it selects are indeed modules in the sense of the general definition of module given in section 2.1, that is, when they show a certain degree of robustness.
   Modularity is relative to the choice of a metric, but we must consider that, often, the preferred description itself suggests the choice of a metric, or a limited set of "natural" metrics according to which to proceed in order to asses modularity. For example, in spatially distributed systems, a natural metric is euclidean distance: entities placed in spatial proximity are more likely to be considered part of the same module. For the reason that natural metrics are tied to the preferred description, modularity is relative also to the choice of the preferred description. But, as we have seen, in science the preferred description is usually imposed by the discipline into which we are working, so this choice is not really free.

7. There are, at least in principle, descriptions which can *not* be subject to a modular redescription, that is, which present modularity only at their basic preferred description. Descriptions of this kind, which I would call *antimodular*, will be considered in part IV

6.6.12 Macro level and Micro level

In accordance with a natural intuition, a single element representing more parts can be thought of as "composed" (possibly in a complex, non-additive way) by those parts, and as such can be seen as a "macro" element, whose components are the "micro" parts. In the case of an *operation* on entities, a single operation composed of more simpler operations can be considered a "macro" operation[55].

Thus, I call a *macroentity* the single entity in the modular redescription whose state represents the aggregate value of the states of the entities constituting a module of the original description. The latter entities internal to a module will be called *microentities*.

I will call a *macroinstruction* a macro-operation composed of elementary operations of the original description, or some aggregate operation obtained by any whatsoever computable transformation from a group of elementary operations of the original description.

As it could be probably useful, I will extend this use of "macro/micro" terminology to all the others concepts concerning descriptions.

The modular description can then be seen as a machine operating on macroentities by means of macroinstructions. Each macroinstruction can be seen as a functional module, if seen from the standpoint of the original, low-level description. From the standpoint of the modular redescription, the corresponding macroinstruction can be called a *macromodule*.

Likewise, a modular redescription can be called a *macromodular* description, or *macrodescription* of the original description, which is then a *micromodular* description, or *microdescription*.

The whole machine operating in the macromodular description can be called its *macrodynamics*, and the machine of the original description is the *microdynamics*.

The global state of a description at a certain instant in time (we must remember that a description is a discrete dynamical system) is called its *state*, and represents the configuration of the values of all the variables of the description at that moment. The state of the macrodescription is the *macrostate*, the corresponding state of the original description is the *microstate*.

It is quite clear that this conception of *macro* and *micro* is completely *relative*: a macro-something is *macro* only in relation to more micro-somethings: a $macrodescription_1$ is "macro" only because it is the result of a modular redescription of a given description (let's say $microdescription_1$), and in turn $macrodescription_1$ could (potentially) be redescribed into another modular redescription, $macrodescription_2$, which would become a macrodescription with respect to $macrodescription_1$, while, relatively to this new $macrodescription_2$, $macrodescription_1$ could then be seen as the microdescription.

The language in which the macrodescription is expressed, the *macrolanguage*, can be, quite naturally, seen as a *high-level language* with respect to the *low-level language* of the original description, and this high-level/low-level distinction is as well completely relative. According to this relativity and to the definition given above[56] of *level of description* as the language into which a given description is expressed, these high-level and low-level languages constitute a *higher level* and a *lower level* of description, respectively.

The language in which the macromodular description is given is the *macrolanguage*, the language of the original description is the *microlanguage*.

[55] "Macro" is even a term of art of computer programming referring to a sequence of elementary instructions constituting a composite instruction.

[56] In section 6.6.6.

The *level* of the macrodescription is the *macro level*, and the level of the original description is the *microlevel*.

In accordance with the idea of hierarchical modularity exposed in the preceding chapters, we could see each hierarchical level in a *hierarchical modular description* as the *level of description* (in the technical sense of *level of description* given above) which derives from a *modular redescription* (again, in the technical sense stated above) of the immediately lower hierarchical level.

To better understand all these questions of description, validity of the modular redescriptions and levels of description, we could resort to some schematic illustration, as those in fig. 6.6, 6.7 and 6.8.

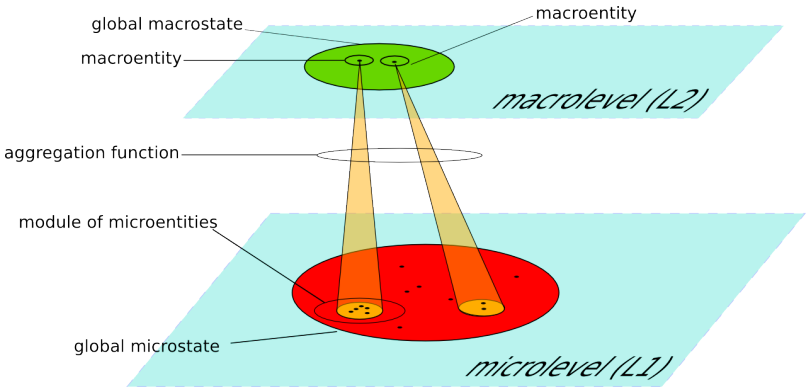

Figure 6.6: a proper abstraction, or aggregation function, which maps entities composing a module of the microdescription to single entities of the macrodescriptions in the macrolanguage (the macromodules).

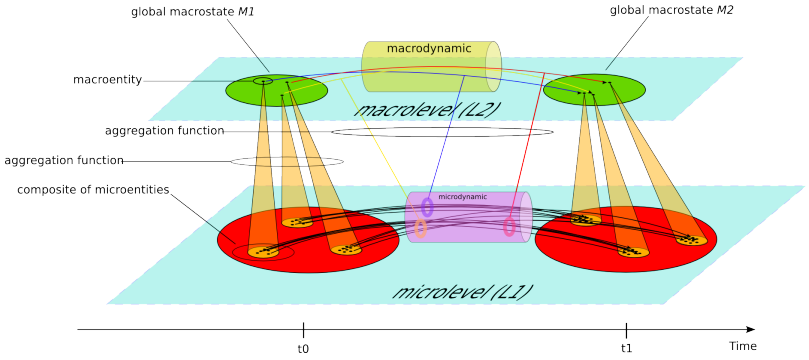

Figure 6.7: a valid macrodescription.

6.6.13 Levels and the specification/implementation relation

The macro/micro relation holds between a more abstract representation obtained by abstraction from a more detailed one, and the more detailed one. The entities and operations at the more abstract level are, in general, aggregates (aggregated in some computable way) of the entities and operations of the original, detailed description. As we have seen in section 4.1.5 and 4.1.5.1,

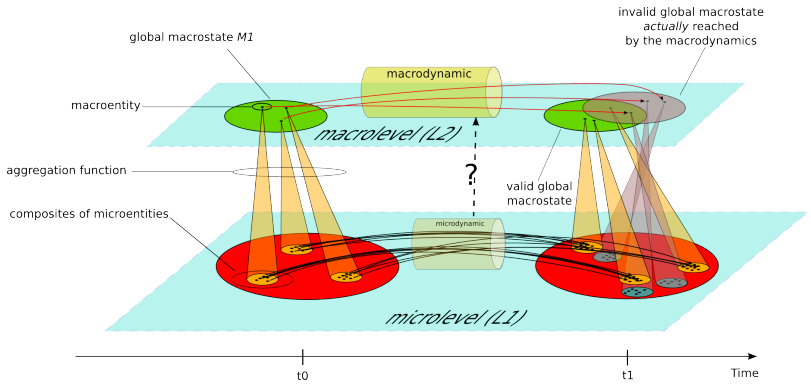

Figure 6.8: an invalid macrodescription.

a *specification* of a computation or a computer program constitutes a form of aggregate representation of the computation. In the case of computer programs, or in case a systems gets described as a computation, we could then see the higher-level description as a specification of the lower-level one. As explained in section 4.1.5, the relationship specification/implementation is relative.

It could be useful, at least in certain cases, even outside questions of computer programming, to see the relation macro/micro as a form of specification/implementation relation. This standpoint about hierarchical levels of description as standing in the specification/implementation relation, must obey a condition: that the macromodules capture functional modules of the lower level. This condition is however already fulfilled if we take for granted, as in the definition given above, that a modular redescription is a *valid* functional modular redescription. The coincidence between detected structural and functional modularity must not be taken for granted, however, because, as seen in section 6.2, there are cases in which this coincidence does not hold.

All in all, when the macrodescription is a functional valid macrodescription, we can consider the relationship between it and the microdescription as a relation specification/implementation: the microdescription implements the specification provided by the macrodescription[57].

The specification/implementation relationship which holds between hierarchical *levels of descriptions* is an *epistemic* way to view the problem of levels of description, and does not give rise to a *constitutive* hierarchy like classical hierarchies describing "layers of reality", which have a much more ontological flavor. My framework about levels allows for a more general abstraction hierarchy, of which a constitutive hierarchy is only a subcase (even the relation between a whole and a part which constitutes it can be seen as an abstraction, because the whole can be seen as the property that all of its parts have in common: they have in common the belonging to that whole). This setting could probably be a problem for some philosophical positions: for example, a strictly *ontic* position on mechanism, like Carl Craver's, could not consent, probably, to assimilate the idea of level of description proposed here to the idea of hierarchical level in a mechanistic explanation: from such a standpoint, the levels of a mechanistic explanation must describe ontic levels of organization, which do not allow for multiple realizability of the higher level by the lower level configurations: this can be permitted only during early phases of a research, in which only

[57] Without discussing this at length, I think this relation could also be applied between what in philosophy of mind is usually called the "realization" relationship between a function and its realizers: the function is a specification and its realizers are its possible implementations.

mechanism sketches or schemata[58] are provided, that is, very high-level descriptions which name some functional modules with a cursory "name", leaving all the detail of its implementation aside. But, the ideal of an ontic mechanistic explanation is, once found a description of a mechanism at a certain level, that of providing full detail of its implementation at the immediately lower level, and so on in a fully "reductionistic" way[59], until a bottom level full implementation is given: for example, in neurosciences, this would be a neuron-level implementation. My model of levels of description based on implementation and specification is not so rigid, and allows for a multiple realization of each level. This renders it more flexible and general, but also more "epistemic" and liable of "antirealism". A discussion on this could be due, but I will only touch upon this question in a moment.

I would like to also highlight that the framework I propose here is different from the conception of new mechanists, in that it is more *liberal*. Descriptions and redescriptions can be based on *any* relevant feature of a certain phenomenon. The description can fit whatever interest is moving the observer or the researcher. In many cases, different kinds of abstractions will produce descriptions which can fit some pre-existing type of explanation: some descriptions will be seen as *phenomenological* models, that is, models not of the functionality of a process, but of the mere process as a sequence, some others as *mechanstic* models, catching the actual structure generating a process, some others as completely anti-realistic purely functional models, highly mutlirealizable by other, less abstract functional models, or even by mechanistic models, which, as such, will become the most plausible realistic realizers of the abstract function. The interesting point is that many of these models can be in some cases transformed, by abstraction or realization (or better, implementation) one into the other, with the only condition that higher level redescriptions remain *valid*. This condition of validity ensures that the abstraction is not meaningless: that *some* property of the original phenomenon is caught and conserved, that is has explanatory relevance. But, holding these condition, the choice is quite free. The validity condition makes, however, this quite free choice constrained by limitations of computational nature, namely the validity condition and the *antimodular emergence* (a notion which is clarified in section 13.3) that shows up when the system is, relative to the chosen metric for modularity detection, inherently antimodular (that is, that it does not possess modularity relative to that metric), or when it is too large to allow for the finding of a valid macromodular description, for reasons of computational complexity of the modularity detection algorithms employed. These computational limitations constrain the choice of the possible metric, and in a way constrain our epistemic possibilities. In this sense, the risk of antirealism is reduced by these constraints: if the position I embrace is antirealistic, then it is a *constrained antirealism*.[60]

These constraints ensure that only valid redescriptions are taken into account: only for valid redescriptions, the realization process is like that of computational implementation, and the relationship between a description and its higher-level redescription is like that between implementation and specification. The relation is that of realization of a function: valid redescriptions produce an implementation hierarchy. Constitutive hierarchies could produce invalid higher-level redescriptions. For example, a constitutive abstraction, which partitions the system into random chunks and maps each chunk into a single high-level entity, mapping operations of the lower-level language into the same operations between these higher-level elements, would most probably *not* produce a *valid* redescription of a system, because the high-level system's elements do not correspond to *functional* subsystems of the original system. The validity condition is simply the

[58] See section 11.1.6.

[59] Reductionistic according to a certain conception of reductionism, widely shared among scientists at least, even if not among many mechanistic philosophers.

[60] Other considerations on constrained antirealism are in 1.5.1.

condition to have identified role-functional relevant modules, but the choice of this functional decomposition is constrained by antimodular emergence.

These constraints can be viewed as an epistemic bias towards cognitively and computationally treatable descriptions of the world.

6.6.14 A meta-consideration on levels of description

It seems to me that the theoretical framework proposed above could provide a useful and quite faithful redefinition of the classic concept of "level of description", a concept which is not in general, at least in its typical philosophical uses, immediately perspicuous and clear-cut. It is a concept used mostly in a loose, often metaphorical way. A redefinition like the one above could hopefully render the concept more analytically treatable, especially because its treatment can rely on an already well established exact discipline, namely computer science. Even without trying to apply to this notion of level of description as programming language all the formal results which can be attributed to formal languages, this reformulation should allow, as we will see, to translate a series of concepts related to levels of description into more familiar and easier to handle operations which pertain to computer languages and programs, a point of view which, at least in the eye of computer literates, can be considered even more "practical" and *less abstract* (as strange as it may seem) than reasoning about abstract philosophical objects. Computer programs, while they are certainly "abstract" objects, are not that abstract when interacted with under the form of their physical realization in the physical computers we use everyday: their creation, correction, and modification is certainly a process acting on symbolic representations, but the fact that computer programs can be dynamically *executed*, with the consequence that they *actively* produce feedback toward the user, renders them some sort of "concretely observable" phenomenon, and compels the programmer in her continuing process of correcting and refining a program, to resort to an immediately experimental mindset and method, "experimental" in the sense proper of scientific investigation. As will be highlighted also in section 6.9, from a philosophy of science-like standpoint, programming compels to constant "interventions" (in the sense of the "interventionist" theories of causality[61]) in order to better highlight new facets of the behavior of an object that, while being abstract, is so highly manipulable to become "concretely" present. Along these lines I hope the translation I'm proposing here between epistemological concepts and computer science concepts could be fruitful.

It appears quite clearly, though, that the highly simplified view of science that I propose here as a paradigmatic model of science is particularly fit to model certain special sciences which heavily rely on *mechanisms*, rather than theories, as they theoretical construct. The very idea of preferred description is a veritable paradigm of mechanistic description: an organized set of parts which interact giving rise to some phenomenon, a model which perfectly fits the requirement of mechanistic explanation as currently conceived in philosophy of science (a topic which is to be treated in section 10). This kind of description or explanation is surely typical of some *special* sciences, primarily biologically-derived disciplines, but also social sciences. Given that the idea of description as posed here is *epistemic*, it fits quite well also purely functional explanations, namely computational explanations, which are used primarily in cognitive sciences. Limitations of this kind must be taken into account: I do not want to propose this framework as universally applicable to science in general.

[61] For example Woodward (2003). See also a brief discussion on Woodward and his account of causation in section 6.9.

6.7 Temporal decoupling of hierarchical levels

As noted in section 2.2.4, two theorems demonstrated in Simon & Ando (1961) state that, in nearly-decomposable hierarchical systems, there is a temporal decoupling between levels in the hierarchy: the higher we go in the hierarchy, the slower the dynamics turn out to be. This holds throughout the whole stack of different hierarchical levels, and it is what allows the observer to focus selectively on different levels and predict that level's dynamics: dynamics at the selected level should be, if the hierarchical division in the model is sensible, orders of magnitude slower than the dynamics at the immediately lower level and faster than the dynamics of the immediately higher level. In a way, this captures the notion of *robustness*, which a subsytem has to be endowed with to be considered a *module*: a subsytem is to be considered a module of a hierarchical modular level if it is robust, that is if it keeps itself more or less stable in an *aggregate* way against the much faster changing interactions between the parts inside it.

The presence of decoupling between timescales at different levels is actually a *criterion* to detect hierarchical modularity, at least dynamical modularity, a notion which has been analyzed in section 6.2. Detection of a hierarchy in a dynamical modular system is *based on* timescale decoupling: at a lower level, at a finer timescale, the inter-modules dynamics is *supposed*, by approximation, as static, while the intramodular dynamics performs its evolution. The fact that the criterion for hierarchical modularity could be seen as a matter of values of magnitude in numerical coefficients in a matrix, like in near-decomposability proper, is simply an expression of the same temporal criterion: coefficients in the matrix represent *influence rate* (influence of whatever form, according to each case) between lower-level elements of the system. *Rate* is a *temporal* quantity[62]: higher rate entails a higher "speed" (or frequency, or probability of occurrence during a certain time) of exchange of some kind of influence. For this reason, elements linked by a connection with a high influence rate will interact more rapidly with each other, than with elements to which they are connected by a lower-rate connection. For this reason, different modules, that is, elements at a higher scale than the scale of the elements of which they are composed, interact at a lower rate with each other than the rate of interaction of their intra-module, lower level elements: this is at the origin of the inter-level temporal scale decoupling in a hierarchical system.

As we have seen, in a *general* conception of hierarchical modularity, aggregation must not necessarily consist in the algebraic *sum* of some *quantities*, or similar functions such as the mean, but it can consist in *any kind of computable function* whatsoever which maps a configuration of the module subparts to the module aggregate state. As showed in section 5.2.2, in the case of cellular automata a form of aggregation can in some cases be obtained by subsuming under a single placeholder all the possible microconfigurations realizing the same process module, as in the case of gliders or spaceships. In this case the aggregation function is a computable function that can be described as a "spaceship detector". It is not an arithmetic or statistical function like in the classic cases of aggregation, but the placeholder tracking the position of the spaceship obtained by means of this function can surely be considered the aggregate value of the spaceship's constituent cells. Even in this case, for example in the case of the spaceships in Conway's Game Of Life, like the one considered in figg. 5.7 and 5.8 of section 5.2.2, the higher level's dynamics is decoupled from that of the original CA, because the aggregate position of the spaceship advances of 2 pixels every 5 timesteps of the low-level original CA.

We could ask ourselves if this timescale decoupling always holds. The answer is trivially negative: it cannot hold as such in a hierarchy which is only *structural*, that is a static system without any

[62] In terms of slope of a curve which represents a function of time, or in terms of frequency, or probability of occurring of an event in a given temporal unit.

dynamical aspect: for example, a static taxonomy. However, in many systems, especially natural ones, a dynamics can be implemented *on* the static structure, and then temporal decoupling can occur.

But does timescale decoupling always occur in *dynamical* systems? It seems this decoupling is at least a very generally applicable property. Here are some examples:

- in a cellular automaton functioning in a *glider regime*, wich is a long-time regime with well-definined, modular macro structures such as "gliders" or "spaceships" allowing the system to be seen, under certain high-level descriptions, as a system performing complex computations[63], the high-speed low level dynamics, because of the multiple realizability of the gliders, gives rise to a slower dynamics at the glider level. Moreover, given that gliders are bigger than one cell, that to convey information they must usually travel distances bigger than their dimension, and that the "speed of light" is fixed[64], they will certainly take a quite long time (a good number of low-level timesteps) to exchange information one with the other in order to perform even a minimally sensible computation, while at the low level the configurations realizing the glider level are ever-changing.

- Timescale decoupling holds in modular computer programs, and in the case of modular decomposition, during reverse-engineering[65], (or also during programming[66]) of a computer program into modules, the criterion is more or less the same: we identify parts of the program that act on a well defined subset of the program's variables, without affecting other variables, and treat them as modules or procedures performing a computation according to a certain specification. During the execution of each program module, values of variables external to the module do not vary, and, similarly, a new output value gets produced by the program module only *after* its internal computation has been completed: while the internal computation is underway, the whole external program context is to be considered a *fixed* context. This is equivalent to the case of a decomposition of the system's dynamics according to near-decomposability, in which environmental parameters external to the module are considered fixed, while the module performs its internal dynamics[67].

[63] See sections 5.2.2 and 14.5.1.

[64] As explained in section 5.2.2, the "speed of light" is the maximum speed which can be reached by a "traveling" dynamical macrostructure in the CA. Each CA-rule has a speed of light, which is related to the width of the neighborhood of the rule, and is equal to the maximum radius of this neighborhood. In Elementary CAs, which have rules of radius 1, the speed of light is equal to one cell per timestep.

[65] See section 4.3.

[66] In the case of programming it is the programmer which of course plans a hierarchical structure for the program: she certainly does not *detect* it. But even the planning is based on temporal decoupling: a subroutine is supposed to perform a computation which takes only a fraction of time of the whole computation of the higher-level part of the program which calls the subroutine (in fact, often the subroutine is called repeatedly by this higher-level module). Another difference when programming with respect to reverse engineering, is that, if one tried to reverse engineer a discrete dynamical system like a cellular automaton *as* a computer program (see section 5.2.4 and 14.5.2), it seems unlikely she would encounter proper subroutine *calls*: in their place, she would find only repeated sequences of instructions: what a programmer would call "macros". This is because a non-explicitly-programmed complex system cannot optimize itself in advance by devising the best way to reduce its redundancy like a programmer would do, that is, by identifying parts of code likely to repeat, and by confining them into a subroutine, which will then get repeatedly called during the execution of the computation: of course, this kind of in-advance planning is not to be expected in a system occurring by chance or in any case never explicitly programmed by a human.

[67] In the case of explicitly programmed code which makes use of repeated calls to the same subroutine, the entire state of the machine is saved in a memory area called the *stack* just before any call for a subroutine, to be retrieved immediately after the subroutine has completed its computation giving back control to the part of the program which had called it. This way, the entire context external to the subroutine gets "frozen" in the stack during the execution of the subroutine, and can be thus be safely considered fixed.

Actually, according to many authors, temporal scale decoupling is to be considered a general rule in hierarchical modular system. Salthe (1985) mentions Simon, von Bertalannffy, Valentine and others[68] as supporting this view.

However, some dynamical systems cannot be decomposed on the basis of the timescale decoupling criterion, because they simply don't show different dynamics at clearly distinct timescales: this occurs in systems whose elements are so interconnected that no significant segregation of subsets occurs in of them: dynamical influence spreads evenly more or less at the same rate across all the system, making the systems' dynamic occur at the same pace everywhere[69]. The same non-decoupling of time scales can appear in nonlinear systems: nonlinear interaction between internal elements of a module and the extra-module environment can trigger disproportionately ample variations of the extra-module context even following quantitatively small, short-term variations of the intramodule environment, and this falsifies the assumption that at the finer timescale the external context can be considered stable. Prediction of those systems' dynamics would have to resort to the non-decomposed, non-aggregated exact model, which can be computationally very complex.

6.8 Modularity, economy of description, explanation

A hierarchical modular system allows for a form of economy in producing its description: by focusing on a single hierarchical level, or by ignoring details of levels below a certain one, the information required for a description is reduced. As Simon notes:

> If you ask a person to draw a complex object – such as a human face – he will almost always proceed in a hierarchic fashion. First he will outline the face. Then he will add or insert features: eyes, nose, mouth, ears, hair. If asked to elaborate, he will begin to develop details for each of the features – pupils, eyelids, lashes for the eyes, and so on – until he reaches the limits of his anatomical knowledge. His information about the object is arranged hierarchically in memory, like a topical outline. When information is put in outline form, it is easy to include information about the relations among the major parts and information about the internal relations of parts in each of the suboutlines. Detailed information about the relations of subparts belonging to different parts has no place in the outline and is likely to be lost. *The loss of such information and the preservation mainly of information about hierarchic order* is a salient characteristic that distinguishes the drawings of a child or someone untrained in representation from the drawing of a trained artist.[70]

This can be considered a form of *information compression*, and specifically a form of *lossy* compression: part of the information is discarded, at the cost of a certain approximation in the produced description, but with the advantage of more ease in producing, communicating, and understanding it. Or it could be seen as a form of *abstraction*: a case of partial consideration of the information present in the system[71]. Decision on what to discard is analogous to the decision

[68] Salthe (1985), p. 72.

[69] It can be objected that there must be some temporal decoupling between distant parts, because in physical systems the speed of information trasmission or of causal influence is limited at best by speed of light. But this limitation does not hold for idealized dynamical systems of for systems whose time is *discretized* into time steps or "clock cycles", as is typical in digital systems: in a boolean circuit, for example, within a timestep, a signal can reach every point of the net. In CAs there is a speed of light because they are particular DDS with a *local* rule, which limits the spread of perturbations at each timestep, and this renders them partly parallel and partly serial systems.

[70] Simon (1962), p.477. Italic is mine.

[71] See section 6.6.1.

based on the trade-off between precision in predicting the system's dynamics and economy of the system's description which was hinted to in sections 2.3 and 6.5.

Another form of information compression often allowed by modularity is a lossless compression consisting in *reduction of redundancy*: this occurs when multiple identical copies of the same module recur throughout the system: in this case it is not necessary to separately describe each of the instances of this recurrent type of module: it suffices to cite the module type and the locations of its instances (of its *tokens*) in the system's structure.

This simplification in describing a hierarchical modular system allows for easier *understanding* of the system's description, while a possible non-modular, non-hierarchical system, that is, a system which possesses a *completely flat* hierarchy, would require a description of such a complexity as to possibly overwhelm our cognitive resources. Herbert Simon highlights this property of the descriptions of non-modular systems:

> The fact then that many complex systems have a nearly decomposable, hierarchic structure is a major facilitating factor enabling us to understand, describe, and even "see" such systems and their parts. Or perhaps the proposition should be put the other way round. If there are important systems in the world that are complex without being hierarchic, they may to a considerable extent escape our observation and understanding. Analysis of their behavior would involve such detailed knowledge and calculation of the interactions of their elementary parts that it would be beyond our capacities of memory or computation.[72]

Due to its ability to induce economy of description, it can be easily argued that modularity plays an important role in explanation. Herbert Simon's first example of a nearly-decomposable system in his seminal paper[73] already contains hints to a link between modularity of a system and the explanation of its behavior: the modular formula of the system's aggregate dynamics[74], citing only *aggregate* variables, is *simpler* than the original formula taking into account all the unaggregated variables. If the aggregated formula is employed in the explanation of the system's dynamics, a significant economy of explanation can be achieved, and this would surely enhance the explanation's intelligibility. The non-aggregate formula would have cited a higher number of variables. The aggregate formula reduces this complexity by orders of magnitude, renders explanation more practicable and enhances its understandability.

In general, modularity allows for a *coarse-graining* of a system, obtained by producing a modular macrodescription of the system: to each module, composed of a group of entities of the original preferred description, corresponds a single entity in the macromodular redescription. An example is given in fig. 3.15 of section 3.2.5. This coarse-graining occurs in two phases: first, we proceed to detect modularity in the original description, a task which corresponds to finding disjoint subsets (the modules) of the original description's entities, identified according to a certain metric on the original description. We could then replace, inside *another*, new description, each one of the found modules with a *single* entity (its "name") and replace the relations between entities of different modules of the original description with relations *between the modules*, that is, between single entities of the modular redescription. Since the replacement of a module, which is a group of entities, with its "name", constitutes a transformation between languages by means of a *proper* abstraction[75], the redescription so obtained, that is, the description based on the names of the modules, is at a level which is *higher* than the level of the original description. In a dynamical

[72] Simon (1962), p. 477.
[73] Simon (1962).
[74] See section 6.5.
[75] See sections 6.6.3, 6.6.4, and 6.6.6.

system, the new high-level relations between entities of the redescription constitute the aggregate dynamics, and, as discussed in section 6.6.12, they must be chosen so as to obtain a *valid* coarse-graining of the original system. If the validity condition holds, a *macromodular* description[76] is obtained.

A coarse-grained representation is a representation containing *less* detail than the original one, and reduction in the detail of a given description is certainly correlated to a better chance for a human subject to *understand* it: given that human cognitive resources and capabilities are limited (this is an experimentally proved fact: namely, our attention span and short-term memory are quite restricted), they can be easily overwhelmed by too much information. As a consequence, there is always a trade-off between informativity of a representation and its intelligibility. A well-chosen level of description is the one which manages to convey the highest possibile information in a still intelligible way: the goal is that of maximizing these two contrasting parameters together.

It can be objected that understanding is not necessary for explanation: according to a classic model of scientific explanation, Hempel and Oppenheim's deductive-nomological model (which will be introduced in section 8), human intelligibility is not an interesting problem for the philosopher who tries to characterize what an explanation is: explanations are simply linguistic devices conforming to certain formal requirements. This is in line with the neopositivistically-inspired view of the proponents of this model, which consider intelligibility a purely psychological question, and as such devoid of philosophical interest.

Now, it is my opinion that understandability of explanations *must* be taken into consideration. It seems to me that the post-neopositivistic, "standard view" neglect of this aspect is on the wrong side. Inside an epistemic view of explanation like the one I propose here, which sees explanation as an eminently communicative and cognitive act, understandability of the explanation cannot be ignored. The same view is certainly shared by authors, like Cory D. Wright and William Bechtel, who take the same epistemic side regarding mechanistic explanations, and by Robert Cummins and supporters of functional analysis in general[77]: all these explanatory practices would not make much sense, should they produce unintelligible representations. Functional analysis, when lacking understandability, seems in general devoid of sense as an explanation, because it cannot be interpreted in an ontic way, as mechanistic explanation could[78].

Modularity seems definitely related, through its ability to simplify a description by coarse-graining, with *understanding* of a system's representation: a macromodular representation is certainly more easily graspable than the original, low-level one: a flow chart, which is a macromodular representation of a program, is more easy to understand than the corresponding, apparently unstructured list of lower-level instructions of the program (and this is mainly the reason for the very existence of flow-charts). A functional high-level representation like the one in fig.3.14[79] is certainly more understandable than the corresponding original network composed of a myriad of macromolecules. Hierarchical representations are, as we will see at once, an even better and more efficient way to convey a big quantity of information about a system.

But even setting considerations of intelligibility aside, high-level modularity is *necessary* to produce some types of explanation. Robert Cummins's analytical explanatory strategy, as we have seen in section 9, explicitly requires a *hierarchical decomposition* of the functioning of a system,

[76] See section 6.6.11.

[77] These kinds of explanation will be discussed in section III.

[78] That does not mean that functional analysis could not make still sense, even if unintelligible, as a formal redescription of computations, to be used in an automated way: for example by compilers which translate between programming languages.

[79] In section 3.2.3.

in order to explain it. This is possible only when some form of functional high-level modularity is actually present in the system, and, if present, when it is detectable[80]: the subfunctions required for functional analysis can legitimately be considered functional high-level *modules*, so a lack of macromodularity prevents functional analysis and consequently functional explanation. Similarly, the idea of a mechanistic explanation, as highlighted in section 10, requires the finding of a coincidence between what can be considered a structural and a functional hierarchical modular descriptions of the system. So, in the first place, it seems that at least explanations of a certain kind, namely mechanistic or functional explanations, per se *require* modularity, even when we ignore issues about the *intelligibility* of these explanations.

Besides economy of description at each of its higher levels taken singularly, hierarchical modularity also allows for *multilevel* explanation, which certainly enhances *comprehension*, by allowing the fine-tuning to the observer's needs of the amount of information relayed by the description: given an appropriate hierarchical decomposition, a system can be described at any desired level of description, with different results on the intelligibility of the explanation: the more abstract, coarser-grained levels allow for a very simplified explanation, which usually induces better understanding, while the choice of proceeding down to lower, more detailed levels, enhances the amount of detailed information on the system conveyed by the explanation, even if possibly at the cost of understanding: the most detailed possible explanation is the one which describes the system in terms of the bottom-level entities[81] of its preferred description, and, in many cases, the sheer amount of information contained in such a description can hinder its intelligibility. There is always this trade-off between amount of detail and understandability, but hierarchical modular representations allow also for the understanding of the relations of inclusion (or realization) *between* the hierarchical levels, and this enables the observer to mentally travel up and down across levels, in order to produce an internal multi-scale insight of the system's structure, a mental model which surely enhances intelligibility of the whole system. This fits well with the conception of mechanistic explanation[82] advocated by William Bechtel and his collaborators, who consider mechanistic explanation as not a merely reductionistic approach, but a type of explanation which requires the exhibition of the relation between multiple levels of description: a mechanism, which is the object of an investigation, can be usually seen as composed of sub-mechanisms (that can be considered its modules) and it is itself at least implicitly situated inside an overarching context, a larger mechanism of which the observed one is a module. The same authors support an epistemic, cognitive and communicative view of explanations which highlights the importance of their potential intelligibility, a potentiality which is enhanced by the multi-level hierarchical structure of the explanation, when this hierarchical description can be found.

6.9 High-level modularity conditions experimental research and computer programming

The explanatory capacity of functional analysis according to Cummins' account, which we will encounter in section 9.2, is probably nothing new, at least in computer science: any programmer, even if she does not have to do it explicitly, at least implicitly *explains* to herself how the program she is developing works by concerted execution of its subroutines. The act itself of designing a certain software requires that the programmer knows the specification of the whole program (e.g, the specification to be "a web browser"), and development proceeds by analyzing (more or less

[80] This condition of detectability of modularity will play a central role in my theoretical proposals in part IV.
[81] Bottoming out itself is usually a matter of choice or convention, anyway. See section 11.1.5.
[82] Detailed discussion of this conception is in sections 11.2.

in Cummins' sense, as we will see) this global function, which is the specification, into smaller subfunctions which together make up the implementation of the specification, each of which in turn gets decomposed, if possible, in simpler subfunctions, and so on. The subsequent practice of writing the program, that is, the lying down of the sequences of instructions composing each subroutine, usually starting from the simplest subroutines, constitutes the phase of practical implementation, which is almost impossible without a former, at least implicit, *explanation* on the part of the programmer herself, of the whole system in the above hierarchical terms. And this can be seen as a form of *functional explanation* the way Cummins understands it. So, functional hierarchical explanation is necessary for *programming* in computer science.

Likewise, in empirical science, it seems that functional, multilevel explanation is essential not only *after* a theory about a phenomenon has been devised, when explaining an already known phenomenon, but also *in the making* of a theory. A slight digression is needed about a debated topic in philosophy of science: the basic idea of *interventionistic* accounts of causation like that by James Woodward[83], is that a causal relation is present between two entities when a hypothetical variation of the state of an entity, purposefully induced by an experimenter (the *intervention*), would produce a variation in the state of the other entity. If this hypothetical circumstance holds, we can say that the two entities are related by a causal relationship. In a mechanistic account of explanation[84], the aim of the experimenter during scientific research is to advance and refine the mechanical description of a mechanism by progressively discovering all the causal relations holding between its parts. If we think of a mechanism as a network of causally related parts, it is necessary, in order to progressively discover its structure, to proceed by intervening on each part separately, and see if some consequent variation occurs on other parts. As Woodward specifies, intervention on the state of a part requires that, temporarily and at least virtually, the structure of causal links going from other parts of the mechanism towards the part on which we are intervening, be temporarily *disrupted*[85].

So, interventions on mechanisms require that the mechanism is temporarily *modified* by eliminating some of the connections between its parts. This can be represented in a system of equations which, in discrete dynamical systems[86], taken together represent the global update function.

[83] See Woodward (2003).

[84] I, rather freely, expand here the interventionistic account of causation to interventions on mechanism, by using a modified terminology with respect to the terminology used by Woodward. For a discussion of models of mechanistic explanation, see section 10.

[85] To clarify with an example: let's say that in a TV set, there is a LED which is invariably dark when the TV is turned off, and lighted up when the TV is on. Now, there is a correlation between the status of the TV screen and the status of the control light. This correlation can be due to two situations: the first is that the control light receives power from the same power generator which supplies the screen of the TV, or, in other words, that there is a common cause for the lightening up of the screen and of the LED; the second situation (false, in this case), is that it is the LED itself the power generator which supplies power to the screen, so there is a casual connection between the LED and the screen. Of course in this second case, by intervening on the status of the LED, we would demonstrate, à la Woodward, that this causal relation between LED and screen holds. But, let's suppose that the actual situation is the first, that is, that both the LED and the screen have a common power supply: how should we proceed in order to *intervene* on the led? If we intervened on it by turning it on and off with the usual act of turning on and off the whole TV set by acting on the remote control, we would have that, only apparently we are making an intervention on the LED, because we are not taking into account that the LED is still connected to the rest of the system as usual. This is not a correct form of intervention, because it would make appear as if were indeed the LED to cause the screen going on and off. The correct way to intervene would be, first, to detach the LED's cables from their usual connections (an action which constitutes a modification of the causal structure of the system), and then to power the LED separately with, let's say, a battery, in order to see if this still produces a variation in the TV screen. With this correct intervention, we would rightly conclude that it is not the LED to cause illumination of the screen. The correct intervention required a modification of the causal structure of the system.

[86] As said, I mainly concentrate on this class of systems. Woodward does not specifically target such a class, but I think his arguments can be applied without modification to it.

Woodard writes:

> More generally, a system of equations will be modular if it is possible to disrupt or replace (the relationships represented by) any one of the equations in the system by means of an intervention on (the magnitude corresponding on) the dependent variable in that equation, without disrupting any of the other equations[87].

Woodward claims that the set of equations representing correctly (i would say *validly*) a causal system, *must* be modular, because, given that detection of causal relationships requires intervention on a part of the system, and intervention requires temporarily disrupting only the causal influence which bears on that certain part, were the system completely not modular, this precise disconnection of one causal path effected during the intervention would disrupt not only the part of the equation interested by the intervention, but also other parts of the system. With this claim about modularity, he basically means that modularity in the equations must at least concern the single variables, and this in turn reflects the fact that modularity should be present in the system at least at the lowest level, that of the preferred description. This is compatible with the fact that usually (if we want to avoid metaphysical paradoxes of causation like those highlighted by Jaegwon Kim on dowward causation[88]) we should construe as causal the *lowest* level. Of course, modularity is usually always present at the lowest levels, because a preferred description, if it is representing a system as a mechanism, should supply a set of distinct, elementary parts each of which, as such, is a module per se. But, if we want to redescribe a system mechanistically at higher-levels, we could certainly construe relations between high-level parts as prima facie "high level" causal relations[89]. In that case, in order to proceed by intervention, Woodward-style, modularity is needed also in the equations representing the system's dynamics at these *higher* levels. All considered, this condition holds if the update function's structure is *hierarchically* modular, and this in turn represents the fact that the system is functionally, and probably also, dynamically, and, quite likely, structurally, *hierarchically modular*.

During the phase of discovery, already devised partial mechanistic explanations can suggest and guide further observation: the researcher needs first to produce roughly sketched explanations, comprising missing parts or very vaguely-defined modules, in order to guide the research. She is supposed then to try to progressively refine these models, by getting suggestion precisely from their incomplete structure in order to guide research towards the areas in which further experimentation is needed. This will put to test predictions based on these incomplete models, would conduct to their revision, and to further cycles of model refining. All these phases require that macromodularity is present and detectable in the system in order to produce all the needed intermediate explanations.

A similar situation can arise during the construction of a network model, especially for biologically interesting networks. For example, determination of network structure of the gene regulatory networks is often effected by the method of gene knockout. Due to the high genetic redundancy in eukaryotes, knockout of multiple genes is required, but the high number of more than two genes combinations renders this endeavor unfeasible, as noted in Gulbahce & Lehmann (2008). Other techniques to assess gene-to-gene interactions can require much experimental effort, and seldom

[87] Woodward (2003)., p. 48.

[88] See, for instance Kim (1989a) and Kim (1989b).

[89] To be clear: i don't think high-level causation is plausible. But if in describing a mechanism we bottom out at a certain level which is not the lowest possible level, we would be certainly satisfied with the idea of searching for causality at that level, *as if it were* causality proper. For that matter, as can probably appear from considerations I make in section 9.2 and 1.5.1, I am not a fan of the idea of *metaphysical* low-level causation either.

complete descriptions of gene regulatory networks can be obtained by this purely bottom-up approach. A top-down approach could consist in identifying structural modularity in the already known portion of the network, which is usually quite coincident with functional modularity in these kinds of networks[90], and use this information to infer the role of newly discovered genes on the basis of the already known functional role of the module to which the newly discovered gene belongs. Of course, to make use of this methodology, a modular structure must be present in the network.

A method devised in Clauset et al. (2008) could turn out being very helpful during the research phase, because, after having detected the hierarchical community structure of a partially discovered network, it can, on the basis of that hierarchy, produce hypothesis on where to look experimentally for missing links not yet evaluated between two nodes. The method is quite complicated, but it can be synthesized by saying that it has turned out that many topological and statistical properties of networks are identical in networks which possess hierarchical community structures belonging to the same set of possible hierarchies. In other words, there are classes of hierarchies which are substantially equivalent for what concerns many important characterizing parameters and properties of the networks which are so structured. During the phase of network discovery, the already discovered partial network can be assessed for hierarchical modularity, and the typology of this hierarchy is identified. On this basis, the algorithm can predict which links will be discovered and where, by looking at which couples of nodes which do not result as connected in the partial network are likely to be connected on average in the hierarchies of the same class. The researcher can then focus experimentation precisely on the couples of nodes which should be linked according to this prediction, in order to empirically assess the actual linkage between the two nodes. This way, the rest of the network discovery can be sped up. But, of course, this method presupposes that a hierarchical modularity is already present in the network.

The two cases above highlight the importance of modularity even in the discovery phase of a network of scientific interest.

Thus, it seems that hierarchical modularity, at least according to a certain account of causation, and of mechanistic explanation, is central not only in the explanation of already mechanistically modeled phenomena, but also, and maybe foremost, in the phase of *discovery* of the mechanistic models.

The same necessity for modularity can be expected to arise when analyzing systems not with the aim to detect causal mechanisms, but *functional* relations, as is the case in the reverse-engineering of computational systems, or in cognitive science research: even if in this case the preferred description is not required to correspond to a physical reality (moreover, in pure cognitive functionalistic psychology à la Fodor, which rejects the type identity theory, the basic natural kinds of psychology *must* be functional, and not physical kinds), the counterfactuals which hold between consecutive computational states or mental representations, which are functional states, must, if the system is to be considered deterministic (and it usually is supposed to be such), hold with the same cogency with which counterfactuals supporting causal claims hold[91]. To

[90] See section 6.4.

[91] Some clarification is probably due: Jerry Fodor, of course, talks of causal relationships between mental representations. If some form of physicalism is adopted, however, this opens the door to the well known objection of the causal exclusion by Jaegwon Kim (see, for instance Kim 1989a and Kim 1989b). I would like to avoid this problem by granting causal powers only to the "hardware" level in computational systems, and not to higher-level descriptions. At those higher levels, holds a similar necessity which I would hesitate to call "causality", because this term in my opinion brings with it a too heavy metaphysical burden. Besides, when we consider the idealized computational systems treated by computer science, it does not seem that any sensible idea of causality can be applied to them. It seems that Robert Cummins was first in expressing a not too dissimilar view, which sees

all intents and purposes, reverse engineering of computational systems and experimental cognitive psychology act as a kind of interventionistic experimental research on mental faculties or on computational systems. So, the situation is analogous, and high-level modularity is as well essential to the discovery of these counterfactual dependencies by interventionistic methods, in a way corresponding to the case of mechanistic discovery. It seems then that high-level hierarchical modularity is essential for scientific research in many fields of the special sciences.

6.10 Summary

In this section we have reflected on the notion of modularity, viewed as manifesting itself in three main possible modalities: structural, dynamical and functional. We have concluded that structural, dynamical and functional modularity are quite often correlated. We analyzed several forms of functional modularity in dynamical systems, especially networks, and reflected on the importance of functional modularity in giving informative and at the same time synthetic representation of complex networks, showing this by way of examples. After having declared the support for an epistemic view of explanation, and introduced a theoretical framework aimed to facilitate the discussion on the transformation of descriptions of phenomena of interest into modular hierarchical descriptions, we highlighted several important features of hierarchical modularity in general: first, hierarchical modularity manifests and is based on a decoupling in the timescales at different hierarchical levels which allows for simplification of the theoretical description of a system, a degree of simplification which increases with the degree of abstractness of the modular level of description. Second, hierarchical modular descriptions are essential to certain forms of scientific explanations like functional analysis and mechanistic explanation, and certainly the simplification they permit is essential in giving the produced explanations a high level of intelligibility. Third, it appears that, not only when giving a posteriori scientific explanation based on an already established theoretical model, but also during the *course* of the construction of this model, that is, during the active phase of the experimental research, at least when this research aims at producing a mechanistic or a functional model of a system, high-level and possibly hierarchical modularity turns out being an essential condition for the experimentation itself and for the model which will thereby be constructed.

It seems, in a way, that high-level, hierarchical modularity plays a fundamental role in scientific explanation, and probably in *shaping* science itself, and so that the circumstance of encountering modular systems and descriptions in many scientific areas is to be expected.

---

programs not as pertaining to causality, in that they are abstract objects. I would like to consider regularities in these idealized models as explainable in terms of reasons of *conventional*, or *normative*, or "grammatical" nature: it is a condition taken for granted by computer scientists that a Turing machine must obey the constraints of its idealized physical structure and the instructions of its machine table. This constraint has the same cogency implied in *accepting* a grammatical rule: if someone does not want to obey a certain grammar, she is allowed to, but she would end up simply *not* speaking *that* language. No violation of causality would have occurred. We could obviously conceive, without violating any causality, a Turing machine which does not obey its transition table: it will simply *not be* a Turing machine. It is, in other words, a *normative* question the *definition* of a mechanism, let's say its *specification*: the condition that circumscribes a mechanism and defines what the mechanism is supposed to do. A mechanism obeys its specification *by convention*, simply because if it did not obey the specification, we would not consider it the same mechanism, precisely because *being that mechanism* is something which is *defined* in a certain conventional way. This is not a causal question, in my opinion, at least, and for sure, at *that* level of description: the level of description of the computational mechanism, or of the "mental". This discussion, though, borders dangerously on the infamous "kripkensteinian" paradox of rule-following, and this is not the right place to continue it. I have raised however some related problems in section 14.5.1.

Chapter 7

Some issues about modularity in biology

In this section, I will try to give an overview of the significance of the notion of modularity in today's biological thinking.

I highlight first a main problematic area regarding modularity and biology: does evolution produce modularity, and how? I will try to shed light on the main developments in these investigations occurred in recent times. I will only marginally touch, for reasons of space, many related convoluted problems, like the questions raised by the so-called evo-devo approach, which bear on problems of modularity in the ontogenetic developmental process and its relationship with phylogeny.

In a second part of the chapter, I highlight the importance of modularity in current biological research by reporting various examples, drawn from the biological literature (especially systems biology), of the recent tendency of interpreting biological systems in a primarily modular fashion, and of different ways to produce modular explanations of biological phenomena.

7.1 Evolution and modularity

A natural question to pose about modularity in biological systems is the question on the relationship between evolution and modularity: does evolution produce modular architectures and dynamics in organisms? And, in case of an affirmative answer, has modularity evolved by natural selection, or for other reasons? There is an obvious observational scientific route to follow in order to answer such kind of questions, but there also some a priori considerations seemingly able to shed light on the topic.

A typical line of a priori reasoning in supporting the evolution of modularity is the idea that modularity improves *evolvability*, that is, intuitively, the more or less pronounced capacity of an organism to evolve in the course of phylogenesis. In intellectually pre-evolutionary times, the *complement* of this idea was often taken for granted: the idea that organisms constitute an integrated whole, and that for this reason a change in one part would have resulted in the breakdown of this complex organization. This was, for example, the position held by Georges Cuvier at the turning of XIX century[1]. Given these premises, it stands to reason that the emergence of evolutionary thinking is to be naturally associated with the weakening of this idea of complete structural integration of organisms and the emergence of the idea of modularity as less strict integration between parts of a whole.

[1] See for example Schlosser & Wagner (2004), ch. 1.

7.1.1 Evolution of modularity in Herbert Simon's view

Since Herbert Simon's seminal papers on hierarchical modular organization, it has been argued on a priori grounds that biological evolution should preferentially produce modular architectures, both at the phenotypic and at the genotypic level. This conclusion is typically based on the consideration that evolution would have needed a much longer time to produce the same level of complexity shown in today's organisms, were it to work by cumulative, non-modular assembling of parts into a complex.

To expound this argument, Simon (1962) introduces a kind of parable, the tale of the two watchmakers, Hora and Tempus, which build watches composed of about 1000 parts. While Tempus assembles each of his watches one piece at a time as a progressively growing single complex, Hora makes use of modules: she first assembles subsystems of about ten elements, which turn out as being stable units, then proceeds to connect such subsystems together to form larger complexes, and then to connect these complexes together into a super-complex which constitutes the whole watch. The problem that must be faced is this: if the building process gets disturbed, the already connected pieces of a still non-completed complex tend inevitably to detach one from the other. So, should some event interrupt Tempus' assembling, the aggregate reached so far would fall completely to pieces. On the contrary, an interruption would destroy only the last of Hora's modules, the one still under construction. This way, Hora takes on average about 4000 times less time than Tempus to assemble a watch, according to an estimate by Simon which assumes that, on average, the watchmakers get interrupted every 100 pieces mounted. This advantage is thus obtained by ensuring at each time the presence of stable, robust subassemblies.

Simon's parable has a fairly transparent analogy in biological evolution: should evolution work by aggregating parts into progressively larger complexes without relying on potential intermediate stable forms, the time required for the biological evolution of the complex forms we observe today would have taken way longer than the time this process has actually required.

What is less clear is if this is an explanation of the emergence of modularity during evolution as due to *natural selection*. Simon mentions natural selection, but is not very clear on that: he reckons that the most "stable" biological forms, that is the modular ones, the biological structures analogous to Hora's watches, are the "fittest"[2], but he does not make clear the connection between stability and fitness. The problem is, natural selection acts on each generation based temporally on local properties of the organisms alive in that moment in time. But the stability Simon talks of is a cross-generational, phylogenetic stability, not a developmental one, so it is not clear how this meta-property could be the direct object of natural selection.

We could probably find an answer by looking at the problem from two slightly different point of view.

First, let's consider only the *phenotypic* structure of an organism. It is to be expected that in a scarcely modular phenotype, which, by definition of modularity, shows a high degree of physical and functional connectedness between most of its parts, any damage affecting a single part, or functional mechanism of the organism, could spread its destructive influence indiscriminately to other, often distant parts or functions, making the organism less resilient or robust to external perturbations. This lack of robustness bears directly on the organism's fitness: a non robust organism is less likely to survive or reproduce than a more robust one, because it is easily damaged by external perturbations. So, natural selection would certainly favor robustness of the phenotype, and robustness is particularly present in *modular* phenotypes. So, natural selection

[2] Simon (1962), p. 20.

should directly favor *modularity* of the phenotype. Thus, in a way, we can already claim that darwinian evolution should favor modularity, at least at the phenotypic level.

The second point of view is this: It can be argued that phenotype modularity is a consequence of the modularity of the genotype-phenotype map[3]. And modularity of this mapping brings with it greater evolvability: in a scarcely modular organism, there must be genes with a high degree of pleiotropy. Pleiotropy is the circumstance that a single gene influences many different phenotypic traits, and thus diffuse pleiotropy is precisely a sign of the non decomposability of the phenotype-genotype map, that is, of the absence of its modularity. Diffuse pleiotropy means however that it is *frequent* that large groups of phenotypic traits co-vary even after a limited change in a single gene, because this gene has *likely* a high degree of pleiotropy. In such organisms, limited genetic changes could then very frequently bring about diffuse, probably even harmful, widespread effects on the phenotype. In other words, in scarcely modular organisms, changes in the phenotype occur in clusters of changes, not in single, limited changes, independently of the amount of genetic change which determined them. This renders the organism less *evolvable*, because in these cases variations of the genome produce a smaller range of possible phenotypic variant than in the case of more modular organisms, where small genetic changes produce small phenotypic changes, and where for this reason the set of possible phenotypes is more assorted. Moreover, in the phenotypic space of non-modular organisms, phenotype variants are probably more different one with respect to the other than in the case of modular organisms: in non-modular organisms, pleiotropy produces less fine-grained variations on the phenotypes. Acting on a less varied and more coarse-grained phenotypic space, natural selection has less effectiveness in optimizing the fitness of the evolutionary lineage, and less freedom in gradually shaping the form of the phenotype, because the assortment of possible phenotypes is limited. It would, from a long-term evolutionary point of view, be better for natural selection thus to act on lineages of organism endowed with modularity, because the "power" of selection in shaping evolution would be more effective on them than on lineages composed of organisms which are non modular[4]. Of course, natural selection can not *directly* select for the *class* of lineages. But, as we have seen above, natural selection can directly selects for modularity in single organisms. This way, on long, evolutionary times, modular organisms should probably tend to be more abundant than non modular ones. So, on coarse evolutionary timescales, lineages of modular organism should tend to be more prevalent than lineages of non modular ones. This way, by acting on single organisms as a unit of selection, natural selection meta-selects lineages of modular organisms. We could then say that there is also a meta-selection for modular architectures in biology in general. In accordance with Simon's argument, natural selection would probably require much more *time* to produce complex stable organisms if it had to act on lineages of non-modular ones. Should natural selection act on systems where a single modification affecting a part brings about not only an advantage, but also, at the same time, a lot of potentially disruptive consequences in other parts of the system, because a component's influence spreads without any restriction to all the other components (the analog of Tempus' case), natural selection would employ much more time than when acting on systems in which localized changes do *not* heavily affect other parts of the system: this second case is the case of *modular* systems, in which, by definition[5], components of a subsystem can at most strongly affect the subsystem to which they belong, but, due to the susbsytem's partial isolation from the other ones, this influence can not inflict heavy damage to the other subsystems. In this case, each subsystem is *stable*, or *robust*, against

[3] The genotype-phenotype map is the function which correlates the genes with their phenotypic effects. See section 7.1.3.

[4] It goes without saying that the application here of intentional and apparently teleological terms to natural selection is only a metaphorical use of the terms.

[5] See section 2.1.

external influences: each subsystem is thus a *module*, and this corresponds to Hora's case. Any mutation in the genome can thus affect a limited scope of features of the organism, on which natural selection will act, *leaving intact the previous evolutionary achievements* outside that scope. This way, natural selection can gradually, *cumulatively*, build upon previously selected adaptive features, and this reduces the needed evolutionary time by orders of magnitude. Contrary to what a first reading of Simon's argument seems to imply, this reduced time is, however, an *effect* of natural selection producing the meta-selection of modularity, not a *cause* of the selection for modular organisms.

While Simon's original exposition was dubious, I think that, reformulated as above, his argument could be seen as an argument for the *Darwinian* evolution of modularity, that is, for its evolution as caused by natural selection.

Regarding evolutionary meta-properties, it seems that modularity in general affects evolvability: as Hartwell et al. (1999) notes, should the function of a certain protein affect every process in a cell, natural selection would have a hard time in selecting for favorable mutations of the genes coding for that protein, since any slight modification of the protein would probably affect positively some process, but negatively many others. This has actually been observed: there is empirical evidence that proteins which participate in many different cellular processes have undergone a very slow evolution, if at all. But if a protein's functionality is restricted only to some specific sub-process, that would not be the case. In general, *isolation* of sub-processes and subsystems inside a larger biological system favors the evolvability of the genes coding for the subsystem or subprocess. Isolation of a subsystem is attained by limiting its input and output connectivity from and towards the other subsystems. Such a subsystem can therefore be considered a *module*. In cells, isolation of subsystems, that is of complex of macromolecules performing some specific genetic or metabolic function, can be realized by spatial isolation, or by exploiting the chemical specificity between proteins or DNA and their ligands: certain molecules can act as information transmitters only from and toward specific targets, this way limiting the spread of information between modules. In a modular organism, random mutation during evolution will often affect only certain specific phenotypic traits, without disrupting or in any case heavily modifying other vital functions. This way, natural selection would be able to perform its (alleged[6]) "fine-tuning", by being able to "distinguish" singular, partially isolated phenotypic traits, and, consequently, to positively or negatively "select" the bearers of these traits according to the fitness advantage these traits confer them.

7.1.2 Modularity as emergent self-organization in complex systems and the role of natural selection

Albeit indirectly, as we have seen, Herbert Simon based his argument for modularity on the classic neo-darwinistic ground of evolution seen as shaped mostly if not exclusively by natural selection. He was writing in the '60s, a period dominated by the mainstream view of the New Synthesis. By the early '90s, the cultural climate surrounding evolution had significantly changed: natural selection had been put under threat as the main driving force of phylogenetic change by a series of competing hypothesis. First, there was genetic drift, that is, genetic change in populations due to purely random non-selected fluctuations of genic pools, a proposal originally stemming from Fisher's work inside the New Synthesis' development in the first half of XXth century, but

[6] I'm absolutely not denying that natural selection has a strong influence on evolution. But I don't want to enter here the debate on its relative importance against other factors (environmental or constituted by structural or path-dependent constraints), a debate which is still quite open.

strongly restated by Motoo Kimura's *neutral theory of molecular evolution*[7] around the end of the '60s. Other alternative factors influencing evolution had been proposed, namely *constraints* to the range of the possible variations of organisms, ranging from path-dependent, physical, developmental, or in general structural inescapable constraints. All these hypothesis based on constraints constitute a strand of anti adaptationism which stems first from the famous "spandrels" argument by S.J. Gould and R. Lewontin[8] and later, in the late '90s, from the nascent field of evolutionary developmental biology (so-called *evo-devo*)(See Goodman & Coughlin, 2000.), which sees phylogenesis as the evolution of developmental (ontogenetic) processes.

In this wide historical context, Stuart Kauffman, working in the emerging field of complex systems, put forth a statistical argument aimed to affirm that the appearance of modularity in the genome of complex organisms during evolution is due mostly to the emergent self-organizing properties of a class of complex systems[9], on which natural selection can successively act. In the words of Kauffman: "The essential idea is simple. It is to think of selection as acting on systems that spontaneously exhibit some particular form of order that is *typical of an entire class of similar systems*"[10]. And, "*In sufficiently complex systems, selection cannot avoid the order exhibited by most members of the ensemble.* Therefore, such order is present not because of selection but *despite* it"[11]. What Kauffman calls "order" means here the dynamical properties of the members of the class of systems which are "on the border of chaos", that is, whose dynamics, while being not chaotic but quite ordered, is nevertheless sufficiently fluid to allow those systems to perform some form of computation. An example of a class of systems of this kind is class IV of cellular automata, explained in section 5.2.1[12]. Such a kind of dynamics derives from the fact that, in these classes of complex systems, the degree of connectedness between the elements of the systems is neither too low (a fact that would lead to a system composed of isolated subsystems, which scarcely communicate each other), nor too high, with each element connected on average to all the others (a condition which would bring to indiscriminate diffusion of perturbations across the whole systems, that is: to chaos). Such intermediate degree of connection between elements of the system can be seen as a form of modularity, in which the system is composed of subsystems connected by a limited number of channels, with these subsystems showing internally a higher density of connections. Kauffman assimilates the genome of an organism to a complex network of interacting elements giving rise to the organism's ontogenetic development[13]. This network usually turns out, being on average sparsely connected and its regulative gene-to-gene functions having on average certain properties, to belong to the class of systems bordering on chaos, which are modular and capable of computations. Thus, the whole argument proposed by Kauffman leads to the same conclusion of Simon's one: namely, that evolution has made modular systems emerge. But the difference is that, while Simon's argument relies on the leading role of natural selection in producing modularity, Kauffman's view is that ordered, computational, modular systems constitute precisely the *sole* kind of systems on which natural selection can act, because they are the only ones endowed with sufficient evolvability. Natural selection can then act on systems of that class to choose a *specific* structure, maintaining a selective role. But the order typical of the class of systems on which selection acts comes from prior[14] general structural and

[7] Kimura (1968).
[8] Gould & Lewontin (1979).
[9] For examples of sef-organization, see section 5.2.3.
[10] p. 16.
[11] *ibid.*.
[12] Kauffman took his examples from a more general class of discrete dynamical systems, boolean networks, of which CAs are a subclass
[13] More on this in the next section.
[14] Maybe not only *prior*, but *a priori*? I only pose the question here, because a discussion on the nature of structural constraints in complex systems would require deep involvement in discussions about serious metaphysical

dynamical principles of complex systems which, themselves, do not involve selection. The argument, which Kauffman had been elaborating on the basis of his former work, dating back to the late '60s, is presented in a comprehensive form in a book which was to become very influential, *The Origins of Order: Self Organization and Selection in Evolution*[15]. Kauffman shows, by way of mathematical considerations and by simulation, that, under reasonable assumptions about statistical properties of random directed networks which come to represent genotypes[16], when acting of genotypes with a sufficiently high number of genes, natural selection is not completely free to select whichever genetic trait it deems fit. Complete freedom would subsist only in case the genome were completely devoid of epistatic relations between genes (as well as, presumably, devoid of pleiotropic effects): this would be the case in which no gene is regulated by any other, and each gene gives a completely independent contribute to the organism's overall fitness. In this situation, the fitness landscape[17] of the organism would be very smooth and continuous, with a single peak. In such a situation, mutations affecting a genotype in any point of the landscape would move the genotype toward the peak or away form it. In the first case, these mutations would be certainly selected, and, as mutation goes on, the evolving genotype would assuredly climb to the maximum fitness. In other words, there is a smooth path connecting any point of the adaptive landscape, that is, any given genotype, to the maximum fitness peak. It can be easily seen that, given these conditions, natural selection would have at its disposal the smoothness of control (the *gradualism*) required for it to be able to shape the phenotype toward an increase of its fitness, as is classically required by Darwinism and by the Modern Synthesis as well: genotypes with such an independence between genes (a form of complete *decomposability*, in Simon's terminology) would thus exhibit the highest degree of *evolvability* . In another, opposite ideal case, that in which every genes in the genotype regulates every other, the corresponding fitness landscape would end up being extremely rugged, with a myriad of quite low local optima, due to the fact that almost any mutation, affecting an entire cascade of gene regulations, would likely result in some benefit for some phenotypic aspect, but at the same time will have some negative effect on some other. In this case, natural selection would almost be unable to control the positioning of the evolving genotype across the landscape, because most evolving genotypes would climb to a very close local maximum and get trapped there: natural selection would thus result almost completely irrelevant. The visible features of the reproducing genotypes would not, in this case, be a product of natural selection, but of the intrinsic structural features of hyperconnected random networks. In other words, such hyperconnected genotypes show a severe lack of evolvability. Kauffman's thesis is that, given that evolution must allow natural selection to act upon biological systems, among possible biological systems a class of sufficiently evolvable ones must have been selected. The optimum for evolvability would be the class of systems with

puzzles, bordering on philosophy of mathematics: a discussion which I will not start here.

[15] Kauffman (1993).

[16] Quite naturally, in such models a gene correspond to a node, and the regulatory action of a gene upon another corresponds to a directed link.

[17] The *fitness lanscape* idea, first introduced by Sewall Wright in the 1930s, is a model in which the space of all possible genotypes is seen as a physical space, in which to each point (to each genotype) is associated a scalar value, its *fitness*. Fitness is a complex concepts, but can be very roughly conceived as a measure of the potential reproductive success of the organism carrying that genotype. If we visualize the space of genotypes as a two-dimensional surface, and the fitness associated to each point to a height along an axis perpendicular to the surface, the whole model assumes an aspect similar to that of a geographical landscape, where peaks are point of maximum fitness, with hills and valleys at progressively lower levels of fitness. Starting from one point of the landscape, that is from one genotype, any motion away form it constitutes the transition to a genotype which differs from the one taken at reference. The genotypes immediately surrounding the reference genotypes differ minimally from it, and the closer ones can be seen as the genotypes differing from the one taken as reference by only a single point mutation. Given that a genotype is composed of many genes, the space actually used in the landscape model must be multi-dimensional, with a dimension for each gene. This renders it completely unsuitable for visualization, but such a space is however treatable by algorithmic means.

a genotype made up of completely independent genes. But such a trivially simple kind of genotypes would not be able to guide any meaningfully complex ontological development, leading to very poor organisms in terms of functions. On the other hand, completely interconnected genetic regulatory networks, besides lacking evolvability, are prone to chaotic dynamics, which would end up being unapt to guide a coherent developmental process. Kauffman thinks then that organisms with an *intermediate* degree of connectivity in their genetic regulatory networks must be the systems upon which natural selection has acted. The genotypes of these systems are *on the verge of chaos*: they are connected enough to yield a rich dynamics, but are at the same time able to maintain some order in it. This behavior constitutes a form of dynamical modularity, and, according to Kauffman and many others, they enable their dynamics to realize a form of *computation*. These genotypes present high Evolvability, because, due to their limited connectedness and consequent structural modularity, mutations in one module do not indiscriminately influence, with a possible disruptive effect, all the other parts of the genetic regulative network, and useful mutations in one module can accumulate with other, formerly selected useful mutations in other modules, allowing for a cumulative evolution. Natural selection has thus been able to act on this class of networks *thanks to* their evolvability, but the structural and dynamical properties typical of all members of the class, among which is their intermediate degree of modularity, which are the properties which *enable* evolvability, are not themselves specifically *selected* features, sifted by natural selection: they are general features of this class of moderately-connected random networks,[18] features which are due to *intrinsic* topologico-mathematical[19] structural and dynamical properties of those networks. Thus, the properties of adaptive landscapes mentioned above, which are those which allow evolvability, derive from intrinsic properties of the genetic systems they represent: natural selection has been *constrained* by these preexistent structural properties to act on the most plausible (plausible from a biological standpoint) class of genotypes, the class of computationally-enabled modular ones. The task of natural selection has then been that of selecting *among* elements of such a class of highly evolvable systems: natural selection has certainly chosen[20] which features of genotypes to allow based on their fitness, but it has not chosen the features common to all the members of the class of computationally capable evolvable genotypes.

It is true that the choice *of the class* itself can be seen as a form of selection, because boolean networks outside this class show statistical features which are biologically implausible, such as completely static or completely chaotic behavior. Thus, no organism could be equipped with such a kind of networks, and, for biology as we know it, the only acceptable class of complex systems is that of computationally capable genetic networks. But, once that class of networks is selected, then its typical features, such as, among others, moderate modularity and capacity to transmit information in a controlled manner between nodes, features which are common on average to *all* networks in the class, are *already* present for structural reasons intrinsic to the *class*, and could not even be avoided by natural selection without *exiting* the class of computational networks: such features, which constitute a form of *order*, endure in the course of evolution, not thanks to, but *despite* natural selection. This basic idea by Kauffman, the idea of some kind of order not due to natural selection, an order which is ahistorical and universal among the members of a class of systems, is not against the letter of Darwin, as Kauffman notes[21], but it certainly

[18] There is, according to Kauffman's findings, a "threshold" of connectivity around which a sufficiently fluid dynamical modularity emerges in random networks.

[19] As we will see in section 12, there is a recent philosophical treatment of the scientific explanations which are based on topological properties.

[20] I remind as a disclaimer that I am of course attributing intentional terms to natural selection in a metaphorical way.

[21] Kauffman (1993), p. 487.

contradicts the adaptationistic tenet, which sees any major feature of living systems as *shaped* by natural selection.

Kauffman's argument, as said, is a statistical one, based on a sample of simulated dynamics of random networks: it is not completely decisive, given the absolutely enormous space of possible complex networks corresponding to genomes. Nevertheless, it is arguable that, *in general*, evolution favors computationally-capable moderately modular organisms, even though this could not hold true for certain classes of cases. Kauffman claims that work of the previous 20 years in the analysis of regulative genetic networks in many organisms has shown that almost in all cases that such networks possess the property of being moderately connected and that the regulative functions between their nodes possess a specific property of boolean functions, which Kauffman calls *canalization*[22]. It is expected for such features to give rise to a dynamic network behavior showing modularity and supporting a form of computation. Since the number of canalizing functions is only a small subset of all possible boolean functions, and the number of canalizing functions decreases in an extremely rapid manner as the number of input variables increases, it is *not likely* that in a randomly constructed boolean network canalizing functions abound. Thus, their prevalence in biological networks must be due to some other factor. It is conceivable, according to Kauffman that this is an effect of natural selection. But another plausible hypothesis not involving adaptation is based on the fact that canalizing functions are easier to *construct* by means of molecular interactions. Kauffman seems to be biased toward this later explanation, and this meshes quite well with his general tendency to downplay the role of natural selection, and to attribute the general properties of biological networks to intrinsic properties of complex systems and not to natural selection.

A second, although less statistically supported argument by Kauffman[23], assigns to natural selection an actual role in shaping organisms. The argument aims to show that, in boolean networks of the above mentioned class, characterized by moderate connectivity and by mostly canalizing gene-to-gene regulatory functions, a form of higher-level, hierarchical, computational modularity can arise: in such networks, some "frozen" subnetworks can emerge in the network dynamics. A "frozen" subnetworks is a part of the network whose *state* (the pattern of active or inactive nodes), for reasons dependent on its intrinsic structure, stabilizes and does not change anymore after a fixed time, independently of the state of the rest of the surrounding network. Frozen subparts of the network are thus impenetrable to dynamic perturbations, and so come to constitute a kind of variously shaped boundaries which end up partially partitioning the whole network into more or less separated "unfrozen islands", which can continue to change of state. It is in these islands that information processing comes to take place, allowing for the constitution of a hierarchical chain of information exchange. Natural selection has, in Kauffman's view, the power to shape the structure which underlies this hierarchical computational dynamics (which is a dynamics of gene activation in a genetic regulatory network), by selecting specific gene mutations. It is *this* is the actual role of natural selection, in Kauffman's theory: to shape what

[22] Simply put, a boolean function is *canalizing* if at least one of its arguments possesses a value which "fixes" the function value to a certain boolean value, regardless of the values of the other arguments. Thus, for example, the *OR* function of two varibles p and q is canalizing, because if a is set to True, then the OR output value is set to True as well, regardless of the value of b; in this case, the same holds when fixing the value of b. In other words, canalizing functions allow a single variable to fully control the output. This conservation of control on the part of some input reduces the "chaotic" effect of input variables interactions, which can show up in the non-canalizing functions, and favors the appearance of a kind of dynamical modularity, as described in section 6.2.

[23] Kauffman (1993), ch. 12. With "less supported" I mean here that the argument's generalizability is not certain, due to the small numbers of sample simulations which it relies on, with respect to the class of networks considered. But, on the side of theory confirmation, summing up the results of several research papers in the field, Kauffman concludes, albeit with caution, that, in genetic regulatory networks of eukaryotic cells his hypothesis of a "frozen core" of genes, which doesn't change its activation state, is supported by empirical data.

is freely variable in the genotype dynamics, in the context of fixed structural and dynamical features which themselves have not been selected, but derive from "mathematical" properties of complex networks (and, in part, by physico-chemical constraints): selection of different dynamics of the unfrozen part leads, in the case of genetic regulative networks, to different developmental paths in the organism's ontogeny, and thus to different phenotypes[24]. The fact that selection acts on dynamically fluid parts of a system which is endowed with other impenetrable frozen parts prevents the indiscriminate spreading of consequences deriving from a modification in a fluid part to all or almost all the other fluid parts of the system, thereby obtaining an advantage which is evidently similar to that obtained through the modular construction method of Herbert Simon's Hora watchmaker[25]: in these genetic networks, the unfrozen, fluid islands, isolated or moderately connected with other unfrozen island, constitute *modules*, because their internal connectivity is higher than the connectivity between islands. Obviously, natural selection can also shape the frozen part, by modifying the genes which constitute it, but selection cannot directly *prevent* dynamically frozen subnetworks *from appearing at all* in the genetic network: these parts appear by *self-organization* and so the feature of having frozen parts *intrinsically* pertains to the entire *class* of boolean networks with moderate connectivity and mostly canalizing functions, and selection must act on members of this class, for they are the only systems sufficiently flexible to be evolvable, thanks to their intrinsic network features. To quote Kauffman:

> Complex systems, contrary to our naive beliefs, exhibit self-organized behavior. Insofar as selection tunes the ensemble explored but is unable to avoid its generic properties, those quasi-universal features may be expected to shine through across the eons and across phyla.[26]

In my view[27], the fact that each unfrozen island is isolated, or that it remains connected to other islands via a limited number of links, allows us to view such islands as functionally isolated *modules*, similar to computational subroutines[28] which exchange the inputs, or the results of their processing, with other subroutines. The fact that in a boolean network this processing happens mostly in parallel does not, it seems to me, invalidate the analogy of network modularity with modularity in serial, algorithmic, computation. First, it is of course always possible to find two completely computational equivalent systems, one parallel and the other serial. Besides this guaranteed equivalence, it is a fact that only in their *idealized* form boolean networks are completely parallel: only in an idealized boolean network all nodes get updated in a completely *synchronous* manner. As we will see, in more realistic discrete models (and, of course, much more so in *real* genetic networks), signal propagation delays must be taken into account. This way, a boolean network can be seen as endowed with an aspect of seriality in information exchange inside and between modules, which renders the network more similar to an algorithmic processor, and allows to base partly on the temporal scale decoupling[29] between hierarchical levels the detection of its modular hierarchy.

While I think that Herbert Simon's argument could be interpreted, as I showed in section 7.1.1, as an argument not completely different in nature from Kaufmann's one (actually, in a way, in Simon's view, natural selection makes emerge as the only viably adaptable systems the modular ones, although this flexibility of modular systems does not depend in itself on selection, but it is a

[24] This kind of process modularity is exemplified by modularity in certain cellular automata, as explained in section 5.2.2.
[25] As we will see in what follows, this line of reasoning is used by other arguments.
[26] Kauffman (1993), p. 535.
[27] See section 6.
[28] See section 4.2.
[29] As highlighted in section6.7.

structural independent feature, like in Kauffman's view), and kauffman's indeed acknowledges[30] that his own argument too can be redescribed in a kind of selectionistic darwinian terms (as the claim of natural selection *meta*-selecting *classes* of genotypes, before[31] acting of specific genotypes of the selected class), it is not my aim here to go further along a discussion on the equivalence between the two arguments. I only want to note that, in different ways, both Simon and Kauffman corroborate the hypothesis that evolution is actually *likely* to bring about modularity in biological systems.

Other non-darwinian explanations of the emergence of modularity during evolution have been advanced, and most of them are based on topologico-mathematical structural porperties of certain classes of complex systems in general. It has been suggested in various works, such as Ravasz et al. (2002), Solé & Fernández (2003) and Barabási & Oltvai (2004) that modularity could arise during evolution not because of direct natural selection, but as a by-product of *duplication* and *diversification* of genes in the genome during phylogeny. In particular, Ravasz and colleagues suggest that gene duplication can intrinsically give rise to the kind of scale-free, autosimilar, hierarchical structure[32] which they believe is the typical structure of complex biological networks: duplication acts in a way similar to their method of construction of such scale-free modular networks, and could therefore be a source of modularity in biological networks. Given that there are well known natural mechanisms which produce duplication an diversification of genes, this seems a case in which a general topological explanation of the kind mentioned above is seen as implemented by a specifically *biological* mechanism.

7.1.3 Evolution and modularity of the genotype-phenotype map

Less concerned than Stuart Kauffman with the intrinsically emergent properties of complex systems, many different research groups have also engaged the problem of modularity and evolution. Some serious questions about modularity and evolution have in recent years been raised in the emerging field of evolutionary developmental biology, also known as *evo-devo*. This line of study basically aims to reach a synthesis between evolutionary and developmental biology, two disciplines which historically have evolved independently, with a lack of intercommunication: on one hand evolutionary biology, after the "new synthesis", has come to take as its pillar population genetics, a discipline which statistically studies the spreading of alleles in populations, concentrating thus on the genetic *pool* level and its phylogenesis. On the other hand, developmental biology sees the single genotype as the specification of a program, thus as a mechanical, dynamical system for the gradual construction of the phenotype during ontogeny, and concentrates on this mechanisms of development which brings from the genetic instructions to the phenotype. Around the 1990s, times seemed ripe for the creation of a synthesis between the two disciplines, because it had gradually become clear that the question of how the genotype specifies the phenotype could not be ignored anymore by evolutionary biology: following many studies which showed that natural selection is constrained by features of the developmental process, which is the unavoidable intermediate process between the genotype and the phenotype in metazoan, evolution had increasingly come to be seen as the evolution *of* ontogenetic developmental programs. *Evo-devo* was supposed to bring about this synthesis by studying the relationship between ontogenetic developmental processes and phylogenetic evolution: the study of how evolutionary change of phenotypes is produced by means of changes in ontogenetic processes during phylogeny, and how *intrinsic* properties of these processes *constrain* evolvability. Evo-devo stresses also that de-

[30] Kauffman (1993), ch. 13.

[31] "before" in a metaphorical sense: there is actually only a single, as it were, "act of selection": the differential survival rate of each genotypic variant at each generation.

[32] See section 3.2.1.5.

velopmental processes are non-linear, and that for this reason simple changes in these processes can bring about unexpected phenotypic new traits, which are the material on which natural selection can act to produce evolutionary novelty.

Evo-Devo is now a very active field, which, since its inception, has raised some diatribes about the constraints imposed upon evolvability by the developmental processes, diatribes involving especially the more orthodox advocates of the New Synthesis, for whom evo-devo does not respect two fundamental tenets of the traditional neodarwinistic position: first, evo-devo sees evolvability as constrained by *intrinsic*, non necessarily selected properties of developmental processes, while neodarwinism sees evolution as constrained exclusively by natural selection. Second, the non-linearity of developmental processes negates another point of neodarwinism, which sees the forms of organisms as susceptible only of *gradual* inter-generational variations. The strong reason behind these controversies is that evolutionary constraints deriving from ontogeny could put in doubt the adaptationistic tenet that sees natural selection as able to *freely*, *gradually* mold phenotypes in an *optimal* way, by choosing the fittest of them among a nearly continuous range of phenotypical varieties, being in this way a force able to actively shape organisms during the course of evolution. This optimality of adaptation due to natural selection could turn out being impossible in case the developmental process were not linear, a circumstance in which small genetic change could produce a disproportionate big change in the phenotype, because reaching optimal adaptation requires the gradual, monotonic evolutionary "climbing" of fitness towards the maximum peak, and this gradual climbing is possible only in a fitness landscape which is "smooth", that is, in which variants are many and vary with graduality, and fitness of similar variants is similar as well. But this smoothness of the variation of fitness among variants can be disrupted if the genotype-phenotype mapping is not linear, for in these cases a small genetic difference can produce a disproportionate phenotypic difference, with a correlated disproportionate variation of fitness. Another way the shaping power of natural selection can be hindered is when some specific phenotypic variants, due to certain intrinsic properties of the developmental process, simply cannot be produced: for example, all adult centipedes sport, according to the species, a variable number of leg-bearing segments, a number which is in any case an *odd* number[33]: it seems quite likely that this property is not a *selected* trait, but that it is due simply to the *intrinsic inability* of the developmental process in centipedes to produce an even number of leg pairs. This shows that, at least in some cases, phenotypic variability is not continuous, and that natural selection can not shape phenotypes in a perfectly gradual and free fashion. In other words, the possible range of variation that a trait can show, and on which selection can act, is not comprehensive of every possible variation in the phenotypic space, but is constrained by *developmental* factors.

This tension between neodarwinism and developmental biology is the object of a still open debate, although there are probably reason to downplay it: for a discussion, see Minelli (2009), which argues that evo-devo is a feasible and fruitful synthesis, and that it is not even the case that historically the modern synthesis has ignored developmental processes, given that certain works of J.B.S. Haldane and Julian Huxley show great interest for the relationship between evolution and development.

The questions raised in evo-devo about modularity are not easy questions: genetic regulative networks are the engines of cellular functioning and development. It is by means of their integrated dynamics that ontogenetic development of an organism is brought about. It stands to reason that, consequently, modular structure and modular dynamics of genetic networks could produce a form of modularity in the organism's developmental process and, ultimately, in its

[33] See Minelli (2000).

phenotype. If modularity actually holds across the genetic and phenotypic levels, in other words, if the genotype-phenotype map is modular, this would increase evolvability, because a change in a genetic module would not come to affect, with possible negative consequences on fitness, all parts of the phenotype, but only some more or less delimited portions of it, enabling natural selection to discriminate the specific contribution which the modified phenotypic parts bring to the organism's fitness, and consequently to select positively or negatively only the genetic module or modules which generate those phenotypic modules.

7.1.4 Modularity as due to natural selection

A prolific group of researchers working on modularity in evo-devo is the originally Austrian group centered around Günter P. Wagner, now at Yale. Wagner (1996) downplays the possibility that modularity is an effect of possible self-organization principles, explicitly focusing on trying to explain the emergence of modularity in organisms as due to *natural selection*.

Working inside the Evo-Devo tradition, Wagner sees the evolution of phenotypic modularity as deriving from the evolution of modularity of the genotype-phenotype mapping function.

In his view, evolution of a phenotypic module is a complex of traits, serving a functional role, whose member traits are tightly integrated due to pleiotropic effects of the group of genes which determines the phenotypic module, with relatively few pleiotropic effects of the same genetic group towards traits external to the phenotipic module in question.

According to Wagner, modularity can evolve by means of two basic modalities. One is *parcellation*, which takes a group of genes with diffuse pleiotropic effects toward a set of traits, and prunes some of the pleiotropic influences in order to obtain a partition of genes which produces a partition of the phenotipic traits. The other modality is *integration*, by which some unrelated genes, each of which produces independently a trait, become more integrated by the appearance of new criss-crossed pleiotropic effects between the genes and the traits, thus grouping the corresponding traits into a module.

Wagner tries to identify the possible selective forces that can produce such effects, and comes up with a series of candidates, taken from the existing literature: mainly *selection for adaptation rate* and a combination of *stabilizing* and directional selection.

The argument proposing modularity as evolved due to selection for *adaptation rate* is based on the consideration that selection for modularity enhances the speed of evolution of the modular class of organism selected. This argument is affine to the original Simon's argument, and, apparently, not completely incompatible with Kauffman's one on meta-selection of a type of system which is evolvable. However, according to Wagner, this cause of the evolution of modularity is plausible only in presence of asexual reproduction, otherwise it would require a too strong amount of linkage disequilibrium among genes.

Based on the result of simulations and on theoretical considerations, the paper concludes instead that modularity has probably evolved by natural selection acting on phenotypic traits, and specifically by a *combination of stabilizing and directional selection*, which tends to progressively eliminate pleiotropic effects, producing a progressively more modular genotype-phenotype maps in evolutionary histories in which selection must act in different ways on two or more traits originally associated by pleiotropic effects: that is, when variation of one trait in a certain direction is adaptive, while at the same time stability of the other trait is required, election tends to favor a progressive decoupling between the two traits, by negatively selecting pleiotropic effects of genes on these traits.

In Wagner's theoretical model, modularity arises when in the course of evolution, in different temporal periods different selection pressure has been exerted on different traits:

> characters that tend to be under simultaneous directional selection get integrated into a module of phenotypic change, while the characters that rarely adapt to environmental changes at the same time will be represented by genes that have no or only limited pleiotropic effects among them(Wagner, 1996, p. 42.)

This position seems to endorse a view of modularity where a form of modularity of the environment "induces" the emergence of modularity in the organism by natural selection. This view is supported by a fascinating work by Kashtan & Alon (2005), who, simulating artificial evolution of neural networks and digital circuits, found that modularity arises in the structure of these evolved networks, only when the environmental pressures varied in a modular fashion, alternating between several goals organized in a hierarhical way, while when subject only to a fixed goal, these systems evolved into a nonmodular structure.

A work by another researcher operating in Evo-Devo, Schlosser (2002), notes how, once some degree of modularity in organisms has appeared in the course of evolution, from that moment on modularity tends to self-sustain and self-enhance during the subsequent course of evolution, because it confers a high probability of producing sufficiently fit variants during phylogenesis when changing the high-level modular structure of the organism by altering *inter*-module connections: this way of making an organism evolve by "tinkering" at a high-level with it, is a possibility offered only by modularity, and which confers an evolutionary advantage, by raising evolvability: it is much more "easier" for evolution to find good variants by way of this high-level modular restructuring than by modifying connections at lower levels (for example intra-module connections), because the "coarse grained" space of possible genetic modular structures is by orders of magnitude less vast than the fine-grained space of genotypes of the classic fitness landscape.

It is of particular interest, as highlighted by Schlosser, the fact that genetic module *duplication* is a useful mechanisms for producing high-level modular restructuring. The important point is that it is expected that genetic module duplication in many cases should not produce negative effects, while at the same time it allows subsequent evolutionary differentiation of the new copy of the module without disrupting the function which it originally fulfilled, because this function is still performed by the original copy of the module. As seen in section 7.1.2, Ravasz & Barabási (2003) noted that duplication and diversification of genes can be seen in itself as a mechanism for the emergence of modularity by self-organization, without the intervention of natural selection, leading to the appearance of scale-free modular networks[34]. The above observations by Schlosser (2002), however, admit the importance of module duplication in a darwinian context, in which the main role is that of natural selection. The two views can be seen as partially compatible: according to Solé & Valverde (2008), it is more likely that natural selection can have the *secondary* role of *pruning* existing links in networks in order to favor the emergence of modular organizations, while basic modularity itself is a spontaneous byproduct of nodes and links duplication.

7.2 A modular functional view of biological systems

Evolutionary considerations aside, presence of modularity can be *prima facie* observed in most biological systems, at least : whole organisms can be considered as composed of organs, and the

[34] For scale-free networks, see section 3.2.1.5.

organism's vital processes can be viewed as brought about by the coordinated activity of functional subsystems and their dynamics. Organs can be seen as composed of cells, cell machinery as composed of functional molecular complexes and the cell's dynamical functioning as composed of modular processes.

The proposal of viewing biological processes and their networks of interactions as dynamically modular, that is, composed of *functional* modules[35] in a way which closely resembles how engineers see electrical circuits, seems potentially fruitful. This idea, already suggested by Cummins (1975), has started spreading during the '90s in genetics and other fields of biology, as the discovery of complex genetic and metabolic networks, composed of hundreds or thousands of interacting components, started to become gradually more feasible.

In this analogy, the highly specific electrical connections between components of an electronic circuit are substituted by the connection between the emitter of a proteic signal and the target site to which the protein binds, with the specificity of the connection guaranteed by the specificity of the structure of the protein and of its ligand.

McAdams & Shapiro (1995), a widely cited paper by pioneers of this approach, proposes to assimilate, where possible, genetic regulatory networks to boolean digital circuits, which, as such, can be described in terms of boolean gates (the elementary modules) and their connections. Models of digital circuits, unlike purely logical boolean networks, where propagation of data between parts of the network is idealized as instantaneous, are *asynchronous*, in that they take into account the different delays in the propagation of signals between their components. This can add plausibility to their being used to model biological interaction networks, which are asynchrounous as well[36]. Asynchronicity can even have a purposeful, functional role in these circuits, and thus it is better if their theoretical models take it into account. Subcircuits of these digital circuits can be viewed as higher-level modules performing higher-level functions, as in the case of feedback circuits, which carry out regulative functions.

The basic idea at work here, which stems quite naturally from what I have already said about modularity in general and modularity in networks, is that modules at any level of abstraction can be seen as *standard components*, which in networks correspond to *network motifs*[37]: the complete digital circuit can be viewed as a hierarchical structure, each level of which is describable as a circuit of interconnected standard parts. In the case of genetic networks, for example, the modules can comprise regulons, operons and single genes. Components at several levels can carry out several kinds of functions, be them regulatory, purely logical, or a memory function, that is the ability to store persistently or for a certain time a logical value. Modules of these kinds are sufficient to realize virtually any type of digital circuit, included computationally capable systems.

Along these lines, Hartwell et al. (1999) strongly advocates the modular view, in order to improve scientific research and explanation in cell biology. The paper tries to argue for modularity in cells under a spectrum of different viewpoints. As we have already seen in the former section, this work argues that evolution constrains the "design" principles which organisms must obey, and that these principles are those of modular organization.

[35] See section 9.

[36] For example, the transcription of a protein and its diffusion to the target sites requires a certain, variable time. This must be considered when modeling such systems. The same holds for electrical circuits, in which propagation of current, albeit very fast, is not istantaneous, and the switching time of logical components from one state to another can be relatively high.

[37] See section 3.2.2.

Hartwell and his collaborators specifically stress that modules are functional[38] units, and, seeing functions as something typical of both biological and artificial systems, (as opposed to simple physical non-biological natural systems), they advocate the description of organisms in terms of their modular functional organization, following a method that is typical of engineers when they describe artificial devices. These authors claim that the modular view allows us to describe the very high level properties of cells as patterns of connections between functional (lower-level) modules. A functional high-level vocabulary (in terms of functions like amplification, error correction, coincidence detection, for example) is needed to allow for a description of biological functions and, in this terms, describe higher functions composed of these.

Albeit conceptually distinct, *functional* and *structural* modularity, as discussed in section 6.2 are quite often related. If *functional* modularity derives from a modular *structure* present in the network, then, methods for detection of community *structure*, like the methods described in section 3, could be put to use for detecting the *functional* modularity in the network as well. That structural and functional modularity overlap to a good extent in biological genetic, proteic, metabolic networks, is confirmed by several works.

Specifically referring to Hartwell's hypothesis of functional modularity of biological systems, Zhou & Lipowsky (2004) and Zhou & Lipowsky (2006) (the latter unpublished), apply Zhou's well-known high accuracy agglomerative algorithm for community detection[39] to the protein-protein interaction network of yeast extracted from the DIP database of experimentally determined protein interactions[40]. They thus determine that the yeast protein interaction network, which comprises 2406 nodes (proteins) and 6117 links (protein-protein interactions) possesses an actual, quite robust hierarchical structural modularity. The specific hierarchical level with highest Q modularity value, 0.616, which is thus supposed to represent the most plausible subdivision into communities of the actual network, is then chosen. It is found to be composed of 449 communities or *elementary modules* with a number of nodes of the order of 10. The important result is that each of these communities is found to be coincident with a subnetwork performing an actual biological function, or a group of biologically related ones. This corroborates the view that there is an actual overlap between functional and structural modularity in the proteic network of yeast. Moreover, as further support, former works which had found more limited functional networks in the yeast protein network[41], appear to have often found part of the same modules discovered by Zhou and Lipowsky's method. These latter modules are also found to be quite *robust* against network perturbations and cohesive, that is, endowed with high *integrity* and *affinity*[42], and they succeed in passing various tests, applied to them in order to exclude that the modules are artifacts. This further corroborates the hypothesis that the found modular description reflects an actual, significant modularity in the proteic network. Groups of modules among the elementary ones come to constitute higher-level communities, which in turn have functional significance, such as what the authors call *the fundamental protein production (translation) "factory"*, a core complex of processes related to transport between cellular nucleus and cytoplasm, control of RNA transcription and splicing, protein synthesis and degradation, ribosome production. For other functions, such as mRNA modification, cytoplasmic protein transport, DNA processing,

[38] The paper seems to endorse a not better specified notion of function as something which has a *purpose*, without trying to philosophically problematize it. I guess this position on functions could be assimilated to the "etiological" conception of biological functions, which is treated in section 9.1.

[39] Zhou's algorithm is described in section 3.2.1.3.

[40] The database is located at http://dip.doe-mbi.ucla.edu/dip/Main.cgi. The page claims that it catalogs experimentally determined protein-protein interaction from a variety of sources, in order to obtain a consistent set of interactions.

[41]

[42] See description of these features in section 3.2.1.3.

often the complex of required elementary modules cuts across levels, comprising modules from different levels, while certain essential functions are performed by single elementary modules, like ATP synthesis. Thus, the global structure of the network can be described as comprising a big core module, the "protein factory", and peripheral parts implementing other functions.

7.3 A computational view of biological processes

The aforementioned work by Hartwell and his co-workers[43] does not simply claim that biological processes have a functional modularity, but puts forward, along the lines of a former, more general hypothesis by physicist J.J. Hopfield[44], a stronger proposal: the idea of viewing cellular processes as *computational* processes.

Hopfield's general hypothesis Hopfield (1994) is that the history of evolution can be seen as a way to select organisms that can make environmental measurements, and, on this basis, make predictions and generate appropriate behavior. Systems of this kind, in the opinion of Hopfield, have the essential aspects of *computational* systems, that is, systems able to manipulate *symbols*, where the input symbols are the environmental measurements, and the output ones are the signals modulating the behavior, be they signals driving muscles or turning genes on, or having another effect. The *computation* is what stands between the input and the output: the computation is the process which generates, starting from the environmental inputs, the symbolic outputs which are appropriate for the survival of the organism.

It is my opinion that the view of biological processes as computation can be a plausible position, which is in line with a highly widespread view of most natural processes as computational (from philosophy of mind to *pancomputationalism*, a position in philosophy of physics), although I think this computational view has to be better argued for, rather than taken for granted. I discuss the conditions for being allowed of talking properly of computation and the fruitfulness of computational explanation in section 14.5.1.

From the former considerations it appears that, in general, modular representation can be seen as -tied to a specific *language*, which has as terms of its vocabulary the names of the basic modules. Modules' names are, in a way, names of functions to be used in a high-level functional description of the system: this is what happens in electronic engineering, where a system can be described at high-level as composed of transistors, resistors, capacitors, and other components connected together. The system is of course susceptible to be described at an even higher level, in terms of amplifiers, voltage comparators, or, in the case of digital circuits, typically in terms of logic gates, and so on for higher levels. For some example of how engineers describe circuits, see fig. 7.1, fig. 7.2 and fig. 7.3.

An example of a genetic regulatory network represented with a schematism analog to that of electronic circuits is given in McAdams & Shapiro (1995), and reproduced in fig. 7.4.

Following this route, Lazebnik (2002) strongly stands up in favor of the adoption in biology of a formal language which, on par with those already utilized by engineers, could allow a precise modular characterization of biological processes.

Speaking of formal languages, one could be brought by association to think of formal logic, but also of programming languages. This association would not be completely inappropriate here, for we could also view (and as we have seen this view is advocated by some authors) digital

[43] Hartwell et al. (1999).
[44] Which is also co-author of Hartwell et al. (1999).

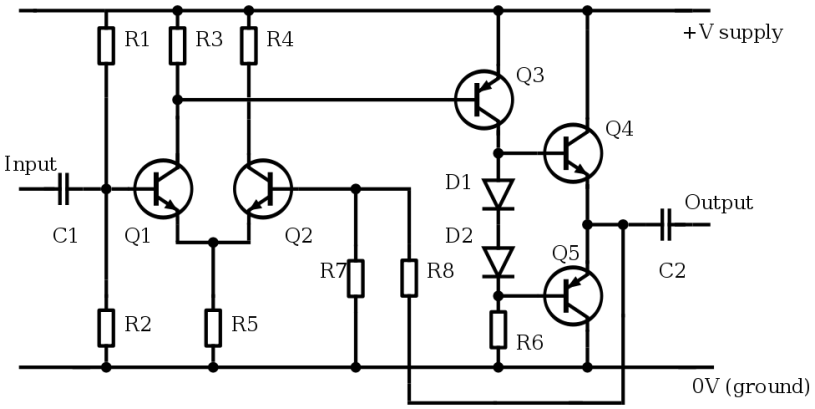

Figure 7.1: the schematic of an analog electronic circuit. While the whole circuit can be seen as a module performing the function of an *amplifier*, which outputs an amplified signal faithfully following the dynamics of its input signal, the circuit appears clearly composed of connected sub-modules, that is electronic components: transistors ($Q1$, etc.), diodes ($D1$ and $D2$), resistors ($R1$, etc), capacitor ($C1$ and $C2$). (Image taken from Wikipedia Commons, at http://commons.wikimedia.org/wiki/File:Amplifier_Circuit_Small.svg).

circuits as implementations of computations. Specifically, given that each module is supposed to perform a certain function, this input-output function represents its *program specification*[45]. It is likely that, in the actual system, this computation is implemented by a digital (digital to a first approximation) circuit which operates with a high degree of parallelism, by means of a network of interconnected logic gates. This would not hinder the possibility to model it as a computer program, which, as such, is usually more likely to have a serial structure[46]: in the worst case, the computer program would simply take more time than the parallel circuit to perform the same function, but it is certain that any parallel digital circuit can be considered equivalent to a class of computer programs. So, we could envision a digital circuit *as* a computer program. Modularity in the circuit would correspond to modularity in the program[47]: each high-level module in the circuit could be considered as a *subroutine* performing the specified function. This view is explicitly endorsed in many works, at least starting from McAdams & Shapiro (1995), which proposes a three-point method to model genetic regulatory networks:

> Consideration of electrical circuit simulations suggests a hybrid approach to genetic circuit modeling that integrates the following ideas with kinetic models: (i) identify the circuit connectivity and model point-to-point signal paths, (ii) simplify transcription control logic by treating it as Boolean logic when justified, and (iii) model the functionality of complex or nonlinear control elements in specialized subroutines.[48]

Specifically, the third point advocates the translation of the function of specific circuit modules into subroutines, that is, into modules of computer programs. It can be noted that, having

[45] See section 4.1.5

[46] This of course must not necessarily be the case: since the '80s, highly parallel universal computers have been devised, for example the Connection Machine, described in Hillis (2015). And, a lower degree of parallelization is present in most machines today, which are usually multiprocessor.

[47] See section 4.2.

[48] p. 654.

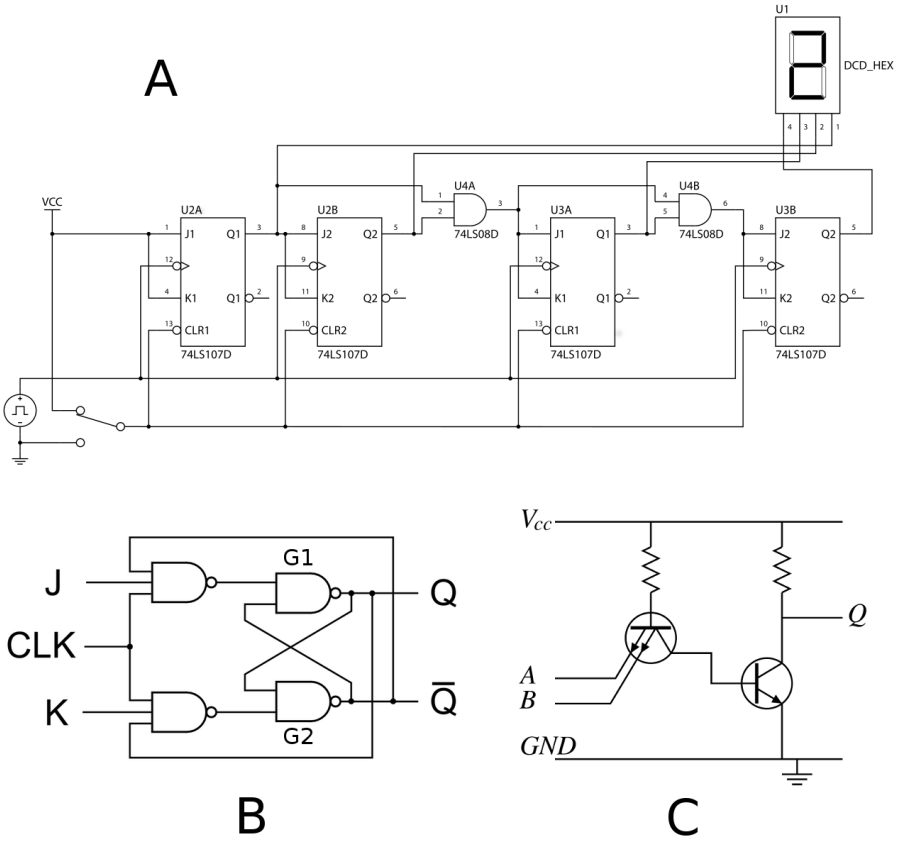

Figure 7.2: image A: a high-level diagram representing a digital circuit. Except for a few single logic gates ($U4A$ and $U4B$), most components are higher-level ones, and can be considered modules performing higher-level functions. In this case, each of the components labeled $U2A$, $U2B$, $U3A$, $U3B$ is a so-called *J-K flip-flop*, which is a type of 1-bit memory cell. Each flip-flop can be seen as internally composed of a certain number of simpler elements, namely (image B) NAND logic gates. Each of the two-input NAND gates labelled as $G1$ and $G2$ in image B are internally structured as a circuit composed of transistors and resistors, as in image C. Of course, a description at level higher than that of flip-flops is plausible: for example, the whole circuit of image A can be defined as a module performing the function of a single digit counter, which counts the impulses sent to its input line and displays the counted number in the display labeled DCD_HEX. As a module, this circuit can be employed as a standard part in other, larger circuits. (Images A, B and C taken from Wikipedia Commons, respectively at http://commons.wikimedia.org/wiki/File:4_bit_counter.svg, http://commons.wikimedia.org/wiki/File:JK-FlipFlop_(4-NAND).PNG and http://commons.wikimedia.org/wiki/File:TTL_npn_nand.svg).

been the genetic circuit already modeled by a *digital* circuit (point *ii* in the quote above), the transition to a computer *simulation* of the function is completely lossless, from the point of view of the precision of the simulation, which perfectly tracks the function implemented by the digital circuit. This leaves open the possibility that the effective implementation realized in the program differs from that realized in the digital circuit: there is an unbounded class of different, albeit completely equivalent, possible algorithmic implementations of the same digital function. This fact poses some questions about the plausibility of this kind of computational explanation, a discussion which I defer to section 14.5.1.

McAdams & Shapiro (1995) actually propose a specific example of one of these subroutines: a

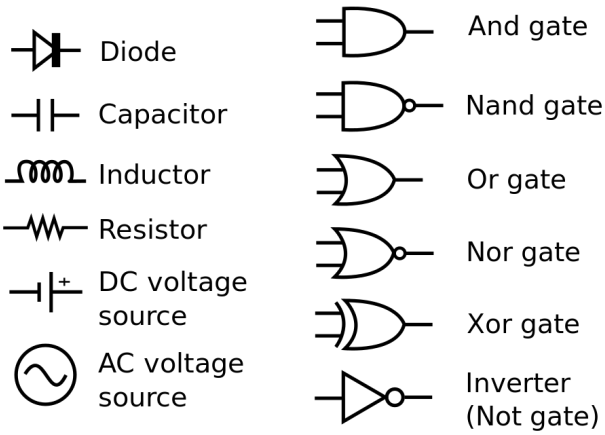

Figure 7.3: a possible dictionary of standard parts for describing a digital circuit at a not very high level. (Image taken from Wikipedia Commons, at http://commons.wikimedia.org/wiki/File:Circuit_elements.svg).

software *object*[49] representing an operon[50]. This would be a software module comprising a data structure and software routines operating on this structure. In such a software module, which takes into account time delays and inertia of genetic transcription, the transcription activated by the promoter is dynamically simulated, and the software object can output on request to an external program a value representing the actual instantaneous level of gene transcription. This way, the software model reveals itself as able to simulate with high fidelity the corresponding digital, simplified version, of the biological circuit. And, if simplification into a digital circuit (point *ii* in McAdams and Shapiro's quote above) does not undermine the *validity* of the model[51], that is, if it is not the case that this simplification introduces excessive approximation with respect to the observed phenomenon, then we can say that the software model ultimately simulates with sufficient fidelity the biological phenomenon.

[49] See section 4.2.3.

[50] an operon is, in bacteria and viruses, a module of genes which controls the production of a complex of enzymes carrying out a specific metabolic function.

[51] For the notion of model validity, see section 6.6.

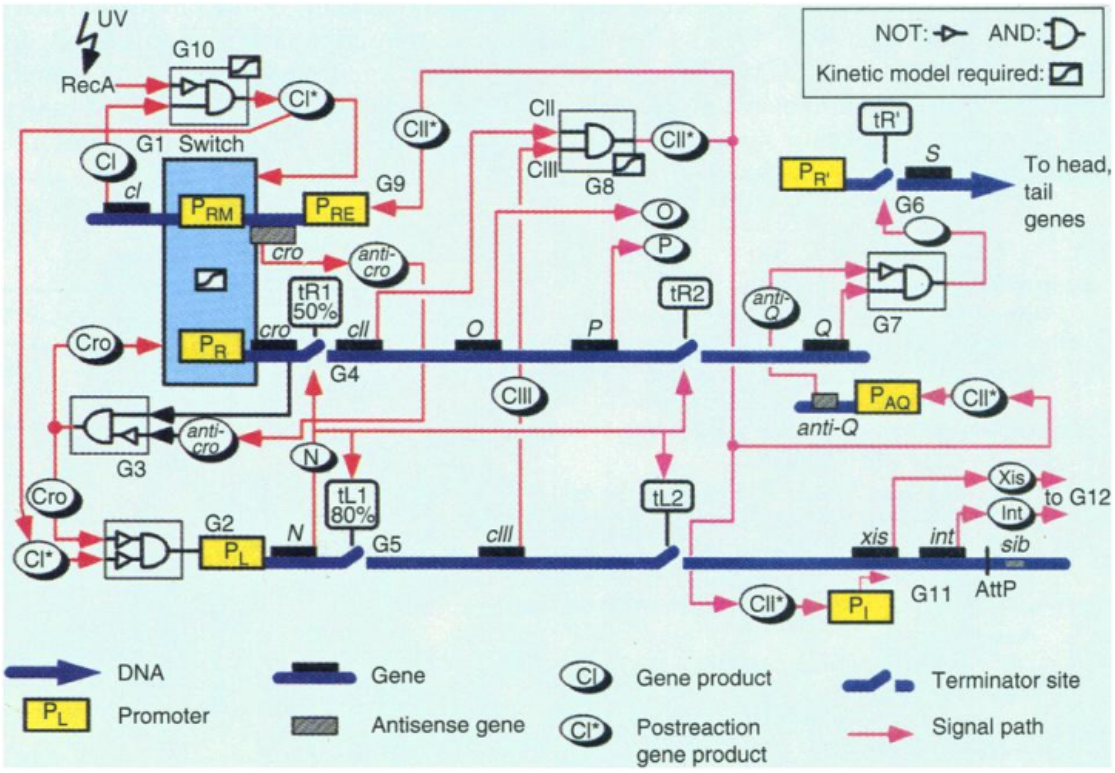

Figure 7.4: . schematic representation of the genetic circuit generating the λ phage lysis-lysogeny dynamics. (The phage is a virus affecting bacteria. Image taken from McAdams & Shapiro 1995, p. 652.).

Part III

Models of explanation

Preamble

This major section presents four prominent theoretical models of scientific explanation put forth in philosophy of science in the last seven decades.

Chapter 8 presents the most classic of these proposals, the so-called *deductive-nomological* model of explanation, first put forth by Carl Gustav Hempel and Paul Oppenheim in the 1940s. This has been the first model of explanation in philosophy of science, coming from the logical-empiricist background, and for this reason leaning toward explanation in *physics*. Neverheless, inside the cultural reductionistic milieu of the time, it was originally intended to be applied to science in general, a fact which has raised many critiques against it starting from the '70s.

Chapter 9 introduces another classic model of explanation, strongly advocated for by philosophers of mind, different from the deductive-nomological model and based on *functional decomposition*, which is at the basis of computational functionalism in philosophy of mind and of explanation in the cognitive sciences. This is a model which comes in several flavors, one of the most prominent of which is the model of *functional analysis*, proposed by Robert Cummins.

In chapter 10 is expounded one of the main opponents of the deductive-nomological model, the so-called "new mechanistic" framework of explanation, proposed in a first version since the '80s by William Bechtel, Robert Richardson and collaborators, and in an affine model formulated in a slightly different way by Peter Machamer, Lindley Darden and Carl Craver in the first 2000s.

Finally, chapter 12 introduces a type of explanation come under focus quite recently in philosophy of science, whose proponent is Philippe Huneman. This is the model of *topological explanation*, which considers certain types of explanations of quite frequent use in certain scientific disciplines, which explain certain features of complex systems in terms of mathematical a-causal properties.

Chapter 8

The deductive-nomological model of explanation

The first, seminal paper on the topic of scientific explanation in philosophy of science is Hempel & Oppenheim (1948). This first model of explanation emerges from a philosophical context characterized by the still strong influence on philosophy of science of the logical positivist view. Hempel himself was coming out of that milieu, and his account of explanation is perfectly in line with the basic tenets of logical empiricism. First, there is the prevalent interest for *physics*, among all sciences: the more fundamental science to which all other are to be reduced, at least in principle. With physics comes a conception of scientific theories as mostly constituted by the mathematical expression of *laws*: universal regularities, which take the form of universally quantified logical expressions. Of course, another fundamental postulate of that philosophical position is the distinction analytic/synthetic, or logical/empirical. Theories are eminently seen as linguistic, syntactic devices, in the form of formal systems, from which predictive statements can be deduced, statements which will be compared to the empirical observations. All these ingredients are well visible in Hempel's definition of explanation. Some quotation will make the point clear. From Hempel & Oppenheim (1948):

> the event under discussion is explained by subsuming it under general laws, i.e., by showing that it occurred in accordance with those laws, by virtue of the realization of certain specified antecedent conditions[1].

> the question "Why does the phenomenon happen?" is construed as meaning "according to what general laws, and by virtue of what antecedent conditions does the phenomenon occur?"[2].

> So far, we have considered exclusively the explanation of particular events occurring at a certain time and place. But the question "Why?" may be raised also in regard to general laws. [...] Thus, the explanation of a general regularity consists in subsuming it under another, more comprehensive regularity, under a more general law[3].

About the basic pattern of scientific explanation:

[1] Hempel & Oppenheim (1948), p.136.
[2] *ibid.*, p.136.
[3] *ibid.*, p.136.

> We divide an explanation into two major constituents, the explanandum and the explanans[4].
>
> By the explanandum, we understand the sentence describing the phenomenon to be explained (not that phenomenon itself); by the explanans, the class of those sentences which are adduced to account for the phenomenon[5].
>
> the explanans falls into two subclasses; one of these contains certain sentences $Cl, C2, ..., Ck$ which state specific antecedent conditions; the other is a set of sentences $L1, L2, ..., Lr$ which represent general laws[6].

According to this view, which is usually called the *deductive-nomological* model of explanation (*DN* model, henceforth), an explanation is constituted by a logical deduction of the explanandum from the explanans.

This kind of explanation must respect, to be a sound explanation, two classes of conditions: *logical* conditions and *empirical* conditions.

The logical conditions are:

- the explanandum must be a *logical consequence* of the explanans: if the explanandum cannot be deduced from the explanans, then the explanans is not an adequate ground to explain the explanandum.
- The explanans must contain at least a general law, which has to be employed in the derivation of the explanandum.
- The explanans must have *empirical content*, that is, it must be susceptible to be put to empirical test.

The *empirical* condition is that the premises of the argument, that is, the explanans, must be *true*.

In this view, there is complete symmetry between *explanation* and *prediction*:

> [...] the same formal analysis, including the four necessary conditions, applies to scientific prediction as well as to explanation. The difference between the two is of a pragmatic character. If E is given, i.e. if we know that the phenomenon described by E has occurred, and a suitable set of statements Cl, C2, ... , Ck, L1, L2, ..., Lr, is provided afterwards, we speak of an explanation of the phenomenon in question. If the latter statements are given and E is derived prior to the occurrence of the phenomenon it describes, we speak of a prediction. It may be said, therefore, that an explanation is not fully adequate unless its explanans, if taken account of in time, could have served as a basis for predicting the phenomenon under consideration. – Consequently, whatever will be said in this article concerning the logical characteristics of explanation or prediction will be applicable to either, even if only one of them should be mentioned. It is this potential predictive force which gives scientific explanation its importance: only to the extent that we are able to explain empirical facts can we attain the major objective of scientific research, namely not merely to record the phenomena of our experience, but to learn from them, by basing upon them theoretical generalizations which enable us to anticipate new occurrences and to control, at least to some extent, the changes in our environment[7].

[4] *ibid.*, p.136.
[5] *ibid.*, pp.136-137.
[6] *ibid.*, p.136.
[7] *ibid.*, p.138.

To sum up all the above conditions, we could say that, in general, a scientific explanation, according to Hempel, must be constituted by a logical deduction of the phenomenon to be explained, the *explanandum*, from a group of premises, called collectively the *explanans*, which consists of some statements representing certain physical initial conditions and at least a physical law, in the form of a logical universally quantified expression. If the phenomenon to explain is so deducible from these premises together with the general law(s), the phenomenon is in this way *explained*. It is evident that, given that the universal statement in the explanans represents a physical law, and that the initial conditions represent certain physical circumstances, a deduction starting from the explanans is a *prediction* of some phenomenon. So, we could say that a phenomenon is *explained* if it is *predicted* from certain empirical conditions.

The Hempel-Oppenheim model of explanation, is thus called, appropriately, the *deductive-nomological* model of explanation, for it involves a logical deduction from a set of premises containing at least a law-like statement.

Hempel and Oppenheim do not consider as proper cases of *explanation* certain ordinary language explanations, typical on non-scientific discourse, commonly regarded as being of "causal" nature. They are not explanations because they do not contain *laws* which allow for prediction:

> Many explanations which are customarily offered, especially in pre-scientific discourse, lack this predictive character, however. Thus, it may be explained that a car turned over on the road "because" one of its tires blew out while the car was travelling at high speed. Clearly, on the basis of just this information, the accident could not have been predicted, for the explanans provides no explicit general laws by means of which the prediction might be effected, nor does it state adequately the antecedent conditions which would be needed for the prediction[8].

In their seminal 1948 paper, Hempel and Oppenheim admit however causal explanation as legitimate, but they consider complete causal explanation only a special case of deductive-nomological explanation: proper causal explanation must contain at least a *law-like* statement, the "causal law"[9]:

> The type of explanation which has been considered here so far is often referred to as causal explanation. If E[10] describes a particular event, then the antecedent circumstances described in the sentences C1, C2, ..., Ck may be said jointly to "cause" that event, in the sense that there are certain empirical regularities, expressed by the laws L1, L2, ..., L, which imply that whenever conditions of the kind indicated by C1, C2, ..., Ck occur, an event of the kind described in E will take place. Statements such as L1, L2, ..., L7, which assert general and unexceptional connections between specified characteristics of events, are customarily called causal, or deterministic, laws[11].

Hempel and Oppenheim recognize the prevalence of causal explanations in the non-physical, so-called "special" sciences, such as biology and psychology. As said, in their view, causal explanations get identified with special cases of DN explanation: the only peculiarity of causal explanations is that, usually, only the antecedent conditions are explicitly mentioned, and that

[8] *ibid.*, pp. 138-139.
[9] In the quoted paper, the authors restrict causal laws only to deterministic cases, to avoid complication in the analysis which would have been brought in by the consideration of statistical laws.
[10] *E* is the explanandum.
[11] *ibid.*, p.139.

the law is taken for granted. This makes a causal explanation apparently employ only non-universal, particular statements: a singular event, the explanandum, is explained by a set of singular events, which are its antecedent conditions, or, in other words, its *causes*.

8.1 Known problems of the DN model

Traditionally, several external objections have been raised against the DN model of explanation. The most general ones complain that most explanations, both in ordinary life and in science, are causal and not deductive-nomological, and thus that the DN type does not outline the necessary conditions for explanation: a typical counterexample along these lines is that the explanation "the impact of the car caused the guardrail to bend" is a legitimate explanation of the deformation occurred to the guardrail, but nevertheless it is not a DN type of explanation, because in it no law-like regularity appears. A typical rebuttal by DN avocates, which, as we have seen, had been already anticipated by Hempel and Oppenheim, is that, even in causal explanation, regularities are *implicitly* invoked. In the above case the regularity would have been something like "any solid object hitting with sufficient velocity an iron strip will leave the strip deformed". This way, DN advocates, which usually endorse, in accordance with the post-logical empiricist tradition, a suspicious attitude towards causation, can eschew the question of causation while at the same time admitting at least certain forms of causal explanation.

There are two harder problems with the DN model, which put in doubt not only its being necessary, but its being sufficient to characterize any type of explanation.

Explanatory Asymmetry. The first is the problem of the *symmetry* of the explanation. In certain cases the laws employed in a DN explanation can be symmetric, like some types of equation, and as such these laws can allow for a reverse explanation making use of the same statements employed in the original explanation. However, while the "forward" explanations usually seem genuine explanations, the reverse ones often do not appear explanations at all, and could even seem absurd. This would, according to these critiques, reveal a fundamental asymmetry which appears as an important property of explanations, a property that the DN model completely overlooks.

For example, it is possible to explain, as prescribed by the DN model, the particular height of the sea tide at a certain place and time by deducing it from Newton's laws and the mass of the moon. Likewise, it is possible, by making use of the same laws in a reverse manner, to deduce the mass of the moon by measuring the height of tides at certain times and places. But this does not seem a plausible "explanation" of the particular mass the moon has: it would even seem absurd to explain the mass of the moon in this way.

This kind of objection is raised, again, by supporters of a primitive notion of causality, who see the asymmetry of causation as requiring asymmetry in genuine explanations as well.

Explanatory Irrelevance. Another typical problem with DN explanations is that of explanatory irrelevance: in certain cases, what appears as a perfectly genuine instance of deductive-nomological explanation, which ends with the deduction of the explanandum from legitimate law-like statements and initial conditions, results at the same time irrelevant as an actual explanation of the explanandum. For example, take the following deduction:

1. all samples of salt blessed by a bishop dissolve in water;
2. we put a blessed spoon of salt in water;
3. the salt dissolves in the water.

This is, formally, a perfectly legitimate DN explanation of why our spoon of salt has dissolved in the water. Nevertheless, the law-like statement, albeit true, is completely irrelevant for the explanation of the actual phenomenon.

Advocates of the DN model would answer that the task of distinguishing inside the set of formally correct deductive-nomological explanations between what is a legitimate explanation of a certain explanandum and what is not, is only a *pragmatic* matter, which depends on the aims and purposes of the researcher who looks for the explanation. All the above counterexamples, are, in principle, compliant with the DN model of explanation, and can appear as non-legitimate explanations only in relation to the pragmatic aspects of explanation regarding the expectation and the needs of the researcher. And, in line with a post-logical empiricist view, the matter of pragmatic aspects is not something within the scope of philosophy of science.

Understanding. Another question which must not be covered by philosophy of science, according to this view on explanation, is the question of *understanding* an explanation. For the DN view, understanding is indeed completely irrelevant to the status of an explanation: any sound logical deduction, however complex and unintelligible it can be, constitutes a perfectly legitimate scientific explanation, if the logical and empirical conditions of DN explanations are fulfilled. Understanding is something that, according to a neopositivist-inspired, strongly anti-psychologistic view like that of the advocates of the DN model, concerns only psychology, not philosophy.

In section 6.6.1, I started advocating instead an *epistemic* stance about explanation, which sees explanations as essentially linguistic and communicative devices, for which intelligibility is needed. I will develop such a stance in the following sections and chapters.

Chapter 9

Functions and functional explanation

9.1 Functions

The notion of *function* is central in many disciplines, ranging from mathematics to computer science, to biology, to philosophy of mind, but it has no single, unproblematic, widely accepted meaning. Some brief reflections on this are in order: I will try to schematize in what follows how the notion of function can be understood in different contexts. It is not my intention to be thorough nor to delve deeply into the many long-standing debates which surround this notion, but only to outline some rough distinction which I think will be of possible use in later discussion in this work.

I will discriminate here between five conceptions of *function*:

- in *mathematics*, a function is something that maps a set to another set, with the clause that each element of the first set gets associated to at most one element of the second set. It is not necessary to specify the mapping between the sets intensionally, that is by an explicit rule: the function can be specified *extensionally*, by showing a set of *ordered couples* of the elements of the two sets: these are all and only the couples of elements which are related through the function.
- in biology, however, when the word *function* is mentioned, we are probably not dealing with functions understood as mere input/output relations, but, in general, with functions with some sort of "teleological" or "causal" flavor: albeit with quite different nuances, functions are in these cases defined as something which is "for" something else, or something which is meant to have some desired effect, or which fulfills a certain causal role with respect to the whole system[1]. Two main theoretical views of the nature of functions are generally considered in biology:

    1. *Causal role functionalism.* According to causal role functionalism, a subsystem performs a certain *function* if it plays a certain *role* in the causal web that constitutes the function performed by the *whole* system. For example, causal role functionalism

[1] I'm not going to enter here the long-standing debate about the nature of teleology or intentionality still underway in philosophy of biology or philosophy of mind: I use here the term "teleological" in a mere metaphorical sense. The same can be said for the debate about the nature of physicalism (see below), of natural kinds (which has been touched upon in the Introduction though), and other similar ongoing metaphysical quarrels in the philosophy of mind or metaphysics of science, which are outside the scope of this work: I will, in what follows, make a quite instrumental use of the notion of function derived from such philosophical positions.

sees a heart as a subsystem of the body which performs the function of "pumping the blood", in order to maintain the organism in life: this is the causal role of the heart in the web of causes and effects which constitute the vital process of the organism. Other than a conception of the notion of function, causal role functionalism is also typically adopted by the metaphysical position known as *non-reductive physicalism* in philosophy of mind. Roughly put, this is the rather still mainstream view, in philosophy of mind and in cognitive sciences, started in the late '60s by Hilary Putnam and later supported by Jerry Fodor and many other authors. This philosophical position maintains that mental states and capacities should be seen essentially as *functions*, each of which plays a certain role in the interconnected web of mental functions which constitutes the whole cognitive system: types of mental states are conceived as *functional kinds* and not as *physical* kinds, albeit functional kinds must be in some way *realized* by physical kinds: this guarantees that the position is still *physicalistic*. This way, *psychological* explanation of mental phenomena can abstract away from the neurological details which implement (or, in other words, which "realize") the mental functions mentioned in the explanation. A typical feature of this notion of function is that a function is *multiply realizable*[2]. This means that more than one specific physical system can *realize* that function, that is, can come to occupy the specified causal role in the causal web constituting the entire system. In other words, there are *more* possible ways to perform the *same* causal role by means of *different* specific physical systems or processes: for causal role functionalism in general, what counts for a function is not *the particular way* a lower-level system[3] realizes it by performing a *specific* process at the lower level, but the fact that this lower-level activity takes the right *place* inside the *web of interconnected causes and effects* which constitute the whole system. For example, the function of being a heart, defined as the function of pumping blood in the circulatory system of an organism, can be "realized" by a natural heart, but also by an artificial pump: it is realized by at least two very different (in constitution and way of functioning) physical systems. The function is thus multiply realizable.

2. *The etiological view of functions.* In evolutionary biology, the idea of the role of a subsystem is usually linked with its evolutionary history: the heart has been "selected for" its ability to pump, precisely because, in past generations, this function has allowed individuals which were able to perform it to thrive and reproduce better than those lacking the function. This phylogenetic, or *etiological* view of biological functions was started in 1973 by Larry Wright in a seminal paper, and subsequently developed mainly by Karen Neander and Ruth Millikan[4]. An explanation based on the etiologial conception of functions is typically employed in answering to a question about why a certain phenotypic trait or ability is showed by an organism: the answer

[2] Actually, multiple realizability is the *defining* feature of psychological *functionalism*, or at least the original inspiration which brought Putnam (1967a) and Putnam (1967b) (a reprint of the former) to conceive this philosophical solution to the mind-body problem. The notion of realizability and the question of non-reductive physicalism are chock–full of metaphysical caveats and ongoing quarrels, which is not surprising, given that they belong to what is still considered the dominant positions in philosophy of mind. Understandably, I defer a deeper analysis to some external work, such as, for a thorough and enlightening exposition of many of these problems, the excellent Polger (2004).

[3] I'm using here the notion of *level* on a purely intuitive ground, to describe the "concrete" (usually physical) level that is below the level of functions, which is a "higher" and "abstract" level. The very notion of multiple realizability of functions asserts this degree of abstraction: the fact that the *same* function is multiply realizable by different lower-level systems, means that its identity ignores the details of its realizations, or in other words, abstracts them away. That said, the notion of levels is definitely in need of a better clarification, a task which is attempted in section 6.6.

[4] See Wright (1973), Millikan (1984), Millikan (1989) and Neander (1991).

> is that the property, which must be genetically determined in some way, has raised, by performing a certain causal role, the fitness of the ancestors of that organism, thereby raising the probability of its genetic base to be itself passed on to future generations, until today. Such a conception of functions allows for the attribution of a function to a *malfunctioning* organ: a malfunctioning heart can still be considered functionally a *heart* even if it is not currently able to pump blood: its identity as a heart is guaranteed by the homologous organ in ancestors having been selected *in the past* for pumping blood. On the other hand, in a non evolutionary causal role view, a currently non-functioning heart cannot be considered a *heart*, in that it does not, at the present moment, fulfill the role of pumping blood in the system.

- In *computer science*, a function is defined by showing an *algorithm* that gets to map a set of possible *inputs* to a set of possible *outputs*. A less detailed or more informal description of a program's input/output mapping is called the program's *specification*[5]. This is not a completely unproblematic notion, as discussed in sections 4.1.3 and 4.1.5. The notion of computational function, or of program, so understood, contains also some teleological aspect. I refer to the aforementioned sections for a discussion.

9.2 Functional analysis

According to most secondary accounts[6], one of the main proponents of the concept of function as causal role is Robert Cummins. However, I would like to consider his position as a third, slightly different flavor of the notion of function, which I will call *Cummin's functionalism* or *explanatory-role functionalism*. In many of his seminal papers Cummins never talked of *causes*, and this seems to me to indicate that it is not advisable to put the potential metaphysical burden which the concept of cause can bring along, on Cummins' conception of functions. Moreover, by his very definition of function, Cummins himself stresses that this notion is conceived in order to give an account of a certain type of *explanation* which is often appealed to in explaining complex systems.[7] Cummins writes:

> Functional analysis consists in analyzing a disposition into a number of less problematic dispositions such that programmed manifestation of these analyzing dispositions amounts to a manifestation of the analyzed disposition. By "programmed" here, I simply mean organized in a way that could be specified in a program or flow chart.[8]

Here, by "disposition" Cummins means a certain *capacity* of the system. While usually linked to the notion of causality, Cummins gives a non-causal account of the notion of *disposition* in Cummins (1974). In that paper, the "realist" conception of dispositions held by D. M. Armstrong, which sees a disposition as the possession of a state which is causally responsible for the disposition's manifestation in certain circumstances, is explicitly rejected by Cummins, which sees an object's disposition in terms of the object's potential possession of a state, which allows for an *explanation* of the disposition by means of an analysis of that state in *non-dispositional* terms (something which preludes to Cummins' subsequent treatment of functional analysis). This seems to me another reason to distinguish his account of function from generic causal role functionalism, by calling it "explanatory-role functionalism": if there is some use of the notion

[5] See section 4.1.5.
[6] Among them, for example, Couch (2011).
[7] This also seems to be the opinion of some commentators, like Amundson & Lauder (1994).
[8] Cummins (2000), p. 125.

of cause in it, this is only as a form of causal explanation. I would rather leave unanswered, at least here, the question if such a type of explanation captures or not some metaphysically *real* characteristic of the explained phenomenon.

In Cummins' view, the consideration of the wider context in which each functional component is immersed adds to this kind of functionalism the ability of explaining the system by describing how its subfunctions act in an organized way: the system can be explained by producing a symbolic description (as we have seen above, Cummins talks of *flow charts*[9]) of the web of functional blocks which make up the whole system. Like in causal role functionalism, the *role* which the function performed by the subsystem under scrutiny plays inside the larger system is what is taken into consideration. The difference here is that an *explanatory* role, not the causal one, is considered. Leaving aside, as announced, this subtle (or not so subtle) metaphysical distinction[10], it seems to me that, however, Cummins-style functions do not entertain relations with the evolutionary history of the system in the way causal role functions, in their etiological variant, do[11]. Unlike etiological explanations, which, as we have seen, are used to answer evolutionary "why" questions, Cummins-style ones are typically employed for answering questions about *how* a certain capacity of the system is brought about: the answer lies in showing how this global capacity can be expressed in terms of sub capacities and sub-sub capacities, and so on recursively.

9.3 Functional explanation of computational systems

Functional analysis as conceived by Cummins is typically applied in describing and explaining complex systems, and as such can be, and is, applied to computational systems. In this case, I think we could assimilate the "disposition" mentioned in Cummins' above definition with what in computer science terminology is called the "specification" of the program[12], that is, intuitively, a declaration of what the program is supposed to do. A specification can also be seen as the *function* in the mathematical sense, which the system as a whole performs, or calculates. A specification can in a way be assimilated to a disposition because it defines what the output value of the program is, in case the input is set to some value: this is indeed quite similar to the philosophical notion of disposition, which can be stated as some property something would manifest should certain conditions happen to hold.

In computational systems, functional analysis can then be considered as the decomposition of the computational system into computational subsystems. Each subsystem can naturally be seen as a *function* in the pure *computer science* sense, but Cummins-style functional analysis adds to

[9] See section 4.1.4.4.

[10] This distinction can be certainly connected with the quarrel between the so-called *ontic* and *epistemic* views in the philosophy of mechanistic explanations, which I will touch upon in section 10. It can also be related to more general questions of realism/antirealism in science, which I will not directly engage in this work.

[11] Amundson & Lauder (1994) agree in seeing Cummins-style functions as fundamentally different from etiological functions. They contrast Cummins' notion of function with the etiological one by calling the former "function without purpose". The same paper argues that this does not mean that Cummins-style functions do not appear in biological explanations, but that, on the contrary, they very often do. It seems to me that calling functions as understood by Cummins "without purpose" could be a little misleading, for, as defined by Cummins, functions *do* have a form of purpose, albeit a weak type of purpose: namely, the specific sub-capacity, which comes to compose the global capacity of the whole system. In my view, the only conception of purposeless functions is the *mathematical* one, although I think even this lack of "purpose" can be debated, if we understand "purpose" in a very loose sense: the *name* itself which can be attributed to certain standard functions (e.g. $sin(x)$), or the analytical expression of some functions (e.g. $y = 3x^2 + 2$) can in a way constitute the *specification* of the function, and as such could be seen as the capacity or disposition it shows. Probably, completely purposeless functions are the mathematical functions which cannot be analytically defined nor easily named.

[12] The notion of *specification* is discussed in section 4.1.5.

this the information that these functions play a role in *explaining* the global function performed by the whole system.

Applied to computational systems, this ends up being a form or role functionalism, but not of "causal role" functionalism, because, in abstract computational systems, no notion of causality is taken into consideration: as Polger (2004) properly highlights, causal role functionalism as usually intended in philosophy of mind (for example by Jerry Fodor), must, for reasons internal to that particular theoretical position, maintain that functional properties (such as the mental properties) are actually endowed with causal powers *qua* functional properties. A well-known argument, usually known as the argument from "causal exclusion", whose major proponent is Jaegwon Kim[13], shows that actual possession of causal powers by these "higher level" functional properties would be contradictory[14]. If the argument holds, the higher-level properties end up being still equipped with causal powers, but not *qua* higher level properties, because these causal powers actually belong to the lower level properties which *realize* the higher level ones. That means that higher level properties still continue to support counterfactuals, but only in virtue of their realizers. Some see this as equivalent to declaring the mere epiphenomenality of higher level properties, and consider this result a problem for special sciences such as cognitive psychology. I do not think, however, this could turn out to be a problem at all for computer science: in this field, it is usually considered obvious that software, *as such*, is not equipped with *causal* powers. Causality is not even taken into consideration by computer science, for it is a notion which, if anything, pertains to empirical science. It is perfectly normal that constructs in computer languages are declared devoid of causal powers and are not explained in term of *causal* features. If something is causal in computers, this manifests only in actual, real-world computers, at the hardware level: causality, if at all, can be considered operating in electronic circuits. But computer science does not deal with electronics, but only with abstract logico-mathematical machines. Nevertheless, explanations which are broadly speaking *causal* can be employed in explaining computational systems for the reason that computational systems do support counterfactuals, in that they deterministically pass from one state to another state at every timestep. This can be seen as isomorphic to causation[15], but I think we would be in a better position by stating that this regularity in computational systems bears resemblance with a sort of grammatical regularity[16].

The explanatory capacity of functional analysis according to Cummins' account is probably

[13] Many works by Kim treat causal exclusion and cognate arguments, starting from Kim (1989c) and Kim (1989b).

[14] There have been, however many attempts to counter this argument, which I'm not going to treat here. According to Polger (2004), much discussion around it has been due to terminological and conceptual misunderstandings.

[15] At least to a counterfactual-based account of causation.

[16] I think that causality proper plays no role in the idea of computation, basically because computer science treats, like mathematics, systems which are at least in principle idealized systems, with no consideration of the notion of causality, a notion which presumably applies to the physical or "real" world. Regularities in these idealized models are not due to physical or causal constraints, but to other reasons, which in my opinion are, to simplify a little, of *conventional* or "grammatical" nature: it is a condition taken for granted by computer scientists that a Turing machine must obey the constraints of its (idealized) physical structure and the instructions of its machine table. This constraint has the same cogency implied in accepting a grammatical rule: if someone does not want to obey a certain grammar, she is allowed to, but she would end up simply *not* speaking *that* language, that's all. We could obviously conceive, without violating any causality, a Turing machine which does not obey its transition table: it will simply *not be* a Turing machine. In computing, causality could at best have (and it *has*, probably) a role in the regular functioning of the "hardware", but computer science does not deal with the hardware. Or, to put it another way, it is my opinion that "hardware" is a relative concept. The reason for stating this will come out clearer in section 6.6. I admit the position I sketched here can sounds dangerously affine to a tentative solution to the hard, long-standing "kripkensteinian" problem of rule-following, but I leave a deeper analysis of the problem to a later occasion.

nothing new, at least in computer science: any programmer, even if she does not have to do it, at least implicitly *explains* to herself how the program she is developing works by concerted execution of its subroutines. The act itself of designing a certain software requires that the programmer knows the specification of the whole program (e.g, the specification to be "a web browser"), and development proceeds by analyzing (more or less in Cummins' sense) this global function into smaller subfunctions, which in turn get decomposed in simpler subfunctions, and so on. The subsequent practice of lying down the sequence of instructions constituting each subroutine, usually starting from the simplest ones, is the phase of *implementation*, which is usually impossible without a former, at least implicit, explanation of the whole system in the above terms. And this can be seen as a form of *functional explanation* the way Cummins understands it.

Chapter 10

Mechanistic explanation

Although the idea of mechanical explanation, at least for what concerns the modern tradition, dates back since Hobbes, Descartes and their time, in contemporary philosophy of science the question of the status and properties of mechanistic explanation has been partially neglected for almost three decades after WWII. This omission is quite possibly due to the preference accorded in philosophy of science to the study of explanation in *physics*, where mechanistic explanations seldom appear, and to a bias toward *observable* phenomena and a corresponding suspicion towards causation, seen as a concept liable for metaphysical taint: these are inclinations that date back to the logical empiricists of the Vienna Circle, and that have been still dominant in philosophy of science from the beginning of the '50s until the '80s. Nevertheless, explanation of a mechanistic kind has been the main form of explanation employed in most biological and, later, psychological scientific research lines (for example in molecular biology, or cognitive neurosciences), during a period which includes the same post-war decades in which philosophy of science has mostly ignored this type of explanatory style.

Especially in the social sciences, however, the notion of mechanisms had began to attract quite early some amount of methodological and philosophical attention on the part of certain authors. In particular, Mario Bunge[1] criticizes what he calls a *descriptivistic* view of explanations, that is the view preferred by logical empiricists and their followers, which proposes to treat phenomena as black boxes, seeing them only as expressions of (law-like) regularities. This is the view exemplified by the the deductive-nomological model of explanation[2] of Hempel and Oppenheim, typical of the still neopositivistically-inspired aforementioned post-WWII "received view" in philosophy of science. As we have seen, in deductive-nomological explanation a correlation between certain conditions and the outcome to be explained is subsumed under a general law. Bunge attributes this sort of superficiality of the obtained explanation to the necessity, for the logical empiricist position, of strictly sticking to the empirical observability of phenomena, with the consequence of cutting out most *intermediate* mechanisms between the initial conditions and the outcome, mechanisms which are quite often *unobservable*[3]. According to Bunge,

> Descriptivism not only curtails scientific research: it also encourages collecting disjointed anecdotal material and the blind search for statistical correlations. This strategy may also

[1] See Bunge (1997).
[2] See section 8.
[3] Unobservability of mechanisms holds true not only for microscopic physical or chemical ones, but also partially for certain macrosopic mechanisms, for example social mechanisms, in which the aggregate interactions between agents are not directly detectable.

> encourage superstitious beliefs rooted in mere coincidences or "synchronies" whereas a demand for a plausible mechanismic explanation would rule them out. Descriptivism enshrines mysteries instead of turning them into research problems.[4]

Bunge thinks that only the revealing of the mechanism underlying a given phenomenon can confer perspicuity and depth to an explanation. For this reason, he started advocating, since the late '50s[5], the use of mechanistic explanations, especially in the social sciences. In accordance with his anti-neopositivistic view, he considers, assuming a realistic stance, that only concrete, physical systems can be mechanisms, anticipating in a way a still open question about the metaphysical status (ontic or epistemic) of mechanistic explanation, which will be touched upon in what follows.

According to Bunge[6], mechanistic models are more deeply explicative than simply descriptive theories, because mechanistic explanations employ ontologically finer entities than those superficially observable. This feature of being more detailed and precise makes them logically stronger than descriptive theories, and, as such, subject to a higher degree of falsifiability. In the light of post-popperian fallibilism, according to Bunge, mechanistic models are, for these reasons, to be preferred to simple descriptive theories.

Also Rom Harré, since at least the end of the '50s, had began to put forth a notion of scientific explanation by means of mechanisms[7], opposed to explanation by subsumption under a law, which is the DN kind of explanation.

The basic idea underlying Harré's and Bunge's positions is that, as an explanation, while the DN model[8] renders unsurprising a given systematic relationship between a class of certain initial conditions and a class of certain outcomes by subsuming it under a general law, this correlation lacks *itself* an "explanation": unsatisfied "why?", and, especially "how?" questions may still be lingering even in presence of a fully valid DN-type explanation. The description of a *mechanism* taking as initial conditions the initial conditions cited in the DN explanation and producing the phenomenon to be explained (that is, the *explanandum*), would precisely come to constitute an explanation of the lawful correlation employed in the DN explanation, an explanation of the correlation *itself*[9]. This seems a position analogous to that of Robert Cummins, who considers psychological "laws" as themselves in need of functional explanation.

It thus appears that this vision of mechanistic explanation, as highlighted by Bunge, is, in a way, *compatible* with the DN model. It is a *deeper* form of explanation than the latter, and can usefully (hopefully) be the complement of a DN-type explanation: the mechanism reveals itself as the producer, with its functioning, of the law-like regularity cited in the DN explanation. This relationship between the two types of explanation is plausible, at least in cases in which the implied mechanism generates a regularity that is sufficiently general to be considered a *scientific law*.

---

[4] Bunge (1997), pp.421-422.
[5] Beginning with Bunge (1959), Bunge (1963) and Bunge (1964) (cited in Bunge 1997, p. 460).
[6] See for example Bunge (1997), p. 460.
[7] See Harré (1959).
[8] For the DN model, see section 8.
[9] In computer science terms, we could say that the correlation between the initial conditions and the explanandum constitutes the "program specification", that is the specification of the input/output relation, while the mechanism producing the explanandum is the program which implements the specification (see section 4.1.5). This is an analogy which I will make use of many times later on.

Chapter 11

The *new mechanistic* school

In more recent years another group of philosopher has begun to undertake, quite independently from the aforementioned research line, an attentive analysis of mechanistic explanation, especially in the biological sciences, as a complete alternative to the deductive-nomological model. This theoretical line is quite often called the "new mechanistic" position.

This philosophical view of mechanisms is based on the consideration that in biological sciences there is an almost complete *lack* of scientific laws, and that, for this reason, it would seem philosophically inappropriate to forcedly impose the DN model of explanation in these research fields. In fact, upon inspection of scientific literature, it turns out that, at least since the inception of molecular biology and genetics, most biological or psycho-biological explanations proposed in scientific papers have been instead of a mechanistic type. Theoretical refining of the intuitive characterization of "mechanism" is precisely the object of the new mechanistic school of thought in the philosophy of science. Authors following these lines do not usually try to connect mechanistic explanations with DN-type ones, not because they reckon it is not to be expected that mechanisms come to produce regularities (quite the contrary[1]), but mainly because it actually turns out that in biological research scientists almost never refer to biological *laws* proper: most biological phenomena are quite specific and bound to specific biological contexts, and appear likewise to be generated by specific mechanisms: consequently, regularities produced by these mechanisms are too limited in scope, and not sufficiently general to be considered *laws* belonging in the same class of what are more proper scientific laws.

Probably the best known exponents of the *new mechanistic* current in contemporary philosophy of science are Lindley Darden, Carl Craver, Peter Machamer, William Becthel and Stuart Glennan. These authors are apparently differentiated in two groups, and this is evident since the origins of this line of thought, which stems from two series of seminal papers. Although works by Bechtel and his associates on the subject had already begun in the early 90's with Bechtel & Richardson (1993), it is probably the publication of Machamer et al. (2000) which sparked a growing interest in the subject.

The two flavors of philosophical reflection on mechanistic explanation stemming from these lines differ in some seemingly secondary theoretical aspect, which can however have far reaching consequences. It is not my goal here to produce a detailed analysis of the evolutionary lines of

[1] Usually, *deterministic* mechanisms are taken into account. It could well be, and it is quite possible, that certain biological mechanisms show intrinsic non-determinism, due to their being based on random phenomena. An example would be that of natural selection. However, this case depends on the decision to consider natural selection a *mechanism*, and this is a highly debated question to date.

these two approaches: my purpose is to delineate these two positions in order to discuss them further in relation to some proposals I intend to put forward in later parts of this work. To this aim, I will now proceed to give an account of two representative papers, Machamer et al. (2000), and Bechtel & Abrahamsen (2005), in order to later expose some critical observation coming from other authors.

11.1 Machamer, Darden and Craver's account of mechanistic explanation

According to Machamer, Darden and Craver (*MDC* henceforth), in many scientific fields, especially in biology, explanation is given in the form of a description of *mechanisms*. Notably, in their seminal Machamer et al. (2000), these authors state that no satisfactory analysis of what mechanisms are and of their role in science had been carried out yet, at the time of their writing. They claim that, nevertheless, much of the history of science can be seen as "written with the notion of mechanism". In their initial study, they deliberately choose to concentrate on biology and neurobiology, not excluding (but leaving it as an open question) that explanations based on mechanisms could characterize other sciences as well.

Machamer, Darden and Craver give a well-known definition of mechanism:

> Mechanisms are entities and activities organized such that they are productive of regular changes from start or set-up to finish or termination conditions[2].

In MDC's philosophy, a *mechanism* must not be intended only as composed of truly physically mechanical, "push-pull" terms, but can take many forms in scientific explanations. A mechanism is meant to explain a phenomenon or process. Descriptions of a mechanism show how termination conditions are *produced* by the set-up condition and intermediate stages. This description is an *explanation* of the phenomenon[3].

Ontologically, MDC see mechanisms as composed of *entities*, which engage, according to their specific properties, in certain *activities*. Division between entities and activities is the main ontological partition of what mechanisms are composed of. Emphasis is put on the *productive* aspect of mechanisms: entities are precisely those things that engage in activities. Entities and activities are correlatives and interdependent: entities with certain properties are necessary for having certain activities, and vice versa. Both entities and activities are necessary for what MDC call an *ontically adequate* description of a mechanism.

As expected, spatio-temporal properties of mechanism components and of their arrangements are fundamental for the functioning of the mechanism: "Entities often must be appropriately located, structured, and oriented, and the activities in which they engage must have a temporal order, rate, and duration"[4].

Mechanisms display a *regularity*: under the same conditions, they work almost always in the same way, from set-up to terminating conditions, with *productive continuity* between stages, without gaps. This *productive continuity* is what makes the connection between stages intelligible. It is

[2] *Ibid.*, p. 3.

[3] Although MDC seem to attribute the status of explanation to *descriptions* of mechanisms, Carl Craver, one of the authors, has subsequently stressed in other papers that he sees, in a way expressly informed by Wesley Salmon's view, explanations as constituted by the *exhibition* of the *actual*, real-world mechanism. MDC formulation of explanations in terms of descriptions could leave this ontic position implicit.

[4] *Ibid.*, p. 3.

often represented with arrows: $A \rightarrow B \rightarrow C$. A missing arrow would indicate an *explanatory gap* in the productive continuity of the mechanism.

11.1.1 Mechanisms and functions

MDC seem to adhere to a form of causal role functionalism: functions are the roles played by entities and activities in the context of the mechanism[5]. They explicitly stress the need for consideration of the context in order to define a function, a context which must be "taken to be important, vital, or otherwise significant". They reject a contrasting view on functions, that they call "substantivalist", which considers functions as *properties* of entities. The position MDC adopt, which compels to take into consideration the wider context in which entities and activities play, seems to be a way to prevent reductionism. A *mechanism* in turn can have a function, if the mechanism, as a whole, contributes to something in a wider antecedent context which is taken as important.

11.1.2 Activities, causes and laws

Entities act as *causes* when they engage in productive activities. MDC state that their view of mechanisms is in some ways compatible with Wesley Salmon's causal mechanical philosophy. Wesley Salmon' s philosophical position is a very important view about explanations in terms of causal processes, a view which back in the '80s broke the orthodoxy of the hempelian model of explanation and its disdain for causation. Salmon's position is the original inspiration of most subsequent philosophical views on mechanistic explanations, especially on the MDC's side.[6]. MDC's position, along these lines, embraces a fully metaphysical view of causation, seen as brought about by what they call *productive activities*.

While in neurobiology and molecular biology there is ample use of mechanisms and scant evidence of natural laws as traditionally understood in physics, MDC admit that the notion of *activity* shares with natural laws the property of being a non accidental regularity, able to support counterfactuals. It has to be considered part of the definition of mechanism the proviso that its functioning must occur in a regular way. So, a sort of necessity is implied by a mechanism, but without the need, according to MDC, to posit some law-like necessity underwriting it.

It is clear that these two combined positions about causation and laws, which see causes as metaphysically real and not requiring a connection to laws in order to bring about regularity, stand in complete opposition to the view advocated by supporters of the DN explanation, which sees causes as just Humean observed regularities and natural laws as *warranting* the regularity of causation.

11.1.3 Diagrams

According to MDC, *diagrams* can be used to describe mechanisms, for they exhibit the spatial and structural features of the entities, along with the activities, which are usually depicted as arrows. But they claim that diagrams, while facilitating apprehension, can actually be substituted by *verbal* descriptions. This, as we will see, goes against Bechtel and Abrahamsen's opinion, who think diagrams are an essential part of mechanism descriptions.

[5] A more thorough discussion on this and other conceptions of function has been carried on in section 9.

[6] Although of such importance in philosophy of science, an analysis of Wesley Salmon's position is outside the scope of this work.

11.1.4 The working cycle of a mechanism

Set-up and termination conditions are part of the definition of mechanism.

- *Set-up Conditions* are idealized static initial conditions, that can be themselves results of a prior process, and comprise the relevant entities and their activities, structural properties, orientations and spatial relations, as well as any environmental enabling condition (the latter often omitted for the sake of simplicity). MDC stress that set-up conditions are not mere *inputs* for the mechanism, separated from it. Rather, they are *part* of the mechanism, crucial to make the process go on.
- *Termination Conditions* are conditions describing an idealized static endpoint of the mechanism, considered and endpoint depending on the observer goals, or for the reason it constitutes the final stage of what is considered a unitary process. MDC stress that termination conditions are not *outputs* of the mechanism, for this would inaccurately suggest something *coming out* of it[7].
- *Intermediate Activities*: these are the intervening entities and activities that produce the termination conditions starting from the set-up conditions. MDC stress that in a complete description of a mechanism, no gaps leaving specific steps unintelligible should appear (something they also call *productive continuity*).

An important remark is that the structure of stages in the process of a mechanism must not necessarily be linear but can show forks, joins or cycles. MDC also stress that often mechanism are actually continuous processes, treated as a series of discrete steps only for convenience.

11.1.5 Hierarchies and Bottoming Out

MDC highlight that mechanisms, in explanations, often occur in *multi-level* hierarchies, where lower level entities, properties and activities act as component of higher level mechanisms. A whole mechanism can in turn be seen as a component of a higher-level one.

Descent along the hierarchy must eventually bottom out. This occurs as a matter of convention. The lowest level of the hierarchy is chosen by the researchers based on a subjective evaluation: the lowest level is the one which is considered by the scientist or the scientific community fundamental, or for which it is not considered interesting trying to further analyze it by decomposing it in smaller parts (for example, in molecular biology, the bottom level is usually considered that of macromolecules, or at most, smaller molecules and ions). Of course, this makes the bottom level relative to a scientific area, and, in a given area, what is considered bottom level can historically change due to changes in the discipline. This view is perfectly compatible with my view on levels of description which was expounded in section 6.6, where the bottom level is what I called *preferred description*. There are, however some relevant differences on the metaphysical status of descriptions between my position and the one proposed by MDC, which is definitely more *ontic*.

Bottoming-out involves identifying bottom level *activities* beside bottom level entities. Specifically, in molecular biology and neurobiology, MDC think bottom level activities can be so categorized:

[7] I find these elucidations by MDC somewhat problematic: it seems obvious that "output", despite its etymology, does not necessarily refer to something which is put *out* of something else considered as a context, and in the same way, input is not necessarily something that must be pushed inside some contextual border. For example, in a computational system any internal configuration which occurs at the end or at the beginning of the computation can be considered input or output. These considerations could, for example, be applied to inputs and outputs of cellular automata, as sketched in section 5.2 and 5.2.4.

- geometrico-mechanical activities (the activities familiar from modern XVII century mechanism: pushing, pulling, turning and such);
- electro-chemical activities: attracting, repelling, bonding and breaking (as, for example, in enzyme activity);
- energetic activities: activities involving thermodynamic phenomena.
- electro-magnetic activities: these are activities at times used to bottom out mechanisms in molecular biology and neurobiology (es. the conduction of electrical impulses by nerve cells).

11.1.6 Mechanism schemata, mechanism sketches, explanation, and scientific theories

According to MDC, a *mechanism schema* "is a *truncated abstract* description of a mechanism that can be filled with descriptions of known component parts and activities"[8]. Often scientists provide only mechanism schemata, for in their research they are typically interested in *types* of mechanisms, omitting the details that make up a specific mechanism, or they do not intend to provide complete descriptions of mechanisms at all levels in a hierarchy.

I think it should be noted how in their definition of mechanism schema MDC have felt the need to specify that the latter is an "abstract description". It seems safe to infer that their concern is that of distinguishing the metaphysical status of *description* from that of *mechanism*. This is due to the plausible claim, that seems corroborated by other interpreters of their philosophical position, that MDC consider mechanisms as *real*, *actual* structures existing in the world[9], and not as *representations* of a mechanistic form. A schema is a *description* of a mechanism, and the *actual* mechanism which it describes is said to *instantiate* the schema. An explanation proper is constituted not by the abstract schema, but by the *actual*, real-world mechanism: "When instantiated, mechanism schemata yield mechanistic explanations"[10]. This point of view (in my opinion a counterintuitive view), that considers the *actual* mechanism and not a representation of it, as the explanation, is typical of MDC, and especially evident in Carl Craver's explicitly *ontic* view of mechanistic explanations, which can be contrasted with an *epistemic* view, typical of other authors, as William Bechtel and colleagues.

Differently detailed schemata come from different *degrees of abstraction*, an operation consisting in removing details from an exemplary specific case[11]: abstraction deals with the amount of detail included in the schema seen as a description of some real instance of a mechanism.

The *generality* of a schema is the scope of the scientific domain in which a mechanism that instantiates it can be found.

MDC stress that abstraction degree and scope are orthogonal properties: narrow scope and high abstraction can coexist, and vice versa.

Regarding scientific theories, MDC see them as occurring in some sciences in the form of mechanism schemata: in neurobiology and molecular biology, often the term "theory" actually refers to "hierarchically organized mechanism schemata of variable, though generally less than universal, scope."[12] According to MDC, in these disciplines *mechanism schemata* play indeed many of the

[8] Machamer et al. (2000), p. 15.
[9] and, consequently, as realizers of their own explanation.
[10] *Ibid.*, p.17.
[11] I have treated at length the notion of abstraction in section 6.6.
[12] *Ibid.*, p. 16.

roles traditionally attributed to *theories* in other scientific fields. In MDC's words, *mechanism schemata*

> "are discovered, evaluated, and revised in cycles as science proceeds. They are used to describe, predict, and explain phenomena, to design experiments, and to interpret experimental results."[13]

As theories, mechanism schemata can allow for predictions on the basis of their structure and properties.

At the same time, schemata also provide a "blueprint" for designing experiments to put them to test:

> "A technician can instantiate a schema in an experiment by actually choosing physical instantiations of each of the entities and the set-up conditions and letting the mechanism work. While the mechanism is operating, the experimenter may intervene to alter some part of the mechanism and observe the changes in a termination condition or what the mechanism does. Changes produced by such interventions can provide evidence for the hypothesized schema"[14].

Sketches are mechanism abstractions different from schemata. They are *incomplete schemata*, that is, mechanism abstractions which, due to a current lack of knowledge, lack some details: for example, it can happen that bottom-out entities cannot be provided, or productive continuity is interrupted by missing stages, usually substituted by black boxes. A sketch thus serves to *guide further research*, and it can either be abandoned in the light of new findings, or *become a schema*.

11.1.7 Intelligibility and multi-level mechanistic explanation

MDC state that the mechanistic world view brings with it some expectations about how phenomena must be rendered *intelligible*:

> "*intelligibility* consists in the mechanism being portrayed in terms of a field's bottom out entities and activities."^

A mechanistic explanation renders a phenomenon intelligible by showing *how it is produced* by bottom entities and activities, i.e. by elucidating the explanandum (the termination conditions) by means of the set-up conditions and the intermediate entities and activities. This can be seen as in line with Bunge's recommendations for a proper explanation cited above[15], which advocate for the need to explicitly represent the intermediate dynamics (the mechanism) which stand between the initial conditions and the phenomenon to be explained, intermediate events which in the DN explanation are concealed under the expression of a law.

Description of a plausible mechanism for generating a phenomenon, according to MDC, is enough to render a phenomenon intelligible. They stress that intelligibility is a property of mechanistic explanation which is independent from the correctness of the proposed explanation: even in cases

[13] *Ibid.*, p. 16.
[14] *Ibid.*, p. 17.
[15] Section 10.

of *wrong* explanations, the fact that the wrong description shows how *plausibly* the mechanism works, is sufficient, according to MDC, to render the phenomenon intelligible.

Classical models of inter-theoretical reduction, in which properties and entities of the lower levels and the laws that govern them explain higher ones through the identification of corresponding terms in the two levels and the logical derivation of higher-level laws from the lower-level ones[16], cannot, according to MDC, accommodate the prevalent *multi-level* character of explanations in molecular biology and the neurosciences, where intelligibility stems from describing entities and activities at *multiple* levels: a multi-level description is required to put an entity or an activity in relation to its *context*, and this putting-in-context is in turn required, for allowing a proper understanding of the entities or of the activities.

Thus, it is not reduction *per se* to be important for intelligibility in molecular biology and neurosciences, but rather the *integration* of different levels into productive relations.

A comment is due about the fact that this MDC's view of intelligibility as based on a depiction of the mechanism in terms of its bottom level components seems to highlight the merits of a *reductive* kind of explanation: although it is specified that the bottom level is dependent on an arbitrary choice, the above claim by MDC seems to downplay the legitimacy of explanations which do not explain in terms of lower-level components: despite MDC's frequent appeal to the importance of multi-level integration, it seems that intelligibility is for them specifically tied to the description of the *bottom level* entities and activities. Explanations which remain more abstract, and do not go to the bottom, would lack intelligibility, according to their account. It could be probably said that such explanations remain simply mechanism schemata, without becoming proper mechanistic explanations. Due to MDC's ontic account of explanation, a proper explanation must in fact be constituted by the real-world, actual entities and activities that produce the explanandum and, as such, these entities and activities cannot be higher-level abstract entities[17]. Of course, a mechanistic explanation has to be based on the description of bottom-level parts by its very nature, but, as we will see in the next section, in the slightly different account of mechanistic explanation by Bechtel and Abrahamsen, a stronger emphasis can be posed on the possibility of rendering an explanation perspicuous through the consideration of *multiple* hierarchical levels, without tying intelligibility mainly to specification of the bottom level.

11.2 Becthel and Abrahamsen's view of mechanistic explanation

Like MDC, William Bechtel and Adele Abrahamsen (*BA* henceforth), in Bechtel & Abrahamsen (2005), a paper which they themselves plausibly recognize as their most mature characterization of mechanisms[18], highlight that the DN model of explanation cannot be applied to all scientific areas: in life sciences, most explanations do not appeal to laws, and even if a law could be provided, it would not explain the phenomenon, for that would simply amount to expressing generalizations on certain properties of a class of phenomena that would not satisfy the biologist's need for explanation. Instead, it appears that in biological literature the most frequent type of explanation is by *mechanism*.

[16] As in the model of intertheoretic reduction typical of the standard view exposed in Nagel (1961).

[17] Although it could be objected that in certain disciplines bottom level entities are still abstract entities. This is true for disciplines, as cognitive psychology, whose basic kinds are not physical but functional. It must be considered that bottom level choice is relative to the discipline and to the interests of the observer. This is something completely analogous, although on a more "ontic" side, to my idea of *preferred description*, explained in section 6.6.

[18] It is the seminal paper they still cite in their most recent publications. See for example Bechtel & Abrahamsen (2010).

BA cite the example of a "law" which states that, under specified conditions, in the last phase of cellular respiration the ratio of oxygen molecules consumed to ATP molecules produced does not exceed 1:3. They stress that this ceteris paribus generalization would not satisfy a biologist's need for explanation, for it would not answer the question about *why* this generalization holds: The law would itself require an explanation, an explanation that would be provided by the description of a mechanism.

Explaining by means of mechanisms amounts to *explaining why by explaining how.*

11.2.1 Main differences between BA and MDC accounts

Acknowledging that in recent times a number of other philosophers (most notably Machamer, Darden and Craver) have advanced proposals for the philosophical treatment of mechanistic explanation, BA note that these approaches partly overlap theirs but also differ in terminology, scope and emphasis. Specifically, they claim that their approach differs to that of MDC in the following features:

- MDC take a dualistic metaphysical approach based on entities and activities, while BA prefer talking of *parts* and *operations*, where *operation* is preferred over *activity* in order to draw attention on the *involvement of parts* in these operations (i.e. as when an enzyme operates on a substrate so as to catalyze changes in the substrate).
- MDC's characterization of mechanisms as productive of regular changes from set-up to termination conditions raises worries, on BA's part, that focus could be shifted mainly on linear processes in which the initial state is stable, while mechanisms, according to BA, when embedded in a larger mechanism network, are continuously responsive to contextual conditions. Moreover, it seems that MDC definition, with its stress on the path between an initial and a final condition, could overlook the fact that most interesting mechanisms are not linear chains of operations, but can comprise feedback loops which render their behavior complex.
- MDC's account of mechanistic explanation, as already hinted at, is *ontic*, while BA's approach is *epistemic*, a difference which has been hinted at, in sections 11.1.6 and 11.1.7.

11.2.2 BA's definition of mechanism

> A *mechanism* is a structure performing a function in virtue of its component parts, component operations, and their organization. The orchestrated functioning of the mechanism is responsible for one or more phenomena[19].

It is immediately clear that BA's approach to mechanisms is *functional.* We have already outlined[20] the main accounts of the notion of function and functional explanation, the most prominent of which is Robert Cummins' functional analysis. As will become clear in the following sections, BA's approach constitutes in a way an extension of Cummins' view of purely functional analysis.

BA stress that the *choice* of the component *parts* is dependent on which phenomenon the attention of the observer is posed on, and that an *operation* requires at least a *part*: an operation

[19] Bechtel & Abrahamsen (2005), p. 423.
[20] See section 9.

typically occurs between two parts, where one part initiates the operation, and can thus be considered the active part, and another part, the passive one, is changed by the operation.

A change can consists in an alteration of the position or a change of other properties of a part, and can even result in the transformation of a part of a certain kind into a part of another kind.

11.2.3 Hierarchical organization of mechanisms

Operations can be organized simply by *temporal sequence*, but biological mechanisms tend to exhibit more complex forms of organization. Mechanisms may involve *multiple levels of organization*:

- a mechanism is often part of a a higher-level, larger mechanism;
- with respect to a certain operation, different parts have different roles; mechanism components are spatially and temporally organized;
- operations are precisely timed, to achieve an orchestrated effect.

According to BA, it is crucial, over supplying a specification of the component parts and operations of a mechanism, also to specify *external circumstances* in which a mechanism operates. In general, this is another instance of the hierarchical encapsulation of mechanisms: external circumstances can be understood as larger overarching mechanisms, while components of a mechanism can be seen as mechanisms themselves. This shows that mechanistic explanation can be "recursive", and that the multilevel character of mechanistic explanations is quite essential.

This is in striking contrast with the the traditional model of reductionistic explanation, as proposed in Nagel (1961). In the traditional reductionistic view, there is recursive explanation, but the bottom level must, at least in principle, be able to explain all phenomena at all other levels, by means of reduction of all the high-level concepts and laws to low-level ones. By contrast, according to BA, in mechanistic explanation, mechanisms at different levels in the hierarchy account for *different* phenomena: an explanation at a given level is not *replaced* by a possible explanation at a lower level.

It is true that going down a level yields a form of reduction, but, seemingly, in BA's account this is simply a *mereological* reduction: a high-level part is replaced by its lower level component parts. While going up to a higher level provides a wider perspective, that puts the former level into the context of a larger mechanism which modulates the functioning of the lower one.

It appears, thus, that BA's position shows a significant antireductionistic nuance, albeit, judging by other works by the authors, it seems that they do not go as far as embracing the so-called "nonreductive physicalistic" position, which explicitly refuses reductive explanation which defer the explanatory work to lower levels[21]. It seems instead that BA simply advocate the necessity for *multilevel explanations*, which, according to them, are different from pure reductionistic explanations. I think it must be noted, though, that other philosophers and many scientists *do* consider the typical mechanistic explanation given in life sciences as a form of reduction. For example, Wimsatt (1976) claims that scientists in general consider the mechanistic explanations that they produce as paradigmatically reductionistic. The fact is, in its common use in the special sciences, reduction probably refers primarily to mereological reduction, while in philosophy of science the term has mostly referred to intertheoretical reduction, which is a completely different matter.

[21] That is, for example, the position held by Jerry Fodor.

11.2.4 Diagrams and simulation in mechanistic explanation

The DN model of explanation[22] assumes that explanations make use of propositional, linguistic representations, and resorts to logic as a tool for reasoning about these representations: according to the DN model, an explanation takes the form of a more or less formalized logical argument, in which the premises are a set of laws and some contextual initial condition, while the conclusion is an observational statement.

Consideration of cognitive aspects of the human subject is completely left out of such a model of explanation, and it is even actively refused, under the influence of the standard, post-neopositivistic theoretical view which sees psychology as completely antithetic to philosophy.

In an opposite way, BA are concerned with cognitive limitations, because they are committed to a conception of explanation which sees explaining as an eminently cognitive and communicative task. Along this lines, while DN explanation are purely linguistic devices, mechanistic explanation can make use of *graphical representation* in the form of figures and diagrams. BA claim that, moreover, when giving mechanistic explanations this use of diagrams is prevalent over the simply verbal description.

BA support this conclusion by referencing cases in contemporary biological literature and history of science. According to them, verbal labels and captions associated with figures in that kind of literature usually provide only a secondary commentary, while the core of the explanatory work is performed by the *graphical* representation: specific of a diagram is its capacity to make use of space, color and shape to convey information, preserving the spatial layout and organization of the described mechanism. Even when information about the specific spatial layout of the mechanism is absent or irrelevant, spatial dispositions of blocks representing operations in the diagram can be used to separate or relate them conceptually. A diagram can also, by making use of one of the spatial dimensions, or by means of arrows or other graphic expedients, inform about the *temporal* orders of operations.

Against the DN model of explanation, BA argue that, if diagrams convey information not easily conveyed explicitly by linguistic representations, deductive inference would not capture the real reasoning required for understanding how a mechanism gives rise to the observed phenomenon, and so that the DN model alone is insufficient: only the apprehension of the mechanism would yield a satisfactory explanation, an this is eased by the use of diagrams in mechanistic explanations. This, according to BA, matches well with results in cognitive psychology showing that for efficient reasoning it is essential to coordinate modes of representation and procedures of inference. These considerations on cognitive efficiency are again in striking contrasts with the theoretical position supporting the DN model which, as we have seen in section 8.1, completely neglects the cognitive aspects related to explanation and its understanding.

According to BA, *simulation* is a privileged mode of reasoning about a mechanism, that allows to compensate the staticity of diagrams and to reason about temporal features in a mechanism.

There are, according to BA, two kinds of simulations:

- *mental animation* can be carried out by a human subject as an activity consisting in inferring the subsequent states of parts of the system starting from the system's present state. However, mental animation is a cognitive activity which shows itself as only partially isomorphic to the operation of the physical system, due to some cognitive limitations in evaluating multiple operations occurring simultaneously.

[22] See section 8.

- Simulation by non purely mental models, which can overcome cognitive limitations:
  - scale models;
  - mathematical models;
  - computational models: this is what BA call, in a more recent paper[23], *Dynamic Mechanistic Explanation* (*DME*).

To sum up, in BA's words

> Representation and inference in mechanistic explanation is quite different from representation and inference in nomological explanation. While it is possible to give a linguistic description of a mechanism, the linguistic account is not privileged. Frequently diagrams provide a preferred representation of a mechanism. Inference involves a determination of how a mechanism behaves, and this is typically not achieved via logical inference but by simulating the activity of a mechanism, either by animating a diagram or by creating mental, computational, or scale model simulations[24].

11.2.5 Discovering mechanisms: decomposition and localization

Models of mechanistic explanation seem more fit than the traditional DN model of explanation for taking into account the phase of scientific *discovery*. In the received view in philosophy of science, discovery is not investigated at all: according to Reichenbach's traditional distinction between the context of discovery and the context of justification, justification can be studied philosophically because it is a logical problem, while discovery is a psychological process and as such not interesting at all for a neopositivistically-inspired antipsychologistic philosophy of science.

For BA, discovery of mechanisms is on the contrary a natural object of study for philosophy, since, according to them, the concept itself of mechanism implies the idea of identifying the working parts and operations and the organization of the system which generates a phenomenon. This is an implicit view of the task of discovering a mechanism, a task which requires two phases: *decomposition* and *localization*.

Decomposition consists in taking the mechanism apart, physically or conceptually. Decomposition can be *functional* or *structural*. The two types are usually undertaken by different researches in different fields, while integration of functional and structural aspects into a complete account usually comes later:

- *functional decomposition* consists in decomposing the system into component *operations*. This is performed by discovering which *lower-level* operations contribute to the overall functioning of the mechanism under scrutiny. Often, operations act on, and transform, *substrates*. So, a successful functional decomposition will identify the operations and their corresponding substrates, that is the passive parts, while structural decomposition (next point below) will identify the active parts performing the operations.
- *Structural decomposition* consists in decomposing the system into component *parts*. What BA highlight is that structural decompositions should identify specifically the *active* parts, that is, the parts that actually perform the operations already found by the functional

[23] Bechtel & Abrahamsen (2010).
[24] Bechtel & Abrahamsen (2005), p. 432.

> decomposition (the parts which Carl Craver calls *working parts*). Only the working parts count in a correct structural decomposition, as it is obvious that there is an infinite number of other possible structural decompositions that don't pinpoint working parts (like, for example, cutting the system arbitrarily into cubes).

Nevertheless, according to BA, it is possible to perform a structural decomposition independently from a functional one, identifying what are *likely* working parts on the basis of other non-functional considerations, and refining that partitioning to gradually *converge* on one that gives working parts. This structural decomposition independent from the functional one occurs especially at the beginning of a research.

Afterwards, having performed the two decompositions above, a phase of *localization* is needed. Localization amounts precisely to the *linking of operation with parts*. This is the stage that concludes mechanistic explanation.

As said, functional and structural localization can, and usually are, carried out separately by different researchers. A strong corroboration of the correctness of the decomposition done comes precisely from the independence of the two types of decompositions: "linking a component operation with an independently identified component part provides evidence that both really figure in the mechanism"[25], while failure to do so can cast doubts on the adequacy of the decomposition done.

It must be noted that, although structural/functional decomposition duality seems in some way to mirror MDC's entity/activity dualism (structural decomposition identifies entities, functional decomposition identifies activities), the overlapping is not complete: functional decomposition, according to BA, also identifies some parts, the passive substrates; however these passive parts do not fulfill MDC's vision of parts as mostly active promoters of activities.

Different experimental methods are involved in decomposition: decomposition in practice involves a variety of experimental procedures. BA highlight the importance of two main methods:

- Inhibition of a component to observe its effect on the global behavior of the mechanism[26];
- recording of internal states of the mechanism in a variety of conditions (for example, when neuroimaging techniques such as functional MRI are applied to the study of the brain).

A third crucial factor for mechanistic explanation is the discovery of the mechanism's *organization*. Often this organization is nonlinear, and this usually requires relying on computational modeling to highlight the often surprising complex behavior a kind of non-linear organization can bring about, such as self organizing behavior[27]. Much philosophical work, according to BA, is yet to be carried out regarding this area of mechanistic explanation.

11.2.6 Testing mechanistic explanations

In the DN model, testing a hypothesis entails making predictions by deducing consequences of the known laws, and experimentally testing if those predictions hold. This method presents

[25] *Ibid.*, p. 434.

[26] This method amounts to a form of intervention on a variable of the observed system. According to some accounts of causality, like that of James Woodward, intervention is the basic method to detect causal links. I will argue in section 6.9 that modularity of the system is necessary for this kind of method to be applied.

[27] I discussed self-organizing behavior in section 5.2.3.

some well known shortcomings, namely underdetermination of theory by data and confirmation holism.

The two problems are related. The first, that of underdetermination of theories, can occur in two forms. One occurs when a theory is confirmed by experimentally observing the actual happening of some of its consequences. Since the same set of consequences can logically descend from an infinite number of different sets of premises, the given theory is actually been confirmed together with the infinite other theories which would give rise to the same consequences tested, so it is impossible to tell which of them is the *right* theory.

The other form of underdetermination of theories, also known as *credit assignment*, stems from the observation made by Pierre Duhem around 1914 that, while testing a scientific hypothesis, a negative experimental result leaves undetermined which hypothesis must thus be rejected as the culprit. This happens because, even if apparently a single hypothesis has been put to test, actually an unbounded number of auxiliary hypothesis regarding initial conditions and background beliefs have been implicitly checked together with the main one, and being impossible to explicit them all, it's impossible to decide which is the one to reject[28].

BA claim that, although itself affected by such problems, the testing of mechanistic explanations mitigates them, because testing of a mechanism is more informative than testing of a hypothesis under the DN model: due to the fact that the experimenter, when setting out to test a model of a mechanism, typically focuses on *specific* components and not on the mechanism *as a whole*, the experimental failings of his expectations are *diagnostic*, for such experimental results target specific components and operations, and are certainly those parts to be in need of revision as a consequence of such experimental negative verdicts: the underdetermination is canceled by the fact that the experimenter already knows, at least in a general way, which specific part or parts of the mechanism, in case of failure, have been responsible for the failure.

In BA's account, the testing of mechanistic explanation, rather than simply consisting in postulating a complete mechanism and putting it to test, typically begins with an oversimplified account of the mechanism, a *sketch*, in which only a few components and aspects of their organization are specified, a mechanism sketch which is to be repeatedly revised and filled in, over time.

11.2.7 Generalizing without laws

An obvious expectation in science is that explanations and theories can be generalized beyond the specific cases around which they have been initially constructed. Generalization is achieved automatically in the DN model, because laws are represented as universal conditional statements. In contrast, mechanism models can be highly specific, and a generalization of a mechanistic explanation seems less likely, for it is strongly bound to a specific context. Nevertheless, BA think that some generalization is possible. To account for it, they employ a concept of *similarity* modeled after certain well-known researches in cognitive psychology[29], in order to apply it to similarity between mechanistic models. The idea is this: often, instances of objects falling under an ordinary-language concept, rather than sharing some common distinctive properties, present a varying degree of similarity toward one *prototypical* item, sharing instead only a "family

[28] Willard van Orman Quine famously expanded this thesis in his seminal Quine (1951) paper, by claiming that the totality of scientific and logico-mathematical knowledge is a web of interconnected statements which can be confirmed or disconfirmed only together as a whole. This conclusion by Quine is known as the problem of *confirmation holism*.

[29] This concept of similarity stems from the famous works on the psychology of membership recognition in categories conducted by Eleanor Rosch and colleagues since the '70s.

resemblance" à la Wittgenstein between them. In an analogous way, different mechanisms can exhibit similarities of varying degree between them. In the words of BA:

> Different mechanisms may exhibit similarity relations to each other without being exactly the same. For example, mechanisms of protein synthesis may be similar in different organisms–or different cell types in the same organism–without being identical. Certain memory encoding mechanisms,to take another example, may be similar across some delimited range of species.[30]

According to BA, generalization is then facilitated by the individuation, in certain areas of research, of *model systems*, which are specific exemplar of mechanisms[31] on which, typically, research focuses at the outset of the investigation of an area, often for reasons not necessarily tied to their being typical systems, but more often tied to ease of study. These model systems, after research has consolidated, usually begin to be taken as a common point from which to try to generalize explanation to other similar mechanisms, by gradually modifying the model system's explanation: it is in this way, completely different from the method of generalization typical of the DN model, that generalization is brought about in mechanistic explanation.

11.3 Functional analysis and mechanistic explanation

> To begin with functional decomposition, here the strategy is to start with the overall functioning or behavior of the mechanism and figure out what lower-level operations contribute to achieving it. These operations are characterized differently in different domains, but often involve transformations to some substrate. The biochemical system that performs metabolism in cells, for example, catabolizes glucose to carbon dioxide and water. The component operations are then characterized in terms of individual chemical reactions on a series of substrates.[32]

Abstracting from the material nature of the operations and substrates cited in the above example, we could conceive functional decomposition as the analysis of which are the operations (that is, *capacities*, in Cummins' terms, or *dispositions* when referred to computational systems) that contribute, in an organized way, to the functioning of the whole system. Bechtel and Abrahamsen go on:

> As another example, in the domain of information processing systems, representations play roles comparable to substrate and product, and information processing activities are the operations (e.g., moving or altering representations).[33]

When taking into consideration *computational* operations which act on substrates, it is easy to see that in this case, according to the view of computationally capable, or *digital* systems which I proposed in section 5.1.1, the substrates are the symbol tokens (or input representations, to use an expression more typical of computational cognitive science), and the operations are the operations performed on these tokens by the computational machine.

[30] Bechtel & Abrahamsen (2005), p. 438.
[31] Such as, for example, the giant squid axon, for the study of neural transmission, where the large size of the axon facilitated experimentation.
[32] Bechtel & Abrahamsen (2005), p. 433.
[33] *Ibid.*

The peculiarity of mechanistic explanations with respect to Cummins' functional decomposition is that, *when applied to real-world systems*, mechanistic explanation does not limit itself to functional decomposition: it must also perform *structural* decomposition, which seeks identification of the structural components[34] of the system which are supposed to perform the functions described by functional decomposition.

The third phase is that of linking functional operations with structural parts, and this is called *localization*. This is often necessary to corroborate the two former types of decomposition, which in actual scientific research are in many cases carried on separately, at least at the beginning of a research paradigm.

But when trying to mechanistically explain a computational system (or, for that matter, a high-level *psychological* or any other kind of abstract system, like an electronic circuit[35]), only the functional characterization is possible, and in this case it seems there is coincidence between Cummins' functional analysis and the mechanistic conception of explanation[36].

It is true, though, that even functional decomposition as understood by BA does not seem to coincide completely with the classic role functionalism which is usually adopted in "special" sciences like cognitive psychology: BA's flavor of functionalism seems more tied to the *realizers* of the functions, at least for what concerns the parts. Of course this can be further debated, given that bottoming out, and so the identification of the lowest level, is contingent on the researcher's interests, and as such could bring to the identification of the lowest level with a quite "high", quite abstract level. But even when so, it seem this kind of functionalism is of a more "reductionistic" way, with respect to role functionalism, because it expects a low (and possibly very low) degree of multiple realizability, given that parts performing operations, that is the active parts, are *expected* to be *uniquely* identified during the phase of structural decomposition. This is probably what makes BA's model of explanation a proper "mechanistic" one.

[34] the *working parts*, in the words of Bechtel & Abrahamsen (2005).

[35] Although it could sound strange, an electronic circuit, as usually conceived by electrical engineering, is a *functional* mechanism, not a physical one: basic building blocks are identified, each with a characteristic input-output (digital or analog) function. The actual *physics* underlying electronics is taken in consideration only when designing a *single* elementary component. This view is explicitly shared by Cummins (1975). I further discuss the question in relation to modularity in section 7.2.

[36] Albeit mechanisms, as conceived by certain authors, namely Carl Craver, seem less of an epistemic type and more of an ontological nature that in the conception of William Bechtel and colleagues.

Chapter 12

Philippe Huneman's topological kind of explanation

Philippe Huneman has recently proposed (in Huneman 2010 and Huneman 2015) to recognize in scientific literature, especially in certain special sciences, a type of explanation, *topological explanation*, which differs from the types of explanation classically taken into consideration.

Topological explanation is a non-mechanistic type of explanation based on topological properties of certain abstract descriptions of a system. Specifically, Huneman describes topological explanation as follows:

> a kind of explanation that abstracts away from causal relations and interactions in a system, in order to pick up some sort of "topological" properties of that system and draw from those properties mathematical consequences that explain the features of the system they target[1].

An idea drawn from topology in mathematics, topological properties are, intuitively, those properties of a system of interrelated parts regarding, so to speak, the system's "shape", that are invariant under any possible continuous deformation of the system. What I called here "shape" must not necessarily be material, but can pertain to an abstract, mathematical space. In the terminology I adopt in this work[2], It could be said that topological properties do not concern a system, but certain *descriptions* of the system. Topological explanation would consist in explaining features of the system by appealing not to causal events between its parts, like mechanical explanation would do[3], but by pointing to some topological features of the system's representation in this abstract space, features which can be called also "structural" in the sense of pertaining to mathematical structures. Accordingly, topological explanations could be also called "structural explanations" (Huneman 2015).

In the words of the proponent, these properties are topological in a very broad sense. Huneman cites many examples of types of topological properties, most of which are properties of quite abstract representations: geometrical spaces, phase spaces, networks, representing parts of a system, behaviors of a system, capacities, or in general features correlated to the system.

[1] Huneman (2010), p. 214.

[2] As explained in section 6.6.

[3] As Huneman highlights, the relationship between mechanistic and topological explanation is not simple. See Huneman (2010), Huneman (2015) and later on in this section.

A *Topological* property is topological in that it specifies the invariance of the system with respect to certain classes of continuous transformations. Topological properties can be also properties of networks, like the mean degree of connectivity, the fact of being scale-free, or small world, or being modular or non-modular[4]. A network possessing such a kind of property does not lose or acquire such a property as a consequence of certain types of transformations (for example, a modular network does not change its modular structure if some nodes get simply displaced without changing their connections, because modular structure is not dependent on the spatial position of its nodes, but on how they are *connected*).

Huneman (2010) proposes as example of topological explanation a study by Richard Lensky and colleagues (reported by Wilke, Wang, Ofria, Lenski, & Adami 2001), in which two populations of bacteria are considered, populations with very different internal distribution of fitness: population A, which comprises a limited percentage of individuals with high fitness, and population B, composed mostly of individuals with more or less the same fitness, albeit lower than the maximum fitness of individuals of A: see fig. 12.1.

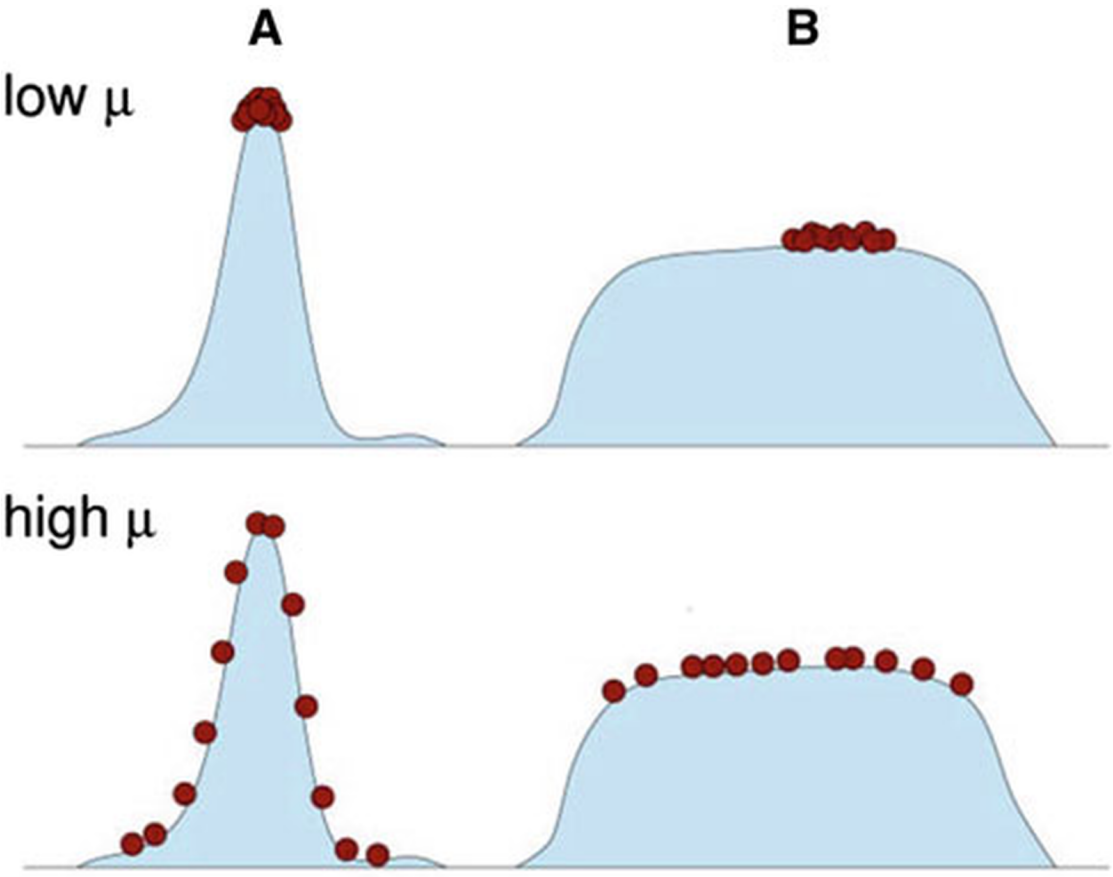

Figure 12.1: fitness distributions of two populations A and B of bacteria, as reported in Wilke et al. (2001) (image taken from Huneman 2010, originally belonging to Wilke et al. 2001).

When populations A and B are are put in competition in an environment inducing high mutation

[4] See section 3.

rate (lower graph in fig. 12.1), in an apparently counterintuitive way population B tends to prevail. This is explainable in a *topological* way by merely mentioning the *shape* of the two *curves* representing population fitness, without the need to consider any empirical or other structural property of the system: high probability of mutation, shifting individuals along the X axis, makes highly fit individuals of population A easily "fall" from the steep peak of high fitness, while for members of population B, shift along the X axis lets them remain more or less on the same fitness "plateau", where fitness is on average higher than the level of fitness on the hillside of the high peak of population A.

As it appears clearly from the examples, this kind of explanation appeals to certain features of certain abstract representations of a system, features which are not causal or empirical, but purely structural and topological, in order to explain certain other properties of the system so represented.

As said, topological features, as understood in this kind of explanation, can range from the shape of curves, as in the examples above, to other general non-causal properties of the representation, such as the distribution of the density of its parts, to the fact that the representation shows some form of modularity or not. On the basis of the presence or absence of similar, purely structural, topological properties, certain dynamical properties of a network can be explained, such as, for example, the fact that perturbations spread indiscriminately or not: spread of perturbation is greater in small-world networks[5] with respect to networks which do not have this property.

What is interesting in topological explanations, as Huneman (2015) highlights, is that in such a kind of explanation it is the topological facts to be explanatory, and not the processes of the system which instantiate these topological properties. From a mechanistic standpoint this relation between processes and topological properties is peculiar: it is precisely for reasons due to certain topological, mathematical features of the system, that certain mechanistic processes (mechanistic in the sense of being dynamical interactions of parts) occur in a certain way. While in science usually mathematics is used in a descriptive way, as a mean to obtain economical descriptions of phenomena, in this case the role of mathematical facts is *explanatory*. In the words of Philippe Huneman:

> Yet what happens with structural explanations in general is that mathematics are rather explanatory: mathematical properties are the reason why some facts happen in nature as outcomes or features of the activities of the hypothesised entities. The mathematics not only represent the mechanisms' settings and functioning, they also explain why a set of mechanisms is constrained in a specific way, necessarily yielding a range of outcomes that possess a given property. This includes a weak, epistemic claim, stating that without the mathematics we would never know that some outcomes are possible, some impossible and some necessary. But it also means the stronger, metaphysical, claim that in some cases the reason why some systems are displaying a constant or regular behavior of some sort (e.g., with a specific steady state, a typical outcome, or inversely, an absence of some particular outcome etc.) is a mathematical—in the present context, topological—fact: such fact grounds counterfactual dependences between sets of possible initial states and sets of end-states. Correlatively, this implies that what makes a difference regarding several sets of systems—for instance, stable and instable communities in the ecological example above—is a topological property; such property is instantiated by all mechanisms in the considered systems, but it's only in virtue of the fact that they instantiate this property that those are themselves explanatory of anything. Topology being about invariance through a class of continuous transformations, topological explanations are explanations in which the possibility and impossibility of some

[5] See section 3.1.2.

> systems to reach some sets of final states or behaviors is explained by the topological fact which they instantiate, specifying which states are topologically equivalent and which are not, hence are not likely to be reached by the system.[6]

It appears clearly the the kind of explanation discussed above can not actually be assimilated to any of the classical types of explanation, some of which (deductive-nomological, mechanistic and functional) we have discussed in the previous sections. It seems thus that topological explanation is a legitimate kind of scientific explanation. Surely, topological explanation is not germane to causal types of explanation, like explanation as understood in the "new mechanistic" sense. Nevertheless, Huneman stresses that these two kinds of explanation can be interrelated in non simple ways. Topologies can in certain cases pose constraints on mechanisms: for example, if a dynamics is implemented on a network, the network's topological, structural features will constrain the mechanisms which take place on this structure (a question similar to those which I have touched upon in section 6.2.1).

In certain cases, mechanistic and topological explanations condition one another. I would like to propose this example: in a self-organizing CA[7], the CA starts with a nearly blank lattice and its intrinsic topology (the connectivity structure) given by the CA rule and the neighborhood which it takes into consideration. By the time its dynamical process begins to self-organize, however, and some "frozen" structures in its lattice appear, the fact that some of this structures act as "walls" impenetrable to external perturbations, provides a new topology to the system, a topology which now "channels" the activity of the non-frozen parts of the lattice: moving particles can now connect distant parts of the lattice only by flowing through the interstices between the frozen structures. In this case, the mechanism has changed the high-level topological structure of the system, and in turn this change in topology constrains the successive dynamics of the mechanism.[8]

[6] Huneman (2015), pp. 5-6.
[7] See sections 5.2.2 and 5.2.3.
[8] I refer to Huneman (2015) for a more thorough treatment of the relationship between topological and mechanistic explanation.

Part IV

Antimodularity

Preamble

This major section treats the question of *antimodularity*, a notion which I propose in order to capture the idea of our inability, in certain circumstances, to describe a system as modular. In a way, antimodularity is thus the opposite of modularity, but it is not simply its negation: antimodularity, as we will see, can be understood in different pragmatic and epistemic ways, tied to questions of computation, and its relation with modularity is not a relation of simple opposition.

I first proceed, in chapter 13, to give an informal definition of antimodularity.

In chapter 14, I discuss the consequences of the occurrence of antimodularity on the possibility of scientific explanation of systems affected by this property.

In section 14.6 I advance the notion of *explanatory emergence*, based on a generalization of the reasons generating antimodularity to other circumstances affecting cases of algorithmic detection of features of complex systems.

Finally, in chapter 15 I briefly discuss the likelyhood of the actual occurrence of antimodularity in real-world science.

Chapter 13

The notion of antimodularity

13.1 Problems with the detection of modularity

As we have seen in several of the preceding sections, detection of structural and functional high-level modularity is essential for the study and modelization of complex systems. In section 6.9 we have discussed the need to outline a modular structure in an observed phenomenon or process, just in order to model it and progressively refine its models.

Usually this detection of the modularity of a phenomenon is carried out by means of the experimental method typical of science, which, albeit based on the same principles in general, gets realized in multifarious ways according to each particular discipline and the particular research paradigm. The experimental method proceeds in a piecemeal fashion, sometimes realizing vast series of experiments in which the initial conditions vary only slightly, in order to separate the effects of what, in the preferred description of the discipline (the description level typical of a given dicipline, as proposed in section 6.6.9), are single entities or variables.

Typically, in the case of certain special sciences, especially biology-derived disciplines, the goal of the experimental study is that of decomposing a system into its elementary, causally relevant parts, in order to proceed to a mechanistic explanation[1] of the system. When dealing with very complex, or very large systems, decomposition can become a daunting task: decomposition proceeds, as said, piecemeal, by means of several experiments each of which, in the most lucky cases, gets at most to distinguish a handful of parts constituting the system. But, usually, especially for functional decomposition, even more than one experiment is needed in order to assess the function of a single part. Some of these research endeavors go on for decades in hundreds or thousands of teams around the world before the entire line of research is able to produce a sufficiently detailed functional and even structural description of the observed system. It will suffice to think of research in the neurosciences, for which even the ascertaining of the anatomy of the central nervous systems has required decades.

There are cases, and these cases have become more and more numerous in recent times, where the lowest-level modularity, the preferred description of the system, has been at least in most part provided, but we lack a clear model of its higher-level functional modularity. As we have seen in section III, an intrinsic requirement of certain types of scientific explanation, namely of mechanistic and functional explanations, is that the system to be explained be considered from the standpoint of multiple hierarchical levels of description, and that the explanation take into

[1] See section 10.

account the relation between these levels. So, it would be important to be able to produce of a given system not only a single modular description, but an entire spectrum of descriptions at different levels of abstraction, that is, what in section 6.6 I called modular redescriptions, or macromodular descriptions of the systems (actually, of its preferred description, which is already given).

In complex or very large systems (large as described by their lowest level, preferred description), this detection of hierarchical modularity is usually unfeasible with non-automatic means, due to the sheer number of parts and relationships between them to take into consideration. Besides, process modularity would be unfeasible in non-linear systems even if their number of parts were small, because non linearity hinders the possibility of prediction of the system by means of analytically solvable equations.

To compensate these difficulties, as we have seen in several of the preceding sections, many algorithmic methods have been devised to detect structural and dynamical modularity in different types of complex systems.

However, most of the known algorithms for modularity detection can end up being an insufficient solution, in case of very large complex systems: the problem with many of these methods resides in their computational cost, which can result so high as to render an unfeasible task the detection of hierarchical modularity in complex system of a certain size. Some systems of scientific interest can be affected by this impracticality of their macromodular description, at least when a sufficiently accurate macromodular description is needed.

The computational cost of most algorithms for modularity detection derives indirectly from the computational hardness of optimal modularity detection in general and in specific applications. First, as we have seen in section 2.2.1.2, there is the *NP-hardness* of aggregability, as proved by Winker (1992). This means that detection of modularity in the dynamical model of a system is a computationally unworkable task, and so it is not to be expected, in general, that dynamical modularity can be found by means of a universal method which can guarantee to produce in feasible time a plausible aggregate, simplified version of the model. Not even an approximate one, as proved in Kreinovich & Shpak (2008).

There is also, in the somewhat more specific field of networks, as explained in section 3.2.1.2, the NP-completeness of optimal community structure detection, proved by Brandes et al. (2008). This result means that the task of producing the best modular[2] hierarchical description of a given network, the one which represents more faithfully a genuinely robust modular structure actually present in the network (and not an apparent modularity due to random fluctuations in the connectivity of nodes), is too computationally intensive to be feasible. For this reason, it is not in general to be expected that the optimal modular description of a network can be found, at least by algorithmic methods, when the network comprises more than a certain number of nodes.

The computational hardness results cited above have spurred the search for algorithms which approximate the search for the optimal detection of modularity. Actually, most of the algorithms for modularity detection in networks described in section 3 approximate the search for the best possible modular redescription, or for the best possible hierarchical description of a given network. These algorithm produce more or less plausible modular redescriptions which are quite certainly not the optimal ones, optimal in the sense of being the most faithful to the real actual modular structure (if any) imposed on the network by the distribution of the connectivity between its nodes.

[2] Modularity in the sense of community structure.

Even if approximate, most of the algorithms for network modularity detection are, as discussed in section 3.3.1, still quite computationally intensive: even if their time complexity is polynomial, the degree of the polynomial can be quite high, for example $O(n^3)$, or even $O(n^4)$ (with n the number of nodes in the network), and so practical applicability of these algorithms can result difficult on networks with a sufficiently high number of nodes. There are faster algorithms, but in general all these methods suffer from a speed/accuracy trade-off[3], which makes the faster algorithms unacceptably inaccurate for certain applications.

In networks, there are also forms of functional modularity, like modularity based on network motifs and descriptions based on network roles, which can be quite difficult to detect as well, as highlighted in sections 3.3.2 and 3.3.3. These two forms of modular description, especially the role-based one, are extremely effective in giving a perspicuous representation of the functional organization of a network, a kind of representation especially useful in scientific applications of network science, like in the case of biological networks.

All the above results of computational hardness or at least of high computational expensiveness affecting the algorithmic detection of modularity entail that a sufficiently faithful detection of structural and functional modularity in certain large complex systems is not guaranteed at all, at least when search for modularity is conducted in absence of further prior information about the system which can usefully constrain this search.

The problem is that, in certain cases, even in scientific research, especially in nascent fields, the researcher lacks such prior information, and must thus recur to purely algorithmic modularity detection. Even when some constraint can be imposed on the search for modularity, in certain occasions the sheer size of the system under observation can render the task computationally too expensive: this can be especially true for extremely large systems of biological interest, like genetic or metabolic networks.

Modularity detection can thus, in sufficiently large systems, be actually prevented by problems of computational cost, if not computational hardness. Whenever this occurs, a system can be *pragmatically* considered antimodular, even if in principle it possesses some modularity, because to all intents and purposes we are practically unable to automatically detect its modularity: its preferred description appears devoid, at least as far as we can know, of any valid high-level modularity: that is, there is no known macromodular redescription of it whose parts are endowed with a sufficient degree of robustness.

Limitations of modularity detection due to computational complexity are not *in principle* limitations: it is true that, in principle, we could obtain the optimal modular description of a network. We should not downplay, however, these aspects of computational hardness as *merely* pragmatic. Especially in cases in which the computational hardness of an algorithm has been established by formally proving it, this pragmatic hindrance becomes something more compelling, assuming the cogency of a logical law: the algorithm for optimization of detected modularity has been proved computationally hard (NP-complete, actually) and there cannot be any hope of rendering it more computationally feasible[4]. Computational hardness in the form of NP-hardness or NP-completeness is a completely *objective* fact which entails a pragmatical infeasibility of the performing of certain computations on certain inputs. Although pragmatical, such a kind of infeasibility is far from being negligible. No matter how we try to improve a computationally hard algorithm, or we augment the power of the system on which it runs: its execution time

[3] See section 3.3.1.2.

[4] At least, given that no proof of $P = NP$ has ever been devised, and that, according to most computer scientists, this possibility is extremely unlikely. See section 17.4.3.1 of the Appendix.

will, at least in certain cases, always overcome any possible improvement in speed[5]. Optimal modularity detection can be probably *approximated* in more reasonable times, but the trade off between speed and accuracy, which is to be expected in approximated algorithms for modularity detection, could, associated with the high number of parts of some complex systems, render the detected modularity excessively approximate or, conversely, make the detection time of a sufficiently precise modular description excessively high when we require a certain degree of precision, even if the approximated algorithm employed is not, from a formal point of view, computationally hard. So, even if this limitation of optimal modular detection is only pragmatic, it affects optimal modularity detection and the scientific practices tied to it in an essentially objective, unavoidable way: for all practical purposes, there is little difference between a task which is in principle uncomputable in general and one which is computable, but is in general computationally hard.

13.2 A definition of antimodularity

In the wake of the results of computational unfeasibility of modularity detection decribed in the former section, I propose to introduce a property, *antimodularity*, which captures some of these difficulties.[6]

Antimodularity is a property of complex systems which I propose to define as the general impossibility of obtaining, by means of algorithmic detection of modularity, a sufficiently valid and useful hierarchical modular description of a system.

There is antimodularity when the most faithful hierarchical description of a system, obtained by algorithmic means which are practically feasible, is nevertheless invalid from a dynamical standpoint or, if somehow valid, too approximate to be useful anyway with respect to the purposes and aims of the researcher or the requirements of the scientific discipline in question.

When a system shows antimodularity, its only possible hierarchical description can comprise only the two *trivial* hierarchical levels, the highest and the lowest one, and the system presents a *flat hierarchy*[7]. In other words, *antimodular systems* are systems which can be described in a modular way and explained by decomposition at only *one* level of description, the level of their elementary, finer parts, which usually is the level of their preferred description.

As defined above, antimodularity stems from the algorithmic impossibility of detecting modularity in a system. But, this impossibility can correspond to *two* circumstances: the actual absence of any modularity in the system or only the pragmatical impossibility to detect a modularity which is indeed present. It would be useful to distinguish these two circumstances. I propose the following terminology:

i. *Intrinsic Antimodularity* (*IA*, henceforth): this is the condition in which antimodularity obtains because the system is actually, objectively, devoid of any robust high-level modular structure, given its preferred description. In other words, antimodularity is intrinsic to the

[5] We are talking here, in the case of exponential growth of the time required for the computation, of times needed to bring to completion a computation on certain data which are on the scale of the age of the universe or even higher order of magnitude. No possible technological improvement could speed such algorithms up in such a way as to render them computable in a reasonable time.

[6] A terminological disclaimer: in what follows I will often use the terms "modularity" and "antimodularity" as referred to *systems*, when in actuality, inside a theoretical framework like the one proposed in section 6.6, we are talking not of systems, but of descriptions of systems, and not of modular descriptions but of modular redescriptions of descriptions. Usually, when talking of a "system" I will talk of its preferred description.

[7] See section 6.6.11.

given preferred description, no matter how accurate the algorithm for its detection is.
Systems intrinsically antimodular can be for example uniformly connected systems, like regular networks (described in section 3.1.1), where there is not enough variability in the connection density in different parts of the network to constitute subnetworks which are more connected internally than towards the external context, that is to constitute modules. In such networks modularity is objectively absent. Objective absence of process modularity can occur in dynamical systems whose evolution in time is so chaotic that it does not allow for the formation of sufficiently robust, non-evanescent structures which can legitimatley be considered modules.
Given the above definition of antimodularity, it is evident that intrinsic antimodularity is a case of (generic) antimodularity.

ii. *Antimodularity* (*AM*, henceforth): this is generic antimodularity, as defined above in this section: it derives, regardless of the actual absence of modularity in the system's preferred description (point *i*), from the excessive computational cost of the task of detecting high-level modularity in such a preferred description. It usually arises due to the too high number of parts in the system's preferred description, a circumstance which makes the employed modularity-detection algorithm end up being computationally too expensive to be brought to completion, either because it is computationally hard, or, although formally not hard, because it is too computationally expensive to be brought to an end anyway. I would call this circumstance simply *antimodularity*. We could call it also *epistemic antimodularity* (or *EA*), epistemic because it stems from our *ignorance* of the actual presence of modularity in the system, but I will make use of this distinction between antimodularity and epistemic antimodularity only were necessary.

It is very important to stress here that antimodularity is, like modularity, *relative* to the *choice* of a metric: this is the metric according to which modularity of the preferred description is assessed. The metric is usually a property of the preferred description's entities, or a relationship between them, or, as described in section 6.6.11, in general any computable abstraction which detects modules of some kind in the preferred description. Antimodularity is relative to this metric in that it manifests itself as the *failure* of modularity detection conducted according to the metric. Like in the case of modularity, changing this metric, antimodularity can show up or disappear. It is as well evident that, given that often preferred descriptions naturally suggest a possible set of plausible metrics for modularity detection, then antimodularity, like modularity, is relative to the choice of the preferred description. In scientific research, however, the choice of the preferred description is not very free, being usually more or less constrained by the scientific discipline in question.

The reason behind the above distinction between antimodularity and intrinsic antimodularity is that, while antimodularity could in some case be eliminated by improving the algorithm for modularity detection algorithm, *intrinsic* antimodularity cannot be so eliminated, for it is not a consequence of the inaccuracy or computational cost of the algorithmic method used, but an objective feature of (the preferred description of) a system, in which modularity is objectively absent. Moreover, *intrinsic* antimodularity can have different consequences than generic anti-modularity for what concerns certain types of scientific explanation, as we will see further on in section 14. Intrinsic antimodularity can have a different effect than generic antimodularity on our capacity to predict the general properties of a phenomenon's dynamics: for example, certain intrinsically antimodular dynamical systems should manifest a highly chaotic behavior, due to the extreme facility of transmission of perturbations inside them, which in turn is due to their high uniform connectivity. In these cases, knowing that the system, which appears antimodular,

is actually *intrinsically antimodular* can allow the observer, in presence of other informations about the non-linearity of connections, to predict at least this general feature of its dynamical behavior.

It would then be auspicable to have an algorithmic method to discriminate between intrinsically antimodular systems and systems which appear antimodular due to the computational cost of modularity detection on them. It is plausible to think that, as it is often due to the statistical distribution of the connections between entities of a system's preferred description, intrinsic antimodularity should be easier to detect than antimodularity, for this general feature can be revealed on a statistical survey of certain statistical parameters of the systems. But, not every case of intrinsic antimodularity can be due to this evident uniformity of connections: especially dynamical antimodularity can arise at a higher level of description also in systems which apparently sport modularity at the preferred description, or even at higher levels: for example, a computer program implementing the simulation of a cellular automaton can be perfectly modular in its structure composed of subroutines, while the dynamics of the CA it simulates can be completely chaotic, when the chosen CA is a member of class III[8].

So, it seems that in general it is not always simple to tell modularity and intrinsic antimodularity apart.

It can nevertheless be feasible in some cases, especially for certain types of systems, to have some hint on the actual presence or absence of modularity in the system, especially structural modularity, without having to try to detect modularity with a dedicated algorithm (should we recur to an algorithm for modularity detection, we could incur in antimodularity due to the algorithm's computational cost). This can happen when the system (as is the case with regularly hyperconncted systems) has some statistical structural property, detectable at a low computational cost, which is however usually correlated with the presence of modularity or intrinsic antimodularity. For example, the fact that a network is scale-free[9] can be detected by simply examining the distribution of the degree of its nodes. Now, being scale-free can suggest a structure with different functional *roles*, in a dynamical network: in a scale-free network certain nodes, the highly connected ones, act as *hubs*, to which non-hubs nodes are connected. Thus, being scale-free implies for a network that some role differentiation exists between its nodes, and this, as discussed in section 3.2.3 is an important form of modularity.

In networks, it is also quite feasible to assess the presence of some kind of community modularity, without recurring to proper modularity detection, with its high computational cost: this general assessment of modularity can be effected by comparing the given network with a set of its randomized versions, where a randomized version is obtained by rewiring in a completely random manner all the links of the original network, but taking care that each node maintains the same number of connections (the same degree) that it has in the original network[10]. Given that a randomized network should not have, on average, modularity, it can then act as a "null model", and comparison of the original network with its randomized version should give, on average, a significant difference if the original network has, intrinsically, some actual modular structure, but should produce a not very significant difference for a network which does not, actually, have modularity, that is, for an *intrisically antimodular* network.

So, in networks at least, it should be quite feasible to decide if, in presence of antimodularity regarding community structure, we are confronting *intrinsic* antimodularity or antimodularity due to the computational cost of the modularity detection algorithm.

[8] For Stephen Wolfram's, classification of CAs, see section 5.2.1.
[9] See section 3.1.3.
[10] See, for example Maslov & Sneppen (2002).

13.3 Antimodular emergence

I propose to call the occurrence of antimodularity, that is, the unavailability of a valid modular description due to the computational complexity of the task of modularity detection, a case of *antimodular emergence*, and to consider it a form of *computational emergence*: it seems indeed sensible to compare the occurrence of antimodularity to a form of *emergence*, in that descriptions showing antimodularity possess some of the proposed features of *weak emergence*, a well-known "metaphysically weak" type of emergence, put forth in some works since the mid-90s by Mark Bedau, which is a notion of diachronic emergence related to certain properties of computational systems. I specifically evaluate here the relation between my proposed notion of antimodularity and that of weak emergence, concluding that antimodularity, under certain circumstances, entails weak emergence, although, as we will see, the converse does not hold for all systems: there are *modular* systems which are, nevertheless, weakly emergent.

The notion of *weak emergence* (*WE*, henceforth) is introduced in the seminal Bedau (1997). In this paper Mark Bedau makes use of a particular terminology, and an explanation of the terms he employs is briefly needed. He states:

> Weak emergence applies in contexts in which there is a system, call it S, composed out of "micro-level" parts; the number and identity of these parts might change over time. S has various "macro-level" states (macrostates) and various "micro-level" states (microstates). S's microstates are the intrinsic states of its parts, and its macrostates are structural properties constituted wholly out of its microstates. Interesting macrostates typically average over microstates and so compress microstate information. Further, there is a microdynamic, call it D, which governs the time evolution of S's microstates. Usually the microstate of a given part of the system at a given time is a result of the microstates of "nearby" parts of the system at preceding times; in this sense, D is "local".[11]

It immediately appears that these terms in Bedau's original definition can be considered having the same meaning of the technical terms I introduced in section 6.6. In particular, the terms "micro-level" "macro-level", "microstate", "macrostate", "microdynamic", have substantially the same meaning in my definitions and in Bedau's understanding of the terms: it clearly appears that what Bedau calls "system" can be interpreted as being a preferred description, playing as usual the role of the microdescription, which is governed by a microdynamics acting on microentities, the "micro-level parts". The global configuration of the microentities is the microstate. There is a macrostate, which is "constituted wholly out of its microstates": this means that a macrostate can be seen as obtained by aggregation, that is, by proper abstraction, from a set of microstates. Indeed, it is specifically stated that the aggregation acts as a sort of "average" on the set of microstates from which it is obtained, compressing the original information carried by the microstates. Thus, it is made explicit that the abstraction from microlevel to the macrolevel is a *proper* abstraction. Bedau, in a footnote, further specifies that the macrostate can be intended in a very liberal way, as a property of sets of microstates, a property of sets of other macrostates, a pattern in the system's microevolution or macroevolution, or many other kinds of even more complex abstraction. Albeit not explicitly stated by Bedau, it can be assumed, I think, that these abstractions must be computable, otherwise I don't see how it could be determined when the macrostate is achieved. Bedau adds another specification, that the system is usually governed by a "local" dynamics. This is better understandable if we consider what he has in mind: further on, Bedau provides examples of weakly emergent processes, examples based on

[11] Bedau (1997), p.377.

cellular automata. It seems that the notion of weak emergence is particularly fit to describe the behavior of certain CAs. It is not explicitly stated, though, that WE applies only to discrete dynamical systems (*DDSs*). However, since in *this* work, as I already stated, it is my intention to consider only DDSs, it is legitimate here to compare the notion of WE to a property which I attribute by definition only to DDSs: namely, antimodularity.

Bedau (1997) proceeds to define the notion of *weakly emergent macrostate* as follows :

> Macrostate P of S with microdynamic D is weakly emergent iff P can be derived from D and S's external conditions but only by simulation[12].

Based on the above terminological considerations, I think Bedau's definition could be safely rephrased as:

> A macrostate is weakly emergent iff it can be derived given the preferred, low-level description of the system and the initial state of the system, but *only* by *microsimulation*, that is, by simulating the system's dynamics step-by step according to its lowest level description (that is, its preferred description).

The condition "only by microsimulation" ("only by simulation" in Bedau's text) in the above statement of weak emergence basically means, as clarified by Bedau at length, that the system's dynamics, and in specific the occurrence of the property seen in an aggregate way as the macrostate of interest, cannot be predicted in any way with respect to the actual microdynamics of the system: the macro-outcome of this dynamics cannot be *anticipated*, but to know if a certain macrostate will be ever achieved, we will have to patiently let the system's microdynamics itself run step-by-step until the macrostate (if any) is achieved. Any other anticipation of the outcomes of the system's dynamics representing the chosen macrostate, is excluded by WE's definition.

The condition of underivability of the macrostate except that by microsimulation entails that, if the dynamics of a weakly emergent system is expressed as a set of difference equations, these cannot be analytically solved, because, it they could, we would have a method to obtain the global microstate of the system at any point in time whatsoever, regardless of the distance of this moment from the current time. Applying to the thereby predicted microstate or to a predicted set of microstates the abstraction which produces the macrostate from microstates, we would obtain a composite function by means of which we could see if the macrostate holds at any moment in time. This way, we would have the possibility of anticipating the actual system's microdynamics, by means of a series of controlled tests of the global microstate of the system at time intervals coarser than the single timesteps of the microsimulation. This possibility would negate the very definition of WE, so it is excluded by the fact that the system is, by hypothesis, weakly emergent. More in general, the condition of underivability except that by microsimulation, means that there cannot be any possible theoretical "shortcut" between the initial state of the system and the macrostate. So, no aggregate valid macrodescription can be given *which is capable of macrosimulation*[13], at least of the particular run of the system starting from a given initial state which ends in the wanted macrostate, because, if an aggregate valid

[12] Bedau (1997), p.378.

[13] It is not excluded by WE definition, however, that a macrodescription not useful for immediate simulation (an *explanatory redescription*, a concept I introduced in section 6.6.10) could be given. See the rest of the current section.

macrodescription producing this macrostate in the run of its simulation could be given, obviously by means of this macrodescription we could derive the macrostate. But this goes against the definition of WE, according to which if the macrostate is WE then it can be derived *only by the microdescription*, and by no other means.

It could seem that, if no aggregate valid macrodescription capable of macrosimulation can be given, no *macromodular* redescription can be given either, because a macromodular redescription *is precisely* a form of aggregate valid macrodescription. And if no macromodular redescription can be given of the system, this means that the system is, by definition, *antimodular*. Yet, this further implication does not hold. There is a caveat here, already highlighted in the former paragraph: the macrodescriptions excluded by WE's definition are those *capable of simulation*. But there are macrodescriptions, as highlighted in section 6.6.10, which are modular, but incapable of simulation: they cannot be run in any obvious way because the specification they provide of the potential DDS dynamics is too vague to be implemented in an obvious way. These are what I called *explanatory redescriptions*, which can be useful only for *explanatory* purposes. To this class belong for example mechanism sketches or schemata[14], which do not provide enough information to allow for their immediate implementation. However, even if simulations on them cannot be *run*, these descriptions can be extermely useful for explanatory purposes, because their high abstractness gives them a high intelligibility.

Another example of a system which is WE but that can nevertheless be redescribed in a modular high-level way is that of a machine capable of universal computation[15]: in such a machine, most dynamical global properties are in general undecidable[16] in the long-run, so in general its macrostates are not predictable except by making the machine run. This fulfills the definition of Weak Emergence given by Mark Bedau. Nevertheless, such machines can be in most cases easily described by means of a hierarchical modular multi-level description: the high-level descriptions act as explanatory devices (as computational explanations, as discussed in section 14.5.1), but do not render the system predictable, because they inherit the same Turing-completeness and with it the general undecidability, and thus unpredictability, of all the other valid description levels[17].

It appears then that antimodularity, which consists in the impossibility of obtaining a modular valid high-level redescription at whatsoever level of abstraction, is not implied by WE, even if the two conditions present a strong affinity. This affinity manifests also in the liberality of the admitted abstractions, and in relativity with respect to the chosen abstraction: detection of the macrostate in Bedau's WE definition is based on an abstraction, and, as Bedau notes, the choice of this abstraction, which individuates a corresponding macrostate, is very free. And WE is relative, by its very definition, to the chosen macrostate: a system can manifest weak emergence relative to a certain macrostate and not to another. This is completely analogous to the choice of a metric relative to which assessment of modularity is performed, in the theoretical framework I propose in this work[18].

I would like to show now that while WE does not entail antimodularity, the converse implication holds under certain conditions, that is, that under certain specific conditions, antimodularity entails Bedau's weak emergence.

---

[14] See section 11.1.6.

[15] That is, a machine equivalent to a Universal Turing machine, and as such capable of any computation, if the Church-Turing thesis holds. See section17.2.5 and 17.3 of the Appendix.

[16] See section 17.2.6.

[17] See a further discussion later on in this section.

[18] Of course, Bedau's conception *predates* this work, so it is certainly *my* framework which can have been inspired by his: credit goes to Bedau's work.

The argument is this: if a system is antimodular, then by definition this means that its *only* valid modular description is its preferred, lowest level description. This implies that the system is not *predictable* by means of a high-level, *modular* simulation: because, if it were, that would mean that the high-level simulation, in that it is capable of predicting the system, represents a high-level *valid* modular description (otherwise, were this modular high-level description not valid, it would have been useless for the prediction, because its alleged predicted outcome would have diverged from the actual outcome of the system). But, in an antimodular system, this high-level modular valid description of the system is excluded by the definition of antimodularity. So, it can be concluded that the dynamics of an antimodular system is not susceptible to be forecast by any *modular* high-level simulation: *if no other non-modular prediction method is applicable*, then the only way to know how the system's behavior will evolve is by simulating the system at the level of its preferred description, that is by microsimulation, as Bedau would say. This last circumstance appears equivalent to the above rephrasing I made of Bedau's weak emergence definition. So, it seems that *antimodularity* $\rightarrow$ *weak emergence*.

The above conclusion, that antimodularity implies weak emergence is not absolutely sure, however, for it depends, as noted above, on the circumstance that an antimodular system, which is not predictable by any high-level *modular* simulation, be impossible to predict by any other *non-modular* means, too. This can be suspected to not always hold: for example, an antimodular discrete dynamical system could perhaps be predicted by analytically solving its update function[19]. If this were the case, this would go against the possibility that the system is weakly emergent, for this analytical solution would provide a method for the derivation of any state of the system at any time, without going through its step-by step low-level simulation. Such a kind of antimodular, non-weakly emergent system, if composed of a large amount of parts, could end up being simply predictable, because its update function is analytically solvable, but at the same time could result completely unintelligible to us due to the excessive number of parts involved in its lowest-level description, which, because of antimodularity, is the only modular description available. I am not sufficiently able here to disprove this eventuality, however it seems to me at least unlikely that systems of this kind are very frequently encountered. In case they were, we would be presented frequently with very curious cases: systems which are perfectly predictable in detail by analytical, mathematical means, but regarding which, of their predictable behavior, we cannot possibly give, due to their antimodularity, any functional explanation at any sensible high level, for the reasons, better explicated in section 14.1, that antimodularity hinders high-level mechanistic and functional explanation.

Another possibility would be that of antimodular systems of which it is possible to give an *aggregate* valid redescription whose language deals with aggregated quantities or entities, but whose machine is not modular: such a redescription would constitute a valid high-level redescription which is nevertheless not actually *modular*: it aggregates only the entities, or ranges of their values, not the *functions*. This kind of non-modular aggregate redescription is not excluded by the antimodularity of the system (which holds by hypothesis). So, the possibility of a system which is antimodular and that at the same time is predictable at a high level, and thus is *not* weakly emergent (because predictability through high-level aggregate valid redescriptions goes against the definition of weak emergence) is not excluded[20].

Actually, there certainly are examples of discrete dynamical systems whose update functions are

[19] Dynamical systems are governed by *update functions*. See section 5.1.

[20] One could think that an ideal gas model like those used in statistical mechanics amounts exactly to such a kind of system: it is perfectly predictable in its aggregate macrostates like temperature, pressure and volume, but we would not be able to understand, due to the sheer amount of particles it involves, its low level detailed mechanical explanation.

analytically solvable or which are redescribable in an aggregate non-modular way, but it can be suspected that they could not be systems capable of complex computations: any system computationally powerful enough is intrinsically unpredictable, because of the undecidability of various properties of the system. So, if it is predictable, the system is certainly not much computationally capable: it would be a trivially periodic system, a system similar to those classified by Stephen Wolfram as in class 1 or 2[21]. The interesting systems considered unpredictable are usually also those that are known as computationally capable, such as the *Game of Life* CA, introduced by John Conway, which has been proved as potentially equivalent to a universal Turing machine, and which, maybe not incidentally, is the CA used as an example of a weakly emergent system in Bedau (1997).

Another objection to the claim that antimodularity implies weak emergence is that it is not completely clear, from the analysis of Mark Bedau's texts, if he intends the impossibility for weak emergent outcomes to be reached by any means except microsimulation, as an *in principle* or only a *pragmatical* impossibility. Antimodularity by definition is considered at least a pragmatical impossibility, due to computational hardness. Should Bedau's weak emergence be considered due to an *in principle* high-level unpredictability, then I think the only option to interpret it, given that in its typical form it concerns discrete dynamical systems, is to view weak emergence as a form of undecidability related to the undecidability of the halting problem[22], and this is actually a juxtaposition which Mark Bedau himself proposes: in dynamical systems which are capable of universal computation, most of their macrostates are weakly emergent for reasons tied to the undecidability of the halting problem. This holds true, for example, for systems which have been *proved* to be Turing-complete, like some cellular automata in Wolfram Class IV, for example ECA rule 110[23].

In light of the above considerations, we could, I think, safely suppose that in most interesting cases *antimodularity* $\rightarrow$ *weak emergence*. A point to highlight is, as we have seen above, that the opposite implication does not hold: there are weakly emergent systems which are not antimodular, that is, systems which do have valid high-level modular descriptions, even if only explanatory, and not useful for prediction. It can then be asked: if the system *has* high-level valid redescriptions, but it is nevertheless weakly emergent, that means that is is not *predictable*: but, why none of these high-level modular description can be used for prediction? The answer could be that they can not because they are too abstract, too vague: they are the kind of modular redescriptions which I called *explanatory redescriptions* in section 6.6.10. But, the reason can also be that these high-level redescription are not useful for prediction because the system is *inherently* unpredictable: this occurs when the system has the power of a universal Turing machine, which, as is well-known, is affected by unpredictability of its evolution due to the algorithmic undecidability of the halting problem[24]: that is, there is no general method to forecast most macroproperties of the outcome of the computation of a universal Turing machine, except that by *executing* it. And, this is exactly the situation of weak emergence: weak emergence compels to execute the computation step-by-step at its lowest level.

So, we could have weakly emergent systems which are nevertheless explainable at a high functional level. I would like to call this circumstance, which is different from antimodular emergence, *Turing emergence*.

[21] See Wolfram (2002) and section 5.2.1.
[22] See section 17.2.6.
[23] see section 5.2.1 for a better explanation.
[24] See section 17.2.6.

The fact that an *unpredictable* system can be functionally *explainable* at a high level ceases to be surprising if we think of this example: take a language interpreter[25] I_1 that is capable of interpreting a Turing-complete language L_1 (most programming languages are turing complete), and write in L_1 *another* language interpreter I_2 for *another* Turing-complete language L_2. This way, any program written in L_2 will be first interpreted by the I_2 interpreter, then in turn the execution of the I_2 program will have to be interpreted by the intepreter I_1. Now, the interpreter I_1 could certainly be written in a multi-level modular fashion, making heavy use of subroutines and procedures, hierarchically organized. Its hierarchical modular description could be then used as an explanation of the functioning of the I_2 interpreter. But, the I_2 interpreter, being the interpreter of a Turing-complete language (the language L_2), is in general inherently undecidable in most of the macroproperties of its dynamical functioning: in other words, I_2 is inherently unpredictable. Nevertheless, its functional structure can be explained by the multi-level modular hierarchical functional description of I_1. Thus, we have a system, I_2, which is *at the same time* inherently *unpredictable* and *explainable* at a high-level.

[25] See section 4.1.4.2.

Chapter 14

Consequences of antimodularity on explanation

In section 6.8 we have highlighted the contribution a modular description of a system can give to the possibility of explaining it. First, it allows for *economy of description* by means of abstraction: the observer can choose according to her needs the right level of abstraction, that is, of detail, at which to describe the system of interest: by choosing a proper abstraction, attention can focus on the level of detail needed for the purposes of the research in question. This favors intelligibility of the description, because the cognitive system of the observer is not overwhelmed by excessive details. In certain cases, economy of description can be obtained by lossless compression in a modular description, when certain modules recur more times inside the description. This also favors understandability. In general, more abstract, coarser-grained modular descriptions, in which to each function is substituted the indication of its role, are more intelligible than low-level microdescriptions, which provide too much information about the implementation. Besides favoring intelligibility of the explanation, which inside the epistemic view of explanation I propose here is an explanatory virtue, hierarchical modularity turns also out as being simply necessary for certain types of explanation, namely functional and mechanistic explanation, in that it provides them with the possibility of multi-level description of a system, and description of the interrelation between levels: multiple level integration, as we have seen in sections 10 and 9, is a necessary feature of functional and mechanistic explanations.

Given that modularity has such a significant influence on the possibility of scientific explanation of certain systems, it is natural to wonder what could be the consequences of *antimodularity* on the possibility of scientific explanation. I take into consideration two well-known models of scientific explanation: functional-mechanistic and deductive-nomological explanation. I then proceed to evaluate a more debated model of explanation, *computational explanation*, and another modality of explanation which has been recently object of scrutiny, that can be called the *mathematical-topological* explanation: this is a non-causal form of explanations apt to explain in many cases certain properties of complex dynamical systems.

The conclusions drawn are that antimodular emergence affects the feasibility of the first two types of explanation, the functional-mechanistic and the deductive-nomological, as well as computational explanation, albeit in different ways, but leaves intact and even *enables*, in certain cases, the topological kind of explanation.

14.1 Antimodularity and functional or mechanistic explanations

I claim that antimodularity negatively affects *mechanistic* explanation, a fundamental form of explanation in biological sciences. A brief detour is in order here to remind what this form of explanation amounts to.

As exposed in section 10, the term *mechanistic explanation* usually refers nowadays in philosophy to a relatively recent model of scientific explanation, put forth since the '90s by several groups of philosophers of biology and of cognitive science. Referring the reader to that section for discussion of the subtle differences between the two main conceptions of mechanistic explanation, a general definition of *mechanism* could be given as:

> A *mechanism* is a structure performing a function[1] in virtue of its component parts, component operations, and their organization. The orchestrated functioning of the mechanism is responsible for one or more phenomena[2].

The definition above defines a mechanism as what I would call a *complex system*, that is a system composed of interacting parts. The point to stress here is that there is a functional view involved: the global function, which represents the explanandum, is explained by describing the organization and interactions of the parts which, by means of their dynamical "orchestrated" functioning, produce the phenomenon. What is needed to explain a given phenomenon is then to first identify the parts and operations involved in its production. To this aim, the system as a whole must be subject to two operations, *structural decomposition* and *functional decomposition*: the first yields the set of elementary parts of the system, while the second identifies component operations. The third phase is *localization*, which consists in linking parts with the operations they perform. This way, a mechanistic explanation is given. This synthetic account sums up the position held by William Bechtel and collaborators: specifically, here I will refer to the position of William Bechtel and Adele Abrahamsen (see section 10). This low-level-only kind of explanation is not always the most desirable: it is important that a hierarchy of mechanisms be considered, and that explanation be *multilevel*, reflecting a possible hierarchical organization of the mechanism. A mechanism may involve *multiple levels of organization*, being often part of a higher-level, larger mechanism: going up in the hierarchy, circumstances external to the larger mechanism can in turn be seen as larger overarching mechanisms, and so on, while, traversing the levels top-down, components of a mechanism can be seen as mechanisms themselves, recursively composed of subparts.

I think this whole vision could be easily rephrased in terms of modularity, along the lines of the vision which I have sketched in the preceding parts of this work: the representation of the mechanism resulting from the composite process of functional decomposition, structural decomposition and localization, is what I have called the *preferred description* of the system: the identification of the basic, lowest level parts which the observer has chosen to identify. Bechtel and Abrahamsen do not stress, as I do, the dependence of this description on a choice on the part of the observer, because they consider implicitly that there are quite natural preferred descriptions of some systems, as there undoubtedly are, for example in molecular biology, where the molecules (or, possibly, atoms) are the most natural elementary parts. The main difference between my view and the view of Bechtel and Abrahamsen is then that my conception of hierarchical modularity is more general, comprehensive of forms of modularity which are non-physical but functional-only, like the modularity of computations.

[1] See section 9.
[2] Bechtel & Abrahamsen (2005), p. 423.

That said, taking into account the definition of antimodularity given in section 13.2, it is easy to show how antimodularity compels to *single-level-only* explanations, neglecting the need, essential for mechanistic explanations, of *multi-level* integration: by definition, antimodularity allows the production of only the two trivial modular descriptions of a system, the lowest level (the preferred description) and the highest-level one. Although for an essential mechanistic explanation these two levels could be sufficient, a more rich hierarchy is certainly wished for in general, in the case of mechanistic explanations, where, as highlighted in section 6.8, the possibility of a rich hierarchical description allows the fine-tuning of the level of detail to the needs of the researcher by means of the choice of the adequate level of description, at the adequate degree of abstraction: the observer will tend to choose coarser levels to have a bird-eye view of the system, in order to gradually proceed towards finer and lower levels, which reveal gradually the implementation details. In other cases, proceeding bottom-up, the observer, after a first low-level description, the typical preferred description, will probably need to partition the set of elementary parts in functional subsystems, modules representable as elements of a higher-level description, in order to catch at once the role that each subsystem plays in the overall functioning of the system. Especially in very large and complex systems, this operation will probably need to be iterated more times up towards higher levels, or the whole hierarchy traversed up-and down until a sufficiently articulated comprehension of the system is formed in the mind of the observer. As stressed in section 6.9, this gradual exploration of the multi-level structure of a system is necessary also in the phase of scientific discovery, in the phase of construction of the mechanistic model of the system by repeated observation and experimental manipulation of the phenomenon to explain.

All the above processes of gradual comprehension and construction of a mechanistic explanation of a system are severely hindered by the presence of antimodularity: antimodularity would limit mechanistic explanation to the level of description representing the most elementary parts of the systems, typically the preferred description of a certain discipline, which is the level with the highest number of parts and thus the most complicated level of description.

This fact could certainly hinder comprehension as well: for large enough systems, their mechanistic explanation at this level is too complex to be understood by human beings, at least too complex to allow for a complete comprehension of the whole description at once. And, unfortunately, antimodularity will hit precisely *large* systems, because it is *due* (at least in one of its possible forms) to the excessive size of the system: the system is too large for automatic modularity detection methods for us to be able to obtain a valid enough modular redescription of it in a reasonable time. As said, inside my epistemic view of explanation, understandability is a quality to be sought for in mechanistic explanation, and on this point other authors agree, notably William Bechtel, Cory Wright and collaborators. Antimodularity damages this needed intelligibility of mechanistic explanations.

One could then hope to understand the preferred description of an antimodular system piecemeal. But, the absence of modularity hinders precisely this possibility: in a non modular system all parts are inextricably interconnected, and the system can be fully comprehended only in a "holistic" way, at least if the required comprehension is comprehension of it as a mechanism.

An important distinction must however be made. According to the definition given above of antimodularity, this property comes in two flavors: *intrinsic antimodularity* and antimodularity *simpliciter*. In the case of intrinsic antimodularity, the system cannot be decomposed at all in high-level modules, because it objectively lacks modularity, at least relative to the metric for modularity detection taken into consideration. In this case, not even piecemeal comprehension is to be expected, because to understand or explain the functioning of a subsystem it is necessary to

succeed in sufficiently isolating the subsystem from the rest of the system: the subsystem must possess a sufficient functional autonomy. But this is precisely what is precluded by intrinsic antimodularity: in an intrinsically antimodular system, functions are distributed across the whole system. For the same reason, in such a kind of system, its experimental study in order to construct a high-level model of it is severely hampered, because this process requires the same type of isolation of subsystems in order to study them separately by intervening on them[3].

In the other possible case, that in which the systems shows simply antimodularity, the impact on mechanistic explanation is more subtle: the net effect of simple antimodularity is that we cannot practically obtain a modular description of the system by means of general algorithmic methods (like those for modularity detection in networks described in section 3). But, if the system is antimodular but *not* intrinsically antimodular, we could *hope* to obtain its modular description in other ways. One could be the piecemeal, slow, experimental fashion traditionally used in science. If the system shows antimodularity but it is actually modular, that is, it is not intrinsically antimodular, then it is in principle possible that this method could actually find its modular structure. It does not seem to me, however, that the likelihood that this could happen without any supplemental information on the system's possible modularity is high: without any other hint, the brute experimental research would in many cases simply apply the method that would have been applied by an algorithm for modularity detection, only in a *manual* way. If this is the case, it is easily realizable that there is not much hope that, where an algorithm has failed, its manual step-by step execution could succeed! To make a very simple example, let's take a social network, of which we want to assess modularity based on a metric of friendship. The network is too large, and modularity detection on it has proved unfeasible. It is thus affected by antimodularity. But what about proceeding by discovering the friendship relations one by one, starting from one person and proceeding to reach other, connected persons? To asses modularity we could observe the mean connectedness of adjacent nodes. But network modularity in the form of community structure[4] is not a local property, because it is based on the assessment of edge betweenness, a metric which gives also information about distant, non-local nodes. So, we would have to wait the complete discovery, one node at a time, of the whole network's structure, before being sure to have all the data needed to detect the proper community structure. And, after having obtained a description of the complete structure, we will still have to run a method for community detection on it. It does not seem, then, that we would have gained anything over the direct automated application of the algoritmhic method. This does not mean that, if actually present, community structure cannot be detected, even if antimodularity affects the system: it could be probably detected with the aid of information *external* to the bare description of the network's structure, which can guide and constrain the detection of modularity. For example, knowing which language is spoken by each individual, would allow for a preliminar approximate coarse-graining of the network structure, based on the hypothesis that the probability for two persons to become friends is higher when they share a common language, and so on the plausible conclusion that a possible modular structure of friendship relationships on the network should follow at least partially the partition of the population into linguistic communities. But this is information which is *external* to the original preferred description, that instead represents the network simply in terms of connected persons. Thus, it is not impossible that methods different from the algorithmic detection based only on the properties of the representation could detect modularity even if the system shows antimodularity, but this can happen when these methods can appeal to information external to the representation.

In actuality, however, when in presence of already collected data which make up a complex

[3] As seen in section 6.9.
[4] See sections 3.2.1 and 3.2.1.2.

representation of a scientifically interesting phenomenon, in many cases algorithms for modularity detection are applied to these data. This is becoming more and more frequent in the study of biological networks, like metabolic networks, and a paradigmatical case of application is Guimerà & Amaral (2005b), with fig. 3.14 in section 3.2.3 representing an example of algorithmically detected modularity obtained by the methods proposed by these authors.

It is evident that a mechanistic explanation tries to answer to "how" questions ("how a phenomenon is brought about?"), by showing the way by which the complex dynamical functioning of a set of interacting parts produces the phenomenon. The same question can be answered to, also just from the *functional* point of view, and this conception, mainly aimed at characterizing explanation in cognitive psychology, has been notoriously advanced by Robert Cummins. In a way similar to that of mechanistic decomposition, functional analysis begins with a characterization of the global phenomenon (the *disposition*[5]) taken as the overall function to be explained in terms of its component subfunctions. This is a typical form of so-called *role functionalism*, in that the concept of function[6] is considered that of a partial role fulfilled by a subsystem in order to bring about the whole functioning of the overarching system. Seen from an explanatory point of view, the function of a subsystem is employed in explaining how the overall function, which is the explanandum, is performed by means of the organized contributions of its subfunctions, which fulfill their roles in a programmed activity. This position is quite close to a computational view, and it is completely compatible with it. Actually, Cummins' functional analysis is the prototype of the typical explanation of cognitive psychology, which mostly consists of functional explanations, often in the form of *computational explanation*, that is, the exhibition of a computer program able to produce the cognitive phenomenon to be explained.

A more thorough characterization of Cummins' position is given in chapter 9, and in section 11.3, where the relationship between purely functional and mechanistic explanation is also better analyzed. What I would like to highlight here is that Cummins himself, since his earlier works, as in the seminal Cummins (1975), stresses that the strategy to seek for in scientific explanations, especially in biological ones, is recursive functional decomposition until a full hierarchy is obtained. Antimodularity would completely hinder this goal, allowing for a two-level only explanation: the highest one, that of the explanandum itself, and, on the other end of the scale, the lowest level, that of the most elementary functions.

14.2 Antimodularity and the deductive-nomological model

In the classic deductive-nomological (*DN*) view of explanation, stemming from the seminal work of Carl G. Hempel and Paul Oppenheim[7], which we have already discussed in section 8, explanation is seen as *logical deduction* of the explanandum from the explanans, and little attention is directed to the *understandability* of the explanation: a concern about intelligibility of the explanation would have been considered, in the historical post-neopositivistic milieu of the time, an inappropriate trespassing of philosophy of science into the territory of pragmatical, or worse, *psychological* aspects of scientific explanation. In the DN model of explanation, all that matters for an explanation is that it is a valid and sound deduction. Explanation is seen in this model as depending on the possibility of *prediction* of the phenomenon by means of a scientific *law*. The explanation itself amounts to the description of the logical derivation of the explanandum from a group of premises constituted by a scientific law and a set of clauses representing initial conditions of the system.

[5] A discussion on this term of art is in chapter 9.1.
[6] The notion of function is better examined in chapter 9.
[7] Hempel & Oppenheim (1948).

14.2.1 Antimodularity and weak emergence hamper DN explanation

I claim here that antimodularity hampers, at least partially, also the production of sensible DN explanations. In order to support this claim, it is first necessary to show that antimodularity, under certain circumstances, entails another property of complex systems, Mark Bedau's *weak emergence* (or *WE*). This has been already showed in section 13.3, and the circumstances in which this implication is valid have been specified. Let's consider here the case of an antimodular system for which such circumstances hold: antimodularity entailing WE, such an antimodular system will end up being weakly emergent as well.

I want to argue, then, that WE hampers DN explanation. If a phenomenon is weakly emergent, then a DN type of explanation citing an analytically solvable law could not be recurred to in order to explain it, because, if it could, that would mean that the phenomenon is *predictable*, and this is negated by the definition of weak emergence itself, as reported in section 13.3. To clarify: it is excluded by the definition of a weakly emergent phenomenon that it can be predicted by means of a law which, given the initial state, determines in which state the system is going to be at any given time, and that this law has a mathematical expression which can be *analytically* solved. As showed in detail in section 13.3, this is excluded by the very definition of weak emergence, which basically states that a weakly emergent phenomenon (in a discrete dynamical system) is one that cannot be predicted, and that can be reached only by performing the step-by-step microsimulation at the system's lowest level.

So, no DN explanation of a weakly emergent process could be based on such a type of analytically solvable law, because otherwise any weakly emergent phenomenon could be anticipated and predicted, in a way, by making use of the analytical solution of this law (as explicated in section 13.3), and this would go against the definition of weakly emergent process. But we are dealing here with a process which is antimodular by hypothesis, and in conditions that entail WE, so the process is WE by hypothesis, and thus its explanation cannot be based on an analytically solvable law.

Actually, DN explanation requires in the explanans the exhibition of a *law* which allows to deduce, from the initial conditions, the explanandum. But, if the system to be explained is weakly emergent, such a law can not be expressed as an analytically solvable expression: in general, the WE definition excludes that the emergent macrostate can be predicted by means of a law in this sense.

Lacking this kind of analytically solvable law, we should reflect on what kind of law could be employed in a DN explanation of a weakly emergent process.

Given that each discrete dynamical system is governed by a global update function, this is plausibly the law we have available and that we could employ in the DN explanation. In a discrete dynamical system (or *DDS*), this update function maps each possible global state to its next state. Given that the global state of a DDS is constituted by the configuration of the states of each of its elementary parts, the global update function must, in its representation, cite all the elementary parts of which the system is composed. This is explicitly possible only for *finite* DDS, whose configurations are composed of a finite number of parts. Moreover, the number of couples $(currentstate, nextstate)$ is proportional to an exponential function of the number of elementary parts: for example, in a boolean system in which each elementary part can be in one of two possible states, the number of possible global states is given by 2^n, with n the number of elementary parts. For a system composed of even a small number of parts, the update function would become very complicated, at least if it is expressed in the extensional form which consists in explicitly listing all the possible $(currentstate, nextstate)$ couples: as said, the number of

these couples is proportional to an exponential of the number of parts of the system. For a boolean system with 50 parts, the list of this couples comprises 1,12589990684e+15 elements, a completely unmanageable number. Besides, an extensional form like this would not satisfy the requirements that a *law* must have in the DN model of explanation, as conceived by its authors: for Hempel and Oppenheim, the law must be expressed in a universally-quantified expression, which does not mention any individual constant[8]. But an extensional listing of all the possible $(currentstate, nextstate)$ couples does not seem to have this universally purely quantified form: the listing could at most be rephrased as an enormous conjunction of component clauses like "for each current state S, if S is equal to a certain configuration C, then the next state is ...". This does not seem to have the required purely universal form, because at least it contains many constants: all the possible specific configurations, each of which must be cited in a component of the alleged "law". So, it would *not* be possible to use this form of expression of the update rule in a DN explanation, because this violates the requirements of such a kind of explanation.

Any extensional representation of couples of discrete configurations like the representation of the update rule cited above, is theoretically expressible also in an "algebraic form": for boolean DDSs, this would be a logical boolean expression, for example an expression in conjunctive normal form, or disjunctive normal form. However, transformation of the extensional expression into the algebraic formula could turn out being difficult: even if a boolean expression can be obtained from the extensional representation, this expression could end up being huge. To enhance its manageability, the formula must be reduced to its *minimal* equivalent boolean expression. The problem is that this reduction is computationally hard: the most used precise algorithms, such as the *Quine–McCluskey* method, work in a time proportional to an exponential of the input size.

A third problem with such a formula, even if it can be obtained, is that it would not qualify as a "law" which can be cited in a DN explanation according to Hempel and Oppenheim's standards, because the boolean expression representing the update function of a finite DDS would have, as said, to cite all the parts of the DDS in order to consider the global configuration, or state, of the DDS. Such a law-like expression would have to be expressed in a form such as "for any configuration of n parts among all possible configurations of n parts, the next configuration is equal to (p_1 OR p_2) AND (p_2 OR p_4) [...]", where n is the finite number of parts of the system. Thus, such an expression would quantify over the finite scope constituted by the set of all possible configurations of n parts, which, in the case of boolean systems, is constituted by 2^n configurations. In doing so, such an expression would refer to a *finite scope*, the scope of the possible configurations of the parts of that finite DDS. This violates the requirement that a law employed in a DN explanation must be have a *non-limited scope*. In other words, a generalization for a specific DDS is a generalization concerning a *finite*, *particular* system, and this rules out such a generalization as a legitimate *law* to be used in a DN explanation, by the strict standards conceived by DN explanation proponents, as explicitly stated in Hempel & Oppenheim (1948).

To sum up, a DN explanation cannot be applied to a finite DDS, if its update function cannot be analytically solved, and it must be expressed in the form of an explicit (*current global state*, *next global state*) map, thus quantifying over a finite scope, the scope of all possible global states.

Now, a DDS which is affected by WE, would for sure lack analytical solvability of its update function. However, not every DDS must represent its update function as an explicit global state to global state map. There is, actually, a class of DDSs whose update rules usually are quite synthetic: this is the class of cellular automata, whose update rule is *local*, taking into consideration only a neighborhood of limited radius around each elementary part (a cell) of the

[8] See Hempel & Oppenheim (1948).

system. Of course, even for finite CAs the *global* update rule is not local, but in CAs the global update rule can be obtained by *reiteration* over each single cell of the more synthetic, local, CA rule: this is how the next state is computed starting from the current state in a CA[9]. Moreover, CA rules, besides being local, are *invariant in all points of the CA's lattice*: there is actually a theorem, the Curtis–Hedlund–Lyndon theorem, which guarantees that *any* local rule invariant in all points of a discrete lattice defines a cellular automaton, and vice-versa[10]. These features make the CA rule obey the condition, required for DN explanations, of being non-limited in scope, and in this regard, associated with its locality, the form of a CA-rule can in a way be assimilated to the form of a physical *law*, which is the typical example[11] of a law of the type required for DN explanations: the CA law could be expressed as "for any point in the CA lattice, given certain conditions of the cells adjacent to the cell at that point, then [...]", which is a universal form similar to the "for any point in space, given certain conditions of the particles within a certain distance from the particle at that point, then [...]" of a possible physical law.

Of course, by definition of WE, for a CA showing WE, its CA rule cannot be analytically projected at any moment in time farther than the next timestep, so the rule must be applied step-by-step. A WE process generated by a CA could then, in a way, be explained by a possibly very long list of steps of its evolution, a list which can be seen as a list of *deductive* steps inside a formal logical system, in which the premises are constituted by the CA's initial configuration plus the CA-rule, a rule which is repeatedly applied, subsequently, to each intermediate next state obtained by the former application of the CA rule, and where the conclusion of the deduction is constituted by a certain required state of the CA, the state which corresponds to the WE macrostate. Accordingly, by this analogy, the production of this list of consecutive states of the CA and application on them of the CA rule, could in a way be assimilated to a sort of long *DN explanation* of the WE macrostate, given that a DN explanation consists of a logical deduction of the explanandum starting from given initial conditions and a law: the initial condition is the initial global state of the CA and the law is the CA rule.

Even in this case, human comprehension would be hindered by the potential length of the list. According to the theoretical position of post-neopositivistic advocates of the DN model of explanation, however, understanding is an inessential feature of explanations, and it is not required for a good DN explanation. So, in a way, weak emergence and, consequently, antimodularity, does not essentially hinder DN explanation, at least in the case of CAs and other systems whose dynamics follow a *universal* rule.

But antimodularity of a DDS can hamper DN explanation of the system in general: basically, only a sort of *DN-like* explanation could be allowed, and only for CAs. And, even in that case, intelligibility of such a potentially extremely long explanation could end up being next to null.

14.3 Antimodularity and topological explanations

I proceed in this section to consider the consequences of antimodularity on the possibility of explaining a complex system by means of a type of explanation recently identified by Philippe Huneman, who proposes to call it *topological explanation*. We have already considered this kind of explanation in section 12. As better explained there, topological explanation occurs when certain features of certain abstract representations of a system, features which are purely

[9] See also section 5.2.

[10] See Hedlund (1969) and Arcaya & Romero (2007).

[11] It is not surprising that almost all examples of legitimate laws given by the DN explanation proponents are of *physical* laws, given the logical empiricist background of the authors.

topological, are employed in order to explain certain other properties of the represented system. Topological features can be for example the shape of a curve, or a modular or nonmodular structure of a description. In this kind of explanation, it is not *causal* features of the system which result explanatory relevant, but what does the explanatory work are purely formal, abstract geometrico-mathematical, *topological* properties indeed, of some representation of the system. In section 12 we have already seen an example, taken from Huneman (2010), of such a kind of explanation, based on curves representing the fitness distribution of two bacterial populations. I would like here to make two other examples, centered on certain topological properties of network representations.

We will take into consideration a possible topological explanation of the type of dynamics occurring on a network. Let's suppose the dynamics reveals itself as being chaotic. This could be topologically explained by mentioning the non-modular structure of the network: in a non-modular network all nodes are connected on average with the same intensity to all other nodes, and this could explain the chaotic dynamics ensuing[12]. This is not a mechanistic explanation, because it does not specify the particular causal interactions between nodes which give rise to the chaotic dynamics: it mentions only a mathematical, topological feature, that of the network being extremely connected, that is, antimodular. Now, this kind of explanation requires that the network is *actually* antimodular, that is, that it is *intrinsically antimodular*. Detection of intrinsic antimodularity, as suggested in section 13.2, is feasible in certain cases, so it could be possible to determine if we can recur to a kind of topological explanation based on intrinsic antimodularity.

Topological explanation could also have recourse to the mention of a *modular* structure: for example, this can be the case when *robustness* of a network's dynamics to local perturbations is explained, topologically, by mentioning the network's *modular* structure: a modular structure ensures that perturbations remain local or get channeled towards specific target parts of the network, often with different intra-module and inter-module diffusion rate, without spreading indiscriminately across the whole network[13]. But, let us suppose that we do not find modularity in the network, and so that it results antimodular. Again, we could exclude, in certain cases, by algorithmic means simpler than proper modularity detection, that this is intrinsic antimodularity. In this case, even if the network results to us antimodular because of the limitations of algorithmic modularity detection, and thus we cannot give of it a proper functional hierarchical explanation, we could still, by knowing that the network *has* some modularity even if we cannot precisely detect it, explain by *topological* explanation certain features of its dynamics, such as its robustness, by merely citing the fact *that* the network is modular.

So, if a network shows antimodularity, this means that we cannot have available a sufficiently precise modular description to be used for a mechanistic, functional, multi-level explanation. That does not mean that fast algorithms for detection of the simple *presence* or *absence* of modularity cannot suggest that the network has some degree of modularity anyway or that it completely lacks it, and this mere information could allow for the supply of a topological explanation, albeit not of a mechanistic one.

To sum up, it seems that antimodular emergence does not hinder the possibility of topological explanation, but that in reality the presence or absence of intrinsic antimodularity is precisely one of the features which can typically *allow for* the production of certain topological explanations.

[12] Usually dependent on other factors as well, such as the average degree of nodes, and the type of connections. See section 7.1.2 and Kauffman (1993).

[13] See, for example, Maslov & Sneppen (2002), and section 3.2.4.1.

14.4 Explanation and prediction

The circumstance that an *unpredictable* system can be at the same time functionally *explainable* at high-level, a circumstance which occurs in modular, computational, sufficiently powerful systems, and which I have called Turing emergence in what precedes, is an interesting fact, because it shows that prediction and explanation, contrary to the expectation of post-neopositivistic philosophers like Hempel, are disjoint: while unexplainability (that is, antimodularity) in many interesting conditions entails undpredictability (weak emergence), unpredictability does not render, per se, a system unexplainable. This is a curios result, for it proofs, in a way, that *prediction* is not necessary for explanation, and thus that the deductive-nomological model of explanation, even if it were free from other shortcomings, at least it would not be a model of explanation comprehensive enough, leaving room for explanations which are not based on prediction: explanations which are functional, or mechanistic, but cannot be considered also DN, except by using as a "law" the very functioning itself of the mechanism, or of the global function. But this would be excluded by the requirements of DN explanations, which we have seen in section 14.2.1: a DN explanation could not accept *as a law* a mechanism. If we, however, consider DN explanations as simply a higher-level type of mechanistic explanation, which is, in a way, a specifciation of a mechanism, where the mechanism implements this specification (in a way similar to Mario Bunge's view,cited in section 10), then we could unify the two types of explanations, mechanistic and DN one, into a single type. The DN explanaiton would simply be a more "high-level" type of explanation, specifying the "what", while the mechanistic explanation would describe the "how". See section 10 for a discussion.

14.5 Antimodularity and computational explanation

14.5.1 Computation and computational explanation

In the next sections I will try to reflect on the possible consequences of antimodularity on computational explanation, by evaluating the circumstances under which a dynamical system can be explained by means of a computational explanation. In order to do this, it is necessary to try to clarify what a computational explanation is. But, to this aim, it can seem natural to ask first what a *computation* is. Indeed, while this concept seems generally undisputed, a minimum of reflection can make it appear a little problematic.

The fact is, since its inception with Turing, computation has been considered a purely formal question. This, I think has been a good way to treat it, because logico-mathematical methods have allowed to prove fundamental results on computation which no other approach could have discovered, providing a formal framework with a capacity of pervasiveness in all fields of human knowledge which maybe only mathematics has had before. The formal approach to computation has brought to light the essential features of computational facts: the properties of computability, its power and limits, have been detailed, and continue to be, in a thorough and rigorous way that no other approach to the problem of computing could have developed. So, isn't computation precisely a question of mathematics? Of course it is. Nevertheless, given a computational system, one could ask a simple question: does it compute? In the wake of what stated above, this can seem an easy question, decidable on formal grounds, but from another point of view it is not.

To see this difficulty, let's imagine that, for fun, a programmer wrote a computer program constituted by a list of randomly chosen instructions, and that, for some improbable circumstances, the randomly produced program does indeed run without crashing, producing an output for each input it is fed with. Another programmer enters the room and asks to the first programmer

What does it compute?, receiving as answer *Well... it computes!*. Would that answer satisfy the second programmer? Or, let's think of another program which spits out unrecognizable binary strings, or even decimal numerical strings, when fed with other numbers: what does it compute? Without knowing anything of the program structure, we could say that what it computes could be anything: from trigonometric functions to derivative of the probability distribution of some demographical model. Wouldn't it be more sensible not to say that it computes, in the sense of processing some meaningful information, but that, for the moment, it simply is a digital process?

The point I would like to highlight is that, at least from an intuitive standpoint, recognizing that something computes requires the ability to explain *what* it computes, that is, the ability to provide a *specification* of the program, in terms of a meaningful explanation of what function is computed. It is my contention that a computation properly intended must be distinguished from a mere manipulation of symbols, that is, from a mere digital processing: a computation in the meaningful sense needs to be, as such, a *meaningful* digital processing. And, perhaps surprisingly, I think that at this point a *purely formal* conception of computation can be recovered, after having apparently refused it, in order to define what a meaningful computation is.

To clarify the question, let's start with mere digital processing: a digital process is simply the formal manipulation of tokens. Following John Haugeland, I will call a mere digital processing a *syntactic engine*: a digital process, acted by a "machine" in the sense of theoretical computer science. But, processing performed by a syntactic engine achieves the status of meaningful computation, (or meaningful "information processing") when the types of the tokens it manipulates are given a meaning, and the operations it performs on these tokens are given a meaning as well. This attribution of meaning, though, must not be seen as something "vague", or irreducibly "mental" or irreducibly not mechanical, but, it can be simply reduced to a formal, computable *mapping* of some sort. And this is how, in my view, the problem of intentionality can be avoided and a purely formal syntactic vision of computation is gaining back again a central position: the attribution of meaning is simply the computable association between some set of meaningful symbols (of which nothing is said about their meanings and how they acquire them) and the set of symbols manipulated by the digital process. Of course, the problem of intentionality has been simply deferred, here: we start with the idea of a set of *already* meaningful symbols, which are furthermore perfectly legitimate discrete symbols candidate for a perfectly legitimate kind of computation understood as digital processing. By means of a computable function acting on them, we map (the *encoding*) in some way this set of symbols to the set of symbols constituting the possible input configurations of the syntactic engine, which we wanted to "endow" with meaning. Correspondingly, the possible output configurations of the syntactic engine must be mapped in some way to the set of meaningful symbols (the *decoding*). That is all that is needed to turn the digital process performed by the syntactic engine into a meaningful computation: by way of this mapping, the inputs and the outputs of the syntactic engine come to assume a meaning, and as a consequence, the relationship (the specification) between its input and its outputs gets a meaning: and this meaning is precisely *the* computation the whole system, so mapped, performs. This is for example the case in which we can describe the specification as "it is an image processor". But before the mapping, the same syntactic machine was simply to be described as "it is a transformation matrix such and such...". These mappings, this encoding and decoding of an input and of an output, can be seen as operations of abstraction as understood in section 6.6.1. Once the mapping is established, we can operate a global intentional attribution, and say *what* computation the system, provided with the mapping, is performing. Only then the system can be seen as *computational*. Computation is *attribution* of computation, to systems which *per se*, are simply rule-governed discrete dynamical systems[14].

[14] Of course it can be raised a problem here: if the system is considered as already rule-governed, that means

This intentional, or semantic, view of computation is not a new position: in 1980 Jerry Fodor wrote: "To think of a system (such as the nervous system) as a computer is to raise questions about the nature of the code in which it computes and the semantic properties of the symbols in the code. In fact, the analogy between minds and computers actually implies the postulation of mental symbols. There is no computation without representation"[15].

But, a clarification is crucial here, in order to avoid misunderstandings. I am not saying (and Fodor neither) that computation is an *intrinsically* intentional, or semantic process, whatever that could mean: there is no doubt that a computing machine is an absolutely formal, *syntactic* engine: it operates transformations of its internal configurations according to the *form* of these configurations, not according to any "intentional content" whatsoever. The "semantic" or "intentional" attribution is *external* to the machine, and acted by the *observer* on the purely formal tokens which the machine processes. Such a semantic view is opposed by some authors, like Gualtiero Piccinini, who view computation as definable on purely mechanistic terms, without the need of recurring to any semantic attribution[16].

The idea of devising a map between meaningful symbols and configurations of a machine in order to say what it computes, is however not usually the way a programmer works: when designing a program, the programmer *starts* with an idea of what the computation must do, and proceeds to write the corresponding program guided by this idea. It is this idea that specifies *what* the program must do, and it is accordingly called, as a term of art, *program specification*. It can be considered *the* function which the program, or the computation, *computes*. When programming in a structured or modular style, the programmer then proceeds to analyze the supposed specification into sub-functions, and to implement them one by one, in turn subdividing them into smaller functions, until only single instructions are found.

So, given such a mapping and the fact that we choose the mapping, does this mean that *any* machine can be seen as a machine which is *computing*? Let's clarify things up: first, we are talking about *digital* computation here, the kind of computation classically studied in computation theory since the works of Alan Turing, Alonzo Church, Emil Post, and others in the 1930s. So, to be considered computational, a system must at least be considered digital, that is it must possess, and operate on, a finite set of possible stable distinct states. To this aim we must be able to robustly distinguish discrete configurations of the machine, and this is not always feasible (think of distinguishing stable configurations in a turbulent fluid), so not every system is so conceivable. And, the process must susceptible to be viewed as characterized by a deterministic rule that makes it pass from one stable configuration to another. Conditions for the occurrence of such a possible interpretation of a system are not trivial, and have been the object of subtle philosophical analysis. In the terms of the theoretical framework I propose in this work, though, it is easy to state these conditions: the system must be a modular redescription, as defined in section 6.6.10[17]. Specifically, a system, to be computationally capable, must be a DDS.

---

that an original intentional attribution has already been done. It is outside the scope of this work to tackle here this and other similar thorny questions, analogous to the infamous "kripkensteinian" rule-following problem.

[15] Fodor (1981).

[16] See for example Piccinini (2008).

[17] Along these lines about the possibility of redescribing any system as digital, a famous extended debate, stemming from Putnam (1988), and getting to Chalmers (1996) (through Searle 1990) has tried in the past decades to define the condition for distinguishing computational mechanisms from other systems. While a detailed analysis of this discussion would be very interesting, this problem is slightly different from ours because it touches also the relationship between the real, physical, *continuous* world and its discrete models, a relationship which I purposely left out of my philosophical analysis, which considers only systems which are *already* digital. For this reason, and for reasons of space, i leave it to another occasion. Suffice to say here that, against a purely "intentional" view of computation like the one outlined by Putnam, Chalmers argues that to be computational, a system must possess what in the framework I proposed would be described as some form of functional modularity.

To put all the above more analytically, I would like to highlight some points of my position:

1. In order to compute, a system or mechanism must possess *computational capability*. As said, in order to be computationally capable, a system must be susceptible to be *seen* as digital, or discrete, and deterministic. This is mainly represented by the class of systems which I referred to informally in section 5.1, the *discrete dynamical systems* (*DDS* henceforth).

Note that *computational capability* is different from *computation*: a computationally capable mechanism, *per se*, is not computing anything, although it could, under the following condition.

2. In my view, *computation* is *attribution of computation*, and this is an intentional question, requiring that a computationally able mechanism be interpreted as performing *a specific* computation, that is, requiring that it be possible to answer to the question "what is this mechanism computing?". To this aim, two important sub-conditions must be satisfied:
    a. an algorithmic mapping between linguistic symbols and possible input and output configurations of the system must be realized;
    b. we must be able to say *which* is the particular function relating input to output configurations, that is which is the *specification* of the computation, in order to say what the system is computing.

It must be noted that, again, for a programmer, the two points *a* and *b* above get accomplished in the inverted order: when a programmer writes a program, what is needed is not an *interpretation*, but the *establishing* of a *norm*: first the specification[18] (point *b*) is arbitrarily chosen and considered the norm to which the program will have to conform, and then *on the basis of the specification*, the programmer chooses the *mapping* (point *a*) from symbols to input and output configurations which he deems best, in order to proceed to the *implementation* of the program, that is, the specification of the parts and the structure of the program which will, at the end, best be able to realize the chosen specification according to the chosen mapping. Thus, the choice of the mapping *determines* the choice of the *specific* structure of the *program*. All this series of operations constitutes the *implementation* of the chosen specification.

I take the occasion here, speaking of *implementation*, to express that, along the lines of Galton (1993), and Partridge & Galton (1995), I consider the relation specification-implementation a very universal one: an *implementation* is the act of specifying a method to "realize"[19] a given overall specification. When considering a program, there is not, however, a *unique* overarching specification and a single level of implementation, for the two notions are *relative*, exactly like those of "higher" and "lower" description level[20], and that of function, which[21] is the partial role something fulfills relative to the scope of a global function. *Relative* in this case means that something which is the implementation of a specification, can in turn be considered a lower-level specification to be implemented at an even lower level. Actually, I see the relation between levels of description, inside the theoretical framework which I propose in section 6.6, as a relation specification/implementation. In other words, given a specification there is the need to find one possible implementation of it, and in the style of structured or modular programming, such

[18] A term of art in computer science. See section 4.1.5.
[19] In a sense closely akin to that of the property of *realization* in philosophy of mind. I will not scrutinize the notion here, but see Polger (2004) for a good discussion, and section 9.1.
[20] See section 6.6.
[21] Recall section 14.1 and section 9.

an implementation will be decomposable itself into modules, which, structured in a system of modules, implement the specification. Each module, being a specific input-output function[22] constitutes itself, in turn, a specification, which will be implemented at a lower level, and so on. I tried to give a finer-grained classification of the types of specification in section 4.1.5.2.

It seems to me plausible to say that the *same* abstract structure underlies the notions of structured programming, functional decomposition, and hierarchical levels of modular descriptions. All of these visions are isomorphic to a *hierarchy* in which each macro-component is multiply realizable by sub-components, and so on[23].

Following the same lines, it must be highlighted that, in this view of the recursive decomposability of system representations, there is not an *absolute* bottoming out: using the terminology I introduced earlier, the bottoming out coincides with the description of the system according to the preferred description, which is anyway the product of an intentional *choice* itself. Of course, there are preferred descriptions that are more "natural" than others, like the description corresponding, to remain in the field of computing, to the so-called "hardware" level. I would like to stress that even the apparently physical, resistant, hardware level, is only the preferred description we make of a mechanism: the choice operated at point 1 above[24].

In the examples which I gave above, the ones not involving the act of actively programming, we start with an *uninterpreted* discrete process and aim to *discover* what computation, if any, and how, the process performs. This is the path of *reverse engineering* of a computational or of an allegedly computational process. Actually, the process is instead, usually, that of "forward" engineering: since a computation is almost always something which is *designed* by a human programmer, the programmer starts with an idea of what the computation she is about to implement must do, and proceeds to write the corresponding program. In this case, the programmer starts with the global computational function of the program in mind, its *specification*, which can be a vague idea, or, in better cases, the exact specification of all the possible input-output couples. This can be considered *the* function which the program, or the computation, *computes*. The programmer who wants to follow a structured or modular programming style, then proceeds to analyze the supposed specification into sub-functions, and to implement them one by one, if possible subdividing them in turn into smaller subfunctions. But there is also top-down reverse engineering: cases in which, given an already existing system which performs a known computation, it has to be determined *how* this computation is brought about. This is an attempt to Cummins-style functional decomposition, the functional explanation *par excellence*, which we have encountered in section 9.2, and it is the path usually followed when giving *computational explanations*: its main scientific application is in computational cognitive psychology, where the *specification* is the cognitive faculty constituting the phenomenon to explain, and the researcher tries to infer its computational implementation. When she succeeds, she is able to offer a *computational explanation* of the psychological faculty, in terms of a hierarchical, functional representation of the computation to be explained.

[22] function in a mathematical sense, see. section 9.

[23] Multiple realization does not hold everywere, though: for example, when modelling with network models biological complex systems, it is not always sure that a function, say regulation of the *lac* operon, is multiply realizable: most probably its multiple realizability is only minimal, with slight differences between individuals or species: the reason is that regulation of the lac operon is a very specific notion. Of course, more general or abstract functions can be more multiply realizable: the paradigmatic case is that of mental functions.

[24] It must be stressed that this is not an *arbitrary* choice, but it is exposed to some constraints, of causal, or, from another point of view, normative nature: this is a very thorny question which, as I said, continues to be at the center of a passionate debate, starting from the famous example by Hilary Putnam of the computing "rock", and involving the infamous "kripkensteinian" paradox of rule-following.

14.5.2 Antimodularity, cellular automata and computational explanations

After having assessed the possible consequences of antimodularity on various types of scientific explanation, I turn to consider a fourth possible type of explanation: computational explanation. Here, however, i will focus not on computational explanation as it is typically applied, that is, in explaining psychological phenomena, but on the assessment of the possible effectiveness of computational explanation in explaining the behavior of certain cellular automata: this will serve as a paradigmatic highly simplified model aimed to highlight certain general problems that can be encountered in trying to computationally explain antimodular systems.

In section 14.5.1, I have already expressed the three conditions according to which, in my view, a system can be considered computational. If CAs and dynamic boolean networks can be considered *computational systems*, they could then be subject to *computational explanation*, which is the form of functional decomposition employed in explaining computer programs, and, notably, psychological faculties in cognitive psychology.

I consider the case of trying to computationally explain a CA. There are two basic questions at stake here:

- is a CA a computational system?
- if it is, how can we explain the computation it performs?

So, can a CA be considered a *computational* system? Of course, at its natural, preferred description, that of the *cells* and of its CA-rule, a CA is trivially *computationally capable* (condition 1 stated in section 14.5.1), given its nature of a discrete system evolving in time according to some specified rule.

However, its functioning cannot be explained by means of a computational explanation before an adequate mapping between its configurations and meaningful symbols of some type can be supplied, and, based on this mapping, a meaningful description of the input/output it performs is given (condition 3 of section 14.5.1)[25]. If we wanted to begin to computationally explain what a CA computes at its preferred description level, that of its cells, without giving it a particularly significant interpretation, we could say for example that, "according to the repeated application of its rule, the CA produces a progressive variation of the state of its cells, which can, under various conditions, change from white to black". Here the mapping of the cells' state to linguistic symbols would be to the terms "white" and "black", and this is not very significant to start with. But, the specification given is not very specified indeed: "progressive variation of the state of its cells under various conditions". What about rendering the specification more perspicuous? Now, the specification is given by the repeated application of the CAs rule. So, if we could recognize as meaningful the specification of the rule itself, then we could immediately give a modular computational explanation, modular in the sense of citing the repeated calling of the same function, which in this case is the CA rule. Now, a CA rule, at least in CAs whose cells have only two possible states, is a map between binary configurations. As such, it realizes a boolean function, and so it could be meaningfully expressed as one. Actually, there are some simple CA rules[26] whose boolean functions are known. For example, we could explain by computational

[25] I partially overlook here a further problem: how a CA, which has a potentially non-halting evolution, can be considered an input-output "black box". This is again a matter of convention, depending on the chosen symbolic mapping.

[26] The class of so-called *elementary CAs*, according to Stephen Wolfram's classification. For further details about these and many other questions regarding CAs, see section 5.2.

explanation what a CA governed by rule 30[27] does, by saying "the CA produces a progressive variation of the state of its cells, which will follow the pattern of the repeated application, to each cell q, of the boolean function p XOR (q OR r), where p and r are q's adjacent cells". Would this "explanation" be really explicative? Is it intelligible? According to the standards given in section 14.5.1, this is a perfect computational explanation. But who is capable to perform mentally the simulation of the CA in order to understand what "the repeated application, to each cell q, of the boolean function p XOR (q OR r)" means? Although this in not always the case for CAs, take into account that rule 30 is notorious for producing a sort of chaotic behavior: given an example of one of its typical evolutions like that of fig. 14.1, who would claim that the above explanation can be considered satisfying?

But, even if we accepted explanations like the one above as useful, the possibility of expressing a CA rule as a boolean expression fades as the rule becomes more complex. This is due to two reasons: first, usually the CA-rule is not supplied as an explicit boolean expression, but as a lookup table[28], and, it must be also considered that the complexity of CA rules grows exponentially with the number of cells the rule takes into consideration, thus rules can become immeasurably more complex than simple rules like rule 30. Given that, even if a boolean expression can, in principle, be deduced from the CA-rule table, its boolean expression can be huge, depending on the lookup table. It is then necessary, in order to use it as an explanation, to find the *minimal* boolean expression corresponding to the lookup table, but this turns out to be a computationally hard task[29].

All the above suggests us that we should seek *higher-level* explanations, possibly multilevel computational explanations, in order to have a *useful* explanation. To obtain that, we should be able to recognize the CA as a computationally-capable machine at a level higher than that of its elementary cells. So, we must search inside the CA's dynamics certain classes of modules, that is, persistent enough macrostructures, whose behavior at the macrolevel can be seen as rule-governed. These are the basic modules which we are to use as the high-level computationally capable system we are looking for, a high-level system which can be seen as different from the DDS constituted by the CA and its rule. In other terms, in order to obtain a useful computational explanation of a CA, a first condition is (i) that the CA dynamics be *plausibly* considered as a computation at a level which is *higher* than that of its elementary cells: a form of dynamical *macromodularity* must be detected in the global dynamics of the CA. At this point another condition must hold: (ii) the high-level modular dynamics must successfully *track* the low-level dynamics of the CA, without diverging from it. This condition of *validity* (to use the terminology of scientific computer modeling I adopted in section 6.6) is a quite complex one and is better specified in sections 2.2.1 and 6.6.8, but it basically amounts to this: that the dynamics of the high-level description must conduct, after a certain amount of time, to the same state, described at high-level, which would have been reached by the low-level description after the same amount of time. In other words, that the dynamics of the macrodescription must not diverge in time from the correspondent microdynamics.

Fortunately, certain CAs are endowed with such a form of higher-level robust modularity[30]: there are CAs which can generate *gliders* (see fig. 1.2), which end up realizing, in many cases,

[27] Again, see section 5.2.

[28] Similar to *truth tables* in boolean logic. See, again, section 5.2.

[29] The most used precise algorithms, like the Quine–McCluskey method, are exponential time.

[30] Actually, glider-based modularity is not so roubust: it holds given certain conditions in the initial configuration, and gliders are not very robust to perturbation. Nevertheless, in certain systems this form of dynamical modularity is sufficient to enable the system to perform computations. See sections 5.2.2 and 5.2.3 for an in-depth discussion.

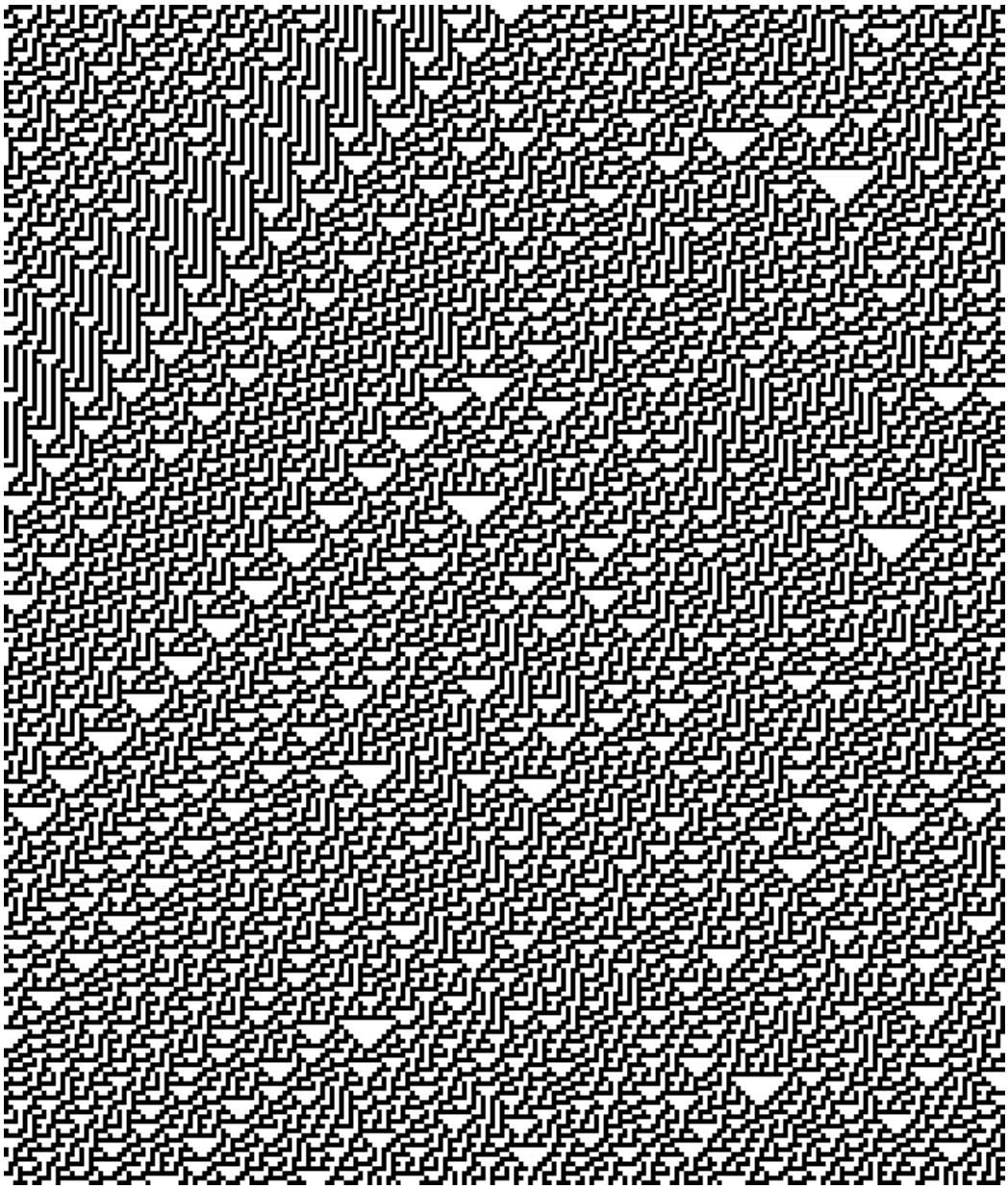

Figure 14.1: chaotic segment of evolution of the elementary CA Rule 30. Time flows from top to bottom, each row of pixels representing the global configuration of the system at each timestep. Each pixel represents the state of one of the elementary parts of the CA, its *cells*.

predictable interactions one with the other, as in the case or Rule 54[31], and these predictable interactions can be seen, as we have discussed in section 5.2.4, as the high-level implementations of boolean functions, with gliders acting as traveling "bits".

This interpretation in terms of gliders is not always possible, however: certain "chaotic" CAs, like rule 30, never show (see again fig. 14.1) subconfigurations robust enough to be considered dynamical modules able to render the high-level representation computationally capable. A brief aside must be made here. A possible objection to this alleged incapacity of rules like Rule 30 to perform computations could be raised: why can't we contrive a mapping from sets of chaotic configurations to meaningful symbols, this way rendering even a chaotic CA computationally capable at high-level? An answer implies a discussion on the complexity of the mapping between system configurations and symbols, a discussion which cannot be fully developed here. Basically, I agree with Martin Schüle[32] on the following constraint: that the mapping between symbols and system configurations must not be so complex as to require a too powerful computation in order to do the mapping. Schüle argues, rightly, in my opinion, that, should we require Turing machine-level computations (which is the class of most powerful computations) to do the mapping from chaotic configurations to meaningful symbols, then we could suspect that the actual computation which the resulting system we will thereby be able to see as performing, could in actuality not have been carried out by the CA with its mapped configurations, but almost exclusively *by the mapping itself*, which, as said, would be a Turing machine-level computation. The constraint to impose is then that the computation required to do the mapping should be less powerful (that is, the machine required to compute it should be of lower degree in the Chomsky hierarchy[33]) than the computation we require the system to do, once the mapping is established.

There is an important point here: this impossibility to individuate, without an unacceptable complex mapping, stable dynamical modules in a CA (like in the case above or Rule 30) can be seen as a form of *intrinsic antimodularity* of the high-level description of the CAs. This is where antimodularity, in this form, already prevents this first step required to provide a computational explanation, the step which allows the CA to be seen as computationally capable at a high level. So, it seems that, at least in this form, *intrinsic antimodularity actually prevents computational explanation.*

But what about CAs which actually *can* be viewed as computationally capable at a high level? It is actually sure that, for certain CAs, mostly in Wolfram class IV[34], their high-level interpretation as computing systems *is* possible: there is a complex mapping, devised by Matthew Cook[35], with which he has been able to prove that rule 110, another elementary CA, can be seen as a computational system on the level of the universal Turing machine, that is, of the highest computational power. Also the most famous CA, John Conway's *Game of Life*, has been proved to be Turing complete[36].

So, it is a proved fact that, under certain interpretations, some CAs can be seen as computing. But, as I highlighted in section 14.5.1, if we want to give a *computational explanation* of a system, another condition must be satisfied: that the system is actually *computing*, and not just that is computationally *capable*. To fulfill this condition, we must first be able to say *what* it is computing: that is, we must be able to express its input/output relationship, its *specification*. We must note, though, that we are here in the position not of software engineering, where one

[31] See for instance Martínez et al. (2014), and section.
[32] see Schüle (2014).
[33] See section 17.2.9 of the Appendix.
[34] See section 5.2.1.
[35] See Cook (2004).
[36] See Rendell (2002).

starts with a specification in order to implement it, but that we are working in the *reverse-engineering* field: we have a machine, the CA, which we know that is computationally capable, and we should, in order to computationally explain it, *produce* its specification.

We should then attempt the reverse engineering of the specification. This is not an easy task, as explained in section 4.3.1: the set of all possible inputs can be infinite, or, if finite, is often too vast to allow for every possible input to get supplied to the program, in order to allow for an inference of the complete program specification. Even in finite input sets, the number of possible input combinations grows exponentially. So, we encounter here an obstacle: it is hard to obtain the exact specification of an unknown program. Actually, if the program operates a computation of Turing-machine level, it is even in principle *impossible* to be sure in advance if we will be ever able to complete the reverse-engineering of its specification, because that would be equivalent to solving the halting problem: if a program never halts for a certain input we will not be able to complete the reverse-engineering of its specification, but for the undecidability of the halting problem we cannot know in advance if it never halts for certain inputs[37]. This means that doing the reverse-engineering of the specification of a potentially Turing-complete CA is more or less a hopeless task.

But we needed the program's specification in order to computationally explain it. And this specification is very hard to infer.

However, a specification in the form of the mere extensional listing of the input/output function, is not the only way a specification can be given. Moreover, this is not the most perspicuous way to give a specification, even if it is the most precise: a list of input/output couples can be meaningless. Another, more perspicuous form in which a specification can be given, is in an *aggregate* form: a more or less synthetic way to sum up the whole input/output function. There are several ways to aggregate a specification, which I try to classify in section 4.1.5.2: I refer the reader to that detailed treatment, but suffice to say here that one of these ways is that of giving the specification in terms of its decomposition in subfunctions. This could be seen as a form of hierarchical decomposition.

There is a big advantage in being able to produce a *hierarchical*, *modular* representation of a given specification of a computation: if such a representation could be devised by some means, it would be possible to put to test each module *separately* in order to seek for the specification of *only* that module. This task would most likely turn out being more feasible *by order of magnitudes* than that of submitting every possible input to the whole program in order to directly infer the global specification, because a module is identifiable by the very fact that it should be only loosely or sparsely connected to the other modules, and this translates in a probable reduction in the number of possible inputs to the module, and a consequent easier exploration of that module's input space[38].

The fact that it has been possible to find the single specification of each module due to the system's decomposability, hopefully allows, if the specification of each module is not too complicated, for a form of *aggregation*, as discussed in section 4.1.5.1: if we are able to further abstract the module specification by "naming" it in a meaningful succinct way[39], giving the module a name which is representative and explanatory of the function it performs (as for example when we say that a module performs the "multiplication" operation), then each module's specification can be substituted by this more concise definition of *what* function the module performs. Then

[37] See details in section 4.3.1.
[38] Even if this is not *guaranteed*. See section 4.3.1 for a better discussion.
[39] This could be done along the lines which are highlighted in section 4.1.5.1.

a global specification of the whole system can be given *in terms* of a description (usually in the graphical form of a *flow chart*) of the modular structure of the system as a directed network of connected modules, where modules are seen as nodes labeled with their succinct "names" representing their specification, and their input and output connections are the directed links between nodes.

The interesting point of what is described above is that the modular structure of the software, if present, has allowed for a succinct and computationally treatable reconstruction of the global specification of the program. This specification is given not in its explicit, extensional form, that is, the list of all its possible input/output couples, but in the form of a hierarchical functional *explanation*. We were dealing in this section with reverse-engineering of software, but this has given us the occasion of describing a notion which is exactly similar to the notion of *functional explanation* typical of certain scientific disciplines, a notion which is better expounded in section 9. Computational explanation, as employed, for example, in cognitive psychology, is exactly a type of functional analysis.

So, this type of explanation seems possible, after all. But *it requires that a functional modularity of the computation can be found*, and this, in turn, requires two conditions:

1. that the system is actually *computationally capable* at a *high level*: this is not guaranteed, because *intrinsically antimodular* systems, like the CA Rule 30 hinted at above, are not even susceptible to be seen as *computationally capable* at high level;

2. that, even if the system is computationally capable, and actually possesses dynamical modularity, this modularity can be actually *found*. This finding could however be hindered by some factors: even if the computationally capable process *is* modular, how could we recursively decompose its still unknown implementation into modules? There are dynamical methods to infer program modularity from the observation of the program's behavior (I refer to section 4.3 for a partial survey on such methods). However, most of them are affected by strong limitations, due to too high computational time complexity, or limitations in principle, due the the halting problem, on what can be reconstructed, and for these reasons these methods are able to operate only a partial, approximate reconstruction of the program's dynamical, functional modularity, which is the representation which can be used as computational explanation. To sum up all these conditions, here again *antimodularity* can appear, in the form of a pragmatic impossibility, due to the excessive computational cost of the modularity detection algorithm, or of its excessive use of approximated methods for modularity detection (resorted to in order to eschew computational cost), which makes the algorithm unable to produce a *valid* modular description of the computation, but only a partial one.

But, could partial reconstructions of the program's functional hierarchy still be used in explanations? Well, it seems, intuitively, that the functional models so produced would be very constrained by *ceteris paribus* clauses, in order to keep them inside the range of *known* input/output relations, and to not make their lack of validity emerge. So, it seems that an explanation based on them should also be so constrained in its applicability. It could appear as an acceptable explanation, but it would possibly, depending on the degree of its lack of completeness, possess low predictive power about response of the program to novel inputs, because in the worst cases such a hierarchical model would in a way be only a *post-hoc* explanation of the behavior of the system already observed during the process of modularity detection.

It may well be that, in computational cognitive science, such a kind of limited explanation could be accepted, and, moreover, it is likely that there are *only* this kind of explanations in some subfields of cognitive psychology. In that science, the task of finding the specification and the functional relations between modules, is left to human experimentation, and that is most probably a slower process than algorithmic ones.

To conclude this section, I propose to consider this failure of computational explanation due to antimodularity, a form of *computational emergence*. This computational emergence regarding computational explanations can be seen as due to two forms of antimodularity: the first occurs when the system is too chaotic to be recognized as modular, and it lacks the features needed to be considered computationally capable at a higher (or at an adequate) level of description. The second, when the system is too complex to be decomposed, by recursive specification mining and subsequent program modularity detection, into a valid, that is, non-partial functional hierarchy able to computationally explain the system.

When this form of antimodular emergence shows up, the system, even if it is actually modular, cannot be described as modular, and thus it is only explainable at its lowest level. In the case of a CA, this would be the cell level. But, as we have seen in the example above, this would probably not constitute an intelligible form of explanation. When the system showing antimodular emergence is an actual program, written in a known programming language, then it can be represented under the form of the ordered listing of the programming language instructions constituting it, but this list cannot be sensibly decomposed into modules in order to better explain it in a more intelligible way.

Maybe the most notable consequence that a form of antimodular emergence afflicting computational explanation can be expected to have, is upon the possibility of explanation of certain cognitive phenomena, both in cognitive science and in neurosciences. This could well be the object of further study.

14.6 Explanatory emergence

Given that lack of understanding due to the presence of antimodularity in a system can seemingly affect most kind of explanations, I propose the following definition:

> *explanatory emergence* is a property of systems or descriptions of them that consists in the fact that, for absolute or pragmatic computational reasons, they resist *understandable* explanations.

This definition is much more general than that of antimodularity, and, in it, the term *computational* can be understood in a very wide sense, including procedures which, although not performed by computational machines, are performed by following "algorithmic" methods, where *algorithmic* is to be understood, in a loose sense, as a more or less precisely defined method which has to be executed with an acceptable degree of rigor.

This is still a sketched definition, but the main idea behind it is to capture difficulties in explaining complex phenomena not necessarily due to impediments to *modularity* detection, nor necessarily to lack of *modularity* alone, but due to any automatic or manual procedure, whose product is a representation supposed to be essential for the scientific explanation of some phenomenon, which requires too much time to be feasibly brought to an end, or which, while susceptible to be completed in a reasonable time, produces nevertheless too approximate results to be of use for scientific explanation.

Such a broad definition would cover, for example, cases like the identification, by experimental observation, of the detailed mechanisms of some complex phenomena, such as the reconstruction of gene regulatory networks, or of the functional organization of nervous systems, which can require long and tedious series of piecemeal experimental observations, often conducted in the course of decades. Of course, this definition covers also all cases of *antimodular emergence*, and other computational methods in general, such as methods of *data mining*, which can be necessary in certain research areas, like genetics or cell biology, to extract, (to *mine*) valuable information (not only modular structure) from huge datasets which have already been collected and are already available.

Chapter 15

Are there antimodular systems in science?

Antimodularity appears to depend on the *choice* of a relation between the elementary parts of a system, a relation which is used to assess the system's modularity[1]. When assessing modularity by means of an algorithm for modularity detection, antimodularity can occur in two cases: (*i*) when, given this chosen relation, modularity detection according to it turns out to be too computational demanding to be brought to completion in a feasible time, or (*ii*) when, although modularity detection is successfully completed by means of an approximate algorithm, the produced modular description appears *too approximate* to be capable of *validly* representing the original system.

A brief parenthesis must be opened here: it might seem that the two cases above forget to take into consideration the third possibility of *intrinsic* antimodularity. Actually, this is not the case: let's say we know that a system is intrinsically antimodular. How can this antimodularity show up, when performing modularity detection on the system? There are two possibilities: *a*) the system is too large for the algorithm to complete the detection in a feasible time, and we stop the algorithm before it ends its task, or we even renounce launching it, for we know, based on the system's size, that it would take too long; *b*) the algorithm comes to a natural stop in a reasonable time, but it produces a modular structure which is not *valid*, in terms of dynamical tracking of the original system. Now, the intrinsic antimodularity of the system could show up in either of these cases: in the first implicitly, in the second through the fact that the algorithm has not been able to find a sufficiently valid modular redescription of the system. Given that we know that the system is intrinsically antimodular, we can attribute the algorithm inability to the presence of intrinsic antimodularity: being devoid of modularity, it is normal that the algorithm has failed in finding a minimally valid modular description.

Now, it seems to me, we must evaluate the likelihood that circumstances such as the one expressed above (*i* and *ii*) can be encountered during scientific research. It must be stressed that computational complexity of modularity detection concerns algorithms for detection of modularity which do not employ any other information about the systems than those included in their preferred description, that is, the level of their elementary parts and their relations. By adding constraints on how the elementary parts can be grouped into modules, the task can be highly simplified. This is equivalent to devising ad hoc algorithms for modularity detection, and ad hoc algorithms could turn out being less computationally demanding than generic ones. For example, as already highlighted in section 1.1.5, in the case of the search for modularity in a

[1] I am talking of *system* here, but I am of course actually talking of its *preferred description*.

genetic network, the added information about groups of genes which always co-express could ease the task of grouping genes into modules. Upon brief reflection, it seems this is exactly what science does: it searches for empirical constraints to help us choose among the possible theories of the world, which, per se, if unconstrained, are unbounded in number. It is then likely, it seems, that scientific research produces modular descriptions of the empirical phenomena which it observes. Or at least it has until now proceeded this way. Could new developments in science make research focus on systems of such a complexity that even the known, empirically found constraints about them end up being too few to allow the successful completion of modularity detection on such systems?

Let's see: if these are biological systems, there are interesting arguments which aim to prove that evolution *must* have produced modular systems, as explained in section 7.1. If these arguments hold, I think intrinsic antimodularity can be considered quite unlikely in biology. But, I also think that while Kauffman's argument and similar ones appear convincing[2], it seems to me that arguments based on natural selection[3] are weaker, because the role of natural selection in shaping organisms, and so in shaping something so clear-cut as modular structures, has been in the last decades seriously questioned (it' s a long-standing debate into which I do not want here to explicitly take side). Nevertheless, most biological systems appear, to a close inspection, undisputably modular. The brain itself is very different from a mere bunch of interconnected neurons, and possesses a clear modular and functional organization, with clearly distinguishable parts, segregated connections between them, preservation of mapping between modules, and other evident features revealing a high degree of modularity. The cell also appears highly modular, but here modularity of certain complex networks of interacting elements is certainly less evident, and, as we have seen when considering some case studies, it is now normal to recur to *computational* methods for detecting modularity in such systems, for example in the genetic regulatory network. Systems of this kind are the preferred object of a recent biological discipline, *systems biology*. I suspect that this discipline could soon focus its research on systems of such a complexity that the algorithm for their modularity detection could fail, due to the computational complexity of the required task. There are signs of this: certain studies explicitly admit that the size of the system, upon which modularity detection has been tested, had to be of limited size, because otherwise the algorithm would have taken too much time. For example, Sales-Pardo et al. (2007) acknowledge:

> The computational cost of this step, the slowest one in our algorithm, limits network sizes to $\sim 10,000$ nodes. However, the cost can be reduced by using faster, but less accurate, methods for ordering the matrix, such as principal component analysis[4].

Here we see the trade-off between accuracy of modularity detection and its cost. These are *already* cases of antimodularity at work.

Another type of scientific endeavor in which, it seems to me, antimodular emergence or explanatory emergence is around the corner, is in data mining of the already existing scientific literature[5]: this is a type of meta-research which, due to the sheer size of the data to process, could well be affected by antimodular emergence, or in general, explanatory emergence.

To give a provisional conclusion, there are signs that antimodular emergence, and in general explanatory emergence, are around the corner in certain scientific fields. And there is the risk

[2] See section 7.1.2.
[3] Section 7.1.4.
[4] Sales-Pardo et al. (2007), p. 15227.
[5] See an example of this kind of research in section 1.5.2.

that, once appeared, they can be there to stay. This phenomena could well hurt scientific research, but are at the same time the sign that science has recently expanded, with the aid of powerful computational methods, into new territories: the fact that these same methods begin to show intrinsic limitations in their power is the consequence of the expansion of the scope of science that they have allowed to begin with. Further theoretical and empirical research is needed to confirm or reject the hypothesis that these forms of computational emergence can damage scientific research: algorithms are continually improved, devising ways to circumvent, at least partially, computational intractability. But intractability results are absolutely objective limitations and it cannot be envisioned any way to definitely escape them.

Final Remarks

Chapter 16

Summing up

Before bringing this work to an end, I think a summing up is due of all the long line of inquiry and reasoning which structures it: the scope covered has been vast, and quite certainly too vast to have been covered adequately. A brief rehearsal of the path followed could certainly better outline the main ideas that I intended to convey.

In this work, I was mainly concerned with the notion of *hierarchical modularity* in complex systems, its algorithmic detection and its use in *explaining* structure and dynamical behavior of such systems, an explanation provided by means of hierarchical modular representations. Specifically, I highlighted the pragmatic influence of hierarchical modularity on the possibility of *scientific explanation* of complex systems. Inside a proposed epistemic stance toward explanation and descriptions, I consider a *system* what, according to a chosen basic description, can be considered as composed of elementary, discrete, interrelated parts.

I applied the notions of modularity and explanation mainly to *descriptions*, that is, theoretical *models* of empirical systems, leaving aside for the moment the thorny question of the relationship between an empirical system and its theoretical model. In my view, which regards scientific explanation as an epistemic matter, which involves human understanding and communication, descriptions are *chosen* by the researcher according to her explanatory aims and purposes: this highlights a pragmatic aspect which in my opinion is constantly present in scientific explanation.

Modularity, a well known notion since his first conceptualization by Herbert Simon in the 1960s, basically manifests itself as the possibility of *decomposing* the system into recognizable, sufficiently defined and persistent subsystems (the modules) each one composed of parts which are more strongly related to each other than to parts belonging to other modules. Hierarchical modularity manifests as the possibility of describing a system as a full hierarchy of levels, each level composed of loosely interrelated modules, and each module in turn possibly decomposable into sub-modules belonging to the lower level.

Algorithmic detection of hierarchical modularity has turned out being plagued by an in principle computational intractability (NP-completeness) affecting the search for the best possible hierarchical description of a system, and in any case by a quite high computational cost of the known approximate algorithms, circumstances which hinder the applicability of techniques of modularity detection to systems exceeding a certain size.

I proposed the notion of *antimodularity*, which consists in the lack of a feasible or reliable hierarchical modular description, a lack due either to absence of modularity in the system's basic description, or to the impossibility, due to the size of the system under assessment and the

computational cost of algorithmic methods, to algorithmically obtain a hierarchical automatic modular description which fits the needs of the observer.

It must be stressed that both modularity and antimodularity are *relative* to the *choice*, on the part of the observer, of a particular description of the system: varying the basic description of the system, its modular structure can consequently vary.

I also showed that antimodularity, in systems complex enough, entails a form of so-called computational emergence, namely Mark Bedau's *weak emergence*, which amounts to the impossibility of predicting the outcome of a complex system's dynamics without resorting to the slowest, lowest level simulation of the system.

I assessed the consequence of antimodularity on four models of scientific explanation, namely mechanistic explanation in the epistemic sense of William Bechtel and Cory D. Wright, the classic Hempel-Oppenheim deductive-nomologic model, the computational explanation typically employed in cognitive psychology, and a so-called *topological* type of explanation, recently proposed by Philippe Huneman. I concluded that antimodularity, by impeding the obtainment of a full hierarchical description, negates the possibility of multi-level explanation, thus damaging understandability of the explanation by allowing only for description of the system at the lowest level, the level of elementary parts, a description which, in sufficiently wide systems, can result unintelligible because of the sheer number of interrelated parts which it describes.

As a consequence, antimodularity mostly damages mechanistic and computational explanations, which rely on multiple interrelated levels of descriptions and require intelligibility. By entailing weak emergence, antimodularity impedes deductive-nomological explanations, which rely instead on the availability of a *law* allowing for prediction of the system. Nevertheless, some antimodular, weakly emergent complex systems, such as cellular automata, which possess a law-like rule holding everywhere inside the system, could be explained by a sort of reiterated deductive-nomological explanation, which, quite likely, would lack understandability due to the high number of virtually meaningless reiterations of the law-like rule constituting it. Topological explanation, on the other hand, seems to be immune from consequences of antimodularity, which, being a topological kind of property itself, should even in certain cases *enable* this kind of explanation.

I subsumed under the concept of *explanatory emergence* all the results about the unfeasibility of certain multilevel explanations due to antimodular emergence, and on the consequent fading of understandability due to occurrence of antimodularity.

It must be stressed that hierarchical and high-level modularity is needed not only for *a posteriori* explanation of a known phenomenon, but also during the phase of scientific discovery, specifically, as already noted by James Woodward, during the search for casual relationships between parts of a mechanism either at low and and at a higher level, and in general in planning and guiding ongoing research towards the most useful scopes. Likewise, multilevel modular explanation is also essential during the development of computer programs on the part of human programmers. Modularity in networks has also showed being potentially of great aid in guiding the progressive discovery of the structure of very large networks, like networks (genetic, metabolic, proteic) of crescent biological importance and at the focus of attention of systems biology.

After having assessed the possible consequences of antimodularity on some sciences, I discuss, by examining some scientific literature, the likelihood for scientific research to incur antimodular emergence, concluding that it is quite likely that some cases of antimodularity appear, especially in systems biology.

The above summarizes the main line of reasoning I wanted to follow, as it has been unfolded in the former chapters. This constitutes the main scope of the present work, which is mainly a work of philosophy of science.

However, I think it is worth mentioning here some preliminar, or, better, experimental reflections I put forth at the end of chapter 1, which, albeit probably a little hasty, manifest I think a great potential of further, refined, future development.

First, I sketch, without entering full discussion, a possible metaphysical view that could stem form the considerations of antimodularity exposed before: I call this view *constrained antirealism*. It sees the empirical world we naturally perceive, as well as the world described by science, as the result of a process of modularity detection, whose detected modules constitute what are commonly known as *natural kinds*. The main point is this: given that modularity detection is constrained by factors of insurmountable computational complexity, and that for this reason the finding of the best modular description is in principle precluded, it seems not likely that the world's actual subdivision in natural kinds could correspond to is best possible subdivision. I briefly debate also an evolutionary objection based on adaptation of perceptual systems as warranting natural kinds: I claim that neither science nor natural selection could have had the time, even on the geological scale, to produce an optimal ontology of the world, because this task is in principle computationally intractable.

From a more metaphysical point of view, we could produce other considerations. In the course of the analysis I carried out here, modularity has turned out to be an almost all-encompassing notion: the very idea of modules has revealed to be universal, at least in most human epistemic activities: it is very difficult to perceive, to communicate, to understand and gain knowledge without making use of modular models of the real phenomena we are talking about. For its universality and its main features (those of being in a certain degree delimited, distinct, partially isolated from the context, and at the same time related with other modules), it is possible that the idea of module conflates in many ways with that of *object*. In this case, modularity, probably, can be seen as conflating with *ontology*. But, I would refrain from such a facile identification, if only for the reason that doing metaphysics has always seemed to me a rather risky business. My personal inclination goes towards, if not a radically antimetaphysical position, a light metaphysical stance. In this view, I tried to propose in section 1.5.1 a vision which, instead of reducing modularity to ontology, tries to reduce ontology to modularity detection an to its limits: this way, a heavily ontological question seems to be relieved of its weight and brought in a more epistemological terrain. As said, this is only a roughly sketched position: it defers a serious discussion of epistemological (or, rather, ontological) constraints on our possible perception and detection of modularity to other ongoing debates, such as those on the nature of mathematical facts and that on pancomputationalism. It may well be that such a move is a bad one, that renders my position well more heavy than it seems. As said, all these questions will have to be pondered in another occasion.

Metaphysical considerations aside, in section 1.5.2, I took some liberty in drawing the possible, alleged consequences on history of science of a recent and growing recourse to computational methods in scientific research, starting with the simulation of complex systems: I reflected on the plausibility of simulations as explanations, especially in cases in which the system is anti-modular, and, consequently, simulation can be executed, but the underlying dynamical model is unintelligible, because, due to antimodularity, the system must be simulated at the lowest possible level for lack of a modular high-level redescription. I then considered automated modularity detection used in order to find structure in big datasets, basing the discussion on real cases of data mining on a corpus of medico-biological literature, in which the automated system discov-

ered important functional relations which had escaped human examination. Possibly indulging in drawing some bold consequence, I concluded by suggesting that this growing use of computational methods in science could have already provoked, or can well be on the verge of provoking a major shift in certain disciplines: especially in some fields, related to biology, new techniques for automatic collection of data have started to produce huge databases of experimental data which are badly in need of being scrutinized in search of an interesting macrostructure (mainly *modular* macrostructure), which does not show up by itself, given the sheer amount of records involved. Computer and programs can come to the aid, operating a sort of "Baconian" method, and while this could open a host of unforeseen and exciting discoveries, I suspect it could also come with some side effect: such a partially automated science could soon involve a problem of at least partial loss of intelligibility with respect to the human observer, unintelligibility affecting both the theoretical explanation so obtained, and, given the role of theories and models in guiding further experimentation, the path which that research could begin to follow, which is a path which could even begin to appear to us progressively obscure. While I think it is still early to see the shape of things to come, it seems to me that the mere envisioning of this change in scientific research as happening, constitutes a novel, unprecedented, and potentially game-changing situation which philosophy of science and history of science should not forget to point their attention on.

The subject matter of this work is multifaceted and not easy to label: being about the consequences of antimodularity on possible scientific explanations of certain systems, it is a work of philosophy of science. Given that the proposed property, antimodularity, depends on certain computational constraints affecting modularity detection, and that I recommend, in relation with the discussion on computational explanation, an intentional conception of what computation is, then this is a work of philosophy of computing, in the double sense of putting certain computational notions to philosophical use, and of proposing a philosophical reflection on the notion of computation itself. Considered that systems showing antimodularity are likely to be found amongst biological systems, the long-standing discussion in biology about modularity, and a host of examples I report from that discipline, it is in a way even a work of philosophy of biology. Finally, it doesn't refrain to draw some metaphysical and historical conclusions from the discussed notions and their consequences, consequences in which computing and computers play a role of ever increasing importance.

Appendix

Preamble

In the first chapter of this Appendix, section 17, I will try to give a concise characterization of some basic classic concepts and questions which make up the subject of *Computer Science* (*CS*). A basic knowledge of this subject is required, to allow a better comprehension of the arguments in the former theoretical philosophical chapters. I have tried to conform to standard terminology, so the reader who is already well acquainted with the subject matter can safely ignore this appendix. I am going to pass over many subtleties and fundamental mathematical results which do not concern us here, trying to give an essential but precise enough account. Should the reader find this exposition lacking in clarity or incomplete, my suggestion is to consult one of the many good introductory graduate-level books to the field of computation theory or theoretical computer science[1].

The second chapter, section 18, is a substantial overview of the contents of the thesis for readers of French, basically equivalent to chapter 1.

The third chapter, section 19, is an analogous overview of the contents of the thesis for readers of Italian.

[1] For example Sipser (2012), or Lewis & Papadimitriou (1998), which I'm partly following. For the part on computational complexity, also Garey & Johnson (1979), Bovet, Crescenzi, & Bovet (1994), and Arora & Barak (2009).

Chapter 17

Computer science basics

17.1 General notions

A ***state*** is the particular condition in which a machine is in, at any given moment.

An ***alphabet*** is a finite set of symbols[1].

A ***string*** is a finite sequence of symbols taken from a given alphabet.

A ***language*** is a set of strings.

An ***abstract machine*** is a theoretical (often formal) model of a a computing machine, used in mathematical arguments on the limits and possibilities of computation. Starting as a representation of a material mechanism, such a model neglects any physical limitation of the device, to present the core mechanism in an idealized way, free of time, space and other physical limitations. The paradigmatic example of an abstract machine is the so-called Turing machine, as presented by Alan Turing (1936), which I am going to explain later in the chapter.

Historically, the theory of computation begins with the aforementioned work by Turing, in which the author proposed an abstract machine model[2], while outlining a mathematical theory of the computational capacity of this machine along with a theory of the *limits* of its computational ability. This work set up the theoretical framework in which much of the subsequent work in the field would have been carried on: a framework in which much importance is placed upon the demonstration of the computational ability of a class of machines, but equally upon demonstrations of its computational limits. I will dedicate some brief paragraphs in this chapter to some different types of machines classified by computer scientists[3].

17.2 Automata theory

17.2.1 Finite automata

A *Finite Automaton* (*FA*) is an abstract machine with a finite set of states, a finite set of possible input symbols (the *input alphabet*), and a *transition table*. The state the machine is currently

[1] I'm not analyzing in this section the notion of *symbol*. Some reflection on the nature of symbols is carried out in section 5.1.2.

[2] this abstract machine, the Turing machine, is to receive due treatment later in the chapter.

[3] I will not treat here many type of abstract machines, the knowledge of which is not necessary for the rest of the dissertation. For example, I will skip over nondeterministic deterministic pushdown automata, for the reason that their exposition is not necessary here.

in, is called the *current state*. If certain conditions hold (basically, activation of some input), the machine can switch from the current state to one of the available states (the *next state*), according to the instructions contained in the transition table, which pairs current states with possible inputs, dictating which next state the machine will enter. There are two special states: the initial state and the final (or *accept*) state.

During the machine's functioning, a finite sequence of symbols taken from the input alphabet is presented to it. When the input sequence is completed, if the machine turns out to be in the *final* state, it is said that it *accepts* its input, otherwise that it does not accept it. Every specific different machine characterizes the strings of symbols which it accepts. As we have seen, in CS terminology, a series of finite strings of symbols is called a *language*. The set of strings of the input alphabet accepted by a specific machine is called the *recognized language* of that machine. A specific automaton recognizes only one language.

A FA can typically be described by a *state diagram*, like the one in fig. 17.1.

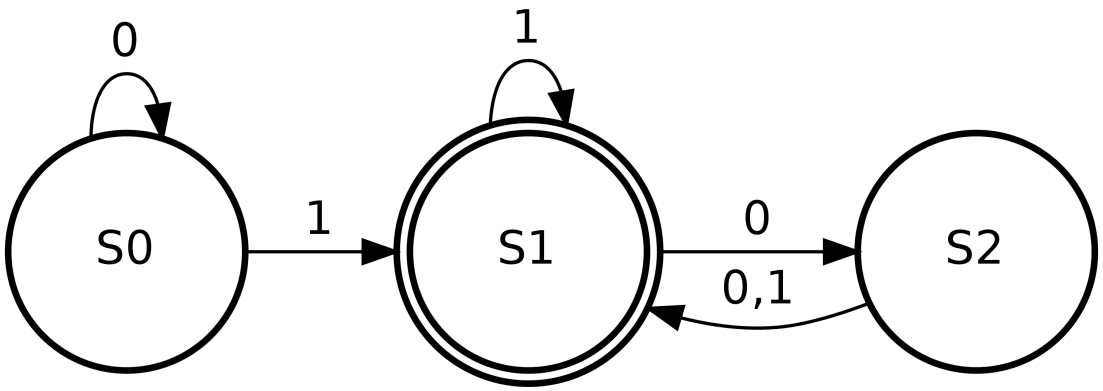

Figure 17.1: state-diagram of a finite automaton.

The machine represented in this diagram can be fed with finite input strings from the alphabet $\{0,1\}$. The diagram can be read as follows: the machine starts in state *S0* (the initial state). If it receives a *1*, it switches to state S1 (arrow labeled with "1"). If it receives *0*, it switches to state *S0* (arrow labeled with "0", pointing to *S0* itself: in this case, the machine remains in state *S0*). And so on for other states and inputs. The state S1, drawn as a circle with a double border, represents the final state. It turns out that this particular machine accepts strings ending with *1* or with an even number of *0*'s, for only in these cases it ends up resting on the S1 state after the input sequence is over (if it received an odd number of *0*'s it would rest on *S2* instead, which is not an accept state). In other words, the machine accepts the *language* constituted by the finite strings composed by "1"or "0" symbols and comprising an even number of *0*'s or ending with *1*.

As typical in the theory of computing, a special importance is placed upon the class of computations that a FA can perform. This is equivalent to inquiring on which *languages* the class of FAs can *accept*. FAs can accept a class of languages which are called *regular languages*. These are quite simple languages, and many obvious strings do not belong to them. For example, a language L_d containing strings of any length, each string composed by two same-length sequences of two kinds of symbols (say, for example *aaaabbbb*, or 111111000000) is *not* a regular language, in that there is no FA that can recognize it[4]. Intuitively, this limitation comes from the fact that,

[4] I omit the proof here. See Sipser (2012), p. 80.

to accept a string of such a form, it must be checked if the substring composed of symbols of the second type is of equal length to the substring composed of symbols of the first type. Since the input is to be presented to the machine sequentially, this means that in order to perform this check, the automaton must "count" the symbols of the first substring, store this number, then count the symbols of the second substring, and compare this new number with the former, accepting the string if they are equal. Given that no limitation in principle is placed on the total length of the strings composing the language L_d, it turns out that the machine has to keep "in memory" at least the first number, and given that this number can grow arbitrarily, and that to keep a stable information inside the machine this has to be encoded in a machine *state*, this means that this kind of language cannot be recognized by any automaton with a *finite* number of states, thus not by any FA.

Intuitively, the set of languages recognized by a machine characterizes its computational capability: in the case of FA, there are strings which it cannot recognize, however complex the structure of the FSA can be conceived. But, as we will see, there are other, more capable kinds of machines, which can recognize these languages. As we will see, there is a hierarchy of machines , called the Chomsky hierarchy, in which the FAs are the least powerful ones.

An obvious variation on the FA model is a finite automaton with *outputs*. There are two classical form of finite machines with outputs: *Moore machines* and *Mealy machines*. Their difference is not relevant in this context. Suffice to say that they are finite automata in which a transition, in addition to putting the automaton in a new state, produces one or more output symbols. The difference with classic FAs resides in the fact that these automata, seen from an external point of view, can produce an output *before* the input sequence is completed, acting this way as *functions* that process the input string into one or more output strings. These finite machines with output have the same exact computational capability of FAs, and, used as recognizers, will recognize exactly the same languages.

17.2.2 Nondeterministic finite automata

The machines I described until now are *deterministic*, in the sense that, for each state the machine is in, given a certain input symbol, the transition table dictates exactly one and only one state (the *next state*) which the automaton is bound to switch to.

By contrast, in a *Nondeterministic Finite Automaton* (*NFA*), given a certain input, the transition table can[5] suggest *more than one* next state to which the automaton should switch. The state diagram of a typical NFA is thus characterized by different arrows with the same symbol as label starting from the same state and going to different next states. For example, see fig. 17.2.

To be able to execute its computation, when faced with two or more possible next states, the automaton "splits" in two or more "copies" of itself, each of which will switch into one of the different next states. This splitting occurs again whenever the transition table gives more than one next state. This way the machine follows all the possibilities in *parallel*. The process can be visualized as a ramified graph: fig. 17.3.

When the next presented symbol does not appear on any of the arrows exiting the state a copy of the machine is in, then that copy "dies" along with its branch of the computation, while the other copies go on processing. If any copy of the machine turns out to be in the accept state after the last input symbol is given, the input string has been accepted.

[5] NFAs are a generalization of FAs, so any FA *is* an NFA, in which the transition table happens to give only one next state for each condition.

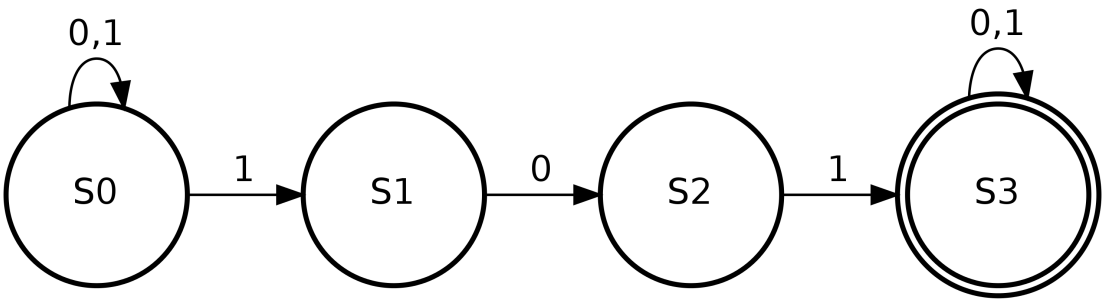

Figure 17.2: state-diagram of a nondeterministic finite automaton.

NFAs don't show more computational power than simple FAs: they recognize the same languages, the regular languages. This means that, for every NFA, there is an equivalent FA. But, the structure of the NFA is simpler: the FA requires more states.

17.2.3 Probabilistic finite automata

Probabilistic finite automata (*PFAs*) are a generalization of NFAs, in which the transitions from a state to another are associated with a *probability* (fig. 17.4).

17.2.4 Pushdown automata

A *pushdown automaton* (*PA*) is a nondeterministic finite state machine equipped with an unlimited *stack*. The stack is a form of memory, in which only the element on top is visible. The heap of elements stacked one on another is pushed down when a new element is added at the top. Conversely, to access an element down along the stack, the topmost element must be removed, making the next element come to light, an so on, till the wanted element emerges. This implements a so-called *Last In, First Out* (*LIFO*) type of memory.

The automaton can, as possible actions triggered triggered by particular input and state configurations, add or remove an element from the top of the stack, and also begin actions according to the visible element at the top. Fig. 17.5 is a typical schematization of a PA.

Pushdown automata can recognize a broader class of languages than regular languages: the class of so-called *context-free* languages[6], which is a proper superset of regular languages. Thus, push-down automata are computationally more capable than FA and NFA. As we will see, the class of context-free languages is not the more comprehensive class of languages, yet: there are languages which the PAs cannot recognize. The limitations in computational ability for the PAs derives from the fact that their memory, although unbounded, can be read only at the top, and old items can be recovered from the stack only after having erased newer ones. This limitation is absent in the most capable kind of computational machines, the Turing machines, which are about to be explained in the next section.

17.2.5 Turing machines

From an historical point of view, in the mathematical community worldwide there had been, since the late XIX century until about the end of the 1930's, a wide and diversified debate about the

[6] See section 17.2.10.

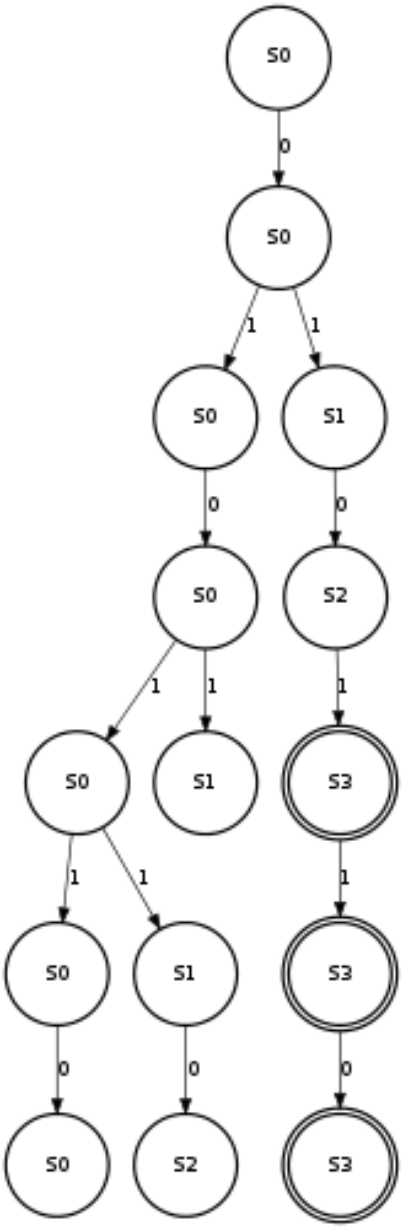

Figure 17.3: representation of the computational process of a NFA receiving the input string *010110*.

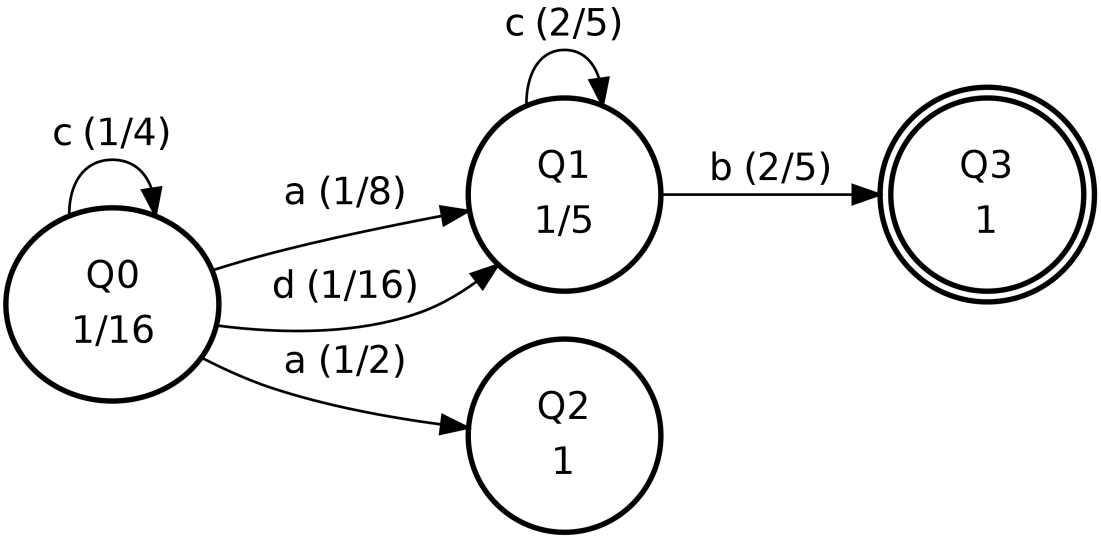

Figure 17.4: state-diagram of a probabilistic finite automaton.

foundations of mathematics. In the context on this debate Alan Turing set out to try to respond to an open question, the *Entscheidungsproblem*. Posed in 1928[7] by David Hilbert, one of the most influential mathematicians of the time and probably the main agent of the then-underway phase of the foundational debate, the Entscheidungsproblem (a German expression which stands for *problem of the decision*) is the quest to devise a general mechanical procedure to detect if a given statement expressed in the language of a formal system is or is not a theorem of that system: that is, if the statement has or has not a derivation inside that formal system[8]. It seems clear that the search for a solution to this problem requires as a necessary step the refining of an intuitive notion such that of a mechanical method, or mechanical procedure or a task that can be performed "by finitely many operations" (as in Hilbert & Ackermann 1928).

In his seminal 1936 paper[9] Turing proposed an abstract machine model, in order to formally define the notion of a *computational* task, a notion that, according to his analysis, ends up corresponding to the kind of mechanical method required by the Entscheidungsproblem. As mentioned above, in the words of Hilbert and in the expectations of the mathematical community of the time, this idea of mechanical procedure had been until then evoked as an intuitive notion: the notion of a task which can be carried on by a human subject following a finite list of simple well-defined rules that do not require resorting to intuition or ingenuity. Thus, aiming at attaining a more rigorous characterization of the concept of mechanical task, Turing proposed the abstract model of a *machine*. This model, the so-called[10], afterward, *Turing Machine* (*TM* henceforth), is basically a finite state machine[11], which in addition can read and write a symbol at a time on

[7] In Hilbert & Ackermann (1928).

[8] To be precise, the question posed by Hilbert is that of the existence of a general mechanical method to demonstrate if a given formula expressed in the language of first-order logic (*FOL*) is or is not a logically valid formula of FOL, that is, the problem concerns the *truth* of a formula. By the completeness theorem for FOL, proved by Kurt Gödel (1930), it follows that the question is equivalent to asking for a method to detect if a FOL formula has or has not a proof (i.e. a syntactical derivation) inside the FOL formal system. After that result, in general the problem has been understood as the quest for this second syntactical formulation.

[9] Turing (1936).

[10] By Alonzo Church.

[11] although coming last in this exposition, historically, the Turing machine comes first, as the first attempt to

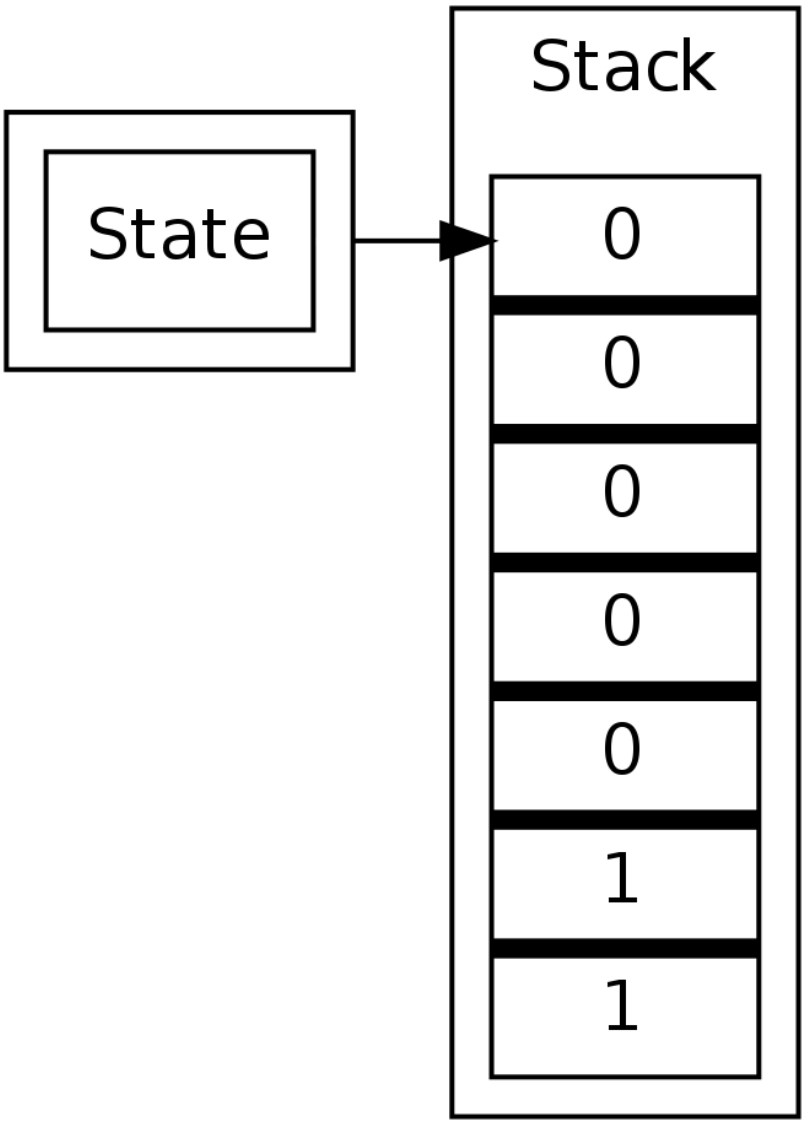

Figure 17.5: schematic representation of a pushdown automaton.

an infinite linear access memory, usually represented as a *tape*, that is a linear row of cells, each containing one symbol taken from a finite alphabet[12]. A typical Turing machine is sketched in fig. 17.6.

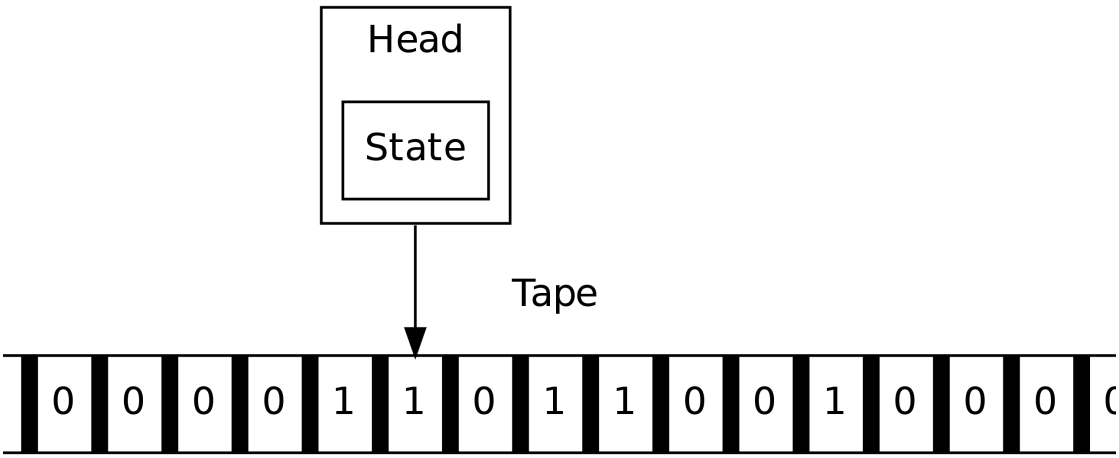

Figure 17.6: schematic representation of a Turing machine.

To access the tape, which is usually considered devoid of symbols at the onset, the machine is supplied with a read/write scanner/printer (the "head") which can move along the tape one step at a time, in either directions. The machine can assume a finite number of internal states, according to a transition table, which determines, at each computational step, based on the current state and current symbol on the tape under the head's position, which action must the machine undertake: that is, which symbol it is to write at the same tape position, and which direction to move the head one step afterwards.

The transition table is formally constituted by a finite list of *quintuples*, each one so composed:

$$< Sta, Sym, Nsta, Nsym, Stepdir >$$

where *Sta* stands for the current state of the machine, *Sym* for the symbol on the tape cell under the current head position, *Nsta* stands for the state which the machine is bound to assume next, *Nsym* for the symbol it is bound to write on the tape at the current head position (overwriting the former symbol), and where *Stepdir*, which can assume only two values, stands for which direction the machine has to move one step along the tape after having accoplished the state change and the writing of the new symbol.

The ordered couple $<Sta, Sym>$ is called the *configuration of the machine*. The configuration unequivocally determines the behavior of the machine at the current step of the computation.

formalize the concept of a calculating machine. The other classes of machines have been studied later by other authors, in the endeavor of ascertaining the limits of different classes of possible machines.

[12] for reasons of space and pertinence, I am giving here only a shallow exposition of the concept of Turing machine and of the Church-Turing thesis. It is up to the reader, with the help of a myriad of sources, to proceed in case to a deeper analysis of these questions, which, as a matter of common knowledge, are among the deepest of modern philosophical, mathematical and scientific thought.

The machine halts, as a matter of convention, when it reaches a particular state designated as the *end state*, or alternatively when there is no entry in the transition table corresponding to the actual configuration.

A specific transition table characterizes a specific TM: there are as many TMs as possible transition tables, and in fact the two terms are usually taken to be interchangeable. This means that there are countably infinite TMs.

As it happens, in his fundamental paper Turing devised a method to encode the transition table of a TM into a string of symbols on the tape. This encoding constitutes the *description* of a Turing machine. It usually consists in a (possibly very large) integer number. He was also able to present the transition table of a specific machine, which is called the *Universal Turing Machine* (*UTM*), which can *simulate* any other machine whose description it gets as input on the tape, followed by the actual data on which the simulated machine is supposed to act on. To be able to simulate a machine means to execute exactly the same sequence of operations on the tape that that machine is supposed to execute. Thus, the UTM ends up being the only TM really needed for, in that it can simulate any other machine, that is it can implement any algorithm.

It's is evident that a major innovation in TMs is the presence of a bidirectional read-write memory. This allows for some computation which is outside the reach of more constrained machines, such as the NFAs. Employed as language recognizers, TMs can recognize a proper superset of context-free languages: the class of *Turing-recognizable* languages. This is the most comprehensive class of language that can be recognized by a machine[13]. Nevertheless, there are limits to the computational capabilities of Turing machines, as we will soon see.

17.2.6 The halting problem and the Entscheidungsproblem

It turns out, and the proof is in the aforementioned paper by Turing, that the Entscheidungsproblem has an answer, and this answer is in the negative. To come to this conclusion, Turing had to set the stage by demonstrating three auxiliary theorems. The first one refers to a special property of Turing machines: it can happen that a machine, during its computations, never reaches an end state (or, equivalently, a configuration not specified in its transition table) so it never comes to a halt. The problem of determining if this is bound to happen, is known as the *halting problem*.

One of the theorems in Turing's paper is an indirect[14] proof that the answer to this question is negative: it is not possible to determine for all cases if a Turing machine is going to ever come to a stop: it can in general not be determined if this is bound to happen or not. Thus, the Turing machine shows the highest computational ability, but it is affected by a fundamental limitation itself.

More formally, the problem can be so stated: it is the request to find a *general* computable procedure to determine if any arbitrarily selected TM comes to a halt, given: a) the specification (the transition table) of that TM, and b) an arbitrarily selected input for that TM. I stressed the word "general", to highlight the fact that the theorem does not state that it is impossible

[13] This is true only if the Church-Turing thesis holds. For a discussion of the thesis, see section 17.3.

[14] Actually, the theorem proved by Turing (in section 8 of the paper) is about the impossibility of finding a mechanical procedure to discriminate, using their description, between TMs that write a finite number of certain symbols on the tape (called by Turing, with a somewhat curios terminology, "circular machines", and TMs that go on forever printings symbols of a certain type on the tape (called "circle-free machines"). He also proves a second theorem which states that there's no algorithm to decide, given a a TM's 'description, if the machine, will ever print a "0" on the tape. Neither of these arguments proofs the halting problem directly (which, by the way, was named "halting problem" later, by Davis. See Copeland 2004, p. 40, n. 61), but they entail it.

to contrive an *ad hoc* method able to predict, based on its transition table, if a *specific, given* machine halts on a *specific, given* input: the theorem states that it does not exist a general, systematic, *fixed* computational method which is always applicable without ad hoc modifications, which is able, based on examination of the transition table of a given machine however chosen and the combined examination of a given input however chosen, to ascertain if that machine will ever come to a halt when provided with that input. The word "computable" means executable by a Turing machine.

In synthesis, the theorem states that there isn't any *single*, *fixed* Turing machine capable of predicting if any however chosen Turing machine will come to an halt on any however chosen input.

Towards the end of the paper, Turing also proved that a logical consequence of the former theorem is that the Entscheidungsproblem gets a negative response as well: there is not a general effective mechanical method to know in advance, without having to *produce* a proof, if a certain mathematical statement is provable or not. This result put an end to a long quest in the foundations of mathematics, but it opened at the same time a new field of logico-mathematical research. Even though Turing has arguably not been the first to give an answer to the Entscheidungsproblem[15], his work is the one which opened two whole new paradigms: the theory of computation and its application, information technology[16].

17.2.7 Nondeterministic Turing machines

As in the nondeterministic finite automata (see section), in a nondeterministic Turing machine (*NTM*), the table of transitions can provide more than one next state for each couple. Accordingly, the machine splits in more than one copy to proceed with a parallel computation. The machine halts if every branch of the spawned tree has halted, or as soon as an accept state is reached.

NTMs don't possess higher computational capability than deterministic TMs: for every NTS there is an equivalent TM. There is nevertheless a difference between a NTM and the corresponding TM: the first can execute certain types of computations more quickly than the second. This intorduces the subject of *computational complexity*, which is to be treated later in this chapter.

17.2.8 Linear bounded automata

We have seen that a Turing machine has a potentially infinite memory (the tape) available. But, of course, actual physical machines can make use not of an infinite, but only of a large, possibly very large amount of memory. A nondeterminstic TM with a *finite* amount of memory is a *Linear Bounded Automaton* (*LBA*). It can recognize a proper subset of the Turing-recognizable languages, the so-called *context-sensitive* languages. This class is a proper superset of context-free languages.

[15] A couple of papers by the American logician Alonzo Church, which a few months later was to be Turing's PhD tutor at Princeton, actually predate Turing's work on the unsolvability of the Entscheidungsproblem: Church (1936a) and Church (1936b). In the section about the "Church-Turing" thesis I will give some hints about the relationship between Church's and Turing's work. See also Copeland (2004).

[16] The influence of Turing's seminal work on the birth of information technology is disputed. According to some, his influence begins only in the '50s. 'See for example Daylight (2012) and Daylight (2013) and Haigh (2013).

17.2.9 The Chomsky hierarchy

We have seen that ability to recognize larger and larger sets of languages means a progressively higher computational capacity of the recognizing machine. This hints at a hierarchy of expressive power of languages and correspondent computational capability of machines. Such a hierarchy has been brought forth by Noam Chomsky (1956) in a seminal paper which has founded a branch of theoretical linguistics, in which he proposes the formalization of the notion of grammar of a language. In this context, the term *grammar* refers, of course, to a formal concept. I will delve a little into the subject in section 17.2.10. Suffice to say here that a grammar is an abstract device that generates *languages*. So, for each type of grammar, a different class of languages is generated. And, as we now, every class of languages has some abstract machine that can recognize it. The Chomsky hierarchy is depicted in fig. 17.7 in the form of a Venn diagram.

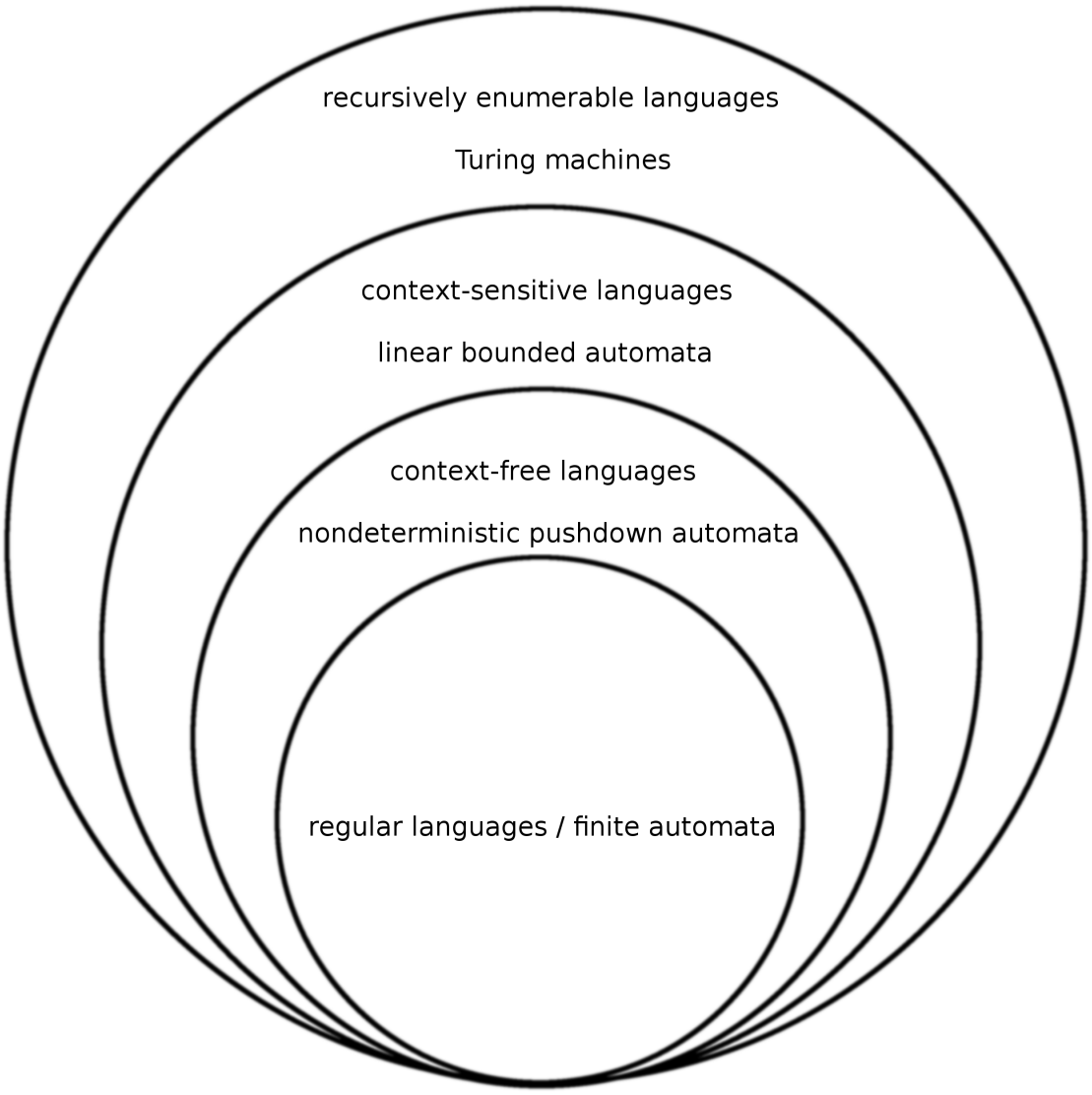

Figure 17.7: the Chomsky hierarchy.

The Chomsky hierarchy has been integrated by the successive discovery of some more machine types with intermediate computational capabilities with respect to the classes appearing in the original hierarchy. A more complete hierarchy is shown in fig. 17.8[17].

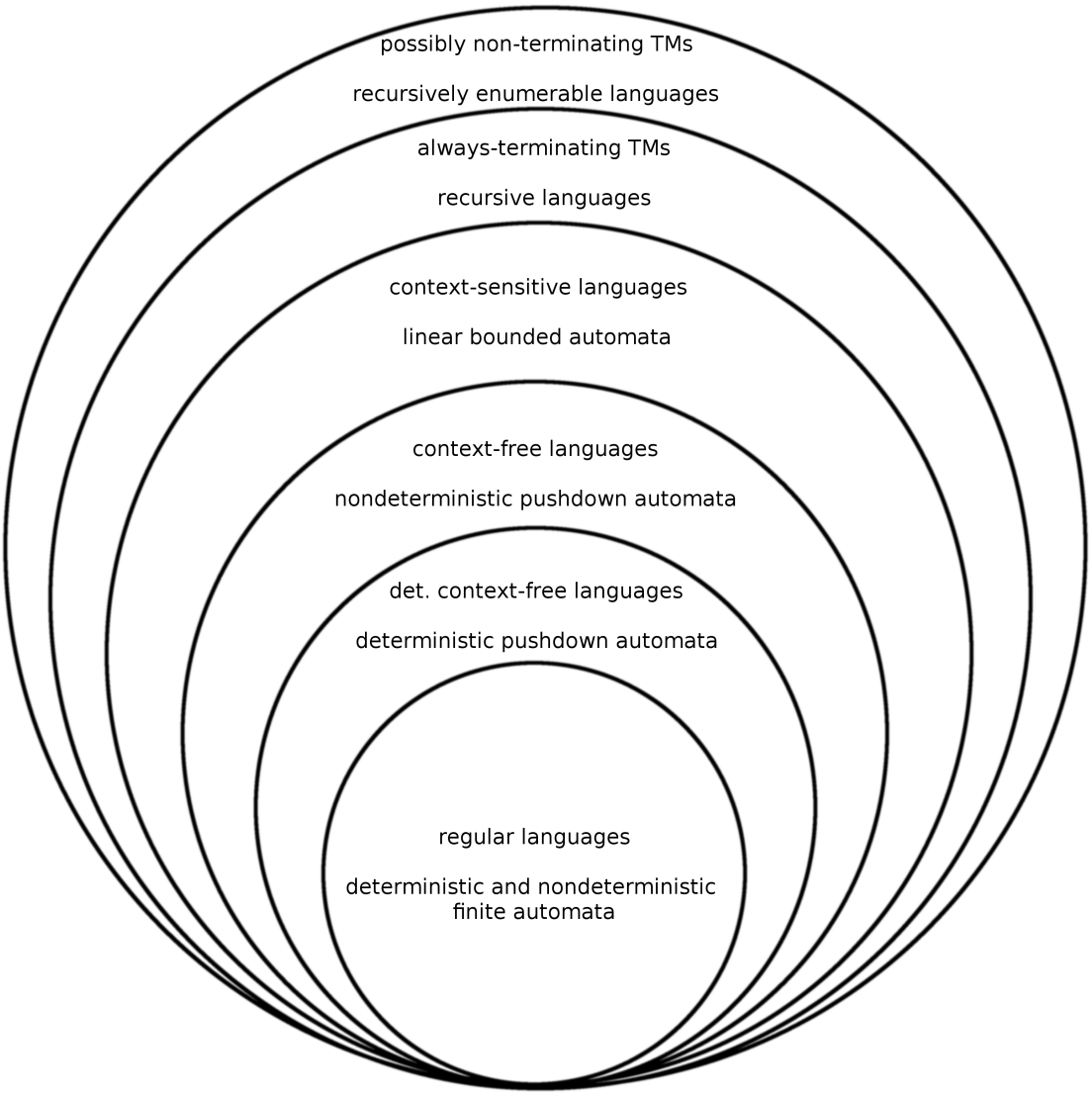

Figure 17.8: an extended Chomsky hierarchy.

The notable thing in this hierarchy is the presence of two top tiers: *recursively enumerable* languages and *recursive* languages. Recursive languages are those that are decidable by a TM that always halts, a machine also called a *decider*. The name derives from the fact that this machine can *decide* if a given string belongs to a recursive language or not, that is, it is going to halt and give an answer whether the string belongs to the language, or not. There is, moreover, a

[17] There are more possible hierarchies: a number of machines and languages with intermediate computational ability, with which we are not concerned here, have been demonstrated in the literature.

superclass of the recursive languages, which is the class of those languages for strings of which a TM can only guarantee to halt and give a (positive) answer only in case the string *does* belong to that language, otherwise the machine never halts. Since it never halts, but we cannot determine, as an effect of the halting problem if it will actually ever halt, this lack of a response in the case of a non-halting machine does not constitute a definite negative answer, so we cannot say that the machine *decides*, but only that it *recognizes* a language. Due to the same halting problem consequences, there is no general effective method to distinguish the class of always terminating machines from the class of all TMs, so the distinction drawn in this hierarchy is not easily applicable.

17.2.10 Grammars

We have seen the automata described above[18] as language *recognizers*: given a string in input, the automaton accepts or refuses such string. Strings which are elements of the language recognized by the automaton are accepted, the other refused[19]. But, given a specific language, that is, a set of strings which is *recognized* by an automaton, how could the strings of this language be *produced*?

Languages are produced by formal structure called *grammars*.

Informally, grammars are devices for producing, given a "start" signal, and an internal starting string, an output string through progressive string construction steps which are not determined, but nevertheless are constrained by certain rules. These rules, called *production rules*, can be seen as rules of *string rewriting*, where a string can be obtained by starting from another string and substituting parts of it with other strings. An example rule:

$$V \to exV$$

In the above expression, V is a placeholder (a *variable*) for any string. Thus, the rule above is a rule for substituting the string corresponding to the variable V with the string exV. In the case V stood for "ample", the resulting string would be "example".

When the process (the *derivation*) comes to a halt[20], the constructed string is supplied as output. Such a string belongs to the language that is characterized by that certain grammar. By repeated application of this process, a grammar can generate all the strings belonging to a certain language, and only them.

17.2.10.1 Context-free grammars

Context-free languages, which are recognized by PAs[21] are generated by *context-free grammars*. Here is an example of a context-free grammar (let's call this grammar G):

$$V = \{S, a, b\}$$

[18] Section 17.2.

[19] Of course, in the case of TMs testing if a string belongs to a recursively enumerable language, the machine can in some cases simply not halt, without explicitly refusing the string.

[20] The process halts, because a grammar always defines a set of *terminal symbols*, that is, symbols which do not appear on the left side of any production rule, but appear on the right side of some rule: when the string under processing ends up being composed by terminal symbols only, the process cannot apply any rule, and halts.

[21] See section 17.2.4.

$$\Sigma = \{a, b\}$$

$$R = \{S \to aSb, S \to ''\}$$

where V is the alphabet, that is, the set of possible symbols, Σ is the set of *terminal symbols*, and R is the set of production rules. The symbol '' on the right side of the second production rule stands for the empty string.

A possible derivation in G is:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$$

A *context-free* language is so called, because the production rules of its grammar allow for substituting with other strings certain substrings embedded inside a string, without taking into account the remaining content of the main string that surrounds them. For example, a rule like:

$$A \to aA$$

can be applied indifferently to the string $abcAAdef$ or to $ghiAAjkl$, resulting in

$$abcaAaAdef$$

and

$$ghiaAAjkl$$

respectively: the two different *contexts* $abcA[...]def$ and $ghi[...]Ajkl$ are not taken into account when evaluating the applicability of the rule, and make no difference in the results of its application.

17.2.10.2 Relationships between the expressive power of grammars

Besides regular and context-free languages, there are other classes of languages which are more difficult to recognize. Correspondingly, they require other types of grammars. The following equivalences between the generative power of grammars and the recognition power of automata hold[22]:

- Regular languages, which are recognized by FAs, can be seen as generated by the so-called *regular grammars*.
- Context-free languages, which can be recognized by PAs, are generated by *context-free grammars*.
- Context-sensitive languages, which can be recognized by LBAs, are generated by *context-sensitive grammars*.

[22] See also fig. 17.7.

- Recursively enumerable languages, which can be recognized by TMs, are generated by *unrestricted grammars*.

The scale of grammatical power outlined above is a hierarchy: that is, every grammar type is properly included in the more powerful one. So, regular grammars are also context-free grammars, which in turn are also context-sensitive ones, which in turn are also unrestricted grammars.

For example, the context-free grammar

$V = \{S, M, A, B\}$, $\Sigma = \{a, b\}$ and $R = \{S \to aMb, M \to A, M \to B, A \to aA, B \to bB, A \to$ $", A \to "\}$

generates the language constituted by the strings of the form aMb, where M is a string of a's or a string of b's. This is a *regular* language, and as such can be recognized by a FA. But, it is easy to see that the grammar G in the example above generates the language composed of all possible strings consisting in a sequence of n a's, followed by a sequence of n b's. As it happens, this language is a *proper* context-free language: it is not regular, for it cannot be recognized by a FA, but requires a PA as a recognizer.

17.3 The Church-Turing thesis

As hinted above, not only Turing, but other mathematicians had undertaken the search for a solution do the Entscheidungsproblem, and more than one came to the conclusion of its unsolvability. In doing so, more than a formal model of computation has been conceived. The first one to come up with such a model was Alonzo Church in 1936, a few months before Turing's solution. Other solutions to the halting problem came later. A remarkable fact, though, is that all such solutions have been proved to be mathematically equivalent: Turing himself proved, in the appendix of his 1936 paper, that his definition of computable functions is mathematically equivalent to Church's one, the so-called *lambda calculus*. So, it appears that all attempts undertaken till now to formally characterize the intuitive notion of mechanical or effective procedures have ended up identifying the same set of functions, the *computable* ones (called also *recursive* functions). This gives support to the idea that the informal notion of mechanical method *has indeed been captured* by the known formal models of computation, for otherwise it would be expected that one of its formal characterization could result non-equivalent to the others, being more comprehensive.

The proved equivalence between all the formal models of computation means also that all of them have the same basic limitations: they cannot solve problems that the Turing machine cannot solve (such as the halting problem). Any system with the same capabilities of the Turing machine is said to be *Turing-complete*, and Turing completeness characterizes the notion of Turing-computability. So, all formal models of computation have been shown to be equivalent, and characterize the same concept of computability: no model of computation which models the notion of mechanical procedure is able to solve the halting problem and other uncomputable problems.

This equivalence is the basic idea behind the so-called *Church-Turing thesis* (*CTT* henceforth). Simply stated, the Church-Turing thesis is the statement that the set of all possible mechanical procedures coincides with the set of possible *computations* effected by a Turing machine.

As appears from what we've seen before, this thesis has not been proven in a formal sense, nor it is *provable*, for it amounts to a statement which equates an informal notion with a formal one[23]. Nevertheless, the thesis is almost universally accepted, and in the rest of this dissertation I will implicitly assume that it holds. I won't go here into a detailed analysis of the thesis, around which a heated debate has coalesced and is still alive[24]. Suffice to say that the problematic aspect of the CTT resides in its equating a notion, that of effective procedure, which can be seen as a basic capability of human cognition, to the capabilities of a *machine*. This opened up the possibility of *Artificial Intelligence* (*AI*), a field which has started to flourish precisely in the post-WWII years after Turing's formulation of computation had settled and had been generally accepted,[25] and which has not ceased to be the subject, along with its cognate psychological hypothesis, the *Computational Theory of Mind* (*CTM*), of a harsh debate.

Another common conception (or, possibily, misconception) about the CTT is that it states a physical upper limit to the computational capabilities of any system, namely, that no physical system can be expected to effect computations which cannot be effected by a Turing machine: that is, it can't physically[26] exist a possible machine (or, in general, a physical system mechanistically explainable) which can solve Turing-unsolvable problems, such as the halting problem. This controversial variant of the thesis is usually defined the *Physiscal Church-Turing Thesis* (*PCCT*), and it is itself object of a current debate.[27]

17.4 Computational complexity[28]

The uncomputability of the halting problem reveals an *in principle* limitation of computation: it shows that some hypothetical procedure, of which a sensible formal definition can indeed be given, cannot actually be executed by the Turing machine. This is an *absolute* limitation, which shows the boundaries of what is computable (assuming of course that CTT holds). Among the procedures which *are* computable, there are nevertheless some that undergo a more practical limitation: they cannot be carried out by a *physical* machine in a plausible time or with the

[23] The thesis is seen by many as a definition, and probably it had been so intended by some of its early proposers. For example, Church (1936a) described his intents this way: "The purpose of the present paper is to propose a definition of effective calculability, which is thought to correspond satisfactorily to the somewhat vague intuitive notion in terms of which problems of this class are often stated, and to show, by means of an example, that not every problem of this class is solvable.". A more recent paper which tries to transform the doubts about the truth of the CTT into doubts about the adequacy of a definition is Soare (1996).

[24] See for example Copeland (2004) and Copeland (2008).

[25] Turing himself is not innocent in this regard: he accepted the idea that the mind can be simulated by a machine, and went so far as proposing, in Turing (1950) a standard method (the so-called *Turing test*) to check if artificial intelligence has been achieved.

[26] This limitation does not apply to abstract models of machines, for a few purely theoretical machines have been devised, that can hypothetically execute tasks which are not performable by conventional Turing machines. These are so-called models of *hypercomputation*. However, according to many, it's impossible to build such machines in the physical world. Here too, an ardent debate is underway. See, for opposite views on the subject, Copeland (2002) and Davis (2004). It can be noted that such machine models do not purport to capture the notion of a finite mechanical procedure, for they presuppose in various ways *actual infinity*, so the CTT is not violated.

[27] See for example Deutsch (1985) and Piccinini (2011).

[28] The subject of computational complexity is by no means analyzable in depth in the bounded space of this chapter. The variety of complexity classes brought to light by research in this field in the last fifty years is overwhelming. There is even a blog, *Complexity Zoo*, attendend by the MIT computer scientist Scott Aaronson, in which it is attempted a systematic record of all the classes and theorems found in the literature to date: https://complexityzoo.uwaterloo.ca/index.php?title=Complexity_Zoo&oldid=6292. For these reasons, I will only highlight here some of the major complexity classes and the (often problematic) relationship between them. For a deeper explanation of the subject, see Sipser (2012), Garey & Johnson (1979), Bovet et al. (1994), Fortnow & Homer (2003) and Arora & Barak (2009).

expenditure of a plausible amount of resources. In the mid-60s, after the field of information technology had already started to mature, many computational machines had been realized, and computer programming had already been collecting a series of standard algorithms and relative problems, a seminal paper by Hartmanis & Stearns (1965) started[29] the theoretical study of these practical limitations, which had not been taken into account in defining the absolute possibilities of computation: there are computational procedures, many of which would be of practical avail, that could *in principle* be carried out, but for which the amount of time or memory required to bring them to completion would end up being unbearably large: for example, for a procedure which takes something like 2^n steps to finish, with n the size of the input string, the actual execution of such computation on an input of size 93 bits would take about 30 billions years, assuming a plausible computer hardware (for today's standards) which can perform 10 billions of operations for second. Consider that the universe, according to current cosmological theories, has existed for about 13.8 billion years. Moreover, a tiny increase in the input size, from 92 to 93 bits, would mean an increase in computational time of another 30 billion years, for the computational time doubles for every bit: this computation would go well past the end of the predicted remaining life of the universe.[30] It is clear that even technological breakthroughs able to accelerate the computing hardware of some orders of magnitude could not resolve the feasibility of computations of this kind, the execution time of some of which can grow much faster than an exponential function. The study of this kind of problems is the study of *computational complexity*.

This limitation to the feasibility of some computations, which is not an absolute one, seems to be still pretty much an insuperable one. The only possible way of overcoming it would be to find another procedure, which bears the same results of the original one, but that can be computed in a time which varies according to a much less steep function of the input size: for example, a quadratic function, or, even better, a linear one. The answer to this quest is in certain cases negative: there seem to be computable procedures that are not transformable into slower-growing ones. That is, as we will see, there seems to be a class of problems whose computational solution is inherently *intractable*.

17.4.1 Time complexity

To measure the *time complexity* of a halting computation, the number of steps a deterministic Turing machine[31] goes on before halting is taken into consideration. The actual number of steps is a function $f(n)$ of the size n of the input.[32] We are interested in the behavior of this function relative to a large input size, in order to find a method for classifying algorithms with similar complexity into *complexity classes*. To reach this goal, we consider an *approximate* estimate of the number of required steps: this way, algorithms with similar execution time will be classified in the same class. A method of approximation adequate to this end, is that of *asymptotic analysis*, which takes into account the asymptotic limit of the function as the input size approaches very high values.

For example, let's suppose that an algorithm's execution time is a polynomial function of n, the input size:

[29] Some early work had preceded this paper. Notably, Yamada (1962). See also Fortnow & Homer (2003).
[30] At least according to some current cosmological model.
[31] Except where specified otherwise, deterministic TMs will be taken as the reference computational model in this section dedicated to computational complexity.
[32] It can be any function, including a costant one.

$$f_1(n) = 6n^3 + 2n^2 + 3n + 20$$

For large enough values of n, say, for example, $n = 500$, the value of this function will be:

$$f_1(500) = 6 \cdot 500^3 + 2 \cdot 500^2 + 3 \cdot 500 + 20 = 750,501,520$$

The term $6n^3$ alone amounts to 750,000,000, so it is evident that, for sufficiently large values of n, the other terms of lesser degree end up having a negligible influence on the total value of the function. It could then be reasonable to approximate the function by reducing it to an expression comprising only the highest-degree term of f_1:

$$f_1(n) = 6n^3$$

With an analogous argument, it can be shown that for sufficiently large values of n, even the contribution of the coefficient becomes irrelevant with respect to the value of the cubed variable. So, we take into consideration only the highest degree term, with coefficient one:

$$f(n) = n^3$$

Similar polynomial functions which actually grow at a different rate, such as, for example,

$$f_2(n) = 2n^3 + n^2 + 4n$$

when undergoing the same kind of simplification would end up being approximated by the same n^3 function.

Other functions would not. For example,

$$f_3(n) = 3^n + 8n^4 + 12n$$

would be approximated by:

$$f(n) = 3^n$$

It seems clear that f_1 and f_2 belong to the same time complexity class, while f_3 does not: it will grow faster than any function belonging to the class $f(n) = n^3$.

The notation for time complexity is the so-called *big-O notation*[33]. For example, for an algorithm whose execution time grows as the function f_1 or f_2, we will write:

$$f(n) = O(n^3)$$

while for an algorithm whose execution time grows as the function f_3 we would write:

[33] The "small o" notation and some subtle technicalities are beyond the scope of this simple introductory section.

$$f(n) = O(3^n)$$

The time complexity expressed in big-O notation represents an *upper bound* to the value of the time required for computation. In other words, it is the "ceiling" function which any function whose approximated form has that big-O form will not surpass for inputs large enough. For example, stating that a certain algorithm has $O(n^6)$ time complexity, means that its execution time will not grow faster than n^6 for inputs large enough.

Bounding functions of the form of the form $O(2^n)$, or $O(2^{n^k})$[34] are *exponential bounds*, while those of the form $O(n^c)$ are called *polynomial bounds*. There also typical slower-growing bounds, such as $O(nlog(n))$, $O(log(n))$ and $O(n)$. The latter is called *linear time*.

17.4.1.1 The *TIME* complexity classes

The *Time complexity class* $TIME(f(n))$, is defined as the class of all languages decidable by a computation whose total run-time is bounded by a given function $f(n)$. For example, language deciders bounded by f_1 and f_2 above are in $TIME(n^3)$, while those bounded by f_3 are in $TIME(3^n)$.

17.4.1.2 The *EXPTIME* complexity class

The class $EXPTIME$ is the set union of all the classes $TIME(2^{n^k})$. That is, it contains problems solvable by programs whose execution time can grow exponentially with input size.

It has been proved[35] that $P \neq EXPTIME$. This means that not all problems have solutions bounded by a polynomial time growth, and so that there are problems which, despite being solvable, are *inherently* hard to solve, for they require too much time to finish[36].

17.4.1.3 *P*, *NP* and complexity classes

It has been noted that a classification of time complexity coarser than *TIME* is usually sufficient to distinguish between problems whose solution can be computed in a feasible time, and intractable ones.

The classes taken into consideration by this coarser classification are classes of algorithmically solvable decision problems. Decision problems in general consist in ascertaining if a given object belongs or not to a given set[37].

Since a decision problem, if it can be solved[38] is solved by an algorithm, a class of solvable problems identifies the class of the algorithms able to solve them. A class of problems also identifies a class of algorithms which can be used to *verify* the correctness of a purported already-found solution to the problem.

---

[34] For k>0.
[35] See Sipser (2012), p. 343.
[36] See later section on intractability.
[37] Typically, as we have seen, the ascertaining if a given string belongs or not to a given language.
[38] Unlike, for example, the halting problem.

17.4.1.3.1 The class *P* The class P ("P" as "Polynomial") is the class of all problems whose algorithmic solutions terminate in a time bounded by a polynomial function: for example, f_1 and f_2 above belong to this class. In general, any problem whose solution's time complexity is of the form $O(n^k)$, is in the class P. Evidently, for plausible values of k, programs belonging to this class can be computed quite easily. According to the so-called *Cobham's thesis*, only programs which are in P are tractable, that is, can terminate in a useful time. This is only a thesis, and it is not accepted by everyone: the reason is that there are programs in P that are of sufficient complexity (for example $O(n^{100})$) to be considered intractable. The class P is a class of problem solvers, that is programs that have the task to *find* the solution to a problem.

17.4.1.3.2 The class *NP* and the *P = NP* problem A question which is different to the search for a solution is the *verification* of the fact that an already found solution is indeed a solution.

The class *NP* ("NP" as "Nondeterministic Polynomial") is the class of all solutions which can be algorithmically verified in a polynomial-bounded time by a *nondeterministic* Turing machine.

Note that all problems in P are also in *NP*: if a solution can be found in polynomial time, the verification of that solution is the polynomial algorithm used to find it. This means that $P \subseteq NP$. But it is a well-known open problem if $P \subset NP$ or not. It is universally assumed that the inclusion *is* proper, that is, that there are problems in NP which are *not* in P. I have implicitly assumed this hypothesis throughout this text. The problems which are not in P are problems whose possible algorithms to find a solution grow faster than polynomial time, to wit, algorithms of at least exponential time complexity.

It must be noted that, even if a problem were not in P, that does not necessarily mean that a more than polynomial time is *always* needed to identify a solution to the problem: for many algorithms, this usually happens only for a subset of the inputs: other cases can often be solved with a polynomial time computation.

17.4.1.3.3 *NP-completeness* It is a longstanding an fundamental puzzle of computer science whether it is the case that $P = NP$ or not. As yet, there has not been any proof or disproof of the $P = NP$ hypothesis.

There are problems in *NP*, the so-called *NP-complete* problems, such that, if it were found that they can be solved by an algorithm running in polynomial time, this finding would entail that for all the other problems in *NP* for which a polynomial time algorithm is not yet known, one could be found too. And conversely, if one of the members of NP is shown to be intractable, with no possible solution to it which runs inside polynomial time, then the *NP*-complete problem is unsatisfiable too.

The reason for this is that any known problem in *NP* is *reducible* to one of the *NP*-complete problems[39,40].

Problems for which the best algorithms known today still run in more than polynomial time, had been known at least since the last 60's. But only in the 70's their study became intensive, and led to the demonstration, achieved independently in North America by Stephen Cook (1971) and

[39] The reduction must be effected in polynomial time. *Reducible*, applied to a decision problem, means that the algorithm to solve it can be substituted by the algorithm to solve *another* decision problem. Of course, the aim is to find an *easier* (in computing resources) problem to which to reduce a given harder one.

[40] See also, later in this section, the explanation of *NP*-hardness.

in the Soviet Union by Leonid Levin (1973), of NP-complete problems (the so-called *Cook-Levin theorem*).

From the fact that it is not known at all if $P = NP$ or not, it is inferrable that *NP*-complete problems are among the ones for which an equivalent algorithm in P has not yet been found, and seems hard to find, so, they probably lie outside of P. In other words, they are among the "hardest" problems in *NP*.

A typical *NP-complete* problem is the so-called SAT, that is the problem of finding a variable assignment which satisfies a given boolean formula[41]. To find the assignement of variables which would render a given boolean expression true, one must, in the worst cases, produce *every* combination of the variables' values, and the number of these combinations if 2^n, with n the number of variables involved. Thus, it is evident that such a computational task has an exponential bound, and so it is computationally hard.

17.4.1.3.4 *NP-hardness* The property for an algorithm that any problem in *NP* is reducible to it, is called *NP-hardness*. The *NP*-complete problems are those *NP*-hard problems which are inside *NP*. There are other problems which are *NP*-hard but lie *outside NP*.

17.4.2 Space complexity

Analogously to time complexity, the space complexity of a halting Turing machine computation is the maximum number of cells on the tape the machine scans before halting, which is a function $f(n)$ of the input size n. We use here too a big-O notation to indicate the bounding function of the computation space, writing:

$$f(n) = O(n^3)$$

for an algorithm which scans a number of cells on the tape approximately bounded by a function which grows as the third power of the input size.

Space complexity is in a way independent[42] from time complexity, because space can be reused, unlike time (a cell on the tape can be read/written more than one time, without space complexity growing as a consequence). For example, the SAT problem mentioned above, has O(n) space complexity, that is, it can be solved in linear space, while, being *NP*-complete, the problem cannot probably be solved neither in linear time nor in polynomial time.

17.4.2.1 The *SPACE* complexity classes

The *Space complexity class* $SPACE(f(n))$ is defined as the class of all languages decidable by a computation of a TM whose total memory scanned is bounded by a given function $f(n)$. For example, language deciders bounded in space by $f(n) = n^5$ are in $SPACE(n^5)$, while those bounded by $f(n) = 2^n$ are in $SPACE(2^n)$.

17.4.2.2 The class *PSPACE*

The class *PSPACE* is the class of all programs whose algorithmic solutions use up to a polynomial bounded amount of cells in the tape of a deterministic Turing machine.

[41] See Sipser (2012), p.272.
[42] Not completely independent: see the later section on the relationships between complexity classes.

There is also a class $NPSPACE$, but it has been proved (Savitch's theorem[43]) that $PSPACE = NPSPACE$.

17.4.2.3 The *EXPSPACE* complexity class

The class $EXPSPACE$ is the set union of all the classes $SPACE(2^{n^k})$. That is, it contains problems solvable by programs whose memory consumption can grow exponentially with input size.

It has been proved[44] that $PSPACE \neq EXPSPACE$. This means that not all problems have solutions bounded by a polynomial space growth, and so that there are problems which, despite being solvable, are *inherently* hard to solve, for they use up too much memory[45].

17.4.2.4 *PSPACE-completeness*

A language is *PSPACE-complete* if it is in $PSPACE$ and every other problem in PSPACE is reducible to it in polynomial time. As for *NP-completeness*, the $PSPACE$-complete problems are the most diffucult in $PSPACE$ to solve.

17.4.3 Relationships between space and time complexity classes and open problems

Having catalogued several complexity classes, we could ask if they are related in some way. It turns out that they are, but the exact relation is an open problem.

It has been proved[46] that:

1. $P \subseteq PSPACE$
2. $NP \subseteq PSPACE$
3. $PSPACE \subseteq EXPTIME$

Result 1, intuitively, stems from the obvious fact that, moving one step at a time, a TM cannot use up more space than that allowed by the number of steps along the tape it has effected till then.

Result 2 comes from the theorem that $NP \subseteq NPSPACE$. But, for Savitch's theorem, $PSPACE = NPSPACE$. So, $NP \subseteq PSPACE$.

To explain informally result 3, a little more complex argument is needed. Imagine that a TM which is in $PSPACE$,[47] observed after having halted, ends up having visited m cells of space. The total configuration of a TM is determined by the position of the head on the tape, the current state of the machine, and the whole sequence of non-blank symbols written on the tape. A halting TM cannot *repeat* any of its configurations during its run, otherwise it would not halt: it would loop forever. The total number c of possible non-repeating configurations is a function of the maximum number m of non-blank cells the machine will have used up by the time it halts.

[43] See Sipser (2012), p. 306.
[44] See Sipser (2012), p. 340.
[45] See later section on intractability.
[46] For the formal proofs, see Sipser (2012), p. 308.
[47] That means that its used space is a function in $O(n^k)$, with $k \geq 0$.

The number a of possible different configurations the machine can *actually* have run through during its computation is then at most c, and c is a function of m, a function which belongs to $O(2^{O(m)})$, that is to $O(2^{O(n^k)})$, given that we have supposed the machine is in *PSPACE*[48]. So, the time (the k steps) the machine has been running before halting is at most equal to this number, a, of possible assumed configurations, because each computational step necessarily changes the machine total configuration, so it "needs" a non-visited configuration. This means that the maximum time the machine can have run before halting is bounded by a $O(2^{O(n^k)})$ function. In other words, that the time complexity of the machine is *EXPTIME*. So, given that the machine was supposed to be in PSPACE, we have the theorem 3 above. In other words, if we let a program expand its used memory even with a linear growth rate, we cannot expect it to necessarily complete its task in less than an exponential time. This is quite obvious: there are programs, like simple solvers for SAT[49], which make an exahaustive search on the space of possible boolean variable assignements, where the memory required grows linearly with the number n of boolean variables considered, while the number of possible combinations of their values grows as 2^n. Result 3 can be viewed as meaning that there are problems in *PSPACE* that are also in *EXPTIME*, and we can suspect that among them are the *PSPACE*-complete, for reasons analogous to those mentioned in the preceding section about *NP*-completeness.

To sum up the relations between the classes we have taken into consideration so far:

$$P \subseteq NP \subseteq PSPACE \subseteq EXPTIME$$

The problematic status of the statement above must be highlighted: it is not known if any of the inclusions in the statement above is a *proper* inclusion. But, we know[50] that $P \neq EXPTIME$. So, at least one of the inclusions is strict, but nobody till now have been able to prove which. There is, however, widespread consensus in the computer science community, that *all* the inclusions are proper.

17.4.3.1 Existence of intractability

When I talked about *NP-complete* and *PSPACE*-complete problems, I mentioned that such problems, for what we know, are probably inherently computationally hard, even if a proof of this fact has not been supplied yet. But, there is always the possibility that such problems end up being actually in P, given that we don't know the actual status of the inclusion relations between the various complexity classes. As we have seen, there are, though, two known *strict* inclusions:

4. $P \subset EXPTIME$
5. $PSPACE \subset EXPSPACE$

[48] For a machine with 2 symbols in the alphabet, the number of possible configurations on a tape with m written symbols is 2^m. The number of possible total configurations of the machine is then this number multiplied for the number of possible states of the machine and for the number of different positions of the head on the tape. If the machine has visited m cells by the end of its computation it cannot have positioned its head in more than m positions. So, the total number of configurations that the machine can have been in, is in $O(2^m)$. But m is a function in $O(n^k)$, so the total number of configurations the machine could have assumed before halting is $O(2^{O(n^k)})$.

[49] See above, in the section on *NP*-completeness.

[50] see section above on *EXPTIME*.

Theorem 4 means that there are decision procedures for some problems which take strictly more than polynomial time, and theorem 5 that there are decision procedures for some problems which take strictly more than polynomial space.
These are indeed problems which are *intractable*, in the sense that the algorithms that solve them run necessarily in $O(2^{n^k})$ time or space. This means that not only faster and more frugal algorithms for those problems have not been found yet, but that they *can not* be found.
These procedures are computable only in principle, but not in practice, for we would run out of space or time well before being able to see them produce a response. This is a practical limitation, but which itself holds in principle, so it is an absolute practical limitation.

Problems which are complete for their classes are instead only *suspected* of intractability. Nevertheless, given that there's no hint that anybody is about to prove that $P = NP$, we can consider such kind of problems intractable as well. Hundreds of well known computational problems in various fields have been proved to be *NP*-complete, or *PSPACE*-complete, so they are to be considered hard.

It is to be remembered though, that the assignment of an algorithm or problem to an intractable class, often means only that intractability manifests only in the worst cases, depending on the inputs.

Chapter 18

Modularité, Antimodularité, Explication: une visite d'introduction .

Ce travail concerne principalement la notion de *modularité hiérarchique* dans les systèmes complexes, sa détection algorithmique et sa utilisation pour expliquer la structure et le comportement dynamique de ces systèmes au moyen de modèles modulaires hiérarchiques. Plus précisément, je mets en èvidence la portée pragmatique de la modularité hiérarchique sur la possibilité de l'*explication scientifique* des systèmes complexes, c'est-à-dire, systèmes qui, selon une description de base choisie par l'observateur, peuvent être décrits comme composés de parties élémentaires discrètes interdépendantes. Je souligne que la modularité hiérarchique doit être considéré une notion *relative*, dépendante du choix, de la part de l'observateur, d'une particulière, fondamentale, *description préférée* du système, qui consiste en une représentation du système comme un ensemble de parties atomiques interdépendantes. Dans un tel type de description, la modularité se manifeste essentiellement comme la possibilité de *décomposition* du système en sous-systèmes reconnaissables, suffisamment définis et persistants (les modules) chacun composé de pièces qui sont plus fortement liées les unes à les autres que à pièces appartenants à d'autres modules ou à l'environnement externe. En fait, de ce point de vue, la modularité hiérarchique ne concerne pas le système réel, physique, en soi, mais seulement ses possibles *descriptions* et *descriptions de descriptions*. Ce sont des modèles théoriques d'un système, et je me concentre ici sur les descriptions modulaires *des modèles*, laissant souvent de côté l'épineuse question de la relation entre le modèle et le phénomène modélisé, à savoir, la relation entre le phénomène empirique et sa première description: ces problèmes mériteraient certainement un traitement séparé approfondi qui ne peut être fourni ici, bien que à la fin, ma proposition saura en quelque sorte toucher même ce genre de questions.

Après avoir examiné les propriétés définissantes de la modularité hiérarchique, je me concentre sur les méthodes algorithmiques connu pour sa *détection*, c'est-à-dire, des algorithmes qui, à partir d'un système complexe (sous la forme de sa *description préférée*, à savoir, sa description comme un ensemble de plusieurs parties interdépendantes), tentent de donner une re-description hiérarchique modulaire du système. Une fois détecté, la modularité hiérarchique semble être la caractéristique essentielle de la description d'un système qui permet l'explication fonctionnelle ou mécanique multi-niveau du système. Ce sont des formes importantes d'explication, largement utilisés en science.

Dans cet ordre d'idées, je me concentre ensuite sur la propriété inverse, l'absence de modularité hiérarchique, que j'appelle *antimodularité*, en essayant de tirer les conséquences de sa possible manifestation dans certaines descriptions de systèmes. L'antimodularité est une propriété com-

plexe, découlant d'une série de circonstances possibles, et ses caractéristiques principales sont celle d'être, comme la modularité, dépendente du choix de la part de l'observateur d'une description de base préférée du système, mais aussi, surtout, celle d'être dépendante de certaines *contraintes computationnelles* qui affectent les possibles algorithmes employées pour la détection de la modularité: la plupart de ces algorithmes sont très exigeants en termes de coût computationnel, et il y a même des résultats théoriques sur l'*intraitabilité computationelle* de la recherche d'une description modulaire optimale d'un système. Cette complexité de calcul entrave inévitablement la recherche de modularité dans des systèmes d'intérêt scientifique d'une taille suffisante. Je propose d'appeler l'effet de cette entrave *émergence antimodulaire*, par analogie avec certaines formes connues de l'émergence computationnelle. Je conclus que l'émergence antimodulaire implique (avec quelques réserves) le *weak emergence* de Mark Bedau, qui est une autre forme de l'émergence computationnelle.

Après avoir défini ce nouveau type d'émergence computationnelle, c'est-à-dire l'émergence antimodulaire, qui est due à la complexité computationnelle que les algorithmes pour la détection de la modularité peuvent manifester dans certains cas, je tente de tirer quelques conséquences possibles de l'émergence antimodulaire sur la possibilité d'*expliquer scientifiquement* les systèmes qu'elle affecte.

Je prends en considération trois modèles classiques de l'explication scientifique: déductive-nomologique, mécaniste, et explication computationnelle, plus un modèle nouveau, récemment proposée par Philippe Huneman, l'explication topologique. Je conclus que l'émergence antimodulaire affecte la faisabilité de tous ces types d'explication, quoique de manières différentes.

Tout d'abord, je prétends que l'antimodularité affecte négativement l'explication mécanistique, une forme fondamentale d'explication dans les sciences biologiques. Prenant côte avec Cory Wright et William Bechtel pour une vue *épistémique* de l'explication mécaniste (par opposition à une vue *ontique*), je montre comment l'antimodularité oblige à recourir à un seul niveau d'explication seulement, en négligeant la nécessité, essentielle pour obtenir des explications mécanistes, de l'intégration multi-niveaux. Le fait de limiter l'explication mécanistique au niveau de la description représentant les parties les plus élémentaires du système, entrave certainement la compréhension: pour systèmes suffisamment grands, leur explication mécanistique à ce niveau est trop complexe pour être comprise par les êtres humains. Et, l'intelligibilité est une qualité à rechercher dans l'explication mécaniste, du moins selon William Bechtel et autres auteurs aussi, comme Petri Ylikoski, qui considère la "saillance cognitive" une des caractéristiques importantes de l'explications.

Pour ce qui concerne les explications déductives-nomologiques classiques, je montre que, *l'antimodularité entraînant l'émergence faible dans des systèmes suffisamment complexes*, il est impossible de recourir à l'explication *déductive-nomologique* (*DN* désormais) Hempel-style pour un système antimodulaire, car, si l'on pouvait, cela voudrait dire que le système est prévisible au moyen d'une loi, ce qui est nié par la définition même de l'émergence faible qui, comme on a dit, est impliquée par l'antimodularité. Ainsi, un système antimodulaire est pas prévisible, du moins pas prévisible arbitrairement à l'avance au moyen d'une loi analytique, et, par conséquent, il ne peut pas être expliquée par une explication DN. De toute façon, si nous prenons en considération un type spécifique de système dynamique complexe, à savoir un *automate cellulaire* (*CA* désormais), dans ce cas un processus antimodulaire généré par un CA peut être expliqué, en quelque sorte, par la production d'une liste potentiellement très longue de déductions basée sur la condition initiale et la règle du CA (qui, comme une règle de CA, pour le théorème Curtis-Hedlund-Lyndon, a la même forme logique d'une loi scientifique), d'une

manière qui ressemble à une longue liste de explications DN pas à pas. Dans ce cas aussi, la compréhension humaine est exclue par la longueur éventuelle de la liste, mais, si l'on se conforme aux attentes des défenseurs post-neopositivistes du modèle DN d'explication, la compréhension n'est pas requise pour une bonne explication. Ainsi, d'une certaine manière, l'antimodularité, et par conséquent l'émergence faible, n'entrave pas l'explication DN, au moins dans le cas des CA et d'autres systèmes dont la dynamique suivre une règle universelle ressemblante à une loi scientifique, et à condition que la compréhensibilité ne soit prise pas en considération.

Je procède à considérer le modèle topologique de l'explication de Philippe Huneman, un type non-mécaniste d'explication qui est basé sur les propriétés topologiques de certaines descriptions abstraites d'un système. Je conclus que, étant la modularité elle-même, ainsi que son absence, une propriété topologique, la présence ou l'absence de modularité ne gêne pas, mais, au contraire, *permet* l'explication topologique.

Je me concentre ensuite sur un troisième type d'explication possible: les CA et les réseaux booléens dynamiques peuvent être considérés comme des systèmes *computationnels*. Comme tels, ils peuvent être sujets à *explication computationnel*. Je considère le cas d'essayer d'expliquer un CA computationnellement. Pour obtenir une explication computationnelle, d'abord le comportement du CA doit être *vu* comme une computation. Je souscris à une vue *intentionnelle* de la computaiton, mais soumise à certaines contraintes mathématiques, et j'essaye de délimiter le champ des dynamiques de système qui peuvent être considérées comme computationnelles. Consideré que certains CA peuvent en effet être considérés comme computationnels, je cherche à évaluer la possibilité de leur explication computationnelle. Pour donner une telle explication, le comportement du CA doit être soumis à rétro-ingénierie afin d'obtenir une spécification de la computation qu'il est censé effectuer. Mais, cette tâche de *specification mining* peut être computationnellement intraitable, et ainsi peut échouer. Même si une spécification globale se trouve, une bonne explication computationnelle équivaut à une forme d'analyse fonctionnelle modulaire hiérarchique, et cela est obtenue par extraction récursive des spécifications des parties du code ou traitement du système par autres méthodes statiques ou dynamiques d'ingénierie inverse. Si ce processus échoue pour des raisons d'intraitabilité computationnelle, ou par manque d'exhaustivité de la hiérarchie fonctionnelle trouvée, le système finirait par être antimodulaire. Dans ce cas, l'antimodularité intrinsèque entraverait une forme compréhensible d'explication computationnelle, pour la même raison qu'elle affecte l'explication mécanistique, avec laquelle l'explication computationnelle, qui est une forme d'analyse fonctionnelle, montre une affinité stricte.

Je ensuite souligne la nécessité de la modularité de haut niveau hiérarchique non seulement pour les explications *a posteriori* d'un phénomène connu, mais aussi lors de la phase de la découverte scientifique, en particulier, comme déjà indiqué par James Woodward, pendant la recherche de relations causales entre pièces d'un mécanisme, soit à bas niveau et a un niveau supérieur. De même, je souligne que l'explication modulaire multi-niveaux est également essentielle lors du développement de programmes informatiques de la part des programmeurs humains.

Je subsume sous le concept d'*émergence explicative* tous les résultats sur l'infaisabilité de certaines explications à plusieurs niveaux et sur la disparition de compréhensibilité qui en résulte, dû à la survenue d'antimodularité, ainsi que tout autre cas dans lequels un système, pour des raisons computationnel, résiste un'explication compréhensible.

Je discute ensuite, en examinant une certaine littérature scientifique, la probabilité pour la recherche scientifique dans certains domaines d'encourir l'émergence antimodulaire, et je conclus qu'il est assez probable que certains cas d'antimodularité apparaissent, en particulier dans la biologie des systèmes.

Je dédie la dernière partie de ce chapitre à des considérations plus amples mais, probablement, moins soutenues et risquées. Tout d'abord, je esquisse une éventuelle vue métaphysique qui pourrait découler de le considérations sur l'antimodularité exposées avant: j'appelle ce point de vue *antiréalisme lié* (constrained antirealism). Cette position voit le monde empirique que nous percevons naturellement, ainsi que le monde décrit par la science, comme le résultat d'un processus de détection de modularité, en conséquence duquel les modules détectés constituent ce qui est communément connu sous la dénomination des *espèces naturelles*.

Consideré que la détection de la modularité est limitée par des facteurs de complexité de calcul insurmontables, et que pour cette raison, la découverte de la *meilleure* description modulaire est en principe exclue, il est peu probable que la subdivision du monde en espèces naturelles corresponde à sa meilleure subdivision possible. L'évaluation complète de cette position métaphysique nécessiterait cependant un large débat sur une hypothèse controversée, le *pancomputationalism*, et sur plusieurs positions en philosophie des mathématiques, ce qui constitue une discussion qui est préférable de laisser à une recherche ultérieure.

Enfin, je prends dans la dernière partie du chapitre une certaine liberté en tirant les possibles conséquences présumées d'un recours récente et croissante à méthodes computationnelles dans la science, sur l'histoire de la science. En commençant par la simulation de systèmes complexes, je réfléchis sur la plausibilité des simulations comme explications, en particulier dans les cas où le système est antimodulaire, et par conséquent la simulation peut être exécuté, mais le modèle dynamique sous-jacente est inintelligible, parce que le système est simulé à un niveau très bas et une redescription modulaire de haut niveau manque ou est trop approximative pour être utile comme explication.

Je considère ensuite la détection automatisée de modularité, utilisée pour trouver une structure dans des grands ensembles de données, basant ces considérations sur des cas réels d'exploration de données sur un corpus de littérature médico-biologique, dans lequel le système automatisé a découvert des relations fonctionnelles importantes qui avaient échappé à l'examination humaine.

Peut-être me permettant de tirer quelques conséquences extrêmes, je conclus en suggérant que cette utilisation croissante des méthodes computationnels dans la science pourrait être sur le point de provoquer un changement de paradigme majeur dans certaine disciplines.

L'objet de cette thèse est multiforme et pas facile à classer: étant au sujet de les conséquences de l'antimodularité, une propriété de certains systèmes, sur la possible explication scientifique de ces systèmes, il est une œuvre de philosophie de la science.

Consideré que la propriété proposée, l'antimodularité, dépend de certaines contraintes computationnelles qui affectent la détection de la modularité, et que je recommande, en relation avec la discussion sur l'explication computationnelle, une conception intentionnelle de la computation, alors ceci est une œuvre de philosophie de l'informatique ou de la computation, dans le double sens d'appliquer certaines notions computationnelles à la réflexion philosophique, et de proposer une réflexion philosophique sur la notion de computation elle-même.

Considéré que les systèmes montrant antimodularité sont susceptibles de se trouver parmi les systèmes biologiques, de la discussion de longue date en biologie à propos de la modularité, et d'une foule d'exemples que je prends de cette discipline, ceci est même un travail de philosophie de la biologie. En ce qui concerne l'explication, je embrasse explicitement une position épistémique, centrée sur la notion de *niveaux de description*, qui sont des dispositifs épistémiques, de sorte que le présent travail a un aspect épistémologique.

Et, comme probablement toutes les position épistémologique, ella a également une portée métaphysique, que je tente d'esquisser vers la fin de ce chapitre[1].

Enfin, cette thèse fait usage de toutes les discussions théoriques mentionnées ci-dessus pour faire la lumière sur leurs conséquences possibles sur la pratique de la science, faisant allusion à la possibilité que un changement historique majeur, peut-être un changement de paradigme, est en cours ou sur le point de se produire dans la science.

Donc, dans un sens, cela est une thèse d'histoire des sciences. Bien que nécessitant encore l'observation et la preuve, je pense que cette hypothèse historique pourrait nous donner une indication de l'ampleur de l'impact que l'adoption généralisée de méthodes de calcul a eu ou pourrait être sur le point d'avoir sur la science telle que nous l'avons connue.

18.1 Modularité

Je commence cette clarification de la notion de modularité avec une esquisse historique: la modularité semble être un concept de base et généralisée, qui a probablement été conçu plus d'une fois, dans des domaines théoriques et pratiques partiellement indépendants et diversifiés, depuis longtemps. Néanmoins, la réflexion philosophique moderne sur la modularité a commencée dans la seconde moitié du XXe siècle, avec la contribution particulièrement pertinente de Herbert Simon. Oeuvrant dans le domaine de l'économétrie, Simon est venu à une conception de la modularité sous la forme du systèmes hiérarchiques *quasi-décomposables*[2], c'est à dire des systèmes qui peuvent être vus, au moins en première approximation, comme des systèmes hiérarchiquement décomposable de manière récursive en ensembles de sous-systèmes robuste et partiellement indépendants. Ce point de vue sur la quasi-décomposabilité, qui a ensuite influencé de nombreux autres auteurs dans différents domaines, est l'idée de base qui inspire ma proposition sur la modularité.

Dans ce travail, j'examine une conception possible de la modularité dans les systèmes complexes, et j'explore les conséquences de la présence de modularité ou de sa absence, l'antimodularité[3] sur l'explication du comportement de ces systèmes. En fait, je n'applique pas la notion de modularité aux systèmes réels, mais à leurs *descriptions* et re-descriptions de descriptions, où une (re)description est comprise, de préférence, comme une computation qui prend une description et la traite de manière à donner une autre description. En prenant une position largement *épistémique*, si pas une pleinement antirealistique (une position qui sera mieux expliquée dans la section 18.5.1 de cette chapitre), selon la position de Cory Wright et William Bechtel sur les explications mécanistes[4], je considère les explications scientifiques comme des dispositifs *épistémiques*, basés sur des *descriptions* des phénomènes, liés à la communication humaine et nécessitants au moins d'un minimum de *intelligibilité* cognitive. En conséquence, je suis intéressé à définir la modularité comme une caractéristique *des descriptions*, qui, si elle est présente, permet certains types d'explication. Quoique c'est la section II à être consacrée à un exposé complet de la modularité et des autres concepts impliqués, je donne ici une explication schématique de ce que je propose dans la thèse.

[1] Et vers la fin du chapitre d'introduction dans le texte en Anglais.
[2] Voir la séminale Simon (1962).
[3] La propriété de l'antimodularité n'est pas, comme nous allons voir, simplement absence de modularité.
[4] Conception épistémique opposée à une conception *ontique* de l'explication mécaniste. Voir la section 18.1.9, 18.4.3, ainsi que Bechtel & Abrahamsen (2005) et Wright (2012).

18.1.1 Modularité dans les systèmes complexes

Procédant le long des lignes exprimées ci-dessus, j'essaie de définir la propriété de la *modularité* dans les systèmes complexes, comme la possibilité pour les systèmes de ce genre d'être décrits comme un ensemble de *modules* faiblement liés, c'est à dire, un ensemble de sous-systèmes robustes et bien-définis, avec leurs pièces internes fortement interconnectées, et chaque sous-système partiellement indépendant du contexte externe, étant seulement faiblement connecté à d'autres sous-systèmes. J'étends cette vue de la modularité à celle de la pleine description *hiérarchique* d'un système en termes de niveaux "supérieurs" et "inférieurs" de description, dont chacun est constitué par modules, et où, à l'exception du niveau le plus bas, chaque des modules est un *macromodule*, c'est à dire, il peut être considéré à son tour comme caractérisé par une organisation interne modulaire de *micromodules*, et ainsi de suite, de manière récursive. Comme on a dit, tout cela concerne les descriptions, et non pas des ensembles d'objets du monde réel (ce qui est conforme à l'essence d'une vue épistémique).

Bien que la distinction macro/micro à l'égard de la modularité dépende bien sûr du choix d'un niveau particulier de description, le point à souligner ici est que l'ensemble de la description modulaire hiérarchique se révèle dépendre, en raison de la définition même de modularité, du *choix* de la part de l'observateur d'une *relation* significative *particulière* entre les parties élémentaires du système, et ce précisément en raison de la façon dont le concept de *module* est défini: un module est un sous-ensemble des parties d'un tout qui sont *reliées* les unes aux autres d'une manière plus forte que la façon dont elles sont reliées à des parties externes au module où elles se trouvent. La reconnaissance d'un sous-ensemble en tant que module, nécessite donc qu'une *relation* entre les parties soit prise en considération en premier lieu, et, en fonction de la relation spécifique considérée, la structure modulaire identifiable peut changer.

Cette définition de la modularité hiérarchique présuppose bien entendu qu'un système complexe soit composé de pièces élémentaires distinctes et reliées, et cela, à son tour, est dû au choix d'une description élémentaire atomique du système: le choix de l'ensemble des *pièces* et celui de la *relation* entre eux consistent en le choix, de la part de l'observateur, selon ses intérêts, de ce que je voudrais appeler une *description préférée* du système. Habituellement, il y a une description "naturelle" du plus bas niveau d'un système en termes de pièces élémentaires, souvent suggérée par les propriétés physiques du système combinées avec les intérêts du chercheur: par exemple, en biologie, un tissu est naturellement décrit comme composé de cellules, une cellule est naturellement décrite comme un système complexe composé principalement de macromolécules interagissantes, et dans les sciences sociales la société est naturellement décrite comme composée d'individus. Le point à souligner est que la modularité hiérarchique est *relative* à un tel choix, étant notamment en fonction du choix de la *relation* entre les parties élémentaires du système (qui généralement est un choix moins contrainte que ce des parties elles-mêmes), que la structure modulaire hiérarchique va dependre. Par exemple, dans une société, nous pouvons considérer les liens affectifs entre les individus, ou, alternativement, nous pourrions choisir des relations de subordination. Ces deux différentes descriptions du système entraînerait très probablement à deux différentes descriptions modulaires hiérarchiques, parce qu'un module est défini comme un sous-système d'éléments fortement interconnectés entre eux et faiblement liés à l'environnement, et cette "connexion" est précisément la relation entre les parties élémentaires considérée dans la description préférée choisie du système: dans les cas d'exemples, une des descriptions est la relation de lien affectif, l'autre celle de pouvoir d'influence.

18.1.2 Modularité, décomposabilité, et économie de la description

La modularité se manifeste comme la possibilité de *décomposer* un système[5] dans sous-systèmes reconnaissables et suffisamment définis, chacun composé de pièces qui sont plus fortement liées les unes aux autres que à parties appartenantes à d'autres modules ou à l'environnement extérieur. C'est la présence de ces variations d'intensité des relations entre les couples de parties du système, ce qui permet la reconnaissance de la modularité: si toutes les parties fussent également reliées les unes aux autres, les modules n'apparaîtraient pas, car un module est (informellement) défini comme un sous-système dont la force de liaison avec le reste du système est *inférieur* (en moyenne) à celle de la connexion entre les parties internes du module. Comme indiqué ci-dessus, la modularité est obtenue par rapport à la relation spécifique entre les parties de bas niveau que nous prenons en considération. Cela est une conception tout à fait semblable à l'originale proposée par Herbert Simon, celle de *quasi-décomposabilité*. La quasi-décomposabilité permet au système d'origine d'être représenté comme un ensemble de sous-systèmes connectés, et cette décomposition peut être réitérée jusqu'à l'obtention d'une description hiérarchique complete. Le point crucial est que le système original, composé de ses parties élémentaires, est donc descriptible dans une manière de haut niveau, sous la forme d'un *autre* système dont les parties correspondent chacune à l'un des modules du système d'origine. Ainsi, la description de haut niveau se révèle être *plus simple* que celle du niveau bas, parce que, en la description de haut niveau, des *groupes* entiers (les modules) de pièces de bas niveau sont représentés comme parties de haut niveau *individuelles*, et en conséquence les parties du niveau supérieur sont moins nombreuses que celles de bas niveau. Si le système que nous décrivons en cette façon est statique, comme par exemple la liste des membres composant le personnel d'une organisation, la description de haut niveau apparaît généralement plus économique et plus claire que la liste originale. L'exemple typique est celui des organigrammes. Dans un organigramme, chaque groupe, constitué des personnes qui travaillent dans le même bureau, est représenté par un seul élément, étiqueté avec le nom du bureau. Le nom du bureau représente le nom agrégé du groupe de personnes qui travaillent dans le bureau.

18.1.3 Modules comme pièces de haut niveau similaires répétées

Il y a une autre amélioration possible de l'économie de description d'un système complexe, s'il est possible d'y déceler plusieurs sous-systèmes qui résultent identiques ou tellement similaires à être éventuellement considérés comme la répétition d'un modèle unique. Dans ce cas, en dehors de l'économie de description due à l'agrégation[6], même une description modulaire qui comprît plus d'un des modules identiques, pourrait être simplifiée en remplaçant chaque occurrence de ce type de modules avec une référence au modèle commun, qui ne sera alors nécessaire de décrire qu'une seule fois. Cette forme de modularité est particulièrement utile dans l'ingénierie, et elle est essentiellement à la base de la conception d'objets complexes qui sont généralement composés de *pièces standards* identiques ou presque identiques, qui se présentent en plusieurs exemplaires dans l'objet.

18.1.4 Modularité structurelle et dynamique

Il est facilement concevable que la modularité peut concerner pas seulement la *structure* d'un système, mais aussi son fonctionnement dynamique: il est concevable, par exemple, et même

[5] Bien sûr, avec «système» ici, je veux dire une *description* d'un système. Dans ce qui suit, je vais souvent utiliser le terme "système" simpliciter, pour signifier sa description standard, généralement sa "description préférée".

[6] Mentionné dans la section précédente.

évident, que la modularité dans la structure d'un programme d'ordinateur (dont la structure est une liste d'instructions) produit une modularité dans l'exécution dynamique du programme, car un programme d'ordinateur est non seulement une liste d'instructions statiques, mais il est censé être *exécuté*, donc la modularité de la liste devrait être reflété dans la modularité dynamique du programme.

La relation entre la modularité structurelle et la modularité dynamique ne se révèle être pas toujours une relation simple: les aspects structurels et dynamiques peuvent être associés mais aussi découplés, bien que dans la plupart des systèmes dynamiques leur structure physique modulaire *induit* une forme de fonctionnement dynamique modulaire, étant donné que dans les systèmes dynamiques leur dynamique est effectuée *sur* la structure prédéfinie du système et cette dynamique est donc contrainte par la structure du système. La relation entre la modularité structurelle et la modularité dynamique est cependant pas complètement claire, et dans la section 6 de cette thèse je vais plus profondément l'examiner et la discuter.

18.1.5 Quasi-décomposabilité et agrégabilité

Une forme de modularité dynamique proposée en début des années '60 par Herbert Simon et Albert Ando[7] dérive de la quasi-décomposabilité du système, et plus particulièrement de la quasi-décomposabilité de son *modèle mathématique* décrivant la dynamique du système. Ce modèle mathématique est généralement une relation de récurrence, ou un système de relations de récurrence, dans lequel l'état de chaque partie élémentaire du système est représenté par une variable: cette équation représente une fonction de mise à jour qui détermine comment l'état des pièces du système varie au cours du temps, et il est donc un modèle mathématique de la dynamique du système. Dans un système qui est quasi-décomposable au sens de Herbert Simon, les variables de cette équation, qui peuvent être en grand nombre, parce qu'elles représentent les parties élémentaires interagissantes du système, peuvent être divisées (modulo une certaine approximation) dans une partition de sous-ensembles de variables, où chaque sous-ensemble est constitué des variables qui n'influencent que faiblement les variables à l'intérieur des autres sous-ensembles: cela correspond au fait que dans un système quasi-décomposable, par définition, les interactions entre certains groupes de parties (c'est-à-dire, entre le modules) sont faibles, tandis que les interactions entre le parties composantes le même module sont plus fortes. De cette façon, la dynamique de chaque module peut être considérée comme évoluante dans le temps en manière semi-indépendante de la dynamique des autres modules, et, en conséquence, les équations décrivantes ces dynamiques semi-indépendantes se révèlent être l'une semi-indépendante de l'autre. Ces équations régissantes groupes semi-indépendants de variables, peuvent alors être considérées comme des *modules fonctionnels*, une re-description modulaire du modèle mathématique original décrivant la dynamique globale du système. La modularité des systèmes quasi-décomposables détermine aussi une sorte de modularité dynamique, ou *de processus*, sous la forme d'un découplage de la dynamique temporelle entre les parties du système: la dynamique *à l'intérieur* des modules est plus rapide que la dynamique des interactions *entre* le modules.

Étant données les conditions décrites ci-dessus, dans certains cas favorables qui dépendent de la forme des équations modulaires, la dynamique globale du système, initialement décrite par la fonction de mise à jour globale (dans laquelle l'état de chaque partie élémentaire est décrit par une variable) peut être, modulo une certaine imprécision acceptée, re-décrite à nouveau sous la forme d'une autre, *plus simple* fonction globale de mise à jour. Cette fonction de mise à jour est plus simple que celle d'origine, parce que dans la nouvelle fonction de mise à jour chaque variable représente une *valeur agrégée* de toutes les variables contenues dans chacun des modules

[7] Simon & Ando (1961).

fonctionnels décrits ci-dessus: le nombre des variables qui doivent être prises en considération pour modéliser la dynamique globale du système est ainsi réduit. Lorsque cette condition subsiste le système est dit *aggregable* (pas tous les systèmes dynamiques sont aggregable), et cela est évidemment une autre forme de *économie de description* permise par la présence de modularité, dans ce cas économie du modèle mathématique. Le prix à payer est un montant d'imprécision qui dépend du fait que, afin d'agréger avec succès la dynamique du système, certaines interactions entre les parties du système dont la force est inférieure à un seuil choisie, sont considérées comme nulles. L'approximation pourrait se reveler inacceptable dans les systèmes non-linéaires, où le comportement à long terme de la description simplifiée pourrait diverger trop du comportement réel du système. Le point à souligner est que, même ici, des choix de la part de l'observateur sont impliqués: le choix de la description préférée (qui, cependant, dans de nombreux cas, est déjà donnée), et un choix sur le degré d'imprécision, acceptable ou non en fonction des objectifs de l'observateur.

Un problème très important qui affecte l'agrégabilité est que cette propriété s'est avérée être une tâche computationelle infaisable: il y a des preuves, dans Kreinovich & Shpak (2006) et Kreinovich & Shpak (2008), que l'agrégabilité, et même l'agrégabilité *approximative*, déjà en systèmes *linéaires*, est *NP-hard*. Cela signifie[8] que, pour un modèle mathématique de la dynamique du système avec un grand nombre de variables, il n'y a pas aucune méthode générale algorithmique qui peut toujours produire dans un temps possible une plausible version agregée simplifiée du modèle. En d'autres termes, cela signifie que la détection de modularité dans le modèle dynamique d'un système complexe est une tâche computationelle infaisable, et que *l'on ne doit pas s'attendre que la modularité dynamique peut être trouvée avec une méthode générale.*

Néanmoins, l'agrégation peut dans de nombreux cas être trouvée plus facilement si nous avons une certaine connaissance qui peut nous guider dans la répartition des variables en sous-ensembles semi-indépendants. Par exemple, dans le cas des réseaux génétiques, nous pourrions savoir sur des bases empiriques qu'un certain groupe de gènes manifestent toujours co-expression, et donc les variables qui représentent ces gènes peuvent être regroupées. Cela pourrait simplifier beaucoup la tâche de trouver une bonne agrégation, une tâche qui est en principe, comme on a dit, trop exigeante d'un point de vue computationnel.

18.1.6 La modularité dans les systèmes dynamiques discrets

Il y a des cas complexes dans lesquels la forme structurelle et la forme dynamique de modularité ne sont pas facilement séparables, car une structure de haut niveau du système, elle-même "émerge"[9] à partir des dynamiques complexes de bas niveau du système. Cela est typique de certains *systèmes dynamiques discrets* complexes, tels que certains réseaux booléens, ou certains automates cellulaires. Alors que je dédie certaines sections du chapitre 5 à expliquer les bases des systèmes dynamiques discrets, et plus particulièrement d'une sous-classe d'entre eux, le soi-disants *automates cellulaires* (*CA* désormais), un très court aperçu peut être donné ici: ces systèmes sont composés d'un certain nombre de pièces simples, dont chacune, à un moment donné, se trouve dans un *état* particulier, choisi parmi un ensemble fini d'états distincts possibles. Il est d'usage de considérer chaque état distincte comme un *symbole*, et d'examiner l'ensemble des symboles possibles comme un *alphabet* (pensons, dans le cas le plus simple, aux symboles *0* et *1*). Non seulement les symboles sont discrets, mais le temps est discret aussi: dans ces systèmes discrets le temps procède point par point, par *time steps* distinctes, que nous pouvons

[8] Voir la section 18.3.

[9] J'utilise le terme «émergence» ici d'une manière intuitive, alors que cette question sera examinée brièvement plus tard dans ce chapitre et, plus profondément, dans les chapitres majeurs de cette thèse.

appeler t_1, t_2, et ainsi de suite. À un moment donné, l'ensemble des états dans lesquels toutes les parties du système se trouvent être, constitue la *configuration* globale du système. Les états de toutes les parties du système sont mis à jour en synchronie à chaque pas de temps successif selon une règle déterministe, une règle qui peut être la même pour toutes les parties du système (comme cela est le cas dans les CA) ou différente pour chaque partie. À un moment initial conventionnel, appelons-le t_0, le système est dans la *configuration initiale*. L'évolution du système est la séquence de configurations globales successives il atteint à mesure que le temps passe, à partir de la configuration initiale. Les classes typiques de tels systèmes sont, comme on a dit, les CA, et une classe plus large, celle des réseaux discrèts. La dynamique de cette évolution peut, pour certains systèmes, être extrêmement complexe, et, dans certains cas, cette capacité dynamique peut résulter manifestement équivalente à la puissance de calcul des machines de Turing universelles, qui sont réputées être[10] la classe la plus puissante des systèmes informatiques. Pour cette raison, le comportement dans le temps des systèmes complexes est, en général, assez difficile à prédire, et, dans le cas de capacités de niveau Turing, il est en principe *algorithmiquement indécidable* en général[11].

Une forme de modularité peut être induite ou apparaître dans certains systèmes dynamiques discrets, soit en leur imposant un état initial spécifique, ou, dans certains cas, par sa émergence spontanée dans le système après un certain temps le long de son évolution, quel que soit la spécifique configuration initiale: un phénomène qui est une forme d'*auto-organisation*. La modularité dans ce sens se résume au fait que certains sous-ensembles de la configuration globale du système viennent à être partiellement ou totalement *gelés* après un certain temps, c'est-à-dire, ils viennent à constituer des parties immuables ou peu changeantes de la configuration globale du système, et de cette façon ils viennent à isoler partiellement autres sous-ensembles de la configuration, en empêchant la propagation de l'influence de chacun de ces sous-ensembles vers les autres. Cette isolation, implicitement, impose une structure virtuelle de haut niveau au-dessus de la structure originale de bas niveau, une superstructure qui peut être vue comme un ensemble de modules dynamiques (les parties non gelées de la configuration) lâchement reliés entre eux (au moyen de les chemins résiduels de connexion qui ne sont pas interrompus par les parties gelées). Pour un exemple d'un réseau discret avec une modularité de haut niveau apparaissant au cours de son évolution, voir fig. 18.1.

D'un manière légèrement différente, l'auto-organisation peut apparaître, en particulier dans les CA, comme l'émergence de sous-configurations de la configuration globale très localisées, partiellement robustes, bien délimitées, et seulement partiellement changeantes, c'est-à-dire les soi-disants *planeurs* (*gliders*), qui apparaissent, pour ainsi dire, se *déplacer* dans la configuration du système. Un exemple est la fig. 18.2.

18.1.7 Modularité dans les systèmes computationnels

Étant une forme de système dynamique discret, un système computationnel peut bien sûr présenter modularité. Les ordinateurs universels ordinaires du monde réel sont des machines hautement modulaires déjà au niveau que l'on appelle le "matériel" (le "hardware"). Mais une autre forme très importante de modularité concerne les *programmes* informatiques. Un programme est essentiellement constitué par une liste d'instructions que le matériel informatique "exécute" pas à pas. Bien sûr, une telle liste peut être dépourvue de modularité apparente, ou elle peut plutôt être structurée par le programmeur d'une manière évidemment modulaire, en la subdivisant en sous-listes disjoints, dont chacune contient principalement des instructions concernantes seulement un

[10] Prise la thèse de Church-Turing pour acquis. Pour une explication, voir l'Appendice, section 17.3.
[11] Comme une conséquence de l'indécidabilité du problème de l'arrêt. Voir la section 17.2.6.

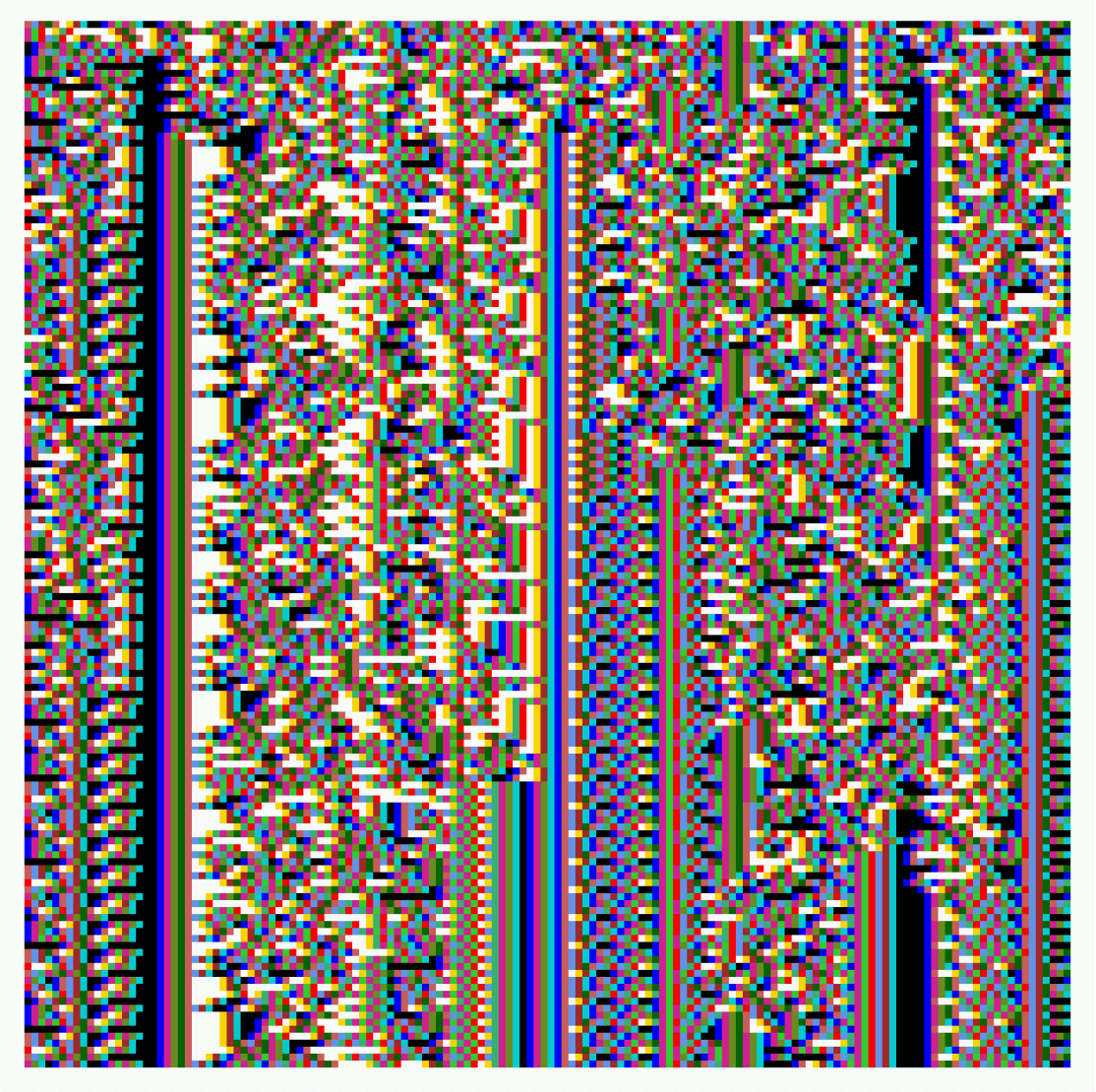

Figure 18.1: une évolution partielle dans le temps d'un réseau discret. Le temps s'écoule de haut en bas, chaque rangée de pixels représentante la configuration du système à chaque pas de temps. Chaque pixel représente l'état de chacune des parties élémentaires du réseau, son *nœuds*. Les lignes verticales épaisses, noires ou à motifs, qui peuvent être distinguées dans l'image, sont des sous-ensembles "gelés" de la configuration. Ils induisent une forme de modularité de haut niveau, en agissant comme "murs" plus ou moins impénétrables, et en cette façon ils rendent le système quasi-décomposable en plusieurs sous-systèmes indépendants. (Image tirée de la Galerie DDLab de Andrew Wuensche, http://uncomp.uwe.ac.uk/wuensche/gallery/ddlab_gallery.html).

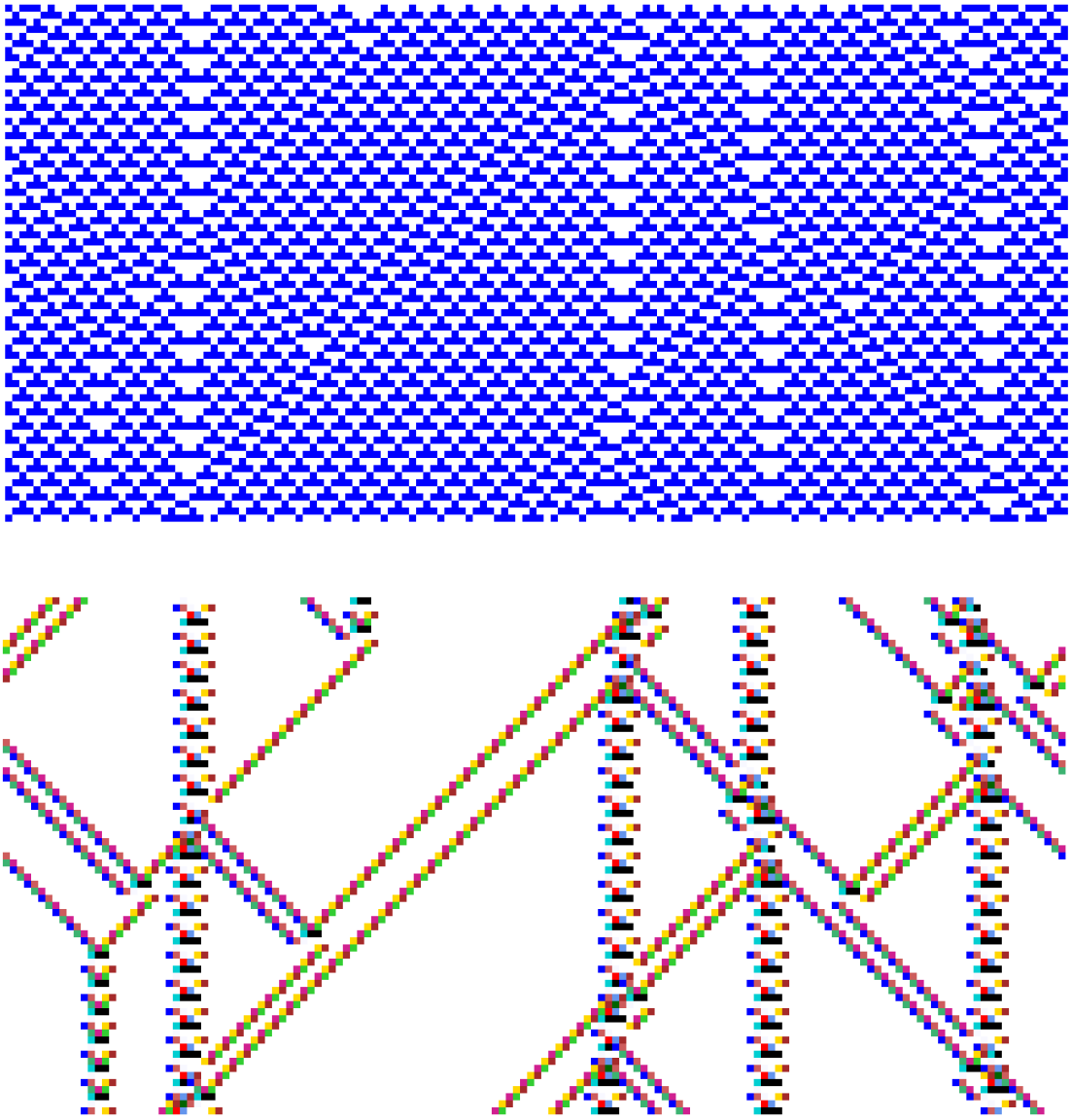

Figure 18.2: planeurs dans un CA, la soi-disant *Règle 54*, selon la classification de Stephen Wolfram (voir Wolfram 2002). Le temps s'écoule de haut en bas, chaque rangée de pixels représentante la configuration globale du système à chaque pas de temps. Chaque pixel représente l'état de chacune des parties élémentaires du CA, ses *cellules*. Ci-dessus: la séquence d'états du CA. Bas: après avoir filtré les parties récurrentes du fond de la configuration, les planeurs apparaissent plus clairement, représentés au cours du temps par des lignes droites, représentantes la délocalisation progressive de ces sous-configurations à l'intérieur de la configuration globale (image originaire provenante de la Galerie DDLab de Andrew Wuensche, http://uncomp.uwe.ac.uk/wuensche/gallery/r54_filted.gif, modifié) .

ensemble limité de variables internes à la sous-liste, à l'exception d'une ensemble de variables d'"input" et d'une ensemble de variables d'"output" qui sont accessibles aussi par des instructions appartenantes à d'autres sous-listes. De cette façon, chacune de ces sous-listes peut être considérée comme un module, et le transfert limité et contrôlé d'informations entre les différents modules est réalisé par les variables d'input/output, c'est-à-dire des ensembles distincts de variables qui sont les seules à être accessibles et manipulables par parties du programme externes au module: un tels module peut être considéré comme une "boîte noire" avec un ensemble limité de lignes d'entrée et de sortie. De cette façon, la propriété typique des modules est réalisée: en considérant comme le rapport choisi entre les parties de la liste des instructions la relation entre une instruction et une variable sur laquelle elle agit, il peut être facilement vu que, quand en une sous-liste d'instructions les variables internes, celles qui ne figurent pas dans les ensembles d'entrée et de sortie, sont d'usage essentiellement interne, et sont moins souvent ou (mieux) jamais manipulées par des sous-listes disjointes externes, cette sous-liste d'instructions peut être considérée comme un module, doté de cohésion interne et structurellement plutot indépendant des autres modules. Dans un programme non-modulaire, la modification du programme de la part du programmeur peut resulter assez difficile à mettre en oeuvre et à controller, car un changement dans une partie du programme pourrait affecter pièces du programme potentiellement très lointaines. Par contre, compte tenu de la connectivité limitée entre les modules, en particulier dans le cas où seulement les variables d'entrée et de sortie sont accessibles aux modules externes, un changement interne à un module, et affectant seulement les variables internes, ne se répande pas indistinctement à d'autres modules, et donc son effets sont plus faciles à contrôler. En général, dans la programmation de l'ordinateur, ce qui est recherché est l'*haute cohésion* interne des modules et un *couplage faible* entre eux.

La modularité sous forme de des parties similaires récurrentes ou identiques est à la base d'une forme connexe de modularité des programmes informatiques, compatible avec celle esquissé ci-dessus: si plus d'un des modules remplit la même fonction d'entrée/sortie, au lieu d'explicitement le réitérer, sous la forme de copies répétées de la séquence de ses instructions à l'intérieur de la liste du programme, le module peut simplement être appelé à plusieurs reprises dans différentes parties du programme, chaque fois avec des configurations d'entrée distinctes, et en recevant par le module des configurations de sortie éventuellement distinctes lorsqu'il termine son exécution. Modules vu de cette façon, comme des sous-programmes appelables, sont souvent appelés *sous-programmes* ou *procédures*. Cette façon de structurer les programmes peut améliorer considérablement leur fiabilité, parce que les tests de chaque procédure peut alors être fait qu'une seule fois, en tant que le système complet est assemblé à partir de modules déjà testés.

18.1.8 Modularité hiérarchique, niveaux, robustesse et validité

Je vais essayer ici de clarifier l'importance d'une propriété des modules que j'ai simplement mentionné, jusqu'ici: la *robustesse*. Intuitivement, pour une description modulaire d'un système dynamique, robustesse signifie qu'un module au niveau supérieur doit résister à une certaine gamme de perturbations au niveau inférieur, en conservant son identité distincte et en persistant malgré la variation d'etat de ses composants constituants de bas niveau. Ou, dans certains cas, un module robuste devra rester le même, malgré les substitutions possibles de certains de ses composants de bas niveau. Dans les descriptions de réseaux modulaires[12], qui sont descriptions modulaires *structurelles*, un module est considéré comme robuste quand il ne change pas son identité malgré la soustraction ou l'ajout de certains liens entre les sous-jacents composants de

[12] Voir la section 18.2.1.

bas niveau[13].

Dans les systèmes dynamiques, un module qui peut subsumer un ensemble aléatoire *quel que soit* des composants de niveau inférieur serait un module très robuste, car il manifesterait une grande persistance face à des variations: il ne changerait jamais malgré tout changement au niveau inférieur! Mais ce serait une sorte de robustesse *banale*. Un module dynamique suffisamment robuste du point de vue explicatif ne doit pas être trop évanescent, ni il devrait avoir son etat fixé: Il devra plutôt être capable de suivre, d'une manière grossière, les dynamiques sous-jacentes. Une robustesse adéquate de modules est essentielle au moment de la production de descriptions hiérarchiques des systèmes dynamiques que nous voulons employer pour l'explication scientifique, parce que un module de haut niveau, pour être explicatif, doit être censé suivre fidèlement la dynamique de bas niveau, bien que à une résolution temporelle, et souvent spatiale, inférieure, ou à une précision inférieure. Un module dans la description d'un système dynamique, pour être utile pour l'explication scientifique, devrait être robuste à un degré qui est choisi par l'observateur, selon ses intérêts: cela est un autre aspect de la relativité de la modularité hiérarchique au choix de l'observateur. En tout cas, un module de haut niveau ne doit pas être trop évanescent, sinon son niveau de description serait inutilisable pour des explications, et il ne devrait pas être trop robuste, car dans ce cas il n'y aurait pas du tout d'effet explicatif: un module le plus possible robuste est gelé, c'est un module qui reste dans le même etat pour toutes les combinaisons possibles de toutes les parties du système, et il constituerait simplement un "nom" global de l'ensemble du système (cela puisse être utile dans certains cas, afin d'identifier les phénomènes pour la première fois).

Mais, la redescription modulaire de haut niveau dans les systèmes dynamiques est une affaire compliquée: la dynamique de la description modulaire de haut niveau ne doit pas *diverger* (au moins pas trop) par rapport à la sous-jacent évolution dynamique de bas niveau de la description préférée du système. Une description modulaire doit être *valide* (pour reprendre la terminologie de la modélisation computationnelle scientifique), afin d'être explicativement utile: elle doit suivre avec une certaine précision, mais à une échelle de plus gros grain, l'évolution dynamique du système. Une description modulaire non valide est explicativement inutile. Ceci peut être mieux comprise par un exemple. Imaginez que nous fassions fonctionner un système dynamique discret une seule fois, afin d'obtenir une partie de son évolution, comme en fig. 18.2. Nous pourrions immédiatement élaborer une représentation modulaire de haut niveau de cette course dynamique, en prenant simplement comme un module chacune des lignes diagonales représentantes les trajectoires des planeurs, et donc nous pourrions produire une explication de haut niveau du ce segment de l'évolution en ne mentionnant que "trajectoires de planeur" (quelque chose du genre de "le troisième planeur entre en collision avec le deuxième et le détruit, continuant vers la grande colonne verticale de gauche", et ainsi de suite ...). Pourrait-cela constituer une description de haut niveau *valide* de la dynamique du système? Très probablement non, parce que nous avons basé notre détection de modularité seulement sur un segment limité de l'évolution possible du système: la prochaine fois que l'on commence la dynamique du système avec une configuration initiale même légèrement différente, on aurait la plus grande probabilité de produire un segment complètement différent d'évolution, et l'ancien description serait rendus inutiles. Le point est, quand on redécrit ces types de systèmes dynamiques à un niveau supérieur, il ne faut pas chercher à redécrire des motifs éphémères, des configurations uniques, de leur évolution, mais seulement des sous-configurations suffisamment robustes, et qui apparaissent dans la dynamique du système à haute fréquence et régularité: seulement de cette façon nous pourrions espérer obtenir

[13] Certains algorithmes de détection de modularité dans les réseaux effectuent ce genre de test afin d'évaluer la robustesse modulaire. Voir la section 3.2.1.2.

une description modulaire de haut niveau *valide*[14]. En des termes qui seront précisés dans la section 18.4.3, nous pourrions de même dire qu'un module devrait idéalement représenter, à haut niveau, une *fonction*, exécutée dans le but de contribuer au comportement global specifique du système. La décomposition fonctionnelle n'est pas arbitraire: toute dissection arbitraire d'un système en n'importe quelles parties ne peut être considérée comme fonctionnelle. Une décomposition arbitraire, conduisant à une description non valide, ne serait pas considéré explicative d'une manière fonctionnelle ou mécaniste.

18.1.9 Modularité et explication

Il semble que la modularité soit liée à l'explication en diverses et fondamentales façons. Déjà les premiers articles de Herbert Simon sur les systèmes quasi-décomposables soulignent que la formule qui représente la dynamique agrégée[15] d'un système quasi-décomposable est *plus simple* que la formule de sa dynamique d'origine, ce qui signifie que l'agrégabilité produit économie de description. Puisque une explication scientifique de la dynamique du système (au moins le type déductive-nomologique d'explication[16]) emploierait sûrement cette formule, on obtient ainsi une forme d'économie d'explication.

En général, la modularité devrait permettre une forme de description "à gros grain" ("coarse-grained"), comprise comme l'opération de prendre un système complexe représenté comme un ensemble de nombreuses parties, de partitionner cet ensemble en sous-ensembles disjoints, et de considerer, à la place du système d'origine, un autre ensemble dans lequel chaque partie correspond à chacun des sous-ensembles disjoints de l'ensemble original. Cela est fondamentalement la même opération, qu'elle soit effectuée sur des ensembles de variables de l'équation, comme dans l'agrégation, où elle apporte modularité dynamique, ou sur un réseau, où la représentation originale peut être substituée par un réseau avec moins de noeuds, ou dans le cas des explications fonctionnelles et mécanistiques[17], où un groupe de pièces ou d'actions en interaction peut être considéré comme une fonction entière, ou un mécanisme entier, et un groupe de mécanismes peut être considéré comme une seule super-mécanisme, dont les parties sont les mécanismes simples. Cette situation s'applique également, dans un sens, à la modularité dans les systèmes informatiques, où une liste d'instructions peut être réécrite dans un langage de haut niveau dans lequel chaque instruction de haut niveau correspond à une sequence d'instructions de niveau inférieur: de plus, un langage de haut niveau est un exemple très typique de description "à gros grain". Dans chacun de ces cas, une économie de description est atteinte, et, sans doute, l'intelligibilité de l'explication est grandement facilitée.

Une autre forme d'économie de description est obtenue dans certaines descriptions modulaires lorsque plusieurs copies récurrentes du meme module peuvent être remplacées par une seule citation du modèle général de ce module, une forme de modularité qui, dans des programmes informatiques, correspond à l'appel du même sous-programme de différents points du programme.

Considérations d'économie ou d'intelligibilité à part, la modularité est *nécessaire* pour produire certains types d'explication. La stratégie explicative analytique de Robert Cummins, que nous allons aborder dans la section 18.4.3 et l'explication mécaniste aussi, explicitement plaident, afin

[14] On peut objecter que la robustesse de modules de haut niveau peut être obtenue par une correspondance complexe et pas immédiatement évidente entre groupes hétérogènes de sous-configurations éphémères de bas niveau et modules de haut niveau. Ceci, je pense, est pas une objection banale, surtout compte tenu de l'importance que je prête à la *relativité* de la modularité. Je vais examiner cette question, qui a des implications profondes, dans la section 14.5.2.

[15] Voir la section 18.1.5.

[16] Voir la section 18.4.4.

[17] Voir la section 18.4.3.

de produire une explication, la décomposition hiérarchique du fonctionnement du système. Bien sûr, cette décomposition est possible seulement dans le cas où une certaine forme de modularité fonctionnelle est présente dans le système, c'est-à-dire, lorsque les modules recherchés peut légitimement être considérés comme des modules fonctionnels. De même, l'idée d'une explication mécanistique semble exiger la découverte d'une coïncidence entre deux types de hiérarchies, à savoir entre une description structurelle et une description fonctionnelle du système, au moins dans la conception du mécanisme proposé par William Bechtel et son groupe: pour ces auteurs, qui ne voient pas l'explication mécaniste comme simplement réductionniste, il est essentiel que l'explication soit *à plusieurs niveaux*, ce qui correspond à une description fonctionnelle mécaniste hiérarchique du système. En adoptant une vue *épistémique* de l'explication[18], ces mêmes auteurs tout naturellement soulignent également l'importance de l'*intelligibilité* cognitive des explications, et cela peut être réalisé par la modularité des descriptions employées dans les explications.

Donc, en premier lieu, il semble qu'au moins les explications d'un certain type, à savoir les explications mécanistiques ou fonctionnelles, *nécessitent* de la modularité, même si on néglige la question de l'intelligibilité de ces explications.

Mais, la modularité hiérarchique permet également des explications à plusieurs niveaux qui augmentent certainement la *compréhension*. Étant donnée une décomposition hiérarchique mécaniste appropriée, un système peut être décrit à n'importe quel niveau de description souhaité, avec des résultats différents sur l'intelligibilité de l'explication: les niveaux à grains grossiers plus abstraits permettent une explication très simplifiée, ce qui induit généralement une meilleure compréhension, tandis que le choix de procéder à des niveaux inférieurs, plus détaillées, améliore l'information sur le système véhiculée par l'explication, probablement au détriment de la compréhension: l'explication la plus détaillée possible est celle qui décrit le système en termes des entités de plus bas niveau[19], et, dans de nombreux cas, la quantité d'informations contenues dans une telle description pourrait entraver son intelligibilité.

18.1.10 Quelques exemples d'applications de la modularité dans la recherche scientifique réelle

Je vais me concentrer ici sur quelques brèves considérations sur l'importance de la modularité dans la recherche et la pensée biologique, parce que la biologie est l'un des domaines dans lesquels la modularité a plus été au centre de l'attention ces derniers temps. Une observation évidente est que les organismes sont sans aucun doute modulaires, à de nombreux niveaux: ils sont, dans une vue biologique, composée de systèmes, d'organes, de cellules, de macromolécules. Il est moins évident si la modularité est présente à certains niveaux intermédiaires qui peuvent être vus comme des systèmes complexes, comprenant de nombreuses régions: par exemple, sont le génome, le protéome, ou les réseaux métaboliques modulaires?

Donc, une première question à poser est: Est-ce que l'évolution produit des architectures et des dynamiques modulaires dans les organismes? Et, si tel est le cas, a la modularité évolué par la sélection naturelle ou pour d'autres raisons? En dehors de l'étude empirique possible de ce problème, certaines considérations a priori ont semblé capable de l'éclairer, au moins depuis l'époque de Herbert Simon. Il y a un certain nombre d'arguments pointant à la conclusion que la sélection naturelle devrait effectivement conduire à l'organisation modulaire. Tous ces arguments sont essentiellement issus de la ligne de raisonnement suivante: dans un organisme

[18] Voir la section 18.4.3.

[19] Le niveau plus bas de description, qui correspond dans mon terminologie à la description preferée, est généralement une question de choix ou d'une convention, aussi selon Bechtel et ses co-auteurs. Voir les sections 18.4.7 et 11.1.5.

non-modulaire, totalement intégré, dans lequel chaque partie affecte potentiellement toutes les autres, un changement évolutif dans une partie pourrait affecter et éventuellement perturber les fonctions exercées par les autre parties, et, compte tenu de cela, le nombre de tentatives d'évolution potentiellement nécessaires pour obtenir un organisme encore fonctionnel après un changement dans l'un de ses parties serait énorme, et il est donc plausible soutenir que, si tel fût le cas, la sélection naturelle n'aurait eu pas le temps, en dépit de l'échelle géologique des temps de l'évolution sur cette planète, pour réaliser l'évolution de systèmes biologiques complexes. Cela est plus ou moins l'argument général pour l'évolution de la modularité proposé dans les années 60 par Herbert Simon[20] et adopté, avec variations, par de nombreux auteurs ultérieurs. À partir des œuvres de Stuart Kauffman dans les années '90, un argument alternatif (mais à mon avis pas si dissemblable, voir la section 7.1.2) est apparu, qui, tout en affirmant la modularité des systèmes biologiques, nie sa origine directe par la sélection naturelle: la modularité est plutôt une propriété auto-émergente d'une certaine classe de systèmes complexes dynamiques, propriété coïncidante avec la "congélation" de certains de leurs sous-systèmes dynamiques (voir ci-dessus, la section 18.1.6), qui n'apparaît pas par sélection directe, mais en vertu des caractéristiques intrinsèques, mathématiques, de ces systèmes[21]. Le génome (vu comme un ensemble de parties interagissantes, c'est-à-dire le réseau de régulation génétique) d'un organisme peut, selon Kauffman, être considéré, avec une certaine approximation, appartenant à cette classe de systèmes dotés d'une tendance à faire émerger la modularité spontanément, une classe qui se révèle être la classe des systèmes les plus *évolvables*: le rôle de la sélection naturelle aurait été celui de opérer une méta-sélection de la *classe* des systèmes évolvables sur laquelle, alors, réaliser son rôle sélective analytique plus fin, comme classiquement conçu dans le darwinisme[22], et cette classe est la classe de systèmes complexes qui, spontanément, montrent une certaine forme de modularité.

Ainsi, il semble, tout considéré, qu'il y a des raisons pour lesquelles les systèmes biologiques évolué doivent avoir de préférence une organisation modulaire. Beaucoup de ces systèmes sont si complexes et composés de si nombreuses parties, que la détection de leur modularité *fonctionnel*, permettant leur explication à plusieurs niveaux, serait d'une grande aide aussi pour la *compréhension* de ces systèmes.

En biologie, depuis les fin des années '90, certaines propositions sur la possibilité de voir des systèmes biologiques complexes comme composés de modules fonctionnels ont été directement inspirées par le point de vue de l'ingénieur sur les systèmes artificiels, notamment les circuits électriques: parmi les propositions les plus éminentes au sujet de cette vue, sont McAdams & Shapiro (1995) et hartwell:1999from[23]. Ce point de vue a été appliqué aux réseaux génétiques et métaboliques, où la grande spécificité des connexions électriques entre les composants d'un circuit électronique est substituée par la spécificité de la relation entre une protéine et son ligand, et l'ensemble du réseau biologique est représenté comme un *circuit numérique*, ce qui est équivalent (avec quelques différences, tenant en compte le temps de propagation des signaux) à des réseaux booléens. Dans ces circuits, chaque module est un spécimen d'un *composant standard*, reliés aux autres modules par câblage, et le circuit numérique complet peut être considéré comme une structure hiérarchique, dans laquelle chaque niveau est descriptible comme un circuit de pièces modulaires, interconnectées, et reproductibles, qui permettent aux modules numériques

[20] Avec la célèbre parabole des deux horlogers, voir la section 7.1.1 et, évidemment, Simon (1962).

[21] Ce genre d'explication fournie par Kauffman peut être considérée comme une forme d'explication *topologique*, selon le modèle récemment proposé par Philippe Huneman. Voir la section 18.4.5 et Huneman (2010).

[22] Je saisis l'occasion ici pour un avertissement: même quand j'écris de la sélection naturelle en termes intentionnels, je ne prône pas de la considérer comme un sujet intentionnel. C'est seulement, de toute évidence, une utile "façon de parler", largement courant en biologie.

[23] Pour une méta-réflexion sur les méthodes de la recherche biologique, la modularité et l'approche de l'ingénierie, on peut aussi voir lazebnik:2002radio.

de haut niveau de réaliser pratiquement tout circuit numérique, même ceux susceptibles d'être considérés comme des systèmes computationnels. Voyez fig. 18.3 pour une exemplification de la vue hiérarchique d'un circuit électronique numérique.

Les pièces modulaires peuvent être, dans le cas des réseaux génétiques, les gènes individuels et, à un niveau supérieur, des complexes de gènes comme les *opérons* des génomes bactériens. Tels composants de plus haut niveau auraient un rôle fonctionnel, comme dans le cas, par exemple, d'un opéron, qui contrôle la production d'un complexe d'enzymes effectuant une fonction métabolique spécifique. Une possible représentation schématique d'un "circuit génétique" est montrée dans la fig. 18.4.

Hartwell et al. (1999) propose que les termes linguistiques ("amplification", "correction d'erreurs", "détection de coïncidence", et ainsi de suite ..) correspondants à des fonctions de mi-niveau et de haut niveau réalisées par des modules à des niveaux hiérarchiques intermédiaires, constituent un vocabulaire de termes, essentiel pour la description fonctionnelle des systèmes biologiques.

18.2 Détection algorithmique de la modularité

J'examine ici des algorithmes pour la *détection de modularité* dans certaines classes de systèmes complexes, c'est-à-dire, algorithmes qui, étant donné un système complexe et une description élémentaire préférée de celui-ci, essayent de produire une description modulaire hiérarchique du système.

18.2.1 Détection de modularité dans les réseaux et complexité computationnelle

Je prends en considération, en particulier, algorithmes pour la détection de modularité dans les *réseaux*, parce que les modèles basés sur des réseaux ont émergé, dans la recherche récente, comme l'un des moyens privilégiés de représenter des systèmes complexes, en particulier les systèmes biologiques. Un réseau peut en général être considéré comme un ensemble de pièces, ses *nœuds*, reliés entre eux de différentes manières au travers des *liens* (une potentielle représentation graphique d'un réseau est fig. 18.5). Il y a deux principaux formes possibles de modularité dans les réseaux, qui ne sont pas incompatibles: structure de communauté et modularité basée sur le motifs de réseau[24]. Alors que la première est basée sur la conception typique de la modularité comme sous-systèmes robustes lâchement connectés, la seconde coïncide avec l'idée du modules comme pièces standard reproductibles.

Dans le chapitre 3 je produis une vue d'ensemble sur les principales méthodes proposées pour la détection des deux types de modularité dans les reseaux, avec une attention particulière à leur faisabilité computationnelle: il se trouve que la plupart des meilleurs algorithmes de détection de modularité dans les réseaux sont très exigeants du point de vue computationnel, et il y a aussi une limite théorique sur leur exactitude. Pour résumer, il a été prouvé que le constat automatisé de la *meilleure* description modulaire d'un système est entravé par une complexité insurmontable du temps de calcul: la tâche est *NP-complete*[25]. En outre, il se trouve que la plupart des algorithmes pour simplement approximer la détection optimale de modularité dans les réseaux, sont eux-mêmes fortement intensifs du point de vue computationnel. En général, il semble que la détection algorithmique de la modularité du réseau est affecté par un compromis

[24] Voir la section 3.
[25] Voir la section 18.3 et la section 17.4 de l'Appendice.

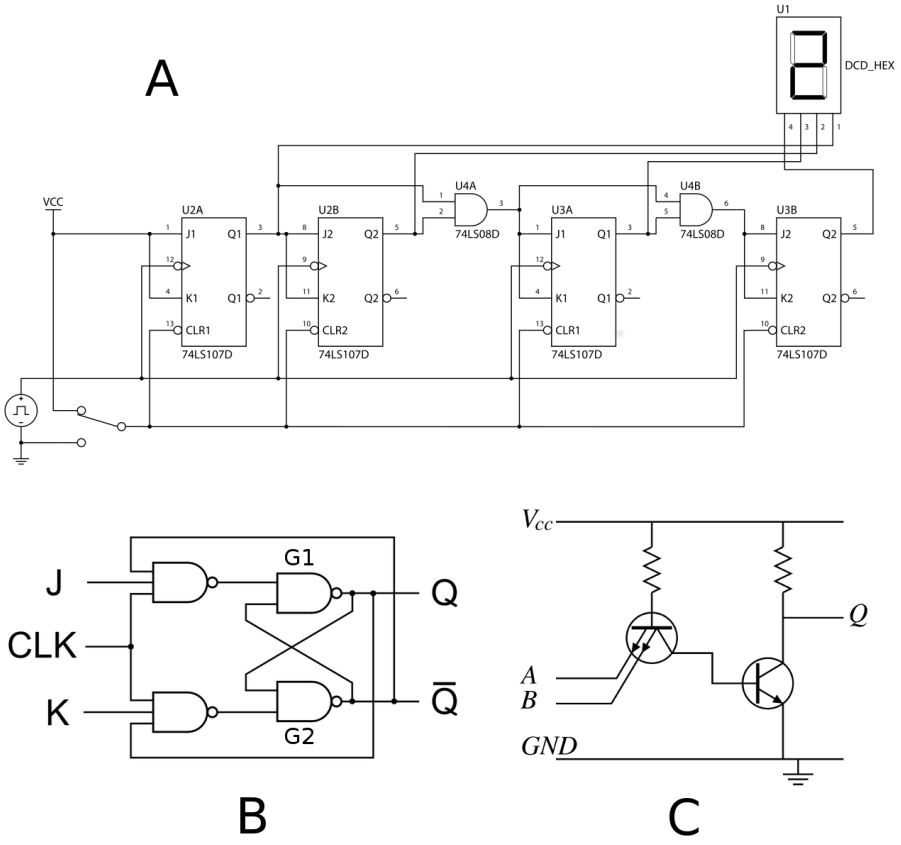

Figure 18.3: image A: un diagramme de haut niveau représentant un circuit numérique. Sauf pour quelques portes logiques simples ($U4A$ et $U4B$), la plupart des composants sont de plus haut niveau, et peuvent être considérés des modules effectuants des fonctions de niveau supérieur. Dans ce cas, chacun des composants étiquetés $U2A$, $U2B$, $U3A$, $U3B$ est un soi-disant *flip-flop J-K*, qui est un type de bascule ou de cellule de mémoire à 1 bit. Chaque bascule peut être vue (image B) comme composée à l'intérieur d'un certain nombre d'éléments plus simples, à savoir portes logiques NAND. Chacune des portes NAND à deux entrées étiquetées comme $G1$ et $G2$ dans l'image B sont structurées en interne comme un circuit composé de transistors et des résistances, comme dans l'image C. Bien sûr, une description au niveau plus élevé que celui de bascules est plausible: par exemple, l'ensemble du circuit de l'image A peut être défini comme un module assurant la fonction d'un compteur de chiffres, qui compte les impulsions envoyés à sa ligne d'entrée et affiche le nombre compté dans l'affichage marqué DCD_HEX. En tant que module, ce circuit peut être utilisé comme un élément standard dans d'autres, plus grands circuits. (Images A, B et C prises à partir de Wikipedia Commons, respectivement à les adresses http://commons.wikimedia.org/wiki/File:4_bit_counter.svg, http://commons.wikimedia.org/wiki/File:JK-FlipFlop_(4-NAND).PNG et http://commons.wikimedia.org/wiki/File:TTL_npn_nand.svg).

entre la complexité de la tâche et la fiabilité de la description modulaire produite, et pour cette raison l'identification de descriptions hiérarchiques approximatives mais acceptables, est algorithmiquement possible uniquement pour les systèmes de taille limitée.

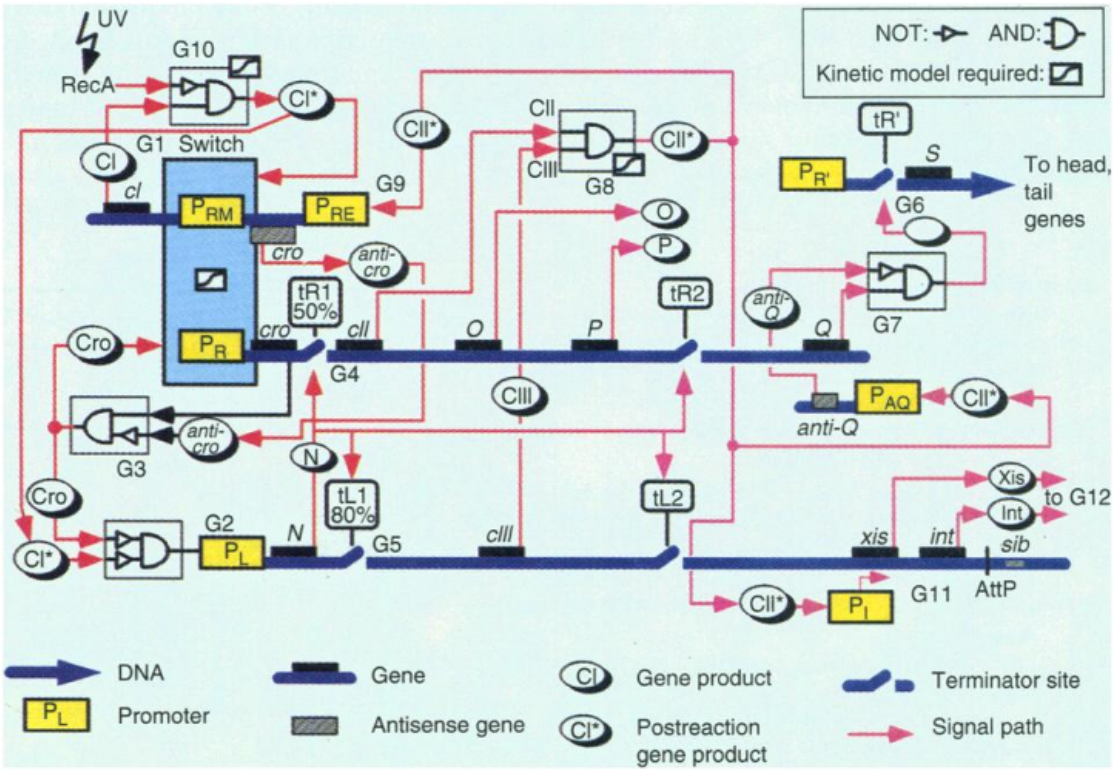

Figure 18.4: représentation schématique du circuit génétique générant la dynamique du cycle lysogénique dans le phage λ. (Le phage est un virus qui affecte les bactéries. Image prise à partir de McAdams & Shapiro 1995, p. 652).

18.2.2 Détection de la modularité des systèmes dynamiques discrets et des systèmes computationnels

En accord avec les considérations sur la modularité dynamique esquissées ci-dessus, je passe à envisager des formes de modularité dynamique dans certains types de systèmes dynamiques discrets computationnels. La possibilité de *détection* de modularité dynamique et computationnelle dans ces types de systèmes, qui peuvent souvent être considérés comme des systèmes informatiques au niveau de la machine de Turing, se révèle être atteinte par l'indécidabilité algorithmique ou, au moins, par un complexité computationnelle importante. Une première discussion préliminaire sur cette question va être menée dans ce qui suit.

18.2.3 Modularité et sciences biologiques: quelque exemple

Compte tenu du fait que la modularité fonctionnelle et la modularité structurelle, même si conceptuellement distinctes, sont souvent liées, les méthodes de détection automatique de la modularité dans les réseaux, qui appliquent à la *structure* du réseau, pourraient, lorsque appliquées à la représentation sous forme de réseau d'un système biologique, donner une description modulaire immédiatement fonctionnelle. La coïncidence fréquente entre l'organisation structurelle et l'organisation fonctionnelle dans les systèmes biologiques est confirmée par de nombreux ou-

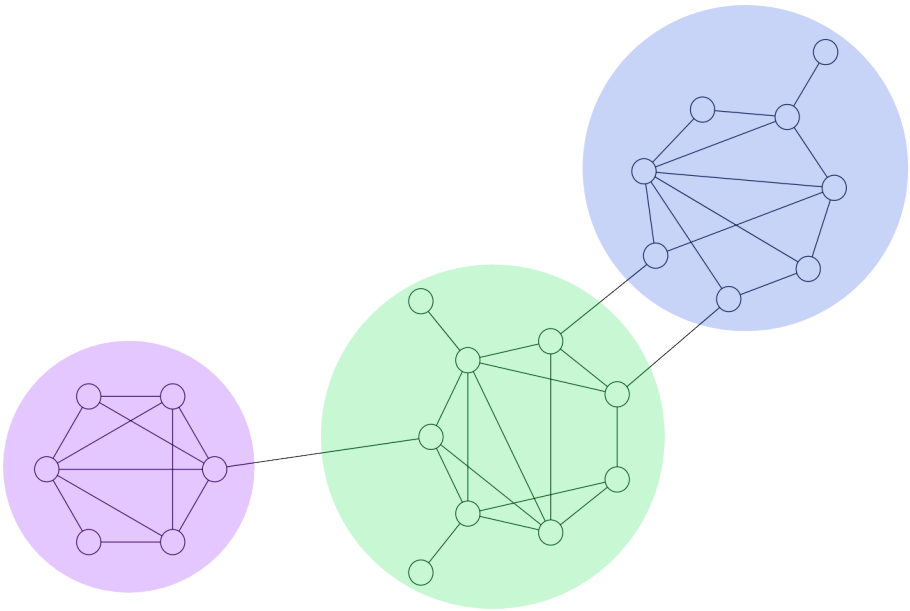

Figure 18.5: un réseau avec une structure de communauté. Dans cette image, disques colorés entourent les communautés, qui montrent haute densité de liens intra-modules, tandis les liens externes inter-modules sont plus rares.

vrages, notamment, entre autres, par une série de recherches par Zhou et Lipowsky[26], dans lequelles l'une des meilleures méthodes pour la détection de la modularité dans les réseaux est appliquée au réseau des interactions protéine-protéine de la levure. Cette application produit une description modulaire comprenant 449 modules, qui se révèlent correspondre à des sous-systèmes fonctionnels déjà bien connus, qui sont des composants d'une description modulaire de niveau encore plus élevé. Un autre travail important qui souligne une coïncidence entre modularité structurelle et fonctionnelle dans les systèmes biologiques est Guimerà & Amaral (2005b), qui applique aux réseaux métaboliques un algorithme de détection de la modularité qui identifie les modules, et après il leur attribue un rôle fonctionnel présumé sur la base de la connectivité structurelle intra et inter-module. Les modules fonctionnels identifiés ont des rôles qui se révèlent être corrélés très bien avec les fonctions biologiques réelles que les métabolites correspondants à chaque module effectivement accomplissent dans l'ensemble du réseau métabolique. Voir fig. 18.6.

18.3 Difficulté calculatoire

La difficulté calculatoire est une limitation pragmatique de certaines tâches computationnelles, qui consiste essentiellement dans le fait qu'elles ne peuvent pas éventuellement être menées à leur terme si la taille de leurs données d'entrée dépasse certaines limites[27]. Cela signifie que, en général, la tâche de calcul en question, tandis que exécutable *en principe*, ne pourrait pas

[26] Voir zhou:2004network et zhou:2006yeast.

[27] Cela est la *complexité temporelle* du programme, qui n'est pas le seul type de complexité calculatoire. Autres types de complexité et un meilleur traitement de tous ces sujets peuvent être trouvés dans la section 17.4 de l'Appendice.

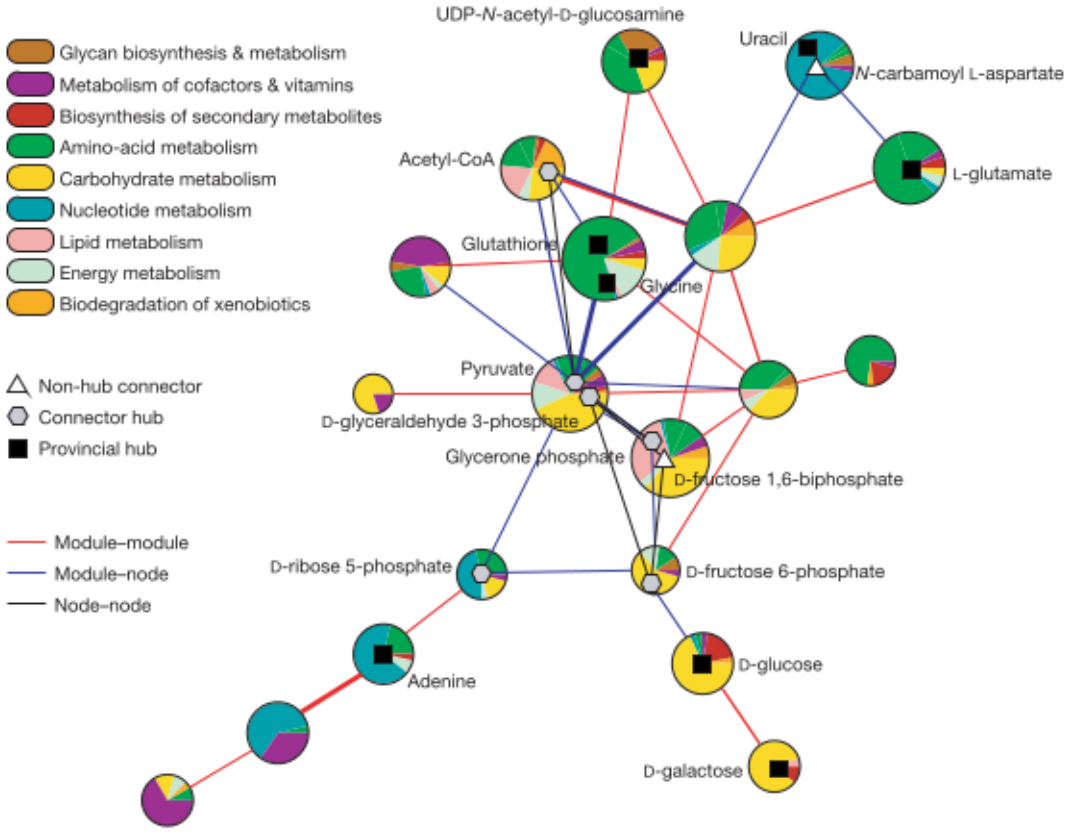

Figure 18.6: représentation modulaire de haut niveau d'un réseau métabolique. Image prise à partir de Guimerà & Amaral 2005b).

être amenée à une fin en temps humaines, ou même en temps astronomiquement possibles, si la taille de l'input dépasse une certaine ampleur. Cette situation est typique, par exemple, pour des problèmes qui ont des temps d'exécution proportionnels à une fonction exponentielle de leur taille: même pour les petites tailles de leur inputs, les temps d'achèvement du programme pourrait atteindre des valeurs irréalisables, car les fonctions exponentielles augmentent leur valeur très vite. Donc, même si la dureté de calcul est seulement une limitation *en principe*, elle est certainement une limitation insurmontable d'un point de vue pragmatique. Les classes les plus typiques de complexité de calcul qui peuvent être considérées comme difficiles, sont les classes des problèmes algorithmiques soi-disants "NP-complets" ("NP-complete") et "NP-difficiles" ("NP-hard") .

Certaines tâches algorithmiques qui ne doivent pas être considérées formellement difficiles, peuvent néanmoins être trop exigeantes computationnellement, pour être d'application pratique. Cela se produit par exemple quand une tâche nécessite d'un certain nombre de pas qui est proportionnel à une certaine puissance entière de la taille de l'input: par exemple, n^4, où n est la taille de l'input. Dans ces cas, étant donné un input suffisamment grand, le programme prendrait certainement trop long à terminer pour être d'une quelconque utilité pratique.

La difficulté calculatoire et le haut coût computationnel pratique sont deux notions sur lesquelles

ma proposition sur l'antimodularité se base, comme nous le verrons dans la prochaine section.

18.4 Antimodularité

Compte tenu de tous les résultats ci-dessus à propos de la difficulté calculatoire de la détection algorithmique de la modularité, je propose de définir la propriété de l'*antimodularitè* en général, comme l'impossibilité d'obtenir, *au moyen d'une détection algorithmique de la modularité*, une description modulaire hiérarchique utile et valide d'un système. Plus précisément, un système montre antimodularité lorsque sa description hiérarchique la plus réaliste et fidèle, donnée par des moyens algorithmiques, est quand même trop approximative pour être une description utile de haut niveau du système, ou lorsque elle est même non valide du tout. Dans ces cas, le seul possible description hiérarchique ne comprend que le deux niveaux hiérarchiques banals: le niveau de l'ensemble du système et le niveau de ses parties élémentaires du niveau plus bas: en d'autres termes, les systèmes *antimodulaires* sont des systèmes qui, intuitivement, peut être expliqués par décomposition à *un* seul niveau, le niveau de leurs pièces les plus fines et élémentaires.

L'antimodularité est due à l'échec de l'application de méthodes algorithmiques pour la détection de la modularité, et cet échec, à son tour, peut être éventuellement imputé à deux conditions:

1. Pas de modularité de niveau intermédiaire peut être raisonnablement supposée dans le système, étant donnée sa description préférée. Autrement dit, le système ainsi décrit en fait n'est pas modulaire du tout. Je nomme ce cas *antimodularité intrinsèque*, ce qui signifie que l'antimodularité est intrinsèque à la description donnée préférée, quoi que soit le degré de précision de l'algorithme pour sa détection. Cette situation peut se produire lorsque les parties du système, selon la description préférée, sont hyperconnectées: par exemple, dans un réseau régulier chaque nœud est connecté à *tous* les autres, et donc pas de modularité peut jamais apparaître.

2. Indépendamment du fait qu'une structure modulaire réelle est présente dans la description préférée du système ou non (comme au point 1), l'antimodularité se produit parce que, étant donné le *nombre élevé de pièces* composants la description préférée du système, l'algorithme pour la détection de modularité finit pour être computationellement trop coûteux pour être porté à son accomplissement, soit parce qu'il est computationnellement difficile (*hard*)[28], ou, bien que formellement pas difficile, parce qu'il est de toute façon computationnellement trop coûteux pour être terminé dans un temps raisonnable. J'appelle cette dernière raison tout simplement *antimodularité* (bien sûr, l'antimodularité intrinsèque est un cas de antimodularité).

La raison derrière cette distinction entre antimodularité intrinsèque et antimodularité tout-court, est que, tandis que l'antimodularité pourrait dans certains cas être éliminée par l'amélioration de l'algorithme de détection de la modularité, l'antimodularité intrinsèque persisterait encore, en tout cas, parce que elle n'est pas due à l'imprécision ou au coût calculatoire de l'algorithme utilisé, mais à une caractéristique objective du système, où, en raison de la distribution uniforme à travers le système de l'intensité de la relation qui concerne ses pièces, la modularité, par rapport à cette relation choisie entre les pièces, est, en fait, objectivement absente.

Il pourrait être utile, une fois en présence d'antimodularité, d'avoir une méthode pour déterminer si elle est antimodularité intrinsèque, ou si une forme de modularité est présente, mais

[28] Voir la section précédente.

elle ne peut pas être détectée. En raison de la distribution statistique de la relation entre les parties, l'antimodularité intrinsèque, au moins l'antimodularité intrinsèque structurelle, devrait être relativement facile à détecter, car l'absence intrinsèque de modularité peut être révélée par des enquêtes statistiques sur la répartition de certaines propriétés à travers un système. Donc, il devrait être assez facile de distinguer l'antimodularité et l'antimodularité intrinsèque, au moins dans certains cas. Il y a, cependant, des exceptions qui sont discutées dans la section 13.2.

À la lumière de ce que nous avons vu jusqu'à maintenant, il semble que la détection de la modularité peut, en systèmes assez grands, être effectivement empêchée par des problèmes de coût de calcul, ou même par l'impossibilité formelle de terminer la computation en un temps raisonnable, de sorte qu'un système peut être *pragmatiquement* considéré antimodulaire, même si en principe il possède une certaine modularité, qui, cependant, nous sommes pratiquement incapables de détecter automatiquement. Une description préférée antimodulaire d'un système ne possède, au moins pour autant que nous pouvons le savoir, aucune description modulaire valide de haut niveau, c'est-à-dire une description dont les parties sont dotées d'un degré suffisant de robustesse.

L'aspect pragmatique de l'antimodularité, de toute façon, ne doit pas être minimisé comme simplement pragmatique: il est une impossibilité pragmatique de mener à terme dans un temps réalisable un programme d'ordinateur, mais, surtout lorsque la complexité calculatoire d'un algorithme a été mathématiquement prouvée, cette entrave pragmatique devient quelque chose de plus puissant, avec la force d'une loi logique: il ne peut y avoir aucun espoir de rendre l'algorithme d'optimisation de la modularité détectée plus docile, lorsque il est prouvé être computationellement complexe. Peu importe comment nous essayons d'améliorer un algorithme computationnellement complexe, ou d'améliorer la puissance du système sur lequel il fonctionne: son temps d'exécution défera, au moins dans certains cas, toute amélioration possible de la vitesse. L'optimisation de la détection de modularité peut être probablement *approchée* en des temps plus raisonnables, mais le compromis entre vitesse et précision, qui est typique des algorithmes approchés pour la détection de la modularité, associé avec le nombre élevé de parties de certains systèmes complexes, pourrait rendre la modularité détectée trop approximative ou, au contraire, rendre trop élevé le temps de détection d'une description modulaire suffisamment précise, même si l'algorithme approchée n'est pas proprement difficile d'un point de vue calcolatoire formel.

Donc, l'antimodularité, au moins pour ce qui concerne la recherche de la meilleure description modulaire, une recherche qui a été révélée être une tâche NP-complet, est un fait pragmatique, mais dans le même temps elle est une propriété *objective* inévitable d'un système, découlant de propriétés absolues de la computation qui ne dépendent pas de contraintes incidentales ou d'un choix fait par l'observateur.

18.4.1 Émergence antimodulaire

Je propose d'appeler l'apparition de l'antimodularité dans un système, un cas d'*émergence antimodulaire*, et de la considerer comme une forme d'*émergence computationnelle*. L'antimodularité apparaît en effet tout à fait semblable à une forme bien connue de l'émergence: l'*émergence faible* (*weak emergence*), une notion proposée par Mark Bedau depuis le milieu des années 90. Cela est une notion d'émergence diachronique liée à certaines propriétés des systèmes computationnelles. Je compare spécifiquement ma notion proposée d'émergence antimodulaire à celle de l'émergence faible, et je conclus que l'antimodularité implique, avec quelques reservations, l'émergence faible, mais que l'implication réciproque ne vaut pas pour tous les systèmes: il y a des systèmes *modulaires* qui sont, en même temps, faiblement émergents. Je décris ici la ligne de

base du raisonnement sur la relation entre l'antimodularité et l'émergence faible, une discussion qui sera élargie dans la section 13.3.

Mark Bedau (1997) introduit la notion d'*émergence faible* (*WE* désormais), qui, dans sa formulation originale, s'applique principalement aux systèmes dynamiques discrets:

> Le macroétat P de S avec microdynamique D est faiblement émergent ssi P peut être dérivé de D et des conditions externes de S, mais seulement par simulation.[29]

Sans nous attarder ici à une explication des termes employés dans la définition ci-dessus, il suffit ici de dire que à mon avis la définition de Bedau pourrait être reformulée en toute sécurité comme:

> Un macroétat est faiblement émergent ssi il peut être dérivé à partir de la description préférée, du bas niveau, du système et de son état initial, mais *uniquement* par *microsimulation*, c'est-à-dire la simulation de la dynamique du système pas à pas à son plus bas niveau de description (le niveau de sa description préférée).

Il semble que, dans la plupart des conditions, (qui sont mieux précisées ci-dessous et dans la section 13.3), l'antimodularité implique l'émergence faible de Mark Bedau. L'argument est, brièvement, ceci: si un système est antimodulaire, alors, par définition, cela signifie que sa *seule* description modulaire valide est son description préférée, c'est-à-dire sa description au plus bas niveau. Ceci implique que le système ne soit pas *prévisible* au moyen d'une simulation *modulaire* de haut niveau: parce que, si cela était, cela voudrait dire que la simulation de haut niveau, en ce qu'elle est capable de prédire le système, représente une description modulaire de haut niveau *valide*. Mais, dans un système antimodulaire, cette description modulaire valide de haut niveau du système doit être exclue, par la définition de antimodularité. Donc, on peut conclure que la dynamique d'un système antimodulaire n'est pas susceptible d'être prévue par aucune simulation modulaire de haut niveau: si aucune autre méthode de prédiction *non-modulaire* est applicable, alors la seule façon de savoir comment le comportement du système va évoluer est par simulation du système au niveau de sa description préférée, c'est-à-dire par micro-simulation. Cette dernière circonstance semble équivalente à la reformulation que j'ai faite ci-dessus de la définition d'émergence faible de Bedau. Donc, il semble que *antimodularité* $\rightarrow$ *émergence faible*. Cette implication n'est pas absolument sûre, cependant, car elle dépend de la circonstance qu'un système antimodulaire donné, qui par définition d'antimodularité ne peux pas être prévisible par aucune simulation *modulaire* de haut niveau, soit également impossible de prédire par tout autre moyen non modulaire. Dans la section 13.3 je montre comment certains systèmes antimodulaire pourrait en effet être prédits par moyens de haut niveau non modulaires, et ainsi résulter antimodulaires mais pas faiblement émergents. Toutefois, je vais soutenir que ces systèmes ne sont probablement pas très intéressants dans leur comportement, et que dans les systèmes complexes les plus intéressants, comme ceux capable de computation, l'antimodularité implique l'émergence faible.

La chose intéressante est que l'implication opposée ne tient pas: il existe des systèmes faiblement émergents qui en même temps ne sont pas antimodulaire, c'est-à-dire des systèmes qui ont des descriptions modulaires de haut niveau valides. Le système reste faiblement émergent même en présence de ces descriptions modulaires, parce que telles descriptions de haut niveau ne peuvent pas être utilisées pour *prédire* le système (une prédiction qui, si possible, rendrait le système *non*

[29] Bedau (1997), p.378.

faiblement émergent, par définition), quoique elles puissent être utilisé pour *expliquer* le système. Cette situation peut se produire pour deux raisons possibles. D'abord, pour la raison que telles descriptions modulaires sont trop vagues, trop abstraites, trop de haut niveau pour être utilisées pour calculer une simulation dynamique du système: par exemple, des organigrammes qui tout simplement décrivent sommairement les rôles fonctionnels que les modules remplissent dans le système, sans fournir assez de détails pour permettre leur mise en œuvre, leur *implémentation*. Ces descriptions de haut niveau modulaires ne peuvent pas simuler dynamiquement le système, de sorte qu'elles ne peuvent pas anticiper de aucune manière ses résultats dynamiques, mais elles peuvent être utilisés pour *expliquer* le système, véhiculant une bonne explication. Dans un autre cas, la raison pour laquelle le système est faiblement émergent malgré ses possibles redescriptions modulaires de haut niveau, est que le système, même si fonctionnellement modulaire à un niveau élevé, est intrinsèquement imprévisibles, et cela est la raison de son être faiblement émergent. Cette situation peut se produire dans les systèmes computationnels universels, qui, comme une conséquence de l'indécidabilité du problème de l'arrêt (la propriété bien connue prouvée par Alan Turing avec sa proposition de systèmes informatiques en 1936[30]), possèdent de nombreuses propriétés dynamiques qui sont intrinsèquement imprévisibles. Les ordinateurs universels du monde réel sont généralement des systèmes de ce genre: ils sont très modulaire, mais potentiellement imprévisibles. Donc, ils sont modulaires et en même temps faiblement émergents.

18.4.2 Antimodularité et modèles de l'explication

Après avoir défini ce nouveau type d'émergence computationnelle, qui est dû à la difficulté calculatoire qui peut se manifester en certains cas dans des algorithmes pour la détection de modularité, je tente de tirer quelques conséquences possibles de l'émergence antimodulaire sur la possibilité d'*expliquer scientifiquement* les systèmes qui en souffrent. Je examine deux modèles bien connus de l'explication scientifique: l'explication fonctionnelle-mécaniste et celle déductive-nomologique (*DN*, désormais). Je évalue ensuite un modèle plus débattu d'explication, l'*explication computationnelle*, et un autre type d'explication qui a été l'objet d'une analyse récente par Philippe Huneman, l'explication *mathématique-topologique*, une forme d'explication adéquate pour expliquer certaines caractéristiques des systèmes dynamiques complexes. Je conclus que l'émergence antimodulaire affecte la faisabilité des deux premiers types d'explication, ainsi que des explications computationnelles, quoique différemment, et laisse intacte la possibilité d'explication topologique, constituant l'émergence antimodulaire elle-même une opportunité pour ce genre d'explication.

18.4.3 Antimodularité et explications fonctionnelles ou mécanistes

Je soutiens que l'antimodularité affecte négativement l'explication *mécanistique*, une forme fondamentale d'explication dans les sciences biologiques. Un bref détour est nécessaire ici pour décrire ce que cette forme d'explication représente.

Le terme *explication mécanistique* se réfère généralement dans la philosophie de la science de nos jours à un modèle relativement récent de l'explication scientifique, proposé depuis les années '90 par plusieurs groupes de philosophes de la biologie et des sciences cognitives plutôt indépendamment. Les représentants les plus éminents des deux lignes principales d'enquête dans ce domaine sont William Bechtel et ses collaborateurs d'une part, et Carl Craver et ses collègues de l'autre[31]. Laissant pour le moment de côté les différences subtiles entre ces deux principales

[30] Comme expliqué dans la section 17.2.6 de l'Appendice.
[31] Les deux œuvres séminales correspondantes sont Bechtel & Richardson (1993) et Machamer et al. (2000).

conceptions de l'explication mécaniste[32], je me base ici sur Bechtel & Abrahamsen (2005), qui est un texte standard dans le sujet. Bechtel et Abrahamsen (*BA* désormais) donnent une définition du *mécanisme* comme ceci:

> Un *mécanisme* est une structure remplissante une fonction[33] en vertu de ses parties composantes, de ses operations composantes, et de leur organisation. Le fonctionnement orchestrée du mécanisme est responsable d'un ou de plusieurs phénomènes[34].

La définition ci-dessus définit un mécanisme comme quelque chose de semblable à ce que j'appelle un système complexe, c'est-à-dire un système composé de parties en interaction. Le point à souligner ici est qu'il y a une vue fonctionnelle impliquée dans l'explication mechaniste: la fonction globale, qui représente l'explanandum, est expliquée en décrivant l'organisation et les interactions des parties qui, au moyen de leur dynamique "orchestrée", produisent le phénomène. Ce qui est nécessaire, selon BA, pour expliquer un phénomène donné, est alors d'abord d'identifier les pièces et les opérations impliquées dans sa production. Dans ce but, le système dans son ensemble doit être soumis à deux opérations, qui BA appellent *décomposition structurelle* et *décomposition fonctionnelle*: la première identifie l'ensemble de pièces élémentaires du système, tandis que la seconde, qui, dans le monde de la science réelle est souvent menée séparément de la première, identifie les opérations composantes. Une troisième opération souhaitable est la *localisation*, qui consiste à relier les pièces avec les opérations qu'elles effectuent. De cette façon, une explication mécaniste est donnée, selon BA. Ce bas niveau d'explication n'est pas toujours le plus souhaitable, et, comme BA soulignent, il est important qu'une hiérarchie de mécanismes soit considérée, et que l'explication soit *multiniveaux*. Selon BA, un mécanisme peut aussi impliquer de *niveaux multiples d'organisation*, étant le mechanisme observé souvent partie d'un mécanisme plus large de niveau supérieur: les circonstances extérieures à un mécanisme donné peuvent être vu comme des mécanismes généraux globals, tandis que les composantes d'un mécanisme peut être considérées comme mécanismes elles-mêmes, elles-mêmes composées, de manière récursive, des sous-parties.

Il me semble que toute cette conception de mécanismes pourrait facilement être reformulée en termes de modularité, le long des lignes de la position que je viens d'esquisser jusqu'à maintenant. Le résultat de la décomposition fonctionnelle et structurelle et de la localisation est ce que j'ai appelée la *description préférée* du système: l'identification des éléments de base, au plus bas niveau, que l'observateur a choisi d'identifier. BA ne stressent pas, comme je le fais, la dépendance de cette description d'un choix de la part de l'observateur, parce qu'ils considèrent implicitement qu'il existe des descriptions préférées naturelles de certains systèmes, et il y en a sans aucun doute, par exemple en biologie moléculaire, où les molécules (ou, éventuellement, les atomes) sont le parties élémentaires les plus naturelles. La principale différence avec mon point de vue est alors que ma conception de la modularité hiérarchique est plus générale et inclusive de formes de modularité non-physiques et exclusivement fonctionnelles, comme celles concernantes la computation.

show how *antimodularity* compels to *single-level-only* explanations, neglecting the need, essential for mechanistic explanations, of *multi-level* integration. Antimodularity would limit mechanistic explanation to the level of description representing the most elementary parts of the systems,

[32] Ces différences, surtout les plus importantes, entre la soi-disante vue *épistémique*, soutenue par William Bechtel (et que je soutiens aussi), et la vue *ontique* de mécanismes, soutenue par Carl Craver, sont discutées au chapitre 10.

[33] Voir la section 9.

[34] Bechtel & Abrahamsen (2005), p. 423.

which is the most numerous and the most complicated, and this fact certainly hinders comprehension as well: for large enough systems, their mechanistic explanation at this level is too complex to be understood by human beings, and understandability is a quality to be sought for in mechanistic explanation, according to some accounts, notably the ones by William bechtel and his collaborators. Others, too, deem intelligibility an essential feature of explanations, for example Petri Ylikoski, which considers "cognitive salience" an important feature of explanations.

Cela dit, il est facile de montrer comment l*antimodularité* oblige à avoir recours à *un seul* niveau d'explication, et à négliger la nécessité, indispensable pour les explications mécanistes, de l'intégration *multi-niveau*. L'antimodularité limiterait l'explication mécaniste au niveau de description représentant les parties les plus élémentaires du système, qui est description la plus nombreuse et la plus compliquée, et ce fait entrave certainement la compréhension: pour systèmes suffisamment grands, leur explication mécanistique à ce niveau est trop complexe pour être comprise par les êtres humains, et la compréhensibilité est une qualité à rechercher dans explication mécanistique, selon certains témoignages, notamment ceux de William Bechtel et ses collaborateurs. D'autres auteurs, aussi, jugent l'intelligibilité une caractéristique essentielle de l'explication, par exemple Petri Ylikoski, qui considère la "saillance cognitive" une caractéristique importante des explications.

Il est évident qu'une explication mécanistique tente de répondre aux questions "comment" ("Comment un phénomène est provoqué?"), au moyen de la description de la façon dont le complexe fonctionnement dynamique d'un ensemble de pièces en interaction produit le phénomène. La même question peut être répondue aussi juste du point de vue *fonctionnel*, et cette conception, visante principalement à caractériser l'explication en psychologie cognitive, a été notoirement avancé par Robert Cummins. D'une manière similaire à celle de la décomposition mécanique, l'analyse fonctionnelle commence par une caractérisation du phénomène global (la *disposition*[35]) pris comme la fonction globale qui doit etre expliquée en termes de ses sous-fonctions composantes. Ceci est une forme typique de ce qu'on appelle *fonctionnalisme du rôle*, où la notion de fonction[36] est considérée comme celle d'un rôle partiel rempli par un sous-système afin de contribuire à la production de l'ensemble de fonctionnement du système global. D'un point de vue explicatif, la fonction d'un sous-système est employée à expliquer comment la fonction globale, qui est l'explanandum, est effectuée au moyen des contributions de ses sous-fonctions organisées, qui exécutent leur fonction dans une activité programmée. Cette position est assez proche d'une vue informatique, et il est complètement compatible avec elle. En fait, l'analyse fonctionnelle de Cummins est le prototype de l'explication typique de la psychologie cognitive, qui se compose principalement d'explications fonctionnelles, souvent sous la forme d'*explication computationnelle*, c'est-à-dire sous la forme de l'exposition d'un programme informatique capable de produire le phénomène cognitive à expliquer.

Une caractérisation plus approfondie de la position de Cummins est donnée dans le chapitre 9, où la relation entre l'explication purement fonctionnelle et l'explication mécaniste est également mieux analysée. Ce que je voudrais souligner ici est que Cummins lui-même, depuis ses premières œuvres (comme dans le séminal Cummins (1975)), souligne que la décomposition fonctionnelle récursive afin d'obtenir une hiérarchie complète, est la stratégie à chercher dans les explications scientifiques, en particulier dans les explications biologiques. L'antimodularité empêcherait complètement cet objectif, en permettant seulement une explication à deux niveaux: le plus élevé, celui de l'explanandum lui-même, et, sur l'autre extrémité de l'échelle, le niveau le plus bas, celui des fonctions les plus élémentaires.

[35] Je dédie une discussion à ce terme technique dans le chapitre 9.
[36] La notion de fonction est examinée dans le chapitre 9.

18.4.4 Antimodularité et modèle déductif-nomologique de l'explicaton

Dans le vue déductive-nomologique classique (*DN*) de l'explication, issue de l'ouvrage fondamental de Carl G. Hempel et Paul Oppenheim[37], l'explication est considérée comme une *déduction logique* de l'explanandum à partir de l'explanans, et ce qui compte est la validité et la correction de la déduction, avec peu d'attention dirigée vers l'*intelligibilité* de l'explication: une telle préoccupation à propos de la compréhensibilité de l'explication aurait été considérée, dans le milieu historique post-neopositiviste de cette époque là, une intrusion inappropriée de la philosophie des sciences dans le territoire des aspects pragmatiques ou pire, *psychologiques* de l'explication scientifique. Dans une telle perspective, tout ce qui compte pour une explication c'est qu'elle soit une déduction correcte. Dans ce modèle, on voit l'explication en tant que fonction de la possibilité de *prédiction* du phénomène au moyen d'une *loi* scientifique. L'explication consiste en une description de la dérivation logique de l'explanandum d'un groupe de prémisses constitué par une loi scientifique et un ensemble de clauses représentant les conditions initiales du système à expliquer.

Pour ce qui concerne les explications déductive-nomologiques, je montre que, puisque l'antimodularité implique l'émergence faible dans les systèmes antimodulaires suffisamment complexes, on ne peut pas recourir à l'explication DN pour un système antimodulaire complexe, parce que, si cela était possible, cela voudrait dire que le système est prévisible, et la prévisibilité est nié par la définition d'émergence faible soi-même. Pour clarifier: il est exclu par la définition d'un phénomène faiblement émergent qu'il peut être prédit par l'intermédiaire d'une loi qui, étant donné l'état initial, détermine l'état dans lequel le système va être à un moment donné, et que cette loi a une expression mathématique qui peut être résolu *analytiquement*. Comme on a dit, cela est exclu par la définition même de l'émergence faible, ce qui indique simplement qu'un phénomène faiblement émergent (dans un système dynamique discret) est celui qui ne peut pas être prédit, et qu'il ne peut être atteint que par l'exécution pas à pas de la microsimulation au plus bas niveau du système. Étant donné que ma notion de antimodularité, dans les circonstances exprimées dans la section 18.4.1, implique l'émergence faible, il se trouve qu'un système antimodulaire assez complexe ne peut être prédit par une expression susceptible d'une solution analytique. Donc, aucune explication DN d'un système antimodulaire complexe pourrait être fondée sur une telle loi analytiquement résoluble.

Cependant, si nous prenons en considération une classe spécifique de systèmes, à savoir les automates cellulaires (*CA*), alors un processus faiblement émergent généré par un CA peut dans un sens être expliqué par la production d'une liste, potentiellement très longue, des étapes de son évolution, une liste qui peut être considérée comme une liste d'étapes *déductives* à l'intérieur d'un système logique formel, dans lequel les prémisses sont constituées par la configuration initiale de l'AC et par la règle du CA, qui est appliquée de façon répétée, premier à la configuration initiale, et ensuite à chaque configuration intermédiaire obtenue à chaque étape déductive. Étant donné que toutes les règle-CA sont, par le théorème de Curtis-Hedlund-Lyndon[38], locales et tout aussi valides en tout point de la matrice du CA, la forme d'un règle-CA peut dans cet égard être assimilée à la forme d'une loi physique, qui, en tant qu'une *loi*, est valide universellement partout. En conséquence, par cette analogie, la production de cette liste d'états consécutifs du CA pourrait d'une façon être assimilée à une longue explication DN, qui doit se composer d'une déduction logique de l'explanandum à partir de prémisses constituees par les conditions initiales données et une loi. Même dans ce cas, la compréhension humaine serait entravée par la longueur potentielle de la liste, mais, selon la position théorique des défenseurs post-neopositivistes du

[37] Hempel & Oppenheim (1948).
[38] Voir la section 14.2.1.

modèle DN d'explication, la compréhension est un élément inessentiel des explications, et elle n'est pas nécessaire pour une bonne explication DN. Donc, en un sens, l'émergence faible et, par conséquent, l'antimodularité, n'entravent pas l'explication DN, au moins dans le cas des automates cellulaires et d'autres systèmes dont la dynamique universelle suit une règle *universelle*. De cette considérations sont exclus des classes différentes de sistèmes dynamiques discrets, par exemple les réseaux booléens en général, dont la dynamique peut suivre localement de règles changeantes localement qui ne sont pas universelles. Dans ces cas, la règle que doit être utilisée serait la règle de mise à jour globale, qui, étant non locale, est généralement beaucoup plus complexe qu'un règle-CA, et, en conséquence, la liste des déductions constituante l'explication selon le style déductif-nomologique de telles systèmes serait encore moin intelligible.

18.4.5 Antimodularité et explications topologiques

Je considère maintenant les conséquences de l'antimodularité sur la possibilité d'expliquer un système complexe par le biais de ce que Philippe Huneman appelle *explication topologique*. Huneman décrit l'explication topologique comme un type d'explication qui fait abstraction des relations causales et des interactions dans un système, dans le but de repérer quelque sorte de propriété "topologique" afin de tirer de cette propriété des conséquences mathématiques qui expliquent certaines caractéristiques du système[39]. Inspirées de la topologie mathématique, les propriétés topologiques d'un système sont les propriétés concernantes d'une manière sa "forme" qui sont invariantes par rapport à les possibles déformations continues du système. Ces propriétés structurelles ne doit pas appartenir à un système matériel, mais peuvent concerner un espace mathématique abstrait. Dans ma terminologie, je dirais que ces propriétés topologiques ne concernent pas un système, mais une description du système. L'explication topologique consiste à expliquer les caractéristiques du système en pointant certaines caractéristiques topologiques de la représentation du système dans cet espace abstrait, et pas au travers des événements causales entre ses parties, comme l'explication mécaniste ferait: l'explication est spécifiquement basée sur une propriété *mathématique*, topologique, qui fait tout le travail explicatif.

L'explication topologique pourrait aussi être fondée sur la présence d'une structure *modulaire*. Cette situation peut arriver par exemple si une explication topologique de la robustesse de la dynamique d'un réseau à des perturbations locales est donnée en mentionnant que le réseau a une *structure de communauté*[40]: cette structure modulaire assure que les perturbations restent locale ou canalisées, sans se propager sans discernement à la même vitesse sur l'ensemble du réseau. Au contraire, l'antimodularité intrinsèque pourrait produire une propagation illimitée des perturbations sur le réseau.

Tout bien considéré, non seulement il semble que l'émergence antimodulaire n'entrave pas l'explication topologique, mais il resulte que l'antimodularité intrinsèque ou son absence pourraient en effet *rendre possible* certaines explications topologiques.

18.4.6 Explication et prévision

La possibilité, évoquée plus haut dans la section 18.4.1, qu'il existe des systèmes qui sont fonctionnellement explicable et, dans le même temps, imprévisible, dont un exemple est la classe de systèmes computationnels universels, donne une indication tout à fait remarquable, à savoir que la prédiction et l'explication sont des entreprises disjointes: l'imprévisibilité ne rend pas, en soi, un système inexplicable. Ce résultat est curieux, car il épreuves, en quelque sorte, que la

[39] Voir Huneman (2010).
[40] Voir les sections 18.2.1 et 3.2.1.

prédiction n'est pas nécessaire pour l'explication, et donc que le modèle déductif-nomologique de l'explication, même s'il fût exempte d'autres inconvénients, ne pourrait pas être le modèle général de l'explication scientifique. En science certaines explications sont acceptables même si elles ne sont pas fondées sur la prévision: explications qui sont fonctionnels, ou mécaniste.

18.4.7 Computation et explication computationnelle

Avant d'évaluer, dans la section suivante, les conséquences possibles de l'antimodularité sur les explications computationnelles, une réflexion sur ce qu'est une explication computationnelle est nécessaire. Plus précisément, nous allons demander si et quand un système donné, le plus souvent un système dynamique, effectue une computation ou non, afin de voir si l'on peut lui donner une explication computationnelle. Pour cet objectif, il semble inévitable de discuter la notion de computation elle-même.

Quand est-ce qu'un certain système compute? Depuis les travail de Turing de 1936, qui a fondé la science informatique, il peut sembler à première vue que le concept de computation est tout à fait clair, et que cela est une question facile, décidable par des moyens formels. Le travail de Turing a en effet fourni une pierre de touche contre laquelle caractériser ce que la computation est. La computation a été traité, depuis sa création avec Turing, comme une question éminemment *formelle*, qui mérite une approche *mathématique*. Et cela a sûrement été, je pense, la bonne façon de traiter la question, car elle a mis en lumière les caractéristiques essentielles de la computation, sa puissance et ses limites, détaillées d'une manière approfondie et rigoureuse qu'aucun autre approche du problème de la computation aurait pu développer.

Néanmoins, la question "Quand est-ce qu'un certain système compute?", d'un autre point de vue, n'est *pas* une question facile. Pour voir cette difficulté nous pourrions recourir à imaginer quelqu'un qui lance un programme sur son ordinateur, et que à notre question "Qu'est-ce que ce programme a fait?" répondît qu'il *compute*, tout court. Aurait cette réponse du sens pour nous? Supposons, encore une fois, qu'un certains programmeur a écrit, pour s'amuser, un programme constitué par une liste aléatoire d'instructions, et que, par un coup de chance, le programme, au lieu de se planter, a marché, produisant des chaîne de caractères apparemment aléatoires sur l'écran. Considéreriez-vous ce programme comme accomplissant une computation? Bien sûr, dans une manière de parler, il compute... mais il compute *quoi*?!? Troisième cas: nous avons un programme en cours d'exécution sur un ordinateur très primitif, qui produit directement de chaînes binaires: à partir de la chaîne 00110101, il produit 1111, à partir de 01000011 il produit 1100. Est-il en train de computer?

Il peut être immédiatement suggéré que la reconnaissance de l'occurrence d'une computation a besoin d'une sorte d'attribution intentionnelle: c'est-à-dire, l'attribution de *computation*, à ce qui autrement pourrait être considérée comme une simple transformation physique de symboles: pour dire qu'un certain processus compute, nous avons besoin de *spécifier ce* qu'il calcule. Pour faire cela, une condition doit être remplie: une *correspondance* doit être établie entre les configurations physiques de la machine, sur lequelles le calcul présumé agit, et des signes significatifs. Seulement une fois que nous faisons cela, nous sommes dans la position d'au moins essayer de deviner *quelle* computation spécifique le système est en train d'exécuter. Cette correspondance est une *interprétation*, qui associe configurations d'entrée et de sortie à des signes significatifs de notre langue. Cette correspondance est elle-même une opération *algorithmique* , qui, à partir de certaines configurations en entrée, produit d'autres symboles.

Cela est quelque chose qui ne concerne pas le problème de l'intentionnalité vu comme un problème philosophique. Nous opérons une *attribution* intentionnelle sur l'ensemble des configurations résultantes de la correspondance choisie entre configurations de la machine et signes linguistiques,

configurations dont les propriétés formelles doivent avoir été choisies, par le biais de cette correspondence, comme capable de *signifier*, quelque chose pour nous. Une fois la correspondance (l'interprétation) est établie, nous pouvons opérer une attribution intentionnelle plus globale, et essayer de dire *quelle* computation le système dans son ensemble accomplit. À mon avis, seulement alors le système peut être considéré comme *computationnel*. La computation est *attribution* de computation, à des systèmes qui, *en soi*, sont tout simplement des systèmes dynamiques discrets régi par des règles[41].

Ce point de vue "intentionnel" ou "sémantique" de la computation ("intentionnel" ou "sémantique" avec tous les avertissements indiqués ci-dessus) n'est pas une nouvelle position, après avoir été adopté par plusieurs auteurs, notamment Jerry Fodor. Ce point de vue sémantique est contesté par certains auteurs, comme Gualtiero Piccinini, qui considèrent la computation comme définissable sur le plan purement mécaniste, sans la nécessité de recourir à toute attribution sémantique[42].

Un problème se pose ici: consideré que l'attribution de calcul dépend du choix d'une correspondance, est-ce que cela signifie que *n'importe quelle* machine, une fois fourni une correspondance appropriée entre des symboles significatifs et les etats de la machine, peut être considérée comme une machine qui effectue des *computations*? Il me semble que certaines conditions doivent être remplies. Premièrement, la machine doit être numérique: nous parlons de computation *numérique* ici. Donc, pour être considérée comme computationnelle, une machine doit au moins être considérée comme numérique, c'et-à-dire qu'elle doit posséder un ensemble fini d'états possibles distinctes et stables sur lesquels elle ira operer. Ce sont ces états qui vont être mis en correspondance avec les symboles. Mais, pour que cela soit possible, ces états doivent être robustes et tels que l'on peut les distinguer, et, en outre, une règle déterministe doivent régir la transition entre configurations stables de ces éléments. Il n'est pas garanti que tout système puisse remplir ces conditions, qui ne sont pas banales (pensez à des configurations distinctes stables dans un fluide turbulent). Mais, à l'intérieur de la vue de la modularité que je propose ici, il est facile de voir que tout ce qui est nécessaire ici est que le système soit un système discret dynamique (ou *DDS*), et un DDS constitue une forme de système fonctionnel modulaire (voir section 18.1.6).

Le point central est que pour nous permettre d'attribuer à un système l'exécution d'une computation, les conditions suivantes doivent être remplies:

a. une correspondance algorithmique doit être réalisée entre des symboles linguistiques et des configurations possibles d'entrée et de sortie du système;

b. nous devons être en mesure de dire *quelle* est la fonction input/output particulière, c'est-à-dire quelle est la *spécification* de la computation, afin de dire ce que le système compute.

Seulement dans le cas où le système est un DDS et ce deux autres conditions sont remplies, le système peut être considéré comme *computationnel*. Seulement alors nous serons en position de dire que le système effectue une computation.

L'ordre des conditions *a* et *b* ci-dessus est inversée par rapport à la façon dont un programmeur humain travaille. Dans ce cas, ce qui est nécessaire est pas une interprétation, mais l'établissement

[41] Bien sûr, il peut être soulevé un problème ici: si le système est considéré comme déjà régi par des règles, ça signifie qu'une attribution intentionnelle originale a déjà été faite. Il est hors de la portée de ce travail d'aborder ici cette question et d'autres questions épineuses semblables, analogues au problème "kripkensteinian" de suivre une règle.

[42] Voir par exemple Piccinini (2008).

d'une norme: dans le cas de la programmation, d'abord la spécification (point *b*) est arbitrairement *choisi*. Elle va être considérée comme la norme selon laquelle le programme devra opérer, et *sur la base de la spécification*, le programmeur choisit la correspondance (point *a*) qu'il juge la meilleure entre symboles et configurations d'entrée et de sortie, afin de procéder à la *implémentation* (la mise en œuvre) du programme, c'est-à-dire à la spécification des pièces et de la structure du mécanisme (le programme) qui sera, à la fin, en mesure de réaliser la spécification choisie selon la correspondance choisie. Ainsi, le choix de la mise en correspondance détermine le choix de la *spécifique* structure du *programme*. Tout cette série d'opérations constitue l'*implémentation* de la spécification choisie.

Parlant de l'*implémentation* le long des lignes de Galton (1993) et Partridge & Galton (1995), je considère la relation spécification-implémentation comme une relation universelle: une *implémentation* est la description d'une méthode de "réaliser"[43] une spécification d'ensemble. Lors de l'examen d'un programme, il n'y a pas, cependant, une spécification globale *unique* et un seul niveau d'implémentation , pour la raison que les deux notions sont *relatives*, exactement comme celles des niveaux de description "supérieur" et "inférieur", et celle de *fonction*, qui[44] est le rôle partiel que quelque chose remplit par rapport au cadre d'une fonction globale. *Relative*, dans ce cas, signifie que quelque chose qui est l'implémentation d'une spécification, peut à son tour être considérée comme une spécification de niveau inférieur qui doit être mise en œuvre à un niveau encore plus bas. En d'autres termes, compte tenu de la spécification, il est nécessaire de trouver un mode de réalisation possible de cette spécification, et dans le style de programmation structurée ou modulaire, une telle mise en oeuvre sera elle-même susceptible d'etre décomposée en modules. Chaque module, étant une fonction spécifique d'entrée-sortie[45] constitue lui-même, à son tour, une spécification, qui sera mise en œuvre à un niveau inférieur, et ainsi de suite.

Il semble raisonnable de penser que le même *hiérarchie* abstraite multi-niveau dans laquel chaque macro-élément est multréalisable par des sous-composantes, et ainsi de suite, sous-tend les notions de programmation structurée, décomposition fonctionnelle, et les niveaux modulaires hiérarchiques de description.

Comme on a dit, un programmeur commence avec une spécification et essaie de la mettre en œuvre. Mais il est possible de commencer avec un processus discrèt *non interprété*, et essayer de *découvrir* quelle computation, et comment, le processus effectue. Ceci est un chemin de bas en haut, et il est typique de la *rétro-ingénierisation* d'une computation. Le chemin inverse, de haut en bas, consistant à commencer avec la spécification d'une computation et à essayer de décomposer de façon récursive la spécification, qui est une fonction, en sous-fonctions, et ceux-ci à leur tour en sous-fonctions plus simples, et ainsi de suite, afin de dire *comment* la spécification (la fonctio globale) est realisée, est plutôt le chemin de l'*explication computationnelle*, qui est l'explication fonctionnelle typique de la psychologie cognitive, où la spécification, qui est la faculté cognitive (l'équivalence cognition/calcul est le principe de base de la psychologie cognitive), est expliquée en termes d'une représentation fonctionnelle hiérarchique.

18.4.8 Antimodularité, automates cellulaires et explications computationnelles

Maintenant, nous pouvons considérer le problème des conséquences de l'antimodularité sur l'*explication computationnelle*. Comme on a dit, l'explication computationnelle est typiquement

[43] Dans un sens étroitement apparenté à celui de la propriété de *réalisation* en philosophie de l'esprit. Je ne vais pas examiner la notion ici.

[44] Voir la section 18.4.3.

[45] "Fonction" dans un sens mathématique, voir section 9.

le genre d'explication utilisé dans les sciences cognitives. Cependant, je ne vais pas évaluer ici quel impact pourrait l'antimodularité avoir sur ces sciences, mais seulement la façon dont elle pourrait avoir un impact sur un "modèle-jouet": je prends ici un automate cellulaire (un *CA*) comme un modèle de système computationnellement capable, afin de voir comment son comportement dynamique pourrait être *expliqué* comme quelque chose qui effectue une computation. L'idée est que, si le CA est antimodulaire dans son comportement dynamique, cette tâche explicative pourrait être entravée, voire rendue impossible. Si tel est le cas, cela signifie que l'antimodularité pourrait avoir un impact sur la possibilité d'explication computationnelle.

Prenons donc le cas d'essayer d'expliquer computationnellement un CA. Deux questions fondamentales peuvent être posées ici:

- Est un CA un système computationnel?
- S'il est un système computationnel, comment pouvons-nous expliquer la computation qu'il effectue?

En ce qui concerne le premier point, bien sûr un CA, étant un système dynamique discret, respecte les conditions énoncées dans la section précédente de ce chapitre, et peut donc être considéré sans doute comme un système capable de calcul. Mais pour pouvoir dire qu'il *compute*, nous devons être en mesure de dire, d'une certaine façon, *ce* qu'il calcule: aucune explication de calcul peut être appliquée avant que la correspondance entre ses configurations et des symboles significatifs pour nous est établie. Nous pourrions essayer de faire correspondre l'état de chaque cellule du CA à un symbole sensible. Par exemple, nous pourrions faire correspondre ces états à «noir» et «blanc», mais cela conduirait à des explications "computationnelles" du genre suivant: "selon l'application répétée de sa règle, le CA produit une variation progressive de l'état de ses cellules, qui peuvent, sous diverses conditions, changer du blanc au noir". Cela ne semble pas une explication très utile. Une explication complète nécessite d'une spécification claire, à savoir la capacité de dire ce que le système calcule. Ici, la spécification est trop vague: "variation progressive de l'état de ses cellules dans diverses conditions". La spécification exacte du CA, vue comme une fonction d'entrée/sortie, est donnée par l'application répétée de sa règle, donc une spécification raisonnable pourrait être donnée en termes de la règle du CA. Mais il y a des problèmes qui se posent ici: généralement, la plupart des règles-CA sont décrites en termes d'une "table de correspondance", qui est une liste extensive décrivante la façon dont la règle détermine la valeur d'une cellule à la prochaine étape sur la base des valeurs de les cellules voisines. Une telle table devient incontrôlablement grand à mesure que le voisinage d'un CA élargit. Donc, pour certains CA, l'exposition de cette table serait impossible, ou de toute façon rendrait complètement dénuée de sens une explication de calcul qui la cite. Dans les CA à deux valeurs, la règle peut être considérée comme indiquante une expression booléenne. Donc, nous pourrions penser de simplifier sa description sous la forme d'une expression booléenne. Ceci, cependant, est très certainement une tâche computationellement complexe, donc, il ne peut pas être garanti de réussir dans un temps raisonnable dans tous les cas.

Tout ce qui précède nous suggère que nous devrions essayer de trouver des explications computationnelles de *niveau supérieur*, et, encore mieux, *multi-niveaux*, afin d'obtenir une explication plus utile. En d'autres termes, nous devons trouver un moyen de reconnaître un CA comme une machine computaionellement capable à un niveau qui est plus élevé que celui de ses cellules élémentaires. Pour obtenir cela, nous pourrions essayer de voir si la dynamique du CA est en mesure de produire certains types de modules dynamiques, à savoir, structures de haut niveau suffisamment persistantes, et dont le comportement au niveau supérieur peut être vu comme

régi par des règles, afin de remplir la condition de pouvoir voir la dynamique du CA à ce niveau plus élevé comme un autre système dynamique discret, de niveau supérieur, différent de la DDS constitué par le CA et sa règle. En d'autres termes, afin d'obtenir une explication computationnelle utile d'un CA, une première condition est (*i*) qu'une forme de *modularité* dynamique de haut niveau puisse être détectée de manière fiable dans la dynamique globale du CA. Une autre condition doit être remplie: (* ii *) la dynamique modulaires de haut niveau doit réussir à* suivre* correctement la dynamique de bas niveau du CA, sans diverger de celle-ci. Cette condition de *validité* (pour reprendre la terminologie de la modélisation informatique scientifique) est une condition assez complexe et elle est mieux spécifié dans les sections 2.2.1 et 6.6.8 de ce travail, mais elle revient essentiellement à ceci: que la dynamique de la description de haut niveau ne doit pas diverger, à mesure que le temps passe, de la dynamique correspondante du CA à son plus bas niveau de description, la description en termes des cellules du CA.

Il se trouve que certains CA sont effectivement dotés d'une telle forme de modularité robuste de niveau supérieur: comme nous l'avons vu, certains CA peuvent générer des *planeurs* (voir fig. 18.2) qui finissent par réaliser, dans de nombreux cas, des interactions l'un avec l'autre qui sont prévisibles, comme dans le cas de la *Règle 54*[46], et ces interactions prévisibles peuvent être considérées comme les implémentations de haut niveau de fonctions booléennes, avec le planeurs agissants comme des "bit" itinérants. Cette interprétation en termes de fonctions booléennes pourrait alors permettre de construire progressivement une explication à plusieurs niveaux en termes de computations sensées, plus ou moins de la manière dont les programmes informatiques peuvent être décrits par des langages de programmation progressivement de plus haut niveau. De cette façon, nous aurions construit une partie des conditions requises pour expliquer le CA au moyen d'une *explication computationnelle*.

Toutefois, cette interprétation en termes de planeurs n'est pas toujours possible: il y a certains automate cellulaires "chaotique", comme la *Règle 30* (voir fig. 18.7), qui ne montrent *jamais* des sous-configurations suffisamment robustes pour être considérées modules dynamiques en gré de rendre la représentation de haut niveau computationnellement capable[47].

Un point doit être souligné ici: cette impossibilité d'individualiser modules dynamiques stables dans un CA, comme dans le cas ci-dessus de la Règle 30, peut être considérée comme une forme d'*antimodularité intrinsèque* de la description de haut niveau du CA. Donc, nous pouvons dire que l'antimodularité, sous cette forme, empêche déjà la première étape, l'étape (* i *) ci-dessus, nécessaire pour fournir une explication computationnelle, une étape qui consiste à considérer le CA comme computationnellement capable à un niveau élevé de description. Donc, il semble que, au moins dans cette forme,* l'antimodularité intrinsèque empêche effectivement l'explication computationnelle*.

Cependant, il est certain que, pour certains CA, leur interprétation de haut niveau comme systèmes computationnels est effectivement possible: il y a une correspondance complexe, conçu par Matthew Cook[48], avec laquelle il a été en mesure de prouver que la *Règle 110*, un autre CA élémentaire, peut être considéré comme un système computationnel au niveau de la machine de Turing universelle. Aussi le plus célèbre CA, le *jeu de la vie* de John Conway, a été prouvé être Turing-complet[49]. Ainsi, il est un fait avéré que, sous certaines interprétations, certains CA

[46] Voir par exemple Martínez et al. (2014).

[47] Pourquoi ne peut-on pas arranger une correspondance entre des ensembles de configurations chaotiques et des symboles significatifs, afin de rendre même un CA chaotique capable de computation à haut niveau? Une réponse implique une discussion sur la complexité de la correspondance entre les configurations du système et les symboles, une discussion qui est développée dans la section 14.5.2.

[48] Voir Cook (2004).

[49] Voir Rendell (2002).

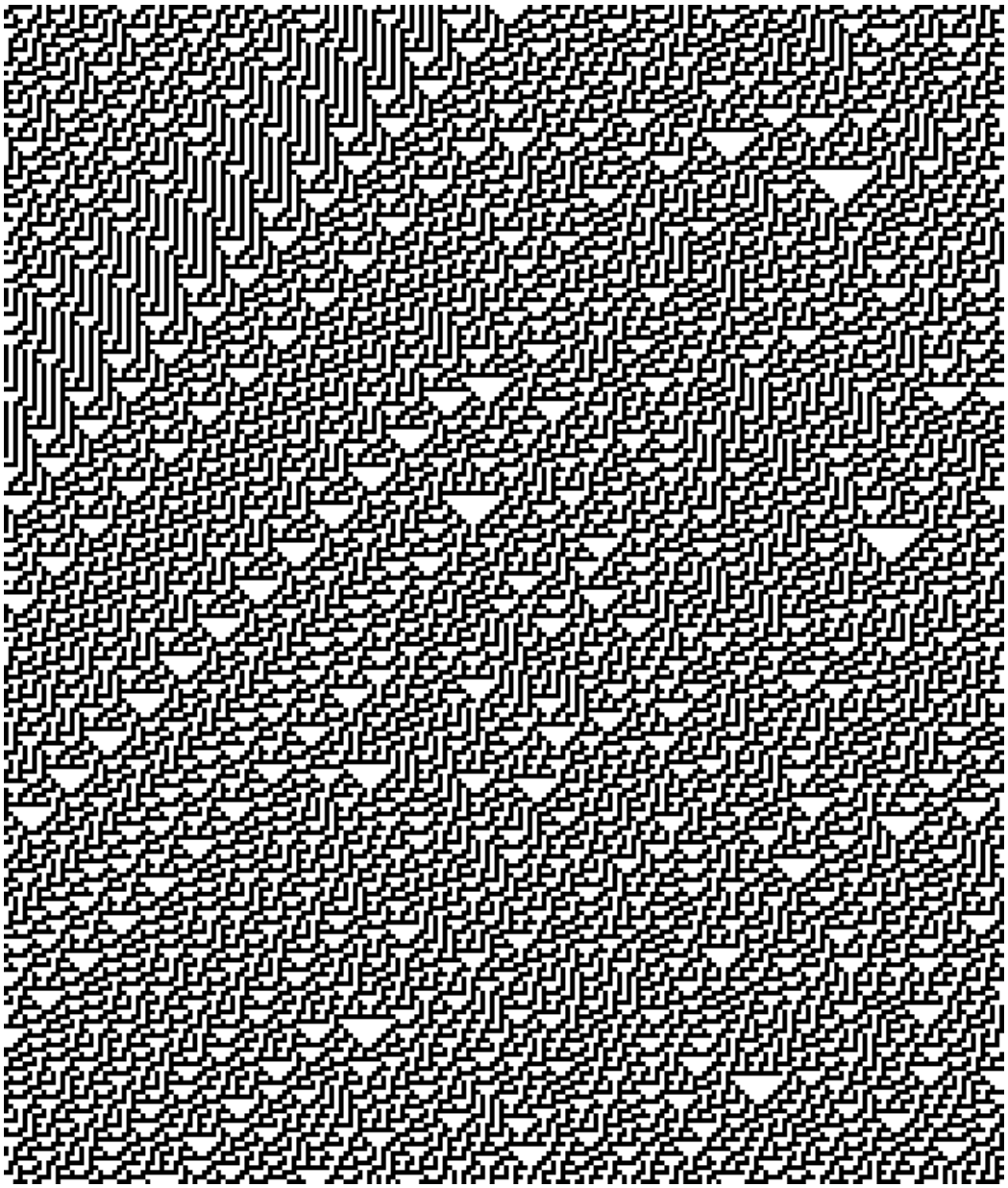

Figure 18.7: un segment chaotique de l'évolution du CA élémentaire *Règle 30*. Le temps coule de haut en bas, et chaque rangée de pixels représente la configuration globale du système à chaque pas de temps. Chaque pixel représente l'état de l'une des parties élémentaires du CA, ses *cellules*.

peuvent être considérés comme effectuant des computations: cette première condition peut être considérée comme établie, au moins pour certaines automates cellulaires.

Mais, si on veut donner une *explication computationnelle* d'un système, une autre condition doit être remplie: que le système en fait *compute*, et pas seulement qu'il est *capable* de computation. Et, à cet effet, nous devons d'abord être en mesure de dire *ce que* il est en train de computer: c'est-à-dire, nous devons être en mesure d'exprimer la relation d'entrée/sortie du système, sa *spécification*. On doit constater que nous travaillons ici dans le champ de la rétro-ingénierie: nous avons une machine, le CA, que nous savons être computationnellement capable, et nous devons, dans le but de l'expliquer computationnellement, *produire* sa spécification.

Mais, la tâche d'appliquer l'ingénierie inverse aufin d'obtenir la spécification d'un programme, est apparemment difficile. Essentiellement, c'est une question de produire toutes les entrées d'un programme et d'observer toutes les sorties correspondantes: cette tache nécessite d'une forme d'*induction*. Laissant les détails les plus fins à une autre occasion, je constate simplement ici qu'il y a une multitude de problèmes liés au fait que les systèmes computationnellement capables au niveaux de la machine de Turing sont touchés par le problème de l'arrêt, ce qui rend la tâche ci-dessus très difficile, voire impossible: en raison du fait que le nombre de couples possibles d'entrée/sortie à observer croît exponentiellement avec la taille maximale de l'entrée, elle est au moins une tâche computationnellement difficile (voir les sections 14.5.2 et 17.2.6 pour plus de détails). Il existe des algorithmes approximatifs qui visent à inférer des spécifications (algorithmes de "specification mining")[50]) qui, toutefois, donnent souvent des résultat trop approximatifs, et ne sont pas capable de rétro-ingénieriser des spécifications du computations à niveau de la machine de Turing.

Mais nous avons besoin de la spécification du programme afin de pouvoir expliquer computationnellement. Et cette spécification est très difficile à déduire.

Cependant, une spécification en termes de la fonction d'entrée/sortie brute n'est pas la seule façon dont une spécification peut être donnée, et, même si elle est la plus précise, cette façon n'est pas le moyen le plus clair pour donner une spécification, parce que une liste de couples d'entrées/sortie peut être incompréhensible. Donc, une autre forme, plus transparente, dans laquel une spécification peut être donnée, est une forme *agrège*: une façon plus ou moins synthétique de résumer l'ensemble de la fonction d'entrée/sortie. Une manière d'obtenir une telle agrégation c'est de fournir la spécification en termes de sa décomposition en sous-fonctions, qui est une forme de décomposition hiérarchique.

Ceci est le point intéressant: si une représentation *hiérarchique modulaire* de la computation pût être conçu par n'importe quels moyens, il serait possible de tester chaque module *séparément* pour chercher de determiner la spécification *uniquement* de ce module, une tâche qui très probablement se révélerait être plus faisable *par ordres de grandeur* que celle d'examiner toutes les entrées possibles pour l'ensemble du programme et les sorties correspondantes, afin de déduire directement la spécification gobale. La raison de cette simplification est qu'un module est identifiable par le fait même qu'il devrait être relié aux autres modules seulement de manière lâche ou faiblement, ce qui se traduit par une réduction probable du nombre d'entrées possibles pour le module, et, en conséquence, par une exploration plus facile de l'espace d'entrée de ce module[51].

Le fait qu'il a été possible de trouver la spécification unique de chaque module en raison de la décomposabilité du système, permet, si les spécifications de chaque module ne sont pas trop compliquées, une forme d'*agrégation*, tel celle discutée dans la section 4.1.5.1: si nous sommes en

[50] Voir la section 4.3.1.1.

[51] Même si cela n'est pas *garanti*. Voir la section 4.3.1 pour une meilleure discussion.

mesure d'abstraire la spécification du module en le nommant d'une manière succincte significative, en appelant le module avec un nom qui est représentatif et explicatif de la fonction que le module remplit (comme par exemple lorsque nous disons qu'un module effectue l'opération de "multiplication"), alors la spécification de chaque module peut être remplacée par cette définition plus concise de *la fonction* que le module effectue. Dans ce cas, une spécification globale de l'ensemble du système peut être donnée *en termes* d'une description (généralement sous la forme graphique d'un diagramme) de la structure modulaire du système comme un réseau orienté des modules connectés, où les modules sont vus comme des nœuds étiquetés avec leurs "noms" succincts représentants leur spécifications, et leurs connexions d'entrée et de sortie sont les liens orientés entre les noeuds.

Donc, ce type d'explication semble possible, après tout. Mais elle exige qu'une modularité fonctionnelle computationnelle *puisse effectivement* être trouvée, et cette condition, à son tour, exige deux autres conditions: d'abord, que le système est en fait computationnellement capable à haut niveau, et ça n'est pas garanti: les systèmes *intrinsèquement antimodulaires*, comme le CA *Règle 30* mentionné au-dessus, ne sont même pas susceptible d'être considérés comme computationnellement capable à un niveau élevé de description. Deuxièmement, une autre condition est que, même si le système est capable de computation, et il possède effectivement une modularité dynamique de haut niveau, cette modularité *puisse être effectivement trouvé*. Cette condition pourrait être entravée par certains facteurs, en raison du coût computationnel élevé des algorithmes pour la détection de modularité, ou de l'imprécision excessive des résultats qu'ils obtiennent, lorsque ces résultats peuvent être obtenus dans un délai raisonnable: les représentations hiérarchiques obtenues pourraient ne pas être suffisamment fidèles à l'organisation fonctionnelle modulaire réelle du système pour constituer des descriptions modulaires *valides*, capables de caractériser en maniére suffisamment precise la computation effectuée.

Il semble, par conséquent, que l'*antimodularité peut entraver ou empêcher l'explication computationnelle aussi*.

Mais, pourraient des reconstructions *partielles* de la hiérarchie fonctionnelle du programme être quand même utilisées dans les explications? Il semble, intuitivement que les modèles fonctionnels ainsi produits seraient très limités par des clauses *ceteris paribus*, afin de les tenir à l'intérieur de la gamme des relations d'entrée/sortie *connues*, et, parmi les relations d'entrée/sortie connues, dans la gamme de ceux qui ne diverge pas trop, par manque de validité du modèle modulaire, par rapport au comportement réel observé du système. Il semble donc qu'une explication basée sur des modèle partiaux devra être limitée dans son applicabilité. Elle pourrait apparaître comme une explication acceptable, mais ce ne serait dans un sens qu'une explication *post-hoc* de la gamme du comportement du système effectivement observée au cours des processus de détection de la modularité, et pas une explication de tous les comportements possibles.

Dans les sciences cognitives computationnelles, un tel type d'explication limitée pourrait bien être acceptée, et, de plus, il est probable qu'il y a *seulement* explications de ce genre dans certains sous-champs de la psychologie cognitive. Dans cette science, la tâche de trouver la spécification et les relations fonctionnelles entre les modules, est laissée à l'expérimentation humaine, et cela est probablement un processus plus lent que les processus algorithmiques automatisés.

Pour conclure cette section, je pense que cet échec à trouver des explications computationnelles de certains systèmes peut être considéré comme une forme de *émergence antimodulaire*. Cette émergence computationnelle concernant les explications computationnelles peut être considéré comme due à deux formes d'antimodularité: d'abord, l'*antimodularité intrinsèque* qui affecte les systèmes chaotiques, systèmes qui ne peuvent même être considérés comme de systèmes *capable*

de computation à un haut niveau de description. La seconde, c'est une antimodularité due au coût excessif de calcul ou à l'imprécision excessive des algorithmes de détection de la modularité.

La conséquence de cette forme d'émergence antimodulaire est que le système affecté par elle est explicable seulement à son plus bas niveau de description, et cela en général ne va pas constituer une forme intelligible d'explication, dans des systèmes suffisamment compliquées. Il est à prévoir que l'émergence antimodulaire pourrait affecter l'explication computationnelle dans les science cognitives et les neurosciences aussi, et en raison de cela il vaut la peine, il me semble, de proceder à l'investigation.

18.4.9 La modularité de haut niveau comme une condition pour la programmation et la recherche scientifique

Au moins pour un programmeur informatique, l'affirmation par Robert Cummins[52] que l'analyse fonctionnelle a une capacité explicative, n'a rien de nouveau: un programmeur, au moins implicitement, développe continuellement des *explications* partielles de la façon dont le programme en cours de construction fonctionne au moyen de l'exécution organisée de ses instructions, et, à un niveau supérieur, de l'exécution organisée de ses sous-programmes ou des modules de niveau encore plus haut. L'action de la programmation elle-même commence à partir de la *spécification* de l'ensemble du programme (par exemple, "être un logiciel de traitement de texte"), et le développement procède en analysant, dans un sens à la Cummins, cette fonction globale, qui est la spécification, dans les petites sous-fonctions qui forment ensemble la mise en œuvre (l'implémentation) de la spécification. À son tour, chaque sous-fonction se décompose, si possible, dans des sous-fonctions plus simples, et ainsi de suite. L'écriture réelle du programme, l'acte d'écrire les séquences d'instructions qui composent chaque sous-programme, généralement en sautillant entre les différents niveaux hiérarchiques, en analysant des fonctions globales en petits sous-programmes, en faisant leur mise en œuvre et en revenant à décomposer les autres fonctions de haut niveau, ou en les composant à partir des sous-routines simples, et ainsi de suite à traver les niveaux hiérarchiques, est une action presque impossible sans une *explication* précédent, au moins implicite, de la part du programmeur il-même, de l'ensemble du système en termes hiérarchiques. Cela peut être vu comme une forme d'*explication fonctionnelle* façon Cummins. Il semble que l'explication fonctionnelle hiérarchique est donc nécessaire pour la *programmation* de l'ordinateur, en informatique.

Dans la science empirique, l'explication fonctionnelle à plusieurs niveaux est probablement essentielle non seulement *après* une théorie ou un modèle d'un phénomène a été mis au point, c'est-à-dire lorsque on explique un phénomène déjà connu, mais aussi *dans la fabrication* d'une théorie. Interpretations interventionnistes de la causalité comme celle de James Woodward[53] considerent qu'une relation de cause à effet entre deux entités est presènte lorsque une variation hypothétique de l'état d'une entité, délibérément induite par un expérimentateur (une *intervention*) produit systématiquement une variation de l'état de l'autre entité: si cette circonstance hypothétique tient, nous pouvons dire que les deux entités sont en relation causale. L'explication mécaniste exige, pendant la recherche scientifique, d'ameliorer et de raffiner graduellement la description mécanique d'un mécanisme en découvrant progressivement toutes les relations causales présentes entre ses parties. Pour cela faire, l'expérimentateur procède en intervenant sur chaque partie séparément, et en observant si une certaine variation conséquente se produit sur d'autres parties. Mais, pour identifier correctement les liens de causalité, l'intervention sur l'état d'une partie nécessite une perturbation temporaire, au moins virtuelle, de la structure des liens de

[52] Voir la section 9.2.
[53] Voir Woodward (2003), et la section 6.9.

causalité allant de d'autres parties du mécanisme vers la partie sur laquelle nous intervenons: les interventions sur les mécanismes nécessitent que le mécanisme soit temporairement *modifié* en éliminant certaines des connexions entre ses parties. Woodward affirme que l'ensemble des équations représentant correctement un système causal, doit être *modulaire*, parce que sinon, puisque la détection de liens de causalité nécessite une intervention sur une partie du système en perturbant temporairement seulement l'influence causal qui porte sur cette certaine partie, si le système fût complètement non modulaire, cette déconnexion d'un trajet précis de causalité ne perturberait pas seulement la partie de l'équation intéressée par l'intervention, mais également d'autres parties du système d'équations. Ainsi, la modularité dynamique est toujours présente dans un mécanisme, au moins au niveau le plus bas, le niveau de la description préférée.

Mais, le point est, si nous voulons redécrire un système en maniere mécaniquste à haut niveau, nous pourrions certainement interpréter les relations entre des parties de haut niveau comme prima facie des relations causales "de haut niveau". Dans ce cas, afin de procéder par intervention façon Woodward, la modularité est également nécessaire dans les équations représentantes la dynamique du système à ces niveaux *haut*. Tout bien considéré, cette condition est vérifiée si la structure de la fonction de mise à jour[54] est *hiérarchiquement* modulaire, ce qui représente à son tour le fait que le système est fonctionnelment, et probablement aussi, de façon dynamique, et, très probablement, structurellement, *hiérarchiquement modulaire*.

La même utilité de la présence d'une structure modulaire et de sa détection faisable, est présente dans la phase de découverte de réseaux complexes, en particulier dans les cas où la découverte de liens entre les nœuds nécessite d'un travail expérimental complexe, comme dans le cas du réseaux de régulation gènetiques et d'autres réseaux d'intérêt biologique. Certaines méthodes algorithmiques récemment proposées, comme celle dans Clauset et al. (2008)[55], pourraient être d'une grande aide dans ce genre de tâche, car, sur la base de la modularité hiérarchique détectée dans la portion du réseau déjà partiellement découverte, elles peuvent produire en manière probabiliste, avec une bonne chance de succès, une prédiction sur les endroits où, dans le réseau, des liaisons manquantes devraient probablement se présenter avec plus d'observation. De cette manière, cet algorithmes peuvent guider utilement l'expérimentation ultérieure.

Tout bien considéré, il semble que la modularité hiérarchique est important, voire essentiel, dans la phase de la recherche scientifique et de la *découverte* expérimentale, en plus d'être presque indispensable, comme nous l'avons vu dans les sections précédentes, pour l'explication d'un phénomène déjà étudié.

18.4.10 Émergence explicative

Consideré que le manque de compréhension en raison de la présence dans un système de l'émergence antimodulaire peut apparemment affecter la plupart de types d'explication, je me propose de généraliser la notion avec la définition suivante:

> l'*émergence explicative* est une propriété de systèmes ou de leurs descriptions qui consiste dans le fait que, pour des raisons computationnelles absolues ou pragmatiques, ce systèmes ou descriptions résistent à toute explication compréhensible.

Ceci est une définition plus générale que celle de l'antimodularité, car elle comprend d'autres effets possibles des contraintes calculatoires sur l'explication des systèmes complexes. Dans

[54] La fonction régissant la dynamique d'un DDS: voir la section 5.1.
[55] Décrit dans la section 6.9.

la section 14.6, j'explique comment je comprends la définition ci-dessus, qui ne convient pas nécessairement à des tâches de calcul automatique, mais aussi à des tâches humaines qui portent sur des explications, et l'utilité éventuelle de cette définition dans le paysage scientifique actuelle.

18.4.11 Est-il probable de rencontrer des systèmes antimodulaires dans la science?

Dans ce qui précède, j'ai proposé d'envisager une notion, l'*antimodularité*, qui correspond à l'impossibilité d'obtenir une description modulaire d'un système à un niveau plus élevé que celui de sa description préférée choisie, qui se trouve au plus bas niveau, et j'ai essayé d'analyser les raisons de l'apparition de l'antimodularité.

L'antimodularité semble dépendre du *choix* d'une relation, qui est spécifiée dans la description préférée, entre les parties élémentaires du système. L'antimodularité peut se produire lorsque, compte tenu de cette relation choisie, la détection de la modularité selon elle se révèle être trop exigeantedu point de vue computationnel pour être menée à bien en un temps possible, ou lorsque, bien que la détection de la modularité est terminée avec succès au moyen d'un algorithme approximatif, la description modulaire ainsi produite semble trop approximative pour être capable de représenter validement le système d'origine.

Quelle est la probabilité que l'un de ces deux circonstances peuvent être rencontrées lors de la recherche scientifique? Il faut souligner que la complexité calculatoire de la détection de modularité concerne algorithmes pour la détection de modularité qui n'emploie pas d'autres informations sur les systèmes que celles incluses dans les description préférées de ces systèmes, c'est-à-dire dans le niveau de leurs parties élémentaires et de leurs relations. En ajoutant des contraintes sur la façon dont les pièces élémentaires peuvent être regroupés en modules, la tâche peut être très simplifiée. Ceci est équivalent à concevoir des algorithmes ad hoc pour la détection de la modularité, et des algorithmes ad hoc pourrait finir par être moins exigeants que les algorithmes d'ordre général pour la détection de modularité. En fait, dans de nombreux cas, cela semble exactement ce que la science fait: elle recherche les contraintes empiriques pour nous aider à choisir parmi les théories possibles du monde. Cela augmente la probabilité que la méthode scientifique puisse produire descriptions modulaires intelligibles des phénomènes.

Mais, nous devons nous demander si les nouveaux développements de la science peuvent déplacer l'accent sur des systèmes d'une telle complexité que même les contraintes à leur sujet qui ont eté empiriquement trouvées, pourraient finir par être trop peu pour permettre la réussite de la détection de modularité sur de tels systèmes.

Dans le cas des systèmes biologiques, nous pouvons être raisonnablement sûr qu'ils sont modulaires, au moins à certains niveaux. Il existe de nombreux arguments, empiriques et théoriques, qui favorisent cette conclusion, qui sont traités dans la section 7.1. Néanmoins, il peut y avoir des systèmes biologiques importants, comme par exemple, des réseaux d'interaction dans le métabolisme de la cellule, qui peut finir par être tellement énormes qu'il produisent éventuellement des effets d'émergence explicative ou antimodulaire en raison du coût computationnel élevé, par rapport à la taille du système des algorithmes pour l'extraction de données ou, plus spécifiquement, pour la détection de la modularité. Un autre type de situation dans lequel on peut prevoir que cette émergence antimodulaire peut se manifester, est dans l'extraction de données au sujet de la modularité de certains systèmes à partir de la *littérature* déjà existante sur certains sujets scientifiques, dont un exemple est présenté dans la section 18.5.2.

18.5 Quelque réflexion supplémentaire sur la modularité, la métaphysique, l'informatique, l'histoire des sciences

Dans les sections ci-dessus, je l'ai souligné la structure principale de ce travail, dans lequel je tente de réfléchir sur la notion de modularité, sa relation avec la description, l'explication, la computation, la compréhension, et de définir les conditions dans lesquelles la modularité se manifeste ou, au contraire, ne peut pas être détectée. Mon objectif général est de décrire l'importance de ce que nous pourrions appeler une "façon modulaire de penser" pour la connaissance humaine du monde, et plus particulièrement la grande importance de la modularité sur une grande partie de la conceptualisation scientifique, en particulier dans les soi-disantes "sciences spéciales", qui recourent principalement à types d'explication, *mécanistes* ou *fonctionnelles*, qui sont les types modulaires d'explication *par excellence*: pour expliquer dans ces deux manières, il est nécessaire de trouver une façon de décrire un système comme modulaire .

J'ai alors concentré mon attention spécifiquement sur les moyens de détecter la modularité dans des grands systèmes complexes, insistant sur le fait que, malheureusement pour la science, ces moyens sont algorithmiquement complexes et pour cette raison ils ne sont pas garantis de donner des résultats utiles. Quand un manque à donner des descriptions modulaires apparaît, cela est ce que j'appelle un cas d'*émergence antimodulaire*. Ce fait entrave et parfois, empêche , la possibilité d'explication scientifique et la compréhension des systèmes suffisamment grandes et complexes pour échapper à une explication modulaire appropriée, systèmes qui, en raison de leur taille, ne peuvent pas être saisi cognitivement sous la forme de leur description non-modulaire à bas niveau, qui est la seule description possible qui leur reste. J'ai appelé cette condition "émergence explicative". Tel est l'objectif principal de ce travail, que je conçois comme une œuvre de philosophie de la science, en se concentrant en particulier sur les disciplines de dérivation biologique et leurs méthodes, et, dans le même temps, sur la computation, vue comme une méthode pour la recherche dans ces disciplines, mais, tout d'abord, comme un cadre théorique prometteur, à la lumière duquel essayer de repenser certaines énigmes classiques de la philosophie, une tentative que je fais en essayant de reformuler comme redescriptions computationellement possibles ce qui a été traditionnellement conçu comme des explications, et comme une relation *spécification/mise en œuvre* la relation entre niveaux d'organisation qui a été traditionnellement considérée comme une relation de constitution.

Les questions ci-dessus, constituent l'épine dorsale de ma proposition en philosophie des sciences, sont plus profondément discutées dans les autres chapitres. Maintenant, je vais consacrer les deux prochaines dernières sections de ce chapitre, à des réflexions au sujet de certaines questions qui en un sens ne concernent pas la matière principale de ce travail comme je la viens de résumer.

La première (section 18.5.1) est une réflexion sur les conséquences possibles de nature métaphysique découlant de mon point de vue sur la nature des explications, qui est un point de vue éminemment épistémique. J'ai dit que cette réflexion est en dehors du champ principal de ce travail parce que je ne veux pas immerger profondément dans un contexte métaphysique mon discours sur la philosophie de l'explication dans la science: Je ne suis pas un métaphysicien, ni ma position est complètement anti-métaphysique, mais j'ai probablement une inclinaison vers l'être prudent losrque on traite avec des questions vivement métaphysiques. Le long des lignes de Kant, je ne peux pas éviter d'être un peu méfiant au sujet des revendications métaphysiques fortes, car elles pourraient avoir parfois des conséquences graves, et il n'est pas très clair quand ces revendications peuvent généralement être considérées suffisamment justifiées. Pour cette raison, je comptais de tenir la réflexion philosophique sur la science qui constitue l'objectif principal de cette thèse plus ou moins libre d'interférences métaphysiques explicites (bien que je suppose que

cela pourrait être jugé une planification très discutable, et probablement une projet de faisabilité incertaine) .

Néanmoins, certaines conséquences métaphysiques de l'approche générale que j'ai adopté et des résultats au sujet du contraintes computationnelles sur la description de la modularité dont j'ai parlé, ouvrent, il me semble, la porte à certains fascinantes (bien que risquées) possibilités de délimiter une position métaphysique particulière, et je ne veux pas manquer la chance d'essayer d'envisager brièvement cette position. Je consacre la section suivante pour ce but "expérimental". L'esquisse fournie ici est au mieux approximative. Ce qui est pas bien pensé ici sera mieux laissé pour une prochaine occasion.

Dans la section suivante (section 18.5.2) je me permets quelque autre initiative, éventuellement imprudente, celle de prévoir des éventuelles conséquences historiques à long terme derivants de la relation entre computation et explication scientifique qui je décris dans la principale ligne de ce travail. Dans ce cas aussi, je veux laisser cette réflexion en dehors de l'épine dorsale philosophico-scientifique de la thèse, mais la raison pour ce choix n'est pas que je pense que la réflexion historique ne soit pas à sa place dans un ouvrage de philosophie des sciences. Bien au contraire, en fait: souvent, dans de nombreuses parties des principaux chapitres, je longe, si possible, les lignes de la reconstruction historique des débats récents plus connus autour des grandes questions en jeu. Je pense, en effet, que l'exposition des idées et des discussions au moins du point de vue chronologique, sinon historique, est éminemment importante pour l'écriture et la réflexion philosophique, même dans une des disciplines essentiellement analytiques comme la philosophie de la science. La raison pour laquelle je laisse ces considérations historiques en dehors du cadre principal de ce travail n'est donc pas un mépris pour l'histoire des idées, de la philosophie ou de la science, mais la nature intrinsèquement intellectuellement risquée de ces réflexions, qui deviendra évident dans le corps de la section dédiée. Néanmoins, je juge telles réflexions comme dotées d'un potentiel de produire recherches plus rigoureuses, qui sera mieux de ne pas ignorer, au moins par mes recherches avenir.

18.5.1 Une tentative métaphysique: Modularité comme ontologie? Antiréalisme contraint

Comme nous avons vu, une des caractéristiques importantes et nécessaires d'un module est sa robustesse: un module est quelque chose qui doit endurer toute une gamme de perturbations, et qui doit persister dans un certain laps de temps (selon quelque échelle de temps). Un autre trait caractéristique de modules est qu'ils jouissent d'une certaine quantité d'indépendance du reste du système et d'autres modules, et cela est du à un certain *isolement*, au moins partiel, du module, à savoir au fait que le module posséde une certaine forme de *frontière* reconnaissable. Ce sont des propriétées que les modules partagent avec les *entités*, ou avec les *objets*, c'est-à-dire avec ceux qui peuvent éventuellement être considérées comme le composantes ontologiques de base du monde. Je dirais que celle-ci n'est pas une simple coïncidence. Je pense qu'il est plausible de dire que notre système perceptif fonctionne par une *détection de modularité* sur les données brutes qui empiète sur nos corps, afin de donner une description du monde en termes d'entités ou d'objets: les objets que nous percevons sont les modules produit par ce procédure de détection de modules. Il est plausible, je crois, prendre en considération les limites de cette détection de modularité effectuée par chaque organisme, à la lumière des limites que nous avons vu qui affectent les algorithmes de détection de la modularité. Bien sûr, il y a des différences majeures: premièrement, les systèmes perceptifs ne sont *pas* des algorithmes en série, comme ceux mis en œuvre sur les ordinateurs standard, et pour cette raison les systèmes perceptifs pourraient presenter une complexité computationnelle inférieure. Cependant, si nous prenons

pour acquis une forme de déterminisme physique (au moins macroscopique) et la finitude de systèmes biologiques de perception, l'équivalence computationnelle de ces processus perceptifs avec un algorithme devrait être assez garantie: les systèmes perceptifs peuvent être vus comme des systèmes computationnels. Nous devons alors considérer que les algorithmes dans certaines classes de complexité sont *intrinsèquement* difficiles, indépendamment de l'architecture de calcul sur laquelle ils sont mis en œuvre: par exemple, même si des architectures hautement parallèles sont utilisées, comme celles des réseaux de neurones, les problèmes de classe EXPTIME ne peuvent être traitées avec succès, parce que les temps nécessaire pour leur achèvement croît de façon exponentielle, tandis que la quantité de parallélisme peut croître au plus linéairement[56], et la fonction exponentielle croît incomparablement plus rapide que n'importe quel polynôme. Un autre objection peut néanmoins être soulevée: les systèmes perceptifs n'ont *pas* besoin de *découvrir* une bonne structure modulaire dans un tas initialement non structuré de stimuli, parce que les systèmes perceptifs sont déjà adaptés à détecter une plus ou moins bien connue structure modulaire dans un monde qui est plus ou moins stable: un organisme est adapté à détecter certains types d'objets dans le milieu environnant, et son système de perception est structuré et *déjà biaisé* vers la détection de *ces types* de modules, à savoir ces sortes d'objets dans le monde. Ce réglage fin a été, dans une vue darwiniste classique, produit par la sélection naturelle. Un tel processus biaisé de détection de modularité pourrait bien être *beaucoup* moins computationnellement complexe que le processus brute de trouver un bonne structure modulaire initialement *inconnue* dans un grand ensemble de données non structurées, ce qui nécessite l'exécution d'une tâche NP-complete, la tâche de l'optimisation de la mesure de modularité Q[57]. Ceci est, au moins, tout à fait probable: les processus perceptifs ne sont probablement pas si complexe du point de vue computationnel. Il y a, cependant, deux répliques. La première est que les processus de perception ne sont pas moins limités par la complexité computationnelle, au moins indirectement, que l'optimisation de la modularité: les processus perceptifs comme ils sont maintenant, ont eté obtenus par la sélection naturelle (au moins selon les explications darwinistes), et il est *ce processus*, le processus de la sélection naturelle, qui a porté le fardeau d'essayer d'optimiser la mesure de modularité Q sur un monde perceptif initialement non structuré. La NP-complétude est dans certains cas si difficile à traiter que pas même la sélection naturelle peut être réputée avoir eu assez de temps pour effectuer une recherche exhaustive parmi l'espace phénotypique des systèmes perceptifs possibles, de sorte qu'il peut être soutenu que les systèmes de perception réels sortis de l'Evolution ont indirectement été affectée par ces mêmes contraintes computationnelles. C'est-à-dire, la complexité computationnelle de la tâche de recherche de la meilleure description hiérarchique modulaire est excessive, même pour une recherche effectuée au cours de l'évolution. La deuxième réponse à la première objection découle de cette dernière considération: très probablement en raison de leur complexité de calcul inférieure, les processus perceptifs sont *moins précis* que l'optimisation algorithmique de Q, et ceci est une autre façon de dire que la détection de la modularité *optimale* sur les données qui proviennent du monde empirique, est *pas* la fonciton que les systèmes de perception biologique effectuent: la perception est indirectement limitée par la difficulté calculatoire de la détection de modularité, dans le sens qu'elle est rendue plus ou moins peu fiable par les effets de cette complexitè.

Un autre indice en cette direction est le fait que selon une interpretation interventionniste de la causalité comme celui de James Woodward, qui a été adoptée en tant que position standard par certains défenseurs de l'explication mécaniste, par exemple Carl Craver, la modularité des équations dynamiques qui régissent le système est nécessaire pour séparer les variables sur lesquelles intervenir lors de l'expérimentation, afin de discriminer les causes simples. Le long de ces lignes,

[56] Ou, éventuellement, quadratiquement, ou au plus cubique.

[57] Q est une mesure de la qualité de la modularité détecté par un algorithme. Optimisation, cela signifie de choisir le meilleur de toutes les descriptions possibles d'un système hiérarchique. Voir la section 3.2.1.2.

la décomposition mécanique d'un phénomène dépend de la modularisation possible de ses états globaux. Bien sûr, l'identification des parties, ou des entités du mécanisme, qui sont les éléments causalement actifs d'un mécanisme, dépend du succès de cette évaluation de causalité par petits bouts. Il semble donc que l'ontologie même d'un mécanisme, l'ensemble de ses *parties*, dépende de la modularizabilité de sa dynamique. Cette dernière considération, à la lumière de la difficulté calculatoire de l'optimisation de la modularité, signifie que, très probablement, la science expérimentale n'est pas et ne sera pas capable de produire l'ontologie du monde la *plus* plausible, non seulement si l'expérimentation est menée par petits bouts par des sujets humains, mais pas même si elle est entièrement automatisée: la complexité de calcul de l'optimisation de Q ou de l'agrégabilité des variables dans les modèles dynamiques[58] est impossible à surmonter. Il semble alors assez probable que l'ontologie de mécanismes découverts par la science ne soit pas la meilleur ontologie possible.

On peut se demander où cette étrange position qui voit la modularité comme ontologie se positionne le long de la ligne réalisme-antiréalisme, et pas seulement dans le sens classique de l'antiréalisme sur les entités non observables posées par les théories scientifiques, mais dans une sens plus large, qui englobe tout et s'interroge sur la *réalité* même des objets macroscopiques de taille moyenne. En outre, si notre action de découper le monde en morceaux sensés est un effet de la possibilité de la détection de modularité, et si cette possibilité à son tour dépend de contraintes computationnelles, il est légitime de se demander où ces contraintes dérivent. Cependant, il est pas dans le champ d'application de cette thèse de se plonger dans des discussions métaphysiques profondes. Donc, le plus que nous pouvons faire ici est d'admettre que si nous identifions modules qui sont réalistement détectable, compte tenu des contraintes découlantes de la complexité computationnelle, nous pourrions dire que ce point de vue est une forme de ce que je qualifierais *antiréalisme contraint*: cela est une position qui peut être considérée antiréalistique, parce que, selon elle, on ne peut pas savoir quelles entités sont *réelles* en soi. Mieux, dans ce point de vue, la question ne pourrait même avoir beaucoup de sens: La réalité des entités, c'est-à-dire leur être doué de robustesse et frontières, est le *résultat* du processus de détection de la modularité, qui est lui-même contraint par la complexité computationnelle. La réalité des entités, dans cette vue, ne constitue pas une caractéristique *intrinsèque* des entités, mais elle dérive davantage de l'objectivité des contraintes computationnelles qui affectent la détection de la modularité. Cela est une forme d'antiréalisme, certainement. Toutefois, il est antiréalisme doté d'*objectivité*, parce que les contraintes computationnelles auquelles cette position est soumise, sont objectives et, bien que leurs conséquences sont de nature pragmatique, le limitations qu'ils posent sont insurmontables[59] . Dans un sens, nous pourrions même juger l'antiréalisme contraint une forme de *réalisme*, parce que l'objectivité absolue des contraintes computationnelles pourrait déposer en faveur de leur *réalité*, dans le sens d'une existence indépendante. L'antiréalisme contraint pourrait alors être considéré comme une forme de *réalisme faible*, qui est une tout autre sorte de réalisme de ceux typiques, même du genre platonique. Dans un sens, je pense que cela pourrait être considéré comme une sorte particulière de *kantisme*, dont les conditions transcendantales prennent la forme de contraintes computationnelles.

18.5.2 Méthodes computationnelles dans la recherche scientifique: un possible tournant historique?

Toutes les considérations ci-dessus pourraient suggérer une thèse historique. Mais je soupçonne que cette thèse, que je voudrais proposer, pourrait éventuellement être accusée de ne pas être

[58] Voir les sections 18.1.5 et 2.2.1

[59] Sauf, bien sûr, que P = NP. Mais cela semble très peu probable pour le moment. Pour une explication, voir la section 17.4.1.3.3.

bien soutenue. La principale raison de ce soupçon est que je ne suis pas sûr qu'elle puisse être considéré comme une thèse *historique*, concernante des faits du passé: elle pourrait bien être une thèse sur l'*avenir*, sur des développements historiques naissants dans la science. En tout cas, il me semble que il y a des faits possibles intéressants, liés à des méthodes computationnels, à la modularité et à l'antimodularité, qui sont très proche dans le temps à l'instant présent et sur le point de se produire, s'ils n'ont pas déjà eu lieu.

Donc, la thèse historique que je suis sur le point de suggérer ici n'est *encore*, probablement, très soutenue per le faits, et elle est sans aucun doute une thèse assez forte. Je suis conscient que cette combinaison de fort impact et de faible soutien est assez dangereux. Ce que je voudrais faire est alors, au moins pour le moment, de donner seulement une suggestion, ou un simple soupçon, vers la possibilité qu'un changement de paradigme majeur dans la science vient de se produire ou pourrait être sur le point de se produire. Ce changement de paradigme a eu lieu ou est sur le point de se produire, grace à la disponibilité et à l'utilisation, à des fins diverses, de puissants machines computationnelles et algorithmes, dans plusieurs aspects de la pratique de la recherche scientifique. Les utilisations de l'ordinateur dans la science dont je voudrais considérer les conséquences ici sont deux types d'utilisation, appartenant à deux phases différentes de la recherche scientifique.

Commençons avec le type le plus évident: la *simulations informatique*. Quelque chose, je dirais, a été ou est en train de changer dans la science depuis le moment où des simulations informatiques complexes ont eté, ou seront, acceptée comme des *explications scientifiques adéquates*. Le point est: depuis sa création comme physique galiléenne jusqu'à une époque très récente, la science moderne a, plausiblement, prise en considération systèmes qui sont explicables en termes relativement simples, ou qui sont susceptibles d'être décrit par des modèles approximatifs suffisamment fidèles, selon les objectifs de l'observateur, au véritable phénomène empirique. Au cours de la thèse, j'ai essayé de montrer que certains systèmes, pour des raisons de *complexité computationnelle* (raisons qui dans un sens sont pragmatiques, mais d'un autre point de vue sont objectives) ne peuvent pas être décrits en manière *modulaire*. Cette *antimodularité* à pour conséquence de rendre ces systèmes susceptibles seulement de descriptions de bas niveau. Le problème est que ces descriptions généralement ne permettent pas un haut degré de clarté, en raison de l'énorme quantité de détails qu'ils portent. Cependant, dans de nombreux cas, la *simulation informatique* peut néanmoins modéliser dynamiquement ces systèmes non modulaires complexes, rendant possible la prédiction, au moins la prédiction par simulation étape-par-étape qui recouvre une certaine gamme temporelle finie: ces systèmes sont, d'une manière dynamique, prévisibles, mais leur comportement ne peut pas être *expliqué* d'une manière compréhensible. Est-ce un vrai problème? Dans la science, jusqu'à nos jours, il semble que les explications *sont effectivement* intelligibles. La plupart des explications de systèmes complexes naturels, jusqu'à maintenant, dans une certaine mesure sont explications intelligibles. Le modèle mécaniste, de plus en plus utilisé en biologie, a toujours souligné la nécessité d'une explication *multi-niveaux*, qui apporte avec elle un haut degré potentiel de intelligibilité. Mais, les modèles mécanistes qui sont touchés par l'antimodularité ne permettent pas l'explication à plusieurs niveaux, et ce fait rend ces modèles mécanistes antimodulaire très difficile à comprendre. Néanmoins, ils sont encore une base possible pour la simulation numérique. Doivent ces modèles mécanistes très complexes de bas niveau être acceptés comme explications appropriées eux-mêmes, sans s'attendre une amélioration de leurs intelligibilité?

Qu'est-ce qui constitue l'*explication* du phénomène, dans ces cas? Est-il le programme exécutant la simulation? Le programme, en général, est modulaire, parce il est écrit par le programmeur, et, en tant que telle, il est potentiellement facile à être entendu, mais si le système simulé est vraiment antimodulaire, par définition d'antimodularité ce programme ne peut pas constituer

une description modulaire *de haut niveau* du phénomène: le programme simulant un système antimodulaire pourrait être compris *dans ses propres termes* comme un programme modulaire hiérarchique, mais pas comme une *description* du système simulé. Certes, il serait un programme compréhensible, mais ce que nous comprendrions serait le fait qu'il simule le phénomène complexe en réitérant un nombre énorme de fois certaines opérations simples, précisément les activités simples des pièces élémentaires du système qu'il simule. Il ne me semble pas que le *programme* pourrait être considéré comme une explication: étant isomorphe à la description dynamique de bas niveau du système, il ne constituerait pas une explication computationnelle *multi-niveau*.

Alors, que devons-nous considérer comme explication d'un phénomène antimodulaire simulé par ordinateur? Je pense qu'il est *l'ensemble* de la simulation dynamique qui doit être pris comme explication, et, étant donné que le phénomène est antimodulaire, la simulation ne peut être que regardée et observée, mais pas *entendue* en détail. Au moins elle ne peut pas être comprise de manière fonctionnelle ou mécaniste multi-niveau. Elle pourrait néanmoins être expliquée de façon significative d'une manière *topologique* (voir la section 18.4.5): en prenant en considération certaines caractéristiques générales du réseau constituant le modèle du système complexe, une conclusion pourrait probablement être tirée à propos de certaines fonctionnalitées de la dynamique qui se produit, par simulation, sur ce modèle.

Donc, nous revenons à la question: faut-il accepter des simulations mécanistes complexes de très bas niveau comme des explications appropriées eux-mêmes, sans s'attendre des nouvelles améliorations de leur intelligibilité? Si la réponse est oui, alors la science a subi un grand changement historique: la science pourrait éventuellement aborder les systèmes qui, étant trop complexes et interconnectés pour être les objets de descriptions modulaires, auraient du être laissés en dehors de la recherche avant l'avènement de la simulation informatique. Cela a déjà eu lieu en partie, au moins depuis trois décennies: il suffit de penser toute la littérature sur la simulation des systèmes complexes et chaotiques qui a prospérée depuis les années 80. Il faut noter que, puisque la plupart de ces systèmes simulés ést antimodulaire, au moins dans certaines régions de leurs espaces de phase, les explications typiques utilisées dans les textes sur ce sujet (comme, par exemple, les œuvres de Stuart Kauffman), sont explications de nature statistique ou *topologique* (dans le sens employé par Philippe Huneman, voir la section 18.4.5), une forme d'explication qui, comme nous l'avons vu, est possible dans les systèmes antimodulaires.

Une autre façon possible dont les ordinateurs pourraient révolutionner la recherche scientifique, une manière qui est une forme spécifique de détection de modularité, est l'aide à la *recherche* du'un modèle théorique. Dans ce cas, un algorithme remplace en partie le chercheur, pas dans le recueil des données brutes, mais dans la tâche de concevoir un modèle théorique approprié pour de données *deja* disponibles. Et pourtant, cela n'è pas toute l'histoire, parce que, parfois, il arrive que même la récolte de données brutes peut être effectuée automatiquement. Cela est particulièrement vrai dans les cas où l'objet de l'étude est lui-même un objet *numérique*: étude de textes, d'une littérature, du contenu ou de la structure de l'internet, et ainsi de suite. En tout cas, après la phase de récolte des données, la nécessité d'un modèle théorique capable de subsumer toutes les données recueillies survient: généralement, le modèle est conçu par le chercheur humaine. Mais qu'en est-il des cas dans lesquels l'expérimentateur est incapable de concevoir un tel modèle? Qu'en est-il des cas dans lesquels la quantité de données est si énorme qu'il ne doit pas être attendu que tout être humain ne sera capable de discerner un motif en ces données, afin d'élaborer un modèle théorique? Ici aussi (sauf dans les cas où les données, bien que complexes, peuvent être *agrégées* d'une manière qui permet une description simple, comme en mécanique statistique), c'est une question de discerner une certaine structure dans les données, une structure modulaire de quelque type, afin que les données soient susceptible de modélisation

et d'explication compréhensible par l'homme. En effectuant ce qui est appelé "data mining", les ordinateurs ont pu, dans certains cas, supplémenter les êtres humains dans cette tâche.

Je voudrais mentionner ici un cas particulièrement surprenant. En utilisant une méthode de détection de la structure de communauté (c'est-à-dire, une forme de détection de modularité[60]) dans les réseaux, Wilkinson & Huberman (2004) ont pu analyser algorithmiquement la littérature de recherche publiée sur le cancer du côlon, et de trouver automatiquement dans le réseau de régulation génique humaine des modules de gènes impliqués dans le cancer du côlon, sans même avoir à fournir préalablement le programme avec les données brutes décrivant le réseau génétique: toutes les données nécessaires ont été automatiquement "extraites" *de la littérature*. Ce cas est particulier parce que, ici, les données elles-mêmes sont déjà entreposées dans un format exploitable par l'ordinateur, et elles ne sont pas structurée ad-hoc, même si, en fin de compte, les données (la littérature académique) proviennent réellement du travail des chercheurs humains. Mais un autre résultat sérieux surprenant est que le système de Wilkinson et Huberman a été capable de trouver des parties du réseau génétique impliquées dans le cancer du côlon, qui avait *échappé* l'attention (la capacité limitée de l'attention) des chercheurs humaines: la machine trouva une nouvelle modèle théorique, probablement impossible à atteindre par les humains, d'un phénomène! Maintenant, dans ce cas, le résultat obtenu est encore susceptible d'être exprimable sous une forme compréhensible, précisément parce qu'il est une description *modulaire* du système génétique sous observation. Mais que faire si un programme, par l'analyse de la littérature clinique, trouve un modèle modulaire qui regroupe en modules des données de nature hétérogène, d'une manière qu'il est peu probable tout être humain ne pourrait venir de concevoir spontanément? Ce modèle pourrait-il être encore compréhensible par des chercheurs humaines? Ou, qu'en est-il des cas dans lequels le modèle, bien que modulaire, est composé de centaines de modules de niveau intermédiaire sans aucune description modulaire de niveau supérieur capable de grouper ensemble certains d'entre eux? Considérons que, en raison de sa complexité computationnelle, l'algorithme de Wilkinson et Huberman est incapable de traiter des réseaux avec plus de quelques milliers de gènes. En raison de la complexité computationnelle excessive de l'algorithme de détection de la modularité de haut niveau, dans les cas de systèmes de taille suffisante nous devrions recourir à un modèle du phénomène qui n'est pas doté, à un niveau supérieur, de modularité visible. Un tel modèle, s'il est valide, pourrait être faisablement utilisé pour effectuer des *simulations* du phénomène observé. Mais il ne sera pas facile d'expliquer le phénomène au moyen du modèle, puisque la décomposition fonctionnelle du modèle a été impossible, et il sera probablement trop complexe pour être compris.

Mais, nous allons spéculer: pourrait-il être possible que le *phénomène* lui-même, decouvert par l'exploration algorithmique des données, finisse par être ni un phénomène *connu*, ni un phénomène facilement *compréhensible*? Qu'en est-il d'un phénomène complexe qui personne aurait même plausiblement considéré, et qu'il est difficile pour nous même de plausiblement décrire, ou de *nommer*? Même si au-delà de l'intuition spontanée humaine ou même de la compréhension humaine, il sera probablement le cas que des programmes informatiques ira découvrir ce genre de phénomènes. Mais, que sera-t-il de la science alors? Je pense que cela pourrait aller potentiellement plus profond: la tâche de détection de modularité a besoin d'une *relation entre les parties* du système, afin d'évaluer sa modularité, et cette relation et les parties sont donnés ensemble dans ce que j'ai appelé la *description préférée*: comme je l'ai soutenu ci-dessus[61], il est cette description préférée, avec les contraintes computationnelles sur la détection de la modularité, à déterminer l'"ontologie" du système sous observation. Mais, est-ce qu'il y a la possibilité de *changer* la description préférée de celle typique, "naturel", à une description "artificiel"? Compte

[60] Voir la section 3.2.1.
[61] section 18.5.1

tenu de la possibilité de description algorithmique des relations plutôt complexes entre les parties, alors même des espèces complexes, *non-naturelles* pourraient être détectées en tant que modules, et être faites objet de la science. Un "monde" bizarre, complètement différent, pourrait sortir de cette description. Une telle description non-standard pourrait même être hiérarchiquement décomposée de façon à être compréhensible, du moins en principe. Ou, elle pourrait donner lieu à une structure hiérarchique si complexe pour être utile pour l'explication seulement en principe: en raison de remappage, un nouvel ensemble de "espèces artificiels" pourrait émerger, et avec eux, des nouvelles disciplines. Bien sûr, cette modularisation doit être capable de détecter des modules suffisamment *robustes*, autrement elle ne constituerait pas une description modulaire *valide* du monde. En outre, on pourrait objecter que la causalité, la description naturelle commune du monde peuplé par des objets causalement cohésifs, est la seule description robuste possible. La question est ouverte.

Tout considéré, un recours généralisé à ce genre de remappage artificiel, ou à des simulations plus familières, comme celle décrites ci-dessus, pourrait certainement apporter dans certains domaines scientifiques une telle série d'innovations dans la méthode et les critères pour constituer un changement de paradigme dans la science, avec la possibilité de l'émergence de nouvelles disciplines scientifiques. Un inconvénient est que l'on devrait abandonner la perspective de la science comme un chemin vers une compréhension progressivement meilleure du monde: la tendance serait vers une forme sans précédent d'"explication scientifique automatisée", peut-être inintelligible.

Ce qui précède est une formulation possible de la thèse historique que je voulais soutenir. Peut-être qu'elle exige une imagination excessive. Et, il est bien clair que je ne produis dans ce travail un support suffisamment solide pour cette affirmation. Ici, J'ai simplement essayé de préparer le terrain en proposant une série de définitions et d'argumenter en premier pour une propriété, l'*antimodularité*, qui, lorsqu'elle survînt dans des phénomènes réels, pourrait avoir comme conséquences des problèmes pour certains modèles d'explication scientifique et la nécessité de recourir à la simulation numérique. L'énoncé conditionnel ci-dessus manque de soutien pour la prémisse: il est probable que l'antimodularité se produise surtout dans certains phénomènes complexes, mais je n'ai pas montré qu'un tel type de phénomènes soit vraiment central et répandu dans la littérature scientifique aujourd'hui. Il aura à voir si cela est le cas.

Mais, je pense en outre qu'il doit être souligné que la necessité croissante de recourir, lorsque une modularité puisse être effectivement trouvée, à sa recherche via moyens *algorithmiques*, et ainsi de recourir à une forme d'explication favorisée par les progrès dans la puissance de calcul, pourrait elle-même favoriser l'intérêt vers des phénomènes particulièrement complexes, ou, dans certains cas, apporter même une chance de «voir» l'existence de phénomènes qui pourraient avoir complètement échappé l'attention de la recherche scientifique non-computationnelle: la découverte automatique de la structure modulaire, lorsque elle soit possible, pourrait produire descriptions modulaires de prime abord inintelligibles, si la machine est capable de grouper les pièces en modules en considérant des relations entre les parties du systeme, relations artificielles mais peut-être significatives, qui avaient été jusqu'alors invisible à la compréhension humaine. Et, à partir de ce moment, une tendance vers une science plus computationnelle, peut-être moins compréhensible, serait une tendance alimentée par rétroaction positive: une telle science, si ces explications sont également utilisées pour *guider* des recherches ultérieures, pourrait évoluer de manières obscures pour nous.

Chapter 19

Modularità, Antimodularità, Spiegazione: una panoramica introduttiva

In questo lavoro, rivolgo l'attenzione principalmente al concetto di *modularità gerarchica* nei sistemi complessi, al suo rilevamento algoritmico e al suo uso nello *spiegare* la struttura e il comportamento dinamico di tali sistemi mediante modelli modulari gerarchici. Specificamente, evidenzio la portatata pragmatica della modularità gerarchica sulla possibilità di *spiegazione scientifica* di sistemi complessi, cioè sistemi che possono essere descritti come composti di parti elementari discrete interrelate. Sottolineo che la modularità gerarchica deve essere considerata una nozione *relativa*, dipendente dalla scelta, da parte dell'osservatore, di una specifica *descrizione preferita* di base del sistema, che consiste in una rappresentazione del sistema come un insieme di parti atomiche interrelate. In tale tipo di descrizione la modularità fondamentalmente si manifesta come la possibilità di *decomporre* il sistema in sottosistemi riconoscibili, sufficientemente definiti e persistenti (i moduli) ognuno composto da parti che sono più fortemente correlate tra loro che rispetto a parti appartenenti ad altri moduli o all'ambiente esterno al modulo. In effetti, secondo questa concezione, la modularità gerarchica non concerne il sistema reale, fisico, di per sé, ma solo le sue possibili *descrizioni* e le possibili *descrizioni di descrizioni*. Le descrizioni sono modelli teorici di un sistema, e mi concentro qui sulle descrizioni modulari *dei modelli*, lasciando per lo più da parte la spinosa questione della relazione tra modello e fenomeno, cioè il rapporto tra il fenomeno empirico e la sua prima descrizione: questi problemi meritebbero certamente un trattamento distinto più approfondito che non può essere fornito qui, anche se, alla fine, la mia proposta toccherà in qualche modo anche questo tipo di questioni.

Dopo aver considerato le proprietà che definiscono la modularità gerarchica, mi concentro sui metodi algoritmici noti per il suo *rilevamento*, cioè, algoritmi che, dato un sistema complesso (sotto forma della sua *descrizione preferita*, la sua descrizione cioè come un insieme di diverse parti correlate), cercano di produrre una re-descrizione modulare gerarchica del sistema. Una volta rilevata, la modularità gerarchica sembra essere la caratteristica fondamentale della descrizione di un sistema che consente spiegazioni a più livelli, spiegazioni funzionali o meccanicistiche del sistema, le quali sono importanti forme di spiegazione, ampiamente utilizzate nella scienza attuale.

Lungo queste linee, mi concentro in seguito sulla proprietà opposta, l'assenza di modularità gerarchica, che io chiamo *antimodularità*, cercando di trarre le conseguenze della sua possibile

manifestazione in alcune descrizioni del sistema. L'antimodularità è una proprietà complessa, risultante da una serie di possibili circostanze, le cui caratteristiche principali sono quella di essere, come la modularità, dipendente dalla scelta da parte dell'osservatore di una descrizione di base preferita del sistema, ma anche, soprattutto, quella di essere dipendente da alcuni *vincoli computazionali* riguardanti possibili algoritmi utilizzati per il rilevamento della modularità: la maggior parte di questi algoritmi sono computazionalmente molto impegnativi, e ci sono anche risultati teorici sulla *intrattabilità* computazionale della ricerca della descrizione modulare ottimale di un sistema. Questa complessità computazionale ostacola inevitabilmente la ricerca di modularità in sistemi di interesse scientifico di sufficienti dimensioni. Propongo di chiamare l'effetto di questo ostacolo *emergenza antimodulare*, per analogia con alcune forme conosciute di emergenza computazionale. Concludo che l'emergenza antimodulare comporta (sotto determinate condizioni) l'*emergenza debole* (*weak emergence*) di Mark Bedau, che è un'altra forma di emergenza computazionale[1].

Dopo aver definito questo nuovo tipo di emergenza computazionale, cioè l'emergenza antimodulare, che è dovuta all'eccessiva complessità computazionale che algoritmi per la rivelazione della modularità possono manifestare in taluni casi, provo a trarre alcune possibili conseguenze dell'emergenza antimodulare sulla possibilità di *spiegazione scientifica* dei sistemi affetti da essa.

Prendo in considerazione tre modelli classici di spiegazione scientifica: deduttivo-nomologica, meccanicistica, e spiegazione computazionale, oltre a un nuovo modello, la spiegazione topologica, recentemente proposto da Philippe Huneman. Concludo che l'emergenza antimodulare colpisce la fattibilità di tutti questi tipi di spiegazione, anche se in modi diversi.

Prima di tutto, sostengo che l'antimodularità influisce negativamente sulla spiegazione meccanicistica, una forma fondamentale di spiegazione nelle scienze biologiche. Schierandomi con Cory Wright e William Bechtel per una concezione *epistemica* della spiegazione meccanicistica (opposta ad una concezione *ontica*), mostro come l'antimodularità costringa a spiegazioni basate su descrizioni ad un solo livello, un livello di descrizione basso, obbligando a trascurare l'esigenza, essenziale per le spiegazioni meccanicistiche, dell'integrazione multi-livello. Il fatto di limitare la spiegazione meccanicistica al livello di descrizione che rappresenta le parti più elementari del sistema ostacola certamente la comprensione: per sistemi abbastanza ampi, la loro spiegazione meccanicistica a questo livello è troppo complessa per essere compresa da esseri umani. E la comprensibilità è una qualità da ricercare nel spiegazione meccanicistica, almeno secondo William Bechtel ed altri autori, come Petri Ylikoski, il quale considera la "salienza cognitiva" una delle caratteristiche importanti delle spiegazioni.

Per quanto riguarda le classiche spiegazioni nomologico-deduttive (*DN* d'ora in poi) , evidenzio che, dal momento che, in sistemi abbastanza complessi[2], *l'antimodularità comporta l'emergenza debole*, questo significa che non si può ricorrere alla spiegazione deduttivo-nomologica à la Hempel per un sistema antimodulare, perché, se si potesse, vorrebbe dire che il sistema è prevedibile mediante una legge, e questo è negato dalla definizione stessa di emergenza debole, che, come detto, è implicata dall'antimodularità. Così, un sistema antimodulare complesso non è prevedibile, almeno non prevedibile arbitrariamente in anticipo mediante una legge analitica, e, quindi, non può essere spiegato da una spiegazione DN. In ogni caso, se prendiamo in considerazione uno specifico tipo di sistema dinamico complesso, vale a dire un automa cellulare (*CA* d'ora in poi),

[1] Qui e nel seguito userò il termine "emergenza" per tradurre il termine inglese "emergence". Non si tratta di una traduzione priva di problemi, perché, anche se ormai diffuso in ambito filosofico come traduzione di "emergence", il termine "emergenza" in Italiano è chiaramente molto più spesso usato per veicolare il significato di stato di urgenza e pericolo, che corrisponde alla traduzione del termine inglese "emergency". Si tenga quindi conto di questo potenziale equivoco.

[2] Una condizione che viene chiarita nella sezione 13.3.

un processo antimodulare generato da un CA può essere spiegato, in un certo senso, producendo un elenco potenzialmente molto lungo di deduzioni basato sulle premesse della condizione iniziale del CA e della sua regola (che, come regola CA, per il teorema di Curtis-Hedlund-Lyndon, ha la stessa forma logica di una *legge scientifica*), in modo simile ad un lungo elenco di spiegazioni DN passo-passo. In questo caso, la comprensione umana è impedita dalla possibile lunghezza della lista, ma, se ci atteniamo alle aspettative dei sostenitori post-neopositivisti del modello DN di spiegazione, la comprensione non è necessaria per una buona spiegazione. Quindi, in un certo senso, l'antimodularità e, di conseguenza, l'emergenza debole, non ostacolano la spiegazione DN, almeno nel caso dei CA e di altri sistemi la cui dinamica segue una regola *universale* legisimile, e finché il problema della comprensibilità è ignorato.

Procedo a prendere in considerazione la spiegazione topologica di Philippe Huneman, un tipo di spiegazione non meccanicistica che si basa su proprietà topologiche di certe descrizioni astratte di un sistema. Concludo che, essendo la modularità essa stessa una proprietà topologica, così come la sua assenza, la presenza o l'assenza di modularità non ostacolano, ma *consentono* la spiegazione topologica.

Mi concentro quindi su un terzo possibile tipo di spiegazione: CA e le reti booleane dinamiche possono essere considerati *sistemi computazionali*. Come tali, essi possono essere soggetti a *spiegazione computazionale*. Considero il caso di cercare di spiegare computazionalmente un CA. Per ottenere una spiegazione computazionale, prima il comportamento del CA deve essere *visto* come una computazione. Condivido una concezione *intenzionale* della computazione, soggetta però ad alcuni vincoli matematici, e cerco di delimitare il campo della dinamica dei sistemi che possono essere visti come computazionali. Dato che alcuni CA possono in effetti essere visti come computazionali, cerco di valutare la possibilità di una loro spiegazione computazionale. Così com'è, per dare una tale spiegazione, il comportamento del CA deve essere retro-ingegnerizzato al fine di ottenere la *specifica* del calcolo che dovrebbe eseguire. Ma, questo compito di *estrazione delle specifiche* (*specification mining*) può essere computazionalmente difficile, e così può fallire. Anche se viene trovata una specifica globale, una buona spiegazione computazionale costituisce una forma di analisi funzionale modulare gerarchica, e questo si ottiene estraendo ricorsivamente le specifiche di parti del codice o sottoponendo il sistema ad altri metodi per reverse-engineering statico o dinamico. Se questo processo non riesce per motivi di complessità computazionale eccessiva dell'algoritmo di estrazione delle specifiche, o per mancanza di completezza della gerarchia funzionale trovata, il sistema finisce per risultare antimodulare. In questo caso, l'antimodularità ostacolerebbe una forma comprensibile di spiegazione computazionale, per la stessa ragione che essa colpisce la spiegazione meccanicistica, con cui la spiegazione computazionale, che è una forma di analisi funzionale, mostra una stretta affinità.

Successivamente sottolineo la necessità della modularità di alto livello gerarchico, e non solo per spiegazioni *a posteriori* di un fenomeno noto, ma anche durante la fase della scoperta scientifica, in particolare, come già notato da James Woodward, durante la ricerca di relazioni causali tra parti di un meccanismo sia a basso livello che ad un livello superiore. Analogamente Sottolineo che la spiegazione modulare multilivello è ugualmente essenziale durante lo sviluppo di programmi informatici da parte dei programmatori umani.

Sussumo sotto il concetto di *emergenza esplicativa* tutti i risultati sulla impossibilità pratica, dovuta a vincoli computazionali, di ottenere certe spiegazioni multilivello e sul conseguente affievolimento della comprensibilità dovuto al verificarsi di antimodularità, nonché qualsiasi altro caso in cui un sistema, per motivi computazionali, risulta refrattario alle spiegazioni comprensibili.

Discuto poi, esaminando materiale tratto della letteratura scientifica, la probabilità che la ricerca

scientifica in certe aree si imbatta effettivamente nell'emergenza antimodulare, concludendo che è abbastanza probabile che alcuni casi di antimodularità appaiano, in particolare in biologia dei sistemi.

Dedico la parte finale di questa introduzione a considerazioni più ampie e, probabilmente, rischiose perché meno corroborate. In primo luogo, abbozzo una possibile visione metafisica che potrebbe derivare dalle considerazioni circa l'antimodularity esposte in precedenza: chiamo questo punto di vista *antirealismo vincolato*. Questa concezione vede il mondo empirico che naturalmente percepiamo, così come il mondo descritto dalla scienza, come risultato di un processo di rilevamento della modularità, in conseguenza del quale i moduli rilevati costituiscono quelli che sono comunemente definiti come *generi naturali*. Dato che il rilevamento modularità è vincolato da fattori di insormontabile complessità computazionale, e che per questo motivo il reperimento della migliore descrizione modulare in linea di principio è escluso, non è probabile che suddivisione del mondo in tipi naturali corrisponda alla migliore suddivisione possibile. La valutazione completa di questa posizione metafisica richiede tuttavia una vasta discussione di un'ipotesi controversa, il *pancomputazionalismo*, e di varie posizioni in filosofia della matematica, una discussione che è meglio lasciare ad un lavoro successivo.

Infine, mi concedo al termine di questa introduzione una certa libertà nel trarre alcune possibili, presunte conseguenze sulla storia della scienza di un ricorso recente e crescente ai metodi computazionali nelle scienze, a partire dalla simulazione di sistemi complessi: rifletto sulla plausibilità delle simulazioni come spiegazioni, soprattutto nei casi in cui il sistema è antimodulare, e di conseguenza la simulazione può essere eseguita, ma il modello dinamico di fondo è incomprensibile, perché il sistema viene simulato a un livello molto basso e una ridescrizione modulare, di alto livello, manca o è troppo carente. Considero poi il rilevamento automatizzato di modularità, utilizzato per trovare struttura in grandi insiemi di dati, sottolineando un caso reale di data mining su un corpus di letteratura medico-biologica, in cui l'automatismo ha scoperto importanti relazioni funzionali che erano sfuggite all'esame umano. Permettendomi di trarre alcune conseguenze forse estreme, concludo suggerendo che questo crescente uso di metodi computazionali nelle scienze potrebbe essere sul punto di provocare un importante cambiamento di paradigma in qualche disciplina.

L'oggetto di questa tesi è multiforme e non facile da etichettare: vertendo sulle conseguenze dell'antimodularità, una proprietà di alcuni sistemi, sulla possibile spiegazione scientifica di essi, si tratta di un'opera di filosofia della scienza. Dato che la proprietà proposta, l'antimodularità, dipende da alcuni limiti di calcolo che interessano il rilevamento della modularità, e che sostengo, in relazione con la discussione sulla spiegazione computazionale, una concezione intenzionale della nozione di computazione, allora questo è un lavoro di filosofia della computazione, inteso nel duplice senso di rendere certe nozioni riguardanti la computazione nozioni utili per la riflessione filosofica, e di proporre una riflessione filosofica sul concetto di computazione in sé. Considerata la discussione di lunga data nel campo della biologia sulla modularità e il fatto che dovrebbe essere probabile trovare sistemi che mostrano antimodularità tra i sistemi biologici, nonché e una serie di esempi che riporto da quella disciplina, il presente lavoro è anche un lavoro di filosofia della biologia. Per quanto riguarda la spiegazione scientifica, abbraccio esplicitamente una posizione epistemica, centrata sulla nozione di *livelli di descrizione*, che sono dispositivi epistemici, e dunque questo lavoro ha un aspetto epistemologico. E, come probabilmente ogni posizione epistemologica, la posizione suddetta ha anche un portato metafisico, che cerco di delineare verso la fine di questa introduzione. Infine, questa dissertazione si avvale di tutte le discussioni teoriche di cui sopra per far luce sulle loro possibili conseguenze sulla pratica della scienza, ventilando la possibilità che un grande cambiamento storico, forse un cambio di paradigma, sia sul punto di realizzarsi nella scienza, se non è già avvenuto. Quindi, in un

certo senso questa è una tesi di storia della scienza. Anche se ancora richiede osservazioni e prove, credo che questa ipotesi storica potrebbe darci un indizio della portata degli effetti che l'adozione diffusa di metodi computazionali ha avuto o potrebbe essere in procinto di avere sulla scienza come la conosciamo.[3]

19.1 Modularità

Intendo iniziare questa chiarificazione del il concetto di modularità con uno schizzo storico: la modularità sembra essere un concetto di base e diffuso, che è stato concepito da lungo tempo, probabilmente più di una volta, in campi teorici e pratici parzialmente indipendenti e diversificati. Tuttavia, la riflessione filosofica moderna su di essa è iniziata nella seconda metà del XX secolo, con il contributo particolarmente rilevante di Herbert Simon. Lavorando nel campo dell'econometria, egli giunse a una concezione di modularità sotto forma di sistemi *quasi- scomponibili* gerarchici[4], vale a dire sistemi che possono essere visti, almeno in prima approssimazione , come gerarchicamente scomponibili in modo ricorsivo in sottosistemi robusti parzialmente indipendenti. Questo punto di vista sulla quasi-scomponibilità, che ha successivamente influenzato molti altri autori in diversi campi, è l'idea di fondo che ispira la mia proposta sulla modularità.

In questo lavoro, esamino una possibile concezione della modularità nei sistemi complessi, ed esploro le conseguenze della presenza di modularità o della sua assenza (antimodularità) sulla spiegazione del comportamento di tali sistemi. In effetti, non applico il concetto di modularità ai sistemi fisici, reali, bensì alle loro *descrizioni*, e a re-descrizioni di descrizioni, dove una (re)descrizione è intesa, preferenzialmente, come una computazione che prende una descrizione e la elabora per produrre un'altra descrizione. Abbracciando una posizione ampiamente epistemica, se non pienamente antirealistica (una posizione che sarà meglio esposta nella sezione 19.5.1 di questa introduzione), lungo le linee della concezione *epistemica* di Cory Wright e di William Bechtel riguardo alle spiegazioni meccanicistiche[5], ritengo le spiegazioni scientifiche dei dispositivi *epistemici*, basati su *descrizioni* di fenomeni, legate alla *comunicazione* umana , e richiedenti almeno un grado minimo di *intelligibilità* cognitiva. Di conseguenza, sono interessato a definire la modularità come una caratteristica *delle descrizioni*, che, se presente, consente alcuni tipi comprensivi di spiegazione. Mentre la sezione II è dedicata ad una esposizione approfondita della modularità e di altri concetti coinvolti, darò qui una spiegazione schematica di ciò che vorrei proporre.

19.1.1 La modularità nei sistemi complessi

Procedendo lungo le linee sopra esposte, provo a definire la proprietà della *modularità* nei sistemi complessi, come la possibilità per un sistema di questo tipo di essere descritto come un insieme di *moduli* collegati debolmente tra loro, cioè un insieme di sottosistemi robusti ben definiti, con parti interne altamente interconnesse, dove ogni sottosistema è parzialmente indipendente dal contesto esterno, essendo collegato solo debolmente ad altri sottosistemi. Estendo questa concezione della modularità a quella della piena *descrizione gerarchica* di un sistema in termini

[3] Un'analisi approfondita della nozione di modularità gerarchica e di antimodularità è condotte nei principali capitoli da della dissertazione. Intendo fornire nel seguito di questa introduzione una panoramica più succinta dei principali contenuti del lavoro. Avverto il lettore che, nel seguito, farò spesso uso dei termini "modularità" e "sistema" invece di "modularità gerarchica" e "sistema complesso", lasciando al contesto il compito di disambiguare il loro significato.

[4] Vedi il seminale Simon (1962).

[5] Opposta ad una concezione *ontica* delle spiegazioni causali. Vedi la sezione 19.1.9, 19.4.3, così come Bechtel & Abrahamsen (2005) e Wright (2012).

di livelli "superiori"e "inferiori" di descrizione, ognuno dei quali è costituito da moduli, e in cui, ad eccezione del livello più basso, ognuno dei moduli a un certo livello è un *macromodulo*, cioè, può a sua volta essere visto come caratterizzato internamente da una organizzazione modulare di *micromoduli*, e così via ricorsivamente. Come detto, tutto questo riguarda descrizioni, non insiemi di oggetti del mondo reale (e questo è in linea con l'essenza di una concezione epistemica).

Mentre la distinzione macromodulare/micromodulare è ovviamente dipendente dalla scelta di un particolare livello di descrizione, il punto da evidenziare è che tutta la descrizione modulare gerarchica risulta dipendere, per la definizione stessa di modularità, dalla *scelta* da parte dell'osservatore di una specifica *relazione* significativa tra le parti elementari del sistema, e ciò proprio per il modo in cui il concetto di *modulo* è definito: un modulo è un sottoinsieme delle parti di un tutto che sono *relate* l'una all'altra in un modo più forte di quanto esse siano relate a parti esterne al modulo cui appartengono. Il riconoscimento di un sottoinsieme come modulo richiede pertanto che una *relazione* tra parti venga prima di tutto presa in considerazione, e, a seconda di quale specifica relazione è considerata, la struttura modulare identificabile può cambiare.

Questa definizione di modularità gerarchica naturalmente presuppone che un sistema complesso sia composto di parti elementari distinte e correlate, e questo a sua volta è dovuto alla scelta di una descrizione elementare atomica del sistema: la scelta dell'insieme delle *parti* e quella della *relazione* che sussiste tra loro, equivale alla scelta, da parte dell'osservatore, secondo i suoi interessi, di quello che chiamerei una *descrizione preferita* del sistema. Generalmente, vi è una descrizione "naturale" di livello più basso di un sistema in termini di parti elementari, spesso suggerita dalle proprietà fisiche del sistema combinate con gli interessi del ricercatore: per esempio, in biologia un tessuto è naturalmente descritto come composto di cellule, una cellula è naturalmente descritta come un sistema complesso composto principalmente di macromolecole interagenti, mentre nelle scienze sociali una società è naturalmente descritta come composta di individui. Il punto da sottolineare è che la modularità gerarchica è *relativa* a tale scelta, e dipende soprattutto della scelta della *relazione* tra parti elementari del sistema, che di solito è una scelta meno vincolata rispetto a quella delle parti stesse. Ad esempio, in una società possiamo considerare legami affettivi tra gli individui, o, in alternativa, potremmo scegliere rapporti di subordinazione. Queste due diverse descrizioni del sistema molto probabilmente determinerebbero differenti descrizioni modulari gerarchiche, perché un modulo è definito come un sottosistema di elementi altamente interconnessi debolmente connessi con l'ambiente circostante, e questa "connessione" è appunto la relazione tra le parti elementari considerata nella descrizione scelta preferita del sistema: nei casi di esempio, una relazione è il rapporto di legame affettivo, l'altra quello di potere.

19.1.2 Modularità, scomponibilità ed economia di descrizione

La modularità si manifesta come la possibilità di *scomporre* un sistema[6] in sottosistemi riconoscibili, sufficientemente definiti, ognuno composto da parti che sono più fortemente collegate tra loro che a parti appartenenti ad altri moduli o all'ambiente esterno. È la presenza di queste variazioni nella forza delle relazioni vigenti tra coppie di parti del sistema, ciò che consente il riconoscimento di modularità: se tutte le parti fossero pienamente collegate tra loro, i moduli non comparirebbero, precisamente perché un modulo è (informalmente) definito come un sottosistema i cui collegamenti con il resto del sistema hanno forza *inferiore* (in media) rispetto a quella dei collegamenti tra le parti interne del modulo. Come notato sopra, la modularità risultante è relativa alla specifica relazione tra le parti di basso livello che stiamo prendendo

[6] Naturalmente, con "sistema" qui intendo una sua *descrizione*. In ciò che segue, spesso userò il termine "sistema" simpliciter per indicare la sua descrizione standard, di solito la sua "descrizione preferita".

in considerazione. Questo è un concetto molto simile a quello originale di Herbert Simon, il concetto della *quasi-scomponibilità*. La quasi-scomponibilità permette al sistema originale di essere rappresentato come un insieme di sottosistemi connessi, e questa decomposizione può essere ripetuta sino al conseguimento di una descrizione gerarchica completa. Il punto cruciale è che il sistema originale, costituito dalle sue parti elementari, è quindi descrivibile in una maniera di alto livello, sotto forma di un *altro* sistema le cui parti corrispondono ciascuna ad uno dei moduli del sistema originale. Quindi, la descrizione di alto livello risulta essere *più semplice* rispetto a quella di basso livello, perché, nella prima, interi *gruppi* (i moduli) di parti di basso livello sono rappresentati come *singole* parti di alto livello, e così le parti del livello superiore sono in numero inferiore rispetto a di quelle di basso livello. Se il sistema che stiamo descrivendo in questo modo è statico, come per esempio l'elenco dei membri che compongono il personale di un'organizzazione, la descrizione di alto livello di solito appare più economica e comprensibile rispetto all'elenco originale. L'esempio tipico è quello degli organigrammi. In un organigramma, ogni gruppo di persone che lavorano nello stesso ufficio è rappresentato da un singolo elemento grafico, etichettato con il nome dell'ufficio. Il nome dell'ufficio rappresenta il nome aggregato del gruppo di persone che lavorano in quell'ufficio.

19.1.3 Moduli come parti similari di alto livello ripetute

C'è un ulteriore possibile miglioramento dell'economia di descrizione di un sistema complesso se è possibile rilevare in esso più sottosistemi che risultano essere identici o talmente simili da poter essere considerati come la ripetizione di un singolo modello. In questo caso, a parte l'economia di descrizione dovuta all'aggregazione[7], anche la descrizione *modulare*, qualora essa comprenda più moduli identici, può essere semplificata sostituendo ogni occorrenza di questi moduli con un riferimento al modello comune, che è quindi necessario descrivere solo una volta. Questa forma di modularità è particolarmente utile in ingegneria, ed è sostanzialmente alla base della progettazione di artefatti complessi, che sono generalmente costituiti da parti *standard* identiche o quasi identiche, che compaiono nel sistema in più copie.

19.1.4 Modularità strutturale e modularità dinamica

È facilmente immaginabile che la modularità possa riguardare non solo la *struttura* di un sistema, ma anche il suo funzionamento dinamico: è concepibile, per esempio, e anche ovvio, che la modularità nella struttura di un programma per elaboratore (la cui struttura è una lista di istruzioni) dia luogo a una modularità nella sua esecuzione dinamica, perché un programma per elaboratore non è solo una lista di istruzioni statiche, ma si suppone che venga *eseguito*, quindi la modularità della lista di istruzioni dovrebbe essere riflessa nella modularità dinamica del programma.

Considerando la relazione tra modularità strutturale e modularità dinamica, questa risulta non essere sempre una relazione semplice: gli aspetti strutturali e gli aspetti dinamici possono essere associati ma anche disgiunti, sebbene nella maggior parte dei casi di sistemi dinamici la loro struttura fisica modulare *induca* una forma di funzionamento modulare dinamico, conseguenza del fatto che in sistemi dinamici la loro dinamica è condotta *sulla struttura* predefinita del sistema, ed è quindi vincolata da essa. La relazione tra modularità strutturale e modularità dinamica non è però completamente limpida e la sezione 6 ne discute più profondamente.

[7] Menzionata nella sezione precedente.

19.1.5 Quasi-scomponibilità e aggregabilità

Una forma di modularità dinamica proposta a partire dai primi anni '60 da Herbert Simon e Albert Ando[8] deriva dalla quasi-scomponibilità del sistema, e in particolare dalla quasi-scomponibilità del *modello matematico* che descrive la dinamica del sistema. Il modello matematico è di solito una relazione di ricorrenza, o un sistema di relazioni di ricorrenza, in cui lo stato di ciascuna parte elementare del sistema è rappresentato da una variabile: questa equazione rappresenta una funzione di aggiornamento, con il tempo come variabile indipendente, che determina come lo stato delle parti del sistema cambi con lo scorrere del tempo, e quindi è un modello matematico della dinamica del sistema. In un sistema che è quasi-scomponibile nel senso inteso da Simon, le variabili di questa equazione, che possono essere in gran numero perché rappresentano le parti interagenti elementari del sistema, possono essere anch'esse suddivise (modulo una certa approssimazione) in una partizione di sottoinsiemi di variabili, ogni sottoinsieme contenente variabili che influenzano solo debolmente variabili all'interno di altri sottoinsiemi: questo corrisponde al fatto che in un sistema quasi-scomponibile, per definizione, le interazioni tra alcuni gruppi di parti (i moduli) sono solo deboli. In questo modo, la dinamica di ciascun modulo può essere considerata come una dinamica che evolve nel tempo in modo semi-indipendente dalle dinamiche degli altri moduli, e, di conseguenza, le equazioni che descrivono queste dinamiche semi-indipendenti si rivelano semi-indipendenti l'una dall'altra. Queste equazioni che governano gruppi di variabili semi-indipendenti possono essere considerate *moduli funzionali*, una re-descrizione modulare del modello matematico originale descrivente la dinamica globale del sistema. Nei sistemi quasi scomponibili, la loro modularità determina anche una sorta di modularità dinamica o *di processo*, sotto la forma di un disaccoppiamento della dinamica temporale tra parti del sistema: la dinamica *all'interno dei moduli* è più veloce della dinamica delle interazioni *tra* moduli.

Date le condizioni di cui sopra, in alcuni casi favorevoli che dipendono dalla forma delle equazioni modulari, la dinamica globale del sistema, originariamente descritta dalla funzione di aggiornamento globale, in cui lo stato di ogni parte elementare è descritto da una singola variabile, può essere, modulo una certa approssimazione accettata, ulteriormente re-descritta sotto forma di un'altra funzione di aggiornamento globale *più semplice*. Questa funzione di aggiornamento è più semplice di quella originale perché nella nuova funzione di aggiornamento ogni variabile rappresenta un *valore aggregato* di tutte le variabili contenute in ciascuno dei moduli funzionali sopra descritti: il numero di variabili che devono essere prese in considerazione per modellizzare le dinamiche globali del sistema viene in questo modo ridotto. Quando questa condizione sussiste (non ogni sistema dinamico è aggregabile), il sistema è detto *aggregabile*, e questa è evidentemente un'altra forma di *economia di descrizione*, in questo caso economia del modello matematico, consentita dalla presenza di modularità. Il prezzo da pagare è una quantità di approssimazione che dipende dal fatto che, al fine di aggregare correttamente le dinamiche, alcune interazioni tra le parti del sistema la cui forza scende al di sotto di una soglia prescelta sono considerate nulle. L'approssimazione potrebbe risultare inaccettabile in sistemi non lineari, in cui il comportamento a lungo termine della descrizione semplificata potrebbe divergere troppo dal comportamento reale del sistema. Il punto da sottolineare è che, anche qui, sono coinvolte scelte da parte dell'osservatore: la scelta della descrizione preferita (che, tuttavia, in molti casi, è già data), e una scelta sull'approssimazione accettabile o meno a seconda degli obiettivi dell'osservatore.

Un problema molto importante che riguarda l'aggregazione è che questa si è rivelata essere un compito computazionalmente intrattabile: ci sono dimostrazioni, in Kreinovich & Shpak (2006), e Kreinovich & Shpak (2008), che l'aggregabilità, e perfino l'aggregabilità *approssimativa*, già in

[8] Simon & Ando (1961).

sistemi *lineari*, è *NP-hard*. Questo significa[9] che non vi è alcun metodo generale algoritmico che, applicato ad un modello matematico della dinamica del sistema, possa sempre produrre in un tempo ragionevole una versione aggregata semplificata plausibile del modello, per i modelli con un numero sufficientemente grande di variabili. In altri termini, ciò significa che il rilevamento della modularità nel modello dinamico di un sistema complesso è un compito computazionalmente impraticabile, e che quindi *non ci si deve aspettare, in generale, che la modularità dinamica possa essere trovata con un metodo generale.*

Tuttavia, l'aggregazione può in molti casi essere trovata più facilmente se abbiamo qualche conoscenza a priori che ci può guidare nel partizionamento delle variabili in sottoinsiemi semi-indipendenti. Ad esempio, nel caso di reti genetiche, potremmo sapere su basi empiriche che qualche gruppo di geni co-esprime sempre, e così le variabili che rappresentano questi geni possono essere raggruppate. Questo potrebbe semplificare molto il compito di trovare una buona aggregazione, un compito che in linea di principio, come detto, è troppo esigente.

19.1.6 La modularità nei sistemi discreti dinamici

Ci sono casi complessi in cui le forme di modularità strutturale e dinamica non sono facilmente separabili, perché qualche struttura ad alto livello del sistema, in sé, "emerge"[10] dalle complesse dinamiche di basso livello del sistema. Questo è tipico di certi *sistemi dinamici discreti* complessi, come ad esempio alcune reti booleane, o certi automi cellulari. Mentre dedico alcune sezioni del capitolo 5 a spiegare le basi dei sistemi dinamici discreti, e più precisamente di una loro sottoclasse, i cosiddetti *automi cellulari* (*CA* da ora in poi), una panoramica molto sintetica può essere qui prodotta: tali sistemi sono costituiti da un gran numero di parti semplici, ciascuna delle quali, in un dato momento, appare essere in un particolare *stato*, scelto all'interno di un insieme finito di stati possibili distinti. Si è soliti pensare a ogni stato distinto come a un *simbolo*, e considerare l'insieme dei possibili simboli come un *alfabeto* (si pensi, nel caso più semplice, ai simboli *0* e *1*). Non solo i simboli sono discreti, ma lo è anche il tempo: in questi sistemi discreti il tempo procede per passi di tempo (*timesteps*) distinti, che possiamo chiamare t_1, t_2, e così via. In qualsiasi momento, l'insieme degli stati in cui tutte le parti del sistema si trovano essere, costituisce la *configurazione* globale del sistema. Gli stati di tutte le parti del sistema sono aggiornati in modo sincrono ad ogni passo temporale successivo secondo una determinata regola deterministica, una regola che può essere la stessa per tutte le parti del sistema (come è nel caso dei CA), o diversa per ciascuna parte. In un momento iniziale convenzionale, chiamiamolo t_0, il sistema è nella *configurazione iniziale*. L'evoluzione del sistema è la sequenza di configurazioni globali successive che il sistema raggiunge col passare del tempo, a partire dalla configurazione iniziale. Classi tipici di tali sistemi sono, come detto, i CA, e una classe più ampia, quella delle reti booleane. La dinamica di tale evoluzione può, per certi sistemi, essere estremamente complessa, in alcuni casi palesemente equivalente alla potenza di calcolo delle macchine di Turing universali, che si ritengono essere[11] la classe più potente di sistemi computazionali. Per questo motivo, l'andamento nel tempo dei sistemi complessi è, in generale, molto difficile da prevedere, e, nel caso di capacità di livello Turing, è in linea di principio *algorithmicamente indecidibile* in generale[12].

Una forma di modularità può essere indotta o apparire in certi sistemi dinamici discreti, sia imponendo loro un stato iniziale specifico, o, in alcuni casi, per via della sua comparsa spontanea

[9] Vedi sezione 19.3.

[10] Uso il termine "emergenza" qui in modo intuitivo, anche se esso sarà discusso brevemente in seguito e, più profondamente, nel corpo della dissertazione.

[11] Data la tesi di Church-Turing per scontata. Per una spiegazione, si veda l'Appendice, sezione 17.3.

[12] Come conseguenza della indecidibilità del problema della fermata. Vedere la sezione 17.2.6.

nel sistema dopo un certo tempo lungo la sua evoluzione, indipendentemente dalla specifica configurazione iniziale: un fenomeno che è una forma di *auto-organizzazione*. Modularità in questo senso equivale al fatto che alcuni sottoinsiemi della configurazione globale del sistema vengono a risultare parzialmente o totalmente *congelati* dopo un certo tempo, cioè, vengono a costituire parti immutabili o poco modificabili della configurazione, in modo da isolare parzialmente altri sottoinsiemi della configurazione, ostacolando la propagazione di influenze da ciascuno di questi sottoinsiemi agli altri. Ciò, implicitamente, impone una struttura virtuale di alto livello sopra la struttura di basso livello originale del sistema, una sovrastruttura che può essere vista come un insieme di moduli dinamici (le parti non fisse della configurazione) debolmente collegati tra loro (mediante i residui percorsi di connessione che non siano interrotti da parti congelate). Per un esempio di una rete discreta con modularità di alto livello che emerge durante la sua evoluzione, vedi fig. 19.1.

In un altro modo leggermente diverso, l'auto-organizzazione può apparire, soprattutto nei CA, come l'emergere di sottoconfigurazioni altamente localizzate, parzialmente robuste, ben delimitate, solo in parte cangianti, della configurazione globale, i cosiddetti *gliders*, che sembrano, per così dire, *viaggiare* attraverso la configurazione globale del sistema. Un esempio è in fig. 19.2.

19.1.7 Modularità nei sistemi computazionali

Essendo una forma di sistema dinamico discreto, un sistema computazionale può ovviamente mostrare modularità. I computer universali comuni del mondo reale sono macchine altamente modulari già al cosiddetto livello "hardware". Ma un'altra forma molto importante di modularità riguarda i *programmi* per computer. Un programma è essenzialmente costituito da una lista di istruzioni che l'hardware del computer "esegue" passo dopo passo. Naturalmente, tale elenco può essere privo di modularità apparente, o può invece essere strutturato dal programmatore in modo evidentemente modulare, suddividendolo in sottoliste disgiunte, ciascuna dei quali contiene principalmente istruzioni relative solo a un insieme limitato di variabili interne alla sottolista, ad eccezione di un insieme di "ingresso" (o "input") e un insieme di "uscita" (o "output") di variabili alle quali si accede anche da istruzioni comprese in altre sottoliste. In questo modo, ciascuna di queste sottoliste può essere considerata un modulo, e il trasferimento limitato e controllato di informazioni tra i diversi moduli è realizzato dalle variabili di ingresso-uscita, che sono gruppi separati di variabili che risultano gli unici ad essere accessibili e gestibili dalle parti del programma esterne al modulo: un tale tipo di modulo può essere considerato una "scatola nera" con un insieme limitato di linee di ingresso e di uscita. In questo modo, la struttura tipica dei moduli è realizzata: considerando come relazione prescelta tra le parti della lista di istruzioni il rapporto tra un'istruzione e la variabile su cui essa agisce, si può facilmente vedere che una sottolista di istruzioni le cui variabili interne, quelle non comprese negli insiemi di ingresso e uscita, sono di uso prevalentemente interno, e sono meno spesso o (meglio) non sono mai manipolate da istruzioni appartenenti a sottoliste disgiunte esterne, può essere considerata un modulo, dotato di coesione interna e strutturalmente piuttosto indipendente dagli altri moduli. In un programma non modulare, una modifica del programma da parte del programmatore può risultare molto difficile da realizzare e da gestire, perché un cambiamento in una parte del programma potrebbe influenzare parti potenzialmente molto distanti. Per contro, data la connettività limitata tra i moduli, soprattutto nel caso in cui solo le variabili di ingresso e uscita siano accessibili a moduli esterni, un cambiamento interno al modulo, e che riguarda variabili solo interne, non si diffonde indiscriminatamente ad altri moduli, e quindi i suoi effetti sono più facili da controllare. In generale, nella programmazione dei computer, ciò che si cerca è *alta coesione* interna dei moduli e *debole accoppiamento* tra di loro.

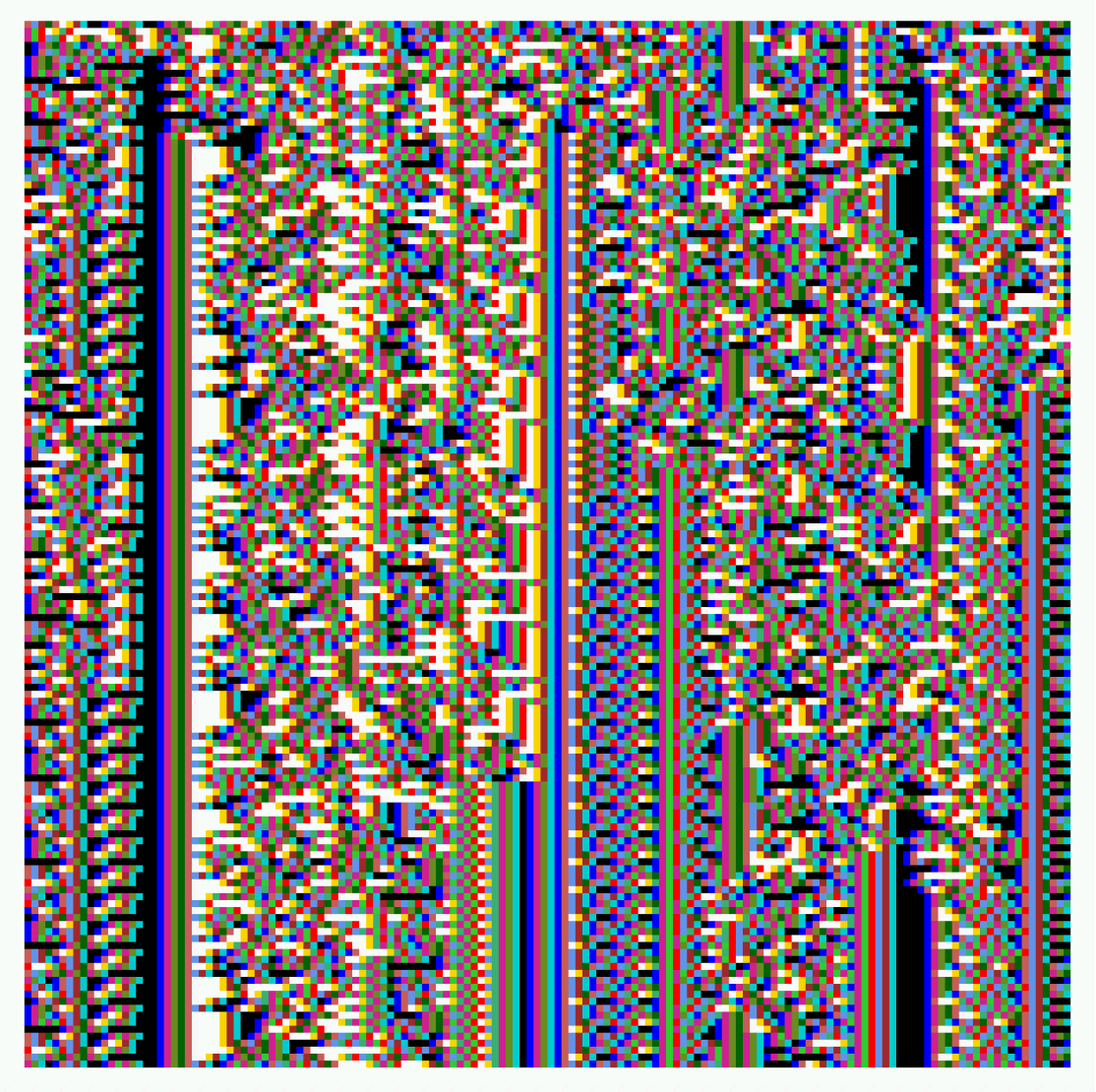

Figure 19.1: un'evoluzione parziale nel tempo di una rete discreta. Il tempo scorre dall'alto verso il basso, ciascuna riga di pixel rappresenta la configurazione del sistema ad ogni passo temporale. Ogni pixel rappresenta lo stato di una delle parti elementari della rete, i suoi *nodi*. Le linee verticali spesse, nero o a motivi regolari, che si distinguono nell'immagine, sono sottoinsiemi "congelati" della configurazione. Essi inducono una forma modularità di alto livello, agendo come più o meno come "muri" impenetrabili, rendendo in questo modo il sistema quasi-scomponibile in diversi sottosistemi indipendenti. (Immagine tratta dalla Galleria DDLab di Andrew Wuensche, http://uncomp.uwe.ac.uk/wuensche/gallery/ddlab_gallery.html).

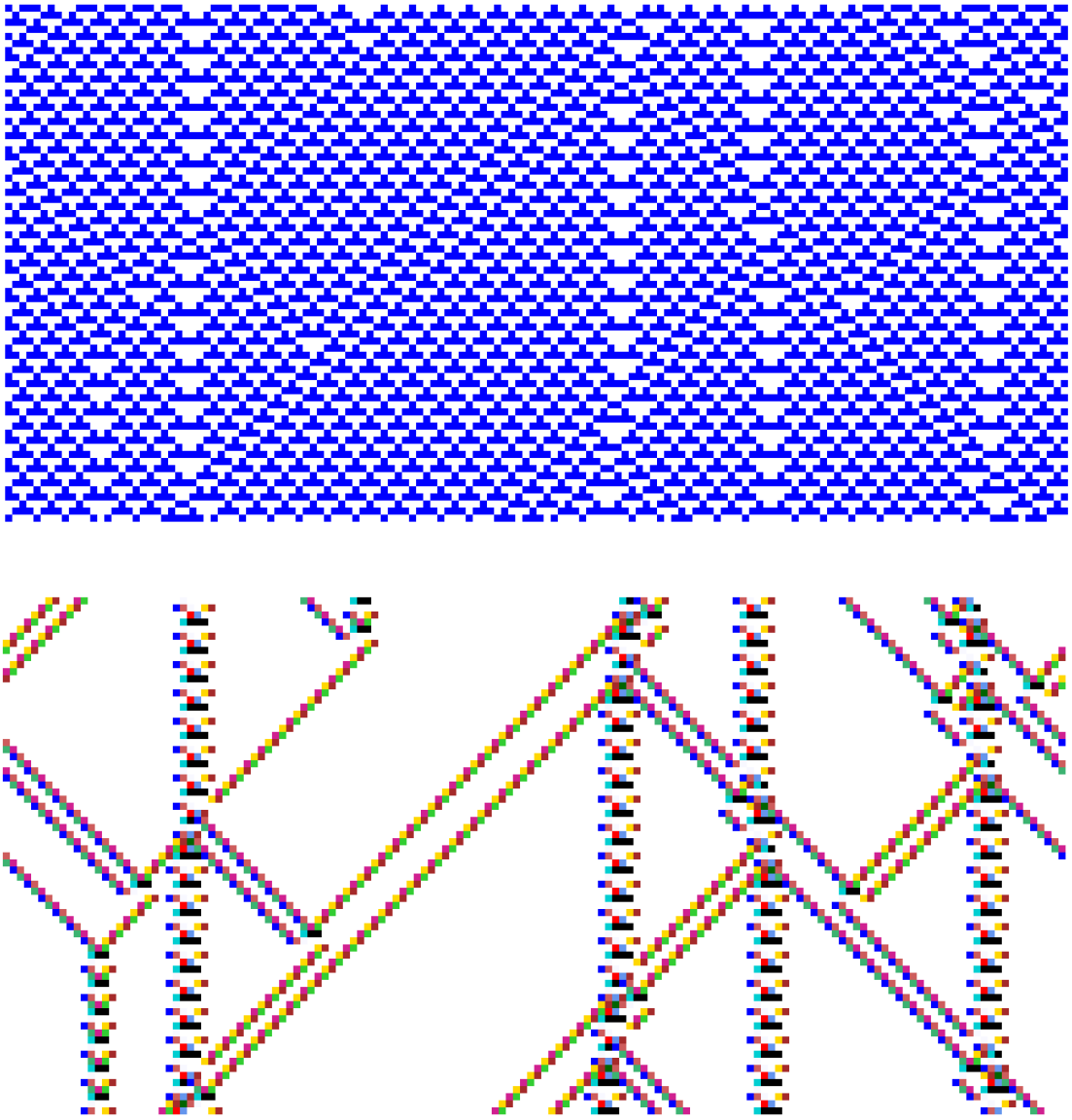

Figure 19.2: alcuni glider in un CA (il cosiddetto *Regola 54* , secondo la classificazione di Stephen Wolfram (Vedi Wolfram 2002). Il tempo scorre dall'alto verso il basso, ogni riga di pixel rappresenta la configurazione globale del sistema ad ogni passo temporale. Ogni pixel rappresenta lo stato di una delle parti elementari della CA, le sue *cellule. Sopra: la sequenza di stati del CA. Sotto: dopo aver filtrato ed escluso dall'immagine pattern ripetitivi di sfondo della configurazione del CA, i glider appaiono più chiaramente, rappresentati nel tempo da linee rette, che raffigurano lo spostamento progressivo di queste sottoconfigurazioni all'interno della configurazione globale (immagine tratta dalla Galleria di DDLab di Andrew Wuensche, http://uncomp.uwe.ac.uk/wuensche/gallery/r54_filted. gif, successivamente modificata).

La modularità nella forma della presenza nel sistema di parti simili o identiche ripetute è alla base di una forma correlata di modularità dei programmi per elaboratore, una forma compatibile con quella delineata sopra: se più di uno dei moduli del, programma esegue la stessa funzione di ingresso/uscita, anziché essere esplicitamente ripetuto sotto forma della ripetizione di copie identiche della sequenza delle sue istruzioni in punti diversi all'interno della lista globale del programma, tale modulo può semplicemente essere invocato, "chiamato" più volte in diverse parti del programma, "chiamandolo" ogni volta con configurazioni di ingresso distinte, e ricevendo sue configurazioni di uscita potenzialmente diverse al momento che il modulo termina la sua esecuzione. Moduli di programma visti in questo modo, come sottoprogrammi richiamabili, sono spesso chiamati *subroutine* o *procedure*. Questo modo di strutturare i programmi può migliorare enormemente la loro affidabilità, poiché il test di ciascuna procedura è sufficiente sia fatto una volta sola, e il sistema completo viene assemblato a partire dai moduli già collaudati.

19.1.8 Modularità gerarchica, livelli, robustezza e validità

Cercherò di chiarire qui l'importanza di una proprietà dei moduli che fino ad ora ho solo menzionato: la *robustezza*. Intuitivamente, per una descrizione modulare di un sistema dinamico, robustezza significa che un modulo a livello superiore deve sopportare una certa gamma di perturbazioni al livello inferiore, mantenendo la sua identità distinta e persistente nonostante la variazione di stato nei suoi componenti costituenti. Oppure, in alcuni casi, un modulo robusto dovrebbe rimanere lo stesso nonostante le possibili sostituzioni di alcuni dei suoi componenti di basso livello. Nelle descrizioni modulari di reti[13], Che sono descrizioni modulari *strutturali*, un modulo è considerato robusto quando non cambia la sua identità nonostante la sottrazione o l'aggiunta di alcuni collegamenti fra i suoi componenti sottostanti di basso livello[14].

Nei sistemi dinamici, la persistenza a fronte delle variazioni renderebbe ovviamente un modulo che può comprendere *qualsiasi* insieme casuale di componenti di livello inferiore un modulo molto robusto: esso non cambierebbe mai a fronte di qualunque cambiamento a livello più basso! Ma questo sarebbe una forma di robustezza *banale*. Un modulo dinamico robusto esplicativamente adeguato dovrebbe essere non troppo evanescente, ma nemmeno dovrebbe permanere in uno stato immutabile: dovrebbe invece essere in grado di seguire, in maniera grossolana, le dinamiche sottostanti. L'adeguata robustezza dei moduli è essenziale quando si producono descrizioni gerarchiche che vogliamo impiegare nelle spiegazioni scientifiche di sistemi dinamici, perché un modulo di alto livello, per essere esplicativo, deve essere supposto in grado di monitorare fedelmente, di rispecchiare diciamo, le dinamiche di basso livello, anche se a una risoluzione temporale più bassa rispetto a queste e spesso anche ad una risoluzione spaziale più bassa, o ad una precisione inferiore. Un modulo appartenente alla descrizione di un sistema dinamico, per essere utile per la spiegazione scientifica, dovrebbe essere robusto in una misura che viene scelta dall'osservatore, secondo i suoi interessi: questo è un altro aspetto della relatività della modularità gerarchica alla scelta dell'osservatore. In ogni caso, un modulo di alto livello non dovrebbe essere troppo evanescente, altrimenti il suo livello di descrizione sarebbe inutilizzabile per delle spiegazioni, né deve essere troppo congelato, poiché in quel caso non ci sarebbe alcun effetto esplicativo: un modulo il più possibile congelato, che rimane nello stesso stato per ogni possibile combinazione di tutte le parti del sistema, potrebbe semplicemente costituire un "nome" del sistema complessivo (questo potrebbe in effetti essere utile in alcuni casi, al fine di individuare fenomeni per la prima volta).

[13] Vedi sezione 19.2.1.

[14] Alcuni algoritmi di rilevamento della modularità di rete effettuano questo tipo di test per valutare la robustezza modulare. Vedi la sezione 3.2.1.2.

Comunque, la ridescrizione modulare ad alto livello in sistemi dinamici è una questione complessa: la descrizione modulare ad alto livello non deve *divergere* (perlomeno non troppo) nella sua dinamica rispetto alla sottostante evoluzione dinamica al livello basso, quello della descrizione preferita del sistema. Una descrizione modulare deve essere *valida* (per usare la terminologia della simulazione al computer per scopi scientifici), per poter essere esplicativamente utile: deve seguire con una certa precisione, anche se su scala più a grana grossa, l'evoluzione dinamica del sistema. Una descrizione modulare non valida è esplicativamente inutile. Ciò può essere meglio compreso con un esempio. Immaginiate che lanciamo l'esecuzione di un sistema dinamico discreto una sola volta e per un tempo limitato, ottenendo una sezione della sua evoluzione: vedi fig. 19.2. Potremmo elaborare immediatamente una rappresentazione modulare ad alto livello di questa esecuzione dinamica, semplicemente prendendo come modulo ciascuna delle linee diagonali che rappresentano la traiettoria di un glider, e quindi produrre una spiegazione ad alto livello del segmento in questione dell'evoluzione del sistema menzionando solo "traiettorie di glider" (sulla falsariga di "il terzo glider si scontra con il secondo e lo distrugge, proseguendo verso la grande colonna verticale di sinistra", e così via . . .). Una descrizione come questa costituirebbe una descrizione *valida* di alto livello della dinamica del sistema? Molto probabilmente no, perché avremmo basato il nostro rilevamento della modularità solo su un segmento limitato della possibile evoluzione del sistema: la prossima volta che si avviasse il sistema con una configurazione iniziale anche leggermente diversa, esso produrrebbe con la più alta probabilità un segmento completamente diverso di evoluzione, e la descrizione precedentemente ottenuta sarebbe resa inutile. Il punto è, quando ridescriviamo tali tipi di sistemi dinamici a un livello superiore, non dobbiamo cercare di ridescrivere pattern unici ed effimeri della loro evoluzione, ma solo sottosistemi adeguatamente *robusti*, che appaiano nella dinamica del sistema con un'alta frequenza e regolarità: solo in questo modo si può sperare di ottenere una descrizione modulare ad alto livello *valida*[15]. In termini che verranno chiariti nella sezione 19.4.3, potremmo anche dire che un modulo dovrebbe idealmente rappresentare, ad alto livello, una *funzione*, eseguita al fine di contribuire al comportamento complessivo del sistema. La scomposizione funzionale non è arbitraria: non ogni dissezione arbitraria di un sistema in parti casuali può essere considerata funzionale. Una decomposizione arbitraria, che porti ad una descrizione non valida, non sarebbe considerata esplicativa in senso funzionale o meccanicistico.

19.1.9 Modularità e spiegazione

Si direbbe che la modularità sia connessa con al spiegazione in modi diversi e fondamentali. Già i primi scritti di Herbert Simon sui sistemi quasi-scomponibili evidenziano che la formula che rappresenta la dinamica aggregata[16] di un sistema quasi-scomponibile è *più semplice* della formula della sua dinamica originale, e questo significa che l'aggregabilità produce economia di descrizione. Dal momento che una spiegazione scientifica della dinamica del sistema (perlomeno una spiegazione di tipo deduttivo-nomologico[17]) utilizzerebbe sicuramente questa formula, questo fatto permette di conseguire un'economia di spiegazione.

In generale, la modularità dovrebbe permettere una forma di descrizione a grana grossa (*coarse-graining*), un'operazione che consiste nel prendere un sistema complesso rappresentato come un

[15] Si può obiettare che la robustezza dei moduli di alto livello può essere ottenuta tramite la mappatura in un modo complesso e non immediatamente evidente di insiemi eterogenei di sottoconfigurazioni effimere di basso livello a moduli di alto livello. Questa, credo, non è un'obiezione banale, soprattutto considerando l'importanza che attribuisco alla *relatività* della modularità. Esamino la questione, che ha profonde implicazioni, nella sezione 14.5.2.

[16] Vedi sezione 19.1.5.

[17] Vedi sezione 19.4.4.

insieme di numerose parti, partizionare questo insieme in sottinsiemi distinti, e considerare, al posto del sistema originale, un altro insieme in cui ogni parte corrisponde a uno dei sottoinsiemi distinti del sistema originale. Questa risulta essere essenzialmente la stessa operazione, sia quando riguarda insiemi di variabili di un equazione, come nell'operazione di aggregazione, dove porta a una modularità dinamica, sia quando riguarda una rete, in cui la rappresentazione originale può essere sostituita da una rete con u numero inferiore di nodi, sia nel caso di spiegazioni funzionali o meccanicistiche[18], in cui un gruppo di parti interagenti o di azioni può può venire ad essere visto come una singola funzione, o un singlo meccanismo, e un gruppo di meccanismi può essere visto come un singolo super-meccanismo, le cui parti sono i singoli meccanismi più semplici. Questo, in un certo modo, vale anche per la modularità nei sistemi computazionali, nei quali una lista di istruzioni può essere riscritta in un linguaggio di più ato livello in cui, ad ogni istruzione di alto livello, corrisponde un gruppo di istruzioni di livello più basso: oltretutto, quello di un linguaggio di alto livello è una forma del tutto tipica di redescrizione a grana grossa. In ognuno di questi casi, si raggiunge un'economia di descrizione e, ragionevolmente, la comprensibilità della spiegazione ne viene grandemente facilitata.

Un'altra forma di economia di descrizione è ottenuta in certe descrizioni modulari in cui più copie di un singolo modulo ricorrente possono essere sostituite da una singola citazione del modello generale di quel modulo, una forma di economia di descrizione che corrisponde, nei programmi informatici, al fatto di chiamare la stessa subroutine da punti differenti del programma.

Inoltre, considerazioni di economia o di intelligibilità a parte, la modularità è *necessaria* per produrre certi tipi di spiegazione. La strategia esplicativa analitica di Robert Cummins, che toccheremo nella sezione 19.4.3 insieme con la spiegazione meccanicistica, esplicitamente invocano una scomposizione gerarchica del funzionamento del sistema, al fine di spiegarlo. Naturalmente, questa decomposizione è possibile nel caso in cui sia presente nel sistema qualche forma di modularità funzionale, cioè quando i moduli ricercati possono legittimamente essere considerati moduli funzionali. Allo stesso modo, l'idea di una spiegazione meccanicistica sembra richiedere il ritrovamento di una coincidenza tra due tipi di gerarchie, una coincidenza tre una descrizione strutturale e una descrizione funzionale del sistema, almeno nella concezione di meccanismo portata avanti da William Bechtel e dal suo gruppo : per questi autori, che non vedono la spiegazione meccanicistica semplicemente come riduzionistica, è essenziale che la spiegazione sia *multilivello*, e ciò corrisponde a una descrizione meccanicistica gerarchica funzionale del sistema. Abbracciando una visione *epistemica* delle spiegazioni[19], questi autori naturalmente sottolineano anche l'importanza dell'*intelligibilità cognitiva* delle Spiegazioni, e questa può essere ottenuta tramite la modularità delle descrizioni impiegate nelle spiegazioni.

Quindi, in primo luogo, sembra che almeno spiegazioni di un certo tipo, ovvero spiegazioni meccanicistiche o funzionali, *richiedano* modularità, anche quando si trascurino questioni circa l'intelligibilità di queste spiegazioni.

Ma la modularità gerarchica permette anche spiegazioni multilivello che certamente migliorano la *comprensione*. Data un'adeguata scomposizione gerarchica meccanicistica, un sistema può essere descritto a qualsiasi livello desiderato di descrizione, con risultati diversi sulla intelligibilità della spiegazione: i livelli grossolani più astratti consentono una spiegazione molto semplificata, che di solito induce una migliore comprensione, mentre la scelta di procedere fino a livelli inferiori, più dettagliati, migliora le informazioni sul sistema veicolate dalla spiegazione, probabilmente a prezzo della comprensione: la spiegazione più dettagliata possibile è quella che descrive il sistema

[18] Vedi al sezione 19.4.3.
[19] Vedi sezione 1.4.3.

in termini di entità del livello più basso[20], e, in molti casi, l'enorme quantità di informazioni contenute in tale descrizione potrebbe ostacolare la sua intelligibilità.

19.1.10 Modularità e scienze biologiche: qualche esempio

Mi concentrerò qui su alcune brevi considerazioni riguardo l'importanza della modularità nel pensiero e nella ricerca biologici, e questo perché la biologia è uno dei campi in cui la modularità è stata più al centro dell'attenzione negli ultimi tempi. Un'osservazione ovvia è che gli organismi sono indubbiamente modulari a molti livelli: possono, da un punto di vista biologico, grosso modo essere visti come composti di sistemi, organi, cellule, macromolecole. È meno evidente se la modularità risulti presente a certi livelli intermedi che possono essere considerati come sistemi complessi, comprendenti molte parti: per esempio, il genoma, il proteoma o la rete metabolica sono modulari?

Quindi, una prima domanda da porci è: l'evoluzione produce architetture modulari dinamiche modulari negli organismi? E, se questo è il caso, la modularità è evoluta per selezione naturale o per altri motivi? Al di là dell'eventuale studio empirico di questo problema, alcune considerazioni a priori sono sembrate in grado di gettar luce sulla questione, almeno fin dai tempi di Herbert Simon. C'è un certo numero di argomenti che puntano a concludere che la selezione naturale dovrebbe realmente condurre ad un'organizzazione modulare. Tutti questi argomenti sostanzialmente derivano dal seguente ragionamento: in un organismo completamente integrato, non modulare, in cui ogni parte potenzialmente influisce su ogni altra, il cambiamento evolutivo in una parte potrebbe incidere e possibilmente alterare le funzioni svolte da altre parti, e, dato questo, il numero di tentativi evolutivi potenzialmente necessari per ottenere un organismo ancora funzionante dopo un cambiamento in una sua parte sarebbe enorme, quindi è plausibile affermare che, se così fosse, la selezione naturale non avrebbe avuto il tempo, nonostante la scala geologica dei tempi evolutivi reali, per determinare l'evoluzione dei sistemi complessi. Questo è più o meno l'argomento generale a favore dell'evoluzione della modularità inizialmente proposto negli anni '60 da Herbert Simon[21] e adottato, con alcune variazioni, da molti autori successivi. A partire da alcuni lavori di Stuart Kauffman nei primi anni '90, è apparso un argomento alternativo (ma a mio parere non così dissimile, vedi la sezione 7.1.2) che, pur affermando la modularità dei sistemi biologici, nega la sua diretta provenienza dalla selezione naturale: la modularità sarebbe invece una proprietà auto-emergente di una certa classe di sistemi complessi dinamici, dovuta al "congelamento" di alcuni dei loro sottosistemi dinamici (vedi sopra, sezione 19.1.6), un fatto che non nasce dalla selezione diretta, ma in virtù delle caratteristiche intrinseche, matematiche di questi sistemi[22]. Il genoma di un organismo (visto come un complesso di parti interagenti, nella forma della rete di regolazione genetica) può, secondo Kauffman, essere considerato, con una certa approssimazione, appartenente a questa classe di sistemi dotati di una tendenza a fare emergere la modularità spontaneamente, una classe che si rivela essere la classe della maggior parte dei sistemi *evolvibili*: il ruolo della selezione naturale sarebbe stato quello di operare una meta-selezione della *classe* di sistemi in grado di evolversi, all'interno della quale, poi, operare il suo più fine ruolo analitico selettivo, come classicamente concepito nel darwinismo[23], e

[20] Il fatto stesso di giungere a queste entità di più basso livello, questo "toccare il fondo", che corrisponde nella mia terminologia al raggiungimento della descrizione preferita, è di solito una questione di scelta o convenzione, anche secondo Bechtel e i suoi co-autori. Vedere le sezioni 19.4.7 e 11.1.5.

[21] Con la celebre parabola dei due orologiai, vedi la sezione 7.1.1 e, naturalmente, Simon (1962).

[22] Questo tipo di spiegazione fornita da Kauffman può essere vista come una forma di spiegazione *topologica*, secondo il modello recentemente proposto da Philippe Huneman. Vedi la sezione 19.4.5 e Huneman (2010).

[23] Colgo l'occasione qui per un'avvertenza: anche quando parlo della selezione naturale in termini intenzionali, non sto proponendo di considerare la selezione un soggetto intenzionale: è solo, ovviamente, un'utile "façon de parler", ampiamente diffusa in filosofia e biologia.

questa classe è la classe di sistemi complessi che, spontaneamente, mostrano una qualche forma di modularità.

Quindi sembrerebbe, tutto considerato, che ci siano ragioni per le quali i sistemi biologici che si sono evoluti debbano avere preferibilmente una organizzazione modulare. Molti di questi sistemi sono così complessi e composti di tante parti, che il rilevamento della loro modularità *funzionale*, consentendo la loro spiegazione multilivello, sarebbe di grande aiuto anche nella *comprensione* di tali sistemi.

In biologia, sin dai tardi anni '90, alcune proposte sulla possibilità di vedere sistemi biologici complessi come composti da moduli funzionali sono state direttamente ispirate dal punto di vista ingegnenristico sui sistemi artificiali, in particolare sui circuiti elettrici: tra le proposte più importanti da questo punto di vista vi sono McAdams & Shapiro (1995), e Hartwell et al. (1999)[24]. Questo punto di vista è stato applicato a reti genetiche e metaboliche, dove l'elevata specificità dei collegamenti elettrici tra i componenti di un circuito elettronico è sostituita dalla specificità del rapporto tra una proteina e il suo ligando, e l'intera rete biologica viene rappresentata come un *circuito digitale*, che è equivalente (con alcune differenze, tenendo conto dei ritardi di propagazione dei segnali) a un tipo di rete booleana. In questi circuiti, i moduli sono le occorrenze separate di *componenti standard* , collegati da connessioni, e il circuito digitale completo può essere visto come una struttura gerarchica, in cui ogni livello è descrivibile come un circuito di parti modulari ripetibili interconnesse, che rendono in grado le unità digitali di alto livello di realizzare praticamente qualsiasi circuito digitale, anche quelli che possono essere visti come sistemi computazionali. Vedi fig. 19.3 per un'esemplificazione della visone gerarchica dei circuiti elettronici digitali.

Le parti modulari possono essere, nel caso di reti genetiche, singoli geni e, ad un livello superiore, complessi di geni come gli *operoni* dei genomi batterici. Tali componenti di livello superiore avrebbero un ruolo funzionale, come è il caso, per esempio, di un operone, che controlla la produzione di un complesso di enzimi che svolgono una funzione metabolica specifica. Una possibile rappresentazione schematica di un "circuito genetico" è riportata in fig. 19.4.

Hartwell et al. (1999) ha proposto che i termini linguistici ("amplificazione", "correzione degli errori", "rilevamento coincidenza", e così via ..) corrispondenti a funzioni di medio e alto livello svolte dai moduli situati a livelli gerarchici intermedi, vengano a costituire un vocabolario di termini, essenziale per la descrizione funzionale dei sistemi biologici.

19.2 Rilevamento algoritmico della modularità

Esaminerò qui algoritmi conosciuti per il *rilevamento della modularità* in certe classi di sistemi complessi, cioè algoritmi che, dato un sistema complesso e una descrizione elementare preferita di esso, cercano di produrre una descrizione modulare gerarchica del sistema.

19.2.1 Rilevamento algoritmico della modularità nelle reti e complessità computazionale

Prendo qui in considerazione, in particolare, algoritmi per il rilevamento della modularità nelle *reti*, perché i modelli di rete sono emersi come uno dei modi preferiti di rappresentare sistemi complessi, in particolare i sistemi biologici, nella ricerca recente. Una rete può in generale essere

[24] Per una meta-riflessione sui metodi della ricerca biologica, la modularità, e l'approccio ingegnenristico, vedi anche Lazebnik (2002).

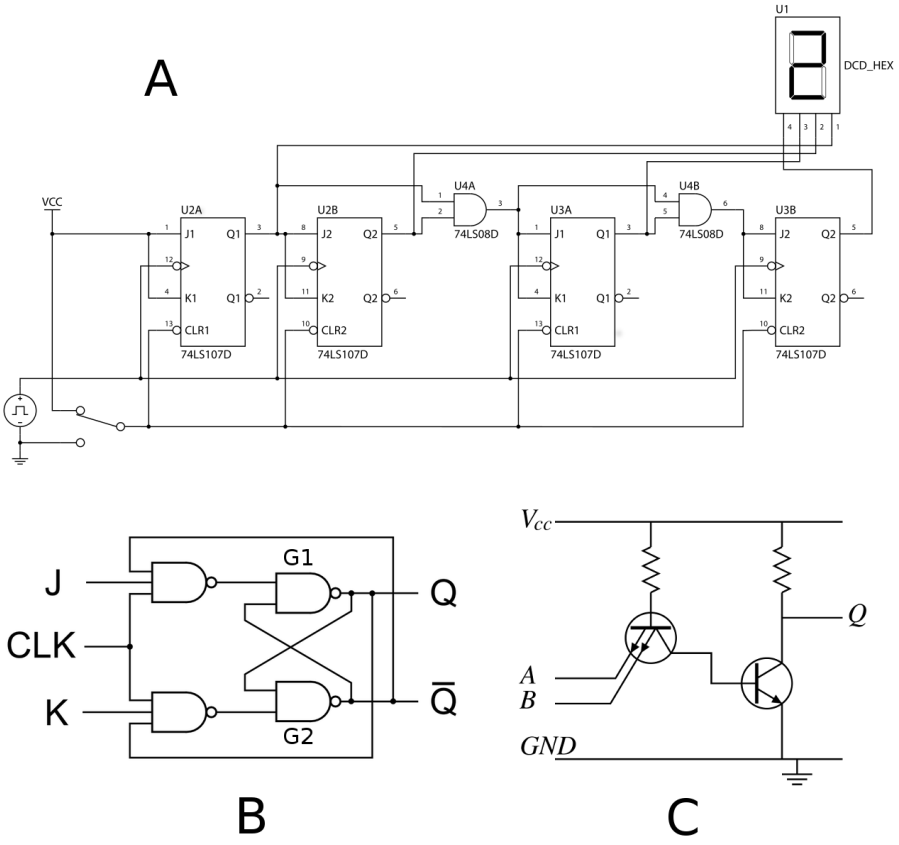

Figure 19.3: immagine A: un diagramma di alto livello che rappresenta un circuito digitale. Fatta eccezione per un paio di porte logiche singole ($U4A$ e $U4B$), la maggior parte dei componenti sono di livello superiore, e possono essere considerati i moduli che svolgono funzioni di livello superiore. In questo caso, ciascuno degli elementi etichettati $U2A$, $U2B$, $U3A$, $U3B$ è un cosiddetto *flip-flop*, un tipo di cella di memoria a 1 bit. Ciascun flip-flop può essere visto (immagine B) come composto internamente da un certo numero di elementi semplici, cioè porte logiche NAND. Ciascuna delle porte NAND a due ingressi etichettate come $G1$ e $G2$ nell'immagine B sono strutturate internamente come un circuito composto da transistor e resistenze, come nell'immagine C. Naturalmente, la descrizione a livello superiore a quello dei flip-flop è possibile: per esempio, tutto il circuito dell'immagine A può essere definito come un modulo che svolge la funzione di contatore a un'unica cifra, che conta gli impulsi inviati alla sua linea di ingresso e visualizza il numero contato sul display etichettato $DCD\ _HEX$. Come modulo, questo circuito può essere utilizzato come una parte standard in altri circuiti più grandi. (Immagini A, B e C tratte da Wikipedia Commons, rispettivamente da http://commons.wikimedia.org/wiki/File:4_bit_counter.svg, http://commons.wikimedia.org/wiki/File:JK-FlipFlop_(4-NAND).PNG and http://commons.wikimedia.org/wiki/File:TTL_npn_nand.svg).

considerata come un insieme di parti, i suoi *nodi*, collegate tra loro in diversi modi tramite connessioni , o *link* (una possibile rappresentazione grafica di una rete è fig. 19.5). Ci sono due principali possibili forme di modularità nelle reti, non incompatibili tra loro: struttura di comunità e di motivi di rete[25]. Mentre l prima si basa sul tipico concetto di modularità instesa come sottosistemi robusti debolmente collegati, la seconda coincide con l'idea di moduli come parti standard ripetibili.

[25] Vedere la sezione 3.

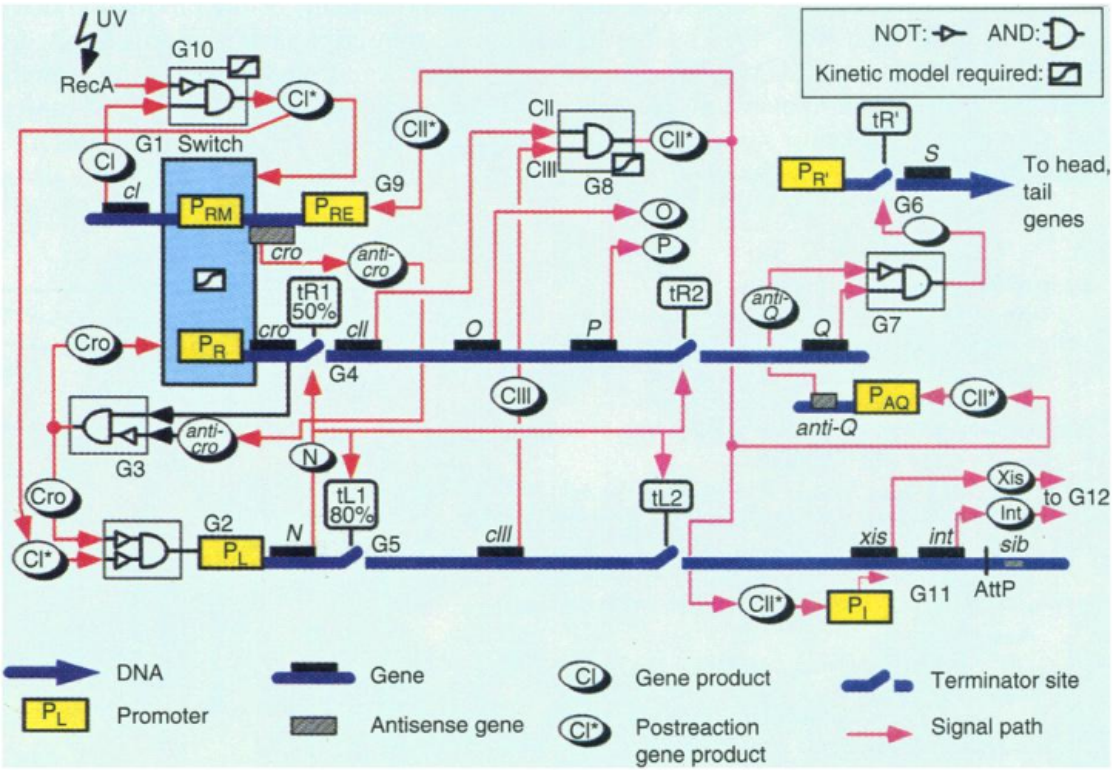

Figure 19.4: rappresentazione schematica del circuito genetico generante la dinamica del ciclo lisogenico del fago λ phage. (Il fago è un virus che attacca i batteri. Immagine tratta da McAdams & Shapiro 1995, p. 652).

Il capitolo 3 fa una panoramica dettagliata sui principali metodi proposti per la rilevazione dei due tipi di modularità, con una particolare attenzione alla fattibilità computazionale di questi metodi: risulta che la maggior parte dei migliori algoritmi per il rilevamento di modularità nelle reti sono computazionalmente molto impegnativi, e che vi è anche un limite teoricamente stabilito sulla loro accuratezza. In sintesi, è stato dimostrato che l'accertamento automatico della *migliore descrizione modulare possibile* di un sistema è ostacolato da un'insormontabile complessità di tempo di calcolo: il compito è *NP-completo*[26]. Inoltre, è un dato di fatto che la maggior parte degli algoritmi per almeno *approssimare* l'individuazione ottimale della modularità nelle reti sono di per sé anch'essi altamente intensivi dal puto di vista computazionale. In generale, sembra che il rilevamento algoritmico della modularità di rete venga influenzato da un compromesso tra complessità del compito e affidabilità della descrizione modulare prodotta, e che, per questo motivo, sia algoritmicamente possibile solo per sistemi dimensioni limitate l'identificazione di loro descrizioni gerarchiche approssimative ma accettabili.

19.2.2 Rilevamento della modularità nei sistemi dinamici discreti e nei sistemi computazionali

In linea con le considerazioni sulla modularità dinamica fin qui fatte, procedo qui a prendere in considerazione forme di modularità dinamica in alcuni tipi di sistemi dinamici discreti capaci

[26] Vedi sezione 19.3 e, nell'Appendice , la sezione 17.4.

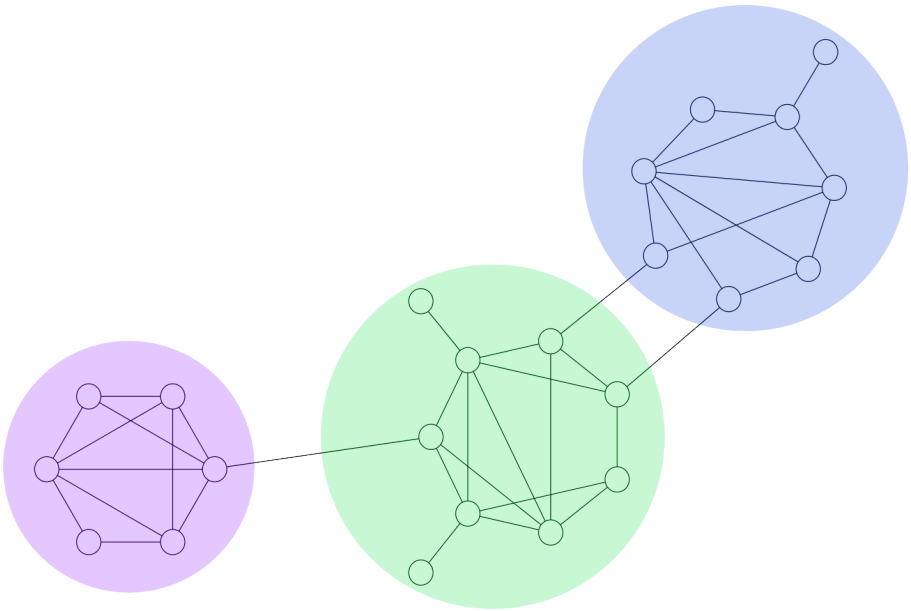

Figure 19.5: una rete con struttura di comunità. In questa immagine, i dischi colorati circondano le comunità, che mostrano alta densità di legami intra-modulo, mentre i collegamenti esterni tra moduli diversi esterni sono più radi.

di computazione. La possibilità di *rilevazione* della modularità dinamica e computazionale in questi tipi di sistemi, che spesso possono essere considerati sistemi computazionali a livello della macchina di Turing, risulta essere afflitta da indecidibilità algoritmica o, almeno, dalla complessità computazionale del compito. Una prima discussione introduttiva su questa questione sarà effettuata nel seguito.

19.2.3 Alcune applicazioni reali nella ricerca scientifica della rilevazione di modularità

Dato che la modularità funzionale e quella strutturale, anche se concettualmente distinte, sono spesso legate, i metodi di rilevazione automatica della modularità nelle reti, che si applicano alla *struttura* della rete, potrebbero, se applicati a rappresentazioni sotto forma di rete di un sistema biologico, produrre una descrizione modulare immediatamente funzionale. La coincidenza frequente tra l'organizzazione strutturale e l'organizzazione funzionale nei sistemi biologici è confermata da molte opere, in particolare, tra l'altro, da una serie di ricerche di Zhou e Lipowsky[27], in cui uno dei migliori metodi di rilevazione della modularità nelle reti viene applicato alla rete di interazioni proteina-proteina del lievito, producendo una descrizione modulare comprendente 449 moduli, che risultano corrispondenti a sottosistemi funzionali già noti, e che sono componenti di una descrizione modulare ad ancora più alto livello. Un altro lavoro importante che evidenzia la coincidenza tra modularità strutturale e funzionale nei sistemi biologici è Guimerà & Amaral (2005b), che applica a reti metaboliche un algoritmo per la rilevazione della modularità il quale identifica i moduli e quindi assegna loro un ruolo funzionale presunto, sulla base della connettività strutturale intra-modulo e inter-modulo. I moduli funzionali identificati hanno ruoli che

[27] Vedi Zhou & Lipowsky (2004) e Zhou & Lipowsky (2006).

risultano correlare abbastanza bene con le funzioni biologiche reali che metaboliti corrispondenti a ciascun modulo effettivamente compierno nella rete metabolica complessiva. Vedere fig. 19.6.

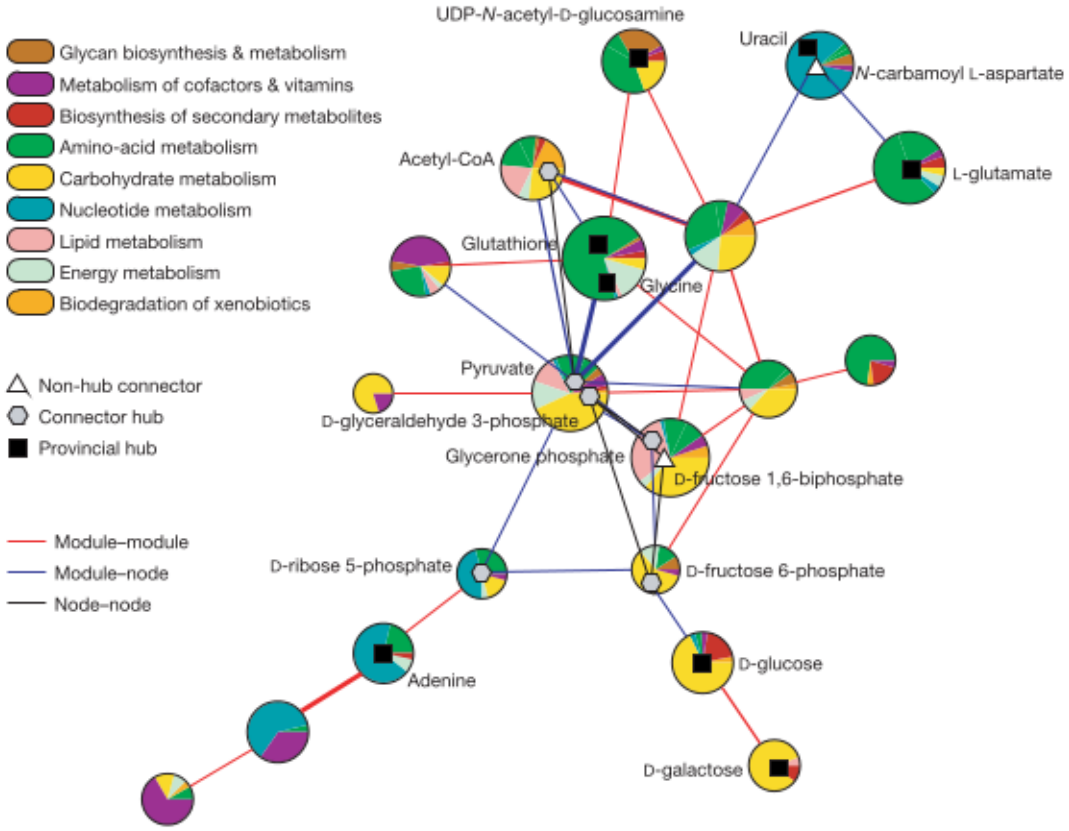

Figure 19.6: rappresentazione modulare ad alto livello di una rete metabolica. (Immagine tratta da Guimerà & Amaral 2005b).

19.3 Intrattabilità computazionale

L'intrattabilità computazionale è una limitazione pragmatica di alcuni compiti di calcolo, che consiste essenzialmente nel fatto che essi non possono essere portati a termine se la dimensione dei loro dati di ingresso supera certi limiti[28]. Ciò significa che, in generale, il compito computazionale in questione, mentre è eseguibile *in linea di principio*, potrebbe non essere mai portato a termine in tempi umani, o anche tempo astronomicamente plausibili, se la dimensione dell'input supera una certa grandezza. Questo è tipico, per esempio, dei problemi che hanno tempi di esecuzione proporzionale a una funzione esponenziale delle dimensioni del loro input: anche per piccole dimensioni del loro ingresso, il loro tempo di completamento può raggiungere valori ingestibili, perché le funzioni esponenziali crescono in maniera molto ripida. Quindi, anche se l'intrattabilità computazionale non è una limitazione *in linea di principio*, essa è certamente un limite insormontabile da un punto di vista pragmatico. Le classi più tipiche di complessità

[28] Questa è la *complessità temporale* dei programmi, che è non l'unico tipo di complessità computazionale. Altri tipi di complessità e un migliore trattamento di queste questioni possono essere trovati nella sezione 17.4.

computazionale che possono essere considerate difficili sono le classi dei cosiddetti problemi algoritmici *NP-completi* e *NP-hard*.

Alcuni compiti algoritmici che non sono da considerare formalmente intrattabili, possono tuttavia essere troppo computazionale esosi, per essere di applicabilità pratica. Questo accade per esempio quando un'attività richiede una serie di passi che è proporzionale a una potenza intera del formato di input: per esempio, n^4, dove n è la dimensione dei dati di ingresso. In questi casi, dato un input sufficientemente grande, il programma prenderebbe certamente troppo tempo a completare l'elaborazione, per essere di un qualche uso pratico.

L'intrattabilità computazionale e e l'eccessivo costo computazionale pratico sono due nozioni di base su cui la mia proposta sull'antimodularità si incentra, come vedremo nella prossima sezione.

19.4 Antimodularità

Tenuto conto di tutti i risultati di cui sopra circa la difficoltà computazionale del rilevamento algoritmico di modularità, propongo di definire la proprietà dell'*antimodularità* in generale, come l'impossibilità di ottenere, *mediante rilevamento algoritmico della modularità*, un'utile descrizione modulare gerarchica valida di un sistema. Più precisamente, un sistema mostra antimodularità quando la sua descrizione gerarchica più fattibile e fedele, prodotta da mezzi algoritmici, è comunque troppo approssimativa per costituire in ogni caso un'utile descrizione di alto livello del sistema, oppure quando essa è persino completamente non-valida. In questi casi, l'unica descrizione gerarchica possibile comprende solo due livelli gerarchici banali: il livello di tutto il sistema e il livello delle sue parti singole, di livello più basso: in altre parole, i sistemi *antimodulari* sono sistemi che, intuitivamente, possono essere spiegati dalla decomposizione ad *un unico* livello, il livello dei loro componenti elementari, più fini.

Lantimodularità è dovuta al fallimento della applicazione dei metodi algoritmici per il rilevamento modularità, e questo a sua volta può essere eventualmente attribuito a due condizioni:

1. Nessun livello intermedio di modularità può essere ragionevolmente supposto nel sistema, a partire dall'osservazione della sua descrizione preferita. Cioè, detto in maniera semplice, il sistema così descritto è in realtà *non* modulare affatto. Io chiamo questo caso *antimodularità intrinseca*, il che significa che l'antimodularità è intrinseca alla descrizione preferita data, non importa quanto sia preciso l'algoritmo per la sua individuazione. Questa situazione può verificarsi quando le parti del sistema, secondo la descrizione preferita, sono iperconnesse: per esempio, in una rete regolare ogni nodo è collegato a *tutti gli altri*, e quindi nessuna modularità può mai presentarsi.

2. Indipendentemente dal fatto che una struttura modulare reale sia presente nella descrizione preferita del sistema o meno (come al punto 1), l'antimodularità sorge perché, dato l'*elevato numero di parti* che compongono la descrizione preferita del sistema, l'algoritmo per il rilevamento di modularità finisce per essere computazionalmente troppo costoso per essere portato a termine, sia perché esso è computazionalmente intratabile[29], o, anche se formalmente non è intrattabile, perché esso è comunque troppo impegnativo computazionalmente per essere portato a termine in ogni caso. Io chiamo questa ultima ragione semplicemente *antimodularità* (naturalmente, l'antimodularità intrinseca è un caso di antimodularità).

[29] Vedi sezione precedente.

La ragione di questa distinzione tra antimodularità e antimodularità intrinseca è che, mentre l'antimodularità potrebbe in alcuni casi essere eliminata migliorando l'algoritmo di rilevamento della modularità, l'antimodularity intrinseca rimarrebbe ancora in ogni caso, non essendo essa dovuta alla inesattezza o al costo computazionale dell'algoritmo impiegato, ma a una caratteristica oggettiva di un sistema in cui, grazie alla distribuzione uniforme in tutto il sistema della forza della relazione che riguarda le parti, la modularità, relativamente a quel rapporto scelto tra le parti, è in realtà, oggettivamente, assente.

Potrebbe essere utile, una volta in presenza di antimodularità, avere un metodo per dire se questa è antimodularità intrinseca, o se invece la modularità è presente, ma non può essere rilevata. Essendo dovuta alla distribuzione statistica della relazione tra le parti, l'antimodularità intrinseca, almeno l'antimodularità intrinseca strutturale, dovrebbe essere ragionevolmente facile da rilevare: l'assenza intrinseca di modularità può essere rivelata da indagini statistiche sulla distribuzione di alcune proprietà attraverso il sistema. Quindi, dovrebbe essere abbastanza facile distinguere antimodularità e antimodularità intrinseca, almeno in certi casi. Vi sono, tuttavia, delle eccezioni che verranno discusse nella sezione 13.2.

Alla luce di quanto abbiamo visto finora, sembra che il rilevamento di modularità possa, in sistemi sufficientemente grandi, venire effettivamente impedito da problemi di costo computazionale, o anche da intrattabilità computazionale, per cui un sistema può essere *pragmaticamente* considerato antimodulare, anche se in linea di principio esso possiede un certo grado di modularità, che, tuttavia, siamo praticamente incapaci di rilevare automaticamente. Una descrizione preferita antimodulare di un sistema non possiede, almeno per quanto ne possiamo sapere, una descrizione valida modulare di alto livello, cioè una descrizione le cui parti sono dotate di un sufficiente grado di robustezza.

L'aspetto pragmatico dell'antimodularità, comunque, non deve essere minimizzato come meramente pragmatico: è un'impossibilità pragmatica di portare a compimento in un tempo possibile un programma di computer, ma, soprattutto quando l'intrattabilità computazionale di un algoritmo è stata matematicamente dimostrata, questo ostacolo pragmatico diventa qualcosa di più pressante, assumendo la cogenza di una legge logica: non ci può essere alcuna speranza di rendere l'algoritmo per l'ottimizzazione della modularità rilevata, che è risultato essere computazionalmente intrattabile, più computazionalmente trattabile: non importa quanto cerchiamo di migliorare un algoritmo intrattabile, o aumentare la potenza del sistema su cui viene eseguito: il suo tempo di esecuzione saprà, almeno in certi casi, sconfiggere sempre qualsiasi possibile miglioramento nella velocità. L'ottimizzazione del rilevamento di modularità può essere probabilmente *approssimata* in tempi più ragionevoli, ma il compromesso tra velocità e precisione, che è tipico degli algoritmi approssimati per il rilevamento di modularità, connesso all'elevato numero di parti di alcuni sistemi complessi, potrebbe rendere la modularità reperita eccessivamente approssimativa o, al contrario, rendere troppo elevato il tempo di rilevamento di una descrizione modulare sufficientemente precisa, e questo anche se l'algoritmo approssimato non è, da un punto di vista formale, computazionalmente difficile.

Quindi, l'antimodularità, almeno per quanto riguarda la ricerca della migliore descrizione modulare, ricerca che è stato dimostrata essere un compito NP-completo, è una proprietà pragmatica ma allo stesso tempo *oggettiva*, inevitabile di un sistema, proprietà derivante da proprietà computazionali che non dipendono da vincoli contingenti o da una scelta operata dall'osservatore.

19.4.1 Emergenza antimodulare

Propongo di chiamare il verificarsi dell'antimodularità in un sistema un caso di *emergenza antimodulare*, e di considerrlo una forma di *emergenza computazionale*. L'antimodularity appare in

effetti molto simile a una forma ben nota di emergenza: l'*emergenza debole* (*weak emergence*), una nozione proposta da Mark Bedau a partire dalla metà degli anni '90. Si tratta di una nozione di emergenza diacronica connessa a certe proprietà di sistemi computazionali. In particolare paragono la mia nozione di emergenza antimodulare a quella di emergenza debole, concludendo che l'antimodularità comporta, con qualche clausola, l'emergenza debole, ma che l'implicazione conversa non vale per tutti i sistemi: ci sono sistemi *modulari* che sono, allo stesso tempo, debolmente emergenti. Delineo qui la linea di base del ragionamento sul rapporto tra antimodularità ed emergenza debole, una discussione che viene ampliata nella sezione 13.3.

Mark Bedau (1997) introduce la nozione di *emergenza debole* (*WE*), che, nella sua formulazione originaria, vale soprattutto per sistemi dinamici discreti:

> Il macrostato P di S con microdinamica D è debolmente emergente sse P può essere derivato da D e dalle condizioni esterne di S, ma solo tramite una simulazione[30]

Senza soffermarmi in una spiegazione approfondita dei termini impiegati nella definizione di cui sopra, è sufficiente notare qui che la definizione di Bedau potrebbe essere riformulata senza dubbio nel modo seguente:

> Un macrostato è debolmente emergente sse può essere derivato a partire dalla descrizione di basso livello (dalla descrizione preferita) del sistema, associata allo stato iniziale del sistema, ma *solo* tramite *microsimulazione*, cioè, simulando la dinamica del sistema passo-passo secondo la sua descrizione di livello più basso (cioè, la sua descrizione preferita).

Risulta che, nella maggior parte delle condizioni, condizioni che saranno meglio specificate in seguito qui e nella sezione 13.3, l'antimodularità comporta l'emergenza debole di Bedau. L'argomento è, in breve, questo: se un sistema è antimodulare, allora, per definizione, questo significa che la sua *unica* descrizione modulare valida è il livello di descrizione più basso, la sua *descrizione preferita*. Ciò implica che il sistema non sia *prevedibile* mediante una simulazione *modulare* di alto livello: perché, se lo fosse, ciò significherebbe che la simulazione di alto livello, in quanto in grado di predire il sistema, rappresenta una descrizione modulare *valida* di alto livello. Ma, in un sistema antimodulare, questa descrizione modulare valida di alto livello è esclusa dalla definizione stessa di antimodularità. Quindi, si può concludere che la dinamica di un sistema antimodulare non è suscettibile di essere predetta da alcuna simulazione modulare di alto livello: se nessun altro metodo non modulare di predizione è applicabile, l'unico modo per sapere come il comportamento del sistema evolverà è simulandolo a livello della sua descrizione preferita, cioè per microsimulazione. Quest'ultima circostanza appare equivalente alla riformulazione che ho fatto più sopra della definizione di emergenza debole di Bedau. Così, sembra che *antimodularità* $\rightarrow$ *emergenza debole*. Questa implicazione non è del tutto certa, tuttavia, poiché essa dipende dal fatto che un sistema antimodulare, che non è prevedibile da alcuna simulazione *modulare* di alto livello, sia ugualmente impossibile da prevedere anche con qualsiasi altro mezzo non modulare. Nella sezione 13.3 evidenzierò come certi sistemi antimodulari potrebbero effettivamente essere predetti con mezzi non modulari di alto livello, e quindi risultare non essere debolmente emergenti, ma sosterrò anche che questi sistemi non sono probabilmente molto interessanti nel loro comportamento, e che in sistemi complessi più interessanti, come quelli capaci di computazione, l'antimodularità comporta l'emergenza debole.

[30] Bedau (1997), p.378.

La cosa interessante è che l'implicazione conversa non tiene: esistono sistemi debolmente emergenti che allo stesso tempo non sono antimodulari, cioè hanno descrizioni modulari di alto livello valide. Il sistema rimane debolmente emergente anche in presenza di queste descrizioni modulari, perché tali descrizioni di alto livello non possono essere utilizzate per prevedere il suo comportamento (circostanza che, se possibile, renderebbe il sistema *non* debolmente emergente, per definizione), ma possono essere utilizzate solo per *spiegare* il sistema. Ciò può accadere per due possibili ragioni: perché tali descrizioni modulari sono troppo vaghe, troppo astratte, troppo di alto livello per essere impiegate per calcolare una simulazione dinamica del sistema: per esempio, si pensi al caso di diagrammi di flusso che semplicemente descrivono sommariamente il ruolo funzionale che i moduli ricoprono nel sistema, senza fornire indicazioni sufficienti per consentire la loro implementazione. Queste descrizioni modulari di alto livello non possono simulare dinamicamente il sistema, quindi non possono anticipare in alcun modo i suoi risultati dinamici, ma possono essere utilizzate per *spiegare* il sistema, veicolando una buona spiegazione. Oppure, in un altro caso, il motivo per cui il sistema è debolmente emergente pur avendo possibili ridescrizioni modulari di alto livello, è che il sistema, anche se funzionalmente modulare ad alto livello, è *intrinsecamente imprevedibile*, e questo è il motivo del suo essere debolmente emergente. Questo può avvenire in sistemi computazionalmente universali, che, come conseguenza dell'indecidibilità del problema della terminazione (la nota proprietà dimostrata nel 1936 da Alan Turing insieme con la sua proposta dei sistemi computazionali[31]), posseggono molte proprietà dinamiche intrinsecamente imprevedibili. I computer universali del mondo reale sono generalmente sistemi di questo tipo: essi sono altamente modulari, ma potenzialmente imprevedibili. Così, sono modulari e allo stesso tempo debolmente emergenti.

19.4.2 Antimodularità e modelli della spiegazione scientifica

Dopo aver definito questo nuovo tipo di emergenza computazionale, dovuto alla quantità di durezza computazionale che può essere manifestata in alcuni casi da algoritmi di rilevamento della modularità, cerco di trarre alcune possibili conseguenze dell'emergenza antimodulare sulla possibilità di *spiegare scientificamente* i sistemi che essa affligge. Esamino due ben noti modelli di spiegazione scientifica: la spiegazione funzionale-meccanicistica e quell deduttivo-nomologica (*DN*, d'ora in poi). Valuto quindi un modello più controverso di spiegazione, la *spiegazione computazionale*, e un altro tipo di spiegazione che è stata oggetto di una recente indagine, la spiegazione "matematico-topologica", una forma di spiegazione adeguata a spiegare alcune caratteristiche dei sistemi dinamici complessi. Concludo che l'emergenza antimodulare colpisce la fattibilità dei primi due tipi di spiegazione, e anche, seppure in modo diverso, della spiegazione computazionale, mentre non impedisce la possibilità di spiegazione topologica, venedo a costituire persino un fattore abilitante di questo tipo di spiegazione.

19.4.3 Antimodularità e spiegazioni funzionali e meccanicistiche

Affermo che l'antimodularità influisce negativamente sulla spiegazione *meccanicistica*, una forma fondamentale di spiegazione nelle scienze biologiche. Una breve digressione è dovuta, qui, al fine di descrivere ciò che questa forma di spiegazione costituisce.

Il termine *spiegazione meccanicistica* di solito si riferisce al giorno d'oggi in filosofia ad un modello relativamente recente della spiegazione scientifica, sviluppato a partire dagli anni '90 da diversi gruppi di filosofi della biologia e delle scienze cognitive che lavorano piuttosto indipendentemente. Gli esponenti più in vista delle due linee principali di indagine in questo campo risultano essere

[31] Come spiegato nella sezione 17.2.6 dell'Appendice.

William Bechtel e i suoi collaboratori, da un lato, e Carl Craver e i suoi colleghi, dall'altro[32]. Lasciando per il momento da parte le sottili differenze tra queste due concezioni principali di spiegazione meccanicistica[33], mi baso qui su Bechtel & Abrahamsen (2005), che è un testo standard per l'argomento. Bechtel e Abrahamsen (* BA * d'ora in poi) danno la seguente definizione di *meccanismo*:

> Un *meccanismo* è una struttura che svolge una funzione[34] in virtù delle sue parti componenti, operazioni componenti, e della loro organizzazione. Il funzionamento orchestrato del meccanismo è responsabile di uno o più fenomeni[35].

La definizione di cui sopra definisce un meccanismo come quello che in questo capitolo ho chiamato un *sistema complesso*, cioè un sistema composto da parti interagenti. Il punto da sottolineare è che c'è una concezione funzionale coinvolta: la funzione globale, che rappresenta l'explanandum, viene spiegata tramite la descrizione dell'organizzazione e delle interazioni delle parti che, mediante il loro funzionamento dinamico "orchestrato", producono il fenomeno. Ciò che è necessario, secondo BA, per spiegare un dato fenomeno, è quindi innanzitutto identificare le parti e le operazioni necessarie per la sua produzione. A questo scopo, il sistema nel suo insieme deve essere sottoposto a due operazioni, che BA chiamano *decomposizione strutturale* e *decomposizione funzionale*: la prima produce l'insieme di parti elementari del sistema, mentre la seconda, che nel mondo della scienza reale è spesso condotta separatamente dalla prima, identifica le operazioni elementari. Una terza operazione desiderabile è la *localizzazione*, che consiste nel collegare le parti con le funzioni che esse eseguono. Una spiegazione meccanicistica, secondo BA, è prodotta in questo modo. Questo tipo di spiegazione di basso livello non è sempre il più desiderabile, e, come BA evidenziano, è importante considerare una *gerarchia* di meccanismi, e che la spiegazione sia quindi *multilivello*. Secondo BA, un meccanismo può anche comportare molteplici *livelli di organizzazione*, essendo esso spesso parte di un meccanismo più grande di livello superiore: circostanze esterne a un determinato meccanismo possono essere viste come meccanismi globali sovraordinati, mentre i componenti di un meccanismo possono essere visti come meccanismi stessi, ricorsivamente composti di sottoparti.

Mi sembra che tutta questa concezione dei meccanismi potrebbe essere facilmente riformulata in termini di modularità, in linea con la visione che ho abbozzato fino ad ora. Il risultato della decomposizione funzionale, della decomposizione strutturale e della localizzazione è quello che ho chiamato la *descrizione preferita* del sistema: l'identificazione delle parti del livello più basso di base che l'osservatore ha deciso di identificare. BA non sottolineano particolarmente, come faccio qui, la dipendenza di questa descrizione da una scelta effettuata dall'osservatore, poiché essi ritengono implicitamente che ci sono descrizioni preferite di alcuni sistemi che sono naturalmente date, e queste ci sono indubbiamente, per esempio in biologia molecolare, dove le molecole (o, eventualmente, gli atomi) sono le parti elementari più naturali. La differenza principale con il mio punto di vista è allora che la mia concezione di modularità gerarchica è più generale, comprendente, forme di modularità esclusivamente funzionali e non fisiche, come la modularità delle computazioni.

Detto questo, partendo dalla definizione di antimodularità, è facile mostrare che *l'antimodularità costringe a ricorrere a spiegazioni a singolo livello*, obbligando a trascurare l'esigenza, essenziale

[32] Le due corrispondenti opere seminali sono Bechtel & Richardson (1993) e Machamer et al. (2000).

[33] Queste differenze, in particolare quelle più significative, tra la cosiddetta concezione *epistemica*, che io, insieme con William Bechtel, sostengo, e la concezione *ontica* dei meccanismi, sostenuta da Carl Craver, sono discusse nel capitolo 10.

[34] Vedi sezione 9.

[35] Bechtel & Abrahamsen (2005), p. 423.

per le spiegazioni meccanicistiche, dell'integrazione *multi-livello*. L'antimodularità limiterebbe la spiegazione meccanicistica al livello di descrizione che rappresenta le parti più elementari del sistema, cioè la descrizione che cita il più alto numero di parti e quindi la descrizione più complicata. Questo fatto ostacola certamente la comprensione: per sistemi abbastanza grandi, la loro spiegazione meccanicistica a questo livello è troppo complessa per essere compresa da esseri umani, e si consideri che la comprensibilità è una qualità che va perseguita nella spiegazione meccanicistica, in particolare secondo William Bechtel e i suoi collaboratori. Anche altri autori sottolineano l'importanza dell'intelligibilità per le spiegazioni, per esempio Petri Ylikoski, che considera la "salienza cognitiva" una caratteristica importante delle spiegazioni.

È evidente che una spiegazione meccanicistica cerca di rispondere a domande riguardanti il "come" ("come un fenomeno è prodotto?"), mostrando il modo in cui il complesso funzionamento dinamico di un insieme di parti interagenti produce il fenomeno. Alla stessa domanda si può rispondere anche dal punto di vista esclusivamente funzionale, e questa concezione, pensata soprattutto per caratterizzare la spiegazione in psicologia cognitiva, è stata notoriamente avanzata da Robert Cummins. In un modo simile a quello della decomposizione meccanicistica, l'analisi funzionale inizia con una caratterizzazione del fenomeno globale (la *disposizione*; dedicherò una discussione a questo termine tecnico nel capitolo 9) assunta come funzione complessivache deve essere spiegata in termini delle sue funzioni parziali componenti. Questa è una forma tipica del cosiddetto *funzionalismo di ruolo*, in quanto il concetto di funzione[36] è considerato essere il ruolo parziale ricoperto da un sottosistema al fine di realizzare il funzionamento globale del sistema sovraordinato. Vista da un punto di vista esplicativo, la funzione di un sottosistema è impiegata nello spiegare come la funzione globale, che è l'explanandum, venga eseguita mediante i contributi organizzati delle sue sottofunzioni, che eseguono la loro funzione in un'attività programmata. Questa posizione è abbastanza vicino a una concezione computazionale, ed è completamente compatibile con essa. In realtà, l'analisi funzionale di Cummins 'è il prototipo della spiegazione tipica della psicologia cognitiva, che consiste principalmente di spiegazioni funzionali, spesso sotto forma di *spiegazione computazionale*, che consiste nell'esibizione di un programma per computer in grado di produrre il fenomeno cognitivo che deve essere spiegato.

Una caratterizzazione più approfondita della posizione di Cummins è fornita nel capitolo 9, dove il rapporto tra spiegazione puramente funzionale e spiegazione meccanicistica è meglio analizzato. Quello che vorrei sottolineare è che Cummins stesso, a partire dalle sue opere precedenti, come il seminale Cummins (1975), sottolinea come la decomposizione funzionale ricorsiva fino ad ottenere una completa gerarchia sia la strategia da perseguire nelle spiegazioni scientifiche, soprattutto in quelle biologiche. L'antimodularity ostacolarebbe completamente questo obiettivo, consentendo solo due livelli spiegazione: quello più alto, quello del explanandum in sé, e, all'altra estremità della scala, il livello più basso, quello delle funzioni più elementari.

19.4.4 Antimodularità e modello nomologico-deduttivo

Nel classico modello nomologico-deduttivo (*DN*) della spiegazione, che deriva dal lavoro seminale di Carl G. Hempel e Paul Oppenheim[37], la spiegazione è vista come una *deduzione logica* dell'explanandum dall'explanans, e quello che conta è la validità e la solidità della deduzione, con una scarsa attenzione rivolta all'*intelligibilità* della spiegazione: una tale preoccupazione per la comprensibilità della spiegazione sarebbe stata considerata, nello storico ambiente postneopositivistico del tempo, un inappropriato sconfinamento della filosofia della scienza nel territorio degli aspetti pragmatici, o, peggio, *psicologici* della spiegazione scientifica. In tale prospettiva,

[36] La nozione di funzione è esaminata nel capitolo 9.
[37] Hempel & Oppenheim (1948).

tutto ciò che conta per una spiegazione è che si tratti di una deduzione corretta. La spiegazione è vista in questo modello come dipendente dalla possibilità di *previsione* del fenomeno mediante una *legge* scientifica. La spiegazione in sé ammonta alla descrizione della derivazione logica dell'explanandum da un gruppo di premess costituite da una legge scientifica e da un insieme di clausole che rappresentano le condizioni iniziali del sistema che deve essere spiegato.

Per quanto riguarda le spiegazioni deduttivo-nomologiche, evidenzio come, dal momento che l'antimodularità, nei sistemi abbastanza complessi, implica l'emergenza debole, non si possa ricorrere alla spiegazione DN nel caso di un sistema antimodulare, perché, se si potesse, vorrebbe dire che il sistema è prevedibile, e questa possibilità è negata dalla definizione di stessa weak emergence. Per chiarire: è escluso dalla definizione di fenomeno debolmente emergente che esso possa essere previsto per mezzo di una legge la quale, dato lo stato iniziale, determina in quale stato il sistema sarà in un dato momento, e che questa legge abbia un'espressione matematica che può essere *analiticamente* risolta. Come detto, questo è escluso dalla definizione di emergenza debole, la quale che afferma in sostanza che un fenomeno debolmente emergente (in un sistema dinamico discreto) è un fenomeno che non può essere previsto, e che può essere reggiunto solo eseguendo passo-passo passo la microsimulazione al livello più basso del sistema. Dal momento che la mia idea di antimodularità, date le circostanze espresse nella sezione 19.4.1, l'emergenza debole, risulta che un sistema antimodulare abbastanza complesso non può essere previsto da un'espressione analiticamente risolvibile. Quindi, nessuna spiegazione DN di un sistema complesso antimodulare potrebbe basarsi su una legge analiticamente risolubile.

Detto questo, se si prende in considerazione una specifica classe di sistemi, vale a dire gli automi cellulari (* CA *), allora un processo debolmente emergente generato da un CA può in un certo senso venire spiegato producendo un, elenco, potenzialmente molto lungo, di stadi della sua evoluzione, un elenco che può essere visto come un elenco di* passi deduttivi* all'interno di un sistema logico formale, in cui le premesse sono costituite dalla configurazione iniziale del CA e dalla regola del CA, che viene ripetutamente applicata prima alla configurazione iniziale e poi alla configurazione intermedia ottenuta ad ogni passo deduttivo. Dato che ogni regola-CA è, per il teorema di Curtis-Hedlund-Lyndon[38], locale e ugualmente valida in qualsiasi punto del reticolo del CA, la forma di una regola-CA può in questo senso essere assimilata alla forma di una l*egge* fisica, che, come legge, vale universalmente. In base a questa analogia, la produzione di questa lista di stati consecutivi del CA potrebbe in certo modo essere assimilata ad una lunga spiegazione DN, la quale, come detto, deve consistere in una deduzione logica dell'explanandum a partire da certe condizioni iniziali e da una legge. Anche in questo caso la comprensione umana sarebbe ostacolata dalla lunghezza potenziale della lista, ma in ogni caso, secondo la posizione teorica dei sostenitori post-neopositivistici del modello DN di spiegazione, la comprensione è una caratteristica inessenziale delle spiegazioni, e non è richiesta per una buona spiegazione DN. Così, in un certo senso, l'emergenza debole e, di conseguenza, l'antimodularità, non ostacolano la spiegazione DN , almeno nel caso dei CA e di altri sistemi la cui dinamica seguire una regola *universale*. Da queste considerazioni sono escluse classi differenti dai CA, ad esempio le reti booleane in genere, la cui dinamica può seguire regole che cambiano localmente e che non sono quindi universali. In questi casi, la regola da impiegare sarebbe la regola di aggiornamento globale, che, essendo non locale, è solitamente molto più complessa di un regola-CA, e, di conseguenza, l'elenco delle deduzioni costituenti la spiegazione stile DN di tale sistemi sarebbe ancora più incomprensibile.

[38] Vedere la sezione 14.2.1.

19.4.5 Antimodularità e spiegazione topologica

Passo a onsiderare ora le conseguenze dell'antimodularità sulla possibilità di spiegare un sistema complesso per mezzo di ciò che Philippe Huneman chiama *spiegazione topologica*. Huneman descrive la spiegazione topologica come "un genere di spiegazione che astrae dalle relazioni causali e dale interazioni in un sistema, al fine di cogliere qualche sorta di proprietà"topologica" di quel sistema e di trarre da quelle proprietà conseguenze matematiche che spiegano le caratteristiche del sistema che esse prendono a riferimento"[39]. Ispirate dalla topologia matematica, le proprietà topologiche di un sistema sono le proprietà riguardanti in qualche modo la sua"forma" che risultano invarianti per possibili deformazioni continue del sistema. Queste proprietà strutturali non devono appartenere a un sistema materiale, ma possono essere parti di uno spazio matematico astratto. Nella mia terminologia, direi che queste proprietà topologiche non riguardano un sistema, ma una descrizione del sistema. Ora, la spiegazione topologica consiste nello spiegare certe caratteristiche del sistema facendo appello non a eventi causali tra le sue parti, comela spiegazione meccanicistica farebbe, ma mettendo il luce alcune caratteristiche topologiche della rappresentazione del sistema in questo spazio astratto. Questo tipo di spiegazione non è meccanicistica, in quanto non specifica le particolari interazioni tra le parti che danno luogo al fenomeno dinamico: la spiegazione si basa in esclusivamente su specifica una caratteristica *matematica*, topologica, che si accolla tutto il lavoro esplicativo.

La spiegazione topologica potrebbe anche essere basata sulla presenza di una struttura *modulare*: questo può accadere, ad esempio quando una spiegazione topologica della robustezza della dinamica di una rete a perturbazioni locali consiste nel menzionare il fatto che la rete ha una *struttura di comunità*[40]: questa struttura modulare assicura che le perturbazioni rimangano locali o incanalate, senza diffondersi indiscriminatamente alla stessa velocità su tutta la rete. Al contrario, l'antimodularità intrinseca potrebbe produrre una diffusione illimitata delle perturbazioni sulla rete.

Tutto sommato, non solo sembra che l'emergenza antimodulara non ostacoli la spiegazione topologica, ma in realtà risulta che l'antimodularità intrinseca o la sua assenza potrebbero *consentire* certe spiegazioni topologiche

19.4.6 Spiegazione e predizione

La possibilità, evidenziata nella sezione 19.4.1, che vi siano sistemi che sono funzionalmente spiegabili e, allo stesso tempo, imprevedibili, di cui un esempio è la classe dei sistemi computazionalmente universali, fornisce un'indicazione abbastanza notevole, e cioè che la predizione e la spiegazione sono sforzi disgiunti: l'imprevedibilità non rende, di per sé, inspiegabile un sistema. Questo è un risultato curioso, perché prova, in un certo modo, che la *predizione* non è necessaria per la spiegazione, e quindi che il modello nomologico-deduttivo di spiegazione, anche se fosse esente da altri aspetti negativi, non potrebbe essere il modello di spiegazione onnicomprensivo per la scienza in generale. Nella scienza alcune spiegazioni sono accettabili anche se non si basano sulla predizione: si tratta delle spiegazioni funzionali, o meccanicistiche.

19.4.7 Computazione e spiegazione computazionale

Prima di valutare, nel prossimo paragrafo, le possibili conseguenze dell'antimodularità sulle spiegazioni computazionali, è necessaria riflessione su ciò che una *spiegazione computazionale* è.

[39] Huneman (2010), p. 214.
[40] Vedere la sezione 3.2.1.

Specificamente, ci chiederemo se e quando un dato sistema, in specifico un sistema dinamico, calcola o no, per vedere se può esserne data una spiegazione computazionale. A questo fine, sembra inevitabile discutere della nozione stessa di *computazione*.

A partire dal lavoro epocale di Turing del 1936, che ha fondato l'informatica, può sembrare a prima vista che il concetto di calcolo meccanico o computazione sia assolutamente chiaro: il lavoro di Turing ha infatti fornito una pietra di paragone rispetto al quale caratterizzare ciò che computazione è. Questo è certamente corretto: la computazione è stata trattata, sin dal suo inizio con Turing, come una questione eminentemente *formale*, che merita un approccio *matematico*. E questo è sicuramente stato, credo, il modo * giusto* di trattare la question, perché ha messo in luce le caratteristiche essenziali della computazione: le proprietà della computabilità, la sua forza e i suoi limiti, che sono state dettagliate in un modo approfondito e rigoroso che nessun altro approccio al problema del calcolo avrebbe potuto sviluppare. Come sempre, un approccio matematico si è rivelato come il modo più potente per evidenziare tutti gli aspetti di un fenomeno astratto e di trarre importantissime conseguenze e nuove idee e modelli da questa analisi: uno sguardo all'Appendice[41] sarà sufficiente a convincere di questo il lettore.

Quand'è che un dato sistema effettua una computazione? Sulla scia di quanto detto sopra, questa sembra una domanda facile, decidibile in modo formale, ma in realtà, da un altro punto di vista, essa non lo è. Per vedere questa difficoltà si potrebbe immaginare qualcuno che lancia un programma sul suo computer, e che alla domanda "Che cosa fa questo programma?" rispodesse che *calcola*, simpliciter. Questa risposta avrebbe un senso per noi? Supponiamo, ancora una volta, che qualche programmatore abbia realizzato per divertimento un programma scrivendo un elenco casuale di istruzioni, e che, per un colpo di fortuna, il programma, invece di schiantarsi, abbia girato, producendo stringhe apparentemente casuali sullo schermo. Considereremmo questo programma come qualcosa che effettua una computazione, che calcola? Sì, in un certo senso, esso calcola ... ma calcola *cosa*?!? Terzo caso: abbiamo un programma in esecuzione su un computer molto primitivo, che produce in output direttamente delle stringhe binarie: fornendogli la stringa: 00110101, produce 1111, data 01000011 Produce 1100. Tale programma calcola?

Mi sembra che gli esempi soprastanti suggeriscano immediatamente che il riconoscimento della presenza di una computazione ha bisogno di una sorta di attribuzione intenzionale: è l'attribuzione di *calcolare*, a ciò che altrimenti potrebbe essere visto come una semplice trasformazione fisica di elementi: per dire che un dato processo calcola, abbiamo bisogno di specificare *che cosa* esso calcola. Per fare questo, una condizione deve essere soddisfatta: una corrispondenza deve essere stabilita tra le configurazioni fisiche della macchina, su cui il presunto calcolo agisce, e segni linguistici significativi. Solo una volta fatto questo, saremo in grado di provare almeno a indovinare *quale* calcolo specifico il sistema effettua.

La corrispondenza suddetta è un'*interpretazione*, che fa corrispondere, che "mappa" le configurazioni di ingresso e uscita a segni significativi della nostra lingua. Se si applica questa condizione al terzo esempio sopra e interpretiamo i simboli delle stringhe apparentemente prive di significato, riconosciamo che il sistema sta probabilmente eseguendo moltiplicazioni: basta sostituire "3" a "0011", "5" a "0101", "15" a "1111", "4" per "0100", "3" a "0011", "12" a "1100": questo è, tra l'altro, un ovvia interpretazione, basata sulla numerazione in base 2, ma senza una tale interpretazione, il calcolo svolto dal sistema non avrebbe probabilmente avuto molto senso.

L'interpretazione è una corrispondenza, *mappatura*, che è un'operazione *algoritmica* che, date alcune configurazioni di ingresso, produce altri simboli. Questo non è qualcosa che riguarda il problema dell'intenzionalità intesa come problema filosofico. N oi operiamo un'*attribuzione*

[41] Sezione 17.

intenzionale sul set di configurazioni *derivante* da questa mappatura, configurazioni che devono essere state scelte, nelle loro proprietà formali, come in grado di *significare* qualcosa per noi. Una volta stabilita la mappatura, possiamo operare una attribuzione intenzionale più globale, e dire *quale* calcolo il sistema nel suo complesso sta eseguendo. A mio avviso, solo allora il sistema può essere visto come *computazionale*. La computazione è *attribuzione* di computazione a sistemi che, *di per sé*, sono semplicemente sistemi dinamici discreti governati da regole di.[42]

Questo punto di vista "intenzionale", o "semantico" della computazione ("intenzionale" o "semantico", con tutte le precisazioni di cui sopra) non è una nuova posizione, poiché è stato adottato da vari autori, in particolare Jerry Fodor. Questo punto di vista semantico è avversato da altri, come Gualtiero Piccinini, che vedono il calcolo come definibile in termini puramente meccanicistici, senza la necessità di ricorrere ad alcuna attribuzione semantica[43].

Qui sorge un problema: dato che l'attribuzione della computazione dipende dalla scelta di una corrispondenza, significa forse questo che *qualsiasi* macchina, una volta realizzata una mappatura adeguata tra i suoi stati e simboli significativi, può essere vista come qualcosa che effettua una *computazione*?

Ebben, alcune condizioni devono essere soddisfatte: primo, la macchina deve essere digitale: stiamo parlando di computazione *digitale* qui (il paradigma dominante della computazione, almeno a partire dagli anni '50). Quindi, per considerarsi computazionale, una macchina deve essere almeno considerata digitale, cioè deve possedere, e operare su un insieme finito di possibili stati stabili e distinti. Sono questi gli stati che la mappatura collegherà ai simboli. Ma, affinché ciò sia possibile, questi stati devono essere robusti e distinguibili, e una regola deterministica deve governare la transizione tra configurazioni stabili di questi elementi. Non tutti i sistemi sono in grado di soddisfare tali condizioni non banali (pensiamo a configurazioni stabili distinte in un fluido turbolento). Ma, all'interno della concezione di modularità che ho proposto qui, è facile vedere che tutto ciò che è richiesto è che il sistema sia un sistema discreto dinamico (o *DDS*), cioè una forma di sistema funzionalmente modulare (vedi Sezione 19.1.6).

Il punto centrale è questo: per permettere di attribuire ad un sistema l'esecuzione di un calcolo, le seguenti condizioni devono essere soddisfatte:

a. un corrispondenza algoritmica (un mapping) tra simboli linguistici e possibili configurazioni di ingresso e di uscita del sistema deve essere stabilito;

b. dobbiamo essere in grado di dire *quale* è la funzione particolare che connette configurazioni di input a configurazioni di output, cioè quale è la *specifica* della computazione, per dire ciò che il sistema sta calcolando.

Solo nel caso in cui il sistema sia un DDS e queste due condizioni siano soddisfatte, il sistema può essere visto come effetuante una *computazione*. Solo allora saremo nella posizione di dire che un sistema sta eseguendo un calcolo.

L'ordine delle condizioni *a* e *b* sopra riportate è invertito rispetto a come un programmatore umano opera. In questo caso c'è bisogno non di un'interpretazione, ma dell'istituzione di una

[42] Naturalmente può essere sollevato un problema qui: se il sistema è già considerato come governato da regole, significa che un'attribuzione intenzionale originale è già stata operata. Ma è al di là degli scopi di questo lavoro affrontare qui questa e altre simili questioni spinose, analoghe al famigerato problema "kripkensteiniano" del seguire una regola.

[43] Si veda ad esempio Piccinini (2008).

norma, una *specifica*: nel caso della programmazione, dapprima la specifica (punto *b*) è arbitrariamente *scelta*, è considerata la norma secondo la quale il programma dovrà operare, e *sulla base della specifica*, il programmatore sceglie il *mapping* (punto *a*) da simboli a configurazioni di ingresso e di uscita che ritiene migliore, per poi procedere alla scrittur, all'*implementazione* del programma, che è la specificazione delle parti e della struttura del meccanismo (programma) che sarà, alla fine, in grado di realizzare la specifica scelta in funzione della mappatura prescelta. Così, la scelta della mappatura determina la scelta della *particolare* struttura del *programma*. Tutto questa serie di operazioni costituisce l'*implementazione* della specifica scelta.

Parlando di *implementazione* sulla falsariga di Galton (1993), e Partridge & Galton (1995), ritengo che la relazione specifica-attuazione sia una relazione tendenzialmente universale: una *implementazione* è l'atto di specificare un metodo per "realizzare"[44] una certa specifica generale. Quando si considera un programma, non c'è, tuttavia, un *unica* specifica generale e un unico livello di attuazione, perché le due nozioni sono *relative*, esattamente come quelle di livello di descrizione "superiore" e "inferiore" e quella di *funzione*, che[45] è il ruolo parziale che qualcosa svolge relativamente alla realizzazionre di una funzione globale. *Relativo* in questo caso significa che qualcosa che è l'implementazione di una specifica può a sua volta essere considerata una specifica di livello inferiore da realizzare a un livello ancora più basso. In altre parole, data una specifica vi è la necessità di trovare una sua possibile implementazione, e, nello stile di programmazione strutturato o modulare, tale implementazione sarà essa stessa scomponibile in moduli. Ogni modulo, realizzando una specifica funzione input/output[46] costituisce, a sua volta, una specifica, che sarà implementata ad un livello inferiore, e così via.

Sembra ragionevole pensare che la stessa *gerarchia* astratta multi-livello , in cui ogni macro-componente è multirealizzabile da parte di sotto-componenti, e così via, sia alla base delle nozioni di programmazione strutturata, decomposizione funzionale, e livelli modulari gerarchici di descrizioni.

Come detto, un programmatore inizia con una specifica e cerca di implementarla. Ma si può iniziare con un processo discreto *non interpretato*, con l'obiettivo di *scoprire* quale computazione, nel caso, e in che modo, il processo esegue. Questo è un percorso bottom-up, ed è tipico della*retroingegnerizzazione* di una computazione. Il percorso top-down consistente nell'iniziare con una specifica di un calcolo e cercare di decomporre ricorsivamente tale specifica, che è una funzione, in sottofunzioni, e a loro volta queste in sottofunzioni più semplici, e così via, per descrivere il *modo in cui* la specifica si realizza, è invece il percorso della *spiegazione* computazionale, che è la spiegazione funzionale tipica della psicologia cognitiva, dove la specifica, che è la facoltà cognitiva (l'equivalenza cognizione / computazione è il principio di base della psicologia cognitiva), è spiegata in termini di una rappresentazione funzionale gerarchica.

19.4.8 Antimodularità, automi cellulari e spiegazioni computazionali

Possiamo ora rivolgere la nostra attenzione al problema delle conseguenze dell'antimodularità sulla *spiegazione computazionale*. Come detto, questo è il tipo di spiegazione tipica delle scienze cognitive. Tuttavia, non voglio provare qui a vedere quale impatto antimodularità potrebbe avere su queste scienze, ma solo come essa potrebbe influenzare un "modello giocattolo": prendo come modello di sistema computazionalmente capace un automa cellulare, al fine di vedere come il suo comportamento dinamico può essere spiegato come qualcosa che esegue un calcolo. L'idea

[44] In un senso strettamente simile a quello del concetto di *realizzazione* in filosofia della mente. Non intendo però esaminare tale nozione qui.
[45] Vedi sezione 19.4.3.
[46] Funzione ingresso-uscita in senso matematico. Vedere. sezione 9.

è che, se il CA è antimodulare nel suo comportamento dinamico, l'operazione potrebbe essere ostacolata o addirittura resa impossibile. Se questo è il caso, questo significa che l'antimodularità potrebbe avere un impatto sulla possibilità di spiegazione computazionale.

Consideriamo quindi il caso di cercare di spiegare computazionalmente un CA. Due domande fondamentali possono essere poste al riguardo:

- Un CA è un sistema computazionale?
- Se lo è, come possiamo spiegare la computazione che esso esegue?

Per quanto riguarda il primo punto, naturalmente, un CA, essendo un sistema dinamico discreto, rispetta le condizioni indicate nella sezione precedente di questo capitolo, e può essere considerato in modo sicuro un sistema *potenzialmente* in grado realizzare una computazione. Ma per dire *che* esso *calcola*, bisogna poter dire, in qualche modo, *che cosa* calcola, *qiale* calcolo esegue: nessuna spiegazione computazionale può essere applicata prima che una mappatura venga stabilita tra le configurazioni del CA e dei simboli significativi per noi. Potremmo provare a mappare lo stato delle sua cella a qualche simbolo sensato, per esempio, potremmo mappare questi stati a "nero" e "bianco", ma questo porterebbe a spiegazioni "computazionali" di questo tipo: "in base all'applicazione ripetuta della sua regola, il CA produce una variazione progressiva dello stato delle sue cellule, che possono, sotto condizioni diverse, cambiare da bianche a nere". Questa non mi sembra una spiegazione molto utile. Una spiegazione completa richiede una specifica più significativa e trasparente, cioè la capacità di dire *ciò che* il sistema calcola. Qui, la specifica è troppo vaga: "progressiva variazione dello stato delle sue cellule in varie condizioni". La specifica esatta del CA, intesa come funzione di ingresso / uscita, è data dall'applicazione ripetuta della sua regola, per cui una specifica ragionevole potrebbe essere fornita in termini della *regola* del CA. Ora, riguardo a questo, si presenterebbero alcuni problemi: di solito, la maggior parte delle regole CA sono descritte nei termini di una "tabella di corrispondenza", cioè un elenco estensionale di come la regola determini il valore di una cella al passo successivo della dinamica del CA sulla base dei valori delle cellule vicine. Tale tabella diventa incontrollabilmente grande come all'espandersi dell'"area di vicinato" (il *neighborhood*) del CA. Così, per alcuni CA, l'esibizione di questa tabella sarebbe impossibile, o comunque renderebbe completamente senza senso una spiegazione computazionale che la esibisca. In CA a due valori, la loro regola può essere vista come una specifica espressione booleana. Quindi, si potrebbe pensare di semplificare la sua descrizione nella forma di una espressione booleana. Questo, tuttavia, è quasi certamente un compito computazionalmente intrattabile, quindi, non può essere garantito riuscire in tutti i casi in tempi ragionevoli.

Tutto questo ci suggerisce che dovremmo cercare di trovare spiegazioni *di livello superiore* , possibilmente spiegazioni computazionali *multilivello*, in modo da ottenere una spiegazione più utile. In altre parole, si deve trovare un modo per vedere un CA descritto ad un livello superiore rispetto a quello delle sue celle elementari come una macchina computazionalmente capace. Per ottenere ciò, potremmotentare di vedere se la dinamica del CA è in grado di produrre certi tipi di moduli dinamici, cioè, strutture di alto livello abbastanza persistenti, il cui comportamento al livello più alto possa essere visto come governato da regole, al fine di soddisfare la condizione di vedere le dinamiche CA a questo livello superiore come un altro sistema dinamico discreto, un sistema a livello più alto, che può essere visto come diverso dal DDS costituito dal CA e la sua regola. In altri termini, per ottenere una spiegazione computazionale utile di un CA, una prima condizione è (*i*) che una forma di modularità dinamica a un alto livello di descrizione possa essere rilevata in modo affidabile nelle dinamiche globali del CA. E un'altra condizione deve valere: (*ii*)

la dinamica modulare di alto livello deve essere capace di *rispecchiare* (di "tracciare") con successo le dinamiche di basso livello del CA, senza divergere da esse. Questa condizione di *validità* (per usare la terminologia della simulazione scientifica tramite computer) è una condizione piuttosto complessa, ed è meglio specificata nelle sezioni 2.2.1 e 6.6.8 di questo lavoro, ma essa equivale sostanzialmente a questo: che la dinamica della descrizione ad alto livello non diverga nel tempo dalla dinamica corrispondente del CA al livello più basso di descrizione, quella delle cellule.

Risulta che alcuni CA sono in realtà dotati di una forma di livello modularità robusta di alto livello: come abbiamo visto (vedi fig. 19.2), alcuni CA possono generare *alianti*, che finiscono per realizzare, in molti casi, interazioni prevedibili l'uno con l'altro, come nel caso della *Regola 54*[47], e queste interazioni prevedibili possono essere viste come implementazioni di alto livello di funzioni booleane, con gliders che agiscono come "bit" viaggianti. Questa interpretazione in termini di funzioni booleane potrebbe poi permettere di costruire progressivamente una spiegazione multilivello in termini di calcoli sensati, allo stesso modo in cui i programmi per computer possono essere descritti da linguaggi di programmazione di livello progressivamente superiore. In questo modo avremmo costruito una parte delle condizioni necessarie per spiegare un CA mediante una spiegazione *computazionale*.

Tuttavia, tale interpretazione in termini di gliders non è sempre possibile: ci sono certi CA "caotici", come la regola 30 (vedi fig. 19.7), che *non mostrano mai* subconfigurazionni abbastanza robuste per essere considerate moduli dinamici in grado di rendere la rappresentazione di alto livello una rappresentazione computazionalmente capace[48].

Un punto deve essere evidenziato qui: questa impossibilità di individuare moduli dinamici stabili in un CA, come nel caso della Regola 30, può essere visto come una forma di *antimodularità intrinseca* della descrizione ad alto livello del CA. Quindi possiamo dire che l'antimodularità, in questa forma, già impedisce il primo passo, passo (*i*) di cui sopra, richiesto per fornire una spiegazione computazionale, una passo che consiste nel considerare il CA come computazionalmente capace ad un livello di descrizione elevato. Così, sembra che, almeno in questa forma, *l'antimodularità intrinseca effettivamente impedisca la spiegazione computazionale*.

Tuttavia, è provato che, per certi CA, la loro interpretazione di alto livello come sistemi computazionali è possibile: c'è una mappatura complessa, ideata da Matthew Cook[49], con la quale egli ha saputo dimostrare che *Regola 110*, un'altra regola CA elementare, può essere vista come un sistema computazionale sul piano della macchina di Turing universale. Anche il più famoso CA, il *Game of Life* ideato da John Conway , è stato dimostrato essere Turing completo[50]. Quindi, è un fatto dimostrato che, sotto certe interpretazioni, alcuni CA possono essere visti come computazionali: questa prima condizione può essere vista come stabilita, almeno per certi CA.

Ma, se vogliamo dare una /spiegazione computazionale* di un sistema, un'altra condizione deve essere soddisfatta: che il sistema stia realmente effettuando una *computazione*, e non solo, che sia computazionalmente capace, come sono tutti i sistemi digitali. E, a questo scopo, dobbiamo prima essere in grado di dire *che cosa* esso stia calcolando: cioè, dobbiamo essere in grado di esprimere la relazione input/output, la sua *specifica*. Dobbiamo notare che stiamo lavorando

[47] Vedi per esempio Martínez et al. (2014).

[48] Perché non possiamo inventare una mappatura da set di configurazioni caotiche a simboli significativi, in questo modo rendendo anche un CA caotico computazionalmente capace ad alto livello? Una risposta implica una discussione sulla complessità della mappatura tra configurazioni di sistema e simboli, una discussione che si sviluppa nella sezione 14.5.2.

[49] Vedi Cook (2004).

[50] Cfr. Rendell (2002).

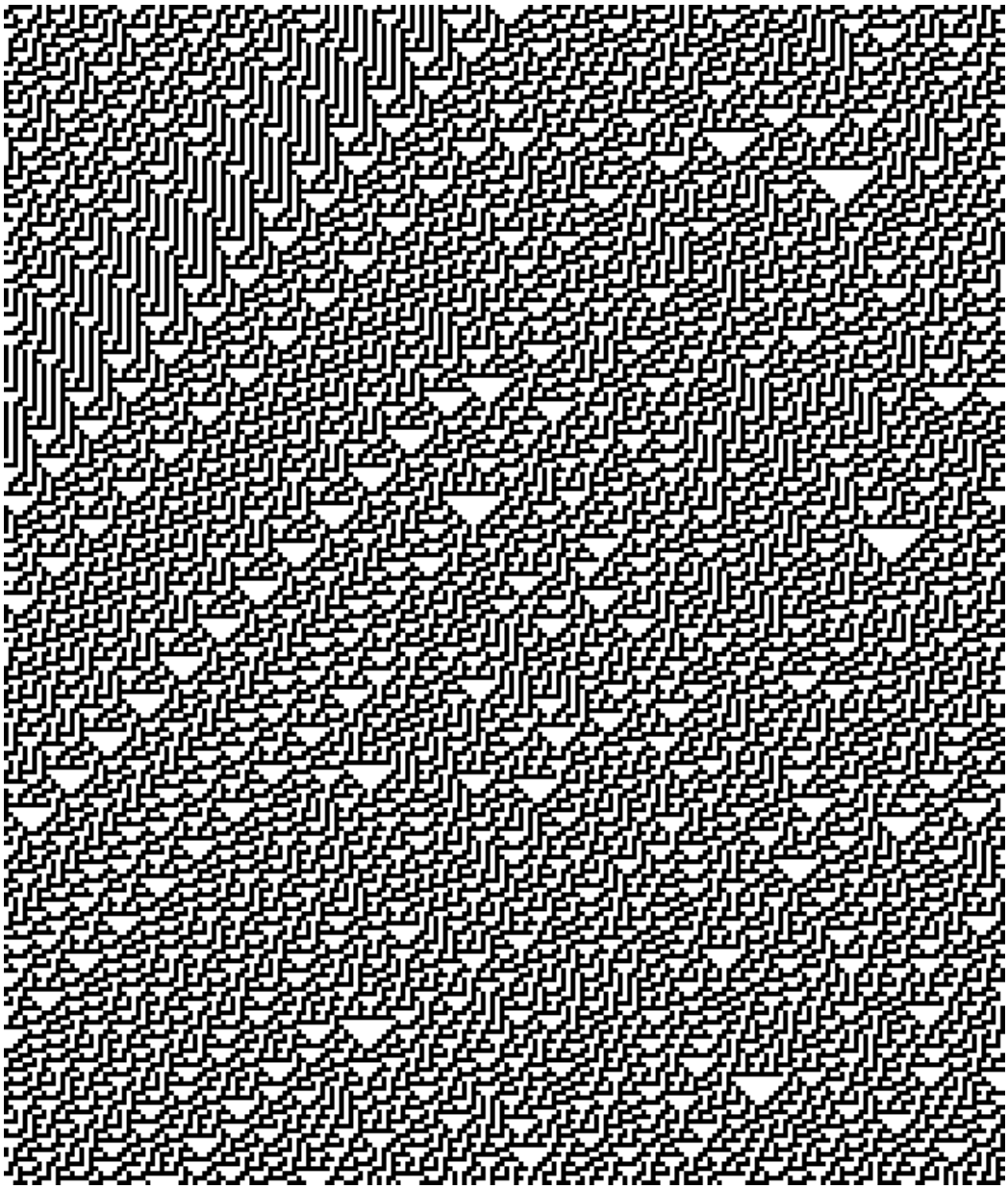

Figure 19.7: segmento caotico dell'evoluzione del *CA elementare Regola 30*. Il tempo scorre dall'alto verso il basso, ogni riga di pixel rappresenta la configurazione globale del sistema ad ogni passo temporale. Ogni pixel rappresenta lo stato di una delle parti elementari della CA, le sue *cellule*.

qui nel campo della *ingengeria inversa*: abbiamo una macchina, il CA, che sappiamo essere computazionalmente capace, e dovremmo, al fine di spiegarlo computazionalmente , *trovare* la specifica di tale computazione.

Ora, il compito di retroingegnerizzare la specifica di un programma, apparentemente è arduo: sostanzialmente, si tratta di produrre tutti gli ingressi di un programma e di osservare tutte le uscite corrispondenti: richiede una forma di *induzione*. Lasciando i dettagli più fini ad un'altra occasione, semplicemente bisogna notare qui che vi è una serie di problemi legati al fatto che i sistemi computazionalmente capaci a livello della macchina di Turing sono affetti dal problema della fermata, e questo rende il compito sopra indicato molto difficile se non impossibile: a causa del fatto che il numero di possibili coppie di ingresso/uscita da osservare cresce esponenzialmente con la dimensione massima (in bit) dell'ingresso, questo è perlomeno un compito intrattabile dal punto di vista computazionale (vedere paragrafi 14.5.2 e 17.2.6 per i dettagli). Ci sono algoritmi approssimati per l'inferenza induttiva della specifiche (algoritmi per "specification mining", vedere la sezione 4.3.1.1), che però danno risultati spesso troppo approssimativi, e non sono in grado di retro-ingegnerizzare specifiche di computazioni di livello Turing.

Ma noi avevamo bisogno della specifica del programma al fine di spiegarlo computazionalmente. E questa specifica è molto difficile da scoprire.

Tuttavia, una specifica in termini di mera funzione di ingresso/uscita non è l'unico modo in cui una specifica può essere data, e, anche se è il più preciso, non è nemmeno il modo più perspicuo, perché un elenco di coppie ingresso/uscita può risultare privo di significato. Quindi, un'altra forma, più perspicua in cui può essere data una specifica, è in forma *aggregata* : un modo più o meno sintetico di riassumere l'intera funzione di ingresso/uscita. Un modo di ottenere questo consiste nel dare la specifica in termini della sua decomposizione in sottofunzioni, il che è una forma di scomposizione gerarchica.

Ora, il punto interessante è questo: se una *rappresentazione modulare gerarchica* della computazione da spiegare potesse essere trovata in qualche modo, sarebbe allora possibile testare ogni modulo *separatamente* nella ricerca della specifica di *solo* quel modulo, un compito che molto probabilmente risulterà essere più fattibile *di ordini di grandezza* rispetto a quello di presentare ogni ingresso possibile all'intero programma, al fine di dedurre direttamente la specifica globale. Questo accade perché un modulo è identificabile per il fatto stesso che esso dovrebbe essere collegato scarsamente o in maniera tenue agli altri moduli, e questo si traduce in una probabile riduzione del numero di possibili ingressi del modulo, e conseguente a un'esplorazione più facile dello spazio degli input[51].

Il fatto che sia stato possibile trovare la singola specifica di ciascun modulo per merito della scomponibilità del sistema, permette auspicabilmente, se la specifica di ciascun modulo non è troppo complicata, una forma di *aggregazione*, come discusso nella sezione 4.1.5.1: se siamo in di rendere più astratta la specifica del modulo "denominandolo" in un modo succinto significativo, dandogli un nome rappresentativo ed esplicativo della funzione che svolge (ad esempio quando si dice che un modulo esegue l'operazione di "moltiplicazione"), allora la specifica di ciascun modulo può essere sostituita da questa definizione più concisa di ciò *quale* funzione il modulo effettua. A quel punto una specifica globale dell'intero sistema può essere data *in termini* di una descrizione (di solito nella forma grafica di un * diagramma di flusso *) della struttura modulare del sistema come una rete diretta di moduli collegati, in cui i moduli sono visti come nodi etichettati con i loro "nomi" succinti che rappresentano le loro specifiche e le loro connessioni di ingresso e di uscita sono i collegamenti diretti tra i nodi.

[51] Anche se in realtà questo non è *garantito*. Vedere la sezione 4.3.1 per una discussione ulteriore.

Quindi, questo tipo di spiegazione sembra possibile, dopo tutto. Ma esso *richiede che una modularità funzionale della computazione possa essere trovata*, e questo, a sua volta richiede due condizioni: primo, che il sistema sia effettivamente computazionalmente capace ad alto livello, e ciò non è garantito: sistemi *intrinsecamente antimodulari*, come il CA *Regola 30*, citato sopra, non sono suscettibili di essere visti come computazionalmente capaci ad alto livello. In secondo luogo, un'altra condizione è che, anche se il sistema è computazionalmente capace, ed effettivamente possiede modularità dinamica, questa modularità *possa essere effettivamente trovata*. Questo potrebbe essere ostacolato da alcuni fattori, dovuti all'elevato costo computazionale degli algoritmi per il rilevamento della modularità, o all'eccessiva approssimazione dei risultati che si ottengono, quando questi possono essere ottenuti in un tempo ragionevole: le descrizioni gerarchiche ottenute potrebbero non rispecchiare in maniera sufficientemente fedele l'effettiva organizzazione modulare funzionale del sistema, da essere considerare descrizioni modulari *valide*, in grado di caratterizzare sufficientemente bene il calcolo eseguito.

Sembra, quindi, che *l'antimodularità possa ostacolare o impedire anche la spiegazione di tipo computazionale*.

Ma, potrebbero ricostruzioni *parziali* della gerarchia funzionale del programma essere usate nelle spiegazioni? Ebbene, sembra, intuitivamente che i modelli funzionali così prodotti sarebbero molto limitati da clausole *ceteris paribus*, aventi lo scopo di tenerli all'interno della gamma di rapporti di ingresso/uscita *noti*, e, tra questi, nell'intervallo di quelli che non divergono troppo, per mancanza di validità del modello modulare, dal comportamento reale osservato del sistema. Così, sembra che una spiegazione basata su di tali ricostruzioni parziali della gerarchia modulare dovrebbero essere limitate nella loro applicabilità. Esse potrebbero apparire come una spiegazione accettabile, ma risulterebbero in un certo senso una spiegazione *post-hoc* della gamma di comportamento del sistema effettivamente osservata durante il processo di rilevamento della modularità, e non di tutti i suoi possibili comportamenti.

Può ben darsi che, nelle scienze cognitive computazionali, un tale tipo di spiegazione limitata potrebbe essere accettata, e, inoltre, è probabile che ci siano *solo* spiegazioni di questo genere in alcuni sottocampi della psicologia cognitiva. In tale scienza, il compito di trovare le specifiche e le relazioni funzionali tra i moduli, è lasciata alla sperimentazione umana, il che è più probabilmente un processo più lento di quelli algoritmici automatizzati.

Per concludere questa sezione, penso che anche questo fallimento nella ricerca di spiegazioni computazionali di alcuni sistemi possa essere considerata una forma di *emergenza antimodulare*. Questa emergenza computazionale per quanto riguardante le spiegazioni computazionali può essere vista come dovuta a due forme di antimodularità: prima di tutto, l'*antimodularità intrinseca* che interess i sistemi caotici, che non possono nemmeno essere considerati computazionalmente *capaci* a un livello superiore di descrizione, e, secondo, l'antimodularità dovuta all'eccessivo costo computazionale o all'eccessiva approssimazione degli algoritmi di rivelazione della modularità.

La conseguenza di questa forma di emergenza antimodulare è che il sistema interessato da essa è spiegabile soltanto al suo livello più basso di descrizione, e questo in generale non costituisce una forma intelligibile di spiegazione, in sistemi sufficientemente complessi. E c'è da aspettarsi che l'emergenza antimodulare possa influenzare spiegazione anche la spiegazione computazionale nelle scienze cognitive e nelle neuroscienze, e questo meriterebbe, mi sembra, ulteriori indagini.

19.4.9 La modularità di alto livello come condizione per la programmazione in informatica e per la ricerca scientifica

Almeno per un programmatore informatico, l'affermazione di Robert Cummins, che incontreremo nel capitolo 9.2, che l'analisi funzionale ha una capacità esplicativa, non è una novità: un programmatore, durante la sua attività, è, almeno implicitamente, immerso un'attività continua di sviluppo di spiegazioni *parziali* di come il programma in costruzione funzioni mediante l'esecuzione organizzata delle sue istruzioni, e, ad un livello superiore, delle sue subroutine o di moduli di livello ancora più elevato. L'atto stesso della programmazione inizia dalla *specifica* dell'intero programma (per esempio, essere "un word processor"), e lo sviluppo procede analizzando, in un senso à la Cummins, questa funzione globale, che è la specifica, in sottofunzioni più semplici che insieme ne costituiscono l'implementazione. A sua volta, ogni sottofunzione viene decomposta, se possibile, in sottofunzioni più semplici, e così via. La scrittura vera e propria del programma, l'atto di scrivere le sequenze di istruzioni che compongono ciascun sottoprogramma, di solito rimbalzando su e giù tra i vari livelli gerarchici, analizzando funzioni globali in subroutine più semplici, la loro implementazione e il tornare a decomporre altre funzioni di alto livello, o a comporre dal basso partendo dalle subroutine semplici e salendo nei livelli gerarchici, è quasi un atto irrealizzabile senza un precedente, almeno implicita, *spiegazione*, da parte del programmatore stesso, di tutto il sistema in termini gerarchici. Questa può essere visto come una forma di *spiegazione funzionale*alla maniera in cui Cummins la intende. Sembra, quindi, che la spiegazione gerarchica funzionale sia necessaria per la *programmazione di computer*, in informatica.

Nella scienza empirica, la spiegazione funzionale multilivello è probabilmente essenziale non solo *dopo* che una teoria o il modello di un fenomeno è stato stabilito, cioè nella fase della spiegazione di un fenomeno già noto, ma anche nella *teorizzazione*, nella fase di *ricerca di un modello*. * Concezioni *intervenzionistiche* della causalità come quella di James Woodward[52] vedono una relazione causale come sussistente tra due entità nel caso in cui una variazione ipotetica dello stato di un'entità, volutamente indotta da uno sperimentatore, cioè un *intervento*, produca sistematicamente una variazione nello stato dell'altra entità: quando questa circostanza ipotetica è valida, si può dire che le due entità siano in relazione causale. Una concezione meccanicistica della spiegazione richiede durante la ricerca scientifica l'avanzamento e la rifinitura della descrizione di un meccanismo scoprendo progressivamente tutte le relazioni causali che sussitono tra le sue parti. A tale scopo, lo sperimentatore procede intervenendo su ciascuna parte separatamente, per vedere se qualche conseguente variazione si verifica su altre parti. Ma, per identificare correttamente i legami causali, l'intervento sullo stato di una parte richiede una temporanea, almeno virtuale, disgregazione della struttura dei collegamenti causali che vanno da altre parti del meccanismo verso la parte su cui stiamo intervenendo: interventi sui meccanismi richiedono che il meccanismo sia temporaneamente *modificato* eliminando alcune delle connessioni tra le sue parti. Woodward sostiene che l'insieme di equazioni che rappresentano correttamente un sistema causale deve essere *modulare*, perché altrimenti, dato che il rilevamento di una relazione causale richiede un intervento su una parte del sistema, interrompendo momentaneamente solo l'influenza causale che va in direzione di quella certa parte, se il sistema fosse completamente non modulare, questa specifica disconnessione di un percorso causale potrebbe perturbare non solo la parte dell'equazione interessata dall'intervento, ma anche altre parti del sistema. Quindi, la modularità dinamica è sempre presente in un meccanismo, almeno a livello più basso, quello della descrizione preferita.

Ma, il punto è, se vogliamo ridescrivere un sistema meccanicistico a livello più alto, potremmo

[52] Vedi Woodward (2003), e la sezione 6.9.

certamente interpretare le relazioni tra le parti di alto livello prima facie come "relazioni causali di alto livello". In tal caso, al fine di procedere all'intervento alla manier di Woodward, la modularità è necessaria anche nelle equazioni che rappresentano la dinamica del sistema *a questi livelli elevati*. Tutto considerato, questa condizione vale se la struttura della funzione di aggiornamento[53] è *gerarchicamente modulare*, e questo a sua volta rappresenta il fatto che il sistema è funzionalmente, e probabilmente anche, in modo dinamico, e, molto probabilmente, strutturalmente, *gerarchicamente modulare*.

La stessa utilità della presenza e della possibile individuazione di una struttura modulare si presenta nella fase di scoperta di reti complesse, in particolare nei casi in cui la scoperta di collegamenti tra i nodi richiede un lavoro sperimentale complesso, come nelle reti di regolazione dei geni, e in altre reti di interesse biologico. Alcuni metodi algoritmici proposti di recente, come quello in Clauset et al. (2008)[54], potrebbero essere di grande aiuto in questo tipo di compito, perché, sulla base della modularità gerarchica rilevata nella rete parziale già scoperta fino a quel momento, questi metodi possono produrre probabilisticamente, con buone possibilità di successo, una previsione su dove nella rete i collegamenti mancanti probabilmente si presenteranno a seguito di ulteriori osservazioni, guidando così proficuamente la successiva sperimentazione.

Quindi, sembra che la modularità gerarchica sia importante o addirittura indispensabile nella fase di ricerca scientifica e della*scoperta* sperimentale, oltre ad essere quasi essenziale, come abbiamo visto nelle precedenti sezioni, per la spiegazione di un fenomeno già studiato.

19.4.10 Emergenza esplicativa

Dato che la mancanza di comprensione dovuta alla presenza di emergenza antimodulare in un sistema è apparentemente in grado di influenzare la maggior parte dei tipi di spiegazioni, mi propongo di generalizzare la nozione con la seguente definizione:

> L'*emergenza esplicativa* è una proprietà di sistemi, o di loro descrizioni, che consiste nel fatto che, per ragioni computazionali di principio o pragmatiche, tali sistemi o descrizioni resistono alle spiegazioni *comprensibili*.

Questa è una definizione più generale di quella di antimodularità, una definizione comprendente altri possibili effetti di vincoli computazionali sulla spiegazione di sistemi complessi. Nella sezione 14.6, illustro la mia concezione della definizione di cui sopra, che non si applica necessariamente a compiti computazionali, ma anche a compiti riguardanti lo sviluppo di spiegazioni quando questi sono eseguiti da esseri umani, e la possibile utilità di questa definizione nell'attuale panorama scientifico.

19.4.11 È probabile incontrare sistemi antimodulari nella scienza?

Antimodularità sembra dipendere dalla *scelta* di una relazione, che è specificata nella descrizione preferita, tra le parti elementari del sistema. L'antimodularità può verificarsi quando, data questa relazione scelta, il rilevamento della modularità in base ad essa risulta essere troppo computazionale pesante per essere portato a termine in un tempo plausibile, o quando, seppur il rilevamento della modularità sia completato con successo mediante un algoritmo approssimativo, la descrizione modulare prodotta appare troppo approssimativa per essere in grado di rappresentare validamente il sistema originale.

[53] La funzione che regola la dinamica di un DDS:!. Vedi sezione 5.1.
[54] Descritto nella Sezione 6.9.

Qual è la probabilità che una di queste due circostanze si possano incontrare durante la ricerca scientifica? Va sottolineato che la complessità computazionale del rilevamento della modularità riguarda algoritmi per il rilevamento di modularità che non impiegano alcuna altra informazione sui sistemi diversa da quella inclusa nella descrizione preferita, cioè il livello delle loro parti elementari e le loro relazioni. Aggiungendo vincoli su come le parti elementari possono essere raggruppate in moduli, tale compito può essere fortemente semplificato. Ciò equivale a elaborare algoritmi ad hoc per il rilevamento della modularità, e gli algoritmi ad hoc potrebbero finire per essere meno computazionalmente impegnativi di un algoritmo generale. In realtà, in molti casi, questo sembra esattamente ciò che la scienza fa: si cercano vincoli empirici per aiutarci a scegliere tra le possibili teorie del mondo. Ciò solleva la aumenta la probabilità che il metodo scientifico sia in grado di produrre desfrizioni modulari intelligibili dei fenomeni.

Ma, dobbiamo chiederci se i nuovi sviluppi della scienza possano spostare l'attenzione su sistemi di una tale complessità che anche i vincoli noti, empiricamente trovati, su tali sistemi, potrebbero finire per essere troppo scarsi per permettere il completamento della rilevazione della modularità su tali sistemi.

Nel caso di sistemi biologici, possiamo effettivamente essere ragionevolmente sicuri che essi sono modulari, almeno a certi livelli. Ci sono molti argomenti, empirici e teorici, che favoriscono questa conclusione, argomenti trattati nella sezione 7.1. Tuttavia, ci possono essere sistemi biologici significativi, come ad esempio le reti di interazione del metabolismo cellulare, che possono finire per essere così enormi da produrre potenzialmente effetti di emergenza esplicativa o antimodulare a causa dell'elevato costo computazionale degli algoritmi per il data mining o il rilevamento modularità in relazione alle dimensioni del sistema. Un altro tipo di situazione in cui ci si può aspettare questa emergenza antimodulare, è il data mining alla ricerca di informazioni sui moduli effettuato sulla *letteratura* già esistente riguardante alcuni temi scientifici, di cui un esempio è presentato nella sezione 19.5.2.

19.5 Alcune riflessioni aggiuntive su modularità, metafisica, computazione, storia della scienza

Nelle sezioni soprastanti, ho delineato la struttura principale di questo lavoro, in cui cerco di riflettere sul concetto di modularità, sulla sua relazione con la descrizione, la spiegazione, il calcolo, la comprensione, e di delineare le condizioni nelle quali la modularità si manifesta o, al contrario, non può essere rilevata. Il mio obiettivo generale è quello di descrivere l'importanza di quello che potremmo chiamare un "modo modulare di pensare" per la conoscenza umana del mondo, e in particolare la grande importanza della modularità su gran parte della concettualizzazione scientifica, soprattutto nelle cosiddette "scienze speciali", che per lo più ricorrono a tipi di spiegazione, *meccanicistica* o *funzionale*, che sono i tipi modulari di spiegazione *per eccellenza*: per spiegare in questi due modi, è necessario trovare un modo per descrivere un sistema come modulare.

Ho poi focalizzato la mia attenzione in particolare su modi per rilevare la modularità nei grandi sistemi complessi, sottolineando il fatto che, purtroppo per la scienza, questi modi sono algoritmicamente complessi e per questo motivo non sono garantiti dare risultati utili. Quando un fallimento nel dare descrizioni modulari si manifesta, questo è quello che io chiamo un caso di *emergenza antimodulare*. Questo fatto ostacola, ed eventualmente impedisce, in alcuni casi, la possibilità di spiegazione scientifica e di comprensione di sistemi complessi abbastanza ampi e complicati da riuscire a sfuggire ad un'adeguata spiegazione modulare, sistemi che, a causa delle loro dimensioni, non possono essere afferrati cognitivamente nella loro descrizione non modulare,

di basso livello, che è la loro unica descrizione possibile. Ho chiamato questa condizione "emergenza esplicativa". Questo è l'obiettivo principale di questo lavoro, che considero un'opera di filosofia della scienza, focalizzantesi specialmente su discipline di derivazione biologica e sui loro metodi, e, allo stesso tempo, sulla computazione, vista come un metodo per la ricerca in queste discipline, ma, prima di tutto, come un quadro teorico promettente alla luce del quale cercare di ripensare alcuni enigmi filosofici classici, un tentativo che faccio quando cerco di riformulare come ridescrizioni computazionalmente fattibili ciò che è stato tradizionalmente concepito come spiegazione scientifica, e come una relazione *specifica / implementazione* una relazione tra livelli di organizzazione che è stata tradizionalmente vista come una relazione di costituzione.

Le domande di cui sopra, che costituiscono la spina dorsale della mia proposta in filosofia della scienza, sono più profondamente discusse, come previsto, nel corpo dell'opera. Vorrei comunque dedicare le prossimi due ultime sezioni di questo capitolo a riflessioni che toccano alcune domande che in qualche misura cadono al di fuori della portata principale di questo lavoro come l'ho appena delineata.

La prima (nella sezione 19.5.1) è una riflessione sulle possibili conseguenze di natura metafisica della mia presa di posizione sulla natura delle spiegazioni, che è una posizione eminentemente epistemica. Avvertivo il lettore che questo è al di fuori del campo di applicazione principale di questo lavoro, perché non voglio immergere profondamente il mio discorso sulla filosofia della spiegazione scientifica in un contesto metafisico: non sono un metafisico, né la mia presa di posizione è del tutto antimetafisica, ma probabilmente ho un'inclinazione verso l'essere sempre cauto quando si tratta di questioni fortemente metafisiche. Seguendo Kant, non posso fare a meno di guardare con un certo sospetto le affermazioni metafisiche forti, perché queste potrebbero a volte avere conseguenze molto gravi, e in genere non è molto chiaro quando esse possano essere considerati adeguatamente supportate. Per questo motivo, ho pensato di mantenere la riflessione principale filosofica sulla scienza, che costituisce l'obiettivo principale di questa tesi, più o meno libera da interferenze metafisiche esplicite (anche se credo che questo potrebbe essere giudicato un progetto molto discutibile, e probabilmente un progetto di dubbia fattibilità) .

Tuttavia, alcune conseguenze metafisiche dell'approccio generale da me adottato e dei risultati sui limiti computazionali della descrizione modulare che ho descritto, aprono, mi sembra, la porta ad un'affascinante anche se rischiosa possibilità di delineare una particolare posizione metafisica, e non voglio perdere qui l'occasione di provare a delineare brevemente questa posizione. Dedico quindi la sezione successiva a tale scopo "sperimentale". Lo schizzo qui presentato è nel migliore dei casi uno schizzo di massima. Ciò che non è ben ponderrato qui, sarà meglio lasciare per un'occasione futura.

Nella sezione che segue (sezione 19.5.2) mi prenderò la libertà di un'altra impresa potenzialmente affrettata e imprudente, quella di cercare di prevedere le possibili conseguenze storiche di lungo periodo della concezione della computazione applicata alla spiegazione scientifica che descrivo nella principale linea di questo lavoro. In questo caso, voglio lasciare questa riflessione fuori dalla spina dorsale filosofico-scientifica della tesi, non perché penso che la riflessione storica sia fuori luogo in un'opera di filosofia della scienza. Al contrario, in realtà: spesso, in molte parti dei capitoli principali, seguo, ove possibile, le linee di ricostruzione storica dei più noti dibattiti recenti che ruotano intorno alle principali questioni in gioco. Credo, infatti, che almeno un'esposizione almeno cronologica, se non storica, di idee e discussioni, sia eminentemente importante per la scrittura filosofica e la riflessione, anche in una disciplina per lo più analiticamente informata come la filosofia della scienza. Il motivo per cui ho lasciato queste considerazioni storiche fuori del campo di applicazione principale del presente lavoro non è quindi un disprezzo per la storia delle idee, della filosofia o della la scienza, ma la natura intrinsecamente intellettualmente rischiosa

di queste riflessioni, che diventerà evidente nel corpo della sezione ad esse dedicata. Tuttavia, considero queste riflessioni come dotate di un potenziale di ulteriore investigazione più rigorosa, che sarà meglio non ignorare, almeno per la mia ricerca futura.

19.5.1 Un tentativo metafisico: modularità come ontologia? Antirealismo vincolato

Come abbiamo visto, una caratteristica importante e necessaria di un modulo è la sua robustezza: un modulo è qualcosa che deve sopportare una serie di perturbazioni, e su una certa scala temporale esso deve persistere per una certa quantità di tempo. Un altro tratto distintivo dei moduli è il fatto che essi godono di una certa quantità di indipendenza dal resto del sistema e da altri moduli, e ciò è dovuto ad un almeno parziale *isolamento* del modulo, al suo possedere qualche forma di *confine* riconoscibile. Si tratta di proprietà che i moduli condividono con le *entità*, o con gli *oggetti*, cioè con quello che si possono plausibilmente considerare le componenti ontologiche di base del mondo. Vorrei suggerire che non si tratta di una semplice coincidenza. Penso che sia plausibile dire che il nostro sistema percettivo opera un "rilevamento di modularità* sui dati grezzi che incidono sul nostro corpo, al fine di produrre una descrizione del mondo in termini di entità o oggetti: gli oggetti che percepiamo sono i moduli prodotti da questo processo di rilevamento della modularità. E 'anche plausibile, credo, prendere in considerazione i limiti di questo rilevamento di modularità effettuato da ciascun organismo, alla luce dei limiti che abbiamo visto pregiudicare gli algoritmi di rilevamento della modularità. Naturalmente, ci sono alcune importanti differenze qui: in primo luogo, i sistemi percettivi *non sono* calcoli algoritmici seriali, come quelli attuate su computer standard, e potrebbero pertanto essere esentati dal manifestare la stessa complessità computazionale di questi algotritmi seriali. Tuttavia, se diamo per scontata una forma di determinismo fisico (almeno macroscopico) e la finitezza dei sistemi biologici percettivi, l'equivalenza computazionale di quei processi percettivi con qualche algoritmo dovrebbe essere in line di principio abbastanza garantita: i sistemi percettivi possono essere visti come sistemi computazionali. Bisogna poi considerare che gli algoritmi appartenenti a certe classi di complessità sono *intrinsecamente intrattabili*, indipendentemente dall'architettura computazionale su cui sono eseguiti: per esempio, anche se si utilizzano architetture altamente parallele, come quelle delle reti neurali, i problemi EXPTIME non possono essere affrontati con successo , perché il tempo richiesto per il loro completamento cresce esponenzialmente, mentre la quantità di parallelismo può crescere al più linearmente[55], mentre la funzione esponenziale cresce incomparabilmente più veloce di qualsiasi polinomio. Un'ulteriore obiezione può tuttavia essere sollevata: i sistemi percettivi *non hanno bisogno* di *scoprire* una buona struttura modulare in un ammasso inizialmente non strutturato di stimoli, perché essi sono già messi a punto per rilevare una struttura modulare più o meno nota in un mondo che è più o meno stabile: un organismo è adattato a rilevare alcuni tipi di oggetti intorno ad esso, e il suo sistema percettivo è già strutturato e predisposto alla ilevazione di *quei* tipi di moduli, cioè quei tipi di oggetti nel mondo. Questa messa a punto è stata, da un punto di vista classicamente darwinista, prodotta dalla selezione naturale. Tale processo di rilevamento della modularità preadattato potrebbe essere molto *meno* computazionalmente complesso del processo grezzo consistente nel dover trovare una buona struttura modulare *inizialmente sconosciuta* in un grande insieme di dati non strutturati, un processo che richiede lo svolgimento di un'attività NP-completa, cioè a dire il compito di ottimizzare la misura modularità Q[56]. Questo è perlomeno probabile: i processi percettivi non

[55] o, eventualmente, in maniera quadratica, o cubica, se immaginiamo qualche futuristico "cubo crescente" computazionale tridimensionale.

[56] Questa è una misura della qualità della modularità rilevata da un algoritmo. Ottimizzare significa scegliere la migliore di tutte le possibili descrizioni gerarchiche di un sistema. Vedere la sezione 3.2.1.2.

sono probabilmente così computazionalmente difficili. Vi sono, tuttavia, due repliche. La prima è che i processi percettivi non sono meno vincolati dalla complessità computazionale, almeno indirettamente, di quanto lo sia l'ottimizzazione della modularità: i processi percettivi come sono ora, sono stati raggiunti attraverso la selezione naturale (almeno per la concezione darwinista), ed è *questo* processo, il processo di selezione naturale, ad aver portato il peso di cercare di ottimizzare la misura di modularità Q su un mondo inizialmente percettivamente non strutturato. La NP-completezza in alcuni casi però è così forte che nemmeno la selezione naturale può essere reputata aver avuto abbastanza tempo per eseguire una ricerca esaustiva nello spazio fenotipico dei possibili sistemi percettivi. Quindi, si potrebbbe sostenere che i sistemi percettivi attuali che derivano dall'evoluzione sono *indirettamente* stati colpiti da questi stessi limiti computazionali, dovuti all'eccessiva complessità computazionale del compito di ricerca della migliore descrizione gerarchica modulare, perfino della sua ricerca *durante il corso dell'evoluzione*. La seconda risposta alla precedente obiezione deriva da quest'ultima considerazione: molto probabilmente, a causa della loro bassa complessità computazionale, i processi percettivi sono *meno accurati* rispetto al compito algoritmico di ottimizzazione di Q, e questo è un altro modo per dire che il rilevamento della modularità ottimale su dati provenienti dal mondo empirico *non è* quello che i sistemi percettivi realizzano: la percezione è indirettamente vincolata dalla intrattabilità computazionale del rilevamento della modularità, nel senso che essa è stata resa più o meno inaffidabile da questa intrattabilità.

Un altro suggerimento in questo senso è il fatto che in una concezione interventionistica della causalità come quella di James Woodward, che è stata adottato come posizione standard da alcuni sostenitori della spiegazione meccanicistica, ad esempio Carl Craver, la modularità delle equazioni dinamiche che governano il regime del sistema è necessaria per separare le variabili su cui intervenire durante la sperimentazione per discriminare cause singole. Dunque, la decomposizione meccanicistica di un fenomeno dipende dalla possibile modularizzazione dei suoi stati globali. Naturalmente, l'identificazione delle parti del meccanismo, o entità, che sono gli elementi causalmente attivi di un meccanismo, dipende la proposizione dal successo nell'effettuare questa valutazione causale portata avnti pezzo per pezzo. Così, sembra che la stessa ontologia di un meccanismo, l'insieme delle sue *parti*, dipenda dalla modularizzabilità della sua dinamica. Quest'ultima considerazione, alla luce della durezza computazionale dell'ottimizzazione della modularità, ci dice che, molto probabilmente, la scienza sperimentale, non è e non sarà in grado di produrre la *più plausibile* ontologia del mondo, non solo se essa è condotta passo-sperimentale dopo passo-sperimentale da parte di soggetti umani, ma anche se venisse completamente automatizzata: l'intrattabilità computazionale dell'ottimizzazione di Q o dell'aggregabilità delle variabili nei modelli dinamici[57] è impossibile da superare. Sembra abbastanza probabile quindi, che l'ontologia dei meccanismi scoperti dalla scienza non sia la migliore ontologia possibile.

Ci si potrebbe chiedere dove questa singolare visione della modularità come ontologia si posizioni lungo la linea realismo-antirealismo, e non solo nel senso normale di antirealismo su entità inosservabili postulate da teorie scientifiche, ma nel senso più ampio, onnicomprensivo, di mettere in discussione la *realtà* anche degli oggetti macroscopici di medie dimensioni. Oltre tutto, se il nostro scomporre il mondo in pezzi sensati è un effetto della possibilità di rilevazione della modularità, e se questa possibilità dipende a sua volta da limiti di calcolo, è legittimo chiedersi da dove questi vincoli derivino. Tuttavia, non è nel campo di interesse di questa tesi l'approfondire discussioni metafisiche profonde. Così, il massimo che possiamo fare qui è ammettere che se identifichiamo moduli la cui rilevazione è fattibile tenuto conto di certi vincoli derivanti dalla complessità computazionale, allora potremmo dire che questo punto di vista è una forma di quello che chiamerei *antirealismo vincolato*: si tratta di un posizione che può essere certamente

[57] Vedi le sezioni 19.1.5 e 2.2.1.

considerata antirealistica, perché, a suo avviso, non si sa quali entità sono *reali* di per sé. Meglio, da questo punto di vista, la questione potrebbe non avere nemmeno molto senso: la realtà delle entità, cioè il loro essere dotate di robustezza e confini, è il *risultato* del processo di rilevamento della modularità, che è limitato dalla complessità computazionale. La realtà delle entità, da questo punto di vista, non è una caratteristica *intrinseca* delle entità, ma deriva in maggior misura dall'oggettività dei limiti di calcolo che affliggono il rilevamento della modularità. Così, questa è certamente una forma di antirealismo. È, comunque, un antirealismo dotato di *oggettività*, in quanto i limiti di calcolo cui è sottoposto sono oggettivi, insormontabili, e, anche se le loro conseguenze sono di natura pragmatica, le limitazioni che essi rappresentano sono insuperabili[58]. In un certo senso, si potrebbe anche ritenere l'antirealismo vincolato una forma di *realismo*, perché l'oggettività assoluta dei limiti di calcolo potrebbe deporre a favore della loro *realtà*, nel senso di esistenza indipendente. L'antirealismo vincolato potrebbe quindi essere considerato una forma di *realismo debole*, anche se un tipo di realismo diverso da quelli tipici, anche dal tipo platonico. In un certo senso, credo che questo potrebbe essere considerato un particolare tipo di *kantismo*, in cui le condizioni trascendentali assumono la forma di limiti di calcolo.

Detto questo, la domanda circa la *natura* di questi limiti di calcolo rimane aperta: la discussione, che sarebbe necessario svolgerebbe è molto spinosa, e toccherebbe una tesi molto attuale e problematica, che è comunemente conosciuta come la tesi del * pancomputazionalismo . *Questa è la tesi che sostiene la natura fondamentale del mondo è computazionale, o informazionale, e che tutti la cosiddetti realtà fisica consiste negli effetti di questo processo computazionale fondamentale. Dibattiti intorno a questo tema sono molto complessi e di lunga data, coinvolgendo la filosofia della fisica e anche la filosofia della matematica. Come detto, questo non è il luogo adatto per esaminare questi problemi in profondità. Per riassumere, posso dire che la mia posizione metafisica, in contrasto con la scelta della maggior dei parte sostenitori della spiegazione meccanicistica, non è quella del realismo scientifico. Questa è la ragione per cui ho abbozzato in qesto lavoro una forma più liberale di spiegazione rispetto al meccanicismo stretto, una forma di spiegazione che prendo come modello elettivo di per la spiegazione dei fenomeni complessi e di calcolo, cioè la spiegazione funzionale multilivello basata sulla relazione ricorsiva* specifica / implementazione*.

19.5.2 Metodi computazionali nella ricerca scientifica: un possibile punto di svolta storica?

Tutte le considerazioni esposte sopra potrebbero suggerire una tesi storica. Ma ho il sospetto che questa tesi, che vorrei proporre, potrebbe essere plausibilmente accusata di non essere ben supportata. Il motivo principale del mio sospetto è che non sono sicuro che essa possa essere vista come una tesi *storica*, che riguarda i fatti del passato: potrebbe benissimo essere una tesi su *futuri*, incipienti sviluppi storici della scienza. In ogni caso, è la mia impressione che ci siano fatti possibili interessanti connessi con metodi computazionali, modularità e antimodularità, molto vicini nel tempo al momento presente, probabilmente sul punto di accadere, se non si sono già verificati.

Quindi, la tesi storica che sto per suggerire qui è *ancora*, per il momento, probabilmente non molto corroborata dai fatti, ma è senza dubbio una tesi abbastanza forte. Sono consapevole che questa combinazione di alto impatto e basso supporto empirico è molto pericolosa per una tesi storica. Quello che vorrei fare è quindi, almeno per il momento, dare solo un accenno, o qualche semplice suggerimento, riguardante la possibilità che un importante cambiamento di paradigma

[58] A meno che, naturalmente, P = NP. Ma questo sembra molto improbabile al momento. Per una spiegazione, vedere la sezione 17.4.1.3.3.

nella scienza sia appena accaduto o possa essere in procinto di accadere. Questo cambiamento di paradigma si sarebbe verificato o starebbe per verificarsi, a causa della disponibilità e dell'uso, per diversi scopi, di potenti macchine di calcolo e algoritmi in diversi aspetti della pratica della ricerca scientifica. Gli usi dei computer in ambito scientifico le cui conseguenze che vorrei considerare qui sono due usi, appartenenti a due diverse fasi della ricerca scientifica.

Cominciamo con il più evidente: le *simulazioni al computer*. Qualcosa, io sostengo, è cambiato o è destinato a cambiare nel campo della scienza dal momento in cui complesse simulazioni al computer sono state accettate o verranno a essere accettate come corrette *spiegazioni scientifiche*. Il punto è: fin dalla sua nascita come fisica galileiana fino a tempi molto recenti, la scienza moderna ha, abbastanza plausibilmente, preso in considerazione sistemi che sono spiegabili in termini relativamente semplici, o suscettibili di essere descritti da modelli approssimativi sufficientemente fedeli (dati gli scopi dell'osservatore) al fenomeno empirico originale. Nel corso della tesi, ho cercato di dimostrare che alcuni sistemi, per ragioni di *complessità computazionale* (motivi che in un certo senso sono pragmatici, ma che da un altro punto di vista sono oggettivi) non possono essere descritti in maniera *modulare* . Questa *antimodularità* ha la conseguenza di rendere tali sistemi passibili solo di descrizioni di livello molto basso. Il problema è che di solito tali descrizioni non consentono un elevato grado di comprensibilità, a causa della enorme quantità di dettagli che portano. Tuttavia, in molti casi, la *simulazione al computer* può comunque modellizzare dinamicamente tali sistemi non modulari, rendendo fattibile la mera *predizione* della dinamica del sistema, almeno una previsione passo-passo per simulazione all'interno di un certo range temporale finito: tali sistemi sono in un certo modo dinamicamente prevedibili, ma il loro comportamento non può essere *spiegato* in maniera comprensibile. Si tratta di un problema reale? Fino ad ora, sembra *in effetti* essersi dato il caso che le spiegazioni scientifiche siano sempre risultate umanamente comprensibili. La maggior parte delle spiegazioni dei sistemi complessi naturali sono state fino ad ora, in qualche misura, di questo tipo. Il modello meccanicistico, di uso diffuso in biologia, ha sempre evidenziato la necessità di una spiegazione *multilivello*, che porta con sé un potenziale grado elevato di comprensibilità. Ma, modelli meccanicistici che sono colpiti da antimodularity non ammetterebbero spiegazioni multilivello, e questo li renderebbe molto difficili da capire, anche se essi constituirebbero in ogni caso una possibile base per la simulazione al computer. Dovrebbero modelli meccanicistici molto complessi di basso livello di questo tipo essere accettati come spiegazioni appropriate essi stessi, sapendo che non possiamo aspettarci alcun ulteriore miglioramento della loro comprensibilità?

Cosa costituisce la *spiegazione* del fenomeno, in questi casi? È il programma che gestisce la simulazione? Ma il programma, che di solito è modulare poiché così strutturato dal programmatore, e come tali soggetto potenzialmente a essere compreso, se il sistema simulato è davvero antimodulara, per definizione di antimodularità non può costituire una descrizione modulare *ad alto livello* del fenomeno: il programma che simula un sistema antimodulare potrebbe essere compreso *nei suoi termini* come programma modulare gerarchico, ma non come una *descrizione* del sistema simulato. Sarebbe certamente un programma comprensibile, ma ciò che capiremmo di esso sarebbe il fatto che simula il fenomeno complesso reiterando un enorme numero di volte alcune operazioni semplici, corrispondentementi precisamente alle attività delle parti semplici del sistema che si sta simulando. Quindi, non mi sembra che il *programma* potrebbe essere preso come spiegazione: essendo isomorfo alla descrizione dinamica di basso livello del sistema, esso non costituirebbe una *spiegazione computazionale multilivello*.

Allora, che cosa dobbiamo considerare come la spiegazione di un fenomeno antimodulare simulato al computer? Penso che sia *l'intera simulazione dinamica* che debba essere presa come spiegazione, e, dato che il fenomeno è antimodulare, la simulazione può essere solo guardato e osservata, ma non *compresa*. Almeno non può essere compresa in modo funzionale o meccanicis-

tico. Potrebbe tuttavia essere significativamente spiegata in maniera *topologica* (vedere la sezione 19.4.5): prendendo in considerazione alcune caratteristiche generali della rete che costituisce il modello del sistema complesso, qualche conclusione si potrebbe probabilmente trarre riguardo certe proprietà della dinamica che si svolge, per simulazione, su tale modello.

Quindi, torniamo alla domanda: dovrebbero le simulazioni meccanicistiche di basso livello molto complesse di questo tipo essere accettate come spiegazioni adeguate esse stesse, senza aspettarsi alcun ulteriore miglioramento nella loro comprensibilità? Se la risposta è sì, allora la scienza ha subito un grande cambiamento storico: la scienza potrebbe allora focalizzarsi su sistemi che, essendo troppo complessi e interconnessi per essere oggetto di descrizioni modulari, avrebbero dovuto essere lasciati fuori della ricerca scientifica prima dell'avvento di simulazione al computer. Questo è già successo in parte, almeno da tre decenni: basti pensare a tutta la letteratura sulla simulazione di sistemi complessi e caotici, che ha prosperato sin dagli anni '80. Si deve notare che, essendo la maggior parte di questi sistemi simulati, almeno in certe regioni del loro spazio delle fase, antimodulari, le spiegazioni tipiche impiegate in testi di quest'area della ricerca (come, per esempio, le opere di Stuart Kauffman), sono spiegazioni statistiche o *topologiche* (nel senso impiegato da Philippe Huneman, vedere la sezione 19.4.5), una forma di spiegazione che, come abbiamo visto, è ancora consentita in sistemi antimodulari.

Un altro possibile modo in cui i computer possono rivoluzionare la ricerca scientifica, un modo di cui una forma specifica è la rilevazione di modularità, è nell'aiutare a *trovare* un modello teorico. In questo caso un algoritmo viene a sostituire in parte il ricercatore, non nella raccolta di dati grezzi, ma nel compito di escogitare un modello teorico che raccordi dati *già* disponibili. Tuttavia, questa non è la questione completa, perché a volte accade che la raccolta di dati grezzi possa essere eseguita automaticamente. Questo è particolarmente vero nei casi in cui l'oggetto di studio è di per sé un oggetto *digitale*: studio dei testi, letteratura, contenuti Internet, struttura di internet, e così via. In ogni caso, dopo la fase di raccolta dei dati, arriva la necessità di un modello teorico in grado di inglobare tutti i dati raccolti: di solito, il modello è elaborato dal ricercatore umano. Ma per quanto riguarda i casi in cui lo sperimentatore non è in grado di elaborare un tale modello? Che dire dei casi in cui la quantità di dati è così grande che non è da aspettarsi che un essere umano sarà in grado di discernere un pattern in essa, al fine di elaborare un modello teorico? Anche qui (tranne nei casi in cui i dati, anche se complessi, possono essere * aggregati * in un modo che consente una semplice descrizione, come in meccanica statistica), si tratta di discernere qualche struttura nei dati, una struttura modulare di qualsiasi tipo, perché i dati siano suscettibili di modellizzazione e di spiegazione umanamente comprensibili. Eseguendo quel che si chiama "data mining", i computer sono stati in grado, in alcuni casi, di essere validamente di supplemento in questo compito.

Vorrei citare qui un caso particolarmente sorprendente. Utilizzando un metodo di rilevazione della struttura di comunità[59]) nelle reti, Wilkinson & Huberman (2004) sono stati in grado di analizzare algoritmicamente la letteratura scientifica esistente sul cancro al colon, e di trovare automaticamente i moduli di geni della rete regolativa genica umana coinvolti nel cancro del colon, senza nemmeno dover fornire preventivamente al programma i dati grezzi che descrivono la rete genetica: tutti i dati necessari sono stati automaticamente "estratti" dalla *letteratura* preesistente. Questo caso è particolare perché, qui, i dati stessi sono già memorizzati in un supporto informatico utilizzabile dalla macchina, e non sono strutturati ad-hoc, anche se, in ultima analisi, i dati (la letteratura accademica) in realtà provengono dal lavoro di ricercatori umani . Ma un altro risultato seriamente sorprendente è che il sistema di Wilkinson e Huberman è stato in grado di trovare parti della rete genetica coinvolte nel cancro del colon che avevano

[59] Cioè, il rilevamento di un tipo di modularità. Vedere la sezione 3.2.1.

eluso l'attenzione (lo span limitato di attenzione) umana dei ricercatori: la macchina ha trovato un nuovo modello teorico di un fenomeno, un modello probabilmente il cui reperimento sarebbe fuori dalla portata delle capacità umane! Ora, in questo caso, il risultato ottenuto è ancora probabilmente esprimibile in forma umanament comprensibile, proprio perché è una descrizione *modulare* del sistema genetico sotto osservazione. Ma cosa succederebbe se un programma, analizzando la letteratura clinica, trovasse un modello modulare che unisce in moduli dati di natura eterogenea, combinandoli in una maniera che è improbabile che alcun essere umano possa venire a concepire spontaneamente? Ad esempio, producendo un modello modulare in cui i moduli sono composti sia dei geni che delle proteine del proteoma, ma che sono correlati fra loro in modo non semplice? Potrebbe questo modello essere ancora comprensibile da parte dei ricercatori umani? Oppure, cosa accadrebbe se il modello, seppur modulare, fosse composto di centinaia di moduli di medio livello senza alcuna descrizione modulare sovraordinata in grado di raggruppare insieme alcuni? Si consideri che, a causa della sua complessità computazionale, l'algoritmo di Wilkinson e Huberman non è in grado di elaborare reti con più di qualche migliaio di geni. A causa della complessità computazionale eccessiva dell'algoritmo di rilevamento della modularità di alto livello, dunque, dovremmo ricorrere ad un modello del fenomeno che non è modulare ad un livello superiore. Tale modello, se valido, potrebbe plausibimente essere utilizzato per eseguire alcune ulteriori *simulazioni* del fenomeno osservato. Ma il fenomeno non sarà facile da spiegare mediante il modello, dal momento che la decomposizione funzionale del modello è stata impossibile, e, così com'è, esso sarà troppo complesso per essere compreso.

Ma, vorrei speculare ulteriormente: potrebbe anche essere possibile che il *fenomeno* stesso, scoperto dal data mining algoritmico, finisca per risultare essere né un fenomeno *noto*, né un fenomeno facilmente *comprensibile*? Che dire di un fenomeno complesso che nessun essere umano avrebbe plausibilmente preso in considerazione, e che è difficile per noi anche solo descrivere plausibilmente, o *nominarlo*? Anche se oltre l'intuizione umana spontanea o perfino oltre la comprensione umana, accadrà, molto probabilmente che programmi informatici vengano a scoprire questo tipo di fenomeni. Ma, che cosa sarà della scienza, allora? Penso che questo problema potrebbe ionsinuarsi potenzialmente più a fondo: il compito di rilevamento della modularità ha bisogno di una *relazione tra le parti* del sistema, al fine di valutare la sua modularità, e questo rapporto e le parti stesse sono date insieme in quello che ho chiamato la *descrizione preferita*: come ho sostenuto in precedenza[60], è proprio questa descrizione preferita, insieme ai limiti computazionali che affliggono la rilevazione di modularità, a determinare l'"ontologia" del sistema sotto osservazione. Ora, che cosa si può pensare della possibilità di *cambiare* la descrizione preferita da quella tipica, "naturale", ad una artificiosa? Data la possibilità della descrizione algoritmica di relazioni complesse tra le parti, allora anche generi complessi, *generi non naturali* potrebbero essere rilevati come moduli e essere fatti oggetto di scienza, lasciando gli algoritmi associare gruppi eterogenei di proprietà del mondo reale in parti e relazioni di una descrizione di alto livello. Un, "mondo bizzarro", completamente diverso da quello comunemente descritto dalla scienza potrebbe venire fuori da quella descrizione. Tale descrizione non standard potrebbe anche essere gerarchicamente decomposta in modo da essere comprensibile, almeno in linea di principio. Oppure, potrebbe dar luogo a una struttura gerarchica così complessa da poter essere utile per la spiegazione solo in linea di principio: a causa di questa "rimappatura", una nuova serie di "generi innaturali" potrebbe emergere, e con loro, nuove discipline. Naturalmente questa modularizzazione deve essere tale da rilevare moduli sufficientemente *robusti*, altrimenti non costituirebbe una descrizione modulare *valida* del mondo. Si potrebbe allora obiettare che la descrizione naturale causale, comune del mondo popolato da oggetti causalmente coesi è l'unica possibile descrizione robusta. La questione è aperta, e ho il sospetto che qualche modularizzazione valida non ovvia e sorprendente

[60] Sezione 19.5.1.

sia è possibile, e forse è già stata proposta, vale a dire la descrizione della fisica quantistica[61].

Tutto sommato, un ricorso diffuso a questo tipo di rimappatura artificiale, o a più familiari simulazioni al computer come quelle sopra descritte, potrebbe certamente apportare in alcune aree scientifiche una tale serie di innovazioni nel metodo e nei criteri, da costituire un cambiamento di paradigma nella scienza, con la possibilità di vedere il sorgere di nuove discipline scientifiche. Il rovescio della medaglia è che dovremmo abbandonare la prospettiva della scienza come un percorso progressivo verso una migliore comprensione del mondo: la tendenza sarebbe verso una forma inedita di "spiegazione scientifica automatizzata", forse incomprensibile.

Quanto sopra è una possibile formulazione della tesi storica che volevo discutere. Forse richiede eccessivo sforzo di immaginazione. Inoltre, è abbastanza chiaro non ho in questo lavoro prodotto un sostegno sufficientemente forte per poter affermare questa tesi. Il motivo è che ho mirato più in basso, in questro lavoro: ho solo cercato di sgombrare il terreno proponendo una serie di definizioni e proponendo prima di tutto una proprietà, l'*antimodularità*, che, se e quando comparisse in fenomeni reali, potrebbe causare problemi per alcuni modelli di spiegazione scientifica e la necessità del ricorso alla simulazione al computer. Il condizionale appena formulato manca di supporto per la premessa: è probabile che l'antimodularità appaia soprattutto in certi fenomeni complessi, ma non ho dimostrato qui che un tale tipo di fenomeni sia così centrale e diffuso nella letteratura scientifica oggi. Bisognerà vedere se è questo il caso.

Ma, come secondo punto, penso che debba essere messo in evidenza che il crescente bisogno di ricorrere, quando la modularità può essere effettivamente trovata, a mezzi *algoritmici* per la sua rilevazione, e quindi il bisogno di ricorrere a una sorta di spiegazione favorita dai progressi nella potenza di calcolo, potrebbe *questo fatto stesso* favore l'interesse verso fenomeni particolarmente complessi, oppure, in alcuni casi, apportare persino la possibilità di "vedere" l'esistenza di fenomeni che potrebbero essere completamente sfuggiti all'attenzione della ricerca scientifica non computazionale: il rilevamento automatico della struttura modulare, ove fattibile, potrebbe produrre descrizioni modulari a prima vista incomprensibili , se la macchina è in grado di raggruppare le parti in moduli sulla base della considerazione di relazioni artificiose, innaturali ma significative, tra le parti, relazioni che erano state precedentemente invisibili alla comprensione umana. E, da quel momento in poi, una tendenza verso una scienza più computazionale, forse meno umanamente comprensibile, è una tendenza alimentata da un feedback positivo: una tale scienza, se queste spiegazioni sono utilizzate anche per orientare ulteriori ricerche, potrebbe avanzare in modi che a noi apparirebbero oscuri.[62]

[61] Probabilmente la descrizione fisica quantistica non è immediatamente una *modularizzazione* discreta come quelle che ho trattato, più compatibili con una visione meccanicistica di sistemi. Tuttavia, la fisica quantistica costituisce una valida descrizione del mondo alternativa rispetto a quella del senso comune, o a quella della fisica classica.

[62] Al fine di mantenere la discussione all'interno dei temi della filosofia della scienza e della storia della scienza, non oso qui chiamare questa scienza futura, per lo più computazionale, potenzialmente incomprensibile, una scienza "postumana", anche se quello che ho scritto potrebbe probabilmente evocare un qualche uso legittimo del termine.

References

Albert, R., & Barabási, A.-L. (2002). Statistical mechanics of complex networks. *Reviews of Modern Physics*, *74*(1), 47–97.

Albert, R., Jeong, H., & Barabási, A.-L. (2000). Error and attack tolerance of complex networks. *Nature*, *406*(6794), 378–382.

Alon, U. (2006). *An Introduction to Systems Biology: Design Principles of Biological Circuits*. CRC Press.

Amaral, L. a. N., Scala, A., Barthélémy, M., & Stanley, H. E. (2000). Classes of small-world networks. *Proceedings of the National Academy of Sciences*, *97*(21), 11149–11152.

Ammons, G., Bodík, R., & Larus, J. R. (2002). Mining Specifications. In *Proceedings of the 29th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '02 (p. 4–16). New York, NY, USA: ACM.

Amundson, R., & Lauder, G. V. (1994). Function without purpose. *Biology and Philosophy*, *9*(4), 443–469.

Appel, K., & Haken, W. (1976). Every planar map is four colorable. *Bulletin of the American Mathematical Society*, *82*(5), 711–712.

Arcaya, I., & Romero, N. (2007). On a Hedlund's theorem and place-dependent cellular automata. *Divulgaciones Matemáticas*, *15*(2), 81–92.

Arenas, A., Díaz-Guilera, A., & Pérez-Vicente, C. J. (2006). Synchronization reveals topological scales in complex networks. *Physical review letters*, *96*(11), 114102.

Arenas, A., Fernández, A., Fortunato, S., & Gómez, S. (2008). Motif-based communities in complex networks. *Journal of Physics A: Mathematical and Theoretical*, *41*(22), 224001.

Arora, S., & Barak, B. (2009). *Computational complexity: a modern approach*. Cambridge University Press.

Baltagi, B. (2011). *Econometrics*. Springer.

Barabási, A.-L., & Albert, R. (1999). Emergence of Scaling in Random Networks. *Science*, *286*(5439), 509–512.

Barabási, A.-L., & Oltvai, Z. N. (2004). Network biology: understanding the cell's functional organization. *Nature Reviews Genetics*, *5*(2), 101–113.

Batada, N. N., Reguly, T., Breitkreutz, A., Boucher, L., Breitkreutz, B.-J., Hurst, L. D., & Tyers, M. (2006). Stratus Not Altocumulus: A New View of the Yeast Protein Interaction Network. *PLoS Biology*, *4*(10).

Bechtel, W., & Abrahamsen, A. (2005). Explanation: a mechanist alternative. *Studies in History and Philosophy of Science Part C: Studies in History and Philosophy of Biological and Biomedical Sciences*, Mechanisms in biology, *36*(2), 421–441.

Bechtel, W., & Abrahamsen, A. (2010). Dynamic mechanistic explanation: computational modeling of circadian rhythms as an exemplar for cognitive science. *Studies in History and Philosophy of Science Part A*, *41*(3), 321–333.

Bechtel, W., & Richardson, R. C. (1993). *Discovering Complexity: Decomposition and Localization as Strategies in Scientific Research*. Princeton University Press.

Beck, J., & Eichmann, D. (1993). Program and interface slicing for reverse engineering. In *Proceedings of the 15th international conference on Software Engineering* (p. 509–518). IEEE Computer Society Press.

Bedau, M. A. (1997). Weak Emergence. *Noûs*, *31*, 375–399.

Bird, R. J., & Wadler, P. (1988). *An Introduction to Functional Programming.* Prentice Hall.

Boccaletti, S., Ivanchenko, M., Latora, V., Pluchino, A., & Rapisarda, A. (2007). Detecting complex network modularity by dynamical clustering. *Physical Review E*, *75*(4), 045102.

Bovet, D. P., Crescenzi, P., & Bovet, D. (1994). *Introduction to the Theory of Complexity.* Prentice Hall London.

Böhm, C., & Jacopini, G. (1966). Flow diagrams, turing machines and languages with only two formation rules. *Communications of the ACM*, *9*(5), 366–371.

Börner, K., Sanyal, S., & Vespignani, A. (2007). Network science. *Annual Review of Information Science and Technology*, *41*(1), 537–607.

Brandes, U., Delling, D., Gaertler, M., Gorke, R., Hoefer, M., Nikoloski, Z., & Wagner, D. (2008). On Modularity Clustering. *IEEE Transactions on Knowledge and Data Engineering*, *20*(2), 172–188.

Bunge, M. (1959). *Causality: The Place of the Causal Principle in Modern Science.* Cambridge University Press.

Bunge, M. (1963). A General Black Box Theory. *Philosophy of Science*, *30*(4), 346–358.

Bunge, M. (1964). Phenomenological theories. In *The Critical Approach to Science and Philosophy: In Honor of Karl R. Popper : (New York, N.Y.): Free Press of Glencoe.* Collier-Macmillan.

Bunge, M. (1997). Mechanism and Explanation. *Philosophy of the Social Sciences*, *27*(4), 410–465.

Caldarelli, G., & Catanzaro, M. (2012). *Networks: A Very Short Introduction.* Oxford University Press.

Cale, W. G., O'Neill, R. V., & Gardner, R. H. (1983). Aggregation error in nonlinear ecological models. *Journal of Theoretical Biology*, *100*(3), 539–550.

Chalmers, D. J. (1996). Does a rock implement every finite-state automaton? *Synthese*, *108*(3), 309–333.

Chikofsky, E. J., & Cross, J. H. (1990). Reverse engineering and design recovery: A taxonomy. *Software, IEEE*, *7*(1), 13–17.

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on Information Theory*, *2*(3), 113–124.

Church, A. (1936a). An Unsolvable Problem of Elementary Number Theory. *American Journal of Mathematics*, *58*(2), 345–363.

Church, A. (1936b). A Note on the Entscheidungsproblem. *The Journal of Symbolic Logic*, *1*(1), 40–41.

Clauset, A., Moore, C., & Newman, M. E. (2008). Hierarchical structure and the prediction of missing links in networks. *Nature*, *453*(7191), 98–101.

Clauset, A., Moore, C., & Newman, M. E. J. (2007). Structural Inference of Hierarchies in Networks. In E. Airoldi, D. M. Blei, S. E. Fienberg, A. Goldenberg, E. P. Xing, & A. X. Zheng (Eds.), *Statistical Network Analysis: Models, Issues, and New Directions*, Lecture Notes in Computer Science (pp. 1–13). Springer Berlin Heidelberg.

Clauset, A., Newman, M. E. J., & Moore, C. (2004). Finding community structure in very large networks. *Physical Review E*, *70*(6), 066111.

Colburn, T. (2004). Methodology of computer science. In L. Floridi (Ed.), *The Blackwell guide to the philosophy of computing and information* (p. 318–326).

Conant, G. C., & Wagner, A. (2003). Convergent evolution of gene circuits. *Nature Genetics*, *34*(3), 264–266.

Cook, M. (2004). Universality in elementary cellular automata. *Complex Systems*, *15*(1), 1–40.

Cook, S. A. (1971). The Complexity of Theorem-proving Procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71 (p. 151–158). New York, NY, USA: ACM.

Copeland, B. J. (2002). Hypercomputation. *Minds and Machines*, *12*(4), 461–502.

Copeland, B. J. (2004). *The Essential Turing*. Oxford University Press.

Copeland, B. J. (2008). The Church-Turing Thesis. In E. N. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy* (Fall 2008.).

Couch, M. B. (2011). Causal Role Theories of Functional Explanation. *Internet Encyclopedia of Philosophy*, (http://www.iep.utm.edu/, 11-19-2014).

Cummins, R. (1974). Dispositions, States and Causes. *Analysis*, *34*(6), 194–204.

Cummins, R. C. (1975). Functional Analysis. *Journal of Philosophy*, *72*(November), 741–64.

Cummins, R. C. (2000). "How Does It Work" Versus "What Are the Laws?": Two Conceptions of Psychological Explanation. In F. Keil & R. A. Wilson (Eds.), *Explanation and Cognition, 117-145*. MIT Press.

Danon, L., Díaz-Guilera, A., Duch, J., & Arenas, A. (2005). Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, *2005*(09), P09008.

Davis, M. (2004). The Myth of Hypercomputation. In C. Teuscher (Ed.), *Alan Turing: Life and Legacy of a Great Thinker* (pp. 195–211). Springer Berlin Heidelberg.

Daylight, E. G. (2012). Turing's Influence on Programming. In *Turing-100*, EPiC Series (p. 42–52). EasyChair.

Daylight, E. G. (2013). Towards a historical notion of "Turing - the Father of Computer Science". *History and Philosophy of Logic*.

Deutsch, D. (1985). Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, *400*(1818), 97–117.

Dijkstra, E. W. (1968). Letters to the Editor: Go to Statement Considered Harmful. *Commun. ACM*, *11*(3), 147–148.

Dijkstra, E. W. (1982). On the role of scientific thought. In *Selected Writings on Computing: A Personal Perspective* (p. 60–66). Springer.

Díaz-Guilera, A. (2008). Dynamics towards synchronization in hierarchical networks. *Journal of Physics A: Mathematical and Theoretical*, *41*(22), 224007.

Erdős, P., & Rényi, A. (1960). On the Evolution of Random Graphs. In *Publication of the Mathematical Institute of the Hungarian Academy of Sciences* (p. 17–61).

Faust, K. (1988). Comparison of methods for positional analysis: Structural and general equivalences. *Social Networks*, *10*(4), 313–341.

Fisher, F. M. (1963a). Decomposability, Near Decomposability, and Balanced Price Change under Constant Returns to Scale. *Econometrica*, *31*(1/2), 67–89.

Fisher, F. M. (1963b). Properties of the Von Neumann Ray in Decomposable and Nearly Decomposable Technologies. In *A. Ando, FM Fisher, HA Simon: Essays in the Structure of Social Science Models, Cambridge, Mass.*

Fodor, J. A. (1981). The Mind-Body Problem. *Scientific American*, *244*, 114–25.

Fortnow, L., & Homer, S. (2003). A short history of computational complexity. *Bulletin of the EATCS*, *80*, 95–133.

Fortunato, S., & Barthélemy, M. (2007). Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, *104*(1), 36–41.

Fortunato, S., Latora, V., & Marchiori, M. (2004). Method to find community structures based on information centrality. *Physical Review E*, *70*(5), 056104.

Galton, A. (1993). On the Notions of Specification and Implementation. *Royal Institute of Philosophy Supplements*, *34*, 111–136.

Gardner, M. (1970). The fantastic combinations of John Conway's new solitaire game. *Scientific American*, *223*, 120–123.

Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

Girvan, M., & Newman, M. E. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, *99*(12), 7821–7826.

Goodman, C. S., & Coughlin, B. C. (2000). The evolution of evo-devo biology. *Proceedings of the National Academy of Sciences*, *97*(9), 4424–4425.

Gould, S. J., & Lewontin, R. C. (1979). The Spandrels of San Marco and the Panglossian Paradigm: A Critique of the Adaptationist Programme. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, *205*(1161), 581–598.

Gödel, K. (1930). Die Vollständigkeit der Axiome des logischen Funktionenkalküls. *Monatshefte für Mathematik und Physik*, *37*(1), 349–360.

Grochow, J. A., & Kellis, M. (2007). Network motif discovery using subgraph enumeration and symmetry-breaking. In *Research in Computational Molecular Biology* (p. 92–106). Springer.

Guimerà, R., & Amaral, L. A. (2005a). Cartography of complex networks: modules and universal roles. *Journal of Statistical Mechanics: Theory and Experiment*, *2005*(02), P02001.

Guimerà, R., & Amaral, L. A. N. (2005b). Functional cartography of complex metabolic networks. *Nature*, *433*(7028), 895–900.

Guimerà, R., Sales-Pardo, M., & Amaral, L. A. N. (2004). Modularity from fluctuations in random graphs and complex networks. *Physical Review E*, *70*(2), 025101.

Gulbahce, N., & Lehmann, S. (2008). The art of community detection. *BioEssays*, *30*(10), 934–938.

Haigh, T. (2013). Actually, Turing Didn't Invent the Computer (draft, Historical Reflections column for Communications of the ACM). In *SIGCIS 2013 Workshop*. Presented at the Old Ideas: Recomputing the History of Information Technology, Portland, Maine.

Han, J.-D. J., Bertin, N., Hao, T., Goldberg, D. S., Berriz, G. F., Zhang, L. V., Dupuy, D., et al. (2004). Evidence for dynamically organized modularity in the yeast protein–protein interaction network. *Nature*, *430*(6995), 88–93.

Hanson, J. E., & Crutchfield, J. P. (1997). Computational mechanics of cellular automata: An example. *Lattice Dynamics*, *103*(1–4), 169–189.

Harré, R. (1959). Metaphor, Model and Mechanism. *Proceedings of the Aristotelian Society*, New Series, *60*, 101–122.

Hartmanis, J., & Stearns, R. E. (1965). On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 285–306.

Hartwell, L. H., Hopfield, J. J., Leibler, S., & Murray, A. W. (1999). From molecular to modular cell biology. *Nature*, *402*, C47–C52.

Haugeland, J. (1989). *Artificial Intelligence: The Very Idea*. MIT Press.

Hedlund, G. A. (1969). Endomorphisms and automorphisms of the shift dynamical system. *Mathematical systems theory*, *3*(4), 320–375.

Hempel, C. G., & Oppenheim, P. (1948). Studies in the Logic of Explanation. *Philosophy of Science*, *15*(2), 135–175.

Hilbert, D., & Ackermann, W. (1928). *Grundzüge der Theoretischen Logik*. Berlin: Springer Verlag.

Hillis, W. D. (2015, March 3). The connection machine / W. Daniel Hillis.

Hopfield, J. J. (1994). Physics, Computation, and Why Biology Looks so Different. *Journal of Theoretical Biology*, *171*(1), 53–60.

Huneman, P. (2010). Topological explanations and robustness in biological sciences. *Synthese*, *177*(2), 213–245.

Huneman, P. (2015). Diversifying the picture of explanations in biological sciences: ways of combining topology with mechanisms. *Synthese*, 1–32.

Itzhack, R., Mogilevski, Y., & Louzoun, Y. (2007). An optimal algorithm for counting network motifs. *Physica A: Statistical Mechanics and its Applications*, *381*, 482–490.

Itzkovitz, S., Levitt, R., Kashtan, N., Milo, R., Itzkovitz, M., & Alon, U. (2005). Coarse-graining and self-dissimilarity of complex networks. *Physical Review E*, *71*(1), 016127.

Iwasa, Y., Andreasen, V., & Levin, S. (1987). Aggregation in model ecosystems. I. Perfect aggregation. *Ecological Modelling*, *37*(3–4), 287–302.

Iwasa, Y., Levin, S. A., & Andreasen, V. (1989). Aggregation in Model Ecosystems II. Approximate Aggregation. *Mathematical Medicine and Biology*, *6*(1), 1–23.

Jeong, H., Tombor, B., Albert, R., Oltvai, Z. N., & Barabási, A.-L. (2000). The large-scale organization of metabolic networks. *Nature*, *407*(6804), 651–654.

Karrer, B., Levina, E., & Newman, M. E. J. (2008). Robustness of community structure in networks. *Physical Review E*, *77*(4), 046119.

Kashtan, N., & Alon, U. (2005). Spontaneous evolution of modularity and network motifs. *Proceedings of the National Academy of Sciences of the United States of America*, *102*(39), 13773–13778.

Kashtan, N., Itzkovitz, S., Milo, R., & Alon, U. (2004a). Topological generalizations of network motifs. *Physical Review E*, *70*(3), 031909.

Kashtan, N., Itzkovitz, S., Milo, R., & Alon, U. (2004b). Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, *20*(11), 1746–1758.

Kauffman, S. A. (1993). *The Origins of Order: Self Organization and Selection in Evolution.* Oxford University Press.

Kim, J. (1989a). Mechanism, Purpose, and Explanatory Exclusion. *Philosophical Perspectives*, *3*, 77–108.

Kim, J. (1989b). The Myth of Nonreductive Materialism. *Proceedings and Addresses of the American Philosophical Association*, *63*(3), 31.

Kim, J. (1989c). Mechanism, Purpose, and Explanatory Exclusion. *Philosophical Perspectives*, *3*, 77–108.

Kimura, M. (1968). *The Neutral Theory of Molecular Evolution.* Cambridge University Press.

Kirkpatrick, S. (1984). Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, *34*(5-6), 975–986.

Kreinovich, V., & Shpak, M. (2006). Aggregability is NP-hard. *ACM SIGACT News*, *37*(3), 97–104.

Kreinovich, V., & Shpak, M. (2008). Decomposable Aggregability in Population Genetics and Evolutionary Computations: Algorithms and Computational Complexity. In A. Kelemen, A. Abraham, & Y. Liang (Eds.), *Computational Intelligence in Medical Informatics*, Studies in Computational Intelligence (pp. 69–92). Springer Berlin Heidelberg.

Kripke, S. A. (1982). *Wittgenstein on Rules and Private Language: An Elementary Exposition.* Harvard University Press.

Kuramoto, Y. (2003). *Chemical Oscillations, Waves, and Turbulence.* Courier Corporation.

Lancichinetti, A., Fortunato, S., & Kertész, J. (2009). Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, *11*(3), 033015.

Laplante, P. A. (2007). *What Every Engineer Should Know about Software Engineering.* CRC Press.

Lazebnik, Y. (2002). Can a biologist fix a radio? — or, what I learned while studying apoptosis. *Cancer cell*, *2*(3), 179–182.

Lee, T. I., Rinaldi, N. J., Robert, F., Odom, D. T., Bar-Joseph, Z., Gerber, G. K., Hannett, N. M., et al. (2002). Transcriptional Regulatory Networks in Saccharomyces cerevisiae. *Science*, *298*(5594), 799–804.

Leicht, E. A., & Newman, M. E. J. (2008). Community Structure in Directed Networks. *Physical Review Letters*, *100*(11), 118703.

Levin, L. (1973). Universal Sequential Search Problems. *Problemy Peredachi Informatsii*, *9*(3), 115–116.

Levy, A., & Bechtel, W. (2013). Abstraction and the Organization of Mechanisms. *Philosophy of Science*, *80*(2), 241–261.

Lewis, H. R., & Papadimitriou, C. H. (1998). *Elements of the Theory of Computation: Harry R. Lewis, Christos H. Papadimitriou*. Prentice-Hall International.

Liu, T.-C. (1955). A Simple Forecasting Model for the U.S. Economy. *Staff Papers - International Monetary Fund*, *4*(3), 434.

Lorrain, F., & White, H. C. (1971). Structural equivalence of individuals in social networks. *The Journal of Mathematical Sociology*, *1*(1), 49–80.

Machamer, P. K., Darden, L., & Craver, C. F. (2000). Thinking About Mechanisms. *Philosophy of Science*, *67*(1), 1–25.

Mangan, S., & Alon, U. (2003). Structure and function of the feed-forward loop network motif. *Proceedings of the National Academy of Sciences*, *100*(21), 11980–11985.

Mangan, S., Zaslaver, A., & Alon, U. (2003). The Coherent Feedforward Loop Serves as a Sign-sensitive Delay Element in Transcription Networks. *Journal of Molecular Biology*, *334*(2), 197–204.

Martínez, G. J., Adamatzky, A., & McIntosh, H. V. (2006). Phenomenology of glider collisions in cellular automaton Rule 54 and associated logical gates. *Chaos, Solitons & Fractals*, *28*(1), 100–111.

Martínez, G. J., Adamatzky, A., & McIntosh, H. V. (2014). Complete Characterization of Structure of Rule 54. *Complex Systems*, *23*(3), 259–293.

Maslov, S., & Sneppen, K. (2002). Specificity and Stability in Topology of Protein Networks. *Science*, *296*(5569), 910–913.

Massen, C. P., & Doye, J. P. K. (2005). Identifying communities within energy landscapes. *Physical Review E*, *71*(4), 046101.

Mazurie, A., Bottani, S., & Vergassola, M. (2005). An evolutionary and functional assessment of regulatory network motifs. *Genome Biology*, *6*(4), 1–12.

McAdams, H. H., & Shapiro, L. (1995). Circuit Simulation of Genetic Networks. *Science*, New Series, *269*(5224), 650–656.

Millikan, R. G. (1984). *Language, Thought, and Other Biological Categories: New Foundations for Realism*. MIT Press.

Millikan, R. G. (1989). In Defense of Proper Functions. *Philosophy of Science*, *56*(2), 288–302.

Milo, R., Itzkovitz, S., Kashtan, N., Levitt, R., Shen-Orr, S., Ayzenshtat, I., Sheffer, M., et al. (2004). Superfamilies of Evolved and Designed Networks. *Science*, *303*(5663), 1538–1542.

Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., & Alon, U. (2002). Network motifs: simple building blocks of complex networks. *Science*, *298*(5594), 824–827.

Minelli, A. (2000). Holomeric vs. meromeric segmentation: a tale of centipedes, leeches, and rhombomeres. *Evolution & Development*, *2*(1), 35–48.

Minelli, A. (2009). Evolutionary Developmental Biology does not Offer a Significant Challenge to the Neo-Darwinian Paradigm. In F. J. Ayalaessor & R. Arp (Eds.), *Contemporary Debates in Philosophy of Biology* (pp. 213–226). Wiley-Blackwell.

Nagel, E. (1961). *The Structure of Science: Problems in the Logic of Scientific Explanation*. Routledge.

Neander, K. (1991). The teleological notion of "function". *Australasian Journal of Philosophy*, *69*(4), 454–468.

Newman, M. E. (2003). The structure and function of complex networks. *SIAM review*, *45*(2), 167–256.

Newman, M. E. (2004a). Analysis of weighted networks. *Physical Review E*, *70*(5), 056131.

Newman, M. E. (2004b). Fast algorithm for detecting community structure in networks. *Physical Review E*, *69*(6), 066133.

Newman, M. E. (2006). Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, *103*(23), 8577–8582.

Newman, M. E., & Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical review E*, *69*(2), 026113.

Palla, G., Derényi, I., Farkas, I., & Vicsek, T. (2005). Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, *435*(7043), 814–818.

Pan, R. K., & Sinha, S. (2009). Modularity produces small-world networks with dynamical time-scale separation. *EPL (Europhysics Letters)*, *85*(6), 68006.

Parnas, D. L. (1972). On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, *15*(12), 1053–1058.

Partridge, D., & Galton, A. (1995). The specification of "specification". *Minds and Machines*, *5*(2), 243–255.

Piccinini, G. (2008). Some neural networks compute, others don't. *Neural Networks*, Advances in Neural Networks Research: IJCNN '07 2007 International Joint Conference on Neural Networks IJCNN '07, *21*(2–3), 311–321.

Piccinini, G. (2011). The Physical Church–Turing Thesis: Modest or Bold? *The British Journal for the Philosophy of Science*, axr016.

Polger. (2004). *Natural Minds*. MIT Press.

Priestley, M. (2011). *A Science of Operations: Machines, Logic and the Invention of Programming*. Springer Science & Business Media.

Putnam, H. (1967a). Psychological predicates. In W. H. Capitan & D. D. Merrill (Eds.), *Art, Mind, and Religion* (p. 37–48). University of Pittsburgh Press.

Putnam, H. (1967b). The nature of mental states. In *Mind, Language and Reality: Philosophical Papers, Volume 2* (pp. 429–440). Cambridge University Press, 1975.

Putnam, H. (1988). *Representation and Reality*. MIT Press.

Quine, W. V. (1951). Two Dogmas of Empiricism. *The Philosophical Review*, *60*(1), 20.

Ravasz, E., & Barabási, A.-L. (2003). Hierarchical organization in complex networks. *Physical Review E*, *67*(2), 026112.

Ravasz, E., Somera, A. L., Mongru, D. A., Oltvai, Z. N., & Barabási, A.-L. (2002). Hierarchical Organization of Modularity in Metabolic Networks. *Science*, *297*(5586), 1551–1555.

Reichardt, J., & White, D. R. (2007). Role models for complex networks. *The European Physical Journal B*, *60*(2), 217–224.

Rekoff, M. G. (1985). On reverse engineering. *Systems, Man and Cybernetics, IEEE Transactions on*, (2), 244–252.

Rendell, P. (2002). Turing Universality of the Game of Life. In A. Adamatzky (Ed.), *Collision-Based Computing* (pp. 513–539). Springer London.

Sales-Pardo, M., Guimerà, R., Moreira, A. A., & Amaral, L. A. N. (2007). Extracting the hierarchical organization of complex systems. *Proceedings of the National Academy of Sciences*, *104*(39), 15224–15229.

Salthe, S. N. (1985). *Evolving Hierarchical Systems: Their Structure and Representation*. Columbia University Press.

Schlosser, G. (2002). Modularity and the units of evolution. *Theory in Biosciences*, *121*(1), 1–80.

Schlosser, G., & Wagner, G. P. (2004). *Modularity in Development and Evolution*. University of Chicago Press.

Schüle, M. (2014). Natural Computation: the Cellular Automata Case. In *AISB 50 - The AISB 2014 Convention at Goldsmiths, University of London*. Presented at the 7th AISB Symposium on Computing and Philosophy: Is computation observer-relative?, London.

Searle, J. R. (1990). Is the brain a digital computer? In *Proceedings and addresses of the american philosophical association* (p. 21–37). JSTOR.

Shen-Orr, S., Milo, R., Mangan, S., & Alon, U. (2002). Network motifs in the transcriptional regulation network of Escherichia coli. *Nature Genetics*, *31*(1), 64–68.

Simon, H. A. (1962). The architecture of complexity. In *Proceedings of the American Philosophical Society* (p. 467–482).

Simon, H. A., & Ando, A. (1961). Aggregation of Variables in Dynamic Systems. *Econometrica*, *29*(2), 111–138.

Sipser, M. (2012). *Introduction to the Theory of Computation*. CENGAGE Learning Custom Publishing.

Soare, R. I. (1996). Computability and Recursion. *The Bulletin of Symbolic Logic*, *2*(3), 284–321.

Solé, R. V., & Fernández, P. (2003). Modularity" for free" in genome architecture? *arXiv preprint q-bio/0312032*.

Solé, R. V., & Valverde, S. (2008). Spontaneous emergence of modularity in cellular networks. *Journal of The Royal Society Interface*, *5*(18), 129–133.

Tinbergen, J. (1951). *Econometrics*. Psychology Press.

Turing, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem. *J. of Math*, *58*, 345–363.

Turing, A. M. (1950). Computing Machinery and Intelligence. *Mind*, New Series, *59*(236), 433–460.

Tzerpos, V., & Holt, R. (1998). Software botryology. Automatic clustering of software systems. In *Ninth International Workshop on Database and Expert Systems Applications, 1998. Proceedings* (pp. 811–818). Presented at the Ninth International Workshop on Database and Expert Systems Applications, 1998. Proceedings.

Valiant, L. (2013). *Probably Approximately Correct: Nature's Algorithms for Learning and Prospering in a Complex World*. Basic Books.

Von Neumann, J. (1945). First Draft of a Report on the EDVAC. *IEEE Annals of the History of Computing*, *15-1993*(4), 27–75.

Wagner, A., & Fell, D. A. (2001). The small world inside large metabolic networks. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, *268*(1478), 1803–1810.

Wagner, G. P. (1996). Homologues, Natural Kinds and the Evolution of Modularity. *American Zoologist*, *36*(1), 36–43.

Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of "small-world" networks. *Nature*, *393*(6684), 440–442.

Weiser, M. (1981). Program slicing. In *Proceedings of the 5th international conference on Software engineering* (p. 439–449). IEEE Press.

Weiser, M. (1982). Programmers use slices when debugging. *Communications of the ACM*, *25*(7), 446–452.

White, D. R., & Reitz, K. P. (1983). Graph and semigroup homomorphisms on networks of relations. *Social Networks*, *5*(2), 193–234.

White, J. G., Southgate, E., Thomson, J. N., & Brenner, S. (1986). The Structure of the Nervous System of the Nematode Caenorhabditis elegans. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, *314*(1165), 1–340.

Wilke, C. O., Wang, J. L., Ofria, C., Lenski, R. E., & Adami, C. (2001). Evolution of digital organisms at high mutation rates leads to survival of the flattest. *Nature*, *412*(6844), 331–333.

Wilkinson, D. M., & Huberman, B. A. (2004). A method for finding communities of related genes. *Proceedings of the National Academy of Sciences*, *101*(suppl 1), 5241–5248.

Wimsatt, W. C. (1976). Reductive Explanation: A Functional Account. In R. S. Cohen, C. A. Hooker, A. C. Michalos, & J. W. V. Evra (Eds.), *PSA 1974*, Boston Studies in the Philosophy of Science (pp. 671–710). Springer Netherlands.

Winker, P. (1992). *Some notes on the computational complexity of optimal aggregation* (No. 184). Diskussionsbeiträge: Serie II, Sonderforschungsbereich 178 "Internationalisierung der Wirtschaft", Universität Konstanz.

Wolfram, S. (1984). Universality and complexity in cellular automata. *Physica D: Nonlinear Phenomena*, *10*(1–2), 1–35.

Wolfram, S. (2002). *A New Kind of Science*. Wolfram Media, Inc.

Woodward, J. (2003). *Making Things Happen : A Theory of Causal Explanation: A Theory of Causal Explanation*. Oxford University Press, USA.

Wright, C. D. (2012). Mechanistic explanation without the ontic conception. *European Journal for Philosophy of Science*, *2*(3), 375–394.

Wright, L. (1973). Functions. *The Philosophical Review*, *82*(2), 139–168.

Wu, F., & Huberman, B. A. (2004). Finding communities in linear time: a physics approach. *The European Physical Journal B - Condensed Matter and Complex Systems*, *38*(2), 331–338.

Wuensche, A. (1999). Classifying cellular automata automatically: Finding gliders, filtering, and relating space-time patterns, attractor basins, and the Z parameter. *Complexity*, *4*(3), 47–66.

Yamada, H. (1962). Real-Time Computation and Recursive Functions Not Real-Time Computable. *IRE Transactions on Electronic Computers*, *EC-11*(6), 753–760.

Yourdon, E., & Constantine, L. L. (1979). *Structured design: Fundamentals of a discipline of computer program and systems design* (Vol. 5). Prentice-Hall Englewood Cliffs.

Zhang, L. V., King, O. D., Wong, S. L., Goldberg, D. S., Tong, A. H., Lesage, G., Andrews, B., et al. (2005). Motifs, themes and thematic maps of an integrated Saccharomyces cerevisiae interaction network. *Journal of Biology*, *4*(2), 6.

Zhou, H. (2003a). Network landscape from a Brownian particle's perspective. *Physical Review E*, *67*(4), 041908.

Zhou, H. (2003b). Distance, dissimilarity index, and network community structure. *Physical Review E*, *67*(6).

Zhou, H., & Lipowsky, R. (2004). Network brownian motion: A new method to measure vertex-vertex proximity and to identify communities and subcommunities. In M. Bubak, G. D. van Albada, P. M. A. Sloot, & J. L. Snoep (Eds.), *Computational Science-ICCS 2004*, Lecture Notes in Computer Science (Vol. 3038, p. 1062–1069). Springer.

Zhou, H., & Lipowsky, R. (2006). *The yeast protein-protein interaction map is a highly modular network with a staircase community structure.*