Collaborative theory construction: towards a conversational abstract machine

Philippe Lemoisson

Numéro d'identification:

ACADEMIE DE MONTPELLIER

UNIVERSITE MONTPELLIER II

Sciences et Techniques du Languedoc

THESE

pour obtenir le grade de :

DOCTEUR DE L'UNIVERSITE MONTPELLIER II

Discipline : Informatique
Formation Doctorale : Informatique
Ecole Doctorale : Information, Structures, Systèmes

présentée et soutenue publiquement par :

Philippe Lemoisson

le 15 décembre 2006

COLLABORATIVE THEORY CONSTRUCTION TOWARDS A CONVERSATIONAL ABSTRACT MACHINE

Composition du jury :

NICOLLE Anne, Professeur, Université de Caen :……... Rapporteur
LUZEAUX Dominique, Ingénieur X, HDR :……..... Rapporteur
ANCEAU François, Professeur, CNAM : …….....Examinateur
BAGET Jean-François, CR, INRIALPES : …….....Examinateur
SALLANTIN Jean, DR, CNRS : …….....Directeur de Thèse
CERRI Stefano A., Professeur, Université Montpellier II : ….. Codirecteur de Thèse

Numéro d'identification:

ACADEMIE DE MONTPELLIER

UNIVERSITE MONTPELLIER II

Sciences et Techniques du Languedoc

THESE

pour obtenir le grade de :

DOCTEUR DE L'UNIVERSITE MONTPELLIER II

Discipline : Informatique
Formation Doctorale : Informatique
Ecole Doctorale : Information, Structures, Systèmes

présentée et soutenue publiquement par :

Philippe Lemoisson

le 15 décembre 2006

COLLABORATIVE THEORY CONSTRUCTION TOWARDS A CONVERSATIONAL ABSTRACT MACHINE

Composition du jury :

NICOLLE Anne, Professeur, Université de Caen :……... Rapporteur
LUZEAUX Dominique, Ingénieur X, HDR :……..... Rapporteur
ANCEAU François, Professeur, CNAM : …….....Examinateur
BAGET Jean-François, CR, INRIALPES : …….....Examinateur
SALLANTIN Jean, DR, CNRS : …….....Directeur de Thèse
CERRI Stefano A., Professeur, Université Montpellier II : ….. Codirecteur de Thèse

Remerciements

Merci à Laurent Gille pour avoir soutenu il y a dix ans environ la conception et le développement d'un outil de modélisation/simulation qui portait en germe la plupart des idées assemblées ici ; maintenant que la réflexion a mûri je formule le vœu d'en partager un jour les fruits avec lui.

Merci à mon Directeur de Thèse pour l'invitation au voyage vers les majestueux sommets surgis du choc tectonique entre Mathématique et Philosophie. En gardant le silence sur les ravins à franchir lors des marches d'approche, il a su protéger mon enthousiasme. Et si à ce stade du parcours je chemine encore lourdement loin de son altitude, ce n'est pas faute d'encouragements ou d'injonctions de sa part ; il a je crois usé de tous les moyens subtils pour tenter d'élever mon esprit. Il a aussi veillé à mon riz quotidien en ouvrant des opportunités pour mes talents d'ingénieur.

Merci à mon Codirecteur de thèse pour sa profonde connaissance du paysage varié offert aujourd'hui par la Science Informatique, et pour ses talents de chef d'orchestre dans la musique des relations humaines. Je dois à l'énergie qu'il déploie sans compter pour constituer des équipes enthousiastes et sans frontières culturelles les rencontres les plus marquantes de ces années de recherche.

Merci à Gilles Pindat pour sa curiosité envers mon travail et pour sa collaboration décisive sur le problème qui constitue l'illustration centrale de ce mémoire; un an s'est écoulé depuis sa proposition de mettre à l'épreuve la machine AUSTiN sur la construction à plusieurs d'une stratégie de résolution du Sudoku par le raisonnement.

Merci à Christophe Bourrier, Emmanuel Castro et Cédric Crouzet sans lesquels le prototype actuel n'existerait pas.

Merci aux professeurs et collègues du LIRMM pour leur attitude bienveillante durant ces années passées ensemble.

Merci à tous les aînés dans la Recherche qui ont su expérimenter avec un esprit rationnel, puis élaborer avec art.

Merci à ma femme et à mon fils pour avoir supporté si souvent l'air absent qui accompagne généralement les cogitations prolongées.

Philippe Lemoisson

Résumé détaillé

Le chapitre 1 introduit notre motivation: aider un groupe confronté à une 'situation réelle' à collaborer pour en construire un 'modèle abstrait' intégrant les compréhensions individuelles respectives de façon compositionnelle ; chaque individu doit pouvoir formuler et tester empiriquement sa propre compréhension et *sans aucun effort supplémentaire* contribuer à une théorie partagée par le groupe.

Le chapitre 2 articule dans un cadre unifié des éléments provenant d'explorations conduites dans diverses disciplines :

- sur les mécanismes biologiques sous-tendant la cognition ;
- sur le rôle du langage en tant que vecteur de partage et de formalisation ;
- sur la place des faits expérimentaux dans la démarche de construction de théorie.

Notre cadre conceptuel permet de distinguer des théories partielles générées de façon indépendante par des individus puis d'opérer leur composition logique unifiée sous forme de calcul ; nous prenons alors inspiration des idées exposées dans "Elephant 2000[1]" pour rédiger le cahier des charges d'une machine abstraite conversationnelle :

- qui réalise la composition logique de théories partielles ;
- qui se substitue à l'homme pour la partie 'déduction' ;
- telle que les composants du calcul interagissent à travers des 'actes de langage'[2].

A titre d'exemple, nous abordons la résolution par un groupe d'un 'Sudoku', en montrant comment les stratégies des différents participants devront s'assembler au cours d'un 'calcul conversationnel' confluent.

Le chapitre 3 détaille une machine abstraite conversationnelle basée sur trois actes de langage : les '*assertives*'[2], les '*directives*'[2] et les '*commissives*'[2]. Il débute par un rapide survol des principaux concepts et introduit sur un mode informel l'ensemble des notions qui seront formalisées ensuite :

- les 'énoncés' représentant les connaissances ;
- les 'actes de langage' ;

[1] voir [McCarthy, 1998] et [2.1.2.2]
[2] voir [2.1.2.1] pour les définitions

- les ‘schémas conversationnels normalisés’ qui assurent le traitement de l’information sous forme d’algorithmes à étapes prenant en charge la réécriture automatique des énoncés;
- les ‘conversations’ elles-mêmes comme successions de ‘transitions atomiques’ correspondant aux invocations, aux activations successives et aux révocations des différents ‘schémas conversationnels normalisés’.

Toutes ces définitions sont assorties d’exemples.

Nous présentons ensuite le contrôle de la conversation vue comme un calcul concurrent et donnons la démonstration de ses propriétés : terminaison en temps fini, confluence (le résultat n’est pas affecté par la concurrence), complexité raisonnable. Nous traitons la question de l’expressivité du modèle computationnel en montrant que chaque nœud peut contenir une Machine de Turing.

Un glossaire récapitule les concepts nécessaires au calcul.

Nous proposons enfin une grammaire générative inspirée des graphes conceptuels pour les ‘énoncés’. La ‘résolution collaborative de Sudoku’ est développée comme illustration du calcul basée sur cette grammaire.

Le chapitre 4 adopte le point de vue de l’écriture sous forme de programmes des théories partielles générées par les membres du groupe :

- les programmes AUSTIN sont définis, ainsi que leur composition ;
- une condition d’acceptabilité par la machine abstraite est fournie ;
- les techniques d’analyse des programmes, toutes basée sur de l’analyse de graphe, sont esquissées.

Un prototype Java de la machine abstraite est utilisé pour la ‘résolution collaborative de Sudoku’ afin de montrer comment les choix du chapitre 3 implémentent le cahier des charges du chapitre 2.

Le chapitre 5 fournit quelques arguments en faveur du compromis adopté à mi-chemin entre un ‘modèle déclaratif et contrôlable’ et un ‘modèle impératif et expressif’, et pose la discussion sur la machine conversationnelle AUSTIN autour de deux questions :

- quelle sémantique logique pour le calcul conversationnel ?
- quelle relation aux ‘systèmes multi-experts à architecture blackboard’ ?

Le chapitre 6 fait une synthèse des divers aspects abordés et ouvre des perspectives pour des travaux ultérieurs.

Executive summary

Chapter 1 introduces our motivation: assisting people in the collaborative building of an 'abstract model' of some 'situation of the real world' embedding their respective understandings in a compositional way. Each individual should be allowed to formulate and empirically test his own partial understanding, and *without extra effort* to contribute to a testable theory shared by the whole group.

Chapter 2 articulates in a unified framework the elements of a multi-disciplinary survey touching on:

- the underlying biological mechanisms supporting cognition;
- linguistic communication as a means for sharing and formalizing;
- the role of experimental facts based on real events in the theory building process.

Our framework illustrates the interleaving between individual generation of partial theories and their computer-assisted logical composition. Taking our inspiration in the ideas described in "Elephant 2000[3]", we formulate requirements for a *conversational abstract machine*:

- addressing the logical composition of partial theories;
- substituting human deductive reasoning;
- and where the components interact through speech acts.

We then exemplify our proposition by the collaborative search of a deduction-based strategy for the 'Sudoku filling problem', where the candidate reasoning patterns interact through a confluent conversation.

Chapter 3 presents the full details of a conversational abstract machine using three different *illocutionary acts*: 'assertives'[4], 'directives'[4] and 'commissives'[4]. It starts with a brief overview of the main concepts: 'sentences' and 'real-time rewriting rules' and informally introduces all the necessary notions for which formal definitions are then given:

- 'sentences' supporting *knowledge representation*;
- 'utterances' supporting *illocutionary acts*;

[3] As expressed in [McCarthy, 1998] and pasted in [2.1.2.2]
[4] Definitions according to J. Austin are given in [2.1.2.1]

- ‘conversational patterns’ embedding *information processing* components in multi-steps algorithms;
- ‘conversations’ intertwining ‘atomic transitions’ which concurrently invoke, activate and revoke conversational patterns.

All these definitions are introduced with the help of examples.
We then exhibit the control of the conversation under the shape of a concurrent calculus and give demonstrations of its properties: end in finite time, confluence (the result is not affected by the concurrency), reasonable complexity. We address the question of the expressivity of the computational model by showing that the ‘conversation nodes’ can be Turing Machines.
A Glossary recapitulates all the concepts of the conversational calculus.
We finally propose a generative grammar for sentences inspired by ‘conceptual graphs’. The conversational calculus and the proposed grammar are illustrated in the ‘Sudoku filling problem’.

Chapter 4 shifts to a programming view:

- AUSTIN programs embedding partial theories are defined as well as their composition;
- the condition for acceptability of a program by the abstract machine is given;
- the main reasoning techniques, all relying on graph analysis are sketched out.

A Java prototype is used to run the ‘Sudoku filling problem’ and demonstrate how the choices of chapter 3 fulfil the requirements listed in chapter 2.

Chapter 5 gives some arguments in favour of our chosen trade-off between the controllability of the declarative paradigm and the expressivity of the imperative techniques, and settles a general discussion about the AUSTIN machine around two questions:

i) can we give a logical semantic to the conversational calculus?
ii) our system has been thought for enhancing the collaboration between ‘domain experts’ who are not computer scientists; how can we compare AUSTIN with current ‘multi-experts systems’?

Chapter 6 summarizes the issues and opens directions for future work.

Table of contents

Table of illustrations

1 Introduction

1.1 Initial motivation: the 'PULL'

This work is aimed a supporting collaborative theory building through automated moderation.

As an engineer, the author has spent about 15 years modelling and simulating economic systems, playing the role of a mediator between 'domain experts' and 'computer scientists'. The challenge was to assemble partial understandings of a problem into a global multi-dimensional simulation.

One example has been the simulation of market shares in air transportation as a function of:

- hypotheses on kerosene taxation in European countries applying the Kyoto protocols aimed at reducing the CO_2 emissions;
- hypotheses on how air flight operators adjust their fleet and routes and prices according to local kerosene costs at start and end points;
- hypotheses on the new local demands of traffic;
- …

In such cases, all hypotheses are interdependent and this is described in the context of 'partial theories' (in the given example: linking traffic demand and prices, linking kerosene costs to traffic offer, linking offer to demand …). The challenge is to tune all partial theories provided by the different 'domain experts' in such a way that the global simulation is stable when parameters vary and meets a consensus between experts. Different approaches exist, with common strands:

i) identifying the set of independent concepts (in the given example: operator type, air route, cost type, activity indicator, aircraft type …) which constitute the multi-dimensional reference;

ii) expressing all the input data within this conceptual reference;

iii) expressing the partial theories as rewriting rules operating on the data;

iv) ruling the composition of the partial theories;

v) analyzing the output with respect to each partial theory;

vi) refining both the reference and the partial theories.

The situation is that of 'collaborative theory construction[5]' in the sense of building a unified set of expressions and derivation rules such that appropriate subsets can be interpreted by different experts as a model for their respective observations and beliefs. The process consists of two complementary "movements": one of 'theory discovery', the other of 'theory checking', and the global challenge is a clever distribution of these two movements between humans and computers; a detailed argumentation[6] in favour of *human-computer cooperation* can be read for instance in [Langley, 2000].

Starting from this background the idea of integrating loosely-coupled logical components in a cooperative loop on which non-computer-literate people keep control, which strongly underlies this work, has gradually emerged through inspiring discussions with my supervisors: Professor Jean Sallantin and Professor Stefano A. Cerri.

A first draft of proposition in this direction can be found in [Lemoisson, Cerri, Sallantin, 2005] around two articulated elements:

- a conceptual framework linking individual cognition to knowledge representation during the subsequent cycles of acceptance, refutation and refinement of hypotheses;
- an experimental protocol allowing conversational interactions between humans and artefacts.

The present work deepens are completes the analyses grounding the framework, clarifies the requirements for computer assistance of the process of 'collaborative theory building', and details the specifications of the 'AUSTIN conversational abstract machine' aimed at sustaining it. An example running on a prototype is provided.

1.2 Problem definition

We consider human-computer cooperation in a context where different experts providing partial models are present and where *no* 'super-expert' exists neither for pre-defining the shared *knowledge representation* nor for distributing the *information processing* between them; moreover we must keep in mind that the final users are not computer scientists.

[5] by "theory" we mean a set of testable hypotheses that produce "true sentences", according to the definitions given in [2.1.2].

[6] Pat Langley, in [Langley, 2000], identifies five distinct types of discovery activity : building taxonomies, discovering qualitative laws, discovering quantitative laws that state mathematical relations among numeric variables, building structural models that incorporate unobserved entities, building process models which explain phenomena in terms of hypothesized mechanisms that involve change over time

Before dealing with these two complementary aspects, we must first clarify our perception of the general 'state of the art':

- Powerful formal languages exist for *knowledge representation*, most of them having a very small number of abstract objects ('entities', 'relations', 'properties') obeying a very small number of assembling rules; they allow powerful conceptualization and easy implementation[7]. Because of the power and simplicity of such formalisms, they can be taught and shared and lead to fruitful collaboration between 'domain experts' and 'computer scientists'; we shall take inspiration from one on them in our approach.

- Two main styles exist for embedding the *information processing* into programs. *Imperative* programs make the algorithm explicit and leave the model implicit; they tend to emphasize the series of steps taken by a program in carrying out an action. *Declarative* programs make the model explicit and leave the algorithm and necessary steps implicit; the general use is to consider as 'declarative' the programs written in a purely functional programming language, like a logic programming language, or a constraint programming language. The imperative style would be closer to empirical approaches and would emphasize "expressivity" while the declarative style would be closer to mathematical objects and would emphasize "controllability". While in declarative techniques like LOGICAL PROGRAMMING[8] the modelling part is much constrained by the goal driven approach and requires special skills, in imperative techniques implementing the metaphor of experts collaborating and communicating through a BLACKBOARD[5] the detailed specifications of the communication events and the finding of an efficient control scheme are far from simple. Dealing with *information processing,* the trade-off between "expressivity" and "controllability" seems the key of success and will be our main challenge.

In the light of these preliminary comments, 'supporting collaborative theory building though automated moderation' addresses two issues:

[7] Such languages have been integrated in methods used in business companies for reshaping Information Systems; for instance the "Merise" method has been widely used in France since 1975.
[8] A detailed discussion about "blackboard architectures" is reported in Chapter 5

i) representing the knowledge, i.e.: organizing a considerable set of data and more or less complex decision rules into 'partial theories', in a manner *as close as possible to the experts' initial conceptual schemes;*

ii) processing the information, i.e.: assembling the 'partial theories' into a unified set of expressions and providing a controllable global derivation, with *as little extra effort as possible*.

1.3 Structure of the thesis

This thesis is organized in four major parts, before we summarize the issues and discuss the further possible investigations in Chapter 6.

The first part is Chapter 2; it starts by a multi-disciplinary survey touching on:

- the underlying biological mechanisms supporting cognition;
- linguistic communication as a means for sharing and formalizing;
- the role of experimental facts based on real events in the theory building process.

Three worlds are associated to the respective aspects of 'cognition', 'language' and 'real events' and then articulated in a unified framework. The purpose of this preparatory part is to clearly distinguish between i) the internal plasticity of our minds, ii) the external formality of languages and iii) the environment in which we live, interact, interpret events and experiment. Two different modes of thinking are defined and put in relation; the questions about meaning and how it can be shared though language is addressed, and a semi-formal definition is given for the central notion of theory. The contrasted resulting picture is strongly influenced by some "default positions" that we expose at the beginning, and also by our past readings in neurobiology. In order to build our own theory in accordance with our own framework, we illustrate the whole cycle by the 'Sudoku filling problem' which plays the role of a sharable experiment. At the end of the first part, we refine our problem definition by enouncing the main characteristics of a 'conversational abstract machine' aimed at moderating the 'collaborative theory building' cycle.

The second main part is Chapter 3, where the 'AUSTIN conversational abstract machine' is designed on the basis of a semi-formalized calculus. We have found very difficult in this central chapter to disentangle the formal aspects from the operational aspects, the major reason for this may be that we our whole approach has been to 'make the calculus work' before 'describing' it. Another reason, which we are now aware of, is that a

complete formalization probably requires very high level mathematical objects that we do not master; the whole description therefore remains at a low abstract level, quite influenced by the operational semantics. As a consequence, we rely on the use of several examples in order to try and keep the attention of the reader.

A first example is given in [3.1] in order to introduce the main concepts; it consists in a very simple conversation aimed at answering collectively a single question.

A second example is developed in [3.2] to illustrate the formalized notion of 'multi-steps algorithm' called 'conversational pattern'; it describes a conversation between a client and a service provider, when the questions asked at a given state of the conversation depend on previous answers to previous questions.

The third and major example is presented in [3.8] and consists in the complete solution of the previously described 'Sudoku filling problem'; this example exemplifies both the calculus and a generative grammar for sentences proposed in [3.7].

These three examples compute already built "partial theories" in the context of a single closed conversation; in order to well understand the limits of this paradigm, the open situation of collaborative building of a "free encyclopaedia that anyone can edit" has been addressed in [8.3].

Although the formalization of the calculus has not reached full maturity, it allows the demonstration of two interesting 'theorems': one about the 'ending in finite time', the other about 'confluence', both corresponding to major requirements.

We then emphasize the proof of concept in Chapter 4 and put our efforts in describing how the compositionality is fully effective, and also how the iterative refinement of the theory is empowered by the capability of AUSTIN to accept a high level conceptual language for knowledge representation and to naturally provide a graphical basis for the analysis of programs.

We end in Chapter 5 by examining the proposed software system, its logical semantics and its similarities with 'blackboard architectures' in the light of two different fields of research: 'logical programming' and 'multi-expert systems'.

2 Collaborative theory construction

We may collaborate on a friendly mode, and yet not agree … sharing the understanding of some phenomenon of the real world among two or more people is most of the time a challenging adventure, with no guarantee of success. While some emit 'likely hypotheses' matching their own observations and build predictions upon them, others refute those by bringing in new facts in contradiction with expected results … therefore new hypotheses must replace the old ones, until the process eventually converges on a consensus.

In this interactive process, the individual propositions (or partial theories) of the contributors are expressed and shared through more of less formal languages; which means that the 'theory builders' are in position to formally agree or disagree with expressed facts or predictions. However it remains the duty of each contributor to check whether some new hypothesis emitted by another will impact or not his own predictions.

Our focus is on the empowerment of the whole cycle by some kind of formal system logically ruling the composition of partial theories.

To achieve this, we first need to examine how theories appear in human minds, how they are expressed, shared and checked within languages, and then try and find a way for delegating the composition of concurrent hypotheses to some 'abstract machine'[9]. This chapter is organized in two parts:

2.1 mainly consists in a general framework which articulates elements gathered a multi-disciplinary survey aimed at collecting requirements for the empowerment of 'collaborative theory building'.

[2.1.1] is an attempt to clarify our perception/action relation to the world; it leads to an informal definition for 'selectionist thinking' as a biology-based way of understanding the word 'understanding';

[2.1.2] analyzes the role of languages, their relation to social interaction and meaning, the consequences of their formalization and how they shelter 'logical deduction';

[9] An abstract machine, also called an abstract computer, is a theoretical model of a computer hardware or software system used in Automata theory. Abstraction of computing processes is used in both the computer science and computer engineering disciplines and usually assumes a discrete time paradigm.

[2.1.3] addresses the process of 'collaborative theory building' as a whole and intertwines 'selectionist thinking' with 'logical deduction' in a single framework.

[2.1.4] finally leads to some generic requirements for an abstract machine in charge of the automatic composition and derivation of partial theories:

- logical components interacting through simplified *illocutionary acts*[10] (*assertives, directives* and *commissives*);
- direct computability of facts and hypotheses constituting the partial theories, with the production of predicted results in the same language as facts;
- *compositionality* with respect to the partial theories.

2.2 exemplifies the resulting framework on the 'Sudoku' example and outlines the main features of an abstract machine sustaining the collaborative inference of theories from facts. This example is based on a close world of finite possible configurations on which a theory can easily be checked. The choice of such an example will help us illustrating how successive drafts of the theory can be tested and improved. In the 'Sudoku' example, the desired theory is a set of rules for filling a grid with digits in such a way that some global properties are preserved; the successive drafts of the desired theory will be tested through 'conversational derivation' involving concurrent algorithms in charge of filling distinct regions of the grid.

2.1 Thoughts about minds, languages and computation

To start with our analysis about the 'what and how of theory building', we need some philosophical default positions. Our framework will rely on those adopted in [Searle, 1999]:

- "There is a real world that exists independently of us, independently of our experiences, our thoughts, our language;
- We have direct perceptual access to that world through our senses, especially touch and vision;
- Words in our language, words like rabbit or tree, typically have reasonably clear meanings. Because of their meanings, they can be used to refer to or talk about real objects of the world".

This proposition is illustrated in Figure 1 by three 'worlds' associated with three colours:

[10] The definitions for illocutionary acts, assertives, directives and commissives are given in [2.1.1.2]

- the BLUE 'world of mind' consisting in all the hidden, intimate processes occurring in individual brains which allow perceptual categorization, movement coordination as well as concept formation;
- the GREEN 'world of linguistic expressions' supporting communication between individuals (acted, spoken, or written languages) and mediating their relation to the world. Our coarse-grained analysis will address the wide spectrum ranging from natural languages of daily life to formal languages of mathematics and informatics; therefore 'linguistic expression' has to be understood as 'expression in a language', be it natural or artificial.
- the ORANGE 'real world' where events or 'changes of state' obey the laws of causality, where theories can be tested on the basis of facts, and where formulas and syntactic rules of the green world can be electronically reified and computed.

Between these three 'worlds' we consider two links:

- "A" for action/expression, roughly meaning that individuals act, speak and write, therefore externalizing something. The sequence "BLUE → A/action → ORANGE" figures the externalization of an event in the outside physical world, like when one bites into an apple. The sequence "BLUE → A/GREEN expression → ORANGE" illustrates an externalization matching some linguistic or pre-linguistic pattern at a socially shared symbolic level, for instance when two people shake hands.
- "I" for perception/interpretation, roughly meaning that individuals have their inner state influenced by, and build associations upon, what happens outside. The sequence "ORANGE → I/perception → BLUE" figures what happens in Proust's mind when he bites into a Madeleine. The sequence "ORANGE → I/ GREEN interpretation → BLUE" happens in the mind of the reader when signs on a sheet of paper appear as constituting the memoir of a thesis.

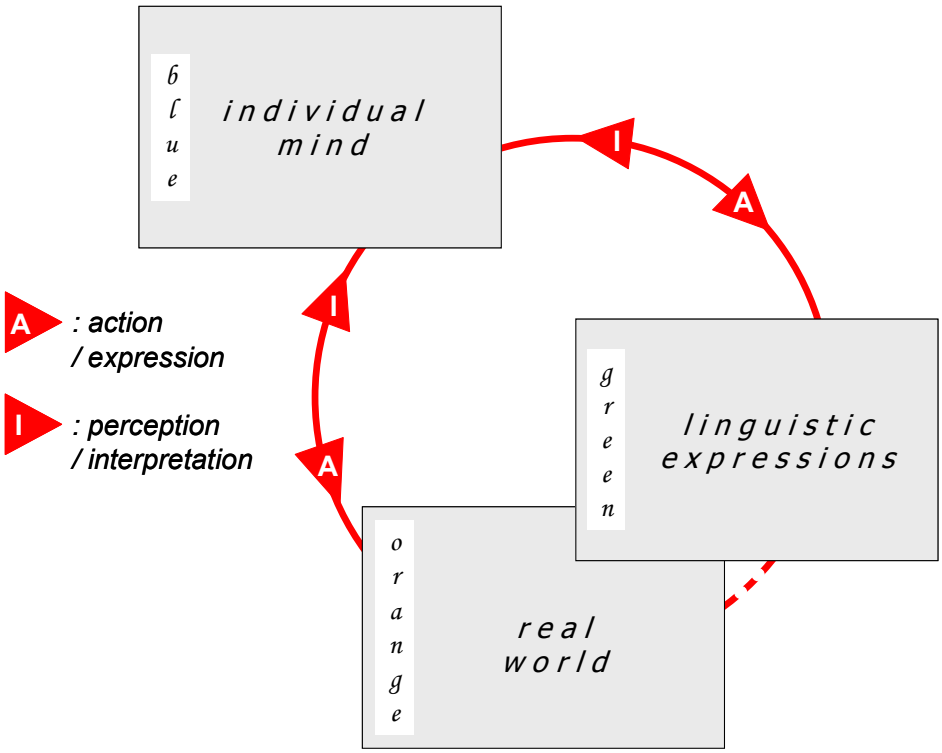


Figure 1: mind, language and world

This very simplified view needs some comments.

The use of the term 'world' is quite similar to [Lorenz & Popper, 1995]; the three 'worlds' may be characterized as i) "physical world of objects and instruments/ ORANGE", ii) "world where the binding of experiments takes place/ BLUE" and iii) "world of the products of human mind/ GREEN".

The "A/I" relation between two individual minds has to be mediated by the outside world, and *can or not* be mediated by language; our focus will be the case when individuals communicate through linguistic expressions. Moreover, sharing some theory through language has to be materialized (associated to a stable ORANGE trace), either with the help of chalk and blackboard or with the help of a computer, and our aim is to emphasize the potential role of the computer.

Another important comment is that the three 'worlds' in Figure 1 worlds do not belong to a flat picture. The real 'world' includes ourselves, but is out of reach: we only have access to our own representation of it, through the mediation of our mind. Moreover,

when we want to communicate such a representation, we enter the ‘world’ of linguistic expressions.

Selecting a starting point in Figure 1 for exposing the framework has therefore not been easy, and the whole enterprise may seem quite questionable. It should be noticed however that our choice of these three ‘worlds’ is guided, and almost imposed, by our problem definition:

- we address the question of collaboration and must therefore distinguish between individual minds (BLUE);
- these minds are not directly accessible: we only know about the partial theories elaborated by the others through emitted linguistic expressions (GREEN);
- in order to improve and compose partial theories, we need to rely on some stateful trace, i.e. to exhibit the proof of concept in the real world (ORANGE).

2.1.1 Blue, the "world of mind"

There is no mystery about how artefacts work … what intelligence is and how our minds relate us to the world is a much more difficult question! A lot of very intelligent people have addressed this question under various perspectives long before we undertook this thesis. It even happens that a whole field of research in "informatics" is categorized as "Artificial Intelligence[11]", for which a very nice (western) story can be found in [Cruvier, 1993]. Most of its pioneers, like Marvin Minsky, Herbert Simon or Allen Newell, all three winners of many prizes and distinctions[12], are also famous researchers in the field of Cognitive Science.

The starting point of our investigation has been to browse though their respective contributions and theories. Among many publications of Herbert Simon stands "Models of Thought" (1979 / his collected papers on human information-processing and problem-solving). Alan Newell was in 1987 the author of "Unified theory of Cognition[13]" on the basis of the "William James lectures" he delivered at Harvard University. In the same year 1987, Marvin Minsky wrote the well known "The Society of Mind".

We will not report in detail about these books ... a moment came when we decided to ground our own understanding on robust biological evidence rather than on well-thought top-down model. The major reasons for this are:

- Firstly, the psychological theories of Cognition might in fact not be ready for unification. In [Newell, 1990], the provided "human cognitive architecture" describes several well-identified functional layers which seem a very tempting basis for exploring further computerization of the thinking, which has been done through the development of SOAR [Laird et al, 1987]; this architecture is however qualified as "speculative" by the author himself and the overall approach is clearly from pre-supposed systems to propositions, almost totally free of experimental data, except from data about "how long it takes to move a finger". Moreover, the more recent theory of "situated cognition" [Clancey, 1997] puts Newell's approach in discussion by insisting on the permanent relation between what we have labelled BLUE and ORANGE, as we shall refer to in [2.1.1.5].

[11] Artificial Intelligence (AI) is defined as intelligence exhibited by an artificial entity. Such a system is generally assumed to be a computer. [Wikipedia].
[12] including the Turing Award obtained by Minsky in 1969, Simon and Newell in 1975
[13] it may be noted that Alan Newell's curriculum has no flavour of Cognitive Psychology: "My basic education is in physics and mathematics. My graduate education is in industrial administration … I've lived my academic life in a computer science department …".

Exploring all the currently available theories of Cognition is clearly out of the scope of this work; we shall try instead to collect coarse-grained features about how our brain works, and especially put the focus on what it is not.

- Secondly, we suspect the tight frequentation of logical artefacts to introduce a bias in understanding human thinking. The question of whether or not human mind can be usefully compared to a computer will be analyzed in the light of the "very bottom" layers of neurobiology and will receive a clear negative answer. The positive news however for a reader of "The Society of Mind" is that the description of mind given by neurobiology will not contradict Minsky's ideas of human intelligence as resulting of the interactions of simple mindless parts[14].
- Thirdly, the integration operated in years 1940 of two major theories: Charles Darwin's theory of the evolution of species by natural selection and of Gregor Mendel's theory of biological genetic inheritance, usually referred to as "modern evolutionary synthesis", seems to us a robust source of inspiration, not sufficiently exploited in Cognition… and this did encourage us to go rather deeply into the "theory of mind based on selectionism" which is further exposed. It will lead to the notion of "selectionist thinking" as the hidden intimate brain activity generating and complementing the "logical thinking" that we exhibit in formalized theories.

In order to explore the BLUE world, we present hereafter the Theory of Neuronal Group Selection, as a theory having passed with some success the test of social controversy and building upon a very small number of robust biological concepts. It is the major research work of a Nobel Prize whose books have accompanied us during the last ten years: Gerald M. Edelman[15].

Apart from requirements induced by our problem definition, the use of three colours for mind, language and computation has been motivated by the conviction that our brain does not work like a computer, nor contains any symbolic representations. Building upon such a conviction requires some kind of argumentation … this is why a number of long quotes have been kept inside the text rather than grouped at the end of the document; we hope this preliminary [2.1.1] sub-section to be of some interest and kindly invite

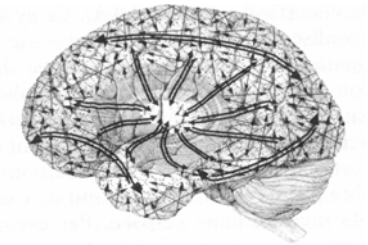
[14] Minsky quotes in [Minsky, 1998]: "What magical trick makes us intelligent? The trick is that there is no trick. The power of intelligence stems from our vast diversity, not from any single, perfect principle".
[15] Gerald M. Edelman received the Nobel Prize for Medicine in 1972; he is Director of The Neurosciences Institute and President of Neurosciences Research Foundation, the publicly supported not-for-profit organization that is the Institute's parent.

passionate or sceptical readers to go through the detailed argumentation and impressing bibliography enclosed in three books [Edelman, 1988], [Edelman, 1992] and [Edelman & Tononi, 2000][16]. On the other hand, any reader ready to accept right away that "human memory is NOT a store of fixed or coded attributes" will find most of this sub-section superfluous and should go directly to [2.1.2] which is more closely related to our further topics.

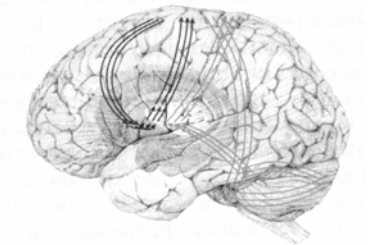
2.1.1.1 A quick review of neuroanatomy

Searching for how the mind works, we have to look into the brain: the following anatomical description has been freely adapted from [Edelman & Tononi, 2000]:

General features: The adult human brain weighs about 3 pounds and contains about 100 billion nerve cells, or neurons… the cerebral cortex contains about 30 billion neurons and 1 million billion connections, or synapses. Figure 2 shows the principles of organization at a high level of anatomical order:



A. the highly interconnected thalamo-cortical system;



B. a set of parallel, unidirectional chains that link the cortex to the cerebellum, the basal ganglia and the hippocampus;

C. a diffuse set of connections allowing small nuclei able to release chemicals to huge portions of the brain, if not to all of it: value systems.

Figure 2: the brain's macro-anatomical organization

[16] From *Nature* about this book: "Explaining consciousness has become the Holy Grail of modern science. Any reckoning of who has found the true path is surely premature. Nevertheless, the account of consciousness provided by Edelman and Tononi is certainly highly plausible and can be recommended as one of the most ambitious accounts around"

Electrical signalling and biochemical traces in synapses: Our mental activity relies on electrical signalling inside neurons and chemical changes in synapses. The general scheme is simple enough to be taught at school, but it becomes quite complex when we take a closer look: "The brain contains a variety of chemical called neurotransmitters and neuromodulators that bind to a variety of receptors and act on various biochemical ways … as a result of the release of the neurotransmitters, electric signalling not only takes place, but leads to change the biochemistry and even in gene expression of the target neuron … this molecular intricacy and the resulting dynamics superimpose several layers of variability on that of the neuroanatomical picture … metaphorically, we can say that we house a jungle in our heads."

2.1.1.2 The brain is a jungle … with reentry!

Following up with [Edelman & Tononi, 2000], we may now zoom on the jungle and discover major features:

- each brain is significantly unique;
- variability upon brains has played a major role in the emergence of brain functions;
- variability upon time inside a single brain affects the way we remember things and events;
- value systems can signal to the entire nervous system the occurrence of a salient event;
- recursive interchange of parallel signals between reciprocally connected areas of the brain continually coordinates the activities of these area's maps to each other in space and time; this key feature is called "reentry".

The following long quote will help and start convincing the reader that our mind cannot work like a computer:

"We know that the brain is interconnected in a fashion no man-made device yet equals. First the billion and billions of connections that make up a brain's connections are *not exact* … Although the overall pattern of connections of a given brain area is describable in general terms, the microscopic variability of the brain at the finest ramifications of its neurons is enormous, and this variability makes each brain significantly unique. These observations present a fundamental challenge to models of the brain that are based on instruction or computation. As we shall see, the data provide strong grounds for so-called selectional brain-theories that actually *depend upon variation* to explain brain function. Another organizing principle that emerges from the picture we are building is that in each brain, the consequences of both developmental history and experiential history are uniquely marked. For

example, from one day to the next, some synaptic connections in the same brain are likely not to remain exactly the same; certain cells will retract their processes, others will have extended new ones, and certain others will have died, all depending on the particular history of that brain. The individual variability that ensues is not just noise or error, but can affect the way we remember things and events. As we shall see, it is also an essential element governing the ability of the brain to respond to and match the countless unforeseeable scenes that may occur in the future…The world certainly is not presented to the brain like a piece of computer tape containing an unambiguous series of signals. Nonetheless, the brain enables an animal to sense the environment, categorize patterns out of a multiplicity of variable signals, and initiate movement … *Value systems* which signal to the entire nervous system the occurrence of a salient event and influence changes in the strength of synapses are typically not found in man-made device… Finally, if we consider neural dynamics (the way patterns of activity in the brain change with time), the most striking special feature of the brains of higher vertebrates is the occurrence of a process we have called *reentry*. Reentry … is the ongoing, recursive interchange of parallel signals between reciprocally connected areas of the brain, an interchange that continually coordinates the activities of these area's maps to each other in space and time… Indeed, if we were asked to go beyond what is merely special and name the unique feature of higher brains, we would say it is reentry. There is no other object in the universe so completely distinguished by reentrant circuitry as the human brain. Although a brain has similarities to a large ecological entity like a jungle, nothing remotely like reentry appears in any jungle."

2.1.1.3 The theory of neuronal group selection

Building upon the well known notions of *neurons* and *synapses* Edelman first defines *neuronal groups* [17] as the basic functional components; these components being themselves grouped in reciprocally connected (reentrant) *neural maps*. His theory of neuronal group selection has three main tenets, as illustrated in Figure 3 [Edelman & Tononi, 2000]:

[17] groups of strongly coupled neurons "firing" synchronously

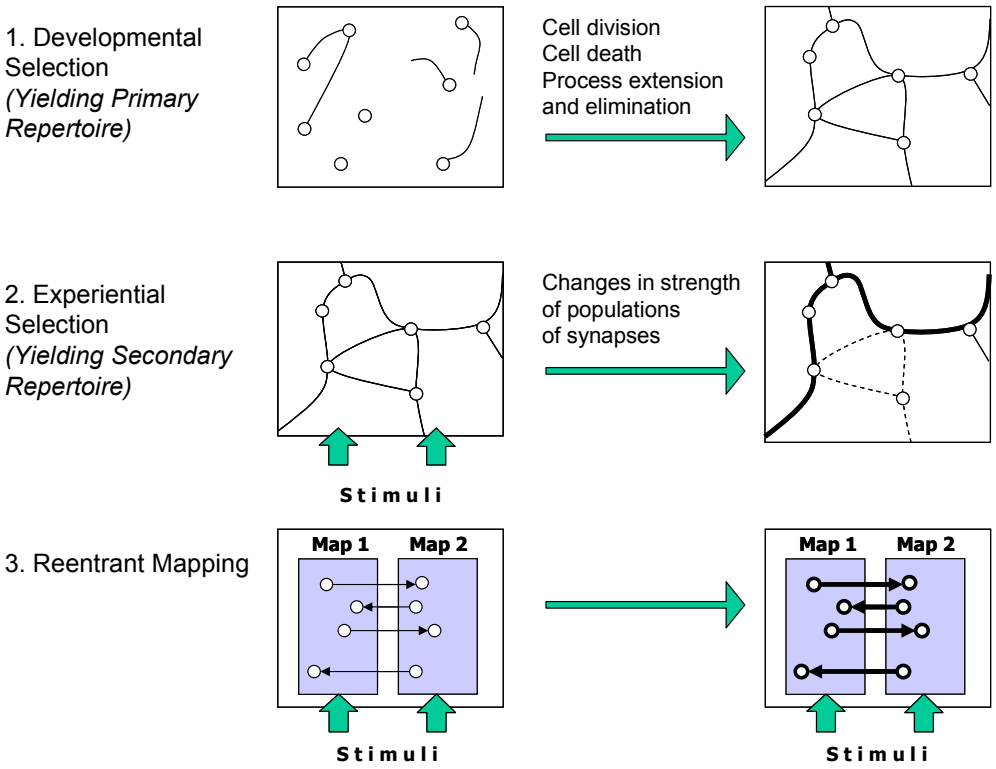


Figure 3: Edelman's Theory of Neuronal Group Selection

1. *Developmental selection*: during the early development of individuals in a species, formation of the initial anatomy of the brain is certainly constrained by genes and inheritance. But from early embryonic stages onward, the connectivity at the level of synapses is established, to a large extent, by somatic selection during each individual's ongoing development… starting from an immense and diverse repertoire of neural circuits, neurons strengthen and weaken their connections according to their individual patterns of electrical activity: Neurons that fire together, wire together. As a result, *neurons in a group* are more closely connected than neurons of other groups.
2. *Experiential selection*: overlapping this early period and extending throughout life, a process of *synaptic selection* occurs within the repertoires of neuronal groups as a result of behavioural experience… This selectional process is constrained by brain signals that arise as a result of the activity of diffusely projecting *value systems*, a constraint that is continually modified by successful output.
3. *Reentry*: the correlation of selective events across the various maps of the brain occurs as a result of the dynamic process of reentry. Reentry allows an animal with a variable and uniquely individual nervous system to partition an unlabeled world into objects and events in the absence of a homunculus or computer program. Reentry

leads to the synchronization of the activity of neuronal groups in different brain maps, binding them into circuits capable of temporally coherent output; it is thus the central mechanism by which spatiotemporal coordination of diverse sensory and motor events takes place. Let us imagine a string quartet in which the players would be linked by myriads fine threads that coordinate their individual performances; in our brains, the "threads" are actually parallel, reciprocal fibres connecting separate maps… reentry assures the integration that is essential to the creation of scene in primary consciousness.

2.1.1.4 A model for associative recall

In order to exemplify in a functioning model the propositions of the above theory, a "Recognition Automaton" was built on the basis of its major premises [Edelman and Reeke, 1982] [Reeke and Edelman, 1984].

The schematic plan of this automaton, illustrated in Figure 4 is the following:

- At the top is an "input array" where stimuli are presented as patterns of light and dark picture elements an a lattice; the input activates in parallel two networks called "Darwin" (left) and "Wallace" (right)
- The Darwin (left) network is designed to respond to local features such as line segments oriented in certain directions of having certain bends, and consists of a "recognizer (R)" connected topographically to the input array, and a higher level called "recognizer of recognizers (R of R)" which gives an abstract transformation of the initial stimulus, represented at the bottom.
- The Wallace (right) network acts as a feature correlation: it first keeps track of the topology of the input (continuity of lines, junctions between lines of various types), and at a higher level maps this on "Recognizer of Motion (R of M)", represented at the bottom, which is insensitive to both rigid and non-rigid transformations of the input, and therefore tends to respond to class characteristics of whole families of related stimuli.
- "R of R" and "R of M" (at the bottom) are reentrant maps.

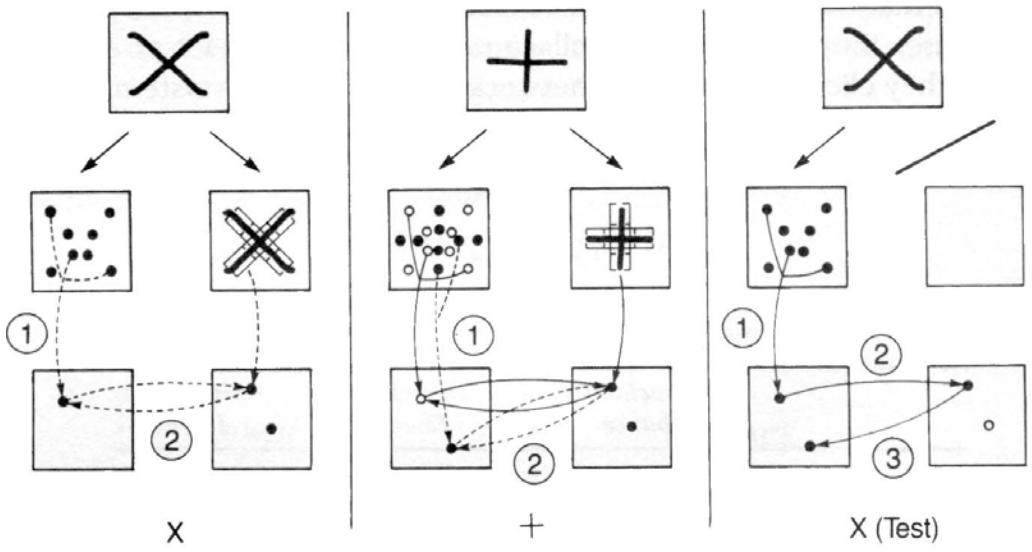


Figure 4: a recognition automaton

Left panel of the figure: When a first stimulus "X" is presented both to Darwin and Wallace, "R" gives the expected topographic response; "R of R" gives a unique pattern characteristic of that stimulus and at the same time, a trace occurs in Wallace, eliciting an appropriate pattern of response in "R of M". Therefore connections that happen to join responding groups (in dotted lines) are strengthened (they will appear in solid lines after this selective strengthening has taken place).

Central panel of the figure: When a second stimulus "+" is presented both to Darwin and Wallace, "R" gives a new topographic response; "R of R" gives a *new* pattern characteristic of that new stimulus and at the same time, a trace occurs in Wallace, eliciting the *same* pattern of response in "R of M", since the input "+" is topologically equivalent to "X". New connections between responding groups (in dotted lines) are strengthened.

Right panel of the figure: The association is tested by presenting "X" again, but to Darwin only! "R of R" gives the same pattern characteristic as before, thus eliciting in "R of M" the common pattern of response to both the "X" and the "+". Pathways (3) then permits "R of M" to stimulate the pattern corresponding to the "+" in "R of R", even though the "+" is not present as an input.

2.1.1.5 Human memory is not a store of "fixed or coded attributes"

Now we come to the delicate point … and also to a good opportunity for establishing a sharp and definitive distinction between the BLUE and GREEN worlds.

Theory building needs coded representations. We give birth to such representations by expressing our individual (blue) associations through (green) languages… yet human memory relies on no intrinsic codes! This idea was quite controversial when Edelman first expressed it some twenty years ago. In order to convince a reader unfamiliar with the question, we again borrow large extracts from [Edelman & Tononi, 2000]:

"There is a widespread assumption that, at least in cognitive functions, the brain is fundamentally concerned with representations and that what is stored in the memory is also some sort of representation… this idea carries with it a huge burden … In the case of human working with computers, semantic operations occurring in the human operator's brain, not in the computer, are necessary to make sense of the coded syntactical strings that are stored physically in the computer … The problem the brain confronts is that signals from the world do not generally represent a coded input. Instead they are potentially ambiguous, are context-dependent, and are not necessarily adorned by prior judgments about their significance… The flaws in yielding to the temptation of saying that the brain represents are obvious: there is no pre-coded message in the signal, no structures capable of the high-precision storage of a code, no judge in nature to provide decisions on alternative patterns, and no homunculus in the head to read a message. For these reasons, memory in the brain cannot be representational in the same way as it is in our devices.

….

The cerebral cortex alone is not sufficient to bear the burden of perceptual categorization and control of movement. That burden is carried out by a *global mapping* that relates the animal's movement and changing sensory input to the action of the hippocampus, basal ganglia, and cerebellum as they connect to cerebral cortex… The activity of a global mapping reflects the fact that perception generally depends on and leads to action. When one moves one's head to follow a moving target, the motor and sensory portions of a global mapping continually readjust. In other words, categorization does not occur solely in a cortical sensory area that then executes a program to activate a motor output. Instead, the results of continual motor activity are considered to be an essential part of perceptual categorization…

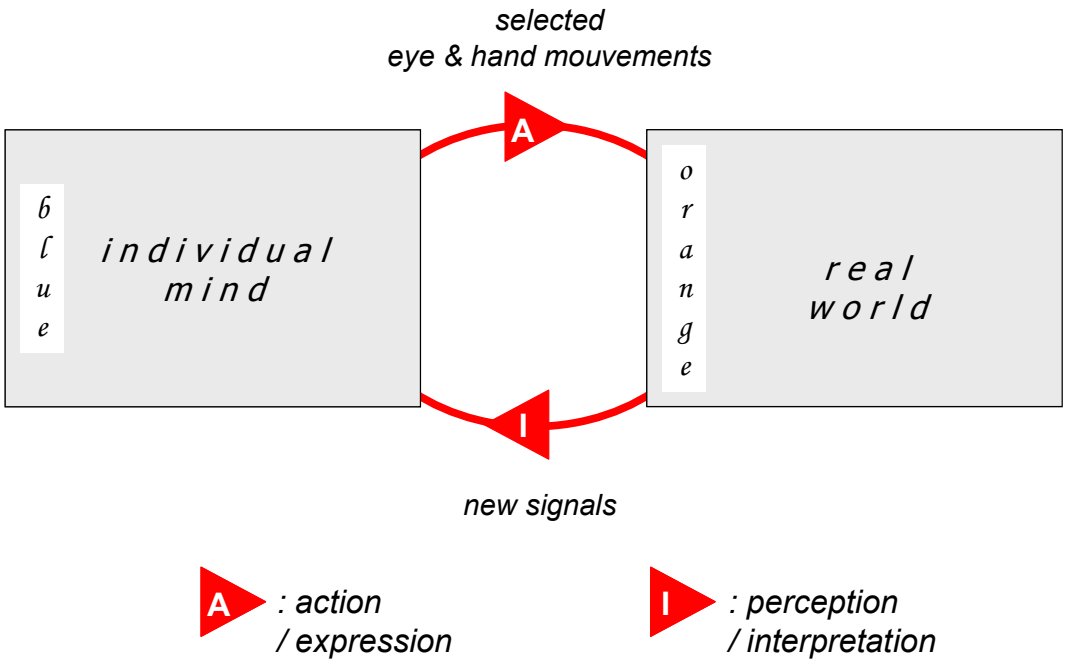


Figure 5: global mappings & perceptual categorization

When we prepare to grasp a glass and drink, a whole set of different circuits, modified by previous synaptic changes, are called up … memory in global mappings is not a store of fixed or coded attributes to be called up and assembled in a replicative fashion as in a computer. Instead memory results from a process of continual re-categorization, which, by its nature, must be procedural and involves continual motor activity leading to the ability to repeat a performance _ in our example, grasping a glass. The ongoing synaptic changes in global mappings that occur as a result of such rehearsals favour degenerate sets of pathways with similar outputs. The contribution of global mappings to memory also carries the major burden of unconscious performance in the brain."

Figure 5 above illustrates in the context of our framework how perceptual categorization is simultaneous with, and reciprocally influenced by action. It is fully in line with the "Theory of situated cognition" of [Clansey, 1997]: "The theory of situated cognition, as I present it here, claims that every human thought and action is adapted to the environment, that is, situated, because what people perceive, how they conceive in their activity, and what they physically do develop together. From this perspective, thinking is a physical skill like riding a bike. In bicycling, every twist and turn of the steering wheel and every shift in posture are controlled not by manipulation of physical equations learned in school, but by a re-coordination of previous postures, ways of seeing, and motion sequences. Similarly, in reasoning , as we create names for things, shuffle around sentences in a paragraph, and interpret what our statement means, every step is controlled not by

rotely applying grammar descriptions and previously stored plans, but by adaptively re-coordinating previous ways of seeing, talking and moving."

Edelman gives some more light to concept formation in terms of a "meta-mapping" relying of the same elementary mechanisms:

"It has been proposed that *concepts*[18] arise as the result of the mapping of the brain itself on the activity of the brain's different areas and regions. By these means, various common features of responses to different signals can be abstracted_ for example, general features of a given form of object motion may be abstractly obtained when the brain, say of a cat, can register a particular state (described here verbally for explanatory purposes) as "cerebellum and basal ganglia active in pattern *a*, neuronal groups in pre-motor and motor regions engaged in pattern *b*, and visual submodalities *x*, *y*, and *z* simultaneously interactive". Higher-order maps register these activities and yield an output corresponding to the notion that an object is moving forward in reference to the cat's body. Forward motion is a concept. Of course, no words are involved. No simple combination of the maps that are reentrantly interactive to yield perceptual categorizations can lead to this abstractive capability. What is required is higher-order mapping by the brain itself of the categories of brain activity in these various regions."

To end with, he once more emphasizes the evolutionary approach which sustains his whole theory:

"Characteristics of the brain giving rise to memory without coded representations are just those characteristics that one would find in a selectional system:

- a set of degenerate neural circuits making up a diverse repertoire;
- a means of changing the synaptic populations upon receiving various input signals;
- and a set of value constraints that increase the likelihood of repetition of an adaptive or rewarding output regardless of which degenerate circuit is used.

…

In reaching for a glass, the satisfaction of thirst will activate *value systems* and lead to the selection of a number of circuits appropriate for performing this action. By these means, structurally different circuits within the degenerate repertoires are each able to produce a similar output, leading to repetition or variation of the act of reaching. Their activity gives rise to the associative properties of memory; for example, an act can trigger another act, a word can trigger other words, or an image can provoke a narrative. These associative properties arise materially from the fact that each different

[18] By concept, we do not mean a sentence or proposition that is subject to the test of the philosopher's or the logician's truth tables. Instead we mean the ability to combine different perceptual categorization related to a scene or an object and to construct an "universal" reflecting the abstraction of some common feature across a variety of such percepts.

member of the degenerate set of circuits used at different times has different alternative network connections… Besides guarantying association, the property of degeneracy also gives rise to the remarkable stability of memorial performance. In a degenerate system, there are large numbers of ways of assuring a given output. As long as a sufficient population of subsets of circuits remains to give an output, neither cell death nor changes in a particular circuit or two, nor switches in the contextual aspects of input signals will generally be sufficient to extirpate a memory. Thus, non-representational memory is extraordinarily robust… Such a memory has properties that allow perception to alter recall and recall to alter perception. It has no fixed capacity limit, since it actually generates "information" by construction. It is robust, dynamic, associative and adaptive."

Our memory is not a store of fixed or coded attributes to be called up and assembled as in a computer; instead, our memory results from a process of continual re-categorization; it is robust, dynamic, associative and adaptive. Coming back to the key question of *information processing* in human minds, we find at the very end of [Edelman, 1988] all the ingredients of the answer assembled in a "big picture" linking information processing to the Theory of Neuronal Group Selection :

"How can a brain operating upon a world without labels by means of neuronal group selection ultimately process information, as it certainly must do in human beings? … To resolve the issue in a general fashion requires linking neuronal group selection to information or learning transmitted between individuals, and ultimately linking such selection to the social transmission and extensive information processing that actually occurs in species such as humans … What is required to connect the theory of neuronal group selection via true learning to information processing is the emergence of a sequence of dependent processes during evolution. These processes are:

1. perceptual categorization via neuronal group selection
2. adaptive learning based on imitation or convention communicated between at least two individuals of species, constituting information processing
3. evolutionary selection of those individuals whose repertoires of neuronal group selection make possible the quickest or most efficient learning of this kind."

Edelman's view[19] is illustrated in Figure 6 where two "Darwinian" loops intertwine:

[19] We have replaced "communication" by "linguistic expressions" and "adaptive behaviour leading to action & Sensory signals" by "real world".

- the grand loop explains the emergence of the primary repertoire (including our *value-systems*!) in our complex brain as the result of a long evolution process based on intraspecies and interspecies competition;
- the small loop explains the wiring of the secondary repertoire by selection mechanisms applied to individual populations of neuronal groups immerged in experiential situations; this selection is ruled by our *value-systems*.

It appears from this picture that the "modern evolutionary synthesis" provides a much broader context than the partial approach in AI research briefly evocated in [2.4]… the way towards artificial mind promises to be long and difficult!

Moreover, social interaction and linguistic communication play a major role in the wiring of the secondary repertoire; this comes as a potential grounding for social constructivism (such as described in [Vygotsky, 1978]) and will be re-asserted in the next section dedicated to languages.

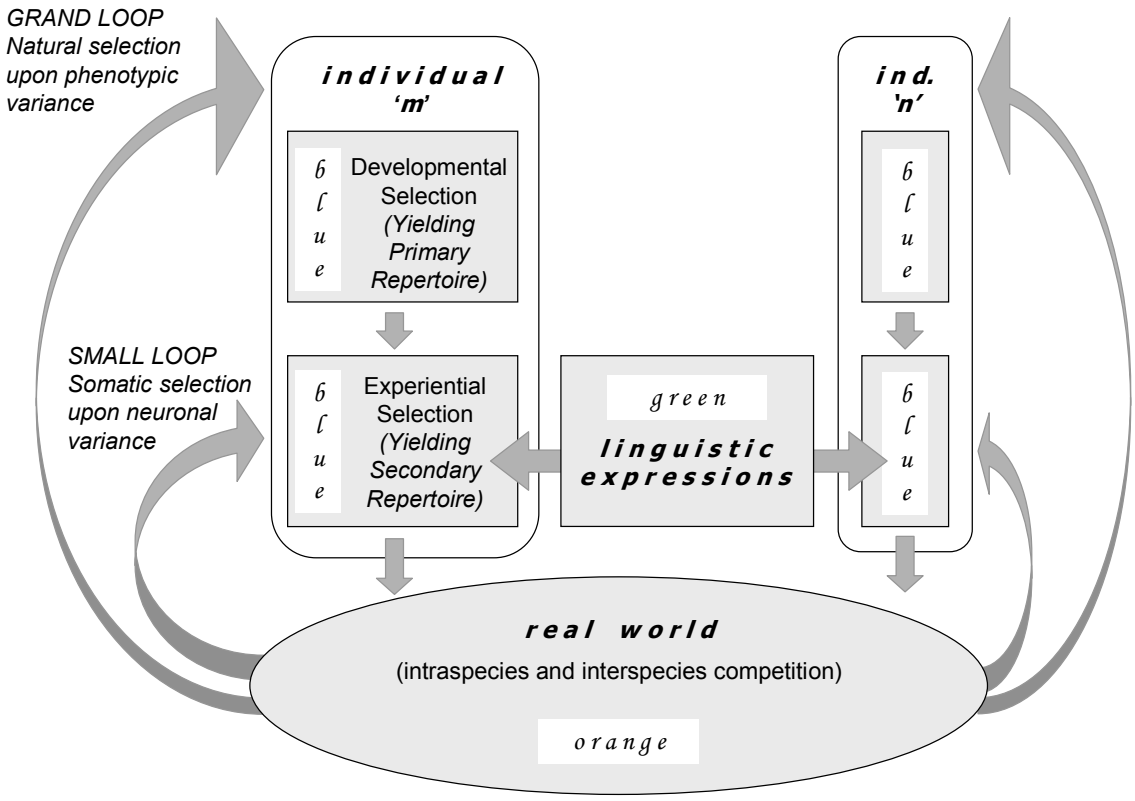


Figure 6: two evolution loops

We propose to summarize our exploration of the BLUE world under an informal definition:

informal definition:

Selectionist thinking is what happens in our associative and adaptive individual memory, in a process of continual re-categorization linking perception to action.

The unlimited 'mind binding power' relies on an immensely complex and constantly self-reconfiguring connected graph of neurons. The process of continual re-categorization is sustained by global mappings experientially selected by our value systems, where each perception/action leaves traces by means of change of strength in populations of synapses. Internal associations happen by mutual influence between these global mappings. *Selectionist thinking* is grounded on the following potentialities:

- selective grouping in aggregates on the basis of co occurrence of events;
- selective association of aggregates on the basis of similarity;
- selective association of aggregates on the basis of observed causality.

2.1.2 Green, the "world of linguistic expressions"

According to Edelman, the BLUE world of "individual mind" and the GREEN world of "stable social patterns and languages" articulate in two complementary ways:

- speech, enabling reference to inner states and objects or events by symbols, allows a whole new world of intentionality, categorization, and discrimination to be experienced and remembered and therefore drastically enhances the flourishing of concepts and thinking;
- individual associations in selective reentrant networks actually process "information" through adaptive learning based on imitation or convention communicated among a group.

"We propose that primary consciousness emerged in evolution when, through the appearance of new circuits mediating reentry, posterior areas of the brain that are involved in perceptual categorization were dynamically linked to anterior areas that are responsible for a value-based memory. With such means in place, an animal would be able to build a 'remembered present': a scene that adaptively links immediate or imagined contingencies to that animal's previous history of value-driven behaviour…

A key step in the evolution of higher-order consciousness was the development of specific kind of reentrant connectivity between the brain systems for language and the existing conceptual regions of the brain. The emergence of these neuronal connections and the appearance of speech allowed reference to inner states and objects or events by symbols. The acquisition of a growing lexicon of such symbols through social interactions, probably initially based on the nurturing and emotive relationships between mother and child, allowed for the discrimination of a self within each individual consciousness. When narrative capabilities emerged and affected linguistic and conceptual memory, higher-order consciousness could foster the development of concepts of the past and future related to that self and to others… A whole new world of intentionality, categorization, and discrimination can be experienced and remembered. As a result, concepts and thinking flourish. [Edelman & Tononi, 2000]"

This can in put in parallel with [Searle, 1983]: "The child develops the capacity to think and to speak hand in hand. How? The child starts with simple pre-linguistic intentionality. It then learns a simple vocabulary that enables it to have a richer *intentionality*[20], which in turn enables a richer vocabulary, and so on up in a boot-strapping process.

[20] [Searle, 1999]: "… the primary evolutionary role of the mind is to relate us in certain ways to the environment, and especially to other people. My subjective states relate me to the rest of the world, and the general name of that relationship is *intentionality*. These subjective states include beliefs and desires, intentions and perceptions, as well as loves and hates, dears and hopes … The requirement that all intentionality be in the heads of individual agents […]

The green world consists of patterns and norms aimed at externalizing, stabilizing and sharing the impermanent mappings occurring in the blue world; it plays the decisive role of feeding the activity of the blue world with stable concepts and making 'reasoning' sharable with others, altogether bootstrapping and enhancing the thinking.

In [Charaudeau, 1992], we read: "Le langage est ce matériau qui permet à l'homme de construire du sens dans le monde tout en entrant en communication avec les autres. Le langage est donc à la fois sens, expression et communication. Il n'est pas l'un et l'autre successivement, mais les trois à la fois."

In our coarse-grained analysis we address the wide spectrum ranging from natural languages of daily life to formal languages of mathematics and informatics: from "sentences interwoven in conversational processes", to "well-formed formula subject to logical inference"; therefore "linguistic expression" has to be understood as "expression in a language", be it natural or artificial.

In [2.1.2.1] we briefly describe how an *institutional reality* emerges from shared experiences among a group, then identify the "minimal complete unit of human linguistic communication" in the vision of Austin and Searle i.e. the *speech act* and look for some principles about how speech acts intertwine in *conversations*;

In [2.1.2.2] we start from the proposition of [McCarthy, 1998] of programs that would interact with humans through *speech acts*, and question how *conversations* involving non-intentional artefacts could be *meaningful for people*;

In [2.1.2.3] we come to the formal aspects, briefly explore the power and limits conveyed by a rigorous syntax and introduce some models of computation supporting *logical deduction*.

2.1.2.1 Meaning built upon social interaction: speech acts and conversations

The American philosopher John R. Searle proposes in [Searle, 1999] a story telling how the interpretation of repeated *collective* experiences by *individuals* opens the way for stable social patterns and bears the emergence of an *institutional reality* which establishes its foundations in the use of *language*; he then comes to what his own teacher called the "minimal complete unit of human linguistic communication": the notion of **speech act**.

---

does not require that all intentionality be expressed in the first-person singular. There is nothing to prevent us from having in our individual heads intentionality of the form, for example, "we believe", "we intend", and so on…."

"…Imagine a group of primitive creatures more or less like ourselves … acting as a group, they build a barrier, a wall around the place where they live. The wall is designed to keep intruders out and keep members of the group in … a function has been assigned to the wall - the function of acting as a boundary barrier – by the inhabitants acting *collectively*… Let us suppose now that the wall gradually decays. It slowly deteriorates until all that is left is a line of stones. But let us suppose that the inhabitants continue to treat the line of stones just as if they understood that it was not to be crossed … the wall now performs its function not in virtue of its physical structure but in virtue of the collective acceptance or recognition by the individuals acting collectively that the wall has a certain status and with that status goes a certain function … the move from physics to the collective acceptance of a "status function" forms the basic conceptual structure behind *institutional reality*.

… I believe that *language* is the fundamental human institution in the sense that other institutions, such as money, government, private property, marriage and games, require language, or at least language-like forms of symbolism".

At the most elementary level of description, a language basically consists of a collection of signs or sounds obeying to some assembly rules. To the question "in which way can signs or sounds exchanged between *humans* support *meaning*?" Edelman has proposed a biological answer in terms of local associations in reentrant networks. When Searle addresses the question of semantics[21] he follows the path initialized by his teacher J. L. Austin and reformulates the question of meaning in term of *speech acts* which are in first place *illocutionary acts*:

"Perhaps the simplest way to call attention to the astounding character of language is to remind ourselves of the following fact: in the lower portion of your face and mine is a cavity that opens by the way of a hinged flap. Periodically, that cavity opens and varieties of noises come out. For the most part, these noises are caused by the passage of air over mucous-covered cords in the larynx. From a purely physical point of view, the acoustic blasts produced by these physical and physiological phenomena are fairly trivial. However, they have remarkable features…Whenever I emit one of these acoustic blasts in a normal speech situation, I can be said to have performed a *speech act*. By mean of these acoustic blasts, I make a statement or ask a question, I give an order or make a request, or I explain some scientific problem or predict an event in the future. All of these, and dozens of other examples like them, where baptized by the British philosopher J. L. Austin as *illocutionary acts*. The

[21] "Now the problem of meaning in its most general form is the problem of how we do get from physics to semantics; that is to say, how do we get (for example) from the sounds that come out of my mouth to the illocutionary act?" [Searle, 1983]

illocutionary act is the minimal complete unit of human linguistic communication… it is the unit of *meaning* in communication[22] [Searle, 1999]."

Figure 7 illustrates the speech acts emerging from acoustic blasts as stable social patterns:

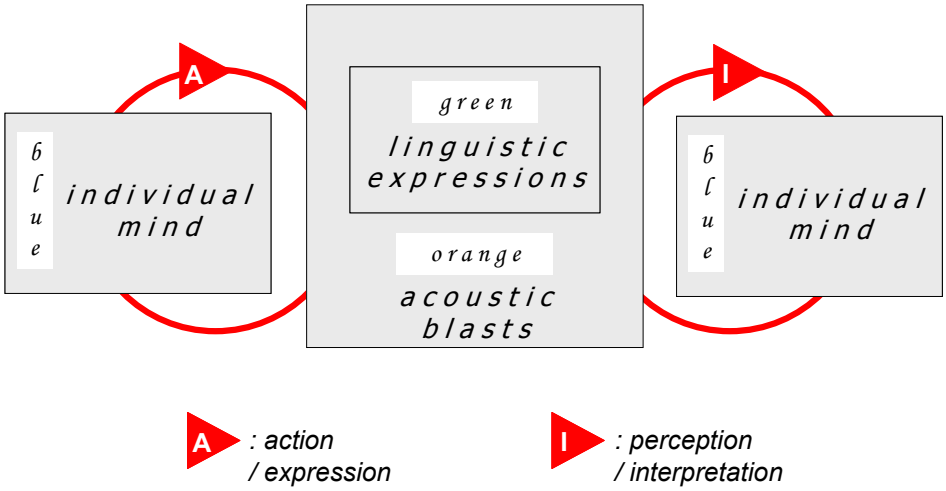


Figure 7: speech acts

Basing his analyses on [Austin, 1962] [Austin, 1979], Searle enumerates exactly five types of illocutionary acts:

1. the *assertive*. "The point of assertive speech acts is to commit the hearer to the truth of the proposition. It is to present the proposition as representing state of affairs in the world"…
2. the *directive*. "The illocutionary point of directive is to try to get the hearer to behave in such a way as to make his behavior match the propositional content of the directive. Examples of directives are orders, commands, and requests … Directives such as commands and requests cannot be true or false, but they can be obeyed, disobeyed, complied with, granted, denied, and so no."
3. the *commissive*. "Every commissive is a commitment by the speaker to undertake the course of action represented in the propositional content. Examples of commissive are promises, vows, pledges, contracts, and guarantees… Promises and vows, like orders and commands, cannot be true or false, but they can be carried out, kept, or broken."

[22] … while the perlocutionary act has to do with further consequences or effects of our actions.

4. the *expressive*. “Examples of expressives are apologies, thanks, congratulations, welcomes, and condolences.”
5. the *declarations*. “In a declaration, the illocutionary point is to bring about a change in the world by representing it as having been changed. The favorite examples are utterances like ”War is hereby declared,” ”You are fired”, “I pronounce you man and wife,”

Such speech acts “naturally” intertwine in the **conversations** of our daily lives; “naturally” does not mean in the absence of rules: “Speaking a language is engaging in a (highly complex) rule-governed form of behavior. To learn and master a language is (*inter alia*) to learn and to have mastered these rules. [Searle, 1969]”. The intertwining of speech acts in social exchanges was even considered in the classics as a “liberal art”[23]. Such an art is analyzed in [Charaudeau, 1992] where a robust and detailed set of rules taking into account individual intentions, goals in communication as well as the effects of the discourse is assembled in the shape of a “grammar of meanings and expressions”.

However our intention here is not to go through the very abundant matter that linguists have produced in the last forty years. Having in mind the question: “can we imagine conversational patterns for machines?” we need not explore all the fine-grained rules expressing the subtle nuances of our internationalities. Instead, we are going to look for a small number of general principles by which conversations are ruled.

This difficult point would probably deserve a thesis by itself; by chance we had the opportunity of a close collaboration with people able to give some light to it. We briefly summarize the three main points of a deep theoretical work undertaken under the direction of J. Breuker in [Breuker et al., 2005] which comes as a synthesis for [Breuker & Greef, 1993] [Clark, 1996] [Grice, 1975] [Sacks et al., 1974]:

- *Conversation* is a form of collaboration in which the discourse produced is the common product. Such a discourse is structured in topics; conversation is constructive and more than a simple exchange of information. As a special form of collaboration, conversation consists of its three necessary components: a *negotiated distribution of tasks*, *control* and *information transfer* according to dependencies between tasks. Control in collaboration is for instance required for indicating to

[23] In the classics, the education of the elite consisted in seven “arts appropriate for the free men”, or “liberal arts”; divided in the “trivium” (grammar, dialectic or logic, rhetoric) and the “quadrivium” (arithmetic = number per se, music = number in time, geometry = number in space, astronomy = number in space and time). The trivium consisted in: i) rhetoric which was concerned with persuasion in the context of assemblies ii) dialectics which were an exchange of propositions aimed at resolving a disagreement through rational discussion iii) grammar which was the study of rules governing the use of language.

agents involved in the task distribution which subtasks they have to perform when; within conversation, the most important *control* issue is in introducing *topics*.

- conversation mainly consists in two processes associated with two roles: the speaker role in charge of "natural language expression" and the hearer role in charge of "natural language interpretation"; both processes rely on common resources (lexical, syntactic and semantic knowledge). The *turn taking* (change of role) is another major control issue. It can be modelled by protocols … yet it has been observed without big surprise that humans never strictly comply to any of the possible protocols.
- two major resources are used in conversation: *common ground* and *models of one another*. *Common ground* is a rather heterogeneous collection of shared (mutual) knowledge and information about the current and past states of the environment, as well as the topic structure; assumed common ground allows the speaker to focus on what is new. *Models of one another* give insight about what can be meaningful for others; they are dynamically adjusted during the conversation according to observed reactions of participants to the others' speech acts.

2.1.2.2 About conversations involving non-intentional artefacts

The idea of programs aimed at interacting with people or other programs through *speech acts* was first proposed by John McCarthy[24] in 1989. He wrote in 1989 an article which would be published 9 years later, about a first set of specifications for programs aimed at interacting with people or other programs through speech acts: "Elephant 2000" [McCarthy, 1998]. We give the whole abstract below, as it can be found at:

http://www-formal.stanford.edu/jmc/elephant/elephant.html

"I meant what I said, and I said what I meant. An elephant's faithful, one hundred percent! Moreover, an elephant never forgets!

Abstract:

Elephant 2000 is a proposed programming language good for writing and verifying programs that interact with people (e.g. transaction processing) or interact with programs belonging to other organizations (e.g. electronic data interchange)

[24] John McCarthy is among the most famous computer scientists; he was responsible for the coining of the term "Artificial Intelligence" in1955, invented the LISP programming language in 1960 and received the Turing Award in 1971.

1. Communication inputs and outputs are in an I-O language whose sentences are meaningful speech acts identified in the language as questions, answers, offers, acceptances, declinations, requests, permissions and promises.
2. The correctness of programs is partly defined in terms of proper performance of the speech acts. Answers should be truthful and responsive, and promises should be kept. Sentences of logic expressing these forms of correctness can be generated automatically from the form of the program.
3. Elephant source programs may not need data structures, because they can refer directly to the past. Thus a program can say that an airline passenger has a reservation if he has made one and hasn't cancelled it.
4. Elephant programs themselves can be represented as sentences of logic. Their extensional properties follow from this representation without an intervening theory of programming or anything like Hoare axioms.
5. Elephant programs that interact non-trivially with the outside world can have both input-output specifications, relating the programs inputs and outputs, and accomplishment specifications concerning what the program accomplishes in the world. These concepts are respectively generalizations of the philosophers' illocutionary and perlocutionary speech acts.
6. Programs that engage in commercial transactions assume obligations on behalf of their owners in exchange for obligations assumed by other entities. It may be part of the specifications of Elephant 2000 programs that these obligations are exchanged as intended, and this too can be expressed by a logical sentence.
7. Human speech acts involve intelligence. Elephant 2000 is on the borderline of AI, but the article emphasizes the Elephant usages that do not require AI."

To our knowledge, this first draft of specifications has not yet given birth to a real programming language fulfilling all of them, although it has probably been a source of inspiration for many. With McCarthy's propositions in mind, let us illustrate a conversation involving *non-intentional artefacts* by the following imaginary case:

the case: In order to increase human satisfaction, a start-up called "Convivial-Coffee" proposes a new kind of coffee machines: at first look, they have the usual aspect and truly remain stupid machines … but they communicate with humans through speech acts, and in case one of them runs out of coffee, it will automatically transfer the request to another one, through a machine-to-machine conversation aimed at fulfilling the initial request.

Our question becomes: which "speech acts" are needed for this conversation? What about the turn taking, the models of one another, the topic structure?

- the case starts when a human desiring a cup of coffee meets a *commissive* written on a coffee-machine: "I shall give coffee if you insert one euro". We consider that such commitments have the status of promises, therefore "cannot be true or false, but they can be carried out, kept, or broken." In a computational model where promises are never broken (as pointed out in Elephant2000 abstract/2.), the commissives become a useful way to "transfer information about future information to be transferred".
- the case follows up with a choice based on *assertives* usually expressed by lit indicators: "I have coffee", "I have tea", "I have no more sugar" … In the desired computational model, assertives directly support information transfer (as pointed out in Elephant2000 abstract/3.).
- what about *expressives*? We may imagine the coffee machine emitting: "I am so terribly sorry … I have no more sugar!", which might look like an *expressive*, but is not: such a "speech act" coming from a non-intentional artefact would have no real better effect compared to the more abrupt: "no more sugar!". People wanting coffee usually do not care of internal states of machines … except if suddenly all lights are off and it smells of burnt electronics … the point of "program alert given through a *expressive*" will be addressed in Chapter 4.
- the word "declarative" is quite often used in Computer Science, and the affectation of values to variables exactly "brings about a change in the world by representing it as having been changed"! In fact, any illocutionary act of a formal language may turn out to become some *declaration* at a programming level … but the world which is changed is the electronic world of the machine, not the world about which the conversation is[25]. Therefore, the only declarations we shall consider will concern the dialog between machine and client. Whereas a declaration of the type: "I declare that you have drunk your coffee!" would look a bit strange, a declaration like "Your coffee is ready!" associated to the corresponding change in the "state of the world" should be appreciated… the point of "program termination expressed through a declaration" will be addressed in Chapter 4.
- to end with, the necessity of *directives* in a language for interactions has been emphasized in [Cerri et al, 2000] as an echo to "Elephant 2000": "When some

[25] Even in languages like SCHEME, the 'define' expression which is a declaration (of a new function) consists in fact in affecting a value to a variable standing for the "first order object called function".

information is not known, a transaction is started aiming at winning the value corresponding to the expression denoting the information. The aphorism is: when in doubt, do not compute, ask (and wait: do something else …). Someone will resume your suspended evaluation, sometimes providing you with what you need in order to go on". In fact, *directives* are exactly what coffee machines from "Convivial-Coffee" need to master for "orders, commands, and requests"!

If we concentrate on machine-to-machine conversations applying the principles of "Elephant 2000", we expect them to be mostly based on *commissives*, *assertives* and *directives* respectively corresponding to reliable promises, information transfer and orders or requests. We are going to exemplify this in more detail, since it will play an important role in our calculus in Chapter 3.

Let us imagine for instance a shortage of sugar and the subsequent conversation between 'H' wanting a cup of coffee and coffee-machines 'A', 'B', and 'C':

- machines A, B, C/ *commissive*: "I shall give coffee if you insert one euro"
- human H/ *directive*/ to machine A: "give me coffee with sugar"
- machine A/ *directive*/ to machine A: "have I got coffee?"
- machine A/ *directive*/ to machine A: "have I got sugar?"
- machine A/ *assertive* / to machine A: "I have coffee"
- machine A/ *assertive* / to machine A: "I have no sugar"
- machine A/ *directive*/ to machines B and C: "have you got coffee?"
- machine A/ *directive*/ to machines B and C: "have you got sugar?"
- machine B/ *assertive*/ to machine A: "I have coffee"
- machine B/ *assertive*/ to machine A: "I have no sugar"
- machine C/ *assertive*/ to machine A: "I have coffee"
- machine C/ *assertive*/ to machine A: "I have sugar"
- machine A/ *directive*/ to machine C: "give H coffee with sugar"
- machine A/ *assertive*/ human H: "machine C will give you coffee with sugar"
- …
- machine C/ *declaration*/ human H: "Your coffee is ready!"

Now can this really work, what are the conversational patterns hard-wired in the non-intentional machines?

… we list hereafter is the main difficulties to be solved:

1. in order to fulfil the initial requirement, A addresses a question to all other machines. It might happen that other machines are engaged in other conversations; it might even happen that some other machines cannot answer directly but must "introspect[26]" through intermediary questions … at which moment can A *take the turn* again, and "consider[26]" that the question has (positively or not) been answered?
2. how does a machine "know" which questions can be understood and answered by the others: does each machine necessarily store *models of other machines* or is there another way where machines can remain independent of one another?
3. how can machines A, B, C adapt to all possible combinations of human desires crossed with presence/absence of ingredients and dynamically introduce the adequate *topics*?
4. how can we guarantee the fact that the conversation will end … before H angrily phones to "Convivial-Coffee" engineers?
5. how can we guarantee an issue of the conversation which does not depend on the order in which answers from B and C have reached A?

Before attempting an answer to McCarthy's requirements, and fully exposing the secret of "Convivial-Coffee" engineers (this will be done in Chapter 3), we would like to put the emphasis on one point:

In this toy-example of conversation between machines, the nature of the algorithms in charge of computing *assertives* and *directives* has not yet been evoked; what we emphasize is that they operate on expressions which are *meaningful* for humans i.e. they use a high level language, exactly as initial *commissives* or final *assertives* … this may seem a detail for 'H' who is only interested in his cup of coffee, but it is extremely important for "Convivial-Coffee" engineers who have designed and tested the machines, separately at first, and then as a (convivial) group of artefacts. These engineers have solved the "give-me-coffee-with-sugar" problem in a compositional way, therefore formalizing fragments of some "theory supporting the delivery of coffee" within multi-steps algorithms, in such a way that these fragments can auto-organize; we emphasize that such a compositionality must be reflected in the computation model.

[26] We use quotes not to forget that A, B and C are non intentional artefacts!

2.1.2.3 Logical interaction between artefacts

While in the last sub-section we examined language as a means for social interaction, we now come to the formal coding aspects, so that conversations can be computed by machines, on the only basis of syntax[27].

In the "pure GREEN" world of syntax, "linguistic expressions" are virtual entities existing independently of emitter's or receiver's mind, independently of physical support, and obeying ultimately achieved stable patterns, therefore computable by machines. Whereas individual interpretation is subject to continuous change, whereas events endlessly pop up and disappear … formal systems are, like diamonds, eternal!

In this sub-section:

- we first of all give a brief introduction to formal languages and examine them in the light of two questions illustrating the power and limits of logical thinking;
- we then rapidly present Turing machines and evoke other equivalent 'computational models' or 'abstract machines'[7];
- we end by explaining which role 'logical deduction' can play in our framework for collaborative theory building.

A brief introduction to formal languages

The theory of formal languages plays a crucial role in today's informatics by establishing a connexion between Mathematics and several applications in Computer Science such as "the computability theory, the theory of automata, the compiler design theory …"

Various definitions exits; however they all have a few features in common, which will be sufficient for taking further steps towards the informal notions of 'logical deduction'. The definition of a formal language 'L' always go through three steps:

1. a finite *alphabet* Σ is given;
2. the infinite set of all possible concatenations built upon Σ is called *expressions* and denoted Σ*;

[27] Inside contemporary grammar, syntax is the study of "patterned relations" that govern the way the words in a sentence come together, independently of their meaning.

[7] An abstract machine, also called an abstract computer, is a theoretical model of a computer hardware or software system used in Automata theory. Abstraction of computing processes is used in both the computer science and computer engineering disciplines and usually assumes a discrete time paradigm.

3. a subset 'L' of Σ*, called 'formal language' because its expressions are formally constrained by a *finite set of rules*, is defined. These rules can be *production rules*: in such a case, the expressions of 'L' are obtained by 'derivations' consisting in a finite number of applications of rules upon a finite set of initial expressions. Or they can be *recognition rules*: in such a case, the expressions of 'L' are those resulting from a valid 'parsing' through a finite number of tests.

There is a great variety of ways for specifying formal languages:

- in *generative grammars*, expressions are built through successive rewritings. Formal grammars for instance are based on a finite set of rules modifying one part of a word at a time; they give birth to *recursively enumerable languages*. In linguistics and computer science, a *context-free grammar* (CFG) is a formal grammar in which every production rule is of the form "NT → w", where NT is a non-terminal symbol and w is a string consisting of terminals and/or non-terminals; *context-free grammars* allow syntax checking through parsing algorithms and are powerful enough to describe the syntax of many programming languages.
- it may happen also that the strings of the formal language are produced by a *regular expression,* i.e. a string matching a set of strings according to a set of rules;
- in *analytic grammars*, expressions are 'parsed' through a number of tests before being accepted or rejected;
- some languages are recognized by a given *automaton,* i.e. a self-operating machine, simpler of equivalent to a Turing Machine[28].
- others are defined as positive results of some *decision problem*, i.e. a question with a Yes/No answer.

Two major theoretical questions, or 'decision problems', occur about formal languages:

a) Syntax checking: is a given succession of expressions a valid 'derivation'/ 'parsing'?

b) Theorem proving / Recognition: is it possible to find a valid 'derivation'/ 'parsing' for a given expression?

[28] Turing Machines are presented further in this sub-section

Due to the finite grounding of formal systems, the 'syntax checking problem' is always decidable (it can be given an answer in finite time through a deterministic procedure).

On the contrary, since the number of times each rule can be conveyed in a 'derivation'/ 'parsing' are not bounded, the number of possible expressions is not bounded. The consequence is that the 'Theorem proving/ Recognition' problem is generally not decidable.

[8.2] gives a first hint about the power and limits of formal languages. On one hand a simple 'Post System' can serve as a foundation for the Propositional Calculus; on the other hand the "Post correspondence problem", which questions the possible derivations inside a Post System, is not decidable.

As exposed in [Alliot & Schiex, 1993], Hilbert proposed in 1908 to provide formal grounding for Mathematics in order to prove their consistency (impossibility to formally derive a expression AND its negation within the system). His students started to work on this program in 1920, focusing their efforts on Arithmetic … until Gödel and then Kleene proved the consistency of Arithmetic to be improvable! Moreover, Arithmetic is undecidable (there is no decision procedure for 'Theorem proving / Recognition') and uncomplete (there are formulas such that neither the formula nor its negation can be 'proved / recognized')

It seems that the more expressive a formal language is, the more difficult it is to reason upon it, in particular when some kind of auto-reference can be simulated inside the language.

At a minimum, the question: "is this sequence of expressions a correct application of the syntactic rules of the formal language?" will always get an answer; this is what we are intending to build upon.

Turing Machines

When formal languages are not explicitly generated, they have to be recognized by some formally defined 'self-operating' machine, i.e. machines obeying to a "computation model". Among several equivalent computation models, Turing Machines are the most famous.

Turing machines were born in 1936, at a time when Church, Gödel, Kleene, Post and Turing were working on the notion of formal systems and calculi; Alonzo Church, Stephen Cole Kleene and Kurt Gödel were looking for a universal model for intuitively

calculable functions, while Emil Post and Alan Turing took the view point of a universal model for effective computation by machines. One of the major conclusions of this community of mathematicians, with the status of a very widely admitted conjecture, is known as the Church-Turing thesis: "For a function, to be intuitively calculable or to be Turing calculable are equivalent". Another formulation, better suited to our purpose is the following: "any possible calculation can be performed by an algorithm[29] running on a computer, provided that sufficient time and storage space are available". The functions from natural numbers to natural numbers which can be computed by Turing machines are called *recursive functions*.

http://en.wikipedia.org/wiki/Turing_machine gives an informal definition of Turing Machines:

"A Turing machine consists of:

1. A tape which is divided into cells, one next to the other. Each cell contains a symbol from some finite alphabet. The alphabet contains a special blank symbol (here written as '0') and one or more other symbols. The tape is assumed to be arbitrarily extensible to the left and to the right, i.e., the Turing machine is always supplied with as much tape as it needs for its computation. Cells that have not been written to before are assumed to be filled with the blank symbol.

2. A head that can read and write symbols on the tape and move left and right one (and only one) step at a time.

3. A state register that stores the state of the Turing machine. The number of different states is always finite and there is one special start state with which the state register is initialized.

4. An action table (or transition function) that tells the machine what symbol to write, how to move the head ('L' for one step left, and 'R' for one step right) and what its new state will be, given the symbol it has just read on the tape and the state it is currently in. If there is no entry in the table for the current combination of symbol and state then the machine will halt."

This very simple model of computation was not aimed at building effective computers; but at giving an insight about what computers can do… or not … In [Turing, 1937] it was

[29] [Knuth, 1973]: "It is generally assumed that an algorithm must satisfy the following five requirements:
- Finiteness: An algorithm must always terminate after a finite number of steps … a useful algorithm should require not only a finite number of steps, but a very finite number, a reasonable number
- Definiteness: Each step of an algorithm must be precisely defined; the actions to be carried out must be rigorously and unambiguously specified for each case
- Input: ...quantities which are given to it initially before the algorithm begins. This inputs are taken from specified sets of objects
- Output: i.e., quantities which have a specified relation to the inputs
- Effectiveness: ... all of the operations to be performed in the algorithm must be sufficiently basic that they can in principle be done exactly and in a finite length of time by a man using paper and pencil"

shown that the particular (Halting) problem of determining whether any particular Turing machine will halt on a particular input was not decidable. [Rice, 1953] even demonstrates that no machine can always correctly decide whether the language of a given Turing machine has a particular 'nontrivial property' (such as halting on a particular input).

It is considered[30] that what we call 'computers' behave like "universal Turing Machines"; they are able to calculate any recursive function, decide any recursive language, and *accept* any *recursively enumerable* language (but not *recognize* it because of the Halting problem).

Among computation models equivalent to Turing machines, we find:

- "Formulation 1" in [Post, 1936]
- [Minsky, 1967]'s computational model with only two instructions i) add 1 to the number in register, and go to next instruction; ii) if number "n" in register is not zero then subtract 1 from it and go to the next instruction, otherwise go to the n^{th} instruction
- Normal Markov Algorithm
- … and some others, like the "Kolmogorov-Uspensky machines" [31] relying on what [Gurevich, 1988] calls "real-time algorithms": "A real-time algorithm inputs a symbol, then outputs a symbol within a fixed number 'c' of steps, then inputs a symbol, then outputs a symbol within 'c' steps, and so on. In the case of more powerful machines, one may want to speak about more complicated data items (than just symbols) as one-step inputs or outputs."

In order to sustain the conversation between the machines of "Convivial Coffee", we need such algorithms; however we shall use the denomination "multi-steps algorithms" in place of "real-time algorithms" in order to avoid the confusion with software systems having operational deadlines from event to response.

[30] None of them is equipped with an « infinite tape », but we can always build a storage capacity large enough for any computation.

[31] Among other computation models, we read in [Durand, 2004] that Kolmogorov proposed under the name of "Kolmogorov-Uspensky machines" a computation model in which all others could integrate. These machines are described in [Kolmogorov, 1963] and in [Uspensky, 1990]. The computation model relies on a tape being a finite directed connected graph able to change its topology by addition or removal of edges and nodes. We have not entered in the details of these machines, but will also need an evolving directed graph as a support for control of conversation in the calculus presented in Chapter 3.

Towards a 'logically ruled' interaction

Starting from the definition of [Magnani, 2001]: "Deduction is an inference that refers to a logical implication. Deduction may be distinguished from abduction and induction[32] on the grounds that only in deduction is the truth of the conclusion of the inference guaranteed by the truth of the premises on which it is based. Deduction refers to the so-called non-defeasible arguments.", we are going to use formal languages for an informal definition of 'logical deduction' is the context of collaborative theory building .

What Magnani emphasizes in 'deduction' is the succession of arguments which guarantees a correct "transmutation" of true expressions into other true expressions. The question of the truth or falsity of expressions itself involves experimental facts and minds able to interpret them, and will be addressed in [2.1.3]. What is in question here is the correctness of the process: either the 'derivation' process linking facts and hypotheses to new (expected) facts or the 'recognition' process analyzing composed expressions.

The correctness of the process is exactly what formal languages are about; the additional property we shall require from an abstract machine aimed at composing 'partial theories' is based on 'confluence', hereafter defined in http://en.wikipedia.org/wiki/Confluence :

"We can think of a rewriting system abstractly as being a set S, together with a relation R which is a subset of $S \times S$. Instead of denoting relations as pairs (s, t), with $s, t \in S$, we write $s \rightarrow_i t$, where i is from some index set I. If $u \in S$, and $u \rightarrow_j v$ for some $j \in I$, then we say v is a *reduct* of u (alternatively v is an *expansion* of u, or u is *reduced to* v). We can also think of a chain of reductions of some element: if $w \in S$, then we may have

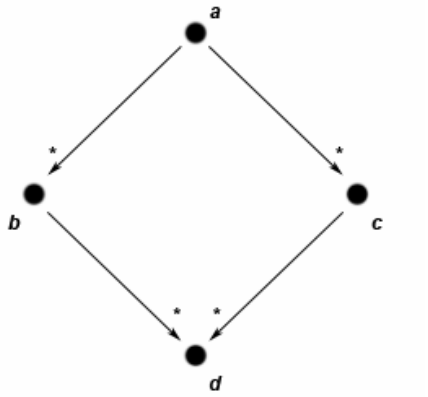
$w \rightarrow_{i1} w_1 \rightarrow_{i2} w_2 \rightarrow_{i3} w_3$

where $i_k \in I$. We call this chain a *reduction sequence*, or just a *reduction of* w. If the reduction sequence terminates, say in w_n, we write

$w \rightarrow_I^* w_n$

reflecting that relations in I were chosen. If I is clear from context then we can omit this subscript.

With this established, confluence can be defined as follows. Let $a, b, c \in S$, with $a \rightarrow^* b$ and $a \rightarrow^* c$. If a is confluent, there exists a $d \in S$ with $b \rightarrow^* d$ and $c \rightarrow^* d$. If every $a \in S$ is confluent, we say that $\rightarrow$ is confluent, or has the *Church-Rosser property*. This property is also called the *diamond property*, after the shape of the diagram shown on the right."



[32] 'abduction' and 'induction' will be defined in [2.1.3.2]; in first approximation, 'abduction' relies to intuitive hypotheses while 'induction' goes from particular facts to generalizing hypotheses.

Let us consider an abstract machine in charge of automatically ruling the interaction between 'multi-steps algorithms'[33] embedding 'partial theories'[34]. A first hope would have been to have the machine answering questions of the type "is this expression a theorem?"; but according to the general case, theorem proving is not decidable. Instead, we may benefit from two aspects:

- each global derivation will lead to CORRECT EXPRESSIONS;
- if (and only if) the abstract machine is confluent, each global derivation will depend on only two things: i) the partial theories and ii) the interaction rules; it will therefore behave globally as a determinist logical artefact. In such a case, the result of the rewriting can be interpreted as THE LOGICAL[35] CONSEQUENCE WITHIN THE FORMAL SYSTEM of the interaction of partial theories.

We first suppose a formal language able to integrate multi-steps algorithms as local rewriting systems and to rule their interaction in a confluent way.

We propose to call 'logical consequence of the interaction within the formal system' the result of the global rewriting.

In order to illustrate the logical composition of partial theories, we must now formalize the notion of 'theory'; this is the subject of next sub-section.

[33] in the sense of [Knuth, 1973] extended by [Gurevich, 1988]
[34] to be defined in the next sub section
[35] We use the adjective 'logical' in contrast with the adjective 'selectionist' of the present sub-section.

2.1.3 A framework for collaborative theory building

Theories are about explanation (true sentences about past events) and prediction (true sentences about future events) of phenomena. The word 'theory' derives from the Greek 'theorein', which means 'to look at'; then what do we look at?

According to Searle's default positions[36], which we adopted at the beginning of this chapter, we look at events and transitions in the *real world*, i.e. the world of events and causality. But looking is not enough for building theories … we have to say something about what we observe: theories need facts and facts have to do with events and their linguistic expressions. Moreover, generalizing and establishing correlations upon facts necessitates the associative power of human mind.

The previous investigations about our associative and selective memory, as well as the empowerment offered by social interaction and formal languages can now be assembled in the framework and contribute to some logically ruled 'collaborative theory building'.

In [2.1.3.1] we start from Peirce's approach of theory building, then introduce the role of controversy among the group by referring to Popper and finally come to a semi-formal definition for 'collaborative theory building';

In [2.1.3.2] we put the focus on the 'theory discovery' and explore the 'selectionist thinking' processes by which we ascertain facts and emit hypotheses upon them;

In [2.1.3.3] we introduce 'computationally assisted deduction' in the cycle and show the impact on 'theory checking'.

2.1.3.1 Theory building: from Peirce to Popper

In this sub-section, we could not miss the reference to Charles Sanders Peirce; here is an extract from http://en.wikipedia.org/wiki/Charles_Peirce :

"Although educated as a chemist and employed as a scientist for 30 years, he is now mostly seen as a philosopher... An innovator in fields such as mathematics, research methodology, the philosophy of science, epistemology, and metaphysics, he considered himself a logician first and foremost. While he made major contributions to formal logic, "logic" for him encompassed much of what is now called

[36] Searle's 3 first default positions:
- There is a real world that exists independently of us, independently of our experiences, our thoughts, our language
- We have direct perceptual access to that world through our senses, especially touch and vision.
- Words in our language, words like rabbit or tree, typically have reasonably clear meanings. Because of their meanings, they can be used to refer to or talk about real objects of the world

the philosophy of science and epistemology. He, in turn, saw logic as a branch of semiotics, of which he is a founder. In 1886, he saw that logical operations could be carried out by electrical switching circuits, thus anticipating the digital computer […]

His approach concerning scientific reasoning is summarized hereafter:

- active process of theory generation, with no prior assurance of truth;
- subsequent application of the contingent theory, aimed toward developing its logical and practical consequences;
- evaluation of the provisional theory's utility for the anticipation of future experience, and that in dual senses of the word: prediction and control".

In [Peirce, 1958], three processes are identified which operate in a cyclic fashion:

i) *abduction* is what we use to generate a likely hypothesis or an initial diagnosis in response to some questioning situation;

ii) *deduction* is used to clarify, to derive, and to explicate the relevant consequences of the selected hypothesis;

iii) *induction* is used to test the sum of the predictions against the sum of the data.

In Figure 8, we start from Peirce's approach and further illustrate the distinction between "logical" and "practical" consequences by separating "logical implication in the green world of linguistic expressions" from "experimentation in the orange world of causality":

1. "likely hypotheses" are the fruit of some *abduction*; in the shape of some linguistic expressions;
2. we suppose that the corresponding language contains derivation rules allowing *deduction* to operate as a checkable derivation ; this leads to predicted results under the shape of new linguistic expressions;
3. in parallel, we experiment with observable *facts* (in the orange world, events are causality related; therefore resulting facts come as a causal consequence of initial facts);
4. *induction* upon predictions and observations leads to diagnostic and further abduction.

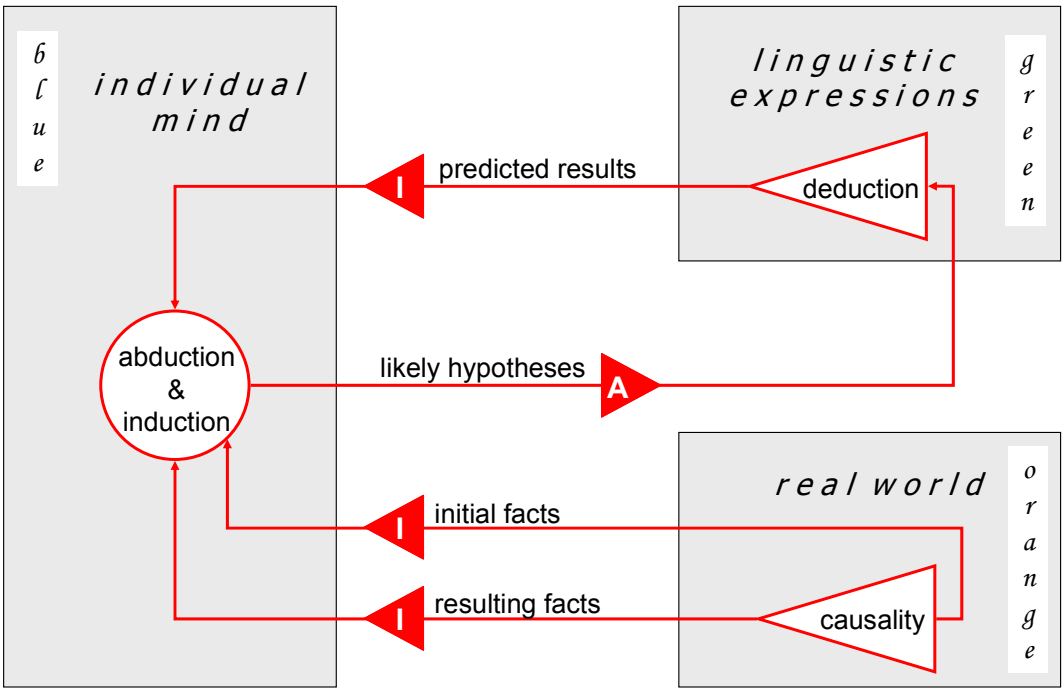


Figure 8: Peirce's approach of Theory building

Peirce's viewpoint is mostly the viewpoint of one scientist having some 'subject of enquiry'; i.e. that the truth or falsity of expressions has to be decided by a single mind by comparing predicted results with facts based on real events. The bottom of the picture illustrating the arbitration of the truth or falsity of expressions by induction/abduction within a single mind upon experiments in the real world is what [Shapiro, 1991] formalizes as an 'oracle' (see [8.3]); taking inspiration from him, we may adopt the following definition of a 'partial theory':

partial theory (semi-formal definition):
Let 'L' be a formal language based on 'expressions' describing facts, hypotheses and predictions, where predictions are constrained by a set of rewriting rules. Let us suppose a local 'oracle' testing 'expressions' and answering true or false on the basis of individual understanding. A *partial theory* is a set of true expressions such that their rewriting produces only true expressions with respect to the local oracle.

When Peirce does not explicitly address the collaboration, Karl R. Popper will later introduce the scientific community and its controversies as a major component in theory building: as emphasized in [Popper, 1959], each time a likely hypothesis is emitted by one scientist, the others are in charge of attempting to falsify or to refute it. Successive cycles of hypotheses and refutations will eventually lead to a temporary consensus called

truth. The scientific status theory is therefore directly related to its capacity to be ascertained or falsified through genuine tests.

Taking into account the collaboration, we may therefore adopt the following:

Theory (semi-formal definition):
Let 'L' be a formal language based on 'expressions' describing facts, hypotheses and predictions, where predictions are constrained by a set of rewriting rules. 'L' is built and shared by a group of people, each with their own understanding, who attempt to falsify partial theories emitted by the others and collectively constitute an 'oracle' deciding the truth or falsity of 'expressions'. A *theory* is a set of true expressions such that their rewriting produces only true expressions with respect to the collective oracle.

The 'collective oracle' is an abstraction for a set of 'local oracles', each associated to an individual mind. According to [2.1.1] such local oracles are based on 'selectionist thinking'; our purpose has therefore become to rule on a logical basis the interaction between selectionist thinking entities; it is interesting to take note at that point from a conjecture formulated by Edelman in the last and more philosophical part of [Edelman & Tononi, 2000]:

"After Homo sapiens and higher-order consciousness appeared, it became possible to create syntactically rich symbol systems, to create codes, and even to create logic… We have taken it as established that after the brain arose in evolution by natural selection, which set up value constraints and major structures, each individual brain operates by process of somatic selection. Instead of being guided by *effective procedures*, it is governed by *effective structures*, the dynamics of which allow its correlated activities to arise by selection, rather than by the rules of logic … Clearly, if the brain evolves in such a fashion, and this evolution provided the basis for the eventual discovery and refinement of logical systems in human culture, then we may conclude that, in the generative sense, evolution is more powerful than logic… In any case, the interesting conjecture is that there appear to be only two deeply fundamental ways of patterning thought: selectionism and logic."

Relying on Edelman's conjecture, we may therefore rephrase [Lukasiewicz, 1970][37] : "reasoning which starts from theories and looks for facts is patterned by logical deduction and can be entrusted to machines, that which starts from facts and looks for theories is patterned by selectionist thinking."

[37] [Lukasiewicz, 1970] "reasoning which starts from reasons and looks for consequences is called deduction; that which starts from consequences and looks for reasons is called reduction"

2.1.3.2 Theory discovery: the role of selectionist thinking

Looking at events of the real world seems in first place rather trivial … but is not! From the physicists we know that observing implies interacting with and therefore transforming what we observe. We also know from neurobiology and psychology that observing transforms the mind of the observer:

- in Edelman's model of global mapping[38], perception and action are tightly related through a huge degenerate set of evolving circuits which are the observer's mind,
- at a more psychological level, the mirroring between brain structures and *events* of the outside is addressed by Piaget in terms of evolving *cognitive structures*: "Cognitive structures change through the processes of adaptation: assimilation and accommodation. Assimilation involves the interpretation of *events* in terms of existing cognitive structure whereas accommodation refers to changing the cognitive structure to make sense of the environment [Piaget]".

That is why sharing facts is indeed a quite difficult undertaking in experimental sciences, as referred to in [Piaget, 1968]: "… it is much more difficult to ascertain facts and to analyze them than to think or to deduct, and here is why experimental sciences were born long after deductive disciplines, the latter playing the role of framework and necessary conditions for the former, but in no way being sufficient …" or earlier in [Popper, 1959]: "Science does not rest upon solid bedrock. The bold structure of its theories rises, as it were, above a swamp. It is like a building erected on piles. The piles are driven down from above into the swamp, but not down to any natural or "given" base; and if we stop driving the piles deeper, it is not because we have reached firm ground. We simply stop when we are satisfied that the piles are firm enough to carry the structure, at least for the time being."

When we look for informal definitions of the word *'fact'* at http://en.wikipedia.org/wiki/Fact , we read:

> "a fact is an honest observation."
>
> "something thought to be actual as opposed to invented, e.g.: in this story, the Gettysburg Address is a fact, but the rest is fiction."
>
> "something which has become real, e.g.: the promise of television became a fact in the 1920s."
>
> "something concrete used as a basis for further interpretation, e.g.: look at the facts of the case before deciding."

[38] see 2.1.1.4

"an objective consensus on a fundamental truth that has been agreed upon by a substantial number of people, e.g.: is no doubting the fact that the Earth orbits the Sun."

"information about a particular subject, e.g.: the facts about space travel."

Since the notion of *fact* cannot be disentangled neither from the observer, nor from the description language, a 'fact for us' is not necessarily a 'fact for them'; more precisely, having the same facts is both a condition and a consequence of sharing the same *institutional reality*[39].

At some level of description, our brain is a complex connected graph full of selected reentrant maps chained in evolving neural circuits that we call 'structures'; we propose to consider that level for the following statements:

- whenever the process of perceptual categorization associates widely overlapping structures to distinct events, such events are interpreted as similar events;
- linguistic expressions emerge from re-occurring response of individual maps to similar events; we can therefore consider the equivalence relation between events defined in the following way: events are equivalent if they are associated in the brain structures to the same linguistic expression e.g. "frog eating dragonfly";

- *facts* can therefore be understood as equivalence classes of *events* for this relation; this allows us to represent an *event* by a *linguistic expression* as illustrated in Figure 9. Facts can be accurate (with only one event in the equivalence class) like: "a frog is eating a dragonfly at position (x, y, z, t)" or generic like: "grass is green".

It is interesting to note that, although our starting viewpoint is of a single mind, the definition of Figure 9 allows us to consider a "shared fact" when some equivalent events are associated to the same linguistic expression by distinct minds. Besides, if the notion of "truth" stands for the degree of correspondence between a representation and what is being represented, the facts that we consider are *true facts*, with respect to those who observe the associated events and express them.

[39] as explained when we investigated about social patterns and language in 2.1.2.1

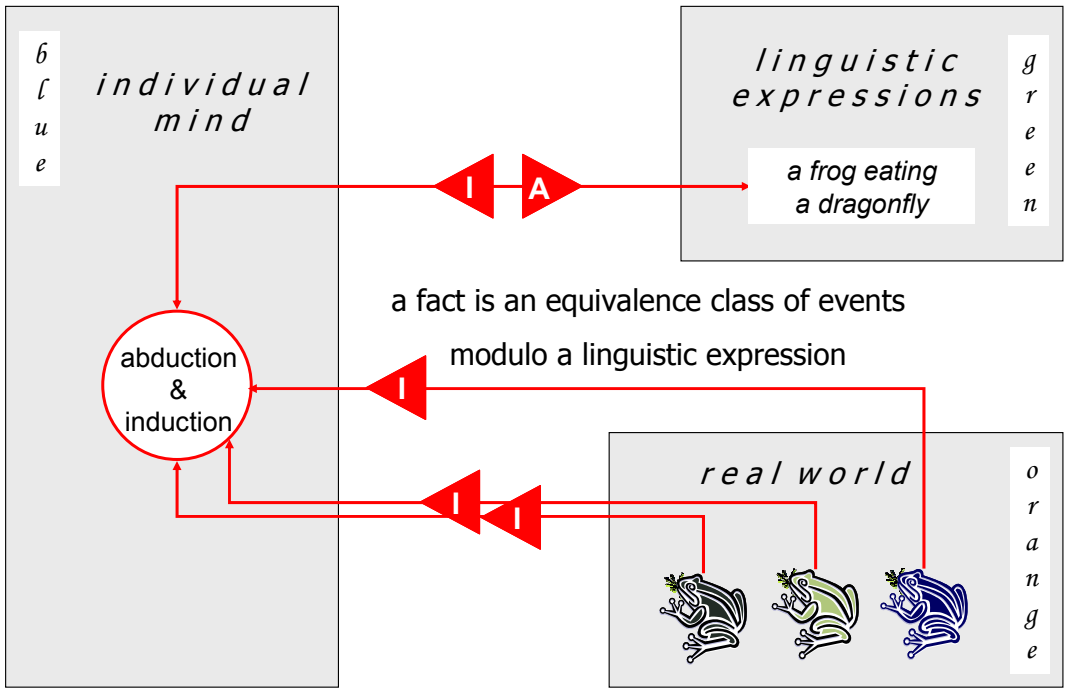


Figure 9: "a fact" in the framework

While *facts* are grounded on events, *hypotheses* are not! Hypotheses are higher level constructions of the mind grounded on the *linguistic expressions* representing facts. Except in the close context of a pre-defined conceptualization, hypotheses cannot be logically deduced by an algorithm, they have to be selectively thought by human minds.

This mainly happens according to two patterns called 'induction' and 'abduction'; we propose to briefly analyze them hereafter:

Selective induction of hypotheses

Let "John arrived first two years ago", "John arrived first last year", "John arrived first this year", be three different single-event-facts, then the expression "John always arrived first" is not an event, but the result of some generalization upon the previous facts; this is a creative process as early expressed in [Mill, 1843]: "Induction is an inferring process; it starts from the known towards the unknown, and any operation which does not imply an inference, any process in which what seems to be the conclusion does not go beyond the initial premises, should not be called induction".

This ability to go beyond the premises was later analyzed and extended to the validation of hypotheses in [Peirce, 1955]: "a common feature of all kinds of induction is the ability to

compare individual statements: using induction it is possible to synthesize individual statements in general laws – inductive generalizations – but it is also possible to confirm or to discount hypotheses".

Induction has its biological roots in what [Edelman & Tononi, 2000]" call 'concept formation': "Concept formation is the ability of the brain to combine different perceptual categorization related to a scene or an object and to construct a "universal" reflecting the abstraction of some common feature across a variety of such percepts". In order to avoid the confusion between true ('selectionist') induction and the mere application of formal rules in a close world, we proposed in [Lemoisson and Cerri, 2005] two patterns for *induction*:

- the inferring of a new concept (meaning an interesting collection of "features" which can be verified on a set of examples and which deserves a name "as a special aggregate") WITHOUT giving a closure to the set of all possible features;
- the inferring of a new hypothesis (meaning a well-formed proposition in a given language which can be verified in a set of situations) WITHOUT restricting the "background knowledge" with which the well-formed proposition would have to be consistent.

Selective abduction of hypotheses

Now let us suppose that the one who arrives first raises a cup. There are all kinds of occasions for raising a cup; under certain circumstances, it will make an experimented observer think that the person doing so is neither a priest celebrating a mass, nor a VIP at lunch, but the winner of some race, ALTHOUGH the observer was not able to watch the race. [Magnani, 2001] calls *selective abduction* the generic following pattern:

- the surprising fact, "A", is observed;
- but if "B" were true, "A" would be a matter of course;
- hence, there is reason to suspect that "B" is true!

The point here is not to give a theory for the phenomenon of abduction (we shall just emit a likely hypothesis); but to show that the very basic biological mechanisms described by Edelman are potentially sufficient for explaining it, and also to insists upon the necessary associative power and selection mechanisms which are required.

If we come back to our example, the following pattern can explain the reasoning relation between the fact 'A' "John raises a cup" and the emission of the likely hypothesis 'B' "John arrived first", called *abduction*:

i. past occurrence of equivalent events have led to overlapping structures in the mind of the observer associated to the expression of a generic fact 'A': "John raises a cup"
ii. through communication between the observer and some friends, some of these past events have been synchronized with other structures associated to the generic fact 'B' expressed by: "John arrived first"
iii. subsequent reinforced reentry between maps supporting linguistic expressions 'A' and 'B' has occurred as a consequence of this past synchronization
iv. a new event, firstly interpreted as a re-occurrence of fact 'A': "John raises a cup! " happens
v. this new event indirectly activates the expression 'B': "John arrived first"

Can selectionist thinking be operated by machines?

There have been several attempts for delegating the 'theory discovery' process to computers, some of them mimicking 'selectionist thinking'; but any codification within a formal system carries expectable limitations:

- we have previously evoked [Edelman and Reeke, 1982] and [Reeke and Edelman, 1984] for their work on artificial associative recall, and many artificial neural networks exist and have powerful applications in 'pattern recognition', but they remain highly specialized artefacts;
- another 'connectionist example' can be found in [Holland et al, 1986] where competing parallel computations are ruled by a selection based on "market laws" but it is explained in [Alai, 2004] that this has not produced any really new discovery (see [8.1]);
- there is also a whole field in AI relying on the use of "selection-based algorithms" but those are ruled by an artificially settled "utility function" in the context of a closed data space [Gruber, 1992] [Alliott, 1993].

As stated in [Langley, 2000], these promising approaches are still not powerful enough for addressing theory building; this is mostly due to their pre-defined and irreversible specialization, either in the perception process, or in the 'utility evaluation', and we may have to wait for some more decades before machines can pass the Turing test[40].

[40] The Turing test is a proposal for a test of a machine's capability to perform human-like conversation. Described by Alan Turing in the 1950 paper "Computing machinery and intelligence", it proceeds as follows: a human judge engages in a natural language conversation with two other parties, one a human and the other a machine; if the judge cannot

2.1.3.3 Intertwining selectionist thinking and logical deduction

On the other hand, computers can be very efficiently used for building powerful 'demonstrators' allowing the checking and refinement of the theory by embedding the 'logical' part of the reasoning.

Figure 10 applies to the stream of work conducted by Jean Sallantin and his group [Nobrega et al, 2000] around the interaction between humans and machines during the theory building process; it articulates the two conjectured reasoning modes into a framework based on the three worlds we have defined at the beginning of this chapter:

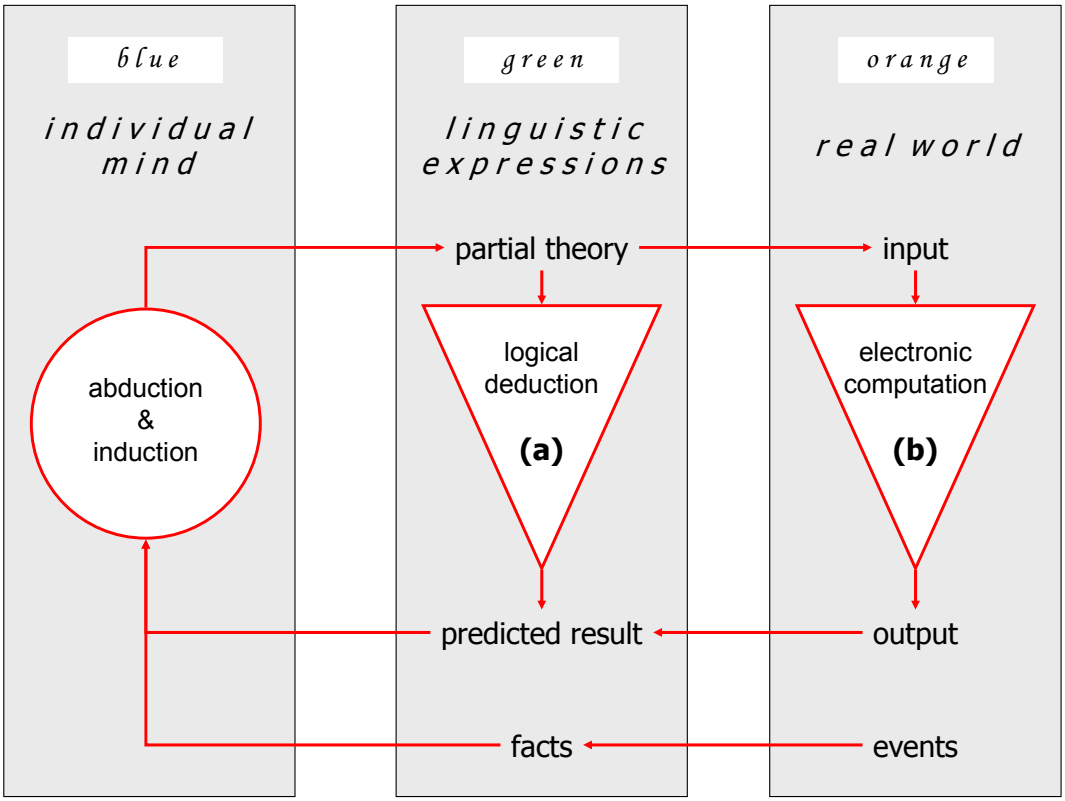


Figure 10: a framework for collaborative theory building

According to Figure 10, the BLUE world is where *theory discovery* takes place through the process of *selectionist thinking* grounded on the following potentialities:

- selective grouping in aggregates on the basis of co occurrence of events;

---

reliably tell which is which, then the machine is said to pass the test. It is assumed that both the human and the machine try to appear human. In order to keep the test setting simple and universal (to explicitly test the linguistic capability of the machine instead of its ability to render words into audio), the conversation is usually limited to a text-only channel such as a teletype machine as Turing suggested or, more recently IRC or instant messaging. [WikipediA]

- selective association of aggregates on the basis of similarity;
- selective association of aggregates on the basis of observed causality.

When these potentialities are associated with linguistic capabilities, they are reinforced by expression, imitation of others and social learning and therefore respectively support conceptualization, induction and abduction. Each individual mind becomes able to produce a *partial theory* about any given situation of the real world.

However, nothing is coded in the mind: thoughts are inside, but formulas are outside[41] . The GREEN world is the world where formulas exist and can be manipulated within formal systems through the process of *logical deduction*, as illustrated by path (a); inside formal systems, partial theories produce predicted results, which are tested against real facts and subsequently falsified and improved. If we want to empower the cycle of 'collaborative theory building' in such a way that path (b) substitutes path (a); we need an abstract machine such that both *input* and *output* of the computation are directly expressed in the language of the partial theories. The use of an abstract machine able to rule concurrent partial theories will ensure the correctness [42] of rewriting, and *compositionality* may come as an extra benefit. Moreover, in case of a *confluent* abstract machine, the result of the global rewriting can be interpreted as the 'logical consequence of the interaction within the formal system', i.e. *predicted results*; so that the falsifiability of the global theory is addressed without extra effort.

Although the transformation of expressions is ruled within the green world, it actually becomes effective in the ORANGE world through computation, therefore becoming sharable and checkable. Provided the fact the abstract machine is compositional and confluent, computation can efficiently serve the collaborative theory building. We recall in the next sub section the requirements we have collected for achieving this initial goal.

[41] To this last point, it might be objected that we think with words, and mentally count with numbers; our answer to this will be two fold:
- as long as words and formulas remain in our heads, they do not exist in an 'objective' way and cannot be shared with others;
- words and formulas participating in our neural processes are not exactly words and formulas, but personal and adaptive memories of words and formulas, i.e. capacities to interpret or to emit words and formulas.

[42] as shown in [8.4], the human manipulation of formulas is subject to errors, even on simple parsing tasks.

2.1.4 Summarizing the requirements

1) Our major starting point has been an analysis of collaboration processes sustained by conversations: meaning is conveyed by speech acts, while the major control issues are the introduction of topics and the turn taking.; then we semi-formally defined a theory as a set of true expressions in a formal language such that their rewriting produces only true expressions with respect to the collective oracle constituted by a group of individuals, each with his/her own capabilities for induction and abduction. We propose that collaborative theory building is enhanced by an abstract machine addressing the logical composition of partial theories.

Following the original idea of John McCarthy, we require this abstract machine to run "quasi-Elephant-2000 programs" and we label it 'conversational', that is to say it should be based on a simplified set of *speech acts* for interaction within components or with users; we propose in Table 1 the following set:

illocutionary act in the sense of Austin	**role in a conversation between artefacts**
assertive	information transfer
commissive	information about future information to be transferred
directive	order, command, request

Table 1: illocutionay acts for conversations between artefacts

According to [2.1.2.2], we must therefore solve the questions of Table 2:

5 questions for a conversational abstract machine
1. the 'turn-taking' question: how long must requests wait for answers? 2. the 'modelling of the others' question 3. the 'topic introduction' question 4. the 'halting' question: will the conversation end in finite time? 5. the 'concurrency' question: we want the conversation to remain deterministic for an outside observer although the chronology of illocutionary acts is not pre-defined.

Table 2: ruling a conversation between artefacts

2) If we want the abstract machine to substitute human deductive reasoning and allow direct falsifiability of the global theory resulting from the composition of the individual partial theories, facts and hypotheses must be understandable by humans and in the same time computable:

- facts expressed through speech acts must be *recognized*[43] *as input* by the abstract machine;
- partial theories must become logical components *recognized as input* by the abstract machine;
- speech acts output by the abstract machine must be *interpretable as predicted results* by the builders of the desired theory; this requires the confluence and compositionality of the abstract machine.

Figure 11 illustrates that if *LH(1)* and *LH(2)* are recognized, then *LH(1)* $+^{44}$ *LH(2)* must also be recognized and integrated for the output of predicted results.

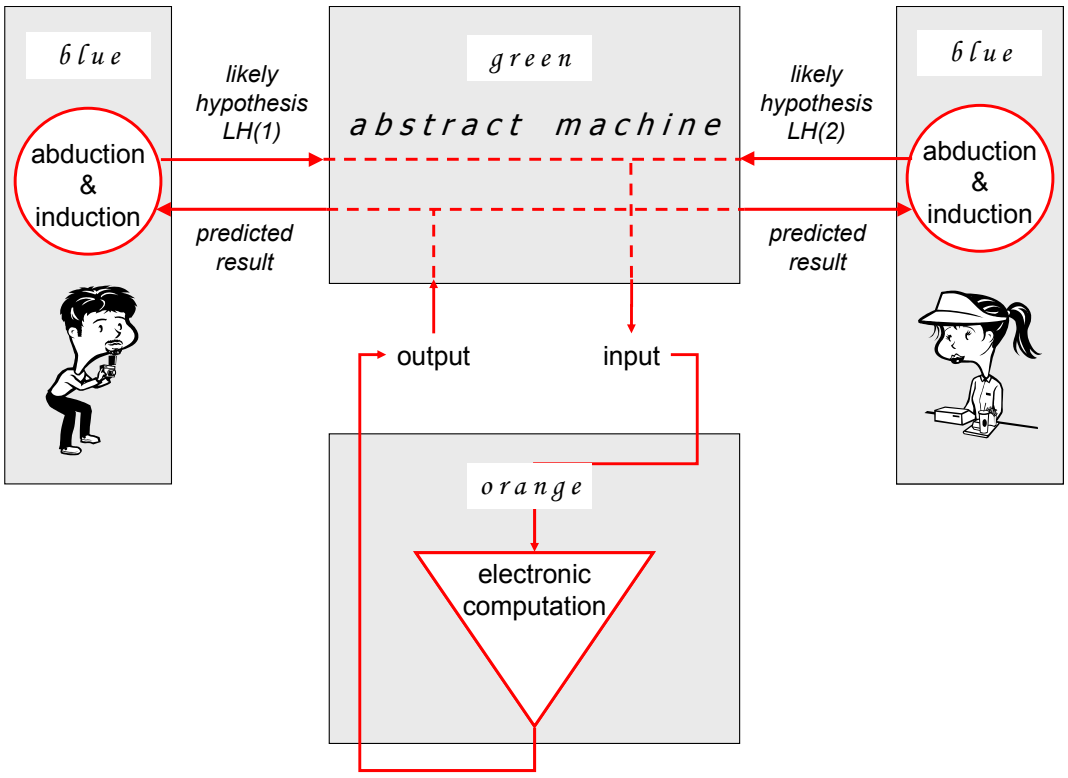


Figure 11: confluence and compositionality

[43] "recognized" means accepted as input and computed in finite time.
[44] to be formally defined (see Chapter 4); its intuitive meaning is "composition".

Figure 12 illustrates a successful collaboration between the 'cells of a grid populated with values'; each cell is a contributor to a desired theory (about the value of empty cell marked by a circle) emitting observations or likely hypotheses:

- the left part illustrates initially disjoint partial theories: each cell emits its own hypotheses (about the value of empty cell marked by a circle) on the basis of its own observations;
- the right part illustrates a shared theory resulting from a successful collaboration: local hypotheses are composed and globally derived and predicted results are tested against all the observations.

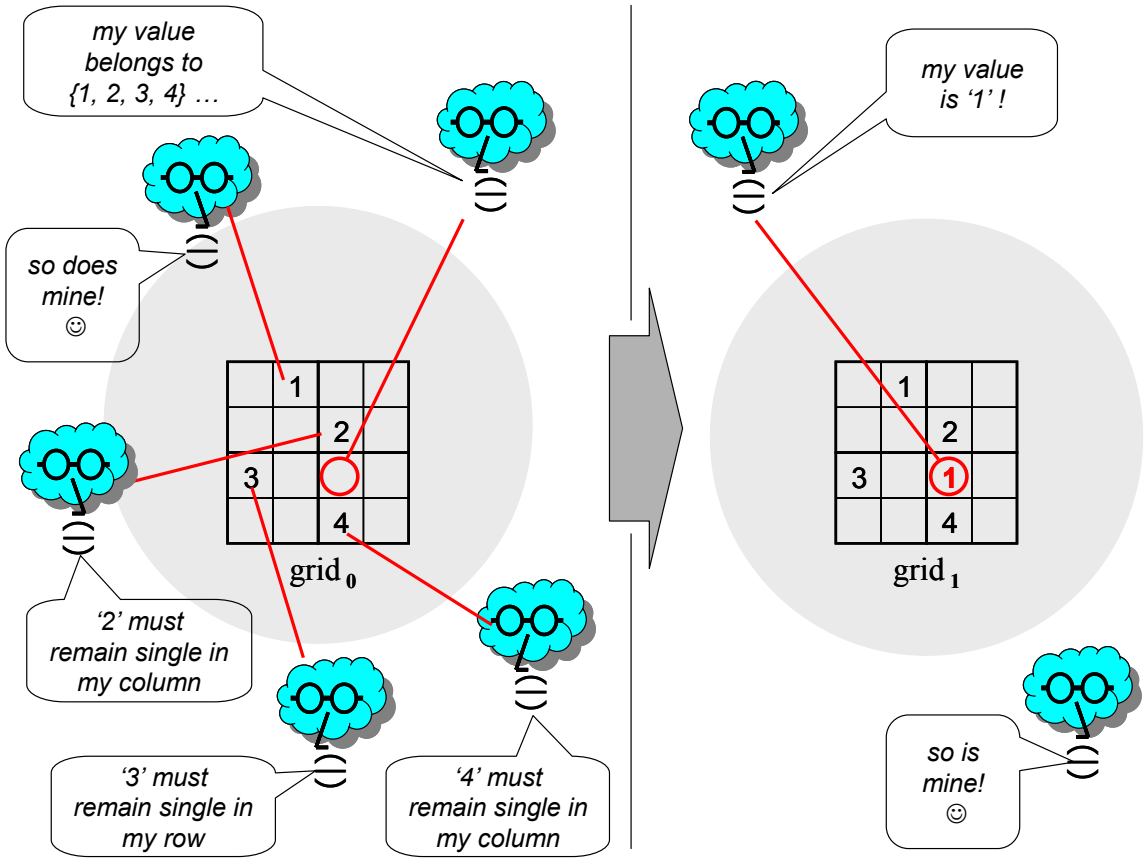


Figure 12: a successful collaboration in theory building

Before addressing the above requirements in Chapter 3, and in order to give concrete handles to the understanding of the abstract machine, we anticipate hereafter in [2.2] an intuitive but precise form of our main example in this thesis: the 'Sudoku filling problem'.

2.2 Filling a Sudoku

'Sudoku' is a logic-based placement puzzle.

The aim of the puzzle is to enter a numerical digit from 1 through 9 in each cell of a 9×9 grid made up of 3×3 subgrids (called 'regions'), starting with various digits given in some cells (the 'givens'); each row, column, and region must contain only one instance of each numeral. Completing the puzzle requires patience and logical ability. An early variant of the puzzle was published in a French newspaper in 1895 and may have been influenced by the great Swiss mathematician Leonhard Euler, who likely created Latin squares. Modern interest in Sudoku stems from a revival in Japan in 1986, leading to widespread international popularity in 2005 [from http://en.wikipedia.org/wiki/Sudoku]"

The name 'Sudoku' is the Japanese abbreviation of a longer phrase, 'Suuji wa dokushin ni kagiru' (数字は独身に限る), meaning 'the digits must remain single'.

For instance, the grid below appeared as "Sudoku n°53" in "Le Monde" newspapers:

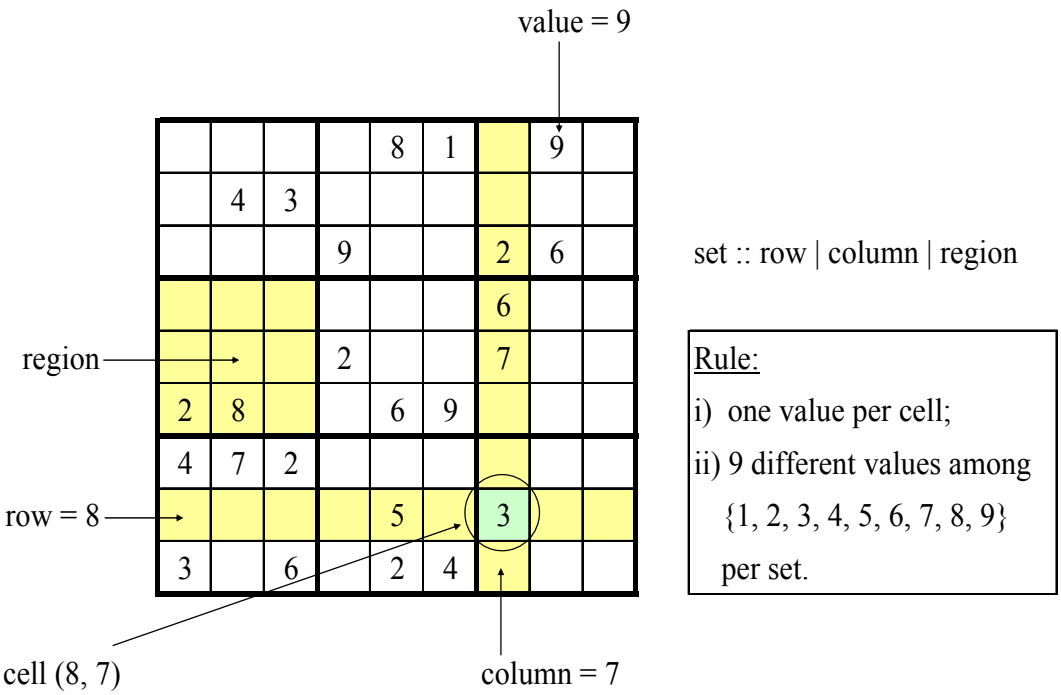


Figure 13: Sudoku n° 53 [Le Monde]

A first viewpoint on 'Sudoku' is to see it as a 'constraints satisfaction problem' and to explore the huge set of possible grids (81! = 81×80×79 … ×2×1) in order to keep only those which respect the rule: 'the digits must remain single'. Efficient solvers already address this viewpoint, a nice page where puzzles are created and solved by machines is:

http://www.cs.qub.ac.uk/~I.Spence/SuDoku/SuDoku.html

But 'Sudoku' becomes a game for humans as soon as among all the possible grids completing the initial one, only one fulfils the rule! Moreover, in popular 'Sudoku', this unique solution can be logically deduced step by step.

To introduce a general discussion on a simple example, we take the following convention: a grid with $2^4 = 16$ cells will be considered as a 2-sudoku, i.e. a sudoku of dimension '2', ordinary sudoku with $3^4 = 81$ will be considered of dimension '3' …

Figure 14 illustrates the step by step resolution of a 2-sudoku, starting from the initially given '$grid_0$':

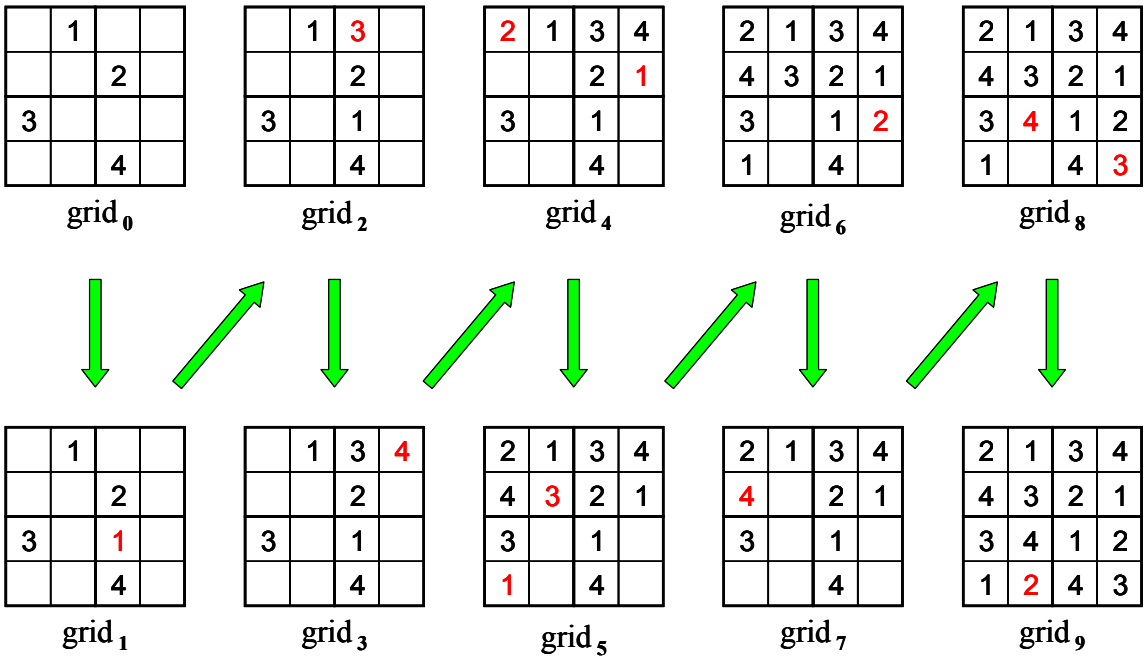


Figure 14: deduction-based filling of a 2-sudoku

- because of the value '3' present in row=3 and the values '2'&'4' present in column=3, the only acceptable value is '1' for the cell (row=3, column=3), therefore: '$grid_1$';
- '$grid_2$' is easily deduced by checking again colum=3;
- '$grid_3$' follows;
- in '$grid_4$' two values come at once: '2' is easily deduced from the row, and '1' from the 'region' i.e. the 2×2 upper-right subgrid (a n-sudoku counts n^2 regions);
- in '$grid_5$' two more values come easily;

- ‘$grid_6$’, ‘$grid_7$’, ‘$grid_8$’and ‘$grid_9$’ follow; ‘$grid_9$’ is the unique solution!

Although the solution is unique, 3-sudokus require “patience and logical ability” because of:

- the number of cells to check;
- the necessity to apply more complex reasoning patterns than the ‘only one left possibility’ solving rule.

Our focus will be on the process of collaboratively elaborating a filling strategy for ‘Sudoku’, as an example of collaborative building of a theory. We look for a method leading to the solution without any backtracks, relying only on the quality of the reasoning. Moreover, we intend to take advantage of the conversational viewpoint by allowing several tests to be processed in parallel by several “participants in a conversation”.

Therefore, our viewpoint on ‘Sudoku’ is not the viewpoint of “exhaustive exploration and testing with backtrack”, but the viewpoint of “concurrent logical conversations stepping forward”. Moreover, our interest is not in the efficiency of a particular “Sudoku filling theory”, but in the formal approach and the subsequent requirements for a conversational abstract machine.

We propose in this chapter to imagine that we are in 1895, when all the abundant literature about Sudoku manual solving techniques like ‘candidate elimination’, ‘what-if’ approach does not exist yet … Donald Knuth (who once warned users of his software: *"Beware of bugs in the above code; I have only proved it correct, not tried it*”) has not yet written the popular ‘Dancing links algorithm’ which solves a general problem of which ‘Sudoku’ is a particular example. Charles Sanders Peirce is 56 years old.

In 1895, a group of friends is looking for a set of rules solving any Sudoku by logical deduction; this set of rules being called a “theory”. Our interest today is not in the efficiency of their particular “Sudoku solving theory”, but in the formal approach and the subsequent requirements for a conversational abstract machine:

- we take inspiration from Shapiro’s formalization of the ‘model inference problem’ presented in [8.3] for describing the process of finding a strategy for ‘Sudoku’;
- then we observe the building of a first version of the theory based on a language for representing the Sudoku, emitting hypotheses and deriving

through an imagined conversational abstract machine based on multi-steps algorithms;

- then we observe some evolutions of the theory and informally check that the abstract machine allows easy updates and control of the prediction of results;
- to end with, we come back to our formalization of the "theorizing between friends" cycle and recapitulate the main features of the conversational abstract machine.

2.2.1 Conceptualizing around 'Sudoku'

Compared to Shapiro's approach, ours follows a different objective. Rather than trying to produce a theory through a *logical deduction* within a pre-defined conceptualization, we wish to *select* a proper conceptualization (i.e. theoretical concepts and a reasoning pattern articulating them) upon which to build a theory predicting correct results.

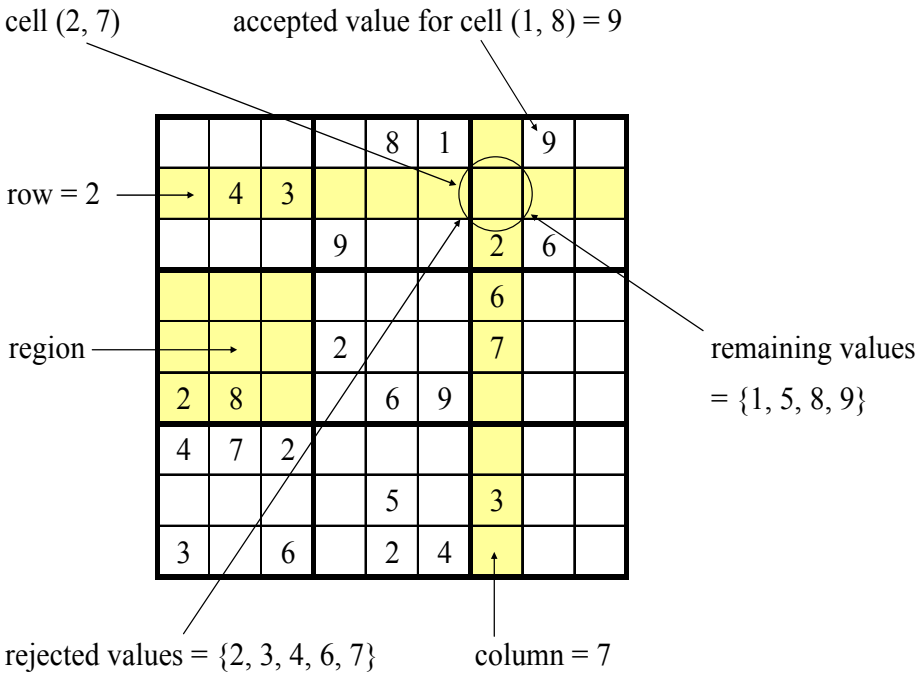


Figure 15: describing a 'Sudoku'

For the ‘Sudoku filling problem’ we suppose the friends of 1895 started with a language ‘L’ consisting in descriptions for ‘grids’ and ‘cells’ and in rules for filling empty cells. ‘L’ is base on the following concepts and functions:

main concepts and functions of ‘L’
- $cell_x^y$ is defined by ‘row = x’ and ‘column = y’ - a grid is a set of couples (cell, value), where the value may be empty - possibleValues = {1, 2, 3, 4, 5, 6, 7, 8, 9} - neighbours (cell) is the appropriate set of cells belonging to the same row, or column or region[45] - acceptedValues (cell) is a function returning either Ø or the unique value deduced by the theory - rejectedValues (cell) is a function returning a subset of possibleValues corresponding to the values forbidden by the derivation of the theory - remainingValues (cell) is the complement of rejectedValues (cell) in possibleValues

Table 3 : a description language for 'Sudoku'

Let ‘L_0’ be the subset of ‘L’ consisting in the ‘initial grid’ defined by the initial ‘givens’; it plays the role of *initial facts*.

Let ‘L_H’ be any subset of ‘L’ consisting of formulas assembled in multi-steps algorithms producing ‘grids’; ‘L_H’ plays the role of *set of likely hypotheses*.

An *oracle* M checks all the rows, columns and regions of a ‘grid’ and says ‘correct’ when the grid is filled so that the structural rules of Sudoku are satisfied. M is operating on a very simple basis, as the major theoretical concepts are known in advance (and have been incorporated in the constants, variables and functions listed above)… yet solving a Sudoku remains a difficult task. This means that incorporating the theoretical concepts into a proper set L_H of hypotheses is the real challenge.

In the theory building cycle, the goal is to find a ‘filling strategy/ reasoning pattern’ ‘L_H’ which will produce together with ‘L_0’ (the) only correct grid. Successive candidates for

[45] a ‘region’ is a 3x3 square appearing on the grid with a stronger outline (see Figure 15)

'L_H' are going to be emitted by the different contributors, and eventually refuted by the oracle.

A first example of 'filling strategy' would be the following reasoning pattern:

- from upper left cell to down right cell in rows, execute: "the smallest of remaining values is accepted".

This 'filling strategy' predicts the grid in Figure 16. The derivation stops at '$grid_8$' since no possibilities remain for filling the third row; such a grid is therefore considered 'false' by the oracle M … which refutes the 'likely hypothesis'!

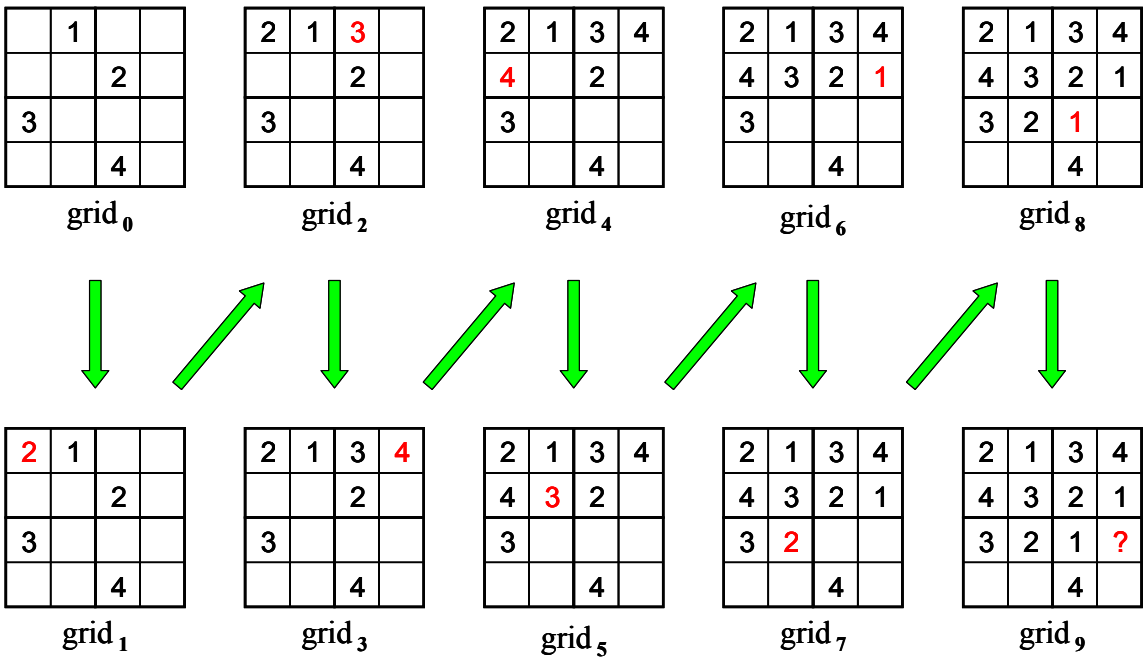


Figure 16: an unsuccessful filling strategy

2.2.2 A first version of the theory

Keeping in mind our requirements based on speech acts, we propose that cells exchange information about their local state with a 'moderator' anticipating the abstract machine, and concurrently follow local reasoning patterns on the basis of the information received; this kind of approach matches a new direction of interest in the Multi Agents Systems community, based on indirect interaction between 'agents' via an 'environment', as explored in [Keil & Goldin, 2005]. Our 'agents' are the 'cells', the 'environment' is the moderator of their conversation aimed at filling the 'initial grid'.

We will not describe here the detailed successive stages of the selection of the strategy, nor give a rigorous formalization of the 'Sudoku filling problem' (this will be done later in the examples of Chapter 3) but rather focus on the major ideas, and the subsequent generic solutions proposed to some of the problems listed in [2.1.4].

Taking the viewpoint of one 'cell', the reasoning pattern (which from the theory-building viewpoint constitutes the cell's likely hypothesis) can be informally expressed via the two following formulas:

- my own value is forbidden for my neighbours, i.e. the cells belonging to the same row, or column, or region (the digits must remain single);
- as soon as there is only one remaining possible value for me (whenever 8 values among the 9 possible values are rejected) , I accept it.

In a semi-formalized way, this gives:

- IF "acceptedValues (myself) = {thisValue}" THEN: "for each $cell_x^y$ in neighbours (myself), 'thisValue' must be added to rejectedValues ($cell_x^y$)";
- IF "remainingValues (myself) = {thisValue}" THEN: "acceptedValues (myself) ← {thisValue}".

Therefore, the concurrent conversations between cells and moderator can be informally described by four possible speech acts:

a. $cell_x^y$ / *directive*/ to moderator: "have I already got an accepted value"?
b. $cell_x^y$ / *assertive*/ to moderator: "this value is rejected for my neighbours"
c. $cell_x^y$ / *directive*/ to moderator: "what are my rejected values"?
d. $cell_x^y$ / *assertive*/ to moderator: "this value is my accepted value"

These speech acts integrate into a partial theory obeying the following 'conversational pattern':

- "emit (a.) ; wait for answer" …
- IF answer to (a.) is positive THEN: "emit (b.); end" ELSE: "emit (c.); wait for answer" …
- IF answer to (c.) is such that only one value is remaining THEN: "emit (d.); emit (b.); end" ELSE: "end"

If we now take the viewpoint of the moderator responsible for providing the answers able to re-activate the multi-steps algorithms, we immediately face the 'turn-taking' question: "How long must requests wait for answers?"

We do not want the issue of the conversation to depend of any arbitration from the moderator like "I wait for 5 minutes, or 3 answers, and then decide": if this was the case, the resulting conversation would add to the strict composition of the local theories embedded in the 'conversational patterns' a bias due to its moderation. We want the moderator to introduce no bias in the reasoning … therefore he must answer the questions on the basis of the totality of the potential answers. This is a first strong requirement for the future conversational abstract machine.

At this point, it is important to note that the moderator needs some kind of indication about the potential answers: each cell which does not yet know its accepted value but will hopefully know it later in the conversation must inform the moderator that it will later emit some pertinent answer. The possibility to express that some *assertives* or *directives* will or will not be emitted is what 'Principle one' requires; it relies on a specific illocutionary act called the *commissive*.

The *commissives* corresponding to a conversational pattern must be emitted as soon as it is invoked and must be then substituted by *assertives* or *directives*. They contain variables that indicate how this further substitution will take place. Taking for instance 'u' and 'v' as variables, the *commissives* (e.) and (f.) pre-announce respectively (b.) and (d.):

e. $cell_x^y$ / *commissive* / to moderator: "value 'u' is rejected for my neighbours"

f. $cell_x^y$ / *commissive* / to moderator: "value 'v' is my accepted value"

b. $cell_x^y$ / *assertive*/ to moderator: "value '9' is rejected for my neighbours"

d. $cell_x^y$ / *assertive*/ to moderator: "value '9' is my accepted value"

As soon as a cell *asserts* (d.) it will remove *commissive* (f.); as soon as a cell *asserts* (b.) it will remove *commissive* (e.). Commissives are mandatory: all cells always "pre-announce" before transferring information. This provides a solution to the 'modelling of the others' question: as soon as all *commissives* on a given topic have been substituted, the moderator knows that all the pertinent *assertives* have been emitted and takes turn in answering.

Let us consider '$cell_2^7$' in Figure 15; this cell is the neighbour of 3x8=24 other cells, and therefore is concerned by several *commissives* indicating rejected values (commissives of type: e.). When (and only when) all these *commissives* (e.) in which '$cell_2^8$'is the

neighbour have been removed, '$cell_2^7$' can (and will) have an answer to its question (c.) "what are my rejected values"?

This allows us to refine and complete our "first strong requirement":

Principle one:

- *assertives* and *directives* are definitive utterances;

- *commissives* are non-definitive utterances;

- all *assertives* and *directives* emitted during the conversation must be pre-announced by adequate *commissives* emitted at the beginning the conversation.

Principle two:

- *answers* to *directives* are decided when "pertinent" *commissives* have been removed ; they consist of the totality of "pertinent" *assertives*.

At this point, we may note that the 'topic introduction' question is now virtually solved: the topic introduction by each 'cell' is ruled by its own 'conversational pattern', these patterns will be automatically informed "at the right moment" by the moderator.

Now we may come to another remark: all 'cells' behave the same way. They are in fact 81 invocations of the same 'conversational pattern': CELL. Allowing the abstract machine to consider generic 'conversational patterns' working under different contexts will save much effort to people contributing to the elaboration of the filling strategy … but this requires the introduction of a special mechanism for invocation under such and such context.

We propose that the moderator decides invocations of the 'conversational pattern' on the basis of:

i) the currently available *assertives* or *directives*;

ii) the *commissives* pre-announcing the future *assertives* or *directives*.

The same common basis should provide knowledge about whether future invocations may occur, or on the contrary whether the conversational pattern may be revoked[46].

In our 'Sudoku' case, the pattern CELL will first be concurrently invoked under the 81 contexts[47] and then revoked.

This is formulated in 'Principle three' given below:

[46] This is discussed in [8.5]
[47] we shall see in chapter 3 that 'CELL' can be revoked only after 'LAUNCHER' is revoked

Principle three:

- *invocations* of 'conversational patterns' are decided on the basis of "pertinent" *assertives* and *directives;*

- *revocations* of 'conversational patterns' are decided when all "pertinent" *commissives* have been removed.

The word "pertinent" is used here in an intuitive way and will be substituted in Chapter 3 by two formally defined relations between sentences: the relation "answerFor" will establish the pertinence in case of Principle two, while the relation "triggerFor" will establish the pertinence in case of Principle three.

We may now recapitulate:

i) we suppose a conversational abstract machine:

- respecting Principles one, two and three
- embedding control rules for 'answering questions' and 'invoking/revoking conversational patterns'.

ii) the first version of the candidate theory consists of the following:

- the 'givens' describing the 'initial grid' are input as *assertives*: "value 'z' accepted for $cell_x^y$"
- an initial *directive*: "what solution for sudoku?" is also input at the beginning in order to start the coputation
- a 'conversational pattern' LAUNCHER first announces by its *commissive* and then *asserts* the contexts for the cells
- a 'conversational pattern: CELL' with its proper *commissives* which will be substituted by the contextualized *assertives* (b.) & (d.) and *directives* (a.) & (c.)…

iii) the moderator invokes LAUNCHER then concurrently invokes 'CELL' under the 9×9 contexts. Each 'cell with an already accepted value' informs its neighbours that this value must be rejected for them. Each cell asks about its rejected values. Hopefully one or more cells count only one remaining value and accept it …

2.2.3 Refining the theory

… the reader will have noticed that the first version of the candidate theory cannot work. This appears as a consequence of the design of the CELL pattern:

- each cell first emits the *commissive*: "value 'v' is my accepted value" which pre-announces some further *assertive*: "this value is my accepted value"
- each cell then emits the *directive*: "have I already got an accepted value?"
- in order to answer the directive, the moderator must wait for the commissive to be removed, but this can happen only when the answer is received … we have a dead-lock! Moreover, the mutual informing and questioning about rejected values between neighbours creates another deadlock!

The responsible for this state of affairs is the current conceptualization: 'time' has not found its place in the concurrent conversations, how could they (the friends of 1895) hope any progress in the collaboration between cells?

A first necessary extension of the language is therefore the **addition of a new concept**: the concept of 'state in the resolution':

- the 'initial grid' is labelled 'state 0'
- the 81 cell invocations gather information from 'state 0' and produce 'state 1' …
- then gather information from 'state 1' and produce 'state 2' …

More precisely, 'time' has been incorporated in the theoretical concept 'state', which together with 'row' and 'column' settles a three-dimensional referential, and the conversation becomes:

- $cell_x^y$ / *directive*/ to moderator: "have I already got an accepted value at state <='n'?"
- $cell_x^y$ / *assertive*/ to moderator: "this value is rejected for my neighbours at state 'n+1' "
- $cell_x^y$ / *directive*/ to moderator: "what are my rejected value at state <='n'?"
- $cell_x^y$ / *assertive*/ to moderator: "this value is my accepted value at state 'n+1' "

Instead of a 'conversational pattern' invoked in 81 contexts (x, y) , we now have a new one invoked in 81× 'number of states' contexts (x, y, n).

A detailed examination of this new conceptualization would show that all the concurrent invocations auto-organize[48]: although questions about all states 'n' are simultaneously asked, the causal dependency coded inside the patterns (questioning state 'n' and

[48] As a drawback, the number of states has to be given a pre-defined value, otherwise the 'conversational pattern' cannot be revoked and its initial commissives prevent any answer; this number is necessarily smaller than 81 since the whole approach will be successful only if one new value at least is accepted at each state!

asserting state ‘n+1’) automatically induces a chronology of answers corresponding to states ‘0’, ‘1’, ‘2’ … via the answering rule!

It will be shown in [4.6] that this second version of the theory incorporating time works without deadlocks … but unfortunately stops before filling the initial grid completely, by lack of deductive power!

To remedy this last point, another extension of the language is required, including the **introduction of an additional reasoning pattern** embedding the viewpoint of a ‘set’ of cells (a set is either a row or a column or a region):

additional concepts and function
- ‘set_i’ is one of the 27 sets (9 rows + 9 columns + 9 regions); - cells (set_i) is a function enumerating the 9 ‘$cell_x^{y'}$ of a ‘set_i’

Table 4: completing the description language for 'Sudoku'

To ‘set_i’ we associate the following speech acts:

g. set_i / *directive*/ to moderator: “what are my accepted (value, cell) at state <= ‘n’”?

h. set_i / *directive*/ to moderator: “which values are rejected for my empty cells at state <= ‘n’”?

i. set_i / *assertive*/ to moderator: “this value is accepted for that cell at state ‘n+1’ ” for my neighbours”

j. set_i / *assertive*/ to moderator: “this value is rejected for neighbours of that cell at state ‘n+1’ ”

These speech acts integrate into a partial theory obeying the following ‘conversational pattern’ SET:

- “emit (g.) ; wait for answer” …
- “emit (h.) ; wait for answer” …
- IF answers to (a.) is positive THEN: “emit (b.); end” ELSE: “emit (c.); wait for answer” …
- IF answer to (g.) and (h.) are such that only one value is remaining for one cell inside the set THEN: “emit (i.); emit (i.); end” ELSE: “end”

The interesting point is that this new ‘conversational pattern’ SET is simply added to CELL without any additional effort.

This third version of the theory is the good one: a concurrent conversation between LAUNCHER, CELL and SET involving ((81×'$cell_x^{y'}$+27×'set_i')×15 'states') invocations successfully solves the 'Sudoku n° 53' given at the beginning of this chapter, as it will be presented in more details in [3.10] and [4.5.2].

2.2.4 Moderating principles for a conversational abstract machine

While solving the sudoku, we have encountered two major kinds of "objects" that we want our abstract machine to compute in a compositional way (see Figure 17): *sentences* and *conversational patterns*.

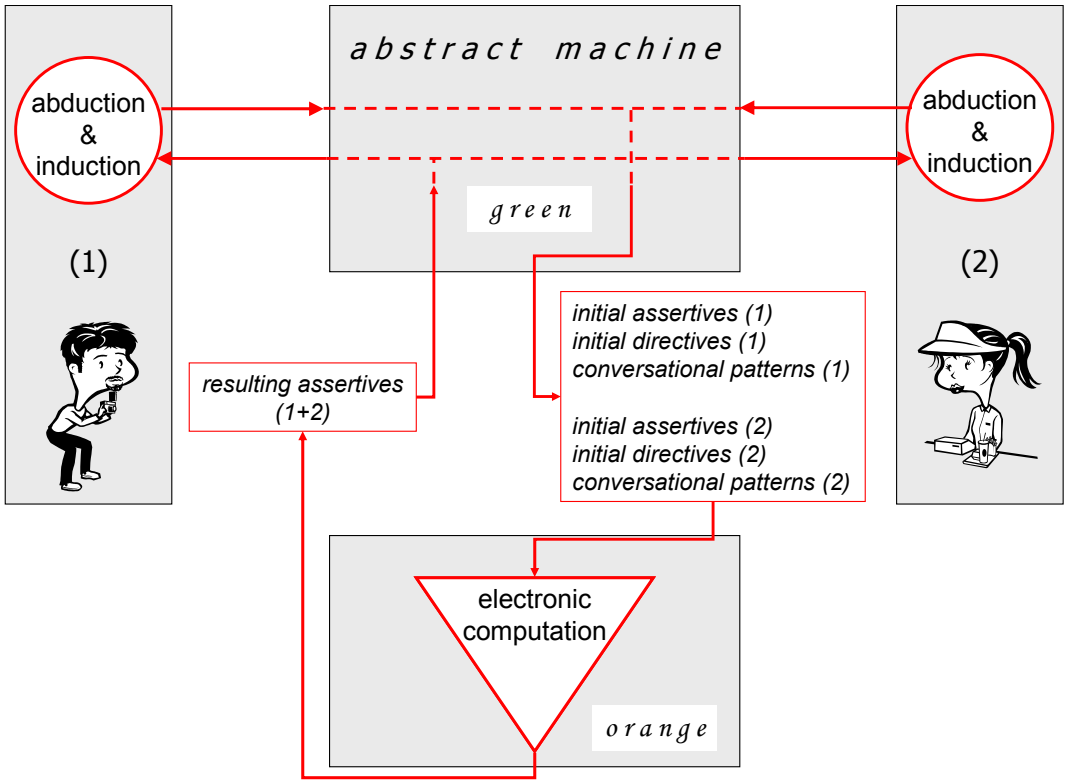


Figure 17: a conversational abstract machine

Assertives and *directives* are *sentences* of some language $\mathcal{S}$ aimed at knowledge representation and information transfer; $\mathcal{S}$ directly addresses the conceptual level.

The fragments of the desired theory take the shape of multi-steps *conversational patterns* directly operated by the abstract machine; the initial external description of these patterns consists of *commissives* expressed in the same language $\mathcal{S}$ while their internal description supports logical deduction.

Inside the abstract machine, the logical derivation mainly consists in invoking, replying to and revoking the conversational patterns. During the derivation, *commissives* are

replaced by *assertives* or *directives*; *directives* are answered by *assertives*. The conversation is global (i. e. computes all the patterns and takes into account all the available information) and the predicted results are *assertives* directly meaningful for the theory builders. The abstract machine is required to obey the following principles:

moderating principles for a conversational abstract machine
Principle one: - assertives and directives are definitive utterances; - commissives are non-definitive utterances; - all assertives and directives emitted during the conversation must be pre-announced by adequate commissives emitted at the beginning the conversation. Principle two: - answers to questions (directives) are decided when "pertinent" commissives have been removed ; they consist of "pertinent" assertives. Principle three: - invocations of conversational patterns are decided on the basis of "pertinent" assertives and directives which play the role of context; - revocations of conversational patterns are decided when "pertinent" commissives have been removed.

Formalizing the above moderating principles into a calculus is the object of the next chapter; two problems will be addressed:

1. how does the moderator decide when the questions expressed by *directives* are ready to be answered?
2. how does the moderator decide when invocation or revocation of 'conversational patterns' is necessary?

2.3 Refining the problem definition

We have expressed in [1.2] our wish to sustain the collaboration between experts in the building of a shared theory, by addressing two issues:

i) representing the knowledge, i.e.: organizing a considerable set of data and more or less complex decision rules into 'partial theories', in a manner *as close as possible to the experts' initial conceptual schemes;*

ii) processing the information, i.e.: assembling the 'partial theories' into a unified set of expressions and providing a controllable global derivation, with *as little extra effort as possible*.

To achieve this double goal, we have started to specify a conversational abstract machine playing the role of controller in the composition of partial theories. To a certain extent the conversational patterns embedding the partial theories might appear as "experts" in charge of collectively solving a problem, in a way close to the *actors*[49] model of [Hewitt, 1977] where the communication mechanisms would be implemented through the protocols ruling the conversation: "We are investigating the problem solving model of a society of experts to supplement the model of a single very intelligent human. We submit that this change in focus has several beneficial results. It provides abetter basis for naturally introducing parallelism in problem-solving since protocols of individual people do net seem to exhibit so much parallelism. The change in focus helps to make mechanisms for the communication of knowledge more explicit … in these ways we hope to develop the communication mechanisms that are necessary to achieve cooperation between expert modules for various micro-worlds in order to perform larger tasks which call for the expertise of more than one micro-world …"

But this comparison of approaches has clear limits. Neither do we aim to simulate the behaviour of scientific societies, nor do we forbid global states, whereas the actors model emphasises locality and direct (non moderated) communication between *actors* as it is expressed further in [Hewitt, 1977]: "… Further more the change should foster better specifications for tasks to be achieved so that appropriate experts can be selected or synthesized… The long term goal is to construct systems whose behaviour approximates the behaviour of scientific societies. That is the ultimate aim is to build systems which model the way scientists construct,

[49] [Hewitt, 1977]: "The basic construct of our computation model is the *actor*. The *behaviour* of each actor id *defined* by the relationships among the events which are caused by the actor. At a more superficial and imprecise level, each actor may be thought of as having two aspects which together realize the behaviour it manifests: the *action* it should take when it is sent a message; its *acquaintances* which is the finite collection of actors that it directly *knows about*."

communicate, test, and modify theories … Actors are a local model of computation. There is no such thing as "action at a distance" nor is there any "global state" of all actors in the universe. Actors interact on a purely local way by sending messages to one another".

Instead, we have the much more restrictive aim of *only enhancing* (not simulating) collaborative theory building among scientists or experts: this means that humans assume the difficult challenge of 'theory discovery', whereas the conversational machine is only in charge of composing these partial theories into a unified theory which can then be easily 'checked'. The major reasons for this choice can be found in [2.1]:

- the primitives for 'theory discovery' (*induction* and *abduction*) belong to *selectionist thinking*[50] , a mode of thinking for which artefacts show little ability;
- on the other hand, interactions between theory builders through conversation need formal rules, both at syntactic and at protocol levels, and will benefit from some kind of confluent moderation.

For "organizing a considerable set of more or less complex decision rules into partial theories in a manner as close as possible to the experts' initial conceptual schemes", we need a language $\mathcal{S}$ directly addressing the conceptual level used by the theory builders and at the same time directly computable by the abstract machine.

For "assembling the partial theories into a unified set of expressions and providing controllable derivation of hypotheses with as little extra effort as possible", we choose the metaphor of a moderated confluent conversation based on *assertives*, *directives* and *commissives* involving multi-steps *conversational patterns*.

The management of the successive global states of the conversation requires complete information about:

- the invocation and revocation of the conversational patterns;
- their potential rewriting of commissives into assertives and directives;
- their local information needs for achieving each step.

Our problem is now to specify this conversational abstract machine.

[50] as defined in [2.1.1.5],

3 AUSTIN: a conversational abstract machine

"AUSTIN" is a theoretical model of computer software aimed at operating upon simplified illocutionary acts; it is therefore called 'conversational'. Our objective is to formalize a synchronized auto-controlled interaction between *conversational patterns* uttering and computing *sentences* in turn, by means of emitting assertives, emitting directives whenever they need information and waiting for the answers, emitting new directives and/or assertives …until they reach a final local state. Under certain conditions, "AUSTIN" will halt on a final global state depending only of the initial state of the conversation.

When describing the abstract machine, we need to introduce a number of original concepts. In order to keep a power of invocation and facilitate a global understanding, we borrow their names from our daily vocabulary … but this should not drive the reader away from the formalization! In order to conciliate 'power of invocation' and 'rigorous syntax' we have adopted the following policy: we shall use italics each time a formalized concept is used for the first time in a paragraph, which means that the reader can refer to the glossary given in [3.6] where s/he will find both an informal definition and the address of the formal definition inside this chapter.

Moreover, we shall refer all along the chapter to examples of increasing complexity. The first two examples are only partial examples inviting the reader to give some meaning to the conversations; neither detailed syntax nor rigorous semantics have been associated to the language used for sentences, nor detailed algorithms for the transformations operated on them. A full example of a language with detailed syntax and associated semantics is given in [3.7] and will support the third example given in [3.8]: the 'Sudoku filling problem'.

This chapter starts by an introduction of the main concepts and denotations; after a quick overview of the calculus, a short conversation is analyzed.

Formal definitions for sentences and utterances are given in [3.2]; this sub-section includes in [3.2.1] a list of the few necessary hypotheses for good calculability properties.

The reducers and the conversational patterns are defined in [3.3]; a illustrated by a more elaborated example is given in [3.3.2].

A typology of the atomic transitions occurring in a global conversation is given in [3.4]; then we exemplify them by coming back to the example of [3.3.2].

In [3.5] we exhibit the control of the calculus; we give demonstrations of its properties and address the questions of complexity and expressivity.

In [3.6] a technical glossary marks the end of the technical presentation of the calculus.

[3.7] exhibits the basis of a generative grammar for sentences obeying the requirements of [3.2.1].

[3.8] addresses the 'Sudoku filling problem' by using the grammar presented in [3.7].

3.1 Introducing the concepts

3.1.1 Formalizing the moderating principles

The viewpoint in Chapter 3 is the viewpoint of a generic re-writing system, therefore slightly different from the viewpoint adopted in the previous chapter: we need to clearly disentangle the semantics of the speech acts from the calculus ruling their interaction. We shall therefore make as few hypotheses as possible on the set $\mathcal{S}$ of *sentences* on which we base the speech acts, later called 'utterances'. Only when we fully address a real problem shall we need a generative grammar for $\mathcal{S}$ and specify sentences accordingly.

By formalizing the moderating principles listed above, we aim at controlling the interaction between concurrent conversational patterns on the only basis of three relations defined on $\mathcal{S}$.

Sentences, utterances and rewriting:

Sentences become illocutionary acts only when embedded in *utterances*; the abstract machine will compute *utterances*, not *sentences*:

- sentences will only carry the 'static information' and will be compared to one another with respect to the embedded knowledge;
- utterances will carry additional information about the dynamics of the conversation; they will be either *definitive* (assertives or directives) or *non-definitive* (commissives).

According to Principle one, all assertives or directives emitted during the conversation must be pre-announced by commissives. The abstract machine must know whether a given commissive will generate an assertive or a directive; this leads to the following typology for utterances:

Illocutionay act	Utterance
commissive to be re-written in an assertive	non-definitive assertion
commissive to be re-written in an directive	non-definitive question
assertive	(definitive) assertion
directive	(definitive) question

Table 5: illocutionary acts and utterances

Non-definitive utterances may be re-written several times, until they become definitive. This means that commissives maybe re-written in commissives (non-definitive), or assertives (definitive) or directives (definitive).

Formalization of the answering mechanism:

According to Principle two, *directives* are requests for which the answer must consist of the totality of the potential pertinent *assertives*. In order to decide when a definitive assertion answers a definitive question:

- a relation "x answerFor y" is defined on the set of sentences;
- by extension, it applies to utterances embedding sentences.

Formalization of the invocation mechanism:

According to Principle three, *conversational patterns* are invoked according to the presence of *assertives* or *directives* matching their triggering condition. In order to decide when a definitive utterance matches the triggering condition of a pattern:

- a relation "x triggerFor y" is defined on the set of sentences;
- the triggering condition is associated to a special sentence embedded in the pattern called its *starter*;
- an utterance will trigger a pattern when its embedded sentence 'x' verifies "x triggerFor *starter*".

A major feature of the calculus:

The two previously defined relations are not sufficient for the complete formalization of our moderating principles. According to Principle two and Principle three, the abstract machine must "know" from the given commissives whether future "pertinent" assertives or directives, with respect to exhaustive answering as well as pattern revocation, may be produced. We therefore need to constrain the rewriting in such a way that it becomes

predictable, at least to a certain extent. This is formalized by a major feature of our calculus:

- a relation "x reductionOf y" is defined on the set of sentences;
- the re-writing of utterances is constrained by the following condition: "resulting sentences reductionOf initial sentences".

3.1.2 Denotations

The following denotations will be used:

Denotations:
- *sentences* will be denoted by simple letters when used in expressions c, a_1, b_1 … s …; when incorporated in the text as examples, sentences will be denoted s // expression // to indicate that the name is "s" and the content is "expression"
- $\mathcal{S}$ will denote the set of all sentences
- A, B, C … will denote subsets of $\mathcal{S}$
- *conversational patterns* will be denoted χ_I
- $(\chi_I,*)$ will denote the conversational pattern "χ_I" associated to the indefinite context "*"
- (χ_I, s) will denote the conversational pattern "χ_I" associated to the context "s"
- ω will denote a *state* of the conversation
- *conversation nodes* of χ_P , i.e. nodes belonging to conversational pattern χ_P, will be denoted χ_P^i or $\chi_P^i(\omega)$ whenever the *state* ω has to be mentioned
- *conversation steps* will either be denoted x_P^0 (embedding the conversation node χ_P^0) or $x_P^i(a)$ (embedding the conversation node χ_P^i and belonging to invocation (χ_P, a) or even $x_P^i(a, \omega)$ whenever the *state* ω has to be mentioned

3.1.3 A quick overview of the calculus

The blackboard metaphor[51], further developed in [5.2], can help the initial visualization of our calculus: *sentences* are written on a shared space and subject to transformation (rewriting) by *conversational patterns* which play the role of 'knowledge sources'. But we are rapidly going to focus on the metaphor of the conversation for presenting the operational semantics of the AUSTIN machine which owes its properties to syntactic comparison between sentences.

[51] A detailed matching between Blackboard concepts and AUSTIN concepts will be found in Table 16.

In Figure 18, sentences are "circles", either white (non-definitive: commissives) or coloured (definitive: assertives or directives). A pattern is represented by "3 adjacent squares with an embedded circle". Two types of patterns are present: when the embedded circle/sentence is white (non-definitive) the pattern is not yet invoked, otherwise it has already been invoked under a context (definitive sentence). We shall see later that parallel invocation of a single pattern by different sentences is accepted and encouraged, therefore leading to a concurrent calculus.

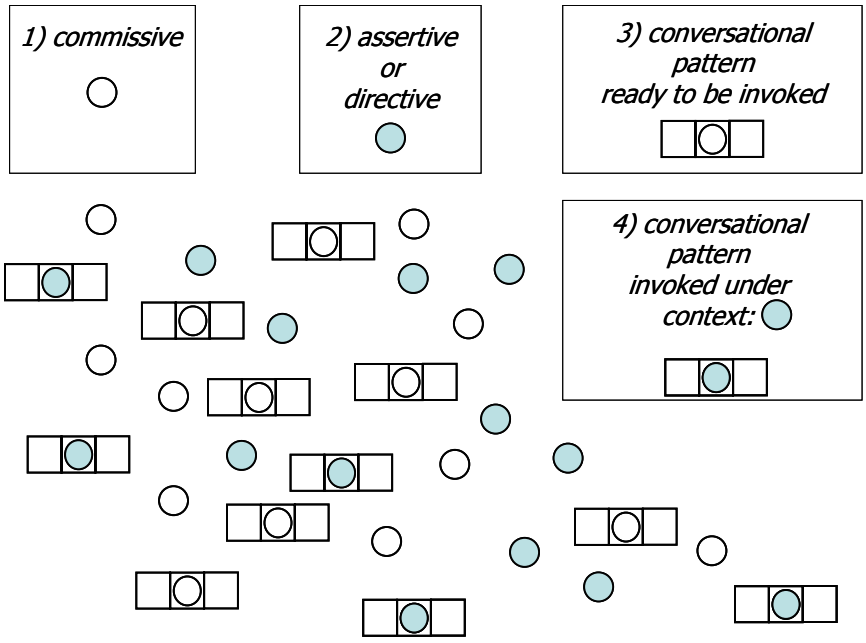


Figure 18: overview of the calculus

Looking closer at the "rewriting rules" enclosed in the *conversational patterns*, we visualize the previously mentioned syntactical features:

- the sentence embedded in the pattern is its *starter*;
- invocation of a pattern/rule (χ_3 in Figure 19) happens whenever a definitive sentence playing the role of invocation context ($s_{5,1}$ in Figure 19) is in relation "triggerFor" with its *starter*;
- a special set of sentences ($\{s_8, s_9\}$ in Figure 19) is associated to each pattern: these are the ones which will be rewritten by the pattern;

- the rewriting of $\{s_8, s_9\}$ by an invoked pattern consists in their substitution[52] by $\{s_{8,1}, s_{8,2}, s_{8,3}, s_{9,1}, s_{9,2}\}$ in such a way that $s_{8,1} <= s_8$; $s_{8,2} <= s_8$; $s_{8,3} <= s_8$; $s_{9,1} <= s_9$ and $s_{9,2} <= s_9$: patterns only write "reductionsOf" initial sentences;
- this rewriting may lead to definitive (case: s_8) or non-definitive (case: s_9) sentences, or both.

We shall see later that the rewriting depends on the invocation context; therefore several parallel invocations will lead to simultaneous rewritings of the original sentences.

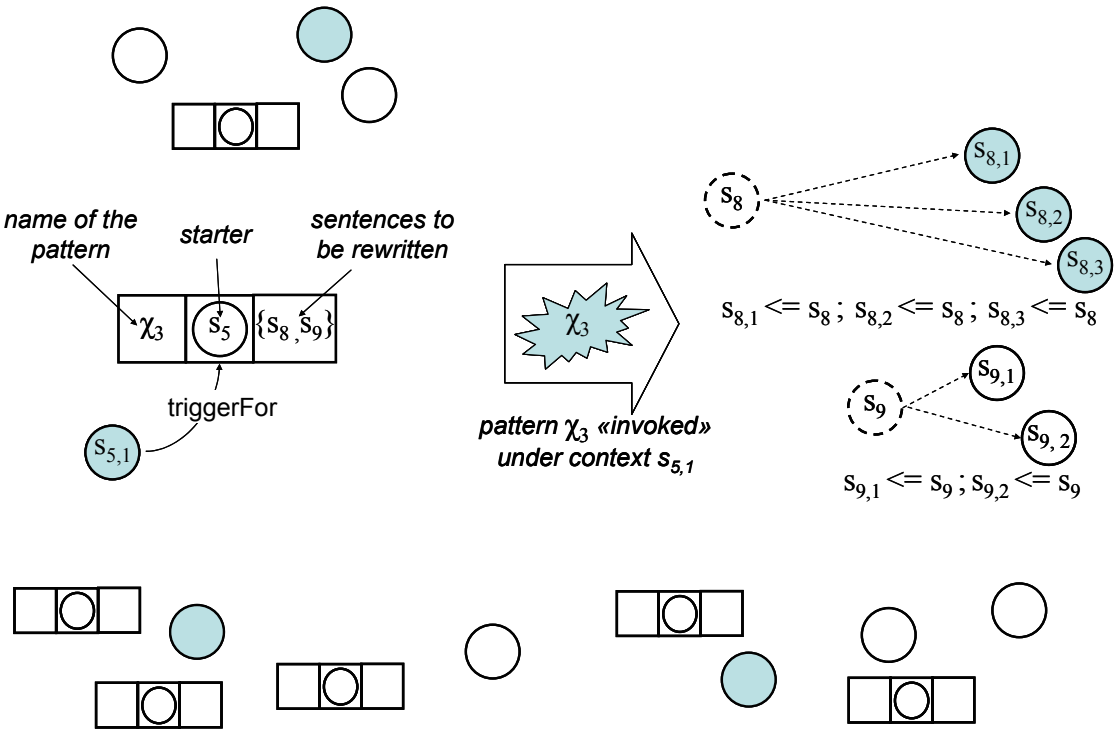


Figure 19: invocation of a conversational pattern

A last and more accurate look is now going to focus on what *conversational patterns* are: more than "context-dependant rule templates"!

Patterns will be defined in [3.3] as "graphs" of causally related "nodes (rule templates)". Figure 20 gives hints on the way these rules templates are successively contextualized:

- the upper part shows an "invocation" : χ_3 gives birth to a rewritten version "χ_3^1 under context $s_{5,1}$"; note that χ_3 remains "ready to be invoked and that

[52] "substitution" has to be understood from the view point of the invoked pattern, the original sentences remain present as long as the "ready to be invoked pattern" is present; this will be managed in the concept of "utterances".

E_3^1 is the contextualized computational result of the first "node" of the "graph/pattern χ_3";

- the central part shows a further activation of "χ_3^1 under context $s_{5,1}$"; it is supposed that directives have been emitted during precedent phase, for which answers are available now: "χ_3^1 under context $s_{5,1}$" is removed and leaves room for "χ_3^2 under context $s_{5,1}$" according to new information;
- the lower part shows the revocation of χ_3 when no more sentence will ever play the role of "triggerFor" its starter: the pattern is removed.

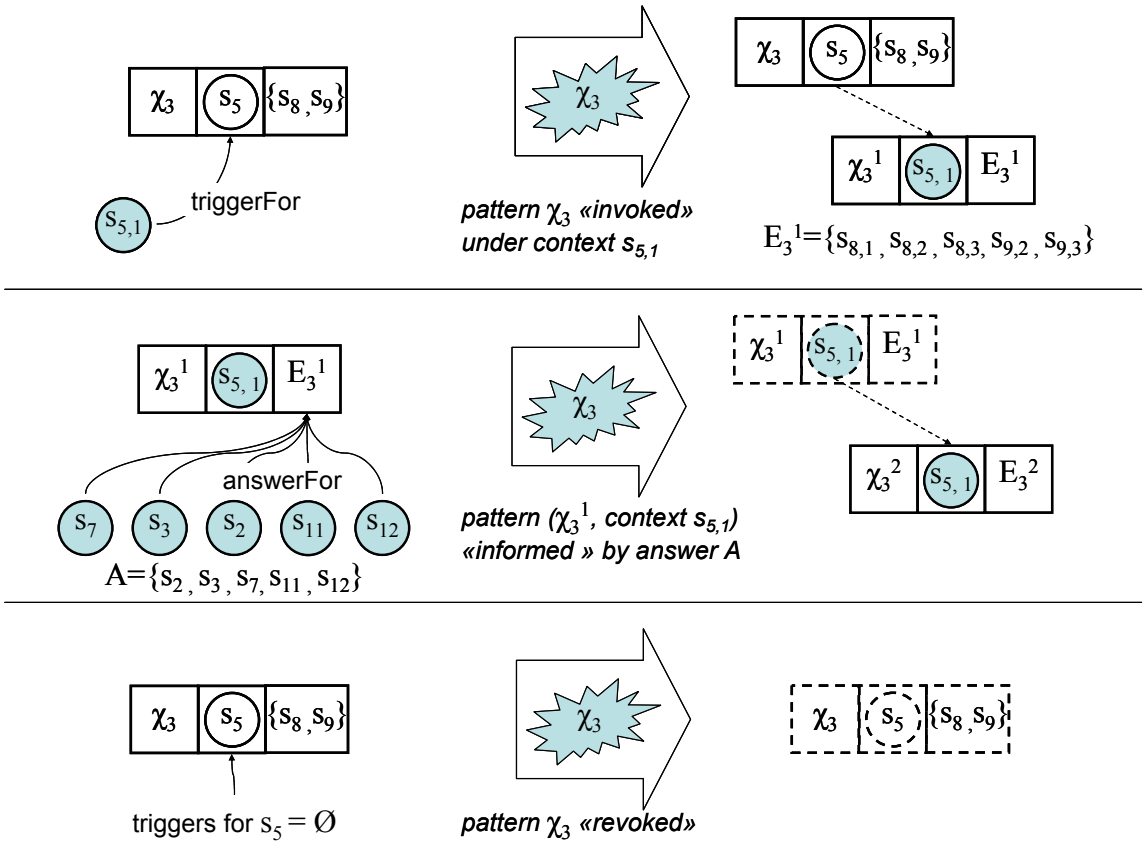


Figure 20: three types of transition

This brief survey of the main concepts has brought in almost all the underlying principles of a "conversational abstract machine" which adds and removes *sentences*, invokes and revokes *conversational patterns*, under a syntactic control driven by 3 relations between *sentences*:

1. *conversational patterns* are invoked and revoked according to the "triggerFor" relation;

2. each *conversational pattern* consists in a pre-defined graph of "nodes" which process initial context and ask questions, wait until further information is delivered (by other patterns) according to the "answerFor" relation, then process this additional information, ask other questions and wait again …
3. the calculus is globally monotonous because of the "reductionOf" relation (this relation is a partial order denoted "<=") imposed on the rewriting.

3.1.4 A first colourful example

A conversation needs some participants, in AUSTIN they are called 'emitters' because they concurrently utter sentences, while the answers to their questions are provided in a controlled way by the abstract machine. We shall see in [3.4] that those emitters are 'conversational patterns' working under some 'contexts'. The purpose of this first example is not to enter all the details of the formalism at once, we kindly ask the reader to accepts that the participants in conversations will be denoted "(χ_I, s)", where "χ_I" is a 'conversational pattern' and "s" a 'context'. "*" is the undetermined context for participants having not yet entered the conversation; "☺" is a special 'conversational pattern' representing a user emitting initial data before the conversation starts.

Let us consider a short conversation involving 3 *emitters* "(χ_A,*)", "(χ_B,*)" and "(χ_C,*)" with the initial question emitted by a user "(☺,*): a1 // what is the number of primary colours? //

These participants need "models of one another" to ensure a cooperative attitude: this is fulfilled by two different features of the calculus:

i) when entering the conversation each participant emits a set of *commissives* informing the others about what s/he is really going to say; these commissives will be later subject to rewriting;

ii) each participant has a fixed *starter* playing a role in its initial activation, like a pre-condition. This starter is a sentence (either a non-definitive assertion or a non-definitive question) known by the others but not subject to rewriting.

Table 6 gives the initial state of the conversation:

- (☺,*) emits: a1 // what is the number of primary colours? //

- (χ_A,*) emits: b // what are the instances of x? // to indicate future questions

- (χ_A,*) emits: d // the number of instances of x is n // to indicate the further emission of assertions that will (because the associated pattern has been adequately designed) provide an answer. In a similar way:

- (χ_B,*) emits: c // y is an instance of x //

- (χ_C,*) emits: c // y is an instance of x //

We immediately notice that 'c' is emitted both by (χ_B,*) and (χ_C,*), the reader may in first approximation consider that two instances of 'c' are present at the beginning of the conversation. However, this does not mean (χ_B,*) and (χ_C,*) will keep on emitting the same *sentences* during the conversation (as will be shown).

Most of these *utterances* will be replaced during the conversation by more informative ones; this is explicitly mentioned by the property *non-definitive* (boolean = **1**) associated to them.

Table 6 also indicates that some sentences play the role of triggers for participants, which means that whenever they are perceived by a participant, a conversational process starts.

utterance				
sentence	**0 = definitive 1 = non definitive**	**emitted by**	**is a starter for**	**relation to other sentences**
a1 // what is the number of primary colours? //	**0**	(☺ , *)		a1 triggerFor a
a // what is the number of instances of x? //	**1**		χ_A	
b // what are the instances of x? //	**1**	(χ_A, *)	χ_B; χ_C	
c // y is an instance of x //	**1**	(χ_B, *); (χ_C, *)		
d // the number of instances of x is n //	**1**	(χ_A, *)		
b1 // what are the instances of primary colours? //	**0**	(χ_A, a1)		b1 <= b b1 triggerFor b
c1 // red is an instance of primary colours //	**0**	(χ_B, b1)		c1 <= c c1 answerFor b1
c2 // yellow is an instance of primary colours //	**0**	(χ_B, b1); (χ_C, b1)		c2 <= c c2 answerFor b1
c3 // blue is an instance of primary colours //	**0**	(χ_C, b1)		c3 <= c c3 answerFor b1

Table 6

Our calculus relies on the fact that each participant is invoked only when its *starter* (either a question e.g. a// what is the number of instances of x? // or an assertion e.g. a' // here is a set of instances of x //) is triggered by one of the sentences already emitted in the conversation:

- (χ_A,*) starts a conversational process whenever a sentence recognized as a trigger for a // what is the number of instances of x? // appears in the conversation; Table 6 indicates that the initial question a1 // is in such a relation[53] with 'a'.
- (χ_B,*) and (χ_C,*) start a conversational process whenever a sentence recognized as trigger for b // what are the instances of x? // appears in the conversation; Table 6 indicates that the question b1 // emitted by (χ_A, a1) is in such a relation with 'b'.

Then, new *definitive* (boolean = **0**) utterances are emitted during conversation:

- (χ_A, a1), representing (χ_A,*) working in the context "a1", emits: b // what are the instances of primary colours? //
- (χ_B, b1) emits: c1 // red is an instance of primary colours //
- (χ_B, b1) emits: c2 // yellow is an instance of primary colours //
- (χ_C, b1) emits: c2 // yellow is an instance of primary colours //
- (χ_C, b1) emits: c3 // blue is an instance of primary colours //
- (χ_A, a1) emits: d1 // the number of instances of primary colours is 3 //

Table 6 indicates how some sentences play the role of answers for other sentences: 'c1', 'c2' and 'c3' play the role of answers[54] for the question 'b1'.

We can now take a look at Table 7 to understand how the conversational processes occur:

- in a first step, (χ_A, *) is invoked by 'a1' recognized as "triggerFor" its starter 'a', this generates (χ_A, a1) which emits 'b1' and 'd1' respectively more informative than 'b' and 'd' ; 'b' and 'd' are removed. We write "b1 ≤ b" and say "b1 reductionOf b"; one major constraint in our calculus being that participants always "reduce" sentences.

[53] We want this information to be computable instead of needing explicit declaration; therefore we define a "triggerFor" relation between sentences and take hypotheses for the associated test to be decidable.

[54] This point is another major aspect of our calculus, and we shall ask for this information to be computable language instead of needing explicit declaration; therefore we define an "answerFor" relation between sentences and take hypotheses for the associated test to be decidable.

- when (χ_A, a1) emits 'b1' it is recognized as "triggerFor" 'b' which plays the role of *starter* for both (χ_B, *) and (χ_C, *); therefore these patterns are both invoked by 'b1'.
- when (χ_B,*) is invoked by 'b1', it emits 'c1' and 'c2' more informative than 'c' which is removed;

transition in conversation	rewriting action		
	(sentence, boolean)	added by	removed by
invocation of χA by a1	(b1 // what are the instances of primary colours? // , 0)	(χ_A, a1)	
	(b // what are the instances of x? // , 1)		(χ_A, *)
	(d1 // the number of instances of primary colours is n // , 0)	(χ_A, a1)	
	(d // the number of instances of x is n // , 1)		(χ_A, *)
invocation of χB by b1	(c1 // red is an instance of primary colours // , 0)	(χ_B, b1)	
	(c2 // yellow is an instance of primary colours // , 0)	(χ_B, b1)	
	(c // y is an instance of x // , 1)		(χ_B, *)
invocation of χC by b1	(c2 // yellow is an instance of primary colours // , 0)	(χ_C, b1)	
	(c3 // blue is an instance of primary colours // , 0)	(χ_C, b1)	
	(c // y is an instance of x // , 1)		(χ_C, *)
answer {c1, c2, c3} is given to question b1 of χA	(d2 // the number of instances of primary colours is 3 // , 0)	(χ_A, a1)	
	(d1 // the number of instances of primary colours is n // , 1)		(χ_A, a1)

Table 7

- 'c1', 'c2' and 'c3' are recognized as the totality of possible answers for 'b1'; altogether they activate the pendant step of (χ_A, a1);

- when (χ_A, a1) is given 'b1' as an answer to its intermediary question, it replaces 'd1' by the less ambiguous 'd2' and since no other participant can be triggered, the conversation ends.

Let us recall the major points learnt from this first example:

- each conversational pattern owns one *starter*;
- each conversational pattern emits at the beginning of the conversation abstract versions of further utterances under the shape of *commissives*;
- each rewriting consists in reductions (more informative versions) of these abstractions.

So far, we took no interest neither in how (χA,$*$) calculates the number of instances, nor in how (χB,$*$) and (χC,$*$) produce their different answers to the same question … in [3.3], we shall explain how "conversational pattern" are composed of interdependent nodes, the internal functional structure of each node and how they interact.

But we must first formalize sentences and ensure that the key relations "reductionOf", "answerFor" and "triggerFor" are computable!

3.2 Sentences and utterances

3.2.1 Sentences

The conversation in [3.1.4] was ruled on the basis of following assumptions:

> (b1 //what are the instances of primary colours?//) reductionOf (b //what are the instances of x? //)
>
> (c1 //red is an instance of primary colours//) answerFor (b1 //what are the instances of primary colours?//)
>
> (b1 //what are the instances of primary colours?//) triggerFor (b //what are the instances of x?//)

If we want to be able to reason about a calculus deriving these assumptions, we need some hypotheses on the language used for expressing sentences. The strictly necessary hypotheses have been gathered in [Definition1]; we shall exhibit in [3.7] a generative grammar satisfying all of them.

Definition 1: Let $\mathcal{S}$ be an infinite set of well formed formula according to a given generative grammar, let "reductionOf" be a pre order on S denoted "$\leq$", let "triggerFor" and "answerFor" two other relations of $\mathcal{S} \times \mathcal{S}$,

Elements of $\mathcal{S}$ are *sentences*. And we suppose that for each couple (s_1, s_2) of sentences, the following decision problems always have an answer in $\mathcal{S}$:

- problem1: "$s_1 \leq s_2$"
- problem2: "s_1 triggerFor s_2"
- problem3: "s_1 answerFor s_2"
- problem4: "$(\exists\ s_1')\ ((s_1' \leq s_1) \wedge (s_1'$ triggerFor $s_2))$"
- problem5: "$(\exists\ s_1')\ ((s_1' \leq s_1) \wedge (s_1'$ answerFor $s_2))$"
- problem6: "$(\exists\ s_2')\ ((s_1$ answerFor $s_2') \wedge (s_2' \leq s_2))$"

When ($\mathcal{S}$, $\leq$, triggerFor, answerFor) is a set of sentences, and E a subset of $\mathcal{S}$:

- *ReductionsOf*(E) = $\{s \in \mathcal{S} \mid (\exists\ s' \in E)\ s \leq s'\}$
- *TriggersFor*(E) = $\{s \in \mathcal{S} \mid (\exists\ s' \in E)\ s$ triggerFor $s'\}$
- *AnswersForReductionsOf*(E) = $\{s \in \mathcal{S} \mid (\exists\ s_1 \in E)\ (\exists\ s_2 \in \mathcal{S})\ ((s$ answerFor $s_2) \wedge (s_2 \leq s_1))\}$

In the informal examples given before, the relation reductionOf corresponds to what we intuitively describe as "more informative":

- // Linda lives in Paris // reductionOf // a woman lives in a city //
- // where does Linda live? // reductionOf // where does some woman live? //

The relation answerFor corresponds to the usual intuition, also it must be noticed that a long sentence may answer a short question:

- // Linda lives in Paris // answerFor // where does Linda live? //
- // Linda lives in Paris in a beautiful apartment owned by her family since the beginning of the past century // answerFor // where does Linda live? //

The less intuitive relation triggerFor must be understood as: "is a specialized invocation context for":

- // Linda lives in Paris // triggerFor // a woman lives in a city //

answerFor and triggerFor will only be used in tests, while reductionOf will be a constraint on emissions of utterances by the participants in the conversation; these three relations playing three distinct roles in the operational semantics of the calculus are illustrated by comparison algorithms in the generative grammar given in [3.9].

3.2.2 Utterances

Among the infinite possible *sentences*, a few are effectively uttered by "emitters" during the conversation; to express this, we need the notion of *utterance* which embeds a sentence together with additional information. An *utterance* may or not be removed from the conversation: only in the last case will it be considered as a valuable trigger or answer for other utterances! A *definitive* utterance will stay in the conversation until its end, while a *non-definitive* utterance will eventually be removed.

It is possible that two different emitters utter the same sentence, the first as definitive, the second as non-definitive; this gives birth to two different utterances: an utterance is in fact a pair (sentence, status)! In case the same pair (sentence, status) has been uttered concurrently by different emitters, we will consider only one utterance with several

emitters and store their list inside the utterance; this will facilitate the analysis of the set of all sentences present at a given state.

Definition 2: An *utterance* is a pair (sentence, status), where:

- "sentence" is taken in $\mathcal{S}$

- "status" is taken in {0, 1}; status = 0 means *definitive*; status = 1 means *non-definitive*

An *utterance* will be denoted (s, bool).

An *utterance* has a list L of *emitters* which will be further formalized as pairs (*conversational pattern*, context).

$\mathcal{U}(\omega)$ denotes the set of all utterances when conversation has reached a given *state* "ω".

Let "s" be a sentence, Q be a subset of $\mathcal{S}$, the following definitions aim at a concise presentation of the calculus:

- $\mathcal{PotentialTriggersFor}$ (s, ω) = $\{u = (s', 1) \in \mathcal{U}(\omega) \mid (\exists\ s'' \in \mathcal{S})\ ((s'' \leq s') \wedge (s''\ \text{triggerFor}\ s))\}$ is the set of all *non-definitive* utterances which can accept as a reduction a trigger for "s"

- $\mathcal{DefinitiveTriggersFor}$ (s, ω) = $\{u = (s', 0) \in \mathcal{U}(\omega) \mid s'\ \text{triggerFor}\ s\}$ is the set of all *definitive* utterances which embed a trigger for "s"

- $\mathcal{PotentialAnswersFor}$ (Q, ω) = $\{u = (s', 1) \in \mathcal{U}(\omega) \mid (\exists\ s \in Q)\ (\exists\ s'' \in \mathcal{S})\ ((s'' \leq s') \wedge (s''\ \text{answerFor}\ s))\}$ is the set of all *non-definitive* utterances which accept as a reduction an answer for a sentence of Q

- $\mathcal{DefinitiveAnswersFor}$ (Q, ω) = $\{u = (s', 0) \in \mathcal{U}(\omega) \mid (\exists\ s \in Q)\ (s'\ \text{answerFor}\ s)\}$ is the set of all *definitive* utterances which embed an answer for a sentence of Q

Informal examples:

- (// Linda lives in Paris //, 0) ∈ $\mathcal{DefinitiveTriggersFor}$ (// a human lives in a place //)
- (// Linda lives in a city //, 1) ∈ $\mathcal{PotentialTriggersFor}$ (// a human lives in a place //) because this non-definitive utterance may be removed and replaced by (// Linda lives in Paris //, 0)
- (// Linda lives in Paris //, 0) ∈ $\mathcal{DefinitiveAnswersFor}$ (// does Linda lives in Paris? //)
- (// Linda lives in a city //, 1) ∈ $\mathcal{PotentialAnswersFor}$ (// does Linda lives in Paris? //) because this non-definitive utterance may be removed and replaced for instance by (// Linda lives in Paris //, 0) or (// Linda lives in Tokyo //, 0)
- (// a woman lives in Paris //, 1) ∈ $\mathcal{PotentialAnswersFor}$ (// does Linda lives in Paris? //) because this non-definitive utterance may be removed and replaced for instance by (// Linda lives in Paris //, 0) or (// Mathilda lives in Paris //, 0)

3.2.3 Changes of state in the set of utterances

Let us come back to the short conversation described in [3.1.2]

Its initial state is composed by the following utterances, a directive emitted by a "user" and a set of commissives associated to the patterns:

- (// what is the number of primary colours? // , 0, {(☺ , ∗)}) is a definitive question emitted by (☺ , ∗)
- (// what are the instances of x? // , 1, {(χ_A, ∗)}) is a non-definitive question emitted by (χ_A, ∗)
- (// the number of instances of x is n // , 1, {(χ_A, ∗)}) is a non-definitive assertion emitted by (χ_A, ∗)
- (// y is an instance of x // , 1, {(χ_B, ∗), (χ_C, ∗)}) is a non-definitive assertion emitted by (χ_B, ∗) and (χ_C, ∗)

Its final state consists of definitive assertions answering definitive questions[55]:

- (// what is the number of primary colours? // , 0, {(☺ , ∗)}) emitted by (☺ , ∗)
- (// what are the primary colours? // , 0, {(χ_A, a1)}) emitted by (χ_A, a1)
- (//red is an instance of primary colours // , 0, {(χ_B, b1)})
- (// yellow is an instance of primary colours // , 0, {(χ_B, b1), (χ_C, b1)}) emitted by (χ_B, b1) and (χ_C, b1)
- (// blue is an instance of primary colours // , 0, {(χ_C, b1)})
- (// the number of primary colours is 3 // , 0, {(χ_A, a1)}) definitive assertion

Two operations correspond to the basic rewriting actions in the set of utterances: "addition" and "removal, this last one being allowed only for *non-definitive* utterances; *definitive* utterances are kept in the conversation!

Definition 3: Let "e" be an emitter in a conversation, either (χ_I, a) or (χ_I, ∗) or (☺ , ∗) , the two operations on utterances are the following:
- add ((s, bool), e): "e" is added to the list L associated to (s, bool); in case no such utterance already existed, the utterance is "created";
- remove ((s, 1), e): "e" is removed from the list L associated to (s, bool); in case L becomes empty, the utterance is "erased".

[55] The non-definitive assertions and questions "disappear" as soon as their list of emitters becomes empty: according to Definition 2 they are no longer considered as utterances.

3.3 Conversation nodes and conversational patterns

Conversational patterns are intended to be "typical fragments of conversation" able to process and produce information in steps (waiting for available information at each step before processing it and producing new one).

We want them able to process a wide range of input in a functional way, therefore flexible with respect to the potential information provided by the others; in the same time, they must provide a "model of themselves" to the abstract machine and remain predictable.

The moderation of the conversation between patterns relies on two ideas:

- each "unit of information processing" will be constrained in its functional output to utter only reductions of a given set of its own *commissives* (pre-announced sentences);
- a pattern will be a finite graph of such "units" with a causal relation between them.

The first idea will be formalized through the notion of *reducer*; the second one will be the basis for the formal definition of *conversational patterns*.

3.3.1 Conversation nodes are "reducers"

During a conversation, atomic transitions will be supported by functional "nodes" or "information processing units". Basically, a "node" will reduce a set of sentences, when given as input:

- either a trigger (case of a *startNode*)
- or a set of answers (case of an *postNode*)

in the following way: let A, B be subsets of $\mathcal{S}$ respectively corresponding to sentences carrying information issued from the conversation and sentences carrying internal information, a *reducer* will be a $2^A \times 2^B \rightarrow 2^B$ function[56] updating the information in the sentences of B in a functional dependency with the information carried by the sentences of A.

Informal examples:

Example 1:

let A = $\mathcal{TriggersFor}$({// what is the number of instances of x? //}),

let E = {// what are the instances of x? //},

[56] 2^E denotes the set of all parts of E

let φ^1 be a function copying the part of the sentences of A written after "instances of" and substituting it in E after "instances of",
when A' = {// what is the number of instances of European countries? //} $\subset$ A is given as an input, then the node based on (A, φ^1, E) produces φ^1 (A', E) = {// what are the instances of European countries? //} as an output.

Example 2:
let A = $\mathcal{AnswersForReductionsOf}$ ({// what are the instances of primary colours? //}),
let E = {// the number of primary colours is n //},
let φ^2 be a function copying the part of the sentences of A written before "is" , putting them in a set, counting them and substituting the result in E after "is",
when A' = {// yellow is an instance of primary colours //, // red is an instance of primary colours //, // blue is an instance of primary colours //, } $\subset$ A is given as an input, then the node based on (A, φ^2, E) produces φ^2 (A', E) = {// the number of primary colours is 3 // }.

Definition 4a: A *reducer* is a triple $\Phi = (A, \varphi, E)$ defined on $(\mathcal{S}, \leq)$ in the following way:
- $A \in 2^S$ and $E \in 2^S$ are finite sets of sentences[57]
- φ is a function $2^A \times 2^{\mathcal{ReductionsOf}(E)} \rightarrow 2^{\mathcal{ReductionsOf}(E)}$ satisfying:

$\forall A' \subset A, \forall E' \subset \mathcal{ReductionsOf}(E)$: $\varphi(A', E')$ finite and $\varphi(A', E') \subset \mathcal{ReductionsOf}(E')$

$\forall A' \subset A, \forall E' \subset \mathcal{ReductionsOf}(E), \forall E'' \subset \mathcal{ReductionsOf}(E)$:
$E'' \subset E' \Rightarrow \varphi(A', E'') \subset \varphi(A', E')$

$\mathcal{R}$ is the infinite set of possible reducers on $(\mathcal{S}, \leq)$.

Reducers will be shown to be a wide class of functions; we will show in [3.5.7] that any intuitively computable function can be embedded in a reducer, provided the fact that an adequate $(\mathcal{S}, \leq)$ is defined.

Definition 4b:

A *node* is a tuple $\nu = (A, \varphi, \varphi', E, F)$ where:
- $A \in 2^S$,
- $E \in 2^S$ and $F \in 2^S$ are disjoint finite sets of sentences
- $\Phi = (A, \varphi, E)$ and $\Phi' = (A, \varphi', F)$ are two reducers sharing the same domain A
- the sentences of E will become *definitive* whenever φ is activated
- the sentences of F will stay *non-definitive* whenever φ' is activated

[57] 2^S denotes the set of all parts of S

Only two cases will be considered:

- A = *TriggersFor*({s}) = {x ∈ $\mathcal{S}$ | x triggerFor s} is the set of all sentences in the relation triggerFor with a given sentence 's' ; in this first case, the node will be called a *startNode* and will be denoted (*Trig* (s), φ, φ', E, F)
- A = *AnswersForReductionsOf* (N) = {s ∈ $\mathcal{S}$ | (∃ s_1∈N) (∃ s_2∈$\mathcal{S}$) ((s answerFor s_2) ∧ ($s_2 \leq$ s1))} for a given set 'N' of sentences; in this second case, the node will be called a *postNode* and denoted (*Ans* (N), φ, φ', E, F)

We shall keep the denotation (A, φ, φ', E, F) only when we do not want to specialize a case among those two.

During a conversation, the activation of the reducers of the node 'ν' will produce the emission of two types of utterances:

- the utterances of E will be reduced into *definitive* utterances u = (s, 0, L) where s ∈ *ReductionsOf* (E)
- the utterances of F will be reduced into *non-definitive* utterances u' = (s', 1, L') where s'∈ *ReductionsOf* (F)

Definitive utterances will play a decisive role when ruling the calculus, the need for distinguishing two reducers inside a node reflects the necessity of keeping them apart.

Let us consider again χ_A defined in [3.1.4]:

its first *node* is the *startNode* ν^0 = (*Trig* (a), φ, φ', E_A^0, F_A^0) with:

- a // what is the number of instances of x? //
- E_A^0 = {b // what are the instances of x? //}
- F_A^0 = {d // the number of instances of x is n //}

its second *node* is the *postNode* ν^1 = (*Ans* (Q^1), φ, φ', E_A^1, F_A^1)

- Q^1 = {b // what are the instances of x? // }; $Q^1 = E_A^0$
- E_A^1 = {d // the number of instances of x is n // }; $E_A^1 \subset F_A^0$
- $F_A^1 = \emptyset$

When χ_A is triggered by: // a1 // what is the number of primary colours? //:

- Q^1 becomes {// b1 // what are the instances of primary colours? //}; // b1 // ∈ *ReductionsOf* (Q^1) ⊂ *ReductionsOf* (E_A^0)
- E_A^1 becomes {// d1 // the number of instances of primary colours is n //}; // d1 // ∈ *ReductionsOf* (E_A^1) ⊂ *ReductionsOf* (F_A^0)

When A'= {c1, c2, c3} ⊂ *AnswersForReductionsOf* (Q) is given as an answer to b1, the reducer (A, φ, E_1) is activated:

- E_A^1 becomes {d2 // the number of instances of primary colours is 3 //}

3.3.2 Conversational patterns are semi-lattices of conversation nodes

The notion of conversational pattern aims at "packaging" a set of conversation nodes in such a way that they produce integrated results depending on questions that cannot be all asked at once.

In this subsection, we take for example "buying a train ticket", as a conversation between a service provider selling train tickets, and "John"; the focus is put on the (rather dry) conversational pattern "χ" followed by the service provider, which is composed of "5 nodes":

The first *Node* is the *startNode* α^0 which emits questions concerning the description of the trip: where from (b)? to where (c)? When (d)? but also questions concerning a priori preferences: smoker or not (e)? special conditions (f)?

transition in conversation	action on utterances		
	(sentence, boolean)	added by	removed by
invocation of α0 by a1	(b1 // where does John start from? //) , 0)	(χ, a1)	
	(c1 // where does John go? // , 0)	(χ, a1)	
	(d1 // when does John leave? // , 0)	(χ, a1)	
	(e1 // is John a smoker? // , 0)	(χ, a1)	
	(f1 // does John benefit from special conditions? // , 0)	(χ, a1)	
	(g1 // is train x OK for John? // , 1)	(χ, a1)	
	(h1 // is "train x & condition y" OK for John? // , 1)	(χ, a1)	
	(k1 // does John confirm "train x & condition y"? // , 1)	(χ, a1)	
	(l1 // "train x & condition y" has been reserved for John // , 1)	(χ, a1)	
activation of β1: answers received for Q1 = {b1, c1, d1}	(g21 // is train 5577 OK for John? // , 0)	(χ, a1)	
	(g22 // is train 5878 OK for John? // , 0)	(χ, a1)	
	(g23 // is train 9867 OK for John? // , 0)	(χ, a1)	
	(g1 // is train x OK for John? // , 1)		(χ, a1)
activation of β2 : answers received for Q2 = {e1, f1, g21, g22, g23}	(h21 // is "train 5577& optiPrice" OK for John? // , 0)	(χ, a1)	
	(h22 // is "train 5878 & miniPrice" OK for John? // , 0)	(χ, a1)	
	(h1 // is "train x & condition y" OK for John? // , 1)		(χ, a1)
activation of β3 : answers received for Q3 = {h21, h22}	(k2 // does John confirm "train 5878 & miniPrice"? // , 0)	(χ, a1)	
	(k1 // does John confirm "train x & condition y"? // , 1)		(χ, a1)
activation of β4 : answers received for Q4 = {k2}	(l2 // "train 5878 & miniPrice" has been reserved for John // , 0)	(χ, a1)	
	(l1 // "train x & condition y" has been reserved for John // , 1)		(χ, a1)

Table 8

utterance				
sentence	0 = def. 1 = non def.	emitted by	is a starter for	relation to other sentences
a // z would like a train ticket //			χ	
a1 // John would like a train ticket //	0	John		a1 triggerFor a
b // where does z start from? //	1	(χ, *)		
c // where does z go? //	1	(χ, *)		
d // when does z leave? //	1	(χ, *)		
e // is z a smoker? //	1	(χ, *)		
f // does z benefit from special conditions? //	1	(χ, *)		
g // is train x OK for z? //	1	(χ, *)		
h // is "train x & condition y" OK for z? //	1	(χ, *)		
k // does z confirm "train x & condition y"? //	1	(χ, *)		
l // "train x & condition y" has been reserved for z //	1	(χ, *)		
b1 // where does John start from? //	0	(χ, a1)		b1 <= b
c1 // where does John go? //	0	(χ, a1)		c1 <= c
d1 // when does John leave? //	0	(χ, a1)		d1 <= d
e1 // is John a smoker? //	0	(χ, a1)		e1 <= e
f1 // does John benefit from special conditions? //	0	(χ, a1)		f1 <= f
g1 // is train x OK for John? //	1	(χ, a1)		g1 <= g
h1 // is "train x & condition y" OK for John? //	1	(χ, a1)		h1 <= h
k1 // does John confirm "train x & condition y"? //	1	(χ, a1)		k1 <= k
l1 // "train x & condition y" has been reserved for John //	1	(χ, a1)		l1 <= l
b10 // John starts from Sofia-Antipolis //	0	John		b10 answerFor b1
c10 // John goes to Paris //	0	John		c10 answerFor c1
d10 // John leaves tomorrow //	0	John		d10 answerFor d1
e10 // John is NOT a smoker //	0	John		e10 answerFor e1
f10 // John is a "mature PhD student" //	0	John		f10 answerFor f1
g21 // is train 5577 OK for John? //	0	(χ, a1)		g21 <= g1
g22 // is train 5878 OK for John? //	0	(χ, a1)		g22 <= g1
g23 // is train 9867 OK for John? //	0	(χ, a1)		g23 <= g1
g210 // train 5577 is OK for John //	0	John		g210 answerFor g21
g220 // train 5878 is OK for John //	0	John		g220 answerFor g22
g230 // train 9867 is NOT OK for John! //	0	John		g230 answerFor g23
h21 // is "train 5577& optiPrice" OK for John? //	0	(χ, a1)		h21 <= h1
h22 // is "train 5878 & miniPrice" OK for John? //	0	(χ, a1)		h22 <= h1
h210 // "train 5577& optiPrice" is NOT OK for John //	0	John		h210 answerFor h21
h220 // "train 5878, & miniPrice" is OK for John //	0	John		h220 answerFor h22
k2 // does John confirm "train 5878 & miniPrice"? //	0	(χ, a1)		k2 <= k1
k20 // John confirms "train 5878 & miniPrice" //	0	John		k20 answerFor k2
l2 // "train 5878 & miniPrice" has been reserved for John //	0	(χ, a1)		l2 <= l1

Table 9

Then comes the *postNode* β^1 which depends on answers about the description of the trip, which will determine the different available trains; β^1 presents a list of available trains and asks the client John to select the convenient trains (g21, g22, g23).
Then comes the node β^2 which depends both on answers concerning the convenient trains and on answers concerning a priori preferences; β^2 presents a list of pairs (train, condition) according to the possibilities offered by the convenient trains, and asks for selection (h21, h22).
To end with, nodes β^3 and β^4 follow sequentially in order to finalize a reservation (k2), (l2).

Table 8 shows how the conversational pattern of the provider drives the conversation. Table 9 shows the utterances of this conversation, either *definitive* (status=0) or *non-definitive* (status=1).

Figure 21 is a graphical representation of the conversational pattern χ; α^0 and the β^i where $i \in \{1, 2, 3, 4\}$ are nodes of an oriented graph. The arrows indicate a relation "drives" between nodes; "ν^1 drives ν^2" $\Leftrightarrow$ "ν^1 emits questions of which the answers are needed by ν^2". The reader may verify that each step has the appropriate "drivers" (there is no needed question which would never have been emitted).

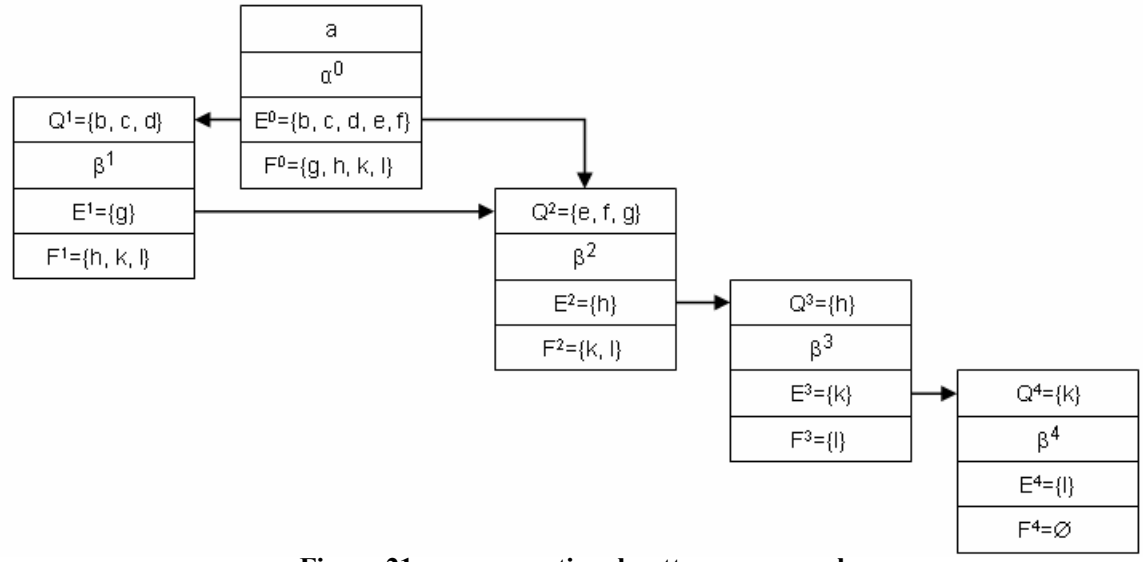


Figure 21: a conversational pattern as a graph

We are going to formalize two relations on nodes; the first one is "drives", the second one is aimed at manipulating transitive graphs and has been named "precedes" on purpose: a node "ν^1" emitting questions of which the answers are indirectly needed by "ν^2" will be activated before "ν^2".

Definition 4c: Let $\nu^1 = (A^1, \varphi^1, \varphi'^1, E^1, F^1)$ be a *node* , let $\nu^2 = (\mathcal{Ans}(N^2), \varphi^2, \varphi'^2, E^2, F^2)$ be a *postNode:*

ν^1 drives $\nu^2 \Leftrightarrow E^1 \cap N^2 \neq \varnothing$

We shall indifferently say ν^1 drives ν^2, or ν^1 is one of the *drivers* of ν^2.

ν^1 precedes $y \Leftrightarrow (\nu^1 = \nu^2) \vee (\exists$ *nodes* $\nu'^1, ..., \nu'^n)$ $((\nu^1$ drives $\nu'^1) \wedge \ldots \wedge (\nu'^n$ drives $\nu^2)$

On the above figure:

- α^0 drives β^1 and β^2 ; α^0 precedes $\beta^1, \beta^2, \beta^3, \beta^4$
- β^1 drives β^2 ; β^1 precedes $\beta^2, \beta^3, \beta^4$
- β^2 drives β^3 ; β^2 precedes β^3, β^4
- β^3 drives β^4; β^3 precedes β^4

Definition 4d: A *conversational pattern* χ is a finite set of *nodes* $\{\chi^i\}_{i\geq 0}$ where:
- $\chi^0 = (\mathcal{Trig}(s), \varphi^0, \varphi'^0, E^0, F^0)$ is a *startNode,*
- $\chi^i = (\mathcal{Ans}(Q^i), \varphi^i, \varphi'^i, E^i, F^i)$ where $i \geq 1$ are *postNodes*,

in such a way that the following "good properties" are verified:

a) "precedes" is a partial order on χ (no loops are allowed in the graph) which makes $(\chi,$ precedes$)$ a meet-semilattice with χ^0 as first/least element. Inside a conversational pattern, we shall indifferently say χ^1 precedes χ^2, or χ^2 is one of the *followers* of χ^1 or χ^1 is one of the *predecessors* of χ^2.

b) $\forall\ i\geq 0, j\geq 0 : \mathcal{ReductionsOf}(E^i) \cap \mathcal{ReductionsOf}(E^j) = \varnothing$

[definitive reductions are distributed in a conversational pattern in such a way that a given sentence is definitively reduced by only one node]

c) let $\chi^i = (A^i, \varphi^i, \varphi'^i, E^i, F^i)$ be a *node*,
let $\mathcal{F} = \{\chi^f = (\mathcal{Ans}(Q^f), \varphi^f, \varphi'^f, E^f, F^f) \mid \chi^f \in \chi \wedge \chi^i$ precedes $\chi^f\}$ be the set of its *followers*
then: $F^i = \cup_{f\in\mathcal{F}} (E^f \cup F^f)$

[non-definitive reductions are distributed in a conversational pattern in such a way that a given node reduces a given sentence only if one of its *followers* also reduces it (either in a definitive way or not)]

d) let $\chi^i = (\mathcal{Ans}(Q^i), \varphi^i, \varphi'^i, E^i, F^i)$ be a *postNode*,
let $\mathcal{D} = \{\chi^d = (A^d, \varphi^d, \varphi'^d, E^d, F^d) \mid \chi^d \in \chi \wedge \chi^d$ drives $\chi^i\}$ be the set its *drivers* inside χ
then: $Q^i = \cup_{d\in\mathcal{D}} (Q^i \cap E^d)$

[the *drivers* of a given node operate a partition on Q, i.e. all preliminary questions will be made definitive by (exactly) one "driver"]

The *starter* "s" of the *startNode* will also be called the *starter* of the whole conversational pattern.

The *nodes* of a *conversational pattern* χ are called *conversation nodes* and will always be denoted χ^i where χ^0 is the *startNode* and the other χ^i are the *postNodes*; this concise denotation replaces a more rigorous one based on the name of the pattern and index of the node.

The following definition shows that nodes can operate on each other inside conversational patterns; this will allow us to express the transitions of the calculus in a rather concise way; the reader might derive from the definition of "|" that the use of some higher level mathematical objects might be more elegant … this mathematical aspect of the formalization is still "under construction".

Definition 4e:
Let $\chi^1 = (A^1, \varphi^1, \varphi'^1, E^1, F^1)$ be a *conversation node*, let $A'^1 \subset A^1$,
let $\chi^2 = (\mathcal{A}ns(N^2), \varphi^2, \varphi'^2, E^2, F^2)$ be a *postNode* such that: $E^2 \subset F^1$, $F^2 \subset F^1$,
let N^{rest} be the rest of N^2 after intersection with $(E^1 \cup F^1)$:
$N^2 = (N^2 \cap (E^1 \cup F^1)) \cup N^{rest}$ and $N^{rest} \cap (E^1 \cup F^1) = \emptyset$,
Then we may denote and define the "reduction of χ^2 by $\chi^1(A'^1)$)" in the following way:

$$(\chi^2 \mid \chi^1(A'^1)) = (\mathcal{A}ns(N'^2), \varphi^2, \varphi'^2, E'^2, F'^2)$$

$(\chi^2 \mid \chi^1(A'^1))$ is a *postNode* characterized by:
$N'^2 = \varphi^1(A'^1, N^2 \cap E^1) \cup \varphi'^1(A'^1, N^2 \cap F^1) \cup N^{rest}$,
$E'^2 = \varphi'^1(A'^1, E^2)$, $F'^2 = \varphi'^1(A'^1, F^2)$
We shall denote:

Demonstration: in order to prove that $(\chi^2 \mid \chi^1(A'^1))$ is a correct *node*, we have to establish that $(\mathcal{A}ns(N'^2), \varphi^2, E'^2)$ and $(\mathcal{A}ns(N'^2), \varphi'^2, F'^2)$ are reducers; it follows from:
i) $N'^2 \subset \mathcal{R}eductionsOf(N^2) \Rightarrow \mathcal{A}nswersForReductionsOf(N'^2) \subset \mathcal{A}nswersForReductionsOf(N^2)$
ii) $E'^2 \subset \mathcal{R}eductionsOf(E^2) \Rightarrow \mathcal{R}eductionsOf(E'^2) \subset \mathcal{R}eductionsOf(E^2)$
iii) $F'^2 \subset \mathcal{R}eductionsOf(F^2) \Rightarrow \mathcal{R}eductionsOf(F'^2) \subset \mathcal{R}eductionsOf(F^2)$

If χ^1 precedes χ^2 precedes χ^3 in a given conversational pattern, we want to be allowed to to successively transform χ^3 according to the knowledge provided by χ^1 and then by χ^2. In other words, we need to reiterate "|" inside a given conversational pattern in successive waves corresponding to the successive activations of the nodes: $((\chi^3 \mid \chi^1) \mid (\chi^2 \mid \chi^1))$

expresses[58] that χ^2 and χ^3 have first been reduced by χ^1 into χ'^2 and χ'^3; then χ'^3 has been re-reduced by χ'^2 .

Lemma1 establishes this important property.

Lemma1:
Let $\chi^1 = (A^1, \varphi^1, \varphi'^1, E^1, F^1)$ be a *conversation node*, let $A'^1 \subset A^1$,
let $\chi^2 = (A^2, \varphi^2, \varphi'^2, E^2, F^2)$ be a *postNode* such that: $E^2 \subset F^1$, $F^2 \subset F^1$,
let $\chi^3 = (A^3, \varphi^3, \varphi'^3, E^3, F^3)$ be a *postNode* such that: $E^3 \subset F^2 \subset F^1$, $F^3 \subset F^2 \subset F^1$,
let $\chi'^2 = (\chi^2 \mid \chi^1(A'^1)) = (A'^2, \varphi^2, \varphi'^2, E'^2, F'^2)$, let $A''^2 \subset A'^2$,
let $\chi'^3 = (\chi^3 \mid \chi^1(A'^1)) = (A'^3, \varphi^3, \varphi'^3, E'^3, F'^3)$,

Then:

$E'^3 \subset F''^2$, $F'^3 \subset F'^2$

and therefore we can define: $(\chi'^3 \mid \chi'^2 (A''^2))$
Demonstration: thanks to Definition 4a
i) $E^3 \subset F^2 \Rightarrow (E'^3 = \varphi'^1(A'^1, E^3)) \subset (F'^2 = \varphi'^1(A'^1, F^2))$
ii) $F^3 \subset F^2 \Rightarrow (F'^3 = \varphi'^1(A'^1, F^3)) \subset (F'^2 = \varphi'^1(A'^1, F^2))$

[58] the correct writing of this is (((χ 3 | χ 1(A)) | (χ 2 | χ 1(A)) (B))) as shown in Lemma1

3.4 Conversations and atomic transitions

3.4.1 Steps in conversations

We are going now to define a *conversation* as the initial state of a process involving several *conversational patterns* confronted with a set of initial utterances. During a *run* of a conversation, each of those patterns may be invoked several times. In subsection [3.2], we have defined *utterances* embedding *sentences*, we now define *conversation steps* embedding *conversation nodes* in order to trace their successive activations and reductions. A *conversation step* is an aggregate consisting of:

- a *conversation node*;
- a context of activation;
- a status indicating whether the node associated to the step has reached a definitive state or not.

Definition 5a: A *conversation step* is a "triple" (node, context, status), where:

- "node" is the reference to a *conversation node*, and therefore is always associated to a conversational pattern; χ_P^i denotes node "i" of *conversational pattern* χ_P.

- "context" is either a sentence or "∗"; "context = a" refers to the invocation by "a" of the associated *conversational pattern*; context = "∗"means "undetermined" and applies only to *startSteps* ready for invocation.

- "status" is taken in {0, 1}; status = 0 means *definitive*; status = 1 means "*non-definitive*".

Therefore we shall distinguish 3 kinds of steps:

- *conversation steps* of the type (*node*, a, 0) having reached a definitive state and no longer participating to the conversation;
- *conversation steps* of the type (*startNode*, ∗, 1) called *startSteps*;

- *conversation steps* of the type (*postNode*, a, 1) called *postSteps*; we shall show that they necessarily follow the invocation of a startStep.

$\mathcal{X}(\omega)$ denotes the set of all current steps; as for utterances, we have two basic operations on steps:

- **add** (step)

- **remove** (step), which is allowed only for *non-definitive* steps

Definition 5b: An *emitter* is a pair (pattern, context), where:

- "pattern" is either a *conversational pattern* or " ☺ " representing a "user" emitting sentences before the conversation starts

- "context" is either a sentence or "∗"; "context = a" refers to the invocation by "a" of the associated *conversational pattern*; context = "∗" means "undetermined"

Since " ☺ " will never been invoked during the conversation, it will always be associated to "∗" in "initial emitter (☺ , ∗)".

We are now ready for a definition of a conversation:

Definition 5c: Let $\mathcal{P}atterns$ be a finite set of indices,

A *conversation frame* X is a chosen set of *conversational patterns* $\{\chi_I\}_{I\in\mathcal{P}atterns}$.

A *conversation initial state* is a pair $\omega_0 = (\mathcal{U}(X, ☺, \omega_0), \mathcal{X}(X, \omega_0))$ where:

- $\mathcal{X}(X, \omega_0)$ is the set $\{x_I^0 = (\chi_I^0, *, 1)\}_{I\in\mathcal{P}atterns}$ of *startSteps* associated to the *startNodes* $\{\chi_I^0 = (\mathcal{T}rig(s_I), \varphi_I^0, \varphi'^0_I, E_I^0, F_I^0)\}_{I\in\mathcal{P}atterns}$ belonging to the *conversational patterns* of X

- $\mathcal{U}(☺)$ is a set of initial definitive utterances (s, 0, {(☺ , ∗)}) emitted by a "user".

- $\mathcal{U}(X, ☺, \omega_0)$ = **add**$(\mathcal{U}(☺), \mathcal{U}(X, \omega_0))$ is obtained through the successive operations {**add** ((s, 0), (☺ , ∗)} and {**add** ((s, 1), $(\chi_I, *)$)}$_{I\in\mathcal{P}atterns,\ s\in E_I^\circ\cup F_I^\circ}$

A *conversation state* is a pair $\omega = (\mathcal{U}(\omega), \mathcal{X}(\omega))$, where $\mathcal{U}(\omega)$ is the set of all current utterances, and $\mathcal{X}(\omega)$ is the set of all current steps.

A *conversation run* is the process resulting of the application of the "rules of conversational calculus[59]" starting from a *conversation initial state*; this process goes through different *conversation states.*

An *atomic transition* is a non-interruptible change from a *state* $\omega = (\mathcal{U}(\omega), \mathcal{X}(\omega))$ to another *state* $\omega' = (\mathcal{U}(\omega'), \mathcal{X}(\omega'))$; no states are allowed to occur between ω and ω'.

[59] please refer to [3.5]

3.4.2 Invocation, activation and revocation of steps

Definition 6a:

Let ω_0 be a *conversation initial state* where $\chi_P \in \mathcal{X}(\omega_0)$ is a *conversational pattern* $\{\chi_P^i\}_{i\geq 0}$,
let $x_P^0 = (\chi_P^0, *, 1)$ be the *startStep* of χ_P where $\chi_P^0 = (\mathcal{T}rig(s_P), \varphi^0, \varphi'^0, E_P^0, F_P^0)$,
let $\omega = (\mathcal{U}(\omega), \mathcal{X}(\omega))$ be the current state the conversation,
let "a" $\in \mathcal{T}riggersFor(s_P)$,

invoke (x_P^0, a) is the *atomic transition* from state ω to state ω' consisting in the following actions:

1. for each "s" in $\varphi'^0(\{a\}, E_P^0)$: **add** $((s, 0), (\chi_P, a))$ to $\mathcal{U}(\omega)$
2. for each "s" in $\varphi'^0(\{a\}, F_P^0)$: **add** $((s, 1), (\chi_P, a))$ to $\mathcal{U}(\omega)$
3. let $x_P^0(a) = (\chi_P^0, a, 0)$: **add** $(x_P^0(a))$ to $\mathcal{X}(\omega)$
4. for each χ_P^i *follower* of χ_P^0, let $x_P^i(a, \omega') = (\chi_P^i(\omega'), a, 1)$: **add** $(x_P^i(a, \omega'))$ to $\mathcal{X}(\omega)$, where $\chi_P^i(\omega') = (\chi_P^i \mid \chi_P^0(\{a\}))$ is the "reduction" of χ_P^i by $\chi_P^0(\{a\})$.

We shall indifferently say that the startStep has been invoked by "a", or that the conversational pattern has been invoked by "a".

1. and 2. are emission of utterances corresponding to reductions of the utterances of E_P^0 and F_P^0 while taking into account the context "a"; these utterances have been emitted by the *invocation* (χ_P, a).

3. is the indication that startStep x_P^0 has been activated under context "a" and has reached its definitive state.

4. is the reduction of all other steps (followers of x_P^0) by startStep x_P^0 so that they incorporate the information provided by "a".

Definition 6b:

Let ω_0 be a *conversation initial state* where $\chi_P \in \mathcal{X}(\omega_0)$ is a *conversational pattern* $\{\chi_P^i\}_{i\geq 0}$,
let $\omega = (\mathcal{U}(\omega), \mathcal{X}(\omega))$ be the current state the conversation,
let $x_P^1(a, \omega) = (\chi_P^1(\omega), a, 1)$ be the *conversation step* of χ_P associated to the context "a" and to the *postNode* $\chi_P^1(\omega)$ in the current state $(\mathit{Ans}(N^1), \varphi^1, \varphi'^1, E_P^1, F_P^1)$
let $A^1 \subset \mathit{AnswersForReductionsOf}(N^1)$,

activate $(x_P^1(a, \omega), A^1)$ is the *atomic transition* from state ω to state ω' consisting in the following actions:

1. for each "s" in $E_P^1 \cup F_P^1$: **remove** $((s, 1), (\chi_P, a))$ from $\mathcal{U}(\omega)$
2. for each "s" in $\varphi^1(A^1, E_P^1)$: **add** $((s, 0), (\chi_P, a))$ to $\mathcal{U}(\omega)$
3. for each "s" in $\varphi'^1(A^1, F_P^1)$: **add** $((s, 1), (\chi_P, a))$ to $\mathcal{U}(\omega)$
4. **remove** $(x_P^1(a, \omega))$ from $\mathcal{X}(\omega)$
5. let $x_P^1(a, \omega') = (\chi_P^1(\omega), a, 0)$: **add** $(x_P^1(a, \omega'))$ to $\mathcal{X}(\omega)$
6. for each χ_P^i *follower* of χ_P^1:
   let $x_P^i(a, \omega) = (\chi_P^i(\omega), a, 1)$: **remove** $(x_P^i(a, \omega))$ from $\mathcal{X}(\omega)$
   let $x_P^i(a, \omega') = (\chi_P^i(\omega'), a, 1)$: **add** $(x_P^i(a, \omega'))$ to $\mathcal{X}(\omega)$,
   where $\chi_P^i(\omega') = (\chi_P^i(\omega) \mid \chi_P^1(\omega)(A^1))$ is the "reduction" of $\chi_P^i(\omega)$ by $\chi_P^1(\omega)(A^1)$
   [this is allowed thanks to Lemma1].

1. 2. and 3. are actions of utterances corresponding to reductions of the utterances of E_P^1 and F_P^1 while taking into account the answers "A^1"; these utterances have been removed or emitted by the *invocation* (χ_P, a).

4. and 5. indicate that postStep x_P^1 has been activated under context "a" and has reached its definitive state.

6. is the reduction of all other steps (followers of x_P^1) by postStep x_P^1 so that they incorporate the information provided by "A^1".

Definition 6c:

Let ω_0 be a *conversation initial state* where $\chi_P \in \mathcal{X}(\omega_0)$ is a *conversational pattern* $\{\chi_P^i\}_{i\geq 0}$,
let $x_P^0 = (\chi_P^0, *, 1)$ be the *startStep* of χ_P where $\chi_P^0 = (\mathit{Trig}(s_P), \varphi^0, \varphi'^0, E_P^0, F_P^0)$,
let $\omega = (\mathcal{U}(\omega), \mathcal{X}(\omega))$ be the current state the conversation,

revoke (x_P^0) is the *atomic transition* from state ω to state ω' consisting in the following actions:

1. for each "s" in $E_I^0 \cup F_I^0$ **remove** $((s, 1), (\chi_P, *))$ from $\mathcal{U}(\omega)$
2. **remove** (x_P^0) from $\mathcal{X}(\omega)$

3.4.3 Another example: buying a train ticket

Let us come back to the example given in [3.3.2], where a service provider follows a "5 nodes" conversational pattern χ.

A *conversation step* is either a *startStep* or a *postStep*; in the first case, it is ready to be activated as soon as a trigger appears. In the second case it will be activated when two conditions are met i) all preliminary questions have reached their *definitive* expression ii) these definitive questions have found an answer. Whereas the second condition is external (all steps of all conversational patterns may potentially give answers), the first condition is internal (only predecessors of a step inside a conversational pattern are allowed to reduce preliminary questions); when all preliminary questions have reached their *definitive* expression, we shall say that the postStep is READY TO BE ACTIVATED[60].

Therefore, although we know from Definition 5a that a conversation step has only two possible values for status "0" or "1", we are going to use 3 colours in order to make the example easier to follow:

"DARK GREY WITH TEXT IN WHITE" will mean: [status = "0": the step has been ACTIVATED],

"WHITE" will mean: [status = "1" and step NOT READY TO BE ACTIVATED, because preliminary questions have not all been reduced].

"LIGHT GREY" will mean: [status = "1" and step READY TO BE ACTIVATED, because preliminary questions have all been reduced].

Each of the states "ω"of the conversation between service provider and client is described in the following way:

- $\mathcal{X}(\omega)$ is described only by showing the nodes of χ
- $\mathcal{U}(\omega)$ is described only by mentioning the changes having occurred to the precedent state

The initial state of the conversation is $\omega 0 = (\mathcal{U}(\omega_0), \mathcal{X}(\omega_0))$ where $\mathcal{X}(\omega_0)$ contains only the *startStep* $(\alpha^0, *, 1)$ of χ ready to be activated, as illustrated in Figure 22:

[60] All this will be formalized in [3.5]

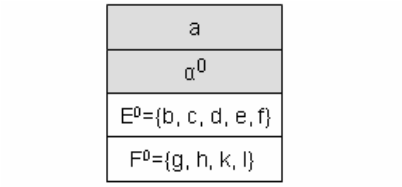


Figure 22: buying a train ticket $\mathcal{X}(\omega_0)$

$\mathcal{U}(\omega_0)$ first contains a1 emitted by (☺ , ∗) as well as the initial non-definitive sentences emitted by (χ, ∗):

sentence	**0 = def. 1 = non def.**	**emitted by**
a1 // John would like a train ticket //	**0**	John
b // where does z start from? //	**1**	(χ, *)
c // where does z go? //	**1**	(χ, *)
d // when does z leave? //	**1**	(χ, *)
e // is z a smoker? //	**1**	(χ, *)
f // does z benefit from special conditions? //	**1**	(χ, *)
g // is train x OK for z? //	**1**	(χ, *)
h // is "train x & condition y" OK for z? //	**1**	(χ, *)
k // does z confirm "train x & condition y"? //	**1**	(χ, *)
l // "train x & condition y" has been reserved for z //	**1**	(χ, *)

The first possible transition is: $\omega_0 \rightarrow$ **invoke** $((\alpha^0, *, 1), a1) \rightarrow \omega_1$

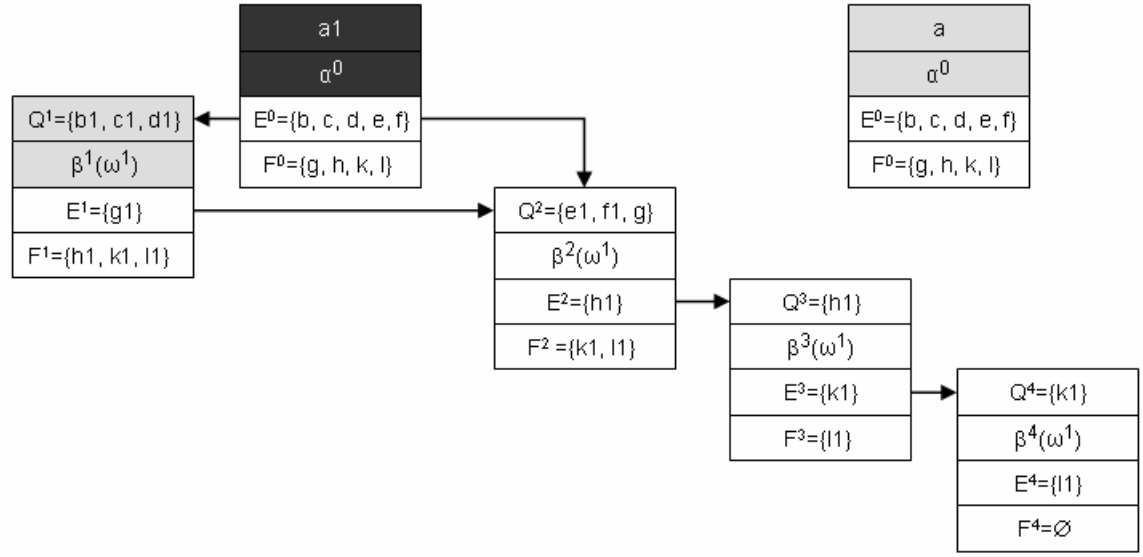


Figure 23: buying a train ticket $\mathcal{X}(\omega_1)$

$\mathcal{X}(\omega_1)$ in Figure 23 still contains $(\alpha^0, *, 1)$, and has been augmented by the activated step associated to the invocation by a1 of α^0 and all the following steps associated to the β^i.

We can see that all the followers of α^0 have been reduced and have therefore reached the respective states $\beta^i(\omega_1)$.

We notice that only $\beta^1(\omega_1)$ is ready to be activated, and this is because α^0 was its only driver. (From Definition 4d/d we know that each preliminary question of a given node will be reduced by one of its drivers, and only this one)

$\mathcal{U}(\omega_1)$ is obtained from $\mathcal{U}(\omega_0)$ by the following actions of $(\chi, a1)$: addition of

sentence	0 = def. 1 = non def.	emitted by
b1 // where does John start from? //	0	$(\chi, a1)$
c1 // where does John go? //	0	$(\chi, a1)$
d1 // when does John leave? //	0	$(\chi, a1)$
e1 // is John a smoker? //	0	$(\chi, a1)$
f1 // does John benefit from special conditions? //	0	$(\chi, a1)$
g1 // is train x OK for John? //	1	$(\chi, a1)$
h1 // is "train x & condition y" OK for John? //	1	$(\chi, a1)$
k1 // does John confirm "train x & condition y"? //	1	$(\chi, a1)$
l1 // "train x & condition y" has been reserved for John //	1	$(\chi, a1)$

Then we have two possible transitions: revocation of $((\alpha^0, *, 1)$ or activation of $\beta^1(\omega_1)$.

- Revocation of $((\alpha^0, *, 1)$ is possible because no other sentences like // a1 // neither are present nor might appear.
- Activation of β^1 is possible because we suppose John has answered and we also suppose nobody else will ever give answers to β^1's preliminary questions. We are going to suppose in this *run of the conversation* that revocation of $((\alpha^0, *, 1)$ happens first: $\omega_1 \rightarrow$ **revoke** $((\alpha^0, *, 1)) \rightarrow \omega_2$

$\mathcal{X}(\omega_2)$ in Figure 24 depicts the loss of $(\alpha^0, *, 1)$, with no consequence on the other steps.

$\mathcal{U}(\omega_2)$ is obtained from $\mathcal{U}(\omega_1)$ by the following actions of $(\chi, *)$: removal of a, b, c, d, e, f, g, h, k, l.

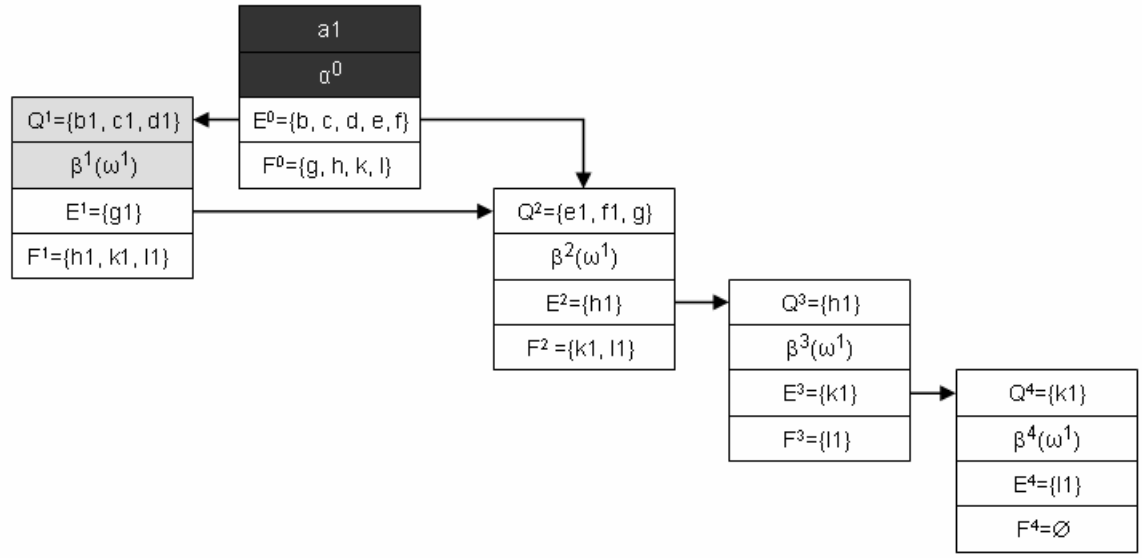


Figure 24: buying a train ticket $\chi(\omega_2)$

Next transition is: $\omega_2 \rightarrow$ **activate** (($\beta^1(\omega_1)$, a1, 1), "answers from John") $\rightarrow \omega_3$

$\chi(\omega_3)$ in Figure 25 depicts the reduction of all β^1's followers. It appears that the reducer φ'^1 has produced Ø, whereas φ^1 has done a great job.

We notice that $\beta^2(\omega_1)$ is now ready to be activated, since its two drivers have now been activated. Part of its preliminary questions (e1, f1: the ones concerning a priori preferences) may have been answered as soon as α^0 has been activated, but the other ones (/g21, g22, g23: the ones concerning convenient trains) have just been asked.

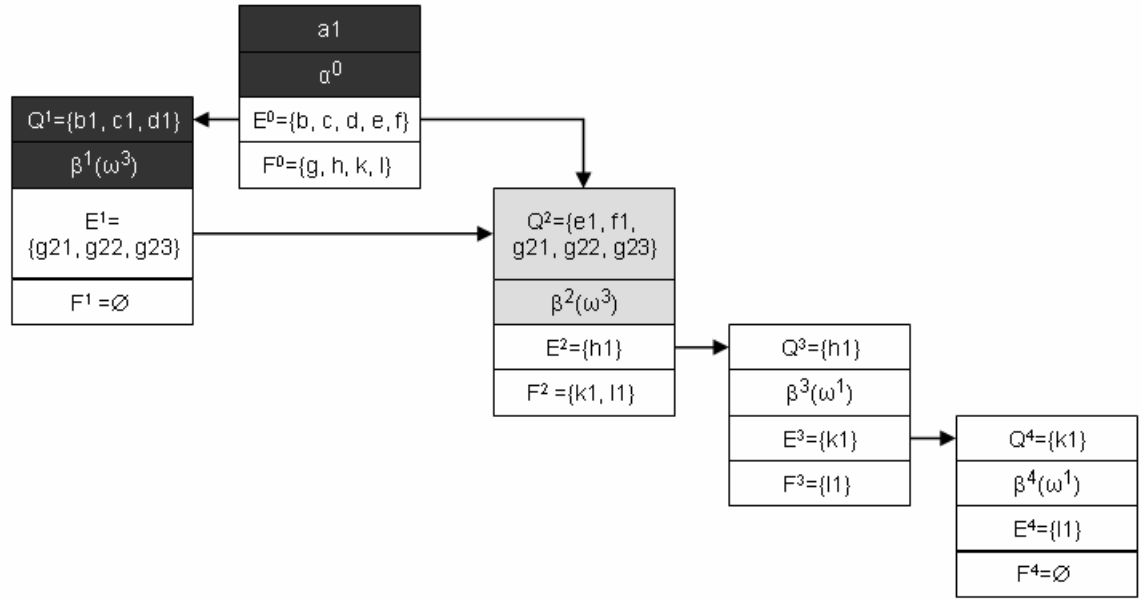


Figure 25: buying a train ticket $\chi(\omega_3)$

$\mathcal{U}(\omega_3)$ is obtained from $\mathcal{U}(\omega_2)$ by the following actions of (χ, a1): removal of g1 and addition of

sentence	0 = def. 1 = non def.	emitted by
g21 // is train 5577 OK for John? //	0	(χ, a1)
g22 // is train 5878 OK for John? //	0	(χ, a1)
g23 // is train 9867 OK for John? //	0	(χ, a1)

Next possible transition is: $\omega_3 \rightarrow$ **activate** (($\beta^2(\omega_3)$, a1, 1), "answers from John") $\rightarrow \omega_4$

$\mathcal{X}(\omega_4)$ in Figure 26depicts the reduction of all β^2's followers. It appears that the reducer φ'^2 has produced Ø, whereas φ^2 has produced h21, h22. β^3 (ω_4) is now ready to be activated.

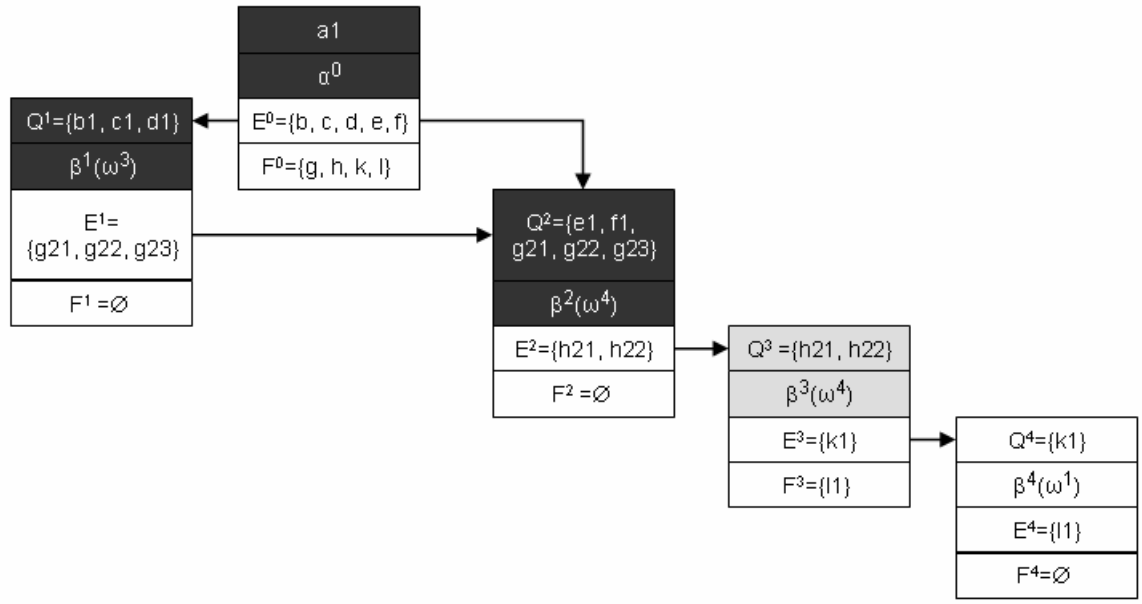


Figure 26: buying a train ticket $\mathcal{X}(\omega_4)$

$\mathcal{U}(\omega_4)$ is obtained from $\mathcal{U}(\omega_3)$ by the following actions of (χ, a1): removal of h1 and addition of: h21 // is "train 5577& optiPrice" OK for John? // and h22 // is "train 5878 & miniPrice" OK for John? //.

Next possible transition is illustrated in Figure 27 where the reduction of β^4 makes it ready to be activated:

$\omega_4 \rightarrow$ **activate** (($\beta^3(\omega_4)$, a1, 1), "answers from John") $\rightarrow \omega_5$

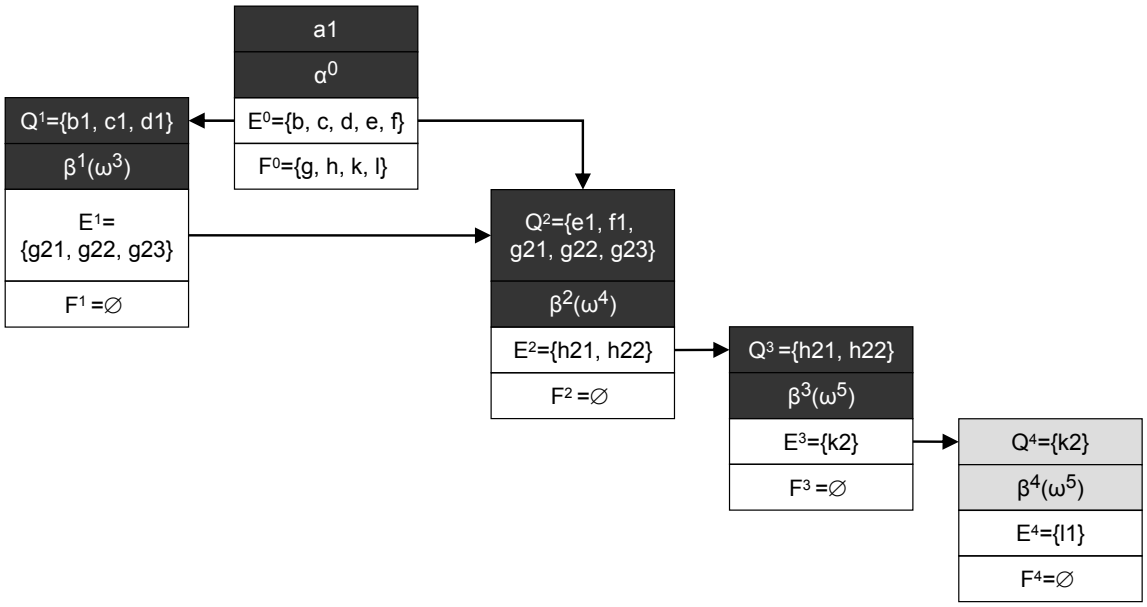


Figure 27: buying a train ticket $\mathcal{X}(\omega_5)$

$\mathcal{U}(\omega_5)$ is obtained from $\mathcal{U}(\omega_4)$ by the following actions of (χ, a1): removal of k1 and addition of: k2 // does John confirm "train 5878 & miniPrice"? //

Last transition is illustrated in Figure 28:

$\omega_5 \rightarrow$ **activate** (($\beta^4(\omega_5)$, a1, 1), "answers from John") $\rightarrow \omega_6$

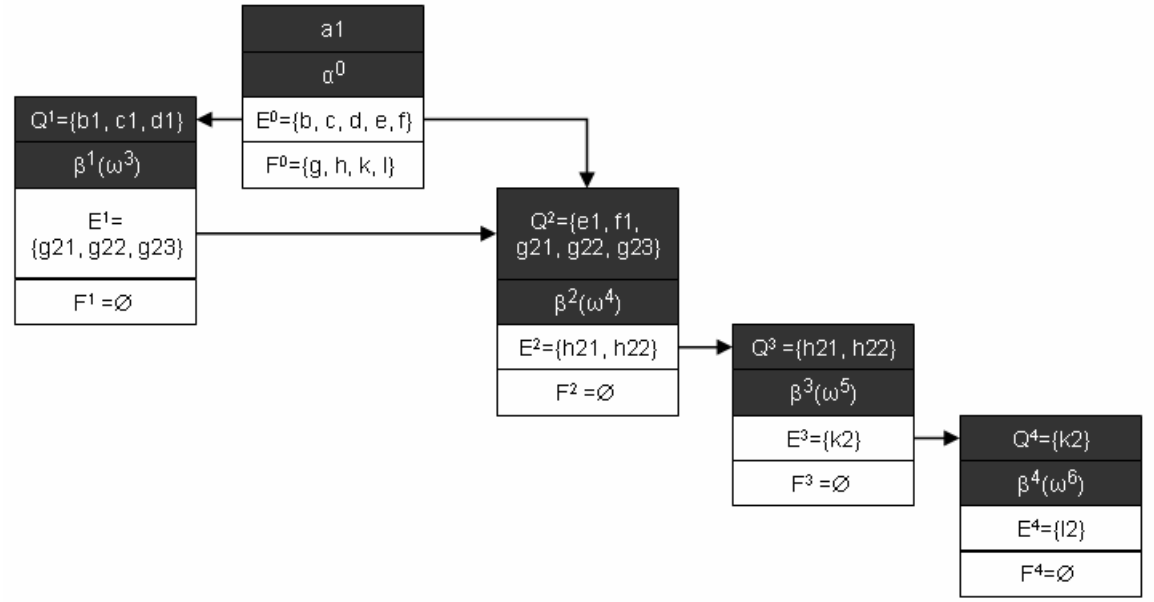


Figure 28: buying a train ticket $\mathcal{X}(\omega_6)$

$\mathcal{U}(\omega_6)$ is obtained from $\mathcal{U}(\omega_5)$ by the following actions of (χ, a1): removal of // l1 // and addition of : l2 // "train 5878 & miniPrice" has been reserved for John //

3.5 Computing the conversation

All along this chapter, we shall denote: $\Omega = \{(\mathcal{U}(\omega), \mathcal{X}(\omega))\}$ the infinite set of all possible states of the conversation.

3.5.1 Testing the current state of the conversation

The current *conversation state* will be analyzed with three questions in mind:

1. are there new triggers for present *startSteps*?
2. is it possible to answer the questions of a *postStep*?
3. are we allowed to remove a *startStep*?

In order to be able to rule the transitions in a deterministic way on a logical basis, we need sophisticated tests. As it has been outlined in the introduction, the relations "reductionOf", "triggerFor" and "answerFor" between sentences will be the basis of the control of the conversation.

The first question is addressed by a test that will answer true if and only if a given utterance "u" can play the role of new trigger for a given *startStep* "x_P^0":

Definition 7a:
Let X be a *conversation frame*, let $\chi_P = \{\chi_P^i\}_{i\geq 0} \in X$, let ω_0 be a *conversation initial state*, let $x_P^0 = (\chi_P^0, *, 1)$ be the *startStep* of χ_P where $\chi_P^0 = (\alpha(s_P), \varphi^0, \varphi'^0, E_P^0, F_P^0)$,
let ω be the current state the conversation,
let u = (a, 0) be a definitive utterance,

testInvoke (x_P^0, u) is a function $\{(x_P^0, u)\} \times \Omega \rightarrow \{(x_P^0, u)\} \times \{true, false\}$ analyzing the current state of the conversation in terms of the possibility for "u" to invoke x_P^0.

- $cond^1$ verifies that "u= (a, 0)" is a definitive trigger for χ_P^0: $cond^1$ = ['a' triggerFor s_P]
- $cond^2$ verifies that the corresponding invocation has not already occurred: $cond^2 = [(\chi_P^0, a, 0) \notin \mathcal{X}(\omega)]$

[**testInvoke** $((x_P^0, u), \omega) = ((x_P^0, u), true)] \Leftrightarrow$ [$(cond^1 = true)$ and $(cond^2 = true)$]

Let X be a set of *startSteps*, let U be a set of definitive utterances, let X × U be the "test domain", **testInvoke** (X × U, ω) is the algorithm that executes:
$\forall (x, u) \in X \times U$, **testInvoke** $((x, u), \omega)$

In order to address the second question, the following test checks whether a given postStep is ready to be activated:

Definition 7b:

Let X be a *conversation frame*, let $\chi_P=\{\chi_P^i\}_{i\geq 0} \in X$, let ω_0 be a *conversation initial state*, let ω be the current state the conversation,

let $x_P^i = (\chi_P^i, a, 1)$ be the *postStep* of χ_P associated to the context "a" and to the *postNode* χ_P^i in the current state $(\mathcal{A}ns(N^i), \varphi^i, \varphi'^i, E_P^i, F_P^i)$

testActivate (x_P^i) is a function $\{x_P^i\}\times \Omega \rightarrow \{x_P^1\}\times 2^S \times \{\text{true, false}\}$

- $cond^1$ first verifies that $x_P^i(a)$ is "activable" i.e. that all steps "driving" $x_P^i(a)$ have become definitive, then verifies that no more potential answers are available for the sentences of N^i: $cond^1 = [((\chi_P^j \text{ drives } \chi_P^i) \Rightarrow (\chi_P^j, a, 0) \in \mathcal{X}(\omega))$ and $\mathcal{P}otential\mathcal{A}nswers\mathcal{F}or(N^i, \omega) = \varnothing]$

- $cond^2$ verifies x_P^i (a) has not already been activated: $cond^2 = [(\chi_P^i, a, 0) \notin \mathcal{X}(\omega)]$

- the test returns the set of all definitive answers related to N^i:

[**testActivate** $(x_P^i, \omega) = ((x_P^i, \mathcal{D}efinitive\mathcal{A}nswers\mathcal{F}or(N^i, \omega)), \text{true})] \Leftrightarrow$ [$(cond^1 = \text{true})$ and $(cond^2 = \text{true})$]

Let X be a "test domain" composed of *postSteps*, **testActivate** (X, ω) is the algorithm that executes:

$\forall x \in X$, testActivate (x, ω)

The activation of postSteps is in a way more delicate than the invocation of startSteps, since they potentially depend on all the other steps in the conversation. The following Lemma is a direct consequence of all the definitions having contributed to our central notion of conversational pattern; it justifies the vocabulary adopted and is a key argument in future proofs of good properties of the conversational calculus:

Lemma2:

Let X be a *conversation frame*, let $\chi_P = \{\chi_P^i\}_{i\geq 0} \in X$ ap pattern consisting of nodes,

let ω_1 = be the state the conversation resulting from **invoke** (x_P^0, a),

let $x_P^k = (\chi_P^k, a, 1)$, $x_P^m = (\chi_P^m, a, 1)$ and $x_P^n = (\chi_P^n, a, 1)$ be 3 *postSteps*,

let ω_2 = be the state the conversation when **testActivate** $(x_P^k, \omega_2) = ((x_P^k, A^k), \text{true})$,

let ω_3 the resulting state of **testActivate** (x_P^k, ω_2),

1) successive activations do not bring new "drive" arrows into the graph of a conversational pattern: $\chi_P^m(\omega_3) \text{ drives } \chi_P^n(\omega_3) \Rightarrow \chi_P^m(\omega_2) \text{ drives } \chi_P^n(\omega_2)$;

2) at state ω_2, all the *predecessors* of χ_P^k have already been activated, moreover only *predecessors* of χ_P^k have influenced the current $\chi_P^k(\omega_2) = (\mathcal{A}ns(N_P^k(\omega_2)), \varphi^k, \varphi'^k, E_P^k(\omega_2), F_P^k(\omega_2))$;

3) all the utterances $(s \in N_P^k(\omega_2), bool)$ emitted by (χ_P, a) are definitive.

Demonstration:

1) Definition 4d/d says that N_P^n is partitioned by the E_P^i , Definition 6b says that the E_P^i will be reduced by successive activations of steps, Definition 4d/b says that such reductions will never intersect.

2) all the of *drivers* χ_P^k have been activated at state ω_2, then by transitivity the drivers of the drivers have been activated; moreover, it follows from Definition 4c that only *drivers* of χ_P^k have an influence on N^k ; and it follows from Definition 6b that only *predecessors* of χ_P^k have an influence on E_P^k and F_P^k

3) from Definition 4c/d: let $\mathcal{D}$ be the set the *drivers* of χ_P^k, then $N_P^k = \cup_{d\in\mathcal{D}}(N_P^k \cap E_P^d)$

since all the χ_P^d have been activated, all the utterances emitted by (χ_P, a) and embedding sentences of E_P^d are definitive!

To end with the third question:

Definition 7c:

Let X be a *conversation frame*, let $\chi_P = \{\chi_P^i\}_{i\geq 0} \in X$, let ω_0 be a *conversation initial state*, let $x_P^0 = (\chi_P^0, *, 1)$ be the *startStep* of χ_P where $\chi_P^0 = (\alpha(s_P), \varphi^0, \varphi'^0, E_P^0, F_P^0)$,
let ω be the current state the conversation,

testRevoke (x_P^0) is a function $\{x_P^0\}\times\Omega \rightarrow \{x_P^0\}\times\{true, false\}$ analyzing the current state of the conversation in terms of the possibility for "x_P^0" to be revoked.

- $cond^1$ first verifies that no more potential triggers exist for the starter of χ_P^0, then verifies that all definitive triggers for the starter of χ_P^0 have gone through the corresponding invocation: $cond^1 = [\mathcal{P}otentialTriggersFor(s_P, \omega) = \varnothing$ and $(\forall u = (a, 0) \in \mathcal{D}efinitiveTriggersFor(s_P, \omega))(x_P^0, a, 0) \in \mathcal{X}(\omega)]$
- $cond^2$ verifies that the revocation has not already occurred: $cond^2 = [(\chi_P^0, *, 1) \in \mathcal{X}(\omega)]$

$[\textbf{testRevoke}(x_P^0, \omega) = (x, true)] \Leftrightarrow [(cond^1 = true)$ and $(cond^2 = true)]$

Let X be a "test domain" composed of *startSteps*, **testRevoke** (X, ω) is the algorithm that executes:
$\forall x \in X$, **testRevoke** (x, ω)

3.5.2 Ruling the transitions

The three *atomic transitions*, which are non-interruptible changes from a state ($\mathcal{U}(\omega)$, $\mathcal{X}(\omega)$) to another state ($\mathcal{U}(\omega')$, $\mathcal{X}(\omega')$), will be conditioned by the three above tests, according to three transition rules of Figure 29:

3 Transition Rules:

Let X be a *conversation frame*, let $\chi_P=\{\chi_P^i\}_{i\geq 0} \in X$, let ω_0 be a *conversation initial state*, let $x_P^0 = (\chi_P^0, *, 1)$ be the *startStep* of χ_P where $\chi_P^0 = (\alpha(s_P), \varphi^0, \varphi'^0, E_P^0, F_P^0)$,

let $x_P^1 = (\chi_P^1, a, 1)$ be a *postStep* of χ_P associated to the context "a" and to the *postNode* χ_P^1 in the current state ($\mathcal{Ans}(N^1)$, φ^1, φ'^1, E_P^1, F_P^1) ,

let u = (a, 0) be a definitive utterance,

let $A^1 = \mathcal{DefinitiveAnswersFor}(N^1, \omega)$,

testInvoke $(x_P^0, u, \omega) = ((x_P^0, u), true) \rightarrow$ **invoke** (x_P^0, a)

testActivate $(x_P^1, \omega) = ((x_P^1, A^1), true) \rightarrow$ **activate** (x_P^1, A^1)

testRevoke $(x_P^0, \omega) = (x_P^0, true) \rightarrow$ **revoke** (x_P^0)

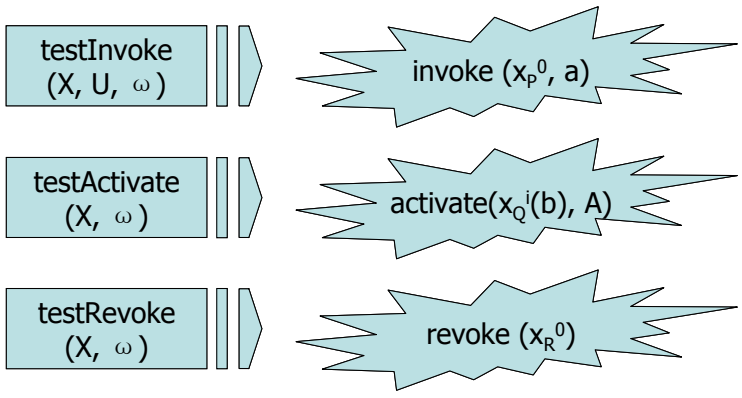


Figure 29: three transition rules

Lemma3 below expresses the very important following fact:

When a test gives ("result", "true") for a given state, it gives the following alternative:

- either the corresponding transition is immediately executed, and leads to a state where the test gives ("false");
- or another transition is executed and leads to a new state in which the test keeps on giving the same ("result", "true").

Moreover, the result of the corresponding transition is independent of the state when it occurs!

Lemma3:

Let **test** be a test and **transition** the associated transition (according to the transition rules),

let "following" an observed chronological order in a real conversation, "ω_2 following ω_1" meaning either $\omega_2 = \omega_1$ or "ω_2 after ω_1 in the conversation".

let ω_{true} the first state of the conversation satisfying: **test**(ω_{true}) = (**result**(ω_{true}), true),

let ω_{init} the initial state for **transition**, ω_{init} following ω_{true},

let ω_{final} the final state for **transition** (since a transition is atomic, there is no state between ω_{init} and ω_{final})

let ω_{after} following ω_{final},

test(ω_{init}) = (**result**(ω_{true}), true)

test(ω_{after}) = false

Moreover, **transition** will give the same result at ω_{true} or at ω_{init}!

Demonstration:

Each test has been explicitly constructed on the conjunction of two conditions:

- $cond^1$ is "transition-invariant", which means that once true, all the further transitions on $U(\omega)$ will keep it true (easy to demonstrate)
- $cond^2$ is true as long as **transition** has not occurred, false afterwards (immediate)

Moreover, the result of **test**(ω_{init}) is exactly **result**(ω_{true}), and **transition** will give the same result at ω_{true} or at ω_{init}!

This is evident in the case of "**invoke**" or "**revoke**"; let us prove it in the case of "**activate**":

Let $x_P^i = (\chi_P^i, a, 1)$ be a *postStep* in the current state ($\mathcal{Ans}(N^i)$, φ^i, φ'^i, E_P^i, F_P^i), according to Lemma2 , all its *predecessors* have already been activated at state ω_{true} and only them have an influence on it, therefore $x_P^i(\omega_{init}) = x_P^i(\omega_{true})$! Besides, a new definitive answer can only be a reduction of a potential one, no definitive answers have been produced between ω_{true} and ω_{init} since $\mathcal{PotentialAnswersFor}(N^i(\omega_{true}), \omega_{true}) = \varnothing$; it follows that $\mathcal{DefinitiveAnswersFor}(N^i(\omega_{init}), \omega_{init}) = \mathcal{DefinitiveAnswersFor}(N^i(\omega_{true}), \omega_{true})$!

3.5.3 Transitions and further tests in a loop

The dynamic generation of tests during the conversation is illustrated by Figure 30:

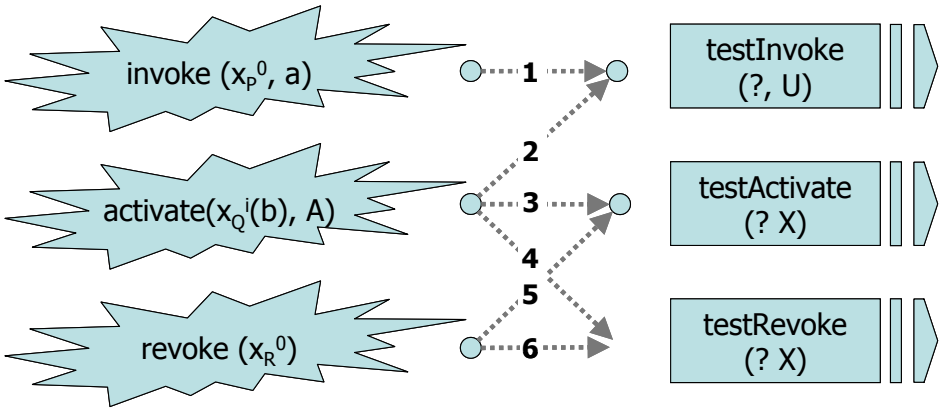


Figure 30: test generation

Each of the three types of transition has specific consequences on the generation of further tests:

- whenever a conversational pattern is **invoked**, it emits definitive utterances, those might invoke other patterns, therefore testInvoke must be operated (arrow: 1); besides, since no non-definitive utterance is removed, neither testActivate nor testRevoke need to be run.
- whereas whenever a conversational pattern is **revoked**, it removes non-definitive utterances, so that we must run testActivate (arrow: 5) or testRevoke (arrow: 6).
- to end with, whenever a conversation step is **activated**, it emits definitive utterances and non-definitive utterances, and removes non-definitive utterances, so that the three tests (arrow: 2) (arrow: 3) (arrow: 4) have to be run again.

Thanks to Lemma 3, the calculus can be ruled very easily: it is of no importance when the tests happen or when the transitions happen! We are allowed to "store" transitions as well as tests in two "pipes", as illustrated in Figure 31, without worrying in which order they will be operated.

Each *atomic transition* popping out of the "transition pipe" generates a new *state*, and generates new tests which are stored into the "test pipe" in order to be operated.

Each test popping out of the "test pipe" either is not successful, or generates new transitions which are stored into the "transition pipe".

Whereas *atomic transitions* must happen "one at a time" (otherwise no notion of state holds) tests may be operated in parallel during any state, therefore their computational cost can be distributed.

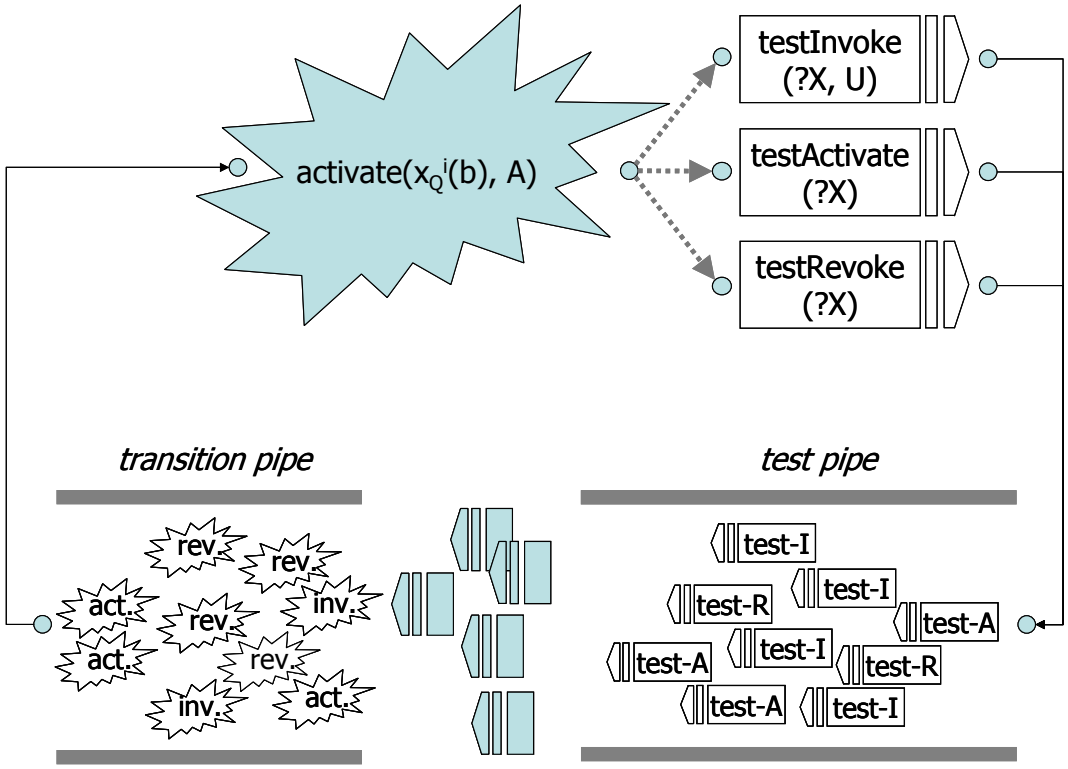


Figure 31: transitions and further tests in a loop

Optimization of the calculus will be the art of:

1. operating different tests in parallel on each given state;
2. reducing the number of sentences to be tested by each of them, by managing "test domains"!

We are not going to tackle this optimization here, only to point to syntactic properties which will allow it.

We first define two new relations on *conversation nodes*.
And then we define sets of steps on the basis of these two relations, in order to manage "test domains" according to the potential triggers and potential answers for those steps.

Definition 8a:

Let X be a *conversation frame*, let $\chi_P=\{\chi_P^i\}_{i\geq 0} \in X$, let $\chi_Q=\{\chi_Q^i\}_{i\geq 0} \in X$,

let $\chi_P^0 = (\alpha(s_P), \varphi_P^0, \varphi_P^{0'}, E_P^0, F_P^0)$ be the *startNode* of χ_P,

let $\chi_P^i = (\alpha(s_P), \varphi_P^i, \varphi_P^{i'}, E_P^i, F_P^i)$ be a *postNode* of χ_P,

let $\chi_Q^0 = (\alpha(s_Q), \varphi_Q^0, \varphi_Q^{0'}, E_Q^0, F_Q^0)$ be the *startNode* of χ_Q,

let $\chi_Q^j = (\mathcal{A}ns(N^j), \varphi_Q^j, \varphi_Q^{j'}, E_Q^j, F_Q^j)$ be a *postNode* of χ_Q,

χ_Q^0 potentiallyTriggeredBy $\chi_P^i \Leftrightarrow \mathcal{T}riggersFor(s_Q) \cap \mathcal{R}eductionsOf(E_P^i \cup F_P^i) \neq \varnothing$

χ_Q^j potentiallyAnsweredBy $\chi_P^i \Leftrightarrow \mathcal{A}nswersForReductionsOf(N^j) \cap \mathcal{R}eductionsOf(E_P^i \cup F_P^i) \neq \varnothing$

Definition 8b:

Let X be a *conversation frame*, let $\chi_P=\{\chi_P^i\}_{i\geq 0} \in X$,

let ω be the current state of the conversation,

let $x^0 = (\chi^0, *, 1)$ denote an undefined *startStep* corresponding to an undefined pattern χ,

let $x^i(\omega) = (\chi^i(\omega), y, 1)$ denote an undefined *postStep* corresponding to an undefined *invocation* (χ, y),

$\mathcal{P}otentiallyTriggeredBy(\chi_P^i, \omega) = \{x^0 \in \mathcal{X}(\omega) \mid \chi^0$ potentiallyTriggeredBy $\chi_P^i\}$

$\mathcal{P}otentiallyAnsweredBy(\chi_P^i, \omega) = \{x^j(\omega) \in \mathcal{X}(\omega) \mid \chi^j$ potentiallyAnsweredBy $\chi_P^i\}$

We can now be more precise about "X" and "U" in Figure 31 while ruling the successive steps of the conversation after each atomic transition:

- (1): X = $\mathcal{P}otentiallyTriggeredBy(\chi_P^0, \omega)$; U = new definitive utterances;
- (2): X = $\mathcal{P}otentiallyTriggeredBy(\chi_Q^i, \omega)$; U = new definitive utterances;
- (3): X = $\mathcal{P}otentiallyAnsweredBy(\chi_Q^i, \omega)$;
- (4): X = $\mathcal{P}otentiallyTriggeredBy$ (*predecessors* of χ_Q^i, ω);
- (5): X = $\mathcal{P}otentiallyAnsweredBy(\chi_R^0, \omega)$;
- (6): X = $\mathcal{P}otentiallyTriggeredBy(\chi_R^0, \omega)$.

It might seem that we have just moved "test cost" into "set computation cost"; it is not the case: each of these set will be itself *reduced* along the successive transitions! This means that they have to be computed once at the beginning, and then checked after each transition, which reduces drastically the number of necessary comparisons. Again, we are not intending to exhaustively explore this point here.

3.5.4 Will this conversation end in finite time?

An informal counter- example will show that some conversations may never end:

Let us define a set of sentences {**a**(t), **s** }:

a(t) = (// Linda leaves Paris "later than t" //, 0), such that **a**(t+1) ≤**a**(t)

s = (// human leave Town "at time x" //), such that **a**(t) triggerFor **s**

If we suppose a conversation based on:

- the initial assertion **a**(0);
- a unique one-node pattern χ^0 = (s, φ^0, φ'^0, E ={**a**(0)}, F = Ø) , the reducer of which transforms **a**(t) in **a**(t+1).

Then χ^0 is potentially triggered by itself and we have an infinite monologue!

We therefore need a condition on the initial conversation frame guarantying an end it finite time; Definition 8c gives a sufficient condition.

Definition 8c:
Let X = $\{\chi_I\}_{I \in \mathcal{P}atterns}$ be a *conversation frame*,
let $\mathcal{G}$ be the directed graph defined on X by the relation "potentiallyTriggeredBy" between corresponding *startNodes*,
X is a *cycle-free conversation frame* iff there are no cycles in $\mathcal{G}$.

Theorem 1:
Let X = $\{\chi_I\}_{I \in \mathcal{P}atterns}$ be a *cycle-free conversation frame*,

Any *conversation run* on X will end in a finite number of atomic transitions.

Demonstration:
If there are no cycles in $\mathcal{G}$, we can rank the patterns χ_N in the following way:
rank(χ_P) = 0 $\Leftrightarrow$ χ_P potentiallyTriggeredBy none
…
rank(χ_P) = n $\Leftrightarrow$ (not rank(χ_P) <n) and ($\exists\chi_Q$ | χ_P potentiallyTriggeredBy χ_Q
and rank(χ_Q) = n-1)
Each pattern will participate in only one "**revoke**" transition. According to Definition 4a of a reducer and to Definition 4d of a conversational pattern, the number of utterances emitted by any invocation (and therefore the number of "**invoke**" transitions) remains finite. By recurrence on the rank, we establish that the total number of "**invoke**" transitions remains finite. Since a pattern is a finite set of nodes, each invocation allows only a finite number of "**activate**" transitions.

3.5.5 Are there several possible final states?

The conversational calculus is aimed at automatic composition of logical components; the property of confluence is therefore a major expectation. This is demonstrated in Theorem 2.

Theorem 2:

Let $X = \{\chi_I\}_{I \in \mathcal{P}atterns}$ be a *conversation frame*,

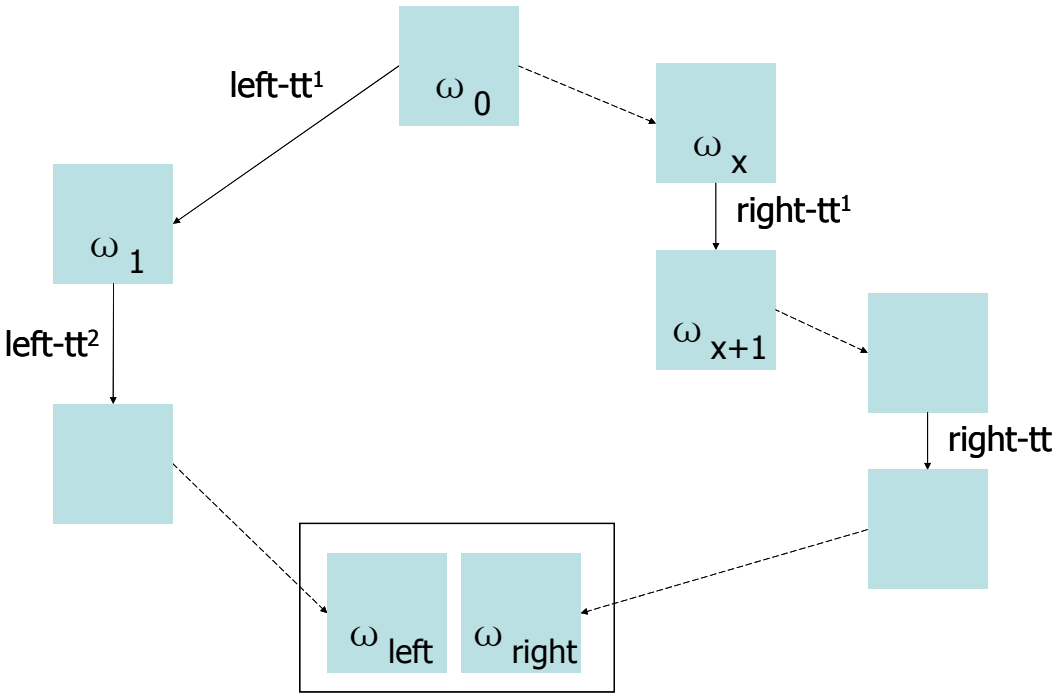
let $\omega_0 = (\mathcal{U}(X, ☺, \omega_0), \mathcal{X}(X, \omega_0))$ be a *conversation initial state* where:

- $\mathcal{X}(\omega_0)$ is the set $\{x_I^0 = (\chi_I^0, *, 1)\}_{I \in \mathcal{P}atterns}$ of *startSteps*
- $\mathcal{U}(X, ☺, \omega_0)$ is the initial set of *utterances*

let "left-conversation" be a first run of conversation ω_0 ending at ω_{left} after a succession of $(\text{left-tt}^i = (\text{left-test}^i, \text{left-transition}^i))_{i \in I}$,

let "right-conversation" be a second run of conversation ω^0 ending at ω_{right} after a succession of $(\text{right-tt}^j = (\text{right-test}^j, \text{right-transition}^j))_{j \in J}$,

$$\omega_{left} = \omega_{right}$$



Demonstration:

1) "left-conversation" and "right-conversation" have the same test-transitions, although the order may be different.

- let us consider left-tt^1; left-test^1 is true at ω_0, so left-test^1 is true at ω_{right} (Lemma2) unless left-tt^1 has previously occurred in "right-conversation". Since ω_{right} is a final state, it is not possible that left-tt^1 follows, therefore left-tt^1 has occurred before ω_{right} in "right-conversation"! let ω_x be the state immediately before left-tt^1 in the right branch, and ω_{x+1} be the state immediately after; left-test$^1(\omega_x)$ = left-test$^1(\omega_0)$, therefore left-transition$^1(\omega_x)$ = left- transition$^1(\omega_0)$, since transitions compute only the result of the tests.
- let ω_1 be the state immediately after left-tt^1 in the left branch: from the viewpoint of cond^1, left-test$^2(\omega_1)$ = left-test^2 (left-transition$^1(\omega_0)$) = left-test^2 (left-transition$^1(\omega_x)$) = left-test$^2(\omega_{x+1})$. Again, either left-tt^2 has occurred before ω_x or will occur after ω_{x+1}: left-tt^2 also belongs to the right branch!
- …

2) Since "left-conversation" and "right-conversation" have the same test-transitions, they lead to the same state because of Lemma2 (two given transitions, either do not influence each other, or appear in the same order).

3.5.6 Tackling the question of complexity

Let us take the following denotations:

τ = number of conversational patterns

υ = initial number of *definitive* utterances

σ_0 = constant = maximal number of conversation nodes inside a conversational pattern

ψ = worst complexity function of a node: when $\mathcal{U}(\omega)$ is composed of "n" utterances, a step can produce $O(\psi(n))$ new utterances.

We are going to define a new notion of "step rank" as follows:

- startSteps have rank 0
- steps with all *predecessors* of rank <= 0 have rank 1
- …
- steps with all *predecessors* of rank <= "k" have rank "k+1"

We reason on an imaginary *conversation run* in which all possibly activated steps of each rank are activated simultaneously (all answers always immediately available) ; therefore maximum number of ranks = σ_0!

Let X(k) be the number of steps at rank "k" and U(k) be the number of *definitive* utterances at rank "k"; X(k) and U(k) are linked by be the following relation:

- $X(k+1) = O(\sigma_0 * X(k)$ (postSteps driven by steps of rank "k") + U(k) (new invocations)) = $O(X(k)+ U(k))$
- $U(k+1) = O(X(k+1) * \psi(U(k)))$

at rank 0: $X(0) = \tau$; $U(0) = \upsilon$

at rank 1: $X(1) = O(X(0) + U(0)) = O(\tau + \upsilon)$

$U(1) = O(X(1) * \psi(U(0))) = O((\tau + \upsilon) * \psi(\upsilon))$

at rank 2: $X(2) = O(X(1) + U(1)) = O((\tau + \upsilon) * (1+\psi(\upsilon)))$

$U(2) = O(X(2) * \psi(U(1))) = O((\tau + \upsilon) * (1+\psi(\upsilon)) * \psi((\tau + \upsilon) * \psi(\upsilon)))$

…

"ψ" plays of course a central role towards global complexity, we must therefore remember that reducers are not aimed at producing information, rather at digesting it. For this reason, "$\psi = O(n)$" is a reasonable hypothesis[61,62]:

$X(0) = \tau$; $U(0) = \upsilon$

$X(1) = O(\tau + \upsilon)$; $U(1) = O(\tau \upsilon + \upsilon^2)$;

$X(2) = O(\tau \upsilon + \upsilon^2)$; $U(2) = O((\tau \upsilon + \upsilon^2)^2)$;

$X(3) = O((\tau \upsilon + \upsilon^2)^2)$; $U(3) = O((\tau \upsilon + \upsilon^2)^{2*2})$;

…

$X(\sigma_0) = O((\tau \upsilon + \upsilon^2)^a)$; $U(\sigma_0) = O((\tau \upsilon + \upsilon^2)^b)$; $a = 2^{(\sigma 0-2)}$; $b = 2^{(\sigma 0-1)}$

When "$n=\max(\tau, \upsilon)$" is the initial size of a conversation, when σ_0 is the maximum number of nodes in a pattern, when the internal complexity of the worst reducer is $\psi = O(n)$: *conversation run* = $O(n \hat{} 2^{\sigma 0})$.

---

[61] in the 9*9 SUDOKU example, ψ is characterized by "each new accepted value generates 3*8 restrictions".

[62] it might happen however that each reducer builds a two-dimensional matrix starting from two independent vectors, which corresponds to "$\Psi = O(n^2)$" and leads to the following :

$X(0) = \tau$; $U(0) = \upsilon$

$X(1) = O(\tau + \upsilon)$; $U(1) = O(\tau \upsilon^2 + \upsilon^3)$;

$X(2) = O(\tau \upsilon^2 + \upsilon^3)$; $U(2) = O((\tau \upsilon^2 + \upsilon^3) * (\tau \upsilon 2 + \upsilon 3)^2) = O((\tau \upsilon^2 + \upsilon^3)^3)$;

$X(3) = U(2)$; $U(3) = O((\tau \upsilon^2 + \upsilon^3)^3 * (\tau \upsilon^2 + \upsilon^3)^{3*2}) = O((\tau \upsilon^2 + \upsilon^3)^9)$;

…

$U(\sigma_0) = O((\tau \upsilon + \upsilon 2)^b)$; $b = x_{(\sigma 0-2)}$; $x_0 = 3$; $x_n = x_{n-1} + x^2_{n-1}$ … still polynomial but with a rather frightening degree!

3.5.7 Turing Machines seen as particular conversational patterns

Let TM be a Turing Machine,

let W be the language recognized by TM i.e. the "words" on which the TM will reach its final state in finite time,

let $\mathcal{S}$ be the infinite set of all possible states of the TM tape ($W \subset \mathcal{S}$), with the addition of a special sentence "$\mathbf{s_{terminal}}$" verifying :

$$\forall s \in \mathcal{S}: s <= \mathbf{s_{terminal}}$$

$$\forall w \in W: w \text{ triggerFor } \mathbf{s_{terminal}}$$

Now the "words" of a TM are "sentences" in conversational calculus … let us now come back to:

Definition 4b:

A *node* is a tuple $\nu = (A, \varphi, \varphi', E, F)$ where:
- $A \in 2^{\mathcal{S}}$,
- $E \in 2^{\mathcal{S}}$ and $F \in 2^{\mathcal{S}}$ are disjoint finite sets of sentences
- $\Phi = (A, \varphi, E)$ and $\Phi' = (A, \varphi', F)$ are two reducers sharing the same domain A
- the sentences of E will become *definitive* whenever φ is activated
- the sentences of F will stay *non-definitive* whenever φ' is activated

Only two cases will be considered:
- $A = \mathcal{TriggersFor}(N=\{s\})$; in this first case, the node will be called a startNode and will be denoted ($\mathcal{Trig}(s), \varphi, \varphi', E, F$)
- $A = \mathcal{AnswersForReductionsOf}(N)$; in this second case, the node will be called a *postNode* and denoted ($\mathcal{Ans}(N), \varphi, \varphi', E, F$)

We shall keep the denotation ($A, \varphi, \varphi', E, F$) only when we do not want to specialize a case among those two.

We may consider the node ν_{TM} defined by:

- $E = \{\mathbf{s_{terminal}}\}$
- $F = \emptyset$
- $A = W$
- φ = TM
- φ' = any constant

Then ν_{TM} can play the role of unique *startNode* of a conversational pattern χ_{TM} which will provide (under the shape of a definitive utterance) an answer (which will be the final state of the TM tape) in finite time when triggered by any word of L!

We can therefore convey several TMs into a *conversation frame* so that they can interact, provided the fact that the final state of TM_1's tape is a sentence recognized by TM_2
The delicate point is the following:
problem7: "does $s \in \mathcal{S}(TM_1)$ belong to $W(TM_2)$? " is known as undecidable !
What will happen then if a "bad word" finds its way through a *conversation run* implying several TMs? It may happen that the calculus does not end in finite time … yet, if it does end, Theorem 2 applies and the obtained final state is unique!

3.6 Technical glossary

activation: (of a *postStep*) activation of the embedded *reducer* when all the necessary information is available (answer to questions emitted by *predecessors*) [Definition 6a]

answerFor: [Definition 1]

atomic transition: a transition is an atomic change of state in the conversation; which means that no intermediate state can occur between the initial state and the final state of a transition [Definition 5c]

conversation: a conversation is defined by a set of *conversational patterns* and a set of initial definitive utterances [Definition 5c]

conversation frame: a chosen set of *conversational patterns* [Definition 5c] A conversation frame is therefore a set of *conversation nodes*, with some good properties so that they implement the formally correct design of *conversational patterns* which are going to interact during one or more *conversation runs*.

conversation node: a unit of information processing during a conversation [Definition 4d]

conversational pattern: a pattern followed during a conversation consisting in a set of *conversation nodes* with a dependency relation [Definition 4d]

conversation run: succession of *states* starting by an initial state defined by a *conversation frame* + an initial set of utterances and computed by a derivation of the 3 *transition rules* of the conversational calculus [Definition 5c]

conversation step: object corresponding to the contextualized activation of a *conversation node*. A conversation step has a status: status "0" means "definitively reduced"; status "1" means "not definitively reduced". [Definition 5a]

cycle-free conversation frame: a *conversation frame* for which any *conversation run* will end in finite time [Definition 8c] [Theoreme 1]

definitive utterance: an *utterance* which will stay in the conversation [Definition 2]

driver: a *conversation node* is a *driver* of another *conversation node* if they belong to the same *conversational pattern* and if the first asks definitive questions of which the answers are needed by the second [Definition 4c]

emitter: one which adds or removes utterances according to the three following cases [Definition 5b]:

- the first case is when a *conversational pattern* χ_I is in a specific context of invocation defined by a sentence "a": this will be denoted "(χ_I, a)";
- the second case is when a *conversational pattern* is waiting for invocation: this will be denoted "(χ_I, ∗)";
- the third case is initial input given before conversation starts: this will be denoted "(☺ , ∗)".

follower: a *postNode* is a *follower* of another *conversation node* if they belong to the same *conversational pattern* and if the first is connected to the second via a succession of intermediary *drivers*. By extension a *postStep* is a follower of another *conversation step* if the associated *nodes* satisfy the condition. [Definition 4d]

invocation: i) used in a dynamic meaning (invocation of a *startStep* of a *conversational pattern*) activation of the embedded *reducer* when triggered by an appropriate *sentence* (playing the role of a context) ii) used in a static meaning (invocation of a *conversational pattern*) synonymous of *emitter* which is a pair (conversational pattern, context of activation) [Definition 5b; Definition 6a]

node: an algorithm based on a couple of *reducers* sharing the same "external input" and reducing respectively sentences from "internal input 1" and "internal input 2". The reduction of "internal input 1" will give birth to definitive utterances, whereas the reduction of "internal input 2" will give birth to non-definitive ones. Activation of algorithm depends on a single test on "external input". [Definition 4b]

non-definitive (utterance): utterance which may be removed during a *transition* of the conversation [Definition 2]

postNode: a *conversation node* which is after the first element of a *conversational pattern*. [Definition 4d]

postStep: a *conversation step* associated to a *postNode* in a given invocation of a *conversational pattern*. [Definition 5a]

predecessor: a *conversation node* is a *predecessor* of a *postNode* if they belong to the same *conversational pattern* and if the second is connected to the first via a succession of intermediary *drivers*. By extension a *conversation step* is a follower of a *postStep* if the associated *nodes* satisfy the condition. [Definition 4d]

reducer: function operating on two sets of sentences "A = external input" and "B = internal input" in such a way that the sentences of B are reduced in a way that depends on sentences of A , which becomes "function(A,B) = output" [Definition 3.3]

reductionOf: less abstract than, less ambiguous than, more informative than; [Definition 1]

revocation: (of a *startStep*) removal of the *startStep* which corresponds to the end of potential participation of the associated *conversational pattern*. [Definition 6c]

run (of a conversation): see *conversation run*

sentence: an element of a pre-ordered set where the relations answerFor and triggerFor are defined with good properties [Definition 1]

starter: a *sentence* playing the role of external input in the reducer of a *startNode*; as soon as a *sentence "s" triggerFor starter* appears in the conversation, the associated *conversational pattern* in invoked. [Definition 4d]

startNode: a *conversation node* which is the first element of a *conversational pattern*. [Definition 4d]

startStep: a *conversation step* associated to a *startNode* which is ready to be invoked. [Definition 5a]

state (of a conversation)n: a state of a conversation is described through two sets: the set of current *utterances*, and the set of current *conversation steps*.[Definition 5c]

transition: see *atomic transition*

transition rule: a rewriting rule in the space of states [Definition 3.5.2]

triggerFor: [Definition 1]

utterance: aggregate embedding a sentence, its status (definitive or not) and its list of emitters [Definition 2]

3.7 A generative grammar for sentences inspired by conceptual graphs

This sub-section presents the generative grammar for sentences. This grammar is used in the current prototype:

- in [3.7.1] we present it and show its expressivity by referring to conceptual graphs;
- in [3.7.2] we informally outline the logical semantic;
- in [3.7.3] we check in a semi-formal way the decision problems listed in [3.2.1].

3.7.1 From conceptual graphs to sentences

Conceptual graphs were introduced in [Sowa, 1976] as a graphical interface for relational databases; later on, as a continuation of Peirce's work, a logical semantic was provided by [Sowa, 1984] in the "existential, conjunctive and positive" fragment of the First Order Logic. A very accurate description of conceptual graphs, as well as a proposition for reasoning with graphs can be found in [Baget, 2001] which provided a significant contribution to the research work of the "Graphes Conceptuels" team in LIRMM[63]; their basic model relies on three main tenets:

- Knowledge is represented by graphs;
- Reasoning is represented by operations on graphs;
- the model is adequate and complete with respect to a logical semantic.

In this document, we use conceptual graphs as a starting point for Knowledge representation; the graphs we manipulate in the calculus are only the "elementary graphs" built around one single relation linking several entities … each of them can be expressed as a single predicate in First Order Logic.

In [Mugnier & Chein, 1996] the notion of "conceptual support" aimed at defining the basic vocabulary for Knowledge Representation is formally described. Its main ingredients are:

- a partially ordered set of Concepts;
- a set of Relations;

[63] Laboratoir d'Informatique, de Robotique et de Micro-electronique de Montpellier

- an enumerable set of “individual markers”, to which a generic marker denoted “*” is added, and an application associating a Concept to each individual marker.

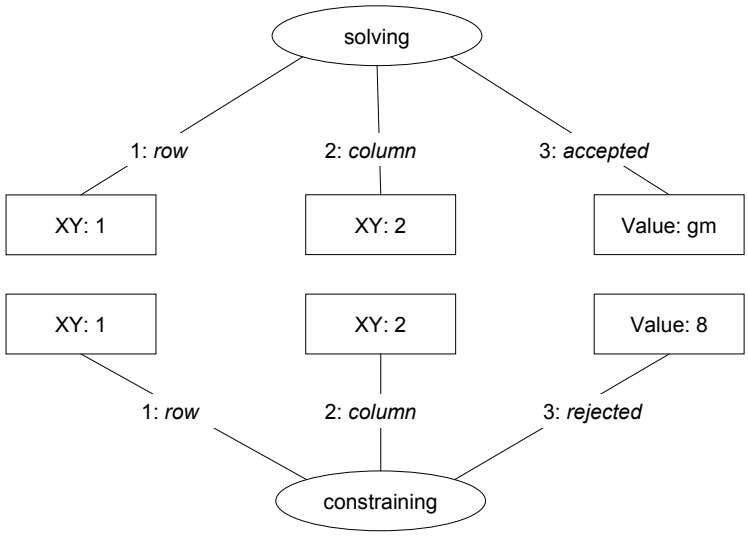


Figure 32: “two in one” conceptual graphs

In Figure 32, we illustrate by one conceptual graph consisting in two elementary graphs:

- “XY”, “Value” are the Concepts;
- “solving”, “constraining” are the Relations;
- {1, 2, 8} are individual markers; “gm” is the generic marker.

These two graphs together express the following:

“there exists one value which solves the cell (row=1, column=2) [Upper elementary graph]” AND “the value ‘8’ is rejected for the cell (row=1, column=2) [Lower elementary graph]”

Our goal in this sub-section is to extend this representation for expressing sentences such as:

assertive1: “there exists one value among {2, 3, 5, 6} which solves the cell (row=1, column=2)”

assertive2: “the value ‘2’ solves the cell (row=1, column=2)”

directive1: “what are all the values rejected for the cell (row=1, column=2)?”

commissive1: “one value among {2, 3, 5, 6} which solves the cell (row=1, column=2) will be later asserted”

commissive2: “the values rejected for the cell (row=1; column=x) will be later requested”

Remark: *commissive1* by *assertive2* intuitively fulfils the commitment, so does replacing *commissive2* by *directive1*.

In order to generate our language we start from the following “high level” components created as identifiers from a finite initial alphabet:

- a partially ordered set of “concepts” belonging to one of the two types *string* or *number* where only concepts of the same type can be compared and where each type gives a meet-semi-lattice, with the ‘meet’ of two concepts denoted ‘$concept_1 \cap concept_2$’;
- an infinite set of “roles”;
- an infinite set of “individual markers” which can be either *strings* or *numbers*;
- a generic marker: ‘ALL’;
- the “usual operators” for *numbers*: ‘<’, ‘<=’, ‘≠’, ‘>=’, ‘>’
- a Boolean “word status” in {0, 1} ; status = ‘0’ means definitive, status = ‘1’ means non-definitive.
- six “sentence types”: ‘AA’, ‘CA’, ‘SA’, ‘DD’, ‘CD’, ‘SD’, for “assertive”, “commissive/assertive”, “starter/assertive”, “directive”, “commissive/ directive”, “starter/ directive”.

A *domain* is either a finite subset of individual markers, or a couple (usual operator, individual marker); for instance: {Arthur} is a domain; {1, 2, 3} is a domain; (>=, 3) is the domain corresponding to all numbers greater or equal to 3.

A *word* is a 4-uple written: [*role* : *concept* : *domain* : *word status*] where the domain must match the concept type; for instance [row : XY : {2, 3, 5, 6} : 0] is the definitive word meaning: “row XYcoordinate is in {2, 3, 5, 6}”.

A *sentence* is an expression ($word_1$. $word_2$ … $word_n$; sentence type) built upon a finite set of words, where all roles are different (the order of words is indifferent), and a sentence type. Sentence types AA or DD imply that all word status equal ‘0’; sentence types CA or CD imply that at least one word status equals ‘1’.

Let us come back to the sentences we desired to express:

assertive1: “there exists one value among {2, 3, 5, 6} which solves the cell (row=1, col =2)” is expressed by ([accepted : Value : {2, 3, 5, 6} : 0] . [row : XY : {1} : 0] . [column : XY : {2} : 0]; AA); all words are definitive.

assertive2: “the value ‘2’ solves the cell (row=1, column=2)” is expressed by ([accepted : Value : {2} : 0] . [row : XY : {1} : 0] . [column : XY : {2} : 0]; AA); all words are definitive.

directive1: “what are all the values rejected for the cell (row=1, column=2)?” is expressed by ([rejected : Value : ALL : 0] . [row : XY : {1} : 0] . [column : XY : {2} : 0]; DD); all words are definitive.

commissive1: “one value among {2, 3, 5, 6} which solves the cell (row=1, column=2) will be later asserted” is expressed by ([accepted : Value : {2, 3, 5, 6} : 1] . [row : XY : {1} : 0] . [column : XY : {2} : 0]; CA); the word [accepted : Value : {2, 3, 5, 6} : 1] is a non definitive word to be replaced by a definitive word for CA to become AA.

commissive2: “the values rejected for the cell (row=1; column=x) will be later requested” is expressed by ([rejected : Value : ALL : 0] . [row : XY : {1} : 0] . [column : XY : ALL: 1]; CD) ; the word [column : XY : ALL: 1] is the commitment (non definitive, to be replaced by a definitive word for CD to become DD)

In order to make the reading of sentences more “natural”, we further adopted in the prototype two conventions:

- any domain with a single element {Arthur} is written ‘Arthur’
- the Boolean 0 is omitted, and the Boolean 1 is written ‘*’

With these two conventions, we can write:

there exists one value among {2, 3, 5, 6} which solves the cell (row=1, column=2)	([accepted:Value:{2, 3, 5, 6}] . [row: XY:1] . [column:XY:2]; AA)
the value ‘2’ solves the cell (row=1, column=2)	([accepted:Value:2] . [row: XY:1] . [column:XY:2]; AA)
what are all the values rejected for the cell (row=1, column=2)?	([rejected:Value:ALL] . [row:XY:1] . [column:XY:2]; DD)
one value among {2, 3, 5, 6} which solves the cell (row=1, column=2) will be later asserted	([accepted:Value:{2, 3, 5, 6}:*] . [row: XY:1] . [column:XY:2]; CA)
the values rejected for the cell (row=1; column=x) will be later requested	([rejected:Value:ALL] . [row:XY:1] . [column:XY: ALL:*]; CD)

Only sentences of the types AA, CA, DD, CD i. e. *assertives*, *directives* and *commissives* will be uttered during the conversation.

SA, SD will be *starters* belonging to nodes in conversational patterns.

3.7.2 Hints about the logical semantics

Let us take the following sentences emitted during a conversation, for each of which an intuitive translation is given:

s_1: ([accepted:Value: ALL:*] . [row: XY: ALL:*]. [column:XY: ALL:*]; CA)

"some values solving some cells will be later asserted"

s_2: ([accepted:Value:2] . [row: XY:1] . [column:XY:2]; AA)

"2 solves the cell (row=1, column=2)"

s_3: ([accepted:Value: ALL] . [row: XY:1] . [column:XY:2]; DD)

"which are the values solving the cell (row=1, column=2)?"

s_4: ([accepted:Value: ALL] . [row: XY: ALL] . [column:XY: ALL]; DD)

"which are all solving triples (value, row, column)?"

s_5: ([accepted:Value: ALL:*] . [row: XY: ALL:*] . [column:XY: ALL:*]; CD)

"solving (some values, some rows, some columns) will be later requested"

Let us now consider two starters, one based on an assertive, the other on a directive:

s_6: ([accepted:Value: ALL:*]. [row: XY ALL:*]. [column:XY ALL:*]; SA)

s_7: ([accepted:Value: ALL:*]. [row: XY ALL:*]. [column:XY ALL:*]; SD)

According to the rules of the calculus, sentences will be substituted by "reductionsOf" themselves; the intuitive meaning of this relation formally defined in [3.9.3] being "more precise (assertive or directive) than". For instance s_2 is a more precise assertive than s_1. In a similar manner; s_3 or s_4 are more precise directives than s_5.

Analyzing s_1, the "moderator" of the conversation "knows" that s_2 may be further emitted; analyzing s_5, it "knows" that s_3 or s_4 may be further emitted.

We intuitively understand that s_2 is a reply to s_3; it is also a partial reply to s_4. In order to check that the calculus has provided all replies to s_4, the "moderator" of the conversation must analyze whether some more replies may come from other invoked patterns: as long as s_1 is present, it will not answer s_4.

Let us now suppose that s_6 (resp. s_7) is the starter of a conversational pattern; we intuitively understand that s_2 (resp. s_3 or s_4) can play the role of invocation context … and that as long as s_1 (resp. s_5) is present, this conversational pattern may be further invoked.

3.7.3 Back to the decision problems:

In 3.3.1 we wrote:

> Elements of $\mathcal{S}$ are *sentences*. And we suppose that for each couple (s_1, s_2) of sentences, the following decision problems always have an answer in $\mathcal{S}$:
>
> - problem1: "$s_1 \leq s_2$"
> - problem2: "s_1 triggerFor s_2"
> - problem3: "s_1 answerFor s_2"
> - problem4: "$(\exists\ s_1')\ ((s_1' \leq s_1) \wedge (s_1'$ triggerFor $s_2))$"
> - problem5: "$(\exists\ s_1')\ ((s_1' \leq s_1) \wedge (s_1'$ answerFor $s_2))$"
> - problem6: "$(\exists\ s_2')\ ((s_1$ answerFor $s_2') \wedge (s_2' \leq s_2))$"

We give here, in a semi-formal way, the definitions for '$s_1 \leq s_2$', 's_1 triggerFor s_2', 's_1 answerFor s_2', and briefly describe how the set $\mathcal{S}$ described in 3.9.1 respects these requirements. This technical part has been kept concise and demonstrations are not extensively given; the intention is only to make the reader understand what kind of comparisons have to be undertaken during the calculus in order to support the logical semantic outlined in [3.9.2].

1) $\mathbf{w_1 \leq w_2}$, where $w_1 = [\ role_1 : concept_1 : domain_1 :$ word $status_1\ ]$ and $w_2 = [\ role_2 : concept_2 : domain_2 :$ word $status_2\ ]$ are words, is defined by the conjunction of:

- $role_1 = role_2$;
- $concept_1 \leq concept_2$;
- $domain_1 \subseteq domain_2$;
- word $status_1$ <= word $status_2$.

2) $\mathbf{w_1 \cap w_2 \neq \emptyset}$, where $w_1 = [\ role_1 : concept_1 : domain_1 :$ word $status_1\ ]$ and $w_2 = [\ role_2 : concept_2 : domain_2 :$ word $status_2\ ]$ are words, means that one of these three cases is met:

- case1: $\mathbf{w_1 \leq w_2}$
- case2: $\mathbf{w_2 \leq w_1}$
- case3: ($role_1$=$role_2$) and (word $status_1$=word $status_2$=1) and ($concept_1 \cap concept_2 \neq \emptyset$) and ($domain_1 \cap domain_2 \neq \emptyset$)

$\mathbf{w_1 \cap w_2 \neq \emptyset}$ is equivalent to "there exists $\mathbf{w_3}$ verifying $\mathbf{w_3 \leq w_1}$ and $\mathbf{w_3 \leq w_2}$"

3) $\mathbf{s_1 \leq s_2}$ is defined by the conjunction of:

- one of the following cases is met: (s_1 is AA and s_2 is CA) or (s_1 is CA and s_2 is CA) or (s_1 is DD and s_2 is CD) or (s_1 is CD and s_2 is CD)

- s_1 and s_2 have exactly the same roles;
- let w_1 and w_2 two words with the same role belonging respectively to s_1 and s_2 , then: $w_1 \leq w_2$

$\mathbf{s_1} \leq \mathbf{s_2}$ is equivalent to '$\mathbf{s_1}$ reductionOf $\mathbf{s_2}$'; it intuitively means that $\mathbf{s_1}$ implies $\mathbf{s_2}$.

4) $\mathbf{s_1} \Rightarrow \mathbf{s_2}$ is defined by:

- for each word $\mathbf{w_2}$ in s_2, there exists a word $\mathbf{w_1}$ in s_1 verifying $w_1 \leq w_2$;

$\mathbf{s_1} \Rightarrow \mathbf{s_2}$ means that $\mathbf{s_1}$ consists in a reduction of $\mathbf{s_2}$ completed with additional words; it intuitively means that $\mathbf{s_1}$ implies $\mathbf{s_2}$.

5) $\mathbf{s_1} \rightarrow \mathbf{s_2}$ is defined by:

- for each word $\mathbf{w_2}$ in s_2, there exists a word $\mathbf{w_1}$ in s_1 verifying $w_1 \cap w_2 \neq \emptyset$;

We know from 2) that for each word $\mathbf{w_2}$ in s_2, there exists a word $\mathbf{w_1}$ in s_1 and a word $\mathbf{w_3}$ verifying $\mathbf{w_3} \leq \mathbf{w_1}$ and $\mathbf{w_3} \leq \mathbf{w_2}$; if we replace all the $\mathbf{w_1}$ by the $\mathbf{w_3}$, we obtain $\mathbf{s_3}$ verifying $\mathbf{s_3} \leq \mathbf{s_1}$ and $\mathbf{s_3} \Rightarrow \mathbf{s_2}$; therefore $\mathbf{s_1} \rightarrow \mathbf{s_2}$ means that $\mathbf{s_1}$ can be reduced into a sentence implying $\mathbf{s_2}$.

6) $\mathbf{s_1}$ triggerFor $\mathbf{s_2}$ means that one of the two following cases is met:

- (s_1 is AA and s_2 is SA) and ($s_1 \Rightarrow s_2$)
- (s_1 is DD and s_2 is SD) and ($s_1 \leq s_2$)

7) $\mathbf{s_1}$ answerFor $\mathbf{s_2}$ is defined by:

- (s_1 is AA and s_2 is DD) and ($s_1 \Rightarrow s_2$);

We may now briefly review in Tab le 10 the decision problems listed in 3.3.1.:

problem$_0$: "$w_1 \leq w_2$"	Decidable because of hypotheses on concepts and definition of domains
problem$_{0'}$: "$w_1 \cap w_2 \neq \emptyset$"	Decidable because of hypotheses on concepts and definition of domains
problem$_1$: "$s_1 \leq s_2$"	Decidable because sentences are finite and problem$_0$ decidable
problem$_{1'}$: "$s_1 \Rightarrow s_2$"	Decidable because sentences are finite and problem$_0$ decidable
problem$_{1''}$: "$s_1 \rightarrow s_2$"	Decidable because sentences are finite and problem$_{0'}$ decidable
problem$_2$: "s_1 triggerFor s_2"	Decidable because problem$_1$ decidable and problem$_{1'}$ decidable

problem$_3$: "s_1 answerFor s_2"	Decidable because problem$_1$,, decidable
problem$_4$: "$(\exists\ s_1')\ ((s_1' \leq s_1) \wedge (s_1'$ triggerFor $s_2))$"	Four decidable cases: - (s_1 is AA) and (s_2 is SA) and (s_1 triggerFor s_2) - (s_1 is DD) and (s_2 is SD) and (s_1 triggerFor s_2) - (s_1 is CA) and (s_2 is SA) and ($s_1 \rightarrow s_2$) - (s_1 is CD) and (s_2 is SD) and ($s_1 \rightarrow s_2$) and ($s_2 \rightarrow s_1$)
problem$_5$: "$(\exists\ s_1')\ ((s_1' \leq s_1) \wedge (s_1'$ answerFor $s_2))$"	Two decidable cases: - (s_1 is AA) and (s_2 is SD) and (s_1 triggerFor s_2) - (s_1 is CA) and (s_2 is SD) and ($s_1 \rightarrow s_2$)
problem6: "$(\exists\ s_2')\ ((s_1$ answerFor $s_2') \wedge (s_2' \leq s_2))$"	Two decidable cases: - (s_1 is AA) and (s_2 is DD) and ($s'_2 = s_2$) and (s_1 answerFor s_2) - (s_1 is AA) and (s_2 is CD) and ($s_1 \Rightarrow s_2$)

Tab le 10: Decidable questions related to sentences

3.8 The 'Sudoku filling problem'

When we left the 'Sudoku filling problem' in 2.2.4., there were three conversational patterns were in charge of the following puzzle:

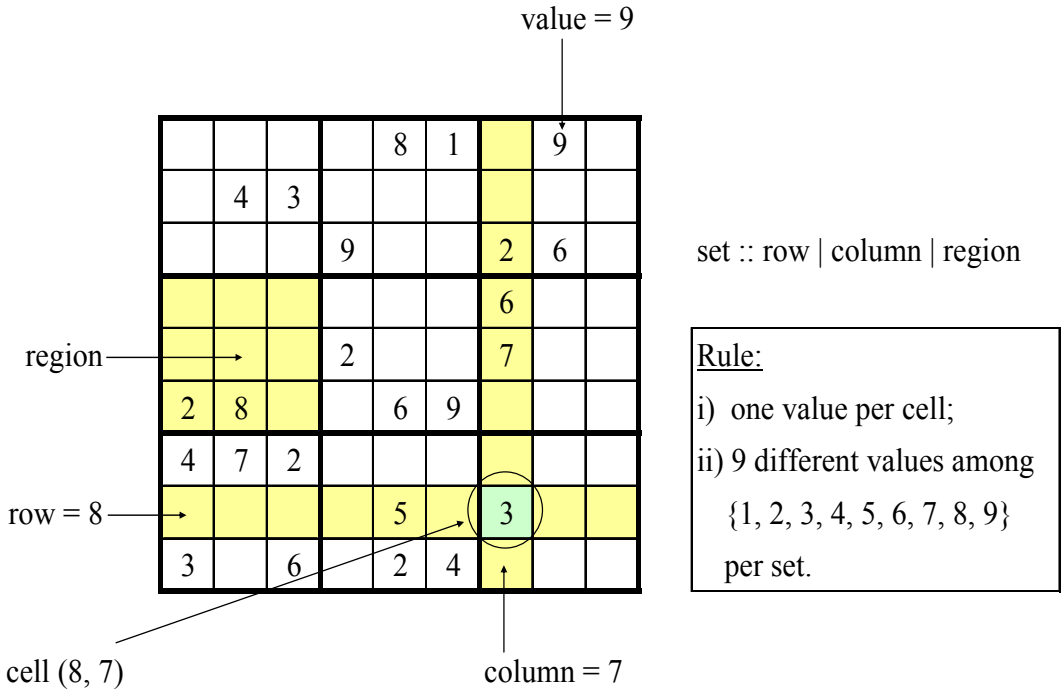


Figure 33 : Sudoku n° 53 [Le Monde]

These conversational patterns were:

- LAUNCHER, in charge of invoking all the others, and presenting the grid resulting from the conversation;
- CELL, in charge of reasoning from the viewpoint of a single cell;
- SET, in charge of reasoning from the viewpoint of a set of 9 cells belonging either to a row or a column or a region[64].

3.8.1 A bunch of concepts and roles

We propose to describe their conversation using the language defined in [3.7]. We only need five concepts; they are presented in a meet semi-lattice in Figure 34:

[64] a 'region' is a 3x3 square appearing on the grid with a stronger outline

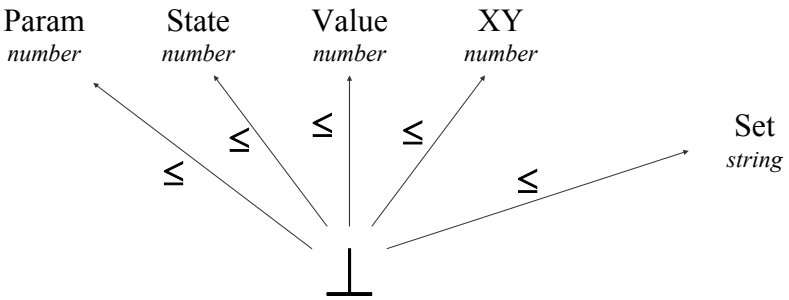


Figure 34: five concepts for the 'Sudoku filling problem'

Table 11 gives informal definitions of the concepts and outlines how there will be conjugated with roles within he words of the sentences:

Concept	will be conjugated with the following:	Role
param	A 9x9 grid will be said to correspond to ‘Param with the role dimension’ = 3; the maximum number of states to explore will be described by ‘Param with the role max’.	dimension max
state	A new ‘state’ is reached when all invocations of the conversational patterns have computed the information available in previous ‘state’ of the grid. State = 0 is the initial state of the grid.	when
value	A possible filling value	accepted rejected test
XY	‘XY’ comes with two roles which are ‘row’ and ‘column’	column row
set	Rows will be denoted (x, 0); Columns will be denoted (0, y); Regions will be denoted (i, j) where ‘i’ and ‘j’ are the coordinates of the little square in the big one, starting from 1.	test

Table 11: informal definitions of the concepts, in relation with the roles

The language used by the conversational abstract machine uses the typology of sentences listed in Table 12:

AA	assertive = (definitive) assertion
DD	directive = (definitive) question
SA	starter waiting for assertive (not emitted during the conversation)
SD	starter waiting for directive (not emitted during the conversation)
CA	commissive to be rewritten into assertive = non-definitive assertion
CD	commissive to be rewritten into directive= non-definitive question

Table 12: types of sentences

3.8.2 Assertives describing the initial grid

We are already able to describe the 24 filled cells of our initial grid, by as many *assertives* linking together the concepts 'Value', 'XY', 'State':

([accepted:value:8].[column:xy:5].[row:xy:1].[when:state:0]; AA)

([accepted:value:1].[column:xy:6].[row:xy:1].[when:state:0]; AA)

([accepted:value:9].[column:xy:8].[row:xy:1].[when:state:0]; AA)

…

We note that a concept like 'XY' may participate to several words of the same sentence, but each role is unique!

This description is completed by:

- asserting 'dimension = 3', which will be translated in 'number of cells = 3^4 = 81' (our convention is that a grid with 2^4 cells is of dimension '2' and is called a 2-sudoku, a grid with 3^4 cells is called a 3-sudoku … the conversational patterns are generic and would be able to work on 2-sudokus, 3-sudokus, 4-sudokus … provided a powerful enough machine is available)
- asserting a parameter 'max = m', indicating that only states '0', '1' … 'm' are to be computed . We know that since 24 values are initially given, the theoretical maximum for states to be computed is 81-24=57[65]; we also know that 81 cells and 27 sets will be invoked at each state.

We choose to assert both in a single sentence, emitted by the user at the beginning. Because of the great flexibility of the language, this information can be provided in many ways; the following one will be convenient for invoking the **launcher**.

([dimension:param:3].[max:state:2]; AA)

[65] if the computation of all information available at a given state gives no 'new accepted cell' then the immediately following state will be strictly identical …

3.8.3 Starters and initial commissives

We can now comment the language of each of the three conversational patterns. The reader is kindly requested to pay a special attention to where the "*" are: it is there that further re-writing will occur.

LAUNCHER: its invocation is ruled by the abstract machine through comparison between available assertives and the starter SAlauncher. It enters the conversation together with 3 commissives. CA01 will be rewritten into assertives invoking **Cell**; CA02 will be rewritten into assertives invoking **Set**; CD03 will be rewritten into a directive asking for accepted values for all couples (row, column) at all states.

SAlauncher = ([max:State:ALL:*].[dimension:Param: ALL:*]; SA)

"pattern is triggered by assertions about max state and dimension"

CA01 = ([test:Value: all].[row:XY: all:*].[column:XY: all:*].[dimension:Param: all:*].[when:State: all:*]; ca)

"contexts (some rows, some columns, some dimensions, some states) where all values must be tested will be later asserted"

CA02 = ([test:Set: all:*].[dimension:Param: all:*].[when:State: all:*]; ca)

"contexts (some sets, some dimensions, some states) will be later asserted"

CD03 = ([accepted:Value: all].[row:XY: all].[column:XY: all].[when:State: all:*]; cd)

"all solving triples (value, row, column) will be later requested for some state"

CELL: its invocation is ruled by the abstract machine through comparison between available assertives and the starter SAcell. It enters the conversation together with 4 commissives. CA06 will be rewritten into assertives accepting values; CA07 will be rewritten into assertives rejecting values; CD08 will be rewritten into a directive requesting accepted values for the current couple (row, column) at all states; CD09 will be rewritten into a directive requesting rejected values for the current couple (row, column) at all states.

SAcell = ([test:Value: ALL:*].[row:XY: ALL:*].[column:XY: ALL:*].[dimension:Param: ALL:*].[when:State: ALL:*]; SA)

"pattern is triggered by assertions about values, rows and columns to be tested under some dimension and in some state of the solving process"

CA06 = ([accepted:Value: ALL:*].[row:XY: ALL:*].[column:XY: ALL:*].[when:State: ALL:*]; CA)

"some accepted values for some rows and columns in some state of the solving

process will be later asserted"

CA07 = ([rejected:Value: ALL:*].[row:XY: ALL:*].[column:XY: ALL:*].[when:State: ALL:*]; CA)
"some rejected values for some rows and columns in some state of the solving process will be later asserted"

CD08 = ([accepted:Value: ALL].[row:XY: ALL:*].[column:XY: ALL:*].[when:State: ALL:*]; CD)
"some accepted values for some rows and columns in some state of the solving process will be later requested"

CD09 = ([rejected:Value: ALL].[row:XY: ALL:*].[column:XY: ALL:*].[when:State: ALL:*]; CD)
"some rejected values for some rows and columns in some state of the solving process will be later requested"

SET: its invocation is ruled by the abstract machine through comparison between available assertives and the starter SAset. It enters the conversation together with the same 4 commissives. CA06 will be rewritten into assertives accepting values; CA07 will be rewritten into assertives rejecting values; CD08 will be rewritten into directives requesting accepted values for the couples (row, column) belonging to the current set; CD09 will be rewritten into a directive requesting rejected values for the couples (row, column) belonging to the current set.

SAset = ([test:Set: ALL:*].[dimension:Param: ALL:*].[when:State: ALL:*]; SA)
"pattern is triggered by assertions about sets to be tested under some dimension and in some state of the solving process"

CA06 = ([accepted:Value: all:*].[row:XY: all:*].[column:XY: all:*].[when:State: all:*]; ca)

CA07 = ([rejected:Value: all:*].[row:XY: all:*].[column:XY: all:*].[when:State: all:*]; ca)

CD08 = ([accepted:Value: all].[row:XY: all:*].[column:XY: all:*].[when:State: all:*]; cd)

CD09 = ([rejected:Value: all].[row:XY: all:*].[column:XY: all:*].[when:State: all:*]; cd)

3.8.4 Inside the nodes of the conversational patterns

Each pattern consists of several nodes, 'driven by eachother' according to dependency to information asked through directives. For each pattern, we list and briefly describe the different nodes in the following way:

- name of the node
- list of its drivers (nodes from the same pattern)
- input: either triggerFor (starter) or answersFor (list of directives)
- output in the form of commissives, assertives and directives

conversational pattern: LAUNCHER

starter:

SAlauncher = ([max:State:ALL:*].[dimension:Param: ALL:*]; SA)

commissives:

CA01 = ([test:Value: ALL].[row:XY: ALL:*].[column:XY: ALL:*].[dimension:Param: ALL:*]. [when:State: ALL:*]; CA)

CA02 = ([test:Set: ALL:*].[dimension:Param: ALL:*].[when:State: ALL:*]; CA)

CD03 = ([accepted:Value: ALL].[row:XY: ALL].[column:XY: ALL].[when:State: ALL:*]; CD)

startNode: LAUNCHER-1

drivers: none

input: triggerFor (SAlauncher) contains the information dim= dimension and maximum number of states; from which the indexes for {row^{i}}, {column^{j}}, {set^{k}} and {state^{n}} can be computed.

output: assertives for Cell and Set invocation; and directive for gathering results replace the commissives.

CA011$^{i, j, n}$ = ([test:Value: ALL].[row:XY:**row**^{i}].[column:XY:**column**^{j}].[dimension:Param:**dim**]. [when:State: **state**^{n}]; AA)

CA021$^{k, n}$ = ([test:Set: **set**^{k}].[dimension:Param: **dim**].[when:State: **state**^{n}]; AA)

DD030 = ([accepted:Value: ALL].[row:XY: ALL].[column:XY: ALL].[when:State: **>0**]; DD)

postNode: LAUNCHER-2

drivers: LAUCHER-1

input: answersFor (DD030)

output: no sentences are output, but a recapitulation of the new state of the grid is provided to the user.

conversational pattern: CELL

starter:

SAcell = ([test:Value: ALL:*].[row:XY: ALL:*].[column:XY: ALL:*].[dimension:Param: ALL:*]. [when:State: ALL:*]; SA)

commissives:

CA06 = ([accepted:Value: ALL:*].[row:XY: ALL:*].[column:XY: ALL:*].[when:State: ALL:*]; CA)

CA07 = ([rejected:Value: ALL:*].[row:XY: ALL:*].[column:XY: ALL:*].[when:State: ALL:*]; CA)

CD08 = ([accepted:Value: ALL].[row:XY: ALL:*].[column:XY: ALL:*].[when:State: ALL:*]; CD)

CD09 = ([rejected:Value: ALL].[row:XY: ALL:*].[column:XY: ALL:*].[when:State: ALL:*]; CD)

startNode: CELL-1

drivers: none

input: triggerFor (SAcell) contains the information row^a, $column^b$ and $state^n$ corresponding to the invocation; a and b which index row and column are constants.

output: commissives and directives replacing the initial commissives; it should be noted that Cell-1 asks questions about states previous to $state^n$ and commits itself to provide information at $state^{n+1}$.

CA061 = ([accepted:Value: ALL:*].[row:XY:**row**$^{\mathbf{a}}$].[column:XY:**column**$^{\mathbf{b}}$]. [when:State: **state**$^{\mathbf{n+1}}$]; CA)

CA071 = ([rejected:Value: ALL:*].[row:XY:**row**$^{\mathbf{a}}$].[column:XY:**column**$^{\mathbf{b}}$]. [when:State: **state**$^{\mathbf{n+1}}$]; CA)

DD080 = ([accepted:Value: ALL].[row:XY: **row**$^{\mathbf{a}}$].[column:XY: **column**$^{\mathbf{b}}$].[when:State: **<= state**$^{\mathbf{n}}$]; DD)

DD090 = ([rejected:Value: ALL].[row:XY: **row**$^{\mathbf{a}}$].[column:XY: **column**$^{\mathbf{b}}$].[when:State: **<= state**$^{\mathbf{n}}$]; DD)

postNode: CELL-2

drivers: CELL-1

input: answersFor (DD080 & DD090) which contain the information about the eventual existence of a value accepted before $state^n$ for the cell; as well as about values rejected before $state^n$.

output: assertives describing accepted values for the cell, as well as rejected values for its neighbours $\{(row^p, column^q\})$ at $state^{n+1}$; p and q which index row and column for the neighbours indicate a matrix.

AA0610 = ([accepted:Value: **v**$^{\mathbf{a,\ b}}$].[row:XY:**row**$^{\mathbf{a}}$].[column:XY:**column**$^{\mathbf{b}}$]. [when:State: **state**$^{\mathbf{n+1}}$]; AA)

$AA0710^{p,q}$ = ([rejected:Value: **v**$^{\mathbf{a,\ b}}$].[row:XY:**row**$^{\mathbf{p}}$].[column:XY:**column**$^{\mathbf{q}}$]. [when:State: **state**$^{\mathbf{n+1}}$]; AA)

conversational pattern: SET

starter:

SAcell = ([test:Set: ALL:*].[dimension:Param: ALL:*].[when:State: ALL:*]; SA)

commissives:

CA06 = ([accepted:Value: ALL:*].[row:XY: ALL:*].[column:XY: ALL:*].[when:State: ALL:*]; CA)

CA07 = ([rejected:Value: ALL:*].[row:XY: ALL:*].[column:XY: ALL:*].[when:State: ALL:*]; CA)

CD08 = ([accepted:Value: ALL].[row:XY: ALL:*].[column:XY: ALL:*].[when:State: ALL:*]; CD)

CD09 = ([rejected:Value: ALL].[row:XY: ALL:*].[column:XY: ALL:*].[when:State: ALL:*]; CD)

startNode: SET-1

drivers: none

input: triggerFor (SASet) contains the information set^{k}, and state^{n} corresponding to the invocation.

output: the set asks questions related to its own cells {(row^{i}, column^{j}}); i and j which index row and column indicate a matrix.

CA061$^{i, j}$ = ([accepted:Value: ALL:*].[row:XY:**row^{i}**].[column:XY:**column^{j}**]. [when:State: **state^{n+1}**]; CA)

CA071$^{i, j,}$ = ([rejected:Value: ALL:*].[row:XY:**row^{i}**].[column:XY:**column^{j}**]. [when:State: **state^{n+1}**]; CA)

DD0810 $^{i, j}$ = ([accepted:Value: ALL].[row:XY: **row^{i}**].[column:XY: **column^{j}**].[when:State: **<= state^{n}**]; DD)

postNode: SET-2

drivers: SET-1

input: answersFor (listDD0810 $^{i, j}$) which contain the information about which cells in the set are still waiting for a value.

output: directives asking for rejected values related to those cells still waiting.

DD0910 $^{p, q}$ = ([rejected:Value: ALL].[row:XY: **row^{i}**].[column:XY: **column^{j}**].[when:State: **<= state^{n}**]; DD)

postNode: SET-3

drivers: SET-1, SET-2

input: answersFor (listDD0810 $^{i, j}$ & listDD0910 $^{p, q}$) which contain the information about free cells and restricted values for them.

output: accepted values for the cells of the set, as well as rejected values for their neighbours {(row^{p}, column^{q}}) at state^{n+1}; i and j which index row and column for the cells and p and q which index row and column for the neighbours indicate a matrix.

AA0610$^{i, j}$ = ([accepted:Value: **v$^{i, j}$**].[row:XY:**row^{i}**].[column:XY:**column^{j}**]. [when:State: **state^{n+1}**]; AA)

AA0710$^{p, q}$ = ([rejected:Value: **v$^{i, j}$**].[row:XY:**row^{p}**].[column:XY:**column^{q}**]. [when:State: **state^{n+1}**]; AA)

3.8.5 The 'Sudoku filling' conversation

Figure 35 demonstrates interaction and causal dependencies between conversation nodes. Five invocations of conversational patterns are represented:

- the unique invocation of LAUNCHER with two nodes Launcher-1 and Launcher-2;
- an invocation of CELL (among 81) with two nodes Cell-1 and Cell-2, at state^{n};
- an invocation of SET (among 27) with three nodes Set-1, Set-2 and Set-3, at state^{n};
- an invocation of CELL at state^{n+1};
- an invocation of SET at state^{n+1};

The semantic dependencies are illustrated by arrows:

- thick dotted arrows mean "triggered by";
- thin curved arrows mean "answered by".

Laucher-1 starts first and is labelled with rank '1'. By emitting the $CA011^{i,\,j,\,n}$ & $CA021^{k,\,n}$, Laucher-1 invokes simultaneously 81×Cell-1 and 27×Set-1 for all states; Cell-1 and Set-1 are therefore labelled with rank '2' in all states; invocations are illustrated by red dotted arrows

At state^{n}, Cell-2 and Set-2 respectively need answers to DD080 & DD090 and $listDD0810^{i,\,j}$ which are related to state^{n}; since all the present nodes assert about state^{n+1} this creates no dependencies: they are labelled with rank '3'and Set-3 needing answers to $listDD0910^{p,\,q}$ emitted by Set-2 is labelled with rank '4'.

But then we have semantic dependencies corresponding to the answering of questions:

- Cell-2 at state^{n+1} must wait for assertives $AA0610^{i,\,j}$ and $AA0710^{p,\,q}$ coming from Cell-2 and Set-3 at state^{n}; Cell-2 at state^{n+1} is labelled with rank '5'.
- Set-2 at state^{n+1} must wait for assertives $AA0610^{i,\,j}$ coming from Cell-2 and Set-3 at state^{n}; Set-2 at state^{n+1} is labelled '4'. Set-3 at state^{n+1} is labelled with rank '5'.
- Launcher-2 must wait for answers coming from all states and is therefore labelled with rank '6'.
- This discussion shows that, although all patterns have been invoked simultaneously, the successive nodes will "take turn" according to the semantic dependencies between sentences which is coded in the comparison algorithms described in [3.7.3].

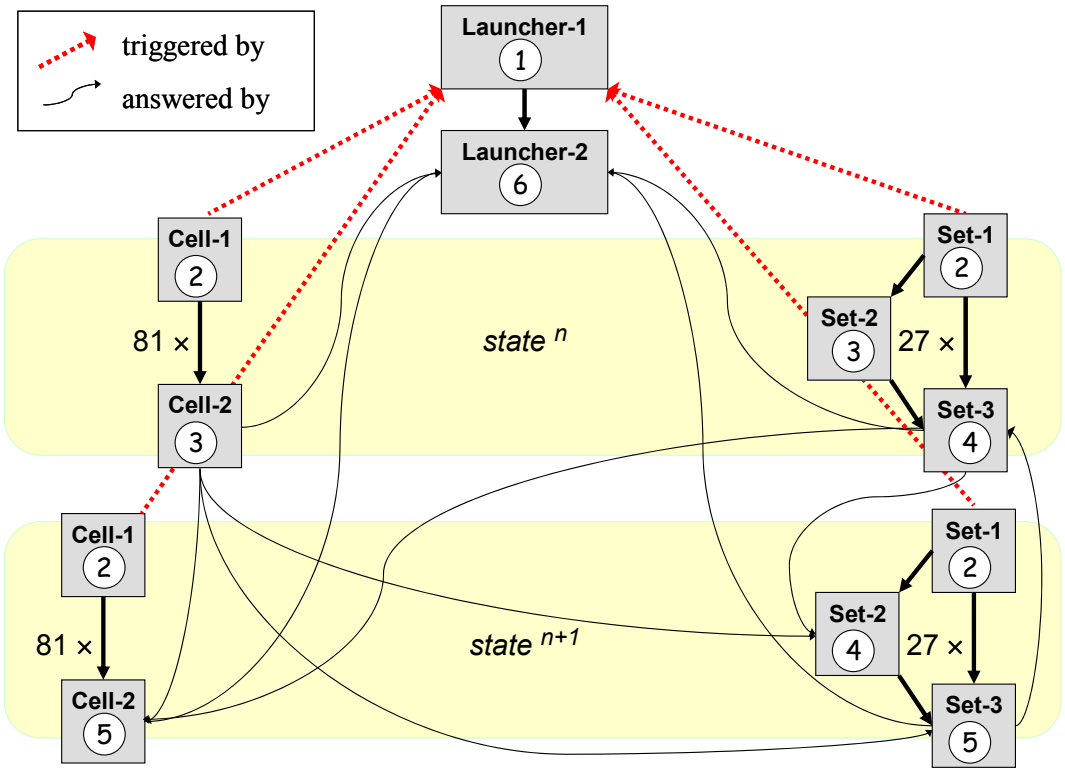


Figure 35: constraints in turn taking during the 'Sudoku filling' conversation

The overall conversation resulting of 'context$^{0'}$ (dimension, m^0= maximum number of states) might for instance respect the following chronology:

1. Launcher is invoked under 'context$^{0'}$;
2. Launcher is revoked;
3. Launcher-1 (context^0) invokes Cell and Set under (81+27) × m^0 respective contexts;
4. Cell and Set are revoked;
5. Cell-1 and Set-1 steps are executed;
6. Cell-2 and Set-2 steps are executed for state$^{(0)}$;
7. Set-3 steps are executed for state$^{(0)}$;
8. Cell-2 and Set-2 steps are executed for state$^{(1)}$;
9. Set-3 steps are executed for state$^{(1)}$;
10. ….

The above chronology is only one possibility among many others left open according to the parallel races taking place in the "test pipe" and "transition pipe" of Figure 31. However, some *transitions* **must** occur before those which are causally related through the *tests*:

- 1.<2.; 2.<4.; 3.<4. ; 3.<5.
- for a given Cell 5.<6.<8.<10…
- for a given Set 5.<7.<9…
- Launcher-2 (context^{0}) will work last

We may therefore have 3. < 2. or 5. < 4. Moreover, going more precisely into the relations between cells might reveal that some invocations of Cell at state$^{(7)}$ may reach the step Cell-2 while others are still at step Cell-1. We know from Theorem 2 that the final results are not affected by this indeterminacy, which may therefore be called "hidden indeterminacy".

4 Compositional programming in AUSTIN

Relying on the concepts of the conversational calculus of Chapter 3, we now wish to compose partial theories embedded in logical artefacts. To do so, we are going to first define as "programs" and then study the logical artefacts that the AUSTIN Conversational Abstract Machine will run.

In [4.1] we summarize the elements of Chapter 3 necessary for this chapter.

In [4.2] we define "programs in AUSTIN", and briefly explain the natural programming techniques we will use in the examples.

In [4.3] we demonstrate the composition of Programs, which directly addresses one of the main goals stated in our introduction.

In [4.4], we address the reasoning techniques and give elements concerning the analysis of the final states of computations.

In [4.5], we illustrate the compositionality by coming back to the 'Sudoku filling problem'.

4.1 Main concepts of the conversational calculus

Conversations mainly consist of *utterances* reflecting the following illocutionary acts:

Illocutionay act	**Utterance**
commissive to be re-written in an assertive	non-definitive question
commissive to be re-written in an directive	non-definitive assertion
assertive	(definitive) assertion
directive	(definitive) question

These *utterances* are couples (*sentence*, status) where:

- *sentence* belongs to an infinite set ($\mathcal{S}$, reductionOf, triggerFor, answerFor); sentences can be either assertions or questions;
- status $\in \{0, 1\}$; status = 0 means definitive; status = 1 means non-definitive.

Utterances are emitted and computed by *conversational patterns*. Conversational patterns are multi-steps algorithms built upon meet semi-lattices of *conversation nodes* which are the "information processing units". Each pattern owns a special sentence called *starter* which plays the role of triggering condition and a finite set of sentences initially emitted as *commissives*; all future utterances of the pattern will be reductionOf these commissives.

A *conversation* is defined by a *conversation frame* denoted 'X' consisting in a set of conversational patterns and an initial set $\mathcal{U}$ (☺) of utterances emitted by a 'user', as an input to be computed (e.g. the initial state of the 'Sudoku' grid). The progressive activation of concurrent conversational patterns invoked through different contexts produces an evolving set of *conversation steps* (node, context, status), where status $\in \{0, 1\}$; status = 0 means definitive; status = 1 means non-definitive.

Conversation states are pairs: $\omega = (\mathcal{U}(\omega), \mathcal{X}(X, \omega))$, where:

- X is a conversation frame;
- $\mathcal{U}(\omega)$ is the set of all present utterances, including the commissives of the patterns;
- $\mathcal{X}(X, \omega)$ is the set of all present steps.

In order to better describe $\mathcal{U}(\omega)$, we first need to define operations on sets of utterances:

Definition 9a:

Let $\mathcal{U}_1 = \{(u_1^i, \{e_1^{i,j}\}_{j\in J(i)})\}_{i\in I}$ a set of utterances u_1^i, each with the list of emitters $\{e_1^{i,j}\}_{j\in J(i)}$,

Let $\mathcal{U}_2 = \{(u_2^m, \{e_2^{m,p}\}_{p\in P(m)})\}_{m\in M}$ a set of utterances u_2^m, each with the list of emitters $\{e_2^{m,p}\}_{p\in P(m)}$,

We denote $\mathcal{U}$ = **add** ($\mathcal{U}_1$, $\mathcal{U}_2$) i.e. "add $\mathcal{U}_1$ to $\mathcal{U}_2$" the set of utterances obtained by the following succession of operations defined in [Definition 3]:

$\mathcal{U} = \emptyset$

{**add** $(u_1^i, e_1^{i,j})\}_{i\in I, j\in J(i)}$ i.e. "addition of the $\mathcal{U}_1$ utterances"

{**add** $(u_2^m, e_2^{m,p})\}_{m\in M, p\in P(m)}$ i.e. "addition of the $\mathcal{U}_2$ utterances"

add is commutative and associative!

When $\mathcal{U}_1 \subset \mathcal{U}_2$, we denote $\mathcal{U}$ = **remove** ($\mathcal{U}_1$, $\mathcal{U}_2$) i.e. "remove $\mathcal{U}_1$ from $\mathcal{U}_2$" the set of utterances obtained by the following succession of operations defined in [Definition 3]:

$\mathcal{U} = \mathcal{U}_2$

{**remove** $(u_1^i, e_1^{i,j})\}_{i\in I, j\in J(i)}$ i.e. "removal of the $\mathcal{U}_1$ utterances"

Let $\mathcal{U}$ (X) be the set of initial *commissives* associated to X, we may now describe the initial state of a conversation by:

- $\mathcal{U}(\omega_i)$ = **add** ($\mathcal{U}$ (X), $\mathcal{U}$ (☺))
- $\mathcal{X}(\omega_i)$ = the set of *startSteps* associated to X

This conversation is automatically run by the abstract machine through the application of the transition rules; we shall denote it in the following way:

Let $\mathcal{X}(\omega_i)$ the set of *startSteps* associated to a *cycle-free conversation frame* [Definition 8c], let $\mathcal{U}(\omega_i)$ = **add** ($\mathcal{U}$ (X), $\mathcal{U}$ (☺)):

$$\omega_i \Rightarrow\Rightarrow \omega_f$$

denotes the exhaustive application of the 3 *transition rules* defined in 3.5.3, starting from ω_i = ($\mathcal{U}(\omega_i)$, $\mathcal{X}(\omega_i)$), until end of the process (cf. Theorem 1: the process ends) into a unique state ω_f = ($\mathcal{U}(\omega_f)$, $\mathcal{X}(\omega_f)$ (cf. Theorem 2: the final state is unique).

4.2 AUSTIN programs: notions of alert, result and residue

We are now going to shift from a vision in term of "conversation between patterns", to a vision in terms of "program execution" inside the conversational abstract machine:

Definition 9b:
Let X be a *conversation frame,* and $\mathcal{U}$ (☺) a set of initial definitive utterances emitted by a "user":
- let us denote $\mathcal{X}$(X) the set of *startSteps* associated to X
- let us denote $\mathcal{U}$ (X) the set of initial *commissives* associated to X
- let us denote $\mathcal{U}$ (☺ , X) = **add** ($\mathcal{U}$ (☺), $\mathcal{U}$ (X)),

We call *program* and denote (☺ , X) the conversation initial state: ω_i = ($\mathcal{U}$(☺ , X), $\mathcal{X}$ (X)).

If X is a *cycle-free conversation frame*, we may "derive" the initial state by applying the unique derivation rule:

$$\omega_i \Rightarrow\Rightarrow \omega_f = (\mathcal{U}(\omega_f), \mathcal{X}(\omega_f))$$

In the theory building process, the production and testing of a partial theory goes through the following steps:

1. likely hypotheses are embedded in one or several conversational patterns which together form a conversation frame X;
2. initial facts are described as a set of assertives;
3. the enquiry is described through one or more *assertives* or *directives* requesting some result of the computation;
4. the order to start is given …
5. … the abstract conversational machine derives the initial state until it halts on a final state and emits 'end';
6. the newly emitted *assertives* constitute the 'predicted results', logically deduced from the initial facts and patterns.

4. and 5. are illocutionary acts in a man-machine conversation; according to Searle and Austin, they are *declarations*[66]:

- '4.' is the declaration by the user that "conversation has started", which actually starts the conversation;
- '5.' is the declaration by the abstract machine that "conversation has come to an end", which actually end the conversation; it is sometimes coded as an "end" instruction in programs.

An extensive description of the conversation requires that borderline utterances are taken into account; we therefore add the sentences 'start' and 'end' to S to the grammar described in [3.9] without interfering with the calculus itself:

- the type 'DEC' for 'declarations' is added;
- the roles 'input' and 'output' and the concept 'Message' are added.

Illocutionay act	**Utterance**
declaration	(([input: Message: 'start']; DEC), 0, {USER}) (([output: Message: 'end']; DEC), 0, {AUSTIN})

[66] "In a declaration, the illocutionary point is to bring about a change in the world by representing it as having been changed. [Searle, 1999]"

4.2.1 Acceptability of programs

There are two conditions for a program to be successfully run by the abstract machine:

1. the conversation frame must be *cycle-free* : this will be checked in [4.4.1] by analyzing the directed graph defined on the conversation frame X by the relation "potentiallyTriggeredBy" between the *startNodes* ;
2. *reducers* embedded in conversation nodes must be correctly written[67] by the 'programmer'!

To these two conditions, we are going to associate two *expressives*[68], emitted by the abstract machine whenever the conditions are not fulfilled, and addressed to the 'user' just before halting (these are sometimes coded as 'exceptions' in programs).

Again, we indicate how the associated sentences can be added to the grammar of [3.9] in order to avoid interference with the calculus:

- the type 'EXP' for '*expressive*' is added;
- the role 'alert' is added.

Illocutionay act	**Utterance**
expressive	(([alert: Message: 'cycle in triggering!']; EXP), 0, {AUSTIN})
	(([alert: Message: 'invalid node!']; EXP), 0, {AUSTIN})

Definition 9c:
Let X be a *conversation frame,* and $\mathcal{U}$ (☺) a set of initial definitive utterances emitted by a "user".
Let us consider the program (☺ , X). The results of the global checking of cycles and of the local checking of reducers are *expressives* does not depend on $\mathcal{U}$ (☺); we store theses results in *Alerts* (∅, X)
(☺ , X) is an *acceptable program* if and only if *Alerts* (∅, X) = ∅.

[67] We have defined reducers as $2^A \times 2^B \rightarrow 2^B$ functions, it might happen that the node written by the 'programmer' does not accept all possible entries from $2^A \times 2^B$, or does not output in 2B; this must be locally checked either at execution or at compilation.

[68] "Examples of expressives are apologies, thanks, congratulations, welcomes, and condolences. [Searle, 1999]"

4.2.2 Residue and result of an acceptable program

Let (☺ , X) be an acceptable program:

$\omega_i = (\mathcal{U}(\text{☺}, X), \mathcal{X}(X)) \Rightarrow\Rightarrow \omega_f = (\mathcal{U}(\omega_f), \mathcal{X}(\omega_f))$

Among the steps $\mathcal{X}(\omega_f)$ are all those which have reached their definitive state and therefore have the status '0'; let us denote them $\mathcal{X}^0(\omega_f)$ and let us denote $\mathcal{X}^1(\omega_f)$ the set of final steps having kept the status '1'. It may happen that $\mathcal{X}^1(\omega_f) = \emptyset$: these are very good news indicating that each invocation has been allowed to reach its final state. It may also happen $\mathcal{X}^1(\omega_f) = \mathcal{X}(X)$, these are bad news indicating either that no *utterance* in $\mathcal{U}(\text{☺}, X)$ has played the role of trigger for any *startStep* of $\mathcal{X}(X)$ and in this case we also have $\mathcal{U}(\omega_f) = \mathcal{U}(\text{☺}, X)$.

The notions of *residue* and *result* aim at formalizing these notions:

Definition 9d:
Let X be a *conversation frame,* and $\mathcal{U}$ (☺) a set of initial definitive utterances emitted by a "user".
If (☺ , X) is an acceptable *program* the conversation can be run by the abstract machine:
$(\mathcal{U}(\text{☺}, X), \mathcal{X}(X)) \Rightarrow\Rightarrow \omega_f = (\mathcal{U}(\omega_f), \mathcal{X}(\omega_f))$

$\mathcal{U}(\omega_f) = \mathcal{U}^0(\omega_f) \cup \mathcal{U}^1(\omega_f)$ where *utterances* of $\mathcal{U}^0(\omega_f)$ have status = '0' and *utterances* of $\mathcal{U}^1(\omega_f)$ have status = '1'

$\mathcal{X}(\omega_f) = \mathcal{X}^0(\omega_f) \cup \mathcal{X}^1(\omega_f)$ where *steps* of $\mathcal{X}^0(\omega_f)$ have status = '0' and *steps* of $\mathcal{X}^1(\omega_f)$ have status = '1'

We therefore define:

Residue (☺ , X) = $(\mathcal{U}^1(\omega_f)), \mathcal{X}^1(\omega_f))$ is the part of the final state consisting in non-definitive utterances and non-activated steps

Result (☺ , X) = $(\mathcal{U}^0(\omega_f), \mathcal{X}^0(\omega_f))$ is the part of the final state consisting in definitive utterances and activated steps

4.2.3 Checking the acceptability before running the conversation

If programs are not acceptable, some alerts must be raised. We propose to integrate the preliminary checking for acceptability at the beginning of the derivation:

Definition 9e:
Let X be a *conversation frame,* and $\mathcal{U}$ (☺) a set of initial definitive utterances emitted by a "user".

Let us denote: $|\Rightarrow\Rightarrow$ the operation consisting in checking acceptability and then (eventually) computing the program:

$(\mathcal{U}(\text{☺}, X), \mathcal{X}(X)) \ |\Rightarrow\Rightarrow \ (Alerts\ (\varnothing, X), Residue\ (\text{☺}, X), Result\ (\text{☺}, X))$

- (☺ , X) is an *acceptable program* if and only if *Alerts* (∅, X) = ∅

- (☺_X, X) is a *concluding program* if and only if *Alerts* (∅, X) = ∅ and *Residue* (☺ , X) =∅.

4.3 Composition of AUSTIN Programs

The complete set of AUSTIN's illocutionary acts is presented in Table 13:

Illocutionay act	**Utterance**
commissive	non-definitive question (s, 1, {PATTERN})
	non-definitive assertion (s, 1, {PATTERN})
assertive	(definitive) assertion (s, 0, {PATTERN})
directive	(definitive) question (s, 0, {PATTERN})
declaration	("start", 0, {USER})
	("end", 0, {AUSTIN})
expressive	alert ("cycle in triggering!", 0, {AUSTIN})
	alert ("invalid node! ", 0, {AUSTIN})

Table 13: AUSTIN's illocutionary acts

Thanks to previous definitions, we are now fulfilling the compositional argument up to a certain extent; let us start by the encouraging aspects:

Definition 10:

Let $(\text{☺}_1, X_1) = (\mathcal{U}(\text{☺}_1, X_1), \mathcal{X}(X_1))$ be a *program*,

Let $(\text{☺}_2, X_2) = (\mathcal{U}(\text{☺}_2, X_2), \mathcal{X}(X_2))$ be another *program*,

The following sets can be considered:

$\mathcal{U}(\text{☺}_3) =$ **add** $(\mathcal{U}(\text{☺}_1), \mathcal{U}(\text{☺}_2))$ is the join set of all 'user' utterances,

$\mathcal{U}(X_3) =$ **add** $(\mathcal{U}(X_1), \mathcal{U}(X_2))$ is the join set of all commissives,

$\mathcal{U}(\text{☺}_3, X_3) =$ **add** $(\mathcal{U}(\text{☺}_3), \mathcal{U}(X_3))$ is the set of all initial utterances,

$\mathcal{X}(X_3) = \mathcal{X}(X_1) \cup \mathcal{X}(X_2)$ is the join set of all startSteps,

We can therefore take the following definitions:

$(\mathcal{U}(\text{☺}_3, X_3), \mathcal{X}(X_3))$ is a *program* called *composition of* $(\text{☺}_1, X_1)$ and $(\text{☺}_2, X_2)$.
$(\mathcal{U}(\text{☺}_3, X_3), \mathcal{X}(X_3))$ is denoted: $(\text{☺}_3, X_3) = (\text{☺}_1, X_1) + (\text{☺}_2, X_2)$.

Moreover, "+" is commutative and associative.

A particular case of program is 'user input'= $(\text{☺}, \varnothing)$ with no conversation frame; successive additional user input can also be considered as program composition.

Unfortunately (but this is not very surprising), there is no guarantee that the composition of two cycle-free programs will turn out to be a cycle-free program; in fact it is quite easy to exhibit counter examples[69]!

More precisely, we have to accept that:

- the composition of *acceptable* programs may be *not acceptable*;
- the composition of *concluding* programs may be *not concluding*, even if acceptable.

4.4 Analyzing AUSTIN Programs

4.4.1 Detection of cycles within conversation frames

Cycles are particular sub graphs of $\mathcal{G}$ [Definition 8c]; they happen when several *patterns* of the *conversation frame* trigger each other in a loop. This kind of situation is typical of "equilibrium reaching problems".

Before tackling the detection, let us expose the policies which can be adopted to cure the problem:

1. eating the loop: this policy will consist in grouping all the involved patterns into one unique big pattern. Of course the "compositional cost" is big, it even looks like a failure! Moreover it supposes that the "equilibrium reaching problem" can be solved inside a single reducer.
2. breaking the loop: arbitrarily deciding which of the involved patterns will start first from the initial user input, and the number of loops to be visited.

Of these two policies, the second one is highly preferable: compositionality is preserved, moreover it is highly probable that "eating the loop" precisely consists in "breaking the loop" at a lower level of computation!

[69] Starting from a conversation frame WITH A CYCLE and breaking it into two cycle-free frames will give a counter example when we recompose the fragments!

The detection of cycles in $\mathcal{G}$ is syntactically expressed through the previously taken Definition 8a (part 1):

Let X be a *conversation frame*, let χ_P and χ_Q be two patterns of X,
let $\chi_P^0 = (\alpha(s_P), \varphi_P^0, \varphi_P^{0'}, E_P^0, F_P^0)$ be the *startNode* of χ_P,
let $\chi_Q^0 = (\alpha(s_Q), \varphi_Q^0, \varphi_Q^{0'}, E_Q^0, F_Q^0)$ be the *startNode* of χ_Q,

χ_Q^0 potentiallyTriggeredBy $\chi_P^0 \Leftrightarrow \mathcal{TriggersFor}(s_Q) \cap \mathcal{ReductionsOf}(E_P^0 \cup F_P^0) \neq \varnothing$

The test to be operated therefore relies on "problem4" (see [Definition 1]):

problem4: "$(\exists\ s_1')\ ((s_1' \leq s_1) \wedge (s_1'$ triggerFor $s_2))$"

If we consider: θ_0 = maximal number of commissives initially emitted by a conversational pattern = constant, the complexity of the detection of cycles is $O(\tau^2 * \varphi)$ with the parameters:

- τ = number of conversational patterns
- φ = complexity of problem 4, depending on $\mathcal{S}$ and quite reasonable[70] in the case of the generative grammar of 3.9.

4.4.2 "Deadlocks" in conversation runs

Non-empty *residues* will happen each time another type of loop is met: when several *patterns* of the *frame* rely on each other for providing answers to questions. This kind of situation is called "deadlock" in distributed systems.

This phenomenon is quite different from the previous one; we have met two deadlocks in the informal description of the 'Sudoku filling problem' given in [2.2.4], e. g. when Cell asked questions about "(previously solved) accepted values", in order to produce "(solved as a consequence) accepted values." These deadlocks have been cured by enriching the conceptual language with the concept 'state', so that questions of Cell are about "values at state ^{n}", while assertions are about "values at state $^{n+1}$". Yet, if we look at the initial commissives of the new version of Cell in 3.10.4, we still find two potential deadlocks:

CA06 = ([accepted:Value: ALL:*].[row:XY: ALL:*].[column:XY: ALL:*].[when:State: ALL:*]; CA)

CD08 = ([accepted:Value: ALL].[row:XY: ALL:*].[column:XY: ALL:*].[when:State: ALL:*]; CD)

[70] problem 4 has no reason to be harder than problem5: "(∃ s1') ((s1' ≤ s1) ∧ (s1' answerFor s2))" and in the current "prototype interpreter", problem5 is solved at each step on the conversation run, and the global response-time remains quite acceptable!

CA07 = ([rejected:Value: ALL:*].[row:XY: ALL:*].[column:XY: ALL:*].[when:State: ALL:*]; CA)

CD09 = ([rejected:Value: ALL].[row:XY: ALL:*].[column:XY: ALL:*].[when:State: ALL:*]; CD)

CA06 is a potential answer to CD08, which therefore might get no answers at all. It happens that these apparent deadlocks are not real ones, because the multi-steps algorithms embedded in the pattern discriminate the states at runtime i.e. when invoked under some contextual state, as illustrated in the chronology of Figure 35. This illustrates the fact that "deadlocks" may be "harmless" and therefore need not be detected a priori before conversation starts. They have to be analyzed and cured on the basis of *conversation runs* which have stopped before full termination!

Detection of deadlocks will rely on:

Definition 8a (part 2):
Let X be a *conversation frame*, let $\chi_P=\{\chi_P^i\}_{i\geq 0} \in X$, let $\chi_Q=\{\chi_Q^i\}_{i\geq 0} \in X$,
let $\chi_P^i = (\alpha(s_P), \varphi_P^i, \varphi_P^{i'}, E_P^i, F_P^i)$ be a *postNode* of χ_P,
let $\chi_Q^j = (\mathcal{A}ns(N^j), \varphi_Q^j, \varphi_Q^{j'}, E_Q^j, F_Q^j)$ be a *postNode* of χ_Q,
χ_Q^j potentiallyAnsweredBy $\chi_P^i \Leftrightarrow \mathcal{A}nswersForReductionsOf(N^j) \cap \mathcal{R}eductionsOf(E_P^i \cup F_P^i) \neq \varnothing$

A graph $\mathcal{H}$ based upon the "potentiallyAnsweredBy" relation between the steps[71] of the *residue* will show a cycle at the end of a *conversation run*, as illustrated in

[71] The relation "potentiallyAnsweredBy" has not been defined upon *steps* but upon *nodes*: the complete discussion is not presented

Figure 36 : the "knot" to be solved is between the steps "x_1^2", "x_3^2", "x_4^3", "x_4^2" and "x_2^2" belonging to four different patterns. As a consequence, "x_5^2" is equally blocked. Such cycles in the graph $\mathcal{H}$ can be diagnosed by the abstract machine itself[72].

[72] One interesting direction of research might be the diagnostic of deadlock and emission of associated expressives by the abstract machine!

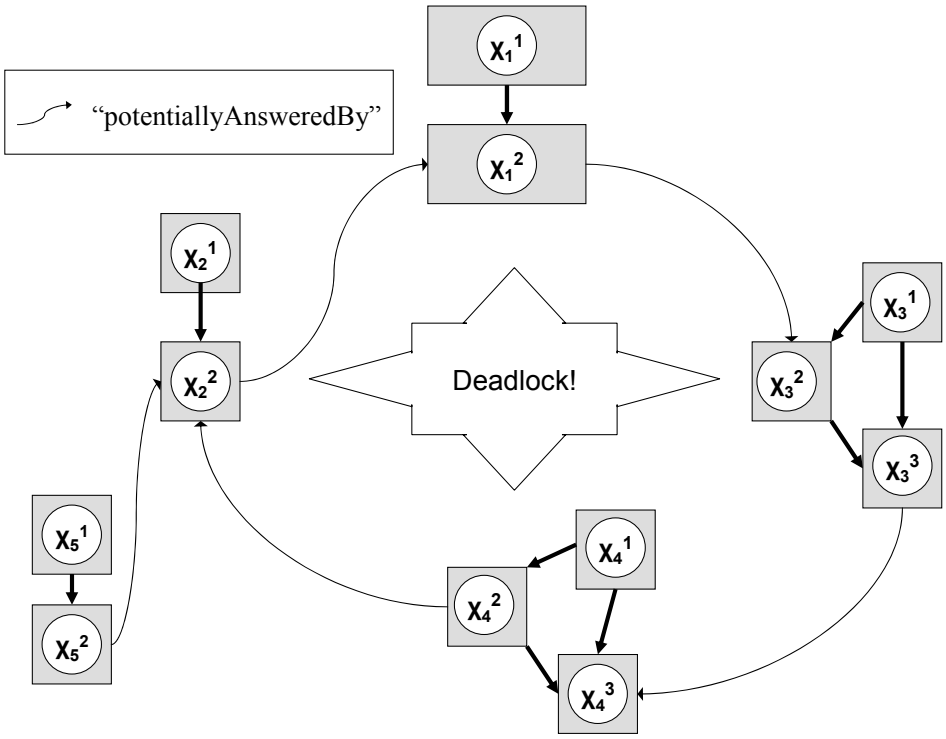


Figure 36 : deadlock!

4.4.3 Termination of a conversation and analys is of residues

Figure 31 in [3.5.3] illustrates a necessary and sufficient condition for detecting termination of a computation:

A conversation ends as soon as both "transition pipe" & "test pipe" are empty!

By incorporating this internal check in the abstract machine, we allow it to declare "conversation has come to an end", and to actually end it.

In order to briefly address the analysis of residues, we must refine the concept of "internal state of a *conversation step*" initially described by its Boolean 'status':

- "status=1/activated" is the initial state of postSteps in invocations
- "status=1/ready" means that all necessary questions have been asked; this is the initial "internal state" for startSteps (before invocation), but also concerns postSteps when all their predecessors have gone through successful internal computation;

- "status=1/informed" means that all necessary answers have been provided and computation is on-going inside the step[73];
- "status=0" means that internal computation has been successful;
- "status=2/crashed" means that an internal error has occurred inside the step, due to a programmer's mistake when building the embedded reducer; the abstract machine should express this through an alert.

On the basis of these detailed "step internal states", the *residue* of a program can be automatically interpreted in the following way:

- empty *residue* = success

In the residue is not empty, the first point to check is whether anything has ever started:

- *residue* = initial state

If no pattern of X has been triggered by $\mathcal{U}$ ($\odot_X$), all startSteps are "ready" and the program has to be revised accordingly!

Otherwise, several possibilities remain:

- non-empty residue & one or several "crashes": reducers have to be checked
- non-empty residue & one or several "status=1/informed": at least one *reducer* is taking time without producing results … this is the famous "Turing Machine halting problem"!
- non-empty residue & no crashes & no "status=1/informedBySteps" & at least one "status=1/informedBySteps-waitingForUser" : waiting for user's arbitration

Otherwise:

- one or several "status=1/ready": presence of deadlock (see 4.4.2)!
- no "status=1/ready": some postSteps are "activated" but not "ready", although their predecessors have been activated ! … at least one conversational pattern has to be checked against [Definition 4d]; there has been a programmer's mistake while building the corresponding *pattern*.

We do not demonstrate here all the above assertions, but only summarize the informal discussion.

There are 3 possible causes for non-empty *residues*:

[73] in extensions of AUSTIN allowing user interference during the conversation, an additional "status=1/informed-waitingForUser" means that all necessary answers have been provided by other steps, BUT the step is still waiting for information from a user.

1) DEADLOCKS: they are related to global information processing and can be solved either in the context of KNOWLEDGE REPRESENTATION (building sentences) or in the context of NODES DISTRIBUTION.

2) programmers's mistake in structuring *conversational patterns*: they can be prevented through appropriate automatic check of pattern validity.

3) programmer's mistake in building *reducers*: they are much more delicate to prevent (even when constraining the "libraries" from which the programmer will retrieve components, their composition of components can hardly be automatically checked)

The very encouraging aspect is that these irreducible remaining difficulties have been confined in independent *reducers* and therefore can be solved in the context of LOCAL INFORMATION PROCESSING.

4.5 Compositionality of AUSTIN programs illustrated in the 'Sudoku filling problem'

In [3.10], we have described the sentences and patterns aiming at solving a Sudoku through a conversation between 'Cells' and 'Sets'. A 'Launcher' is in charge of invoking all Cells and Sets.

In this sub-section, we are first going into the two nodes of the Launcher, where some "Python[74]" code is embedded, so that the reader can discover the two embedded reducers.

Then we are going to experiment with LAUNCHER, CELL and SET in order to illustrate the compositionality.

4.5.1 Going into the nodes of the Launcher

The following code is almost[75] the real code of the current prototype. It consists in three parts:

- a general presentation of the *conversational pattern* 'Launcher', with the list of internally manipulated sentences in which we can distinguish the *starter* and the initial *commissives*
- the *startStep* 'LAUNCHER-1'
- the *postStep* 'LAUNCHER-2'

[74] Python [van Rossum, 1995] is a multi-paradigm language allowing various programming styles.
[75] we have harmonized some terminology and hidden a current implementation feature of the prototype consisting in exporting local variables from one step to its followers.

Comments inside the code are written in italics and preceded by '#'. Not all the functions will be explained; the most interesting one in this example is '**specialize**' which obeys the following syntax:

'sentence 2' = 'sentence 1' . **specialize** ("thisRole": "thisExpression")

'sentence 2' is obtained from 'sentence 1' by the following operations:

i. find the *word* having the *role* 'thisRole' in 'sentence 1' (necessarily unique because of the syntactic rules of the grammar)

ii. ['thisRole': 'thisConcept': ALL : *] replaced by ['thisRole': 'thisConcept': 'thisExpression'] (the word becomes definitive)

In the current prototype, other operations allow to reduce concepts, to specialize a word without making it definitive, to make a word definitive … they are not presented here.

Another interesting operand is '**getIndividualValue**' which obeys the following syntax:

'newExpression' = 'thisSentence' . **getIndividualValue** ("thisRole")

'newExpression' is obtained from 'thisSentence' by the following operations:

i. find the *word* having the *role* 'thisRole' in 'thisSentence' (necessarily unique because of the syntactic rules of the grammar); this *word* must be of the form ['thisRole': 'thisConcept': 'thisExpression']

ii. 'newExpression' = 'thisExpression'

The reader will also notice '**add**' and '**remove** which correspond to the definitions given in Chapter3.

Conversational pattern: LAUNCHER:

 # 'Launcher' is in charge of invoking all Cells and Sets of any grid when given its dimension (our convention is that a 2^3 cells grid is of dimension '2', a 3^3 cells grid of dimension '3' …); 'Launcher' invokes for all states up to a pre-chosen maximum.

Fullname: "Launching all Cells and Sets up to max state"

Sentences:

SAlauncher = ([max:State:ALL:*].[dimension:Param:ALL:*]; SA)

CA01 = ([test:Value:ALL].[row:XY:ALL:*].[column:XY:ALL:*].[dimension:Param:ALL:*].[when:State:ALL:*]; CA)

CA02 = ([test:Set:ALL:*].[dimension:Param:ALL:*].[when:State:ALL:*]; CA)

CD03 = ([accepted:Value:ALL].[row:XY:ALL].[column:XY:ALL].[when:State:ALL:*]; CD)

Starter: SAlauncher

Initial commissives: CA01, CA02, CD03

```
startStep: LAUNCHER_1:
    output CA01, CA02, CD03 # these three commissives are reduced by the node
    body(triggerForSAlauncher):
# initialization of local variables
        stateIni = 0
        stateMax = triggerForSAlauncher.getIndividualValue("max")
        sudoku = triggerForSAlauncher.getIndividualValue("dimension")
        size = sudoku * sudoku
# commissive CD03 -> directive DD030 (definitive question about all states posterior to 'stateIni')
        DD030 = CD03.specialize("when",">stateIni")
        remove(CD03)
        add(DD030)
# commissive CA01 -> commissive CA011(non-definitive assertion: only one word has become definitive!)
        CA011 = CA01.specialize("dimension",sudoku)
# commissive CA02 -> commissive CA021(non-definitive assertion: only one word has become definitive!)
        CA021 = CA02.specialize("dimension",sudoku)
# building the list of Cells and Sets according to dimension
        CAsets = [ ]
        CAcells = [ ]
        for z in range(sudoku): # 'range(sudoku)' goes from 0 to 'sudoku'-1
            xSquare = z+1
            for y in range(sudoku):
                ySquare = y+1
                square = "{ xSquare,  ySquare }"
# commissive CA021 -> list of commissives CAsets(non-definitive assertions)
                CA0211 = CA021.specialize("test",square)
                CAsets.append(CA0211)
            for x in range(size):
                x+=1
                rowSet = "{ x, 0 }"
# commissive CA021 -> list of commissives CAsets (non-definitive assertions)
                CArowSet = CA021.specialize("test",rowSet)
                CAsets.append(CArowSet)
                columnSet = "{ 0, x }"
# commissive CA021 -> list of commissives CAsets (non-definitive assertions)
                CAcolumnSet = CA021.specialize("test",columnSet)
                CAsets.append(CAcolumnSet)
                for y in range(size):
                    y+=1
# commissive CA011 -> list of commissives CAcells(non-definitive assertions)
                    CAcell = CA011.specialize("row",x).specialize("column",y)
                    CAcells.append(CAcell)
```

```
# building all Cells and Sets for all states
# list of commissives CAsets -> assertives AAset (definitive assertions)
# list of commissives CAcells -> assertives AAcell (definitive assertions)
        for state in range(stateMax):
            for CAset in CAsets:
                AAset = CAset.specialize("when",state)
                add(AAset)
            for CAcell in CAcells:
                AAcell = CAcell.specialize("when",state)
                add(AAcell)
        remove(CA01)
        remove(CA02)
```

```
postStep: LAUNCHER_2:
    inreplyto launcher_1.DD030 # the node is activated when all answers to DD030 have been received
    body(answersForDD030):
# importation of appropriate Java modules
        from javax.swing import JOptionPane
# initialization of local variables
        listResults = [ ]
# building "info" showing final results
# \n means "next line
        info = "\n"
        if len(answersForDD030) == 0:
            info += "No values found ...\n"
        else:
            for answer in answersForDD030:
                result ="row: & answer.getIndividualValue("row")"
                result +="col: & answer.getIndividualValue("column")"
                result +="value: & answer.getIndividualValue("accepted")"
                listResults.append(result)
                listResults.sort()
            for result in listResults:
                info += " result \n"
# presenting "info" to the user
        JOptionPane.showMessageDialog(info)
```

A reader familiar with 'Python' will notice that we have simplified the syntax for better readability; a reader unfamiliar with 'Python' may still complain about readability … the point here is not the code itself, but the demonstration that:

1. a high level programming language is available for the building of reducers;
2. only the functional part of the multi-steps algorithms has to be programmed, since the interactions are managed by the abstract machine.

4.5.2 Experimenting compositionality

In this subsection, we define 5 programs and run different compositions; it will illustrate both compositionality, and increased efficiency in 'Sudoku filling'.

The first program is the user input concerning grid n° 53, when conversation is only about the first two states; we denote this program: (☺ $_{53}$, Ø). It describes the 24 filled cells of the initial grid, by as many *assertives* linking together the concepts 'Value', 'XY', 'State'.

An additional *assertive* gives the LAUNCHER the parameters of the computation: it indicates by "dimension = 3" that the grid to solve has 3^4 cells, and by "max state = 2" that only states '0', '1' and '2' must be computed.

Program (☺ $_{53}$, Ø): user input

([dimension:param:3].[max:state:2]; AA)

([accepted:value:9].[column:xy:8].[row:xy:1].[when:state:0]; AA)

([accepted:value:9].[column:xy:6].[row:xy:6].[when:state:0]; AA)

([accepted:value:9].[column:xy:4].[row:xy:3].[when:state:0]; AA)

([accepted:value:8].[column:xy:5].[row:xy:1].[when:state:0]; AA)

([accepted:value:8].[column:xy:2].[row:xy:6].[when:state:0]; AA)

([accepted:value:7].[column:xy:7].[row:xy:5].[when:state:0]; AA)

([accepted:value:7].[column:xy:2].[row:xy:7].[when:state:0]; AA)

([accepted:value:6].[column:xy:8].[row:xy:3].[when:state:0]; AA)

([accepted:value:6].[column:xy:7].[row:xy:4].[when:state:0]; AA)

([accepted:value:6].[column:xy:5].[row:xy:6].[when:state:0]; AA)

([accepted:value:6].[column:xy:3].[row:xy:9].[when:state:0]; AA)

([accepted:value:5].[column:xy:5].[row:xy:8].[when:state:0]; AA)

([accepted:value:4].[column:xy:6].[row:xy:9].[when:state:0]; AA)

([accepted:value:4].[column:xy:2].[row:xy:2].[when:state:0]; AA)

([accepted:value:4].[column:xy:1].[row:xy:7].[when:state:0]; AA)

([accepted:value:3].[column:xy:7].[row:xy:8].[when:state:0]; AA)

([accepted:value:3].[column:xy:3].[row:xy:2].[when:state:0]; AA)

([accepted:value:3].[column:xy:1].[row:xy:9].[when:state:0]; AA)

([accepted:value:2].[column:xy:7].[row:xy:3].[when:state:0]; AA)

([accepted:value:2].[column:xy:5].[row:xy:9].[when:state:0]; AA)

([accepted:value:2].[column:xy:4].[row:xy:5].[when:state:0]; AA)

([accepted:value:2].[column:xy:3].[row:xy:7].[when:state:0]; AA)

([accepted:value:2].[column:xy:1].[row:xy:6].[when:state:0]; AA)

([accepted:value:1].[column:xy:6].[row:xy:1].[when:state:0]; AA)

Programs embedding the reasoning

The three other programs will consist in:

- no user input;
- one single conversational pattern.

We denote them (Ø, $X_{Launcher}$), (Ø, X_{Cell}) and (Ø, X_{Set}) respectively; by composition, we may define:

- $Program_1$ = (☺ $_{53}$, Ø) + (Ø, $X_{Launcher}$) + (Ø, X_{Cell})
- $Program_2$ = (☺ $_{53}$, Ø) + (Ø, $X_{Launcher}$) + (Ø, X_{Set})
- $Program_3$ = (☺ $_{53}$, Ø) + (Ø, $X_{Launcher}$) + (Ø, X_{Cell}) + (Ø, X_{Set})

We have run these programs through the prototype: happily they all are *concluding programs*! … Therefore, no alerts, no residues … the following table recapitulates the assertives found in their *results*:

Program	*Assertives in result*
(☺ $_{53}$, Ø) + (Ø, $X_{Launcher}$) + (Ø, X_{Cell})	([accepted:value:7].[column:xy:5].[row:xy:2].[when:state:2]; AA)
(☺ $_{53}$, Ø) + (Ø, $X_{Launcher}$) + (Ø, X_{Set})	Ø
(☺ $_{53}$, Ø) + (Ø, $X_{Launcher}$) + (Ø, X_{Cell}) + (Ø, X_{Set})	([accepted:value:9].[column:xy:5].[row:xy:7].[when:state:2]; AA) ([accepted:value:9].[column:xy:1].[row:xy:2].[when:state:2]; AA) ([accepted:value:7].[column:xy:5].[row:xy:2].[when:state:2]; AA) ([accepted:value:5].[column:xy:2].[row:xy:9].[when:state:2]; AA) ([accepted:value:2].[column:xy:6].[row:xy:2].[when:state:2]; AA) ([accepted:value:2].[column:xy:2].[row:xy:1].[when:state:2]; AA)

This example demonstrates the compositional power of AUSTIN: the composition of $Program_1$ and $Program_2$, each of them embedding a partial theory leading to partial prediction, predicts more than the union of partial predictions.

5 The AUSTIN machine put in perspective

In the introductive part [1.2] we wrote: "While in declarative techniques like LOGICAL PROGRAMMING the modelling part is much constrained by the goal driven approach and requires special skills, in imperative techniques implementing the metaphor of 'multi-experts systems' through a BLACKBOARD ARCHITECTURE the detailed specifications of the communication events and the finding of an efficient control scheme are far from simple." The objective of this chapter is to give more arguments in favour of those assertions, and to settle a general discussion about the AUSTIN machine, around two questions:

i) can we give a declarative (logical) semantics to the conversational calculus?

ii) our system has been thought for enhancing the collaboration between 'domain experts' who are not computer scientists; how can we compare AUSTIN with current 'multi-experts systems'?

5.1 Questioning the logical semantics

[Van Roy & Haridi, 2004] gives the following introduction to logic programming: "The advantage of logic programming is that programs have two semantics, a logical semantics, and an operational semantics, which can be studied separately. If the underlying logic is chosen well, then the logical semantics is much simpler than the operational.

A logic program is a statement of logic that is given an operational semantics, i.e., it can be executed on a computer. If the operational semantics is well designed, then the execution has two properties: it is correct (all consequences of the execution are valid logical consequences of the program considered as a set of logical axioms) and it is efficient, i.e., it allows writing programs that execute with the expected time and space complexity."

In the following, most of [5.1.1] has been extracted from [Shapiro, 1989] which surveys the family of concurrent logic programming languages through both viewpoints corresponding to declarative (logical) semantics and operational semantics. The point here is not to go into the survey itself, but benefit from the uniform operational framework provided by Shapiro.

5.1.1 Shapiro's framework for concurrent logic programming languages

We start with some preliminary definitions for the basic components of all logic languages:

- a *term* is a variable (e.g., X) or a function symbol of arity $n \geq 0$, applied to n terms (e.g., c and f (a, X, g(b, Y)));
- an *atom* is a formula of the form: p (T_1, …, T_n), where p is a predicate of arity 'n' and T_1, …,T_n are terms;
- a definite *clause* is a formula of the form: $A \leftarrow B_1, \ldots, B_n$; $n \geq 0$, where A is an atom, and B_1, …, B_n is a sequence of atoms. A is called the clause's head; and B_1, …, B_n its body;
- a *logic program* is a finite set of definite clauses;
- a *goal* is a sequence of atoms A_1, …,A_n;
- *unification of two terms T_1, and T_2*: most general ("simplest") substitution θ, whose application to two terms T_1, and T_2 verifies $T_1\theta = T_2\theta$. The *unification* returns θ if there is one most general unifier (this is denoted *mgu* (T_1, T_2) = θ), or *fail* if there is none (this is denoted *mgu* (T_1, T_2) = *fail*).

Declaratively, a logic program is the conjunction of its clauses $A \leftarrow B_1, \ldots, B_n$ which are interpreted as :

$\forall X_1, \ldots, X_n$: A is true if $B_1, \ldots, B_n$ are true

Operationally, logic programs can be viewed as an abstract computational model. A computation in this model is a goal-driven deduction from the clauses of the program. Computations in this model are non-deterministic: from each state of the computation, there may be several possible transitions; the clauses of a logic program are read as transition rules of a nondeterministic transition system.

Shapiro gives a definition of transition systems based on *unification* [Pnueli, 1986] [Gerth et al., 1988] and consisting of:

- a set of states: A state is a pair (G; θ), where G (the goal) is either a sequence of atoms or *true* (empty sequence) or *fail*, and θ is a substitution or 'ε' (empty substitution)
- a set of transitions: A transition is a function from states to sets of states. For states S, S' and transition t, we denote that $S' \in t(S)$ by $S \rightarrow^t S'$. 'Reduce' is a transition: $(A_1, \ldots, A_n; \theta) \rightarrow^{reduce} (A_1, \ldots, B_1, \ldots, B_n \ldots, A_n; \theta \circ \theta')$ if *mgu*

(A_i, A) = θ' for some renamed apart clause A ← B_1, . . . , B_n; in case for some 'i' and for every A: *mgu* (A_i, A) = *fail*, the system writes: 'Fail': (A_1, …, A_n; θ) $\rightarrow^{fail}$ (A_1, …, A_n; θ).

A computation by a transition system on a goal G is a sequence of states c = S_1, S_2, …, S_n where:

- S_1 = (G; ε);
- for each 'k' $S_{k+1} \in t(S_k)$;
- c is finite and of length 'k' only if S_k = (*true*; θ).

"The main difference between logic programming and other computational models is the logical variable and its manipulation via *unification*. The basic data-manipulation operation in logic programs (*unification*) results in a substitution. Operationally, a substitution can be thought of as a simultaneous assignment of values to variables, except that here:

- a variable can be assigned a value only once;
- the value assigned can be itself another variable or a term containing variables.

… The transition system for logic programs realizes, in effect, a proof procedure for logic programs. Each Reduce transition is actually an application of an inference rule, called SLD-resolution [Hill, 1974][Lloyd, 1987], which is a special case of Robinson's resolution inference rule [Robinson, 1965]. SLD-resolution, and hence the transition system, has soundness and completeness properties that link their operational view to the logical view of logic programs [Lloyd,1987]."

[Apt & van Emden, 1982] even establish that SLD-resolution is strongly complete: "Let P be a program, let N be a conjunction of atoms; if P ∪{non N} is inconsistent, then any SLD tree with {non N} as a root contains the empty clause."

From the operational viewpoint of the computational model, the clauses of a logic program support a nondeterministic transition system; such a system may be effectively computed in various ways, essentially divided into "parallel" and "sequential".

When a sequential algorithm is chosen as a computation rule for exploring the tree, like in PROLOG, the completeness of the transition system disappears. In PROLOG, a clause is interpreted in a deterministic way as "to execute procedure A, call B_1, call B_2, … call B_n"; the remaining non determinism relative to the choice of the unification is interpreted as implicit search for all solutions. In PROLOG, the operational semantics depends on a specific order of evaluation of the rules and of the atoms within each rule.

"Most sequential PROLOG systems compute the solutions to a goal by searching depth-first the computation tree induced by different choices of clauses. Typically, one solution is produced at a time, and additional solutions are searched for only by request. Under this behaviour it is possible for a program to produce several solutions and then diverge. The point of divergence is determined by the order of clause selection. Usually a PROLOG program is defined as a sequence (rather than a set) of clauses, and the order of clause selection is textual order. The possibility of divergence in the face of both successful and infinite computations makes PROLOG incomplete as a proof procedure for logic programs. However, this incompleteness is not a major problem in practice. Knowing the PROLOG computation rule, PROLOG programmers order bodies of clauses so that infinite computations are avoided on expected goals…

PROLOG is a convenient language for a large class of applications. However, to be practical it augmented the pure logic programming model with extralogical extensions. … As we shall see later, this deficiency is peculiar to PROLOG and is not inherent to the logic programming model. Indeed, concurrent logic programs can specify both input/output and shared modifiable store in a pure way, relying solely on their different computation rules and different interpretations of nondeterminism. [Shapiro, 1989]"

In order to clearly establish the contrast between sequential languages and concurrent languages, Shapiro recalls an important distinction explicated both in [Harel and Pnueli, 1985] and in [Hewitt, 1985]:

- a transformational system is a closed system receiving an input at the beginning and yielding an output at the end; in the implementation of non deterministic transformational systems, the correct choice among all possible transitions is arbitrated by the program during execution and unsuccessful computations are not output as results. Transformational abstract machines are compared on the basis of equivalence of successful computations.
- a reactive system is an open system which maintains an interaction with its environment, like an operating system or a database management system; it may never terminate. Therefore, partial output is produced independently from the success or failure of the computation. Reactive abstract machines are compared on the basis of equivalence of all computations, either failing or successful.

Any concurrent system needs to specify some "reactive" processes i.e. processes maintaining continuous interaction with each other or with the environment, even if the whole appears as a transformational system.

With the help of this distinction, *concurrent logic languages* can be defined as logic programming languages that specify reactive open systems and thus can be used to implement concurrent systems and parallel algorithms.

A concurrent logic program is a reactive program augmented with synchronization. In order to address synchronization, two new elements must be added to the syntax of a concurrent logical abstract machine: *guard test predicates* and *guarded Horn clauses*:

- guard test predicates is a finite set including integer(X), X < Y, X=Y;
- a guarded Horn clause is a formula of the form: $A \leftarrow G_1, \ldots, G_m \mid B_1, \ldots, B_n$ with m, n > 0. A is the head; the predicate of each G_i, i = 1, …, m, is a guard test predicate, and the variables of G_i occur in A. If the guard is empty (m = 0), then the commit operator '|' is omitted. An empty body (n = 0) is denoted by true.

A guarded clause is similar to an alternative in the guarded command of [Dijkstra, 1976]:

i. the head and guard specify the conditions under which the 'reduce' transition can use the clause, as well as the effect of the transition on the resulting state;
ii. the body specifies the state of the process after taking the transition: A process can halt (empty body), change state (unit body), or become several concurrent processes (a conjunctive body).

Concurrent logic languages differ in what can be specified by the head and the guard.

In [Shapiro, 1989], a transition system close to the previously described one is used. It leads to an abstract machine which does not model input from an outside environment, and therefore is not reactive in the full meaning of the term. Programs are defined as a finite sequence of guarded Horn clauses and concurrency is modelled by the nondeterministic interleaving of atomic actions, the grain of which is one of the major differences between languages. The synchronization mechanism relies on the matching of a goal atom A with a head of a clause $A' \leftarrow G \mid B$ by the '*try*' function.

The '*try*' function works like *mgu* if the goal atom is an equality and the clause is the equality clause X=X:

- $try(T_1=T_2, X=X) = mgu(T_1, T_2)$

When the head predicate of the clause is not the equality predicate, $try(A, A' \leftarrow G \mid B)$ works in the following way:

- it succeeds if A is an instance of A' AND checking Gθ succeeds; in such a case, it returns the most general substitution θ such that A =A'θ (*mgu* (A, A') = θ);
- it fails if the goal atom and the head are not unifiable (*mgu* (A, A') = *fail*) OR if (*mgu* (A, A') = θ) AND checking Gθ fails);
- otherwise, it suspends.

As in a logic program, states of computation are pairs (G; θ). A computation begins from an initial goal (G; ε) and progresses though the following 'reduce' and 'fail' transitions:

- 'reduce' is the transition: $(A_1, \ldots, A_n; \theta) \rightarrow^{reduce} (A_1, \ldots, B_1, \ldots, B_n \ldots, A_n; \theta \circ \theta')$ used when *try* $(A_i, A) = \theta'$ for some renamed apart clause $A \leftarrow G \mid B_1, \ldots, B_n$;
- 'fail' is the transition: $(A_1, \ldots, A_n; \theta) \rightarrow^{fail} (A_1, \ldots, A_n; \theta)$ used when for some 'i' and for every renamed apart clause $A \leftarrow G \mid B_1, \ldots, B_n$, *try* $(A_i, A) = fail$.

We find in [Shapiro, 1989] the following recapitulation, which explains how the transition system works and yields even unsuccessful computations like deadlocks:

"Concurrent processes communicate by instantiating shared logical variables and synchronize by waiting for variables to be instantiated. Variable instantiation is realized in most concurrent logic languages by unification. Three approaches were proposed to the specification of synchronization in concurrent logic programming: input matching (also called input unification, one-way unification, or just matching) [Clark and Gregory, 1981] [Ueda, 1986], read-only unification [Shapiro, 1983], and determinacy conditions [Yang and Aiso, 1986]. All share the same general principle: The reduction of a goal atom with a clause may be suspended until the atom's arguments are further instantiated. Once the atom is sufficiently instantiated, the reduction may become enabled or terminally disabled, depending on the conditions specified by the head and guard […] Using matching for reducing a goal with a clause delays the reduction until the goal is sufficiently instantiated, so that its unification with the clause head can be completed without instantiating goal variables… The dataflow nature of matching is evident: An "instruction" (clause) is enabled as soon as sufficient "data" (variable instantiations) arrive. Although simple, matching is found in practice sufficiently powerful for all but the most complex synchronization tasks

… Note that the suspend result of the try function is not used in the 'reduce' and 'fail' transitions. Its effect, therefore, is to prevent a goal atom from reducing with a clause and from failing. Specifically, if A is a goal atom for which try(A, C) = suspend for some clause C in P, and try(A, C ') = suspend or

fail for every other clause C' in P, then A can participate neither in a 'reduce' transition nor in a 'fail' transition. Such a goal atom is called suspended. A state consisting of a goal in which all atoms are suspended is terminal, as no transition applies to it. Such a state is called a dead-lock state, and a computation ending in a deadlock state is called a deadlocked computation."

In concurrent programming languages, a finite computation is a succession of states (G_1, ε), (G_2, θ_2), … (G_n, θ_n) leading to an observable behaviour (G_1, θ, x) where:

- G_1 is the initial goal;
- θ is the answer substitution of the computation;
- x = *true* (case G_n= *true*) OR *fail* (case G_n= *fail*) OR *deadlock*.

5.1.2 Logical deduction in AUSTIN

AUSTIN is a concurrent system augmented with a synchronisation mechanism. It behaves like a transformational system, and yet relies on reactive processes maintaining continuous interaction with each other:

- the transition system constantly listens to additions and removals of utterances and steps;
- each startStep waits for invocation or revocation;
- each postStep waits for activation.

We propose in Table 14 a mapping of some AUSTIN concepts with terms and atoms:

AUSTIN concepts	**terms, atoms, formulas**
status (Boolean)	term
sentence	term
conversation node	term
$\text{sentence}^1 \leq \text{sentence}^2$	atom: *reductionOf* (sentence^1, sentence^2)
sentence^1 answerFor sentence^2	atom: *answerFor* (sentence^1, sentence^2)
sentence^1 triggerFor sentence^2	atom: *triggerFor* (sentence^1, sentence^2)
utterance *u*	atom: *utterance*(sentence, status) cf Definition 2
conversation step χ_P^i	atom: *step* χ_P^i(conversation node, sentence, status) cf Definition 5a

Table 14: Mapping between AUSTIN and logic / 1

In AUSTIN, instead of a system based on clauses, we have the following three transition rules recapitulated [3.6.2]:

[testInvoke (x_P^0, u, ω) = ((x_P^0, u), true)] → **invoke** (x_P^0, a)
[testActivate (x_P^1, ω) = ((x_P^1, A^1), true)] → **activate** (x_P^1, A^1)
[testRevoke (x_P^0, ω) = (x_P^0, true)] → **revoke** (x_P^0)

When for instance we look deeper into **testRevoke**, we read in Definition 7c:

"Let 'ω' be the current state of the conversation:

[**testRevoke** (x_P^0, ω) = (x, true)] ⇔ [(cond^1 = true) and (cond^2 = true)]

- cond^1 first verifies that no more potential triggers exist for the starter of χ_P^0, then verifies that all definitive triggers for the starter of χ_P^0 have gone through the corresponding invocation:
cond^1 = [*PotentialTriggersFor* (s_P, ω) = ∅ and (∀u = (a, 0)∈ *DefinitiveTriggersFor* (s_P, ω)) (x_P^0, a, 0) ∈$\mathcal{X}$(ω)]

- cond^2 verifies that the revocation has not already occurred: cond^2 = [(χ_P^0, ∗, 1) ∈$\mathcal{X}$(ω)]"

In cond^1, we have twice the expression '∀u': one is visible in the formula, the other one is hidden in *PotentialTriggersFor* (s_P, ω) = ∅. Since we have defined 'u' as a predicate, this means we are operating a second-order calculus on predicates!

Moreover, whenever testRevoke answers 'true', we apply Definition 6c:

"**revoke** (x_P^0) is the *atomic transition* from state ω to state ω' consisting in the following actions:

1. for each "s" in $E_I^0 \cup F_I^0$ **remove** ((s, 1), (χ_P, ∗)) from $\mathcal{U}$(ω)
2. **remove** (x_P^0) from $\mathcal{X}$(ω)"

It appears that two kinds of atoms (utterances and predicates) are removed!

Our mapping has then two be completed in the following way:

AUSTIN concepts	**terms, atoms, formulas**
test (invoke, activate, revoke)	second order formula (quantifiers operate on predicates)
transition (invoke, activate, revoke)	second order formula (either **remove** must be considered as a predicate operating on predicates, or a third status '-1' has to be considered)

Table 15: Mapping between AUSTIN and logic / 2

It appears that the AUSTIN machine could be described as a concurrent second order calculus, where lists of atoms are created as a function of lists of atoms by the reducers.

This logical formalization is out of the scope of this thesis; we only emphasize three points here:

The Close World assumption

In answering questions, as well as in triggering patterns, we have implicitly relied on the default assumption that all that is true is expressed in atoms; this is a variant of the Closed World Assumption.

When we look closer into what happens during the AUSTIN conversations, we see that atoms are generated when commissives are reduced into assertives or directives: the "close world" is the world of what is present in the conversation as well as the potential output expressed by the commissives.

According to formalization choice, we may or not use logical negation (e.g. for expressing the fact that $\mathcal{P}otential\mathcal{T}riggers\mathcal{F}or(s_P, \omega) = \varnothing$; the alternative being the creation of appropriate predicates.

Other possibilities are closer to operational semantics: "negation as failure" is an interpretation of logical negation according to which the negation of a formula is true if and only if the formula cannot be proved true. In PROLOG [Colmerauer and Roussel, 1992] for instance, the absence of evidence equals evidence for absence. It even happens that both logical negation and negation as failure are used, like in answer set programs [Lifschitz, 2002].

Monotonicity

The consequence relation for a given logic is *monotonic* when adding a formula to a theory never produces a reduction of its set of consequences. This is interpreted as a monotonic increase in the knowledge and subsequently disallows reasoning techniques such as 'reasoning by default' (facts may be known only because of lack of evidence of the contrary), 'abductive reasoning' (facts are only deduced as most likely explanations), 'reasoning about knowledge' (the ignorance of a fact must be retracted when the fact becomes known), and 'belief revision' (new knowledge may contradict old beliefs).

AUSTIN conversations are monotonic in the sense that atoms obey the following production rules (a unique partial order '≤' is mentioned whereas it has different definitions for utterances and steps; '-1' is used to indicate 'removal'):

utterance (sentence, 1) -> *utterance* (sentence', 1) sentence' ≤ sentence

utterance (sentence, 1) -> *utterance* (sentence, -1)

utterance (sentence, 1) -> *utterance* (sentence, 0)

step (node, *, 1) -> *step* (node', sentence, 1) node' ≤ node ; sentence ≤ *

step (node, sentence, 1) -> *step* (node', sentence, 1) node' ≤ node

step (node, sentence, 1) -> *step* (node', sentence, 0) node' ≤ node

step (node, sentence, 1) -> *step* (node, sentence, -1)

According to this draft of grammar, only utterances which have 'status = 1' (which means: neither definitive nor removed) may play the role of left members in the production rules. Right members are always "terminal". True knowledge, corresponding to assertives (with status = '0') increases through successive reduction of commissives.

Goal driven versus Event driven

In the computation pattern provided by Shapiro for concurrent logic languages, the ultimate goal is initially known and the challenge consists in finding a proper substitution for establishing a logical path between available data and the goal. This is based on concurrent blind "unification-tries" until a path is found; if all attempts fail, the global result is failure. Shapiro emphasizes that in order to deserve the name of logic programming language, each successful computation must correspond to a proof of the goal statement. According to his survey, the difficulty seems to be in articulating concurrency with systematic exploration of the tree of possible unifications.

AUSTIN is not goal driven, and therefore cannot match Shapiro's framework. AUSTIN instantiates simplified conversations between pre-existing theories; propagates the initial context through automatic substitutions operated by the reducers, and benefits from a semantically driven concurrency[76]. The recording of the conversation traces a logical path between initial context and final consequences through an automatic articulation of the 'partial theories'.

[76] this was illustrated in the Sudoku example where CELL and SET were invoked for all contexts (place in the grid, state in the reasoning) in parallel

5.2 AUSTIN as a Blackboard

This section will appear as complementary to the previous one; it gives a framework for AUSTIN event-driven approach.

5.2.1 The Blackboard Architecture

Most of currently available literature concerning blackboards address either partial aspects linked to organization and control, or focus on the questions of concurrency and parallelism, or insist on applications; a nice collection of such articles is for instance [Jagannathan et al, 1989]. Theoretical surveys on the subject are aimed at comparison with other computational frameworks are no so easy to find; [Velthuijsen, 1992] proposes a formal description of blackboard architecture following this goal, as well as supporting analysis and choice of possible options in blackboard implementations. He voluntarily chooses a formalism emphasizing observable behaviour rather than operational semantics: "With our current formalization it is impossible to express directly how (internal) choices within a described system depend on previous decisions or available information". Since our own approach has been exactly the opposite, the following presentation will usefully complement the view we gave in Chapter 3. Although the references given by Velthuijsen are a bit old for constituting a real 'state of the art', most of the blackboards he mentions are used today; his deep analysis has lost none of its pertinence towards topical issues. This sub-section borrows large extracts from his thesis.

"The blackboard architecture is a knowledge-based system architecture first suggested in [Newell, 1962] and first implemented in the early seventies as the architectural framework for the speech-understanding system Hearsay-II [Lesser et al., 1975]

… The blackboard architecture can be illustrated best by the metaphor that lent its name. Imagine several specialists with different expertise required to solve a problem cooperatively. The specialists are gathered around a blackboard for communicating intermediate results and hypotheses. As soon as data has been written on the blackboard by one of the specialists, other specialists may decide that they can use that information …until a solution to the problem is found or no specialist is able to proceed on any of the data present on the blackboard. The blackboard architecture models this type of problem-solving behaviour. Separate, otherwise independent computer programs called *Knowledge Sources* (KSs), are given the role of specialists. A global *data structure* is used to write and read intermediate results and hypotheses. Knowledge sources communicate only via the blackboard. In many blackboards a *control unit* is added. The control unit mediates between competing knowledge

sources. For this task it can use information on KSs and data stored on the blackboard. Metaphorically, a control unit is given the role of discussion leader.

… A system with a blackboard architecture is basically data driven. It detects and exploits opportunities for system activities …

… A blackboard architecture which allows concurrent execution of knowledge sources is called a parallel blackboard architecture, provided that the blackboard and control unit are not distributed. A distributed blackboard architecture is a blackboard architecture with a distributed blackboard and control unit. ”

5.2.1.1 Definition of the main concepts

Blackboard

The blackboard itself can be a very complex data structure, consisting in panels, each divided in levels storing ‘units’:

“A blackboard unit is a data structure filled with data together with an identifier. A blackboard level is a dynamic abstract store for blackboard units together with an identifier. A blackboard panel is a (possibly partially) ordered set of blackboard levels together with an identifier. A blackboard is an (also possibly partially) ordered set of blackboard panels, together with a number of procedures for accessing its constituents and (optional) additional organising relations.”

KS and KSI

“A knowledge source or KS is an activity, implemented as a piece of program, called the KS action part (Act_i corresponding to KS_i), together with a data structure that describes the action part: the KS descriptor… The condition of a KS defines the set of blackboard states for which the KS is annotated as applicable... A precondition is that part of the condition of a KS that determines applicability of a blackboard by using solely blackboard state information.

A knowledge source instantiation, or KSI, denotes a pair consisting of a KS and a context (bound variables, determined by the blackboard state in which this particular KS became applicable)”

Events

Even when events are descriptive, they remain very simple pairs (attribute, value): “An event is a data structure representing a state change. In the blackboard architectures we examined, events are either implemented as tokens or as attribute-value pairs. Generally, tokens (also named labels: names to indicate a specific interaction with the blackboard) can be more efficiently matched

with conditions, but are much less flexible than more descriptive events and also necessitate a clear a priori understanding of which events are relevant."

5.2.1.2 Formalization of the communications

According to Milner's Calculus of Communicating Systems (CCS) [Milner, 1989], the interaction between processes is signified by a set *L* of labels; '$label_x$' denotes input while '$\underline{label_y}$' denotes output. Typically '$label_x$' can only communicate with '$\underline{label_x}$'.

Other conventions in CCS are the following:

$\alpha.P$	The prefix constructor '.' is used to indicate succession. $\alpha.P$ means: "process *P* follows action α". The CCS formalism does not require α to have finshed before *P* starts, only that α has been observed before *P* begins.
$\underline{done}$	The special communication '$\underline{done}$' indicates that a process has terminated.
$P ; Q$	The construction 'P ; Q' indicates that the process *Q* will proceed after the process *P* emits the special communication '$\underline{done}$'.
$P + Q$	Summation '+' indicates choice. '$P + Q$' means: either *P* or *Q*.
$P \mid Q$	The composition symbol '|' indicates concurrency. If *P* and *Q* are defined as a succession of sub-actions, they may be interleaved (if $P = p_1.p_2$ and $Q = q_1.q_2$, then $P \mid Q = p_1.p_2.q_1.q_2 + p_1.q_1.p_2.q_2 + p_1.q_1.q_2.p_2 + q_1.q_2.p_1.p_2 + q_1.p_1.q_2.p_2 + q_1.p_1.p_2.q_2$).
$P \setminus L$	The symbol '\' is used to hide direct communication opportunities, i.e. (port. P) \ {port} can not communicate over labels 'port' and '$\underline{port}$'.

With the help of Milner's formalism, [Velthuijsen, 1992] describes the dynamic behaviour of a blackboard system by the following equations:

Let 'n' be the number of KS, and 'm' the number of available processors to be scheduled for their execution (it is supposed here that no priorities have been associated with KS):

$BBS = (BB \mid KS_1 \mid \ldots \mid KS_n \mid Sched^{(m)}) \setminus (L = \{read, write, start, end\})$

$BB = \underline{read}.BB + write.BB$

$KS_i = KS_i \mid KSI_i \qquad i = 1, \ldots, n$

$KSI_i = read(x).(Match_i(x); Act_i(x); \underline{done} + \underline{done}) \qquad i = 1, \ldots, n$

$Match_i(x) = 0$ if there is no match, $Match_i(x) = \underline{done}$ otherwise

Act_i(x) = start.Act'_i(x) i = 1, …, n
Act'_i = int-act.Act'_i + read.Act'_i+ write.Act'_i+ end.done where 'int-act' denotes an internal activity of the KS
Sched $^{(m)}$ = Sched | … | Sched 'm' times
Sched = start.end.Sched
Sched $^{(0)}$ = 0

5.2.1.3 About the control: applicability and selection

Applicability of a KS is decided by the mapping $Match_i$ between the current state of the blackboard and the condition of the KS. KS selection is a mapping *sel* determining a subset of existing KSI to be executed. This can be put in equations for sequential, parallel as well as distributed blackboards, but we consider it outside the scope of the current comparison.

5.2.2 AUSTIN: a blackboard with semantically controlled events

There is indeed a close matching between AUSTIN concepts and the ones present in blackboard architectures; we illustrate this in Table 16:

AUSTIN concepts	**Blackboard concepts**
utterance	Blackboard unit
addition or removal of utterance	Blackboard event
conversational pattern	KS
commissives	KS descriptor (data structure describing the action part of the KS)
starter of conversational pattern	KS condition
instantiation of conversational pattern	KSI
conversation node	action Act_i(x)
test (for invocation, activation or revocation of patterns)	$Match_i$(x)
AUSTIN conversational machine	Control unit

Table 16: Mapping between AUSTIN and blackboard concepts

The AUSTIN conversational machine, when considered as a blackboard, has the following characteristics:

i) read/write events have great expressivity; instead of being simple couples (attribute, value) they may be complex objects, e.g. conceptual graphs as exemplified in [3.9];

ii) the blackboard itself is not structured in distinct levels and/or panels, but is structured by the three relations between sentences : reductionOf, answerFor and triggerFor;

iii) because of this expressivity of sentences and this relational structure of the blackboard, the whole control is supported in a simple way: there is no distinction between 'triggering mechanism' and 'precondition', and the rule for answering questions plays the role of '*Match';*

iv) the control strategy is independent of the use case. This means that 'domain experts' need not worry about the control; it is fully embedded in the abstract machine; moreover this control leads to finite and deterministic computation of the initial state of the blackboard;

v) the knowledge sources are themselves structured in nodes which are the basic working components for the abstract machine; reasoning on the computation can therefore be easily done at the node level.

6 Summarizing the issues and planning future work

Starting from the motivation of assisting people in the collaborative building of an 'abstract model' of some 'situation of the real world' embedding their respective understandings in a friendly and compositional way, we have first proposed a framework which illustrates the interlacing between the individual generation of partial theories and their unified and computer-assisted logical derivation.

Taking inspiration in the ideas of McCarthy described in "Elephant 2000[77]", we have then provided the detailed design for an abstract machine ruling the composition of partial theories: AUSTIN.

In AUSTIN, the interaction with 'experts' expressing facts and generating hypotheses, as well as the interaction between logical components embedding partial theories, are based on 'speech acts' very close to the experts' initial conceptual schemes.

Partial theories can easily be embedded in AUSTIN programs under the shape of multi-steps algorithms which auto-assemble without any special effort from the people providing the partial theories. The acceptability condition as well as the main reasoning techniques upon these programs have been described. The AUSTIN computational model can be viewed as a middle way between declarative techniques and imperative programming; therefore aiming at a trade-off between expressivity and controllability.

Moreover, a Java prototype of the abstract machine exists and has been used to run the 'Sudoku filling problem'.

All along this thesis, we were driven by operational matters, until we could watch the abstract machine concretely work; on the way, we investigated many topics belonging to different fields, and became gradually aware of the fact that most of these investigations would benefit from some deepening, which means that the work is not finished.
Many directions unfold at this stage; we list them hereafter according to two main strands:

- a first strand consisting in going from the current prototype to an industrial tool, therefore entering the evolutionist cycle of experientially selected improvement;
- a second strand strengthening the formalization.

[77] As expressed in [McCarthy, 1998] and pasted in [2.1.2.2]

6.1 Future experimental work

Several directions of experimentation in the orange real world seem possible:

6.1.1 Improving interface and diagnostic facilities for programmers

Following the initial motivation of collaborative modelling; an interesting direction of improvement would be the full exploitation of the graphical aspects for the building and analysis of AUSTIN programs presented in Chapter 4:

- graphical design (i.e. designing as graphs) of conversational patterns with automatic implementation of dependencies between conversation nodes;
- graphical building (i.e. dragging and dropping words) of commissives, assertives and directives;
- pre-diagnostic of deadlocks and emission by the abstract machine of associated expressives!

6.1.2 Using AUSTIN for controlling the interaction between 'services'

The interleaving of partial logical derivations upon a set of initial data can be interpreted as a composition of "transformational services". The current flourishing of "Service Oriented Architectures" makes this topic quite in fashion today and solutions close to the conversational metaphor have long been promoted, e.g. [Jonquet and Cerri, 2005]. A natural application of the AUSTIN machine would be to address "service grouping", in the sense of allowing Web Services or Grid Services to dynamically collaborate and integrate into higher level services.

AUSTIN can easily provide successful interconnection between previously independent 'services' as soon as their respective input and output are described as *commissives* in a unified representation language. This means that integration of services will not be magically operated at the 'user' level, but can be easily enhanced at the 'service provider' level.

6.1.3 Parallelizing computation

Parallelisation of the invocation of the *conversational patterns* has already been implemented through separate threads corresponding to different invocations in the current Java prototype.

Parallelisation in the management of the "test pipe" and "transition pipe" on the basis of the ideas of [3.6.3] should allow further computational benefits.

The remaining bottleneck is the control of the global states of the calculus, i.e. the "blackboard" itself. Each time an atomic transition is operated, the set of all utterances is locked for all the others until this transition ends. There is some good hope in this direction though … it can easily be demonstrated that the calculus has the following property: if at one state the conversation becomes partitioned into "graphically independent sub-conversations", this "topology" is preserved until the end. Therefore, analyzing in detail how transitions interfere through overlapping sets of utterances, might lead to the management of independent groups of transitions, with subsequent "subsets of utterances" in which local states are defined.

Exploring the semantically-driven parallelism in the context of demanding computations might lead to useful results.

6.2 Future theoretical work

When putting AUSTIN in perspective, we have somewhat remained in between two blue-green approaches, both promising.

On one hand, studying AUSTIN from the viewpoint of blackboard architecture, and articulating its operational semantics with the formal description proposed by [Velthuijsen, 1992] may appear as the most natural continuation of our work; much could be learnt about the class of problems that AUSTIN can address with some pertinence.

On the other hand, giving a robust logical grounding to AUSTIN has been proposed to us and seems the most interesting path. According to the excellent [Gochet and Gribomont, 1994], it may be considered that any program can be viewed as a particular axiomatic system: running a program can be interpreted as automatically writing a proof within the system. Even if the path promises to be a little long from the basic notions of Set Theory to their application to Logical Programming, achieving such formalization would give a definite academic status to 'collaborative theory construction'.

7 References

[Alai, 2004] M. Alai, “A.I., Scientific Discovery and Realism, Minds and Machines”, Volume 14, Issue 1. February pp: 21 – 42, 2004.

[Alliott, 1993] J.-M. Alliott, “Using genetic algorithms for solving ATC conflicts” Proceedings of CAIA-93, Orlando, Florida, March 1993.

[Alliot & Schiex, 1993] J.-M. Alliot & T. Schiex, “Intelligence artificielle & Informatique théorique”, Cépaduès-éditions, 1993.

[Apt & van Emden, 1982] K. Apt, M. van Emden, “Contributions to the theory of Logic programming”, J. Assoc. Comput. Mach. 29-3:841-862, 1982.

[Austin, 1962] John L. Austin, “How to Do Thing with Words”, Cambridge (MA): Harvard UP, 1962.

[Austin, 1979] John L. Austin, “How to talk: Some simple ways”, in Philosophical papers , edited by James O. Urmson and Geoffrey Warnock, Oxford: Clarendon Press, 1979.

[Baget, 2001] J-F. Baget, “Représenter des connaissances et raisonner avec des hypergraphes » PhD Thesis, Université Montpellier 2, november, 2001

[Bratman, 1988] M.E. Bratman, D.J. Israel, and M.E. Pollack, “Plans and resource-bounded practical reasoning”. Computational Intelligence, 4(4): 349-355, 1988.

[Breuker et al., 2005] J. Breuker, S. A. Cerri, P. Dugénie, M. Eisenstadt and P. Lemoisson, “Conceptual and Organisational Framework for Conversational and Collaboration Processes”, ELeGI European Integrated Project/D20, 2005.

[Breuker & Greef, 1993] Joost Breuker and Paul De Greef, “Modelling system-user cooperation”, in Guus Schreiber, Bob Wielinga, and Joost Breuker editors, KADS: a Principled Approach to Knowledge Engineering, pages 47 –70. Academic Press, London, 1993.

[Cerri, 1999] S. A. Cerri, “Dynamic typing and lazy evaluation as necessary requirements for Web languages”, Presented at European Lisp User Group Meeting, Amsterdam,1999.

[Cerri et al, 2000] S. A. Cerri, J. Sallantin, E. Castro, D. Maraschi, “Steps towards C+C: a Language for Interactions”, AIMSA 2000.

[Charaudeau, 1992] P. Charaudeau, “Grammaire du sens et de l’expression”, Hachette education, 1992

[Clancey, 1997] W. J. Clancey, “Situated Cognition: On Human Knowledge and Computer Representations”, Cambridge University Press, 1997.

[Clark, 1996] H.H. Clark. “Using Language”, Cambridge University Press, 1996.

[Clark and Gregory, 1981] K. L. Clark, and S. Gregory, “A relational language for parallel programming”, in Proceedings of the ACM Conference on Functional Languages and Computer Architecture, ACM, New York, pp. 171-178, 1981.

[Colmerauer and Roussel, 1992] A. Colmerauer and P. Roussel, “The birth of Prolog”, in The second ACM SIGPLAN conference on History of programming languages, p. 37-52, 1992.

[Cruvier, 1993] D. Cruvier, “The tumultuous Story of the Search for Artificial Intelligence”, Basic Books, 1993.

[Dijkstra, 1976] E. Dijkstra, “A Discipline of Programming”, Prentice-Hall, Englewood cliff, 1976.

[Durand, 2004] B. Durand et A. Zvonkin, “Complexité de Kolmogorov, in L’héritage de Kolomogorov en mathémlatiques ”, Belin, 2004

[Edelman, 1988] G. M. Edelman, “Neural Darwinism: The Theory of Neuronal Group Selection”, Basic Books, 1988.

[Edelman, 1992] G. M. Edelman, “Bright Air, Brilliant Fire: On the Matter of Mind”. Basic Books, 1992.

[Edelman and Reeke, 1982] G. M. Edelman, and G. N. Reeke Jr, “Selective networks capable of representative transformation, mimites generalizations, and associative memory”, Proc. Natl. Acad. Sci. USA 80:4384-88, 1982

[Edelman & Tononi, 2000] G. M. Edelman, Giulio Tononi “A Universe of consciousness”, Basic Books, 2000.

[Gerth et al., 1988] R. Gerth, M. Codish, Y. Lichtenstein, and E. Shapiro, “Fully abstract denotational semantics for Flat Concurrent Prolog”, in Proceedings of the IEEE Symposium on Logic in Computer Science. IEEE, New York, pp. 320-333, 1988.

[Gochet and Gribomont, 1994] P. Gochet and P. Gribomont, “Logique, méthodes formelles pour l’étude des programmes”, Hermes, 1994.

[Gold, 1967] E. M. Gold, “Language identification in the limit”, Inform. Control, 10, 447-474, 1967.

[Grice, 1975] H. P. Grice, “Logic and conversation”, in P. Cole and J. Morgan, editors, Syntax and semantics: speech acts, New York, Academic Press, 1975.

[Gruber,1992] H. Gruber, “Comparaison de différentes méthodes d’IA pour la résolution de problèmes”, Rapport technique 92-024, Centre d’Etudes de la navigation aérienne, Juin 1992.

[Gurevich, 1988] Y. Gurevich, “On Kolmogorov machines and Related Issues”, in The Logic in Computer Science Column, Bulletin of European Assoc. for Theor. Comp. science, Number 35, 71-82, June 1988.

[Harel and Pnueli, 1985] D. Harel, and A. Pnueli, “On the development of reactive systems”, in Logics and Models of Concurrent Systems, K. R. Apt. Ed. Springer Verlag, New York, 1985.

[Hewitt, 1977] C. Hewitt, “Viewing Control Structures as Patterns of Passing Messages”, AI Journal: 323-354, 1977.

[Hewitt, 1985] C. Hewitt, “The challenge of open systems”, Byte Mag., 223-242, 1985.

[Jonquet and Cerri, 2005] C. Jonquet and S.A Cerri, “The STROBE Model: Dynamic Service Generation on the Grid”, Applied Artificial Intelligence Journal, Vol. 19, n.9-10, pp. 967-1013, 2005.

[Keil & Goldin, 2005] D. Keil, D. Q. Goldin, “Indirect Interaction in Environments for Multi-agent Systems”, E4MAS: 68-87, 2005.

[Knuth, 1973] D. E. Knuth, “The Art of Computer Programming”, Second Edition, Volume 1/Fundamental Algorithms, Addison-Wesley, 1973.

[Kolmogorov, 1963] A. N. Kolmogorov and V.A. Uspensky, “On the definition of an algorithm”, Translations, series 2, Amee. Math. Soc. 29, 1963.

[Holland et al, 1986] J. Holland, K. Holyoak, R. Nisbett, P.Thagard, “Induction. Processes of Inference, Learning and Discovery”, MIT Press, Cambridge, Mass., London, England, 1986.

[Hill, 1974] R. Hill. “LUSH-resolution and its completeness”, DCL Memo 78, Dept. of Artificial Intelligence, Univ. of Edinburgh, Scotland, 1974.

[Jagannathan et al, 1989] V. Jagannathan, R. Dodhiawala, L. S. Baum, “Blackboard Architectures and Applications”, Perspectives in Artificial Intelligence, Vol 3, Academic Press, inc. Harcourt Brace Jovanovich, Publishers, 1989.

[Laird et al, 1987] J. Laird, A. Newell, and P. Rosenbloom, “Soar: An Architecture for General Intelligence”, Artificial Intelligence, 33: 1-64, 1987.

[Langley, 2000] P. Langley, “The computational Support for Scientific Discovery”, in International Journal of Human-Computer Studies, 53, 393-410, 2000.

[Lemoisson, Cerri, Sallantin, 2005] P. Lemoisson, S.A. Cerri and J. Sallantin , “Conversational Interactions Among Rational Agents”, in Towards the Learning GRID: Advances in Human Learning Services , IOS Press 2005.

[Lemoisson and Cerri, 2005] P. Lemoisson, and S.A Cerri, “Interactive Knowledge Construction in the Collaborative Building of an Encyclopedia”, in Applied Artificial Intelligence Journal, Vol. 19, n.9-10, pp. 933-966, 2005.

[Lesser et al., 1975] V.R. Lesser, R.D. Fennell, L.D. Erman, and D.R. Reddy, “Organization of the Hearsay-II speech understanding system”, IEEE Transactions on Acoustics, Speech, and Signal Processing, ASSP-23(1):11-24, 1975.

[Lifschitz, 2002] V. Lifschitz “Answer set programming and plan generation”, in Artificial Intelligence. 138(1-2): 39-54, 2002.

[Lloyd, 1987] J. W. Lloyd “Foundations of Logic Programming”, 2nd ed. Springer-Verlag, New York, 1987.

[Lorenz & Popper, 1995] K. Lorenz and K. Popper, “L’avenir est ouvert”, Champs Flammarion, 1995.

[Lukasiewicz, 1970] J. Lukasiewicz, “Creative elements in science”, in J. Lukasiewicz, Selected Works, North Holland, Amsterdam, pp 12-44, 1970.

[Magnani, 2001] L. Magnani, “Abduction, Reason and Science / Processes of Discovery and Explanantion”, Kluwer Academic/Plenum Publishers, 2001.

[McCarthy, 1998] J. McCarthy, “Elephant 2000: A Programming Language Based on Speech Acts”, appeared: 6 Nov 1998 written: 1989.

[Mill, 1843] J. S. Mill, “Système de logique deductive et inductive. Livre III de l’induction ”, 1843.

[Milner, 1989] R. Milner, « Communication and Concurrency”, Prentice-Hall International Series in Computer Science, Prentice-Hall, London, 1989.

[Minsky, 1967] M. Minsky, “Computation: Finite and Infinite Machines”, Prentice-Hall, Inc., N.J., 1967.

[Minsky, 1998] M. Minsky, “The Society of Mind”, Touchstone Edition, p. 308, 1998.

[Mugnier & Chein, 1996] M.-L. Mugnier, M. Chein, “Représenter des connaissances et raisonner avec des graphes“, Revue d’Intelligence Artificielle, vol. 10, n°1, pp 7-56, 1996.

[Newell, 1962] A. Newell, “Some problems of the basic organization in problem-solving programs”, in: M.C. Yovits, G.T. Jacobi, and G.D. Golstein, eds., Proceedings Conference on Self-Organizing Systems, pp393-423. Spartan Books, 1962.

[Newell, 1990] A. Newell, “Unified Theories of Cognition”, Harvard University Press, 1990.

[Nobrega et al., 2000] G. Nobrega, E. Castro, P. Malbosse, J. Sallantin, and S.A. Cerri, “A Framework for supervised conceptualizing”, in: Benjamins, R., A., Gomez-Perez, A., Guarino, N. and Uschold, M. (eds) Workshop on Applications of Ontologies and Problem Solving Methods, ECAI 2000: The 14th Biennal European Conference on Artificial Intelligence, Berlin. In press, 2000.

[Peirce, 1955] C. S. Peirce, “Philosophical writings of Peirce”, J. Buchler ed, Dover, New York, 1955.

[Peirce, 1958] C.S. Peirce, “Collected Papers of Charles Sanders Peirce”, Vols. 1-6, Charles Hartshorne and Paul Weiss (eds.), Vols. 7-8, Arthur W. Burks (ed.), Harvard University Press, Cambridge, MA, 1931-1935, 1958.

[Piaget] http://tip.psychology.org/piaget.html

[Piaget, 1968] J. Piaget, “Sagesse et illusions de la philosophie”, PUF, Paris, 1968.

[Pnueli, 1968] A. Pnueli, “Applications of temporal logic to the specification and verification of reactive systems: A survey of current trends”, in Current Trends in Concurrency, Overviews and Tutorials. Lecture Notes in Computer Science, Vol. 224, 1986.

[Popper, 1959] K. R. Popper, “The Logic of Scientific Discovery”, Hutchinson, London, New York. 1959.

[Post, 1936] E. L. Post, “Finite Combinatory Processes - Formulation 1”, Journal of Symbolic Logic, 1, 103-105, 1936.

[Reeke and Edelman, 1984] G. N. Reeke Jr and G. M. Edelman, “Selective networks and recognition automata”, Ann. N.Y. Acad. Sci. 426:181-201, 1984.

[Revonsuo & Newman, 1999] A. Revonsuo, and J. Newman, "Binding and Consciousness", Consciousness and Cognition 8, 123-127, 1999.

[Rice, 1953] H. G. Rice, "Classes of Recursively Enumerable Sets and Their Decision Problems", Trans. Amer. Math. Soc. 74, 358-366, 1953.

[Robinson, 1965] J. A. Robinson, "A machine oriented logic based on the resolution principle", J. ACM 22, 1, 23-41, 1965.

[Sacks et al., 1974] H. Sacks, E.A. Schlegoff, and G. Jefferson, "A simplests systematics for the organization of turn taking in conversation", Language, 50:696–735, 1974.

[Searle, 1969] J.R. Searle, "Speech acts", Cambridge University Press, 1969.

[Searle, 1999] J.R. Searle, "Mind, Language and Society", Basic Books, 1999.

[Searle, 1983] J.R. Searle, "Intentionality", Cambridge University Press, 1983.

[Shapiro, 1983] E. Shapiro, "A subset of Concurrent Prolog and its interpreter", ICOT Tech. Rep. TR-003, Institute for New Generation Computer Technology, Tokyo, 1983.

[Shapiro, 1991] E. Shapiro, "Inductive inference of theories from facts/ Computational logic (essays in honour of Alan Robinson)", Chap7, edited by Jean-Louis Lassez and Gordon Plotkin, The MIT Press, 1991.

[Shapiro, 1989] E. Shapiro, "The Family of Concurrent Logic Programming Languages", @1989 ACM 0360-0300/89/0900-0413, 1989

[Sowa, 1976] J. F. Sowa, "Conceptual Graphs for a Database Interface", IBM Journal of Research and Development, pp6-57, 1976.

[Sowa, 1984] J. F. Sowa, "Conceptual Structures: Information Processing in Mind and Machine", Addison-Wesley, Reading, MA., 1984.

[Turing, 1937] A. Turing, "On computable numbers, with an application to the 'ent-scheidungsproblem'", Proc. of the London Math. Soc. 2nd series, 42 :230-265. Correction ibid Vol. 43 pp 544-546, 1937.

[Ueda, 1986] K. Ueda, "Guarded Horn Clauses", in Logic Programming. Lecture Notes in Computer Science, Vol. 221, Springer-Verlag, New York, pp.168-179, 1986.

[Uspensky, 1990] V. Uspensky, A. Semenov, "Algorithms:main ideas and applications", (translated from Russian by A. Shen), Kluwer, 1993.

[Van Roy & Haridi, 2004] P. van Roy and S.Haridi, "Concepts, Techniques and Models of Computer programming", MIT Press, 2004.

[Van Rossum, 1995] G. van Rossum: “Python Library Reference; Python Reference Manual; Python Tutorial; Extending and Embedding the Python Interpreter”, CWI Reports CS-R9524; CS-R9525; CS-R9526; CS-R9527, 1995.

[Velthuijsen, 1992] H. Velthuijsen, “The Nature and Applicability of the Blackboard Architecture”, PTT Research, 1992.

[Vygotsky, 1978]. L.S. Vygotsky, “Mind in Society. The Development of Higher Psychological Processes”, Edited by M. Cole, V. John-Steiner, S. Scribner and E. Souberman.Harvard University Press. Cambridge, Massachussets, 1978.

[Yang and Aiso, 1986] R. Yang, and H. Aiso, “P-Prolog: A parallel logic language based on exclusive relation”, in Proceedings of the 3rd International Conference on Logic Programming. Lecture Notes in Computer Science, Vol. 225, E. Shapiro, Ed. Springer-Verlag, New York, pp. 255-269, 1986

8 End notes

8.1 A long quote from Mario Alai

From [ALAI, 2004]:

“In [Langley et al, 1987], BACON4 (one of those programs intended to mechanize scientific discovery) can deal with cases in which data are names of objects …when the values of a given property of one object (e.g. the current of one battery) vary with the related objects (e.g. with different wires), then it must be postulated that there is a property of these related objects which is responsible for such variation (e.g., conductance), and whose values are proportional to those of the first property (current) …The authors stress that this amounts to the introduction of theoretical, i.e. unobservable, properties …Nonetheless, this does not seem to be a full-fledged example of theorization, since the new property is not embedded in a whole theory describing it, its nature, causes, functioning, effects, etc. Its introduction, here, is just matter of detecting a regularity in the behaviour of observable objects and properties, and ascribing to it as a cause a property which is identified just by means of this particular effect … Another heuristic tells BACON4 to look for common divisors and their regularities. In this way the program may be applied to chemical research, and starting from the proportions of weights and volumes of elements in compounds it finds the molecular and atomic weights of various elements and compounds. This is not to say it discovers either the

molecular or atomic theory, however; for it finds just numbers, and it is only in the light of our knowledge of atomistic chemistry that we may interpret those numbers as atomic or molecular weights. The last version, BACON5, was created by adding to the earlier ones the notion of physical symmetry, and the rule that if a particular relation holds among a set of variables, (e.g., two objects with respective initial and final velocities), it must be presumed that it holds among variables of the same type (e.g., two different objects with respective initial and final velocities). Hence, on condition of being told which variables are of the same type, the system is able to speed up significantly its search for regularities (such as the law of conservation of momentum, Snell's law of refraction, or Joule's law). BACON4 is thus theory-driven, i.e. it imitates those cases of actual science in which research is not merely based on data, but on theoretic presuppositions as well. Even more theory-driven is a different system, BLACK, which has inbuilt the notion that certain properties are conserved… starting from data about reactions, and assuming that the number of molecules involved in a reaction is proportional to the volumes of the respective elements, DALTON infers the molecular structure of compounds. Moreover, assuming that atoms are conserved in reactions, and that molecules are composed of the smallest number of atoms compatible with the law of conservation and the known molecular structures, DALTON infers the atomic structures of elements and compounds.

… Donald Gillies contrasts them (the results of Simon's work) with the programs produced by disciples of Alan Turing and researchers working in their tradition, such as Ehud Shapiro, Stephen Muggleton, Donald Michie, Edward Feigenbaum, Bruce Buchanan, J.R. Quinland and Ivan Bratko, programs which have successfully been applied to the solution of practical problems and have made new (hence actual) discoveries … For instance, DENDRAL, developed since 1965, accomplishes what an expert chemist might do, inferring the molecular structure of organic compounds from their mass spectrogram. ASSISTANT has been able to diagnose various kinds of disease better than human specialists. GOLEM was able in 1991 to predict the secondary structure of proteins from their primary structure… Gillies credits such rules as "new laws of nature" discovered by GOLEM. But it is far from clear that they qualify as such, because (a) they have a very low level of generality, as the above example shows, (b) as he himself concedes they fail in about 20% of cases, and therefore (c) it is dubious that they describe actual causal relationships, as opposed to contingent statistical correlations… all these programs (DENDRAL, ASSISTANT, GOLEM) produce generalizations connecting a target property (such as a secondary structure character, or a particular disease), to the presence or absence of symptomatic properties (such as primary structure characters, or symptoms

manifested by patients); they do so by checking which symptomatic properties are present or absent when the target property is known to be present. This is to say, they practice, though with sophisticatedly iterated procedures, nothing but Baconian or Millian induction. No wonder therefore these programs have only discovered low level generalizations, and no theories, theoretical laws, entities or explanations. Just as enumerative induction, also Bacon's tables and Mill's canons, in fact, may establish horizontal links among empirically known entities or properties, but no vertical links among observable and non-observable levels of reality."

… An example (of different approach) could be the proposal advanced by John Holland, Keith Holyoak, Richard Nisbett and Paul Thagard (henceforth HHNT) in [Holland et al, 1986]. They try to reconstruct the cognitive processes by jointly relying on cognitive science, philosophy and computer science, and describe a program ("PI", for "processes of induction") which purports to replicate the crucial features of such processes.

Knowledge is represented by HHNT as the construction of mental models of the environment. Environment, in turn, is described as consisting of states and transition functions among them … there can be long chains of states and transition functions, and particularly important chains are those in which the initial state represents a theoretical or practical problem for the knowing subject, and the final state represents its solution (HHNT stress the importance of pragmatics in their approach). For HHNT the mind is like a bulletin board, on which "messages" (i.e. propositions) are posted …In mental models states are represented by categoric messages (e.g.: "all bodies move"; "this body is moving"; etc.), and transition functions by hypothetical messages, or "rules" (e.g., "if a body moves at time t, it will stand still at time t'")…As it happens, however, there is only limited room on a bulletin board, as well as in the human mind. Therefore, not all potentially activated messages will be posted, and messages will compete for room on the board. Besides, mutually incompatible messages that might be posted will be in competition even independently of room scarcity. The competition takes place more or less as it happens on the economic market: messages and rules in a chain can be viewed as suppliers, middlemen and consumers, where selling and buying consists in activating or being activated. When the final ring of a chain constitutes the successful solution to a problem (hence the importance of the pragmatic dimension in this approach), its success is comparable to profit, which is duly shared with each preceding ring, as each buyer pays back the goods or services that reached him through the chain … In practice, the environment's feed-back gradually reinforces successful beliefs and extinguishes unsuccessful ones, just like market reactions make efficient firms flourish and inefficient

ones go bankrupt… Admittedly, PI, HHNT's program, has not (yet) made any new discovery, nor any old one as complex and important as those achieved by Simon's programs."

8.2 Post systems

Post System are defined by:

- a finite alphabet Σ_C of constants: e.g. $\Sigma_C = \{a, b\}$
- a finite alphabet Σ_V of variables with $\Sigma_C \cap \Sigma_V = \emptyset$: e.g. $\Sigma_V = \{x\}$
- a finite set $\mathcal{A}$ of words on Σ_C or axioms: e.g. $\mathcal{A}_1 = \{aabb, ab, aab\}$ OR $\mathcal{A}_2 = \{bb, abaa, b\}$
- a finite set P of rules $(\alpha_1, \alpha_2 \ldots, \alpha_k) \sim> \alpha$ on words on $\Sigma_C \cup \Sigma_V$
  e.g. $P = \{(\alpha_1, x) \sim> \alpha_1 x\}$ concatenation

Starting from each set of axioms, it is possible to produce "theorems" by applying several times the rules of P, for instance the same theorem 'abaababaabb' is produced below by two different Post systems with the same alphabets and concatenation rule, but with different sets of axioms $\mathcal{A}_1$ and $\mathcal{A}_2$:

i. ab $\rightarrow$ abaab $\rightarrow$ abaabab $\rightarrow$ abaababaabb is obtained from $\mathcal{A}_1 = \{aabb, ab, aab\}$

ii. abaa $\rightarrow$ abaab $\rightarrow$ abaababaa $\rightarrow$ abaababaabb is obtained from $\mathcal{A}_2 = \{bb, abaa, b\}$

The general question of deciding for two given sets of axioms whether a common theorem can be derived is known as the *POST's Correspondence Problem* and is not decidable.

Post systems can serve as a foundation for logical systems such as the propositional calculus; this is illustrated in [Alliot & Schiex, 1993] in the following way:

- $\Sigma_V = \{A, B, C, \ldots\}$
- $\Sigma_C = \{\rightarrow, \neg, a, b, c, \ldots\}$
- $P = \{((A, A \rightarrow B) \sim> B)$
- $\mathcal{A}$ = set of axioms

  $A \rightarrow (B \rightarrow A)$

  $((A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C)))$

  $((\neg B \rightarrow \neg A) \rightarrow ((\neg B \rightarrow A) \rightarrow B))$

8.3 Shapiro's model inference problem

[Shapiro, 1991] defines and formalizes the 'model inference problem':

"A model inference problem is an abstraction of the problem faced by a scientist, working in some domain under some fixed conceptual framework, performing experiments and trying to find a theory capable of explaining the results. In this abstraction the domain of inquiry is the domain of some unknown model M for a given first order language L, experiments are tests of the truth of sentences of L in M, and the goal is to find a set of true hypotheses that imply all the true testable sentences."

To solve the 'model inference problem', Shapiro proposes a powerful algorithm going through conjectures and refutations; i. e. able to backtrack and eliminate wrong hypotheses on the basis of counter-examples.

Shapiro's formalization relies on the following ingredients:

1. a first order language in clausal form L, with finitely many predicates and function symbols.
2. a subset L_0 of L : the observational language.
3. a subset L_H of L including L_0 : the hypothesis language .
4. an oracle[78] for the unknown model M, that when given an observational sentence returns true if it is true in M, false otherwise.

On the basis of these ingredients, Shapiro's algorithm browses in a sophistically optimized way through an enumerable set of possible theories, until a proper model of a given set of observations is found, under the following restrictions:

- L_H is fixed: the proposed theories have to find their expression within a pre-defined language with finitely many predicates and function symbols. As the author outlines: "it should be noted that the assumption that the hypothesis language is fixed avoids one of the major problems in scientific discovery, which is the invention of a new conceptual scheme".
- an "admissibility requirement" has to be made on L_0 and L_H, which "essentially says that L_0 must contain enough information to refute any false theory that attempts to be L_0-complete: "the admissibility requirement is a necessary, but not sufficient condition for the solvability

[78] what people in the Turing tradition call 'oracles' corresponds to what experimental sciences call 'experiments': it is because experiments happen outside formal systems that the notion of oracle is required.

of a model inference problem … to understand the concept of a palindrome (a string being the reversal of itself), a person must know first the concept of string reversal. Terms that denote such concepts are called theoretical terms and have to be known to, or built into the inference algorithm as 'theoretical concepts'.

- a condition about M, so that the check of a fact against a conjecture is bound in complexity (it is in general undecidable). This expresses the fact that the algorithm will infer "not too complex" models, which also means that in will infer models in "reasonable" time.

Referring to [Gold, 1967], Ehud Shapiro summarizes: "… the most one can expect of an inductive inference algorithm is that after examining a finite number of facts about the model and making a finite number of wrong conjectures, such an algorithm will correctly conjecture a finite set of hypotheses true in the model, which imply all true observational sentences."

8.4 Human parsing

Here is a little test to which you must not give more than 10 seconds, otherwise it is not valid. Then you may analyze your result next page.

Please count the number of 'F' in the text below:

++++++++++++++++++++++++++++

FINISHED FILES ARE THE RESULT OF YEARS OF SCIENTIFIC STUDY COMBINED WITH THE EXPERIENCE OF YEARS

++++++++++++++++++++++++++++

Done!

You may turn the page …

How many 'F'? Three?

False, there are six ‘F’ in the text … human brain usually skips ‘OF’ …

8.5 Invocation and revocation of patterns: introductive discussion

Let us consider the following speech acts:

i. $cell_x^y$ / *directive*/ to moderator: “what are my rejected value”?

ii. $cell_x^y$ / assertive/ to moderator: “value ‘5’ is rejected for my neighbours”

We wish the ‘moderator’ to decide invocations of the ‘conversational pattern’ on the only basis of currently available *assertives* or *directives*, according to the following examples:

a. the *assertive* “$cell_5^7$ is ready” leads the moderator to activate ‘$cell_x^y$’ under the context ‘row = 5’ and ‘column = 7’

b. the *directive* “what solution for sudoku 9×9 ?” leads the moderator to concurrently ‘launch’ all cells by producing the 81 assertives “$cell_x^y$ is ready” …

When we formally implement this mechanism in the abstract machine, the decision of the moderator about whether such *assertive* or *directive* can activate such pattern will be based on a syntactic comparison between the given sentence (*assertive* or *directive*) and a special sentence (*starter*) embedded in the conversational pattern.

The advantage of deciding invocation on the basis of **current** *assertives* and *directives* rather than relying on **special** ‘*invoking directives’* is the following: thanks to *commissives* which “pre-announce” the current *assertives* and *directives*, the moderator knows in advance the potential invocations. He therefore can *revoke* a ‘conversational pattern’ as soon as it is established that it will not be *invoked* any more! *Revoking* a ‘conversational pattern’ has important consequences: the *commissives* initially emitted by the ‘conversational pattern’ are removed, and the global answering to questions is positively affected.

To understand this, let us come back to the commissive (ii.) “value ‘x’ is rejected for neighbours(y)” emitted at the beginning of the conversation by the conversational pattern ‘CELL’. We need that commissive to be removed if we want ‘$cell_2^8$’ to have an answer to its question (i.) “what are my rejected value?”. This means that we need to revoke ‘CELL’ after it has been concurrently invoked under the 81 contexts in order to answer the questions.

8.6 About asynchronous collaboration: Wikipedia and AUSTIN

AUSTIN has been designed to compute already built "partial theories", by delegating to logical artefacts the rewriting of commissives into assertives or directives (changing its status from *non-definitive* to *definitive*) in the context of a single closed conversation. In Wikipedia[79], "the free encyclopaedia that anyone can edit", the situation is quite different: the stabilizing of aggregates of shared knowledge is left to the appreciation of authors/readers in the context of an endless open conversation; the moderation is much less formal, and yet extremely efficient.

In order to better understand the similarities and differences between the two paradigms, we propose to consider Wikipedia as a use case for AUSTIN, by first proposing a general idea through as simplified conceptualization, then discussing the problems encountered.

8.6.1 A simplified conceptualization

Starting from the stable concepts of **author**, **article**, **topic** and **status** of an article, we suppose that:

- each article is written by one author and is about one topic;
- there are two possible status for an article : "ready" or "under construction".

Each time an author '$Author^i$' starts an article about a given topic '$Topic^j$', he emits:
Commissive (*, $Author^i$, $Topic^j$) = "Status is (*) for the article by ($Author^i$) about ($Topic^j$)", where (*) pre-announces a future value for the status.

When questioned about the status of his article about '$Topic^j$', '$Author^i$' either emits:
Assertive (ready, $Author^i$, $Topic^j$) = "Status is (ready) for the article by ($Author^i$) about ($Topic^j$)", or:
Assertive (under construction, $Author^i$, $Topic^j$) = "Status is (under construction) for the article by ($Author^i$) about ($Topic^j$)"

Because of the interdependency between topics authors do not write articles without having previously read about related topics. Each time an author '$Author^i$' needs some information about a given topic '$Topic^j$', he emits a directive:
Directive ($Topic^j$) = "What about ($Topic^j$)?"

Our moderating rules impose each directive emitted by a conversational pattern to be pre-announced, we therefore add a commissive which will be rewritten into the directive:
Commissive ($Topic^j$) = "What about ($Topic^j$)?"

[79] http://en.wikipedia.org/wiki/Main_Page

Each writing of an article by an author about a given topic is described in the AUSTIN machine by a two-steps conversational pattern in the following way:

conversational pattern: ARTICLE (i, j)

corresponding to the writing by the human Wikipedia contributor '$Author^i$' of an article about '$Topic^j$', for which we suppose that '$Topic^a$', '$Topic^b$'and '$Topic^c$' are pre-requisites

starter: "($Topic^j$)"

the starter is triggered according to: *Directive* ($Topic^j$) triggerFor "($Topic^j$)"

commissives:

Commissive (*, $Author^i$, $Topic^j$)

Commissive ($Topic^a$)

Commissive ($Topic^b$)

Commissive ($Topic^c$)

startNode: ARTICLE (i, j)-1

drivers: none

input: none

output: directives requesting information about '$Topic^a$', '$Topic^b$'and '$Topic^c$', which replace the corresponding commissives.

Directive ($Topic^a$)

Directive ($Topic^b$)

Directive ($Topic^c$)

postNode: ARTICLE (i, j)-2

drivers: ARTICLE(i, j)-1

input: answers for the three directives, according to:

Assertive (ready, $Author^n$, $Topic^k$) answerFor "($Topic^k$)"

Assertive (under construction, $Author^n$, $Topic^k$) answerFor "($Topic^k$)"

special mechanism: when all the answers to all directives have been provided by AUSTIN, the postNode asks a question to '$Author^i$': "is your article ready?". In case 'YES' the node rewrites *Commissive* (*, $Author^i$, $Topic^j$) into *Assertive* (ready, $Author^i$, $Topic^j$). In case 'NO' the node rewrites *Commissive* (*, $Author^i$, $Topic^j$) into *Assertive* (under construction, $Author^i$, $Topic^j$).

output:

Assertive (ready, $Author^i$, $Topic^j$) or *Assertive* (under construction, $Author^i$, $Topic^j$)

On the basis of this pattern, the collaboration goes though the following steps:

1. ‘Author^{i}’ once decides that he starts an article about “(Topic^{j})”; he knows he will need information about “(Topic^{a})”, “(Topic^{b})” and “(Topic^{c})” and therefore starts a conversation with the three directives *Directive* (Topic^{a}), *Directive* (Topic^{b}) and *Directive* (Topic^{c}).
2. ‘Author^{i}’ gets two types of answers: some articles are ready, others are under construction. As soon as he thinks his intention to write is mature enough for the others to know about it, ‘Author^{i}’ creates the conversational pattern ARTICLE (i, j).
3. starting from that moment, whenever a Wikipedia user will ask for “(Topic^{j})”, ARTICLE (i, j) will be invoked. Each time this happens, ‘Author^{i}’ is informed about the status of the many articles about “(Topic^{a})”, “(Topic^{b})” and “(Topic^{c})” and then asked whether his own article is ready … until he once says ‘YES’ because the article has been written by him and made available to the others.

8.6.2 Discussion of the case

A first remark is that we have introduced a “question to a human” within the conversation node ‘ARTICLE (i, j)-2’. This can be understood as a special function, like the ‘random function’ and does not impact the major properties of the calculus: the node remains a correct node (operating a reduction on sentences) as long as its output is in correct relation with its input. In fact it is a very useful feature of the AUSTIN calculus to be able to integrate ‘oracles’ in a confluent conversation.

A second remark is that more generic patterns, like ‘CELL’ in ‘Sudoku’, adapting to each particular case, would be more elegant. This is just a matter of conceptualization, and can easily be addressed by introducing explicit information about which ‘author’ writes about which ‘topic’, and how ‘topics’ are interdependent We can imagine a pattern distributing the original question “What about (Topic^{j})?” into the appropriate questions “What (status) for article by (Author^{i})?”about (Topic^{j})?” and integrating the answers provided in all cases.

A deeper remark is that we have split a single endless Wikipedia conversation in myriads of close AUSTIN conversations. The two status (‘under construction’ and ‘ready’) are both informative and therefore *definitive*, so that there is no change of status for an article within a given AUSTIN conversation. It therefore must happen between AUSTIN conversations … so that authors may give different answers at different moments to the question of the status of the article. This leads to two questions:

- the question of ‘the state of the encyclopaedia’ : “how to capitalize knowledge across the concurrent conversations?”
- the question of ‘the human bottleneck’ : “how to avoid the fact that each author is going to be asked the same questions about the status of each of his articles within each concurrent conversation?”

8.6.3 Asynchronous capitalization upon synchronous conversations

Our two questions about ‘the state of the encyclopaedia’ and ‘the human bottleneck’ show the deep difference in paradigm between Wikipedia as an open asynchronous collaboration and AUSTIN managing closed synchronous conversations. In order to reconciliate the two, we must clearly distinguish between what is “stateful” and what is “stateless”:

- the knowledge is “stateful” (evolve along conversations) : which ‘author’ writes about which ‘topic’, how ‘topics’ are interdependent, and which ‘status’ for which article;
- the conversational patterns are “stateless” (invariant along conversations).

Therefore, the increasing knowledge has to be stored apart from AUSTIN; it will play the role of initial input at the beginning of each conversation and will be concurrently updated by the resulting assertives. The general idea becomes the idea of a shared Wikipedia database for *assertives* (what already is) in relation with as many AUSTIN conversations as necessary. We must observe however that the nature of these residual asynchronous conversations has considerably changed since our first conceptualization, from interaction between artefacts representing authors, to interaction between a search engine and a database.

Résumé: Cette thèse a pour objectif d'assister la construction collaborative d'une théorie ; chaque individu doit pouvoir formuler et tester empiriquement sa propre compréhension partielle, et sans aucun effort supplémentaire contribuer à une compréhension globale partagée par le groupe.

Une série d'explorations touchant aux mécanismes biologiques sous-tendant la cognition, au rôle du langage en tant que vecteur de partage et de formalisation et au cycle global de construction de théorie, conduisent à un cahier des charges pour une 'machine abstraite' qui compose des théories partielles au sein d'une conversation.

La machine abstraite 'Austin' est ensuite spécifiée de façon détaillée ; elle est fondée sur l'utilisation de trois types d'actes de langage : assertions, questions et promesses. Des 'schémas conversationnels normalisés' y concourent au traitement de l'information. De bonnes propriétés sont démontrées : terminaison en temps fini, confluence (le résultat final n'est pas affecté par l'indéterminisme du aux échanges concurrents lors de la conversation), complexité raisonnable, composition naturelle des programmes reconnus par cette machine abstraite, facilité d'analyse graphique pour ces programmes. Une grammaire générative inspirée des graphes conceptuels est proposée pour les énoncés. La 'résolution collaborative de Sudoku' est développée comme illustration du calcul basée sur cette grammaire, puis programmée sur un prototype Java de la machine abstraite.

Par comparaison à la 'programmation logique concurrente', puis aux 'systèmes multi-experts à architecture blackboard', 'Austin' apparaît être un compromis entre le 'modèle déclaratif' et le 'modèle impératif', combinant la contrôlabilité du premier avec l'expressivité du second.

Mots clés: construction de théorie, actes de langage, calcul distribué

Summary: Our work aims at assisting people in the collaborative building of a theory in such a way that each individual formulates and empirically tests his own partial understanding and can without extra effort contribute to a testable theory shared by the whole group.

A multi-disciplinary survey touching on the underlying biological mechanisms supporting cognition, linguistic communication as a means for sharing and formalizing, and the theory building process, leads to a series of requirements for an 'abstract machine' able to compose partial theories within a conversation.

The 'Austin' abstract machine is then fully specified; it is based on three speech acts: 'assertives', 'directives' and 'commissives'. 'Conversational patterns' concurrently achieve the processing of information inside Austin. Some good properties are demonstrated: end in finite time, confluence (the result is not affected by the concurrency), reasonable complexity, natural composition of programs recognized by the abstract machine, graphical analysis facilities. A generative grammar based on conceptual graphs is proposed for the sentences embedded in the speech acts. 'Collaboratively solving a Sudoku' illustrates this grammar, then is implemented on Java prototype of the abstract machine.

By comparison to 'concurrent logic programming', and then to 'blackboard architecture', 'Austin' appears as a trade-off between the controllability of the declarative paradigm and the expressivity of the imperative techniques.

Key words: theory construction, speech acts, distributed calculus