Un cadre théorique pour l'intégration des niveaux d'organisation dans les modèles : Applications à l'activité spatiale et à la simulation de grandes populations de bactéries

Martin Potier

UNIVERSITÉ PARIS-EST
ÉCOLE DOCTORALE MSTIC

THÈSE DE DOCTORAT

pour obtenir le titre de

Docteur de l'Université Paris-Est

Spécialité : Informatique

défendue par

Martin Potier

Un cadre théorique pour l'intégration des niveaux d'organisation dans les modèles. Applications à l'activité spatiale et à la simulation de grandes populations de bactéries.

Directeur de thèse : Olivier Michel
préparée au LACL

Soutenue le 6 juillet 2017 devant le jury composé de :

Rapporteurs:	Arnaud Banos	CNRS — Géographie-Cités
	Hanna Klaudel	Université d'Évry — IBISC
Examinateurs:	Hugues Berry	INRIA
	Nathalie Corson	Université du Havre — LMAH
	Jean-Louis Giavitto	CNRS — IRCAM
	Lisandru Muzy	CNRS — MS&N
Encadrant:	Antoine Spicher	Université Paris-Est Créteil — LACL
Directeur:	Olivier Michel	Université Paris-Est Créteil — LACL

Thèse préparée au LACL
(Laboratoire d'Algorithmique Complexité et Logique)

LACL
Faculté des Sciences et Technologie
61 avenue du Général De Gaulle
94 010 Créteil

Version du 8 juillet 2017 rédigée avec LaTeX

Remerciements

Ce travail de thèse n'aurait sans doute pas pu aboutir sans le support et parfois même l'aide précieuse de toutes les personnes que j'ai rencontré, avec qui j'ai partagé mon lieu de travail ou bien qui m'ont juste pointé dans la bonne direction.

Sans ordre particulier, je souhaite remercier les personnes suivantes :

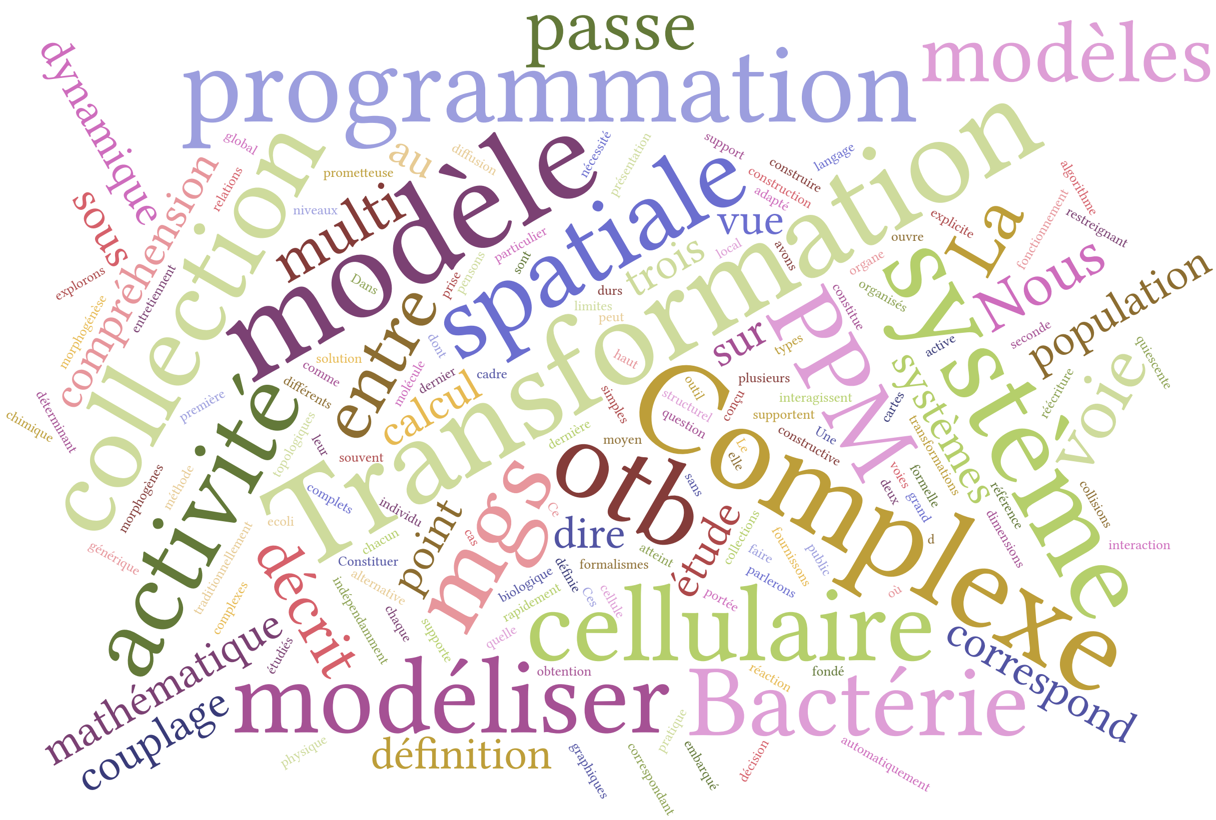
- λ Antoine et Olivier pour m'avoir encadré durant la thèse et m'avoir proposé un sujet original et motivant, ainsi que pour m'avoir permi de le mener à bien ;
- λ Elisabeth pour m'avoir gentiment rappelé qu'écrire une section de remerciements était une partie importante d'un manuscrit de thèse ;
- λ Daniele pour m'avoir permi de le remplacer pendant un de ses cours et qui s'est montré très enthousiaste devant mon usage de Sozi ;
- λ Flore, Clara, Nicolas, pour avoir été présent pour régler des détails administratifs parfois pesant et pour les longues discussions que cela a pu entraîner ;
- λ Toute l'équipe de l'IUT de Fontainebleau `\o/` (Régis, tu gères archlinux comme pas deux) ;
- λ Mes co-doctorants de l'époque : Louis, Thomas, Nghi, Quentin, Yoann, Victor, Rodica, et tous les autres…
- λ Olivier (Hermant, pas l'autre) et Olivia qui, les premiers, m'ont orienté vers le monde de la recherche et m'ont permi de passer au delà de mon angoisse d'ingénieur à trouver un travail, parce que, il le faut, mais de continuer dans la voie qui me plaît, jusqu'à aujourd'hui ;
- λ Olivia (la même que juste au dessus) pour son soutien inconditionnel et pour avoir réussi l'exploit de me déconstruire et me reconstruire sans que je tombe en panne, ce qui lui vaut le statut de vache en furie à vie ;
- λ Au commité de l'ombre, gérant des serveurs dans les salles obscures, communicant entre eux comme des hakerz, et parlant un langage que les simples mortels ne peuvent comprendre, David, Grégoire et…
- λ Sergiu (камарад !) parce que, soyons honnêtes, ce type n'est vraiment pas comme les autres ;
- λ Tous ceux qui sont venu voir le match, en particulier Thomas, Samuel, (Claire-Alice), que je n'avais pas revu depuis trop longtemps ;
- λ Mes parents et de manière plus générale ma famille pour m'avoir toujours soutenu dans mes choix tout en sachant me laisser la place nécessaire pour construire ma propre voie ;
- λ Les auteurs et contributeurs de tous les logiciels libres utilisés pendant l'écriture de ce mémoire[1] ;
- λ Toi qui lit ces remerciements, merci d'avoir lu jusqu'ici ;
- λ Et enfin, tous ceux qui ne sont pas cités ici mais que le méritent amplement, car à mon avis, il serait présomptueux de penser que ce qu'on n'est ne tient qu'à soit même.

Et là, il est grand temps que je m'arrête, car pour être honnête, il me reste bien une moitié de l'internet à remercier.

1. Pour vous faire une idée de la grande quantité des projets libres qui existent et de l'écosystème qu'ils forment, je vous conseille de jeter un œil à la clôture nix des dépendances de l'environnement de compilation pour ce document rédigé avec LaTeX, en voici quelques uns des plus importants : linux, xmonad, texlive et vim.

Table des matières

Table des figures



passe
programmation
modèles
dynamique
modèle
Transformation
Complexe
système
spatiale
collection
activité
otb
mgs
cellulaire
modéliser
Bactérie
correspond
PPM
La
Nous
population
voie
systèmes
compréhension
sous
multi
entre
calcul
au
vue
trois
sur
dire
point
étude
décrit
mathématique
couplage
définition

Chapitre 1

Introduction

Depuis l'avènement de la science informatique, les modèles sont de plus en plus utilisés pour décrire et comprendre le monde qui nous entoure. Automatisés et portés par les langages de programmation, ils sont de plus en plus imposants, au sens où ils incorporent de plus en plus d'information et visent à être de plus en plus complets. Barrant le chemin à une meilleure compréhension de notre environnement, la modélisation des *systèmes complexes* est le prochain col à franchir dans la compréhension du vivant (pour les sciences du vivant) et dans la compréhension des fondements des modèles (pour les sciences informatiques et mathématiques).

Il existe un riche vocabulaire entourant le monde des *systèmes complexes*: émergence, immergence, backward- et forward-causality, … Ces notions sont difficiles à appréhender car elles se fondent sur des cas particuliers (comme les exemples classiques du jeu de la vie [3], des boids [7] ou de la fourmilière [8]) et sur un certain laxisme formel dans la description des modèles, provenant de la confusion entre les objets que le modèle manipule et de notre propre identification de ces mêmes objets. Pour chacun des exemples donnés ci-dessus, nous pouvons nommer des propriétés qui ne font pas partie du modèle :

- Dans le jeu de la vie, un observateur parlera souvent de « glider », de « canons » à glider, de « beacon », de « pulsar », etc. ;
- Dans le modèles des boids, un observateur identifiera un comportement de groupe et cherchera à en identifier le « leader » ;
- Dans le modèle de la fourmilière, un observateur remarquera que les individus sélectionnent le « plus court chemin » pour rapporter la nourriture au nid, et qu'ils traitent les sources de nourriture dans l'ordre « du plus proche au plus lointain ».

Toutes les propriétés mises entre guillemets ci-dessus ne sont pas des propriétés des modèles présentés, il n'y a pas de « glider » dans le modèle du jeu de la vie, il n'y a pas de « leader » dans le modèle des boids ni de « plus court chemin » dans le modèle de la fourmilière. Ce sont des interprétations qui sont faites sur le déroulement d'une simulation, des propriétés d'un modèle *implicite* que l'observateur établit de lui-même. Rendre explicite ce modèle puis lier formellement ces propriétés émergentes au modèle d'origine est un enjeu de taille, car il

permettrait de relier tout ensemble de propriétés *locales* à des propriétés *globales*, c'est-à-dire des propriétés des individus à des propriétés du groupe formé de ces individus.

Un outil classique de la modélisation, les équations différentielles, nous donne l'occasion d'illustrer ce rapport local/global. Un système d'équations différentielles permet de décrire le comportement d'entités locales (au cours du temps, les unes par rapport aux autres, ...). Lorsque ce système d'équations différentielles possède une solution, alors la fonction solution est une *loi globale* du modèle établi au niveau local. Il existe un lien formel entre le comportement local de ces entités et leur comportement global. Il est possible d'étudier cette fonction *directement* en toute généralité. Toutefois, cette solution au système d'équations différentielles n'existe pas forcément. Dans ce cas, il est toujours possible d'obtenir le comportement global de ces entités par intégration numérique, c'est-à-dire en approchant la fonction solution *au cas par cas*. N'ayant pas accès à la fonction solution, il n'est cependant pas possible d'étudier directement le comportement global de cette population.

Nous prendrons partie pour le fait que construire un modèle constitué de plusieurs sous-modèles nous permet sous certaines conditions de répondre à la problématique. Cela doit nous permettre de concilier les comportements locaux et globaux de ce système. Nous appellerons cette technique la *modélisation multi-niveau*, que nous considérons comme une sous-branche de la modélisation multi-modèle.

La modélisation multi-modèle consiste à conjuguer différents modèles d'un système, où chacun des modèles décrit une partie fonctionnelle ou structurelle du système. Ces modèles, développés indépendamment, sont parfois rédigés dans des formalismes si distincts qu'il est difficile de les faire collaborer. Un exemple de discipline reposant sur cette problématique est la *mécatronique* [2], dont le sujet d'étude est le couplage entre systèmes mécaniques et électroniques. La modélisation multi-niveau s'intéresse aux modélisations multi-modèle ayant les propriétés suivantes :

- Le système peut se représenter comme un *empilement* de points de vue appelés *niveaux de description*. Même si ces niveaux ne sont pas agencés dans un ordre total, il existe cependant une relation caractérisable entre eux, comme par exemple une relation d'abstraction.
- Un niveau de description du système peut n'être que partiellement connu et c'est par leur couplage que la modélisation du système se trouve *renforcée* [1].

Sans entrer dans les détails, nous considérerons qu'un modèle est réductionniste s'il peut être découpé en sous-parties autonomes permettant, une fois assemblées de décrire exactement son comportement. La construction précédente permet de garder une vision réductionniste des parties du système, tout en indiquant que parfois il n'est pas possible de fournir un modèle réductionniste du système entier, notamment à cause d'un manque de connaissance sur le système.

Nos travaux sont issus de réflexions sur la modélisation multi-niveau, l'activité comme un niveau de description dans un langage de programmation spatiale et la simulation de la morphogénèse dans de grandes populations de bactéries, ayant pour origine les motivations du projet ANR Blanc Synbiotic.

1.1 Le projet Synbiotic

Dans cette section, nous présentons le projet ANR Synbiotic[1].

Le projet de recherche Synbiotic vise à développer des formalismes et des outils informatiques permettant de spécifier un comportement spatial global et de le compiler automatiquement à travers une tour de langages intermédiaires dans des processus locaux de régulation cellulaire (régulation génétique, métabolique, signalisation). La motivation finale est de permettre l'exploitation des propriétés collectives d'une population bactérienne pour créer des biosystèmes artificiels répondant à divers besoins dans le domaine de la santé, des nanotechnologies, de l'énergie et de la chimie.

Synbiotic s'inscrit dans le domaine des langages de programmation non conventionnels et de l'analyse de propriétés des systèmes dynamiques, à l'interface de l'informatique et de l'ingénierie biologique. Il s'appuie sur les avancées de la biologie synthétique, les progrès réalisés dans la modélisation et la simulation de processus biologiques complexes, et sur le développement de nouvelles approches de la programmation permettant de faire face à de nouvelles classes d'application caractérisées par l'émergence d'un comportement global dans une grande population d'entités irrégulièrement et dynamiquement connectées (le calcul amorphe et le calcul autonome).

La biologie synthétique est un domaine scientifique qui concerne la conception et la fabrication banalisée et standardisée de composants et de systèmes biologiques sans correspondants naturels. Elle est toujours en quête de principes de conception permettant une réalisation fiable et sécurisée à partir de composants biologiques réutilisables.

Dans ce contexte, l'objectif est de concevoir et développer les outils permettant de « compiler » (au sens de la compilation des langages de programmation) le comportement global d'une population, par exemple, de bactéries, en des processus cellulaires locaux à chaque entité. Notre motivation à très long terme est de permettre l'exploitation des propriétés collectives d'une population (bactérienne) pour créer des biosystèmes artificiels répondant à divers besoins dans le domaine de la santé, des nanotechnologies, de l'énergie et de la chimie verte. L'approche originale que nous proposons se fonde sur une tour de langages de programmation, dont l'étage le plus abstrait définit un modèle computationnel pour une population cellulaire et l'étage le plus primitif correspond à un agencement de séquences d'ADN. Chaque langage compile ses constructions propres vers le langage de la couche inférieure et ce, jusqu'au bioware (le « hardware biologique »). Cette approche, similaire à celle suivie avec succès dans le domaine de la synthèse d'architecture matérielle (chaîne de compilation vers le silicium), permet de combler le fossé existant entre la description d'un système au niveau d'abstraction pertinent pour l'application et la prise en compte de tous les détails de son implantation par des processus physico-chimiques. Elle permet de modulariser la conception d'un système, divisant les difficultés et isolant des niveaux d'abstraction qui peuvent évoluer indépendamment. Dans cette approche, un programme ne définit pas une fonction qui associe une sortie à une entrée mais spécifie un système dynamique (biologique) distribué qui essaie de maintenir des

1. La page de présentation du projet est disponible en ligne à l'adresse `http://synbiotic.spatial-computing.org`

invariants en dépit des perturbations et des changements de l'environnement.

Notre objectif est d'adresser la conception de grands systèmes biologiques par une approche langage, de la même manière que VHDL permet la conception de système de traitement de l'information à partir de portes et de blocs logiques élémentaires. Ce projet informatique repose sur trois hypothèses : l'apport des formalismes discrets, un processus de conception fondé sur la compilation d'une tour de langages et la prise en compte des aspects spatiaux.

Notre première hypothèse est que des modèles informatiques discrets sont adéquats pour décrire des biosystèmes et parfois plus pertinents que des approches mathématiques traditionnelles comme les équations différentielles. Cette hypothèse est corroborée par l'important développement actuel des formalismes informatiques dans le domaine de la biologie des systèmes. En particulier, ces formalismes sont plus à même de capturer de manière concise les aspects qualitatifs et quantitatifs des grands réseaux d'interactions biochimiques impliqués dans les processus biologiques. Ces formalismes permettent de découpler les abstractions utilisés dans le processus de conception (signal, gradient, mémoire, propagation, information de position…) des processus biochimiques utilisés pour leur implémentation, de la même manière qu'un bit abstrait de manière robuste une implantation électrique dans une électronique à base de silicium. Par ailleurs, ces formalismes permettent d'aborder la question de la validation : que peut-on garantir sur les comportements du système biologique artificiel, quel est le domaine de viabilité du système, quels sont les perturbations de l'environnement qui sont tolérables, quels est la résilience du système, peut-on garantir que certains états sont inatteignables, tracer les processus, tester les comportements attendus, etc.

Notre seconde hypothèse est qu'à partir de ces formalismes, la compilation est une approche descendante plus souple que l'assemblage direct de composants biologiques prédéfinis. Le processus de compilation permet d'instancier des composants élémentaires génériques dans un organisme particulier, permet de prendre en compte des contraintes d'assemblage (comme l'évitement de cross-talk entre circuits de régulation) ainsi que la simplification et l'optimisation des circuits obtenus par assemblage. Cette approche de haut-niveau correspond à la synthèse d'un système à partir de ses spécifications et repose sur la possibilité de dériver le comportement des parties à partir du comportement d'un tout. Ce problème est notoirement plus simple que celui de l'inférence de propriétés globales à partir de comportements locaux (émergence) et a montré toute son utilité dans le domaine de la synthèse d'architecture, mais aussi dans le cadre de la programmation spatiale et de la programmation amorphe. Un des enjeux du projet est de montrer que ces techniques peuvent être appliquées avec succès à la synthèse de biosystèmes.

Enfin, notre dernière hypothèse est que, même si pour l'instant la biologie synthétique se focalise sur la « programmation d'une seule bactérie », le développement de biosystèmes un tant soit peu complexe reposera sur le fonctionnement intégré de colonies bactériennes et donc sur la prise en compte d'interactions spatiales au sein d'une population de cellules différenciées. Il est en effet douteux qu'une cellule puisse supporter un nombre arbitraire de comportements artificiellement imposés. Au contraire, l'exemple des processus biologiques naturels montrent toute l'importance de l'organisation spatiale et de la compartimentalisation (membrane, vésicule, cargo, compartiment, cellule, biofilm, tissus, organe, etc.) permettant la spécialisation et

le fonctionnement intégré au sein d'un système compris comme une écologie. Par ailleurs, la maîtrise des interactions spatiales ouvre la voie à une ingénierie du développement (« développement » au sens biologique du terme), ce qui permet de rêver à des applications qui vont bien au-delà de la conception de la cellule comme « usine chimique ».

Découpage des tâches. Le projet Synbiotic est découpé en six *Work Packages* (WP), dont les WP 1, 2, 3 et 4 forment une tour de langages reliant le hardware/wetware (WP4) au langage de spécification global (WP1). Voici une description de ces quatre tâches :

WP1 : Ingénierie des formes ce WP est en charge de concevoir, de développer et d'implémenter des exemples d'applications qui pourront être utilisées par les autres WP.

WP2 : Programmation spatiale déclarative ce WP est en charge de l'élaboration d'un langage de programmation spatiale L1, dont la fonction est de décrire les exemples du WP1 (définis en terme de primitives spatiales sur des populations) vers un langage de plus bas niveau servant d'entrée au WP3.

WP3 : Programmation orientée entité ce WP est en charge de l'élaboration d'un nouveau langage L2 prenant en entrée les concepts utilisés en sortie de L1 et les instanciant dans les individus (des bactéries).

WP4 : Programmation orientée réseau de régulation l'enjeu de ce WP est de permettre la traduction de la sortie de L2 vers un médium biologique, afin d'obtenir une implémentation finale *in vivo*.

Les WP 5 et 6 sont transversaux et se chargent de « calculabilité et complexité » et de « Biosûreté et bio-sécurité » respectivement.

1.2 Objectifs et organisation de cette thèse

Dans cette section nous présentons brièvement les objectifs de ce manuscrit et les chapitres qui le composent afin de donner au lecteur une vue d'ensemble des résultats obtenus.

Comme nous l'avons vu, nos travaux sont ancrés dans le contexte du projet Synbiotic. Nous avons contribué aux WP2 (chapitre 3) et au WP3 (chapitre 4). Nous avons également dépassé le cadre du projet en nous intéressant de plus près à la nature des modèles (mathématiques) et à la définition de la modélisation multi-niveau.

Chapitre 2 : Une approche formelle générale des niveaux de modélisation Dans ce chapitre nous proposons une première voie de réponse à notre problématique en nous attaquant au problème de la définition à la fois formelle et générique des niveaux de modélisation. Nous introduisons une définition claire de ce qu'est un *modèle formel* qui nous permet de présenter explicitement notre point de vue. Une fois ce socle à disposition, nous présentons

différentes classes de modèles avec leurs exemples dans notre formalisme mettant en évidence la possibilité d'identifier ces classes de modèles suivant plusieurs critères reposant sur la structure mathématique associée à leur comportement. Nous proposons enfin une construction de niveaux d'abstraction, reposant sur quelques principes issus de la théorie des catégories, introduisant la validation, l'abstraction et la composition des modèles dans ce cadre. Nous instancions finalement sur un exemple les définitions précédentes mettant en lumière les relations existantes entre quatre modèles d'un système proie-prédateur.

Chapitre 3 : L'activité spatiale comme outil de modélisation Dans ce chapitre nous présentons le langage L1 du WP2 en posant un cadre pratique et théorique porté par le projet MGS. Ce dernier développe un langage de programmation dédié à la modélisation et la simulation de systèmes dynamiques à structure dynamique. L'état du système est décrit à travers une structure de données — la collection topologique — qui met l'accent sur les relations topologiques entre les éléments du système. L'évolution du système est spécifiée au moyen d'une structure de contrôle — la transformation — qui décrit, sous forme de règles *locales*, les interactions entre les éléments du système. Nous présentons l'activité spatiale dans le contexte de MGS, qui nous indique la répartition dans une collection topologique de la sous-collection active et nous implémentons un algorithme nous permettant de mettre à jour cette collection sans connaissance de la sous-collection quiescente. Au delà de l'optimisation en temps de calcul qu'il apporte à certaines simulation, cet outil nous permet de capturer une *propriété d'ordre supérieur* du modèle et offre une voie pour la description, dans MGS, de modèle à plusieurs niveaux de description.

Chapitre 4 : Un exemple concret de simulation multi-niveaux Dans ce chapitre nous ouvrons une troisième voie de réponse à notre problématique, tournée vers la pratique, où nous présentons la conception d'un simulateur de populations de bactéries *Escherichia Coli*, nommé OTB, correspondant au langage L2 du WP3 de Synbiotic. En effet, l'évolution de ces populations de bactéries montre un fort potentiel pour la modélisation et la simulation multi-niveau : une population suffisamment importante mène à des propriétés définies uniquement à l'échelle de cette population, comme l'émergence et le maintient de formes particulières — la morphogénèse. Le but de ce simulateur n'est pas, comme dans les chapitres précédents, de fournir une méthode générique pour exprimer ces propriétés mais plutôt de partir de la bactérie définie individuellement et de tester, donc de mettre en évidence, le lien existant entre les propriétés de l'individu (le réseau de régulation génétique) et les propriétés de la population (les formes émergentes). Pour relever les défis posés par la simulation d'un grand nombre d'individus, nous tirons partie du calcul parallèle sur des cartes graphiques grand public et, dans ce cadre, nous avons développé plusieurs algorithmes originaux pour le calcul parallèle dont le principal est PPM, inspiré des automates cellulaires et du voisinage de Margolus.

1.3 Contributions

Voici une liste succincte des contributions scientifiques apportées par cette thèse.

Théorie et formalisation

- établissement d'un cadre formel pour l'expression unifiée des niveaux de modélisation, indépendamment de leur formalisme d'origine ;
- définition de l'activité spatiale ;
- élaboration d'un algorithme pour l'identification et le maintient d'une zone spatialement active dans MGS;
- définition d'un algorithme générique (PPM) pour le calcul parallèle.

Développement

- développement d'un simulateur pour mettre en évidence l'émergence et le maintient de formes dans une population de bactéries *E. Coli*.

Communications orales

- présentation de « Computing Activity in Space » au workshop SCW13 de la conférence AAMAS 2013 à St Paul (Minnesota, USA), du 6 au 10 mai 2013 ;
- présentation de « Topological Computation of Activity Regions » au workshop Work In Progress de la conférence SIGSIM PADS 2013 à Montréal (Québec, Canada), du 19 au 22 mai 2013 ;
- présentation au séminaire du LACL, intitulé « Implementing Multiple Levels of Organization in a Spatial Programming Language » le 10 février 2014, à Créteil ;
- participation au workshop SCW14 attaché à la conférence AAMAS qui s'est tenu à Paris du 5 au 9 mai 2014 ;
- participation au workshop 228 de l'INSERM intitulé : « Experimental approaches in mechanotransduction : from molecules to issues and pathology », du 21 au 23 Mai à Bordeaux, France ;
- présentation d'un tutoriel MGS à la conférence ICCSA 2014, le 24 Juin au Havre, France ;
- présentation de « Managing the Interoperability Between Models of a Complex System » au workshop 6 satellite de la conférence ICCSA 2014, du 23 au 26 Juin au Havre, France ;
- présentation de « Spatial Computing for Integrative Modeling » au ComBio à Turku, Finlande, le 29 octobre 2015.

Communications écrites

Les travaux présentés dans ce manuscrit ont donné lieu à plusieurs publications dans des conférences internationales et nationale [5, 6, 4]. Deux publications sont en cours de préparation en vue de rendre public les travaux du chapitre 2 et du chapitre 4.

Références

[1] Arnaud Banos, Nathalie Corson, Benoit Gaudou, Vincent Laperrière et Sébastien Rey Coyrehourcq. "Coupling micro and macro dynamics models on networks : Application to disease spread". In : *International Workshop on Multi-Agent Systems and Agent-Based Simulation*. Springer, 2015, p. 19–33.

[2] Robert H Bishop et al. *The mechatronics handbook*. T. 1. CRC press Boca Raton, 2002.

[3] Martin Gardner. "Mathematical games : The fantastic combinations of John Conway's new solitaire game "life"". In : *Scientific American* 223.4 (1970), p. 120–123.

[4] Jonathan Pascalie, Martin Potier, Taras Kowaliw, Jean-Louis Giavitto, Olivier Michel, Antoine Spicher et René Doursat. "Developmental Design of Synthetic Bacterial Architectures by Morphogenetic Engineering". In : *ACS Synthetic Biology* 5.8 (2016), p. 842–861.

[5] Martin Potier, Antoine Spicher et Olivier Michel. "Computing Activity in Space". In : *Spatial Computing Workshop 2013*. Saint-Paul, Minesotta, USA, 2013.

[6] Martin Potier, Antoine Spicher et Olivier Michel. "Topological computation of activity regions". In : *SIGSIM Principles of Advanced Discrete Simulation*. Montreal, Québec, Canada, mai 2013, p. 337–342.

[7] Craig W Reynolds. "Flocks, herds and schools : A distributed behavioral model". In : *ACM SIGGRAPH computer graphics* 21.4 (1987), p. 25–34.

[8] Uri Wilensky. "Netlogo ants model". In : *Netlogo model library, Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL. URL http ://ccl. northwestern. edu/netlogo/models/Ants* (1997).

Autres contributions

- Encadrement à 50% du stage de Master 1 de Romain Carriquiry Borchiari intitulé « Modélisation et simulation d'une population de bactéries en OpenCL » duquel la toute première version de OTB est issu.

Chapitre 2

Modèles et modélisation multi-niveau

Sommaire

2.1 Introduction

Dans ce chapitre, nous nous intéressons au développement d'un cadre formel pour la spécification de *modélisations multi-niveau*. Ce type de modélisation est souvent rencontrée dans les approches intégratives des *systèmes complexes*, dans de nombreuses disciplines :

- pour le couplage de modèles pour la description d'une cellule entière (en biologie) ou la description d'un système de villes (en géographie) ;
- pour le raffinement de modèles par la transformation de modèles (en ingénierie des modèles) ou le raffinement de données et de processus (en biologie) ;
- pour la complexification de modèles dans les Cellular Non-linear Networks (CNN, en électronique), ou dans les Memory Evolutive Neural Systems (MENS, en mathématiques), ces deux exemples étant présentés dans la section suivante.

Notre objectif n'est évidemment pas de répondre à ces questions mais d'apporter un cadre de description commun, adapté et opérationnel (au sens où l'effectivité d'un couplage pourrait être démontrée) pour les formaliser.

Ce chapitre est consacré au rapport entre modèles et modélisation multi-niveau. Il nous paraît important d'aborder la question de la modélisation multi-niveau d'abord dans sa généralité, plutôt que le cas particulier que constitue le champ d'une discipline, tel que la biologie, la physique, l'écologie, ou un objet d'étude comme la dynamique des populations, la cellule, les feux de forêts, etc. Même si ces cas particuliers présentent un intérêt pratique et permettent de valider une démarche ou répondre à une question immédiate, ils pourraient se trouver limités par les habitudes et les usages liés à la discipline dont ils sont issue.

L'objet de notre étude est donc de décrire l'étape franchie lors d'un changement de niveau de description. Nous partirons donc d'une définition très générale des modèles qui sera raffinée pour correspondre à des modèles existants et mettre en lumière les différences et similarités entre ces formalismes. Nous nous intéressons exclusivement aux modèles formels, c'est-à-dire des modèles donnés sous la forme d'une définition utilisant le langage des mathématiques et pouvant potentiellement être calculés par un programme informatique. Dans la suite de ce document, sauf exception, nous emploierons le terme « modèle » pour désigner un modèle formel.

Nous remarquons tout d'abord que déterminer des relations entre plusieurs modèles est possible d'autant plus aisément que ces modèles évoluent dans la même « classe », celle des modèles d'un même système, et qu'ils sont rédigés dans un formalisme commun. Par exemple, nous avons agencé les différents modèles de la dynamique temporelle du potentiel de la membrane neuronale en terme de « finesse » de description dans la Fig. 2.1; ces modèles sont tous décrits à l'aide d'équations différentielles ordinaires, déterministes, et décrivent des dynamiques dans le domaine continu.

Plusieurs champs de recherche s'intéressent aux relations entre modèles, chacun ayant sa portée et ses limites. Nous donnons, dans la section suivante, une courte présentation de quelques uns de ces domaines, ainsi qu'un bref aperçu des résultats qu'ils apportent.

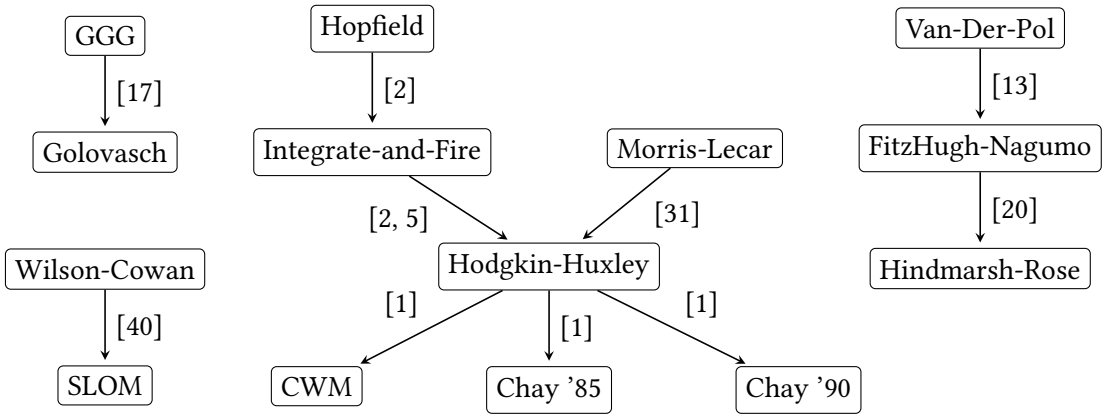


FIG. 2.1: Revue de quelques modèles de neurones. Nous avons fait pointer informellement les flèches vers les modèles ayant un « niveau de détail » plus grand.

2.2 Modélisation multi-niveau

La *modélisation multi-niveau* est un terme que nous avons adopté pour rendre compte des techniques de modélisation couplant plusieurs modèles pour décrire un système. Dans ce contexte, un modèle propose d'observer le système étudié à travers le prisme d'un point de vue particulier, appelé *niveau de description*. La notion de *niveau* met ici l'accent sur le fait que chaque modèle peut ne s'intéresser qu'à une sous-partie du système (peu importe ce que l'on entend par sous-partie ici) et propose donc une description incomplète du système. C'est par le couplage entre les modèles que ces *descriptions* complémentaires se complètent mutuellement, pour finalement obtenir une description plus pertinente du système dans son ensemble. Ainsi, la modélisation multi-niveau met en avant une méthodologie générique pour l'élaboration d'un modèle par composition (d'autres modèles). Dans cette section, nous nous intéressons à différents exemples de composition de modèles pour en analyser les difficultés. Nous en concluons que la notion d'*identité* est au cœur de la modélisation multi-niveau.

2.2.1 Couplage de modèle

Dans cette section nous nous intéressons à deux exemples types de couplages de modèles. Le premier exemple concerne la *modélisation intégrative* du phénotype d'une cellule biologique entière à partir des modélisations des différents mécanismes moléculaires qui la composent. Le second exemple s'intéresse à la *morphogénèse* d'une communauté de villes à partir de modélisations orientées quartier. Dans ces deux exemples, des modèles élémentaires sont utilisés pour en conjuguer les points de vue et obtenir une description plus complète du système. Dans les deux cas également, un changement d'échelle s'opère, entre les considérations micro(scopiques) des modèles élémentaires, et le niveau macro(scopique) du système étudié.

Modélisation intégrative d'une cellule entière

En modélisation, l'approche *intégrative* consiste à construire des modèles en cherchant à expliquer *simultanément* plusieurs phénomènes, permettant de révéler des dynamiques n'apparaissant pas dans des approches indépendantes. Ces modèles sont construits par l'accumulation de modèles préexistants jugés valides et par l'établissement d'un réseau de communication entre ces modèles, voire la conception d'adaptateurs permettant de « traduire » la sortie d'un modèle en l'entrée d'un autre. Cette approche est particulièrement développée dans le monde de la recherche sur le vivant [18], et permet des échanges prolifiques entre les nombreuses disciplines de la biologie (généticiens, physiologistes, comportementalistes, paléontologistes, écologistes, etc.) et donc entre les nombreuses perspectives d'un même système qu'il est possible de récolter.

De nombreux projets utilisent cette approche pour comprendre le fonctionnement d'un organisme biologique complet, comme par exemple une cellule. À titre d'exemple, un modèle computationel d'une cellule entière de *Mycoplasma genitalium* est proposé dans [21]. Ce modèle donne l'expression complète du phénotype de la cellule à partir de son génome en formulant des hypothèses sur de nouveaux mécanismes biologiques. L'approche de ce projet passe par le concept de *modularité biologique* qui permet de déterminer une liste d'observables du système dont les valeurs correspondent à des états de la cellule. Plusieurs modèles permettent d'expliquer séparément comment certaines observables sont liées entre elles. Une fois ces modules établis, les modèles sont assemblés en suivant les relations entre les observables, puis ajustés pour respecter les contraintes physiques et biologiques du système dans son ensemble.

Les solutions élaborées durant ce type de projet sont pour le moment *ad hoc* et la recherche de méthodes génériques pour rendre cette approche utilisable pour la description d'autres systèmes biologiques est en cours. Le projet précédent a par exemple mis en lumière, dans une seconde publication [27], une liste de sept limites à dépasser pour généraliser leur approche intégrative à d'autres organismes. Parmi ces points, les suivants, attireront plus particulièrement notre attention car ils évoquent les limites des outils pour la modélisation intégrative :

Agrégation des données L'acquisition des données disponibles dans la littérature a été compliquée par l'absence d'une source *unifiée* permettant de compiler l'ensemble des connaissances sur un sujet en particulier, dans ce cas sur l'organisme étudié. La modélisation d'une cellule entière bénéficierait grandement d'outils permettant d'agréger et de *normaliser* toutes les données nécessaires à la calibration des modèles ;

Construction et intégration de modèles Un facteur limitant a été la disparité des types de modèles décrivant à différentes échelles d'espace et de temps l'organisme. Unifier la modélisation intégrative passerait donc par une *généralisation* du concept de modèle à travers un cadre mathématique commun à l'ensemble de la modélisation. Comprendre et *analyser* les relations entre modèles dépend de l'existence ce cadre commun ;

Calcul accéléré La simulation numérique est un outil clef de la réussite de ce projet. Chaque simulation de *Mycoplasma genitalium* a été très consommatrice en temps de calcul, ce

qui est un facteur limitant pour la réussite des projets futurs. Élaborer des modèles d'organismes plus complexes dépendra donc fortement de la capacité des simulateurs à explorer une large portion de l'espace des paramètres dans un temps plus restreint. Le calcul général sur carte graphique, via un framework tel que CUDA ou OpenCL, est une piste à explorer pour améliorer l'efficacité des simulateurs ;

Modélisation multi-échelle de la croissance des villes

Les modèles à base d'agents sont souvent utilisés en géographie pour rendre compte des dynamiques observables sur une carte : le système est étudié à travers des effets de masse observés dans les populations d'agents.

Le projet SimPop [37][1], illustre cette approche dès 1996 avec une collaboration entre géographes et informaticiens aboutissant à Simpop1, le premier modèle géographique à base d'agents s'intéressant à la construction de réseaux de villes et à la transition urbaine. En 2006, l'évolution à la fois de la puissance de calcul des machines et de l'expressivité des langages de programmation, permet le développement d'une version améliorée, Simpop2 [4]. Ce nouveau simulateur prend en compte de nombreuses *fonctions urbaines* de différents types, résidentielles, politiques, économiques, culturelles, liées au transport et aux communications, sociales, etc. Le support de modélisation est un graphe dynamique dont les nœuds sont les villes et dont l'évolution repose sur 4 grandes étapes, à savoir la constitution d'un réseau d'échanges, d'un marché d'échanges, d'un équilibrage de ces échanges et l'évaluation des changements.

Le projet SimPop propose également d'autres modèles, tel que SimpopNano qui prend le contre-pied des plateformes précédentes en se concentrant non seulement sur les relations inter-urbaines mais intra-urbaines : les villes qui étaient les agents deviennent les objets d'étude décrits comme des agrégations de quartiers. SimpopNano s'intéresse en effet à l'évolution d'une ville, vue comme un graphe statique dont les nœuds en sont les quartiers et où la dynamique est donnée à travers neuf règles d'évolution locale, chacune correspondant à une fonction urbaine.

Le couplage entre Simpop2 et SimpopNano est un des enjeux du projet SimPop. Simpop3, proposé dans [25], est un simulateur de la dynamique des villes répondant à cette problématique. Le couplage entre ces deux modèles est *vertical* : il assemble trois échelons d'une échelle d'espace géographique, à savoir quartier, ville et système de villes, et exprime l'interdépendance entre les deux modèles :

- le système de villes contraint la dynamique au niveau ville du système ;
- rétroactivement, l'activité intra-urbaine influence les entrées du modèle inter-urbain.

La ville est donc le point d'articulation entre les deux modèles et constitue un niveau de représentation intermédiaire, *mésoscopique*. Le point essentiel consiste à *identifier* les agrégats de quartiers de SimpopNano et les agents-villes de Simpop2. Nous reviendrons en fin de section sur cette notion d'identité.

1. http://www.simpop.parisgeo.cnrs.fr

En terme de dynamique, dans SimpopNano, les villes sont construites à partir des quartiers et sont mises à jour à chaque pas de la simulation. Dans Simpop2, les agents-villes sont à l'origine de la dynamique d'un système de villes. Le couplage se fait simplement car la concurrence sur la ressource agent-ville n'est pas prise en compte, ce qui autorise une indépendance entre les modèles. Ainsi, au cours d'une simulation de Simpop3, les deux sous-modèles sont juste appelés alternativement.

2.2.2 Transformation de modèles

Dans la section précédente, la modélisation multi-niveau était synonyme de modélisation multi-échelle, avec un rapport entre un niveau macro et un niveau micro. Dans cette section, les modèles entretiennent un rapport différent : les modèles décrivent les mêmes objets, mais le niveau de granularité entre ces descriptions varie. L'opération fondamentale de ces approches est le passage d'un niveau de granularité à un autre niveau de granularité par une *transformation de modèles*. Les deux paragraphes suivants illustrent cette approche.

Ingénierie des modèles

Dans le domaine de l'informatique et de la conception logicielle, l'*ingénierie des modèles* (*model-driven engineering* en anglais) est une méthodologie de développement se concentrant sur la partie métier du logiciel, c'est-à-dire ce qui dans un logiciel fait abstraction de l'implémentation. L'objectif de cette discipline est l'automatisation de la conception logicielle à partir d'une représentation abstraite pouvant ensuite être *transformée* en des représentations de plus en plus concrètes jusqu'au logiciel final. L'ingénierie des modèles part du principe qu'un domaine d'application est plus fortement dépendant d'une représentation abstraite des connaissances et des activités liées à ce domaine, que des concepts algorithmiques et des techniques nécessaires à la mise en œuvre d'une implémentation. Par exemple, pour un problème donné, il existe plusieurs logiciels répondant à ce problème, implémentés de façons différentes : il existe bien *quelque chose* d'unique et de plus abstrait qu'un logiciel qui est propre à la fois au problème et à la solution.

Considérant les modèles comme des objets de première classe, l'ingénierie des modèles cherche à développer, maintenir et faire évoluer un logiciel au moyen de *transformations* : des fonctions d'ordre supérieur prenant des modèles en argument. Ces fonctions sont implémentables et exécutables afin que les transformations de modèles soient des procédures automatiques. Ce type de procédure est fréquent en informatique. La compilation est un exemple historique de transformation de modèles, où l'on entend ici les langages de programmation comme des « modèles de code source » : compiler consiste à définir des transformations de code source à code source. Le langage diagrammatique d'*Unified Modeling Language* (UML [33]), qui permet d'exprimer des modèles de conception, est un candidat classique à la transformation de modèles. Dans ce contexte, les transformations se présentent comme des règles de réécriture de graphe (voir [42] pour une étude sur les transformations de graphes appliquées aux transformations de modèles).

Pour notre approche multi-niveau, l'ingénierie des modèles nous apporte deux éléments importants. Le premier élément provient de la définition même de modèle, qui y est vu comme la description d'un système spécifiée dans un *langage bien défini* [22]. Le langage de description est si important qu'il est souvent lui-même décrit comme un modèle ; on parlera alors de *méta-modèle*. On rapprochera ce langage de description de la notion de *formalisme*. Le second élément provient des transformations de modèles dont une taxonomie est donnée dans [30]. On retiendra notamment les catégories suivantes :

Transformation endogène *vs.* transformation exogène : une transformation est exogène lorsque le langage de description du modèle cible est différent du modèle de description du langage source.

Transformation horizontale *vs.* transformation verticale : nous retrouvons ici la notion de granularité mentionnée plus haut ; lors d'une transformation verticale le niveau d'abstraction entre le modèle source et le modèle cible est modifié. La notion d'abstraction jouera un rôle important dans notre proposition.

Raffinement de modèles

Bien que la description précédente de l'ingénierie des modèles soit orientée informatique, les mêmes considérations se retrouvent dans d'autres disciplines. Par exemple, une méthodologie classique de la modélisation à base d'équations différentielles de systèmes (bio)chimiques consiste à élaborer un modèle simple à partir d'un ensemble initial d'espèces et de réactions, puis de l'enrichir soit avec de nouvelles espèces (on parlera de *raffinement des données*), soit avec de nouvelles réactions (on parlera de *raffinement des processus*). À chaque itération, un nouveau modèle est établi puis ajusté pour prendre en compte les observations du système. Le processus de raffinement à l'œuvre génère ainsi une suite de modèles ordonnés du moins détaillé au plus détaillé. La preuve de raffinement entre deux modèles peut se faire de façon classique ou bien via un outil spécialisé comme le langage Z [41].

Une des difficultés du raffinement de modèles se situe dans l'ajustement des modèles construits. Une méthode systématique a été développée pour dériver à partir d'un modèle établi et préalablement ajusté, un modèle raffiné préservant l'ajustement. Cette méthode ne se limite pas aux systèmes d'équations différentielles mais couvre également le champ des réseaux de Petri, des systèmes réactifs et des langages à commandes gardées [19].

2.2.3 Complexification

La modélisation des systèmes complexes constitue l'un des enjeux principaux de l'approche multi-niveau que nous souhaitons développer. L'Institut des Systèmes Complexes (ISC), une structure française pluridisciplinaire de recherche spécialisée sur le sujet, en donne, dans sa proposition de DIM « Approches interdisciplinaires des systèmes complexes »[2], la définition

2. Document disponible en ligne à l'adresse `https://iscpif.fr/wp-content/uploads/2016/09/PreProjetDim2017.pdf`

suivante :

> Les systèmes complexes sont des systèmes présentant un grand nombre d'entités différenciées, dont la multitude des interactions locales conduisent à l'émergence de propriétés globales difficilement prédictible par la seule connaissance des propriétés de ces entités. Le niveau local fait ainsi émerger des formes organisées au niveau global, lequel influence en retour le niveau local (immergence). Interactions locales et structures globales peuvent ainsi se conjuguer dans la description des dynamiques des systèmes complexes.

Cette définition n'est pas sans rappeler les exemples de couplage de modèles que nous avons vu plus haut, qui entrent en effet dans le cadre des systèmes complexes.

Dans cette section, notre intérêt ne se portera pas sur la modélisation des systèmes complexes dans toute sa généralité, mais plus particulièrement sur le concept de *complexification*, c'est-à-dire sur la façon de capturer dans ces modèles les « phénomènes émergents » et sur comment les réifier au niveau local afin qu'ils deviennent des acteurs de leur propre dynamique. Nous présentons pour cela la complexité à travers 3 points de vue théoriques. Le premier exemple montre la difficulté d'appréhender la complexité à cause d'un manque de vocabulaire au niveau bas de description. Le deuxième exemple attribue la complexité à une cause énergétique par le phénomène d'*activité locale*. Finalement, dans un dernier exemple, le processus de complexification est analysé d'un point de vue structurel par l'établissement d'un modèle formel connexionniste.

Raffinement et émergence

En 2005, F. Polack et S. Stepney [35] présentent une approche de la complexité en rapport avec le raffinement de modèles, que nous avons présenté dans la section 2.2.2. L'idée centrale de leur approche est qu'une propriété d'un modèle est émergente s'il n'est pas possible de l'obtenir par un raffinement naïf, c'est à dire par une unique relation de raffinement. Dans cet article, les auteurs prennent pour exemple le raffinement d'une propriété considéré comme émergente, volontairement simple et observable dans le fonctionnement d'un automate cellulaire en deux dimensions, pour mettre en évidence les difficultés que posent le raffinement d'un comportement émergent. Nous présentons la démarche de raffinement simplifiée pour lier une propriété donnée au composants bas niveau d'un système formel, dans notre cas ce système est un automate cellulaire en deux dimensions muni des lois d'évolution du « jeu de la vie ». Elle repose habituellement sur les 3 étapes suivantes :

1. Exprimer la propriété dans un formalisme ;
2. Exprimer l'automate cellulaire dans un formalisme (potentiellement différent du précédent) ;
3. Exprimer la relation entre les deux formalismes, prouvant ainsi que la propriété est exprimable dans le contexte des automates cellulaires.

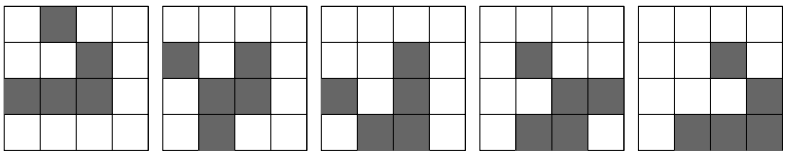


FIG. 2.2: Les cinq configurations possibles d'un glider contenues dans une fenêtre de 4×4 cellules.

Posséder un *glider* [14] est une propriété connue pour être émergente, aussi il serait intéressant de déterminer comment la raffiner. Elle est initialement spécifiée informellement par :

> Le système doit présenter une région, traversée par un motif répété, appelé « glider ». Lorsque deux gliders se rencontrent, il en résulte un unique glider qui se déplace dans la direction d'un des deux gliders incidents.

Cette propriété est précise, cependant elle reste non déterministe. Tentons de la raffiner : nous prendrons naturellement les automates cellulaires comme support cible, car notre objectif est d'identifier les composants bas niveau responsables de ce comportement. Un glider se définit par la fenêtre qu'il traverse et un vecteur de déplacement. Il est aussi nécessaire de décrire la collision de deux gliders et le résultat de cette collision. Pour spécifier un automate cellulaire, nous formalisons d'abord sa cellule, son état, sa fonction de transition et son voisinage. Ensuite, nous choisissons une représentation formelle d'une collection de ses cellules et de l'application synchrone des fonctions de transitions, habituellement appelées fonctions de transition locales (pour une définition complète, voir page 118).

Pour poursuivre la procédure, une sous-division discrète de la fenêtre traversée par le glider est introduite, par exemple sous la forme d'une grille 4×4. Le premier problème qui se pose est que le raffinement de la spécification informelle est plus précise que la spécification originale, ce qui est voulu, mais également plus précise que la représentation ciblée. Dans le cas d'un raffinement classique, au contraire, le raffinement d'une propriété de haut niveau amène à un niveau d'abstraction intermédiaire, c'est-à-dire entre la spécification d'origine et la spécification cible. Lors de la mise en œuvre du raffinement des *données* de la spécification d'un glider, à savoir « fenêtre » et « glider », il apparaît qu'il n'est pas possible de décrire un glider par un unique état spatial : un glider est en fait constitué d'une succession des cinq configurations présentées dans la FIG. 2.2. De plus, la configuration initiale de l'automate est importante : elle doit exactement correspondre à une de ces cinq configurations possibles. Ensuite, lors de la spécification des *opérations* du glider, à savoir « déplacement », « collision » et « résolution », il est déjà difficile d'exprimer ces opérations sur une grille. Pour le déplacement par exemple, la fenêtre pourrait se déplacer sur une grille extérieure, dans une direction une fois tous les quatre pas des états du glider *dans la fenêtre*, ne pas bouger ou bien faire un pas dans la direction orthogonale. Les opérations de collision et de résolution sont tout aussi difficiles à décrire.

Nous constatons qu'il est difficile d'exprimer la relation de raffinement de la propriété « posséder un glider » dans des termes propres aux automates cellulaires en deux dimension en

une étape. De plus, cette relation ne sera manifestement pas unique. Les difficultés à rendre concrète une propriété témoigne de son caractère émergent : elle est observable mais n'est pas concrètement implémentée dans le système. L'idée centrale de [35] est la suivante : il existe une incompatibilité entre les langages de description qui (1) rend la spécification concrète des gliders difficiles et (2) empêche toute élaboration directe d'une preuve de raffinement avec le vocabulaire des automates. Cette caractéristique est propre aux systèmes possédant des propriétés émergentes.

Complexification dans les CNN

L. Chua a introduit en 1998 le concept d'*activité locale* (*local activity*) [28]. Ce concept est décrit comme le maillon manquant permettant d'expliquer l'apparition de motifs complexes dans un médium homogène. L'activité locale, définie initialement dans le cadre des circuits électroniques [7], caractérise la propriété d'auto-organisation, c'est à dire la capacité d'un système à posséder une dynamique qui lui est propre, à travers une construction mathématique explicite. La définition a été par la suite généralisée à la classe plus larges des systèmes de réaction-diffusion non linéaires utilisés en physique, en chimie, en biologie et dans le cadre de la recherche sur le cerveau. L'activité locale permet d'expliquer la brisure de symétrie dans un médium homogène. Dans les paragraphes suivants, nous présentons de manière succincte quelques définitions proposées par les auteurs pour l'activité locale, la complexité et le seuil du chaos (*edge of chaos*).

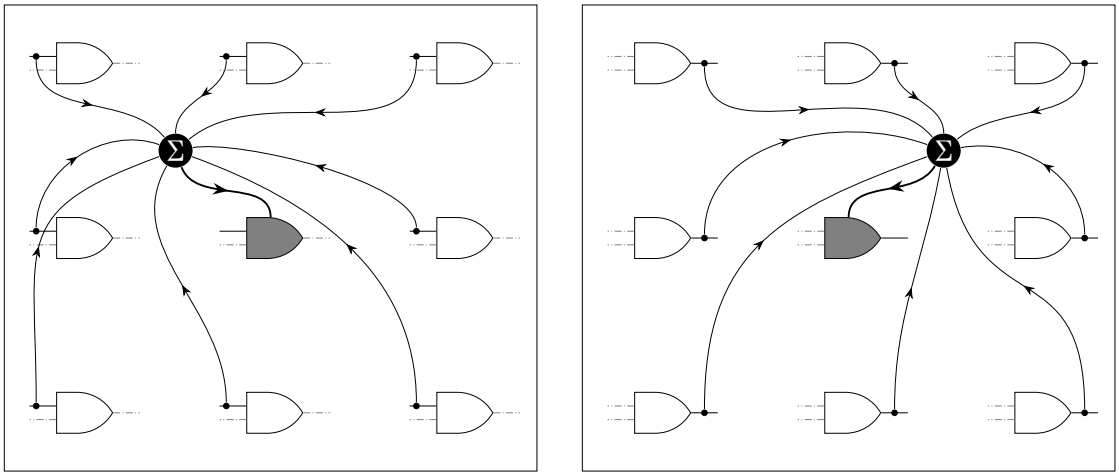


Fig. 2.3: Connexion des synapses pour le signal de contrôle (à gauche) et pour le signal de rétroaction (à droite) pour la cellule au centre, entourée de huit voisines (cette illustration est inspirée de [7])

L'activité locale est à l'origine définie dans le cadre des réseaux non linéaires de cellules (CNN pour *Cellular Non-linear Networks*). Un CNN est un modèle formel constitué d'une grille orthogonale en trois dimensions sur les nœuds de laquelle est disposée une collection de systèmes dynamiques continus et non linéaires. Chaque cellule isolée d'un CNN est décrite par

quatre paramètres : son entrée, son état, son seuil et sa sortie. Les cellules du CNN sont alors couplées à leurs voisines inclues dans une sphère d'influence prédéfinie avec un certain rayon. Pour une cellule donnée, son entrée est la somme pondérée des sorties de ses voisines (signal de rétroaction) *et* la somme pondérée des entrées de ses voisines (signal de contrôle). Une représentation graphique est donnée dans la FIG. 2.3.

L'état d'une cellule est définie par un système d'équations de réaction-diffusion. Il est notamment caractérisé par deux variables vectorielles[3] :

- $\mathbf{V}_a$, une fonction continue du temps, homogène à une tension et représentant l'état interne de la cellule ;
- $\mathbf{I}_a$, une fonction continue du temps, homogène à une intensité et représentant l'entrée de la cellule.

L'activité locale s'intéresse aux fluctuations d'énergie induites par des variations de l'entrée d'une cellule initialement placée en un point d'équilibre. En notant $\mathbf{v}_a$ et $\mathbf{i}_a$ les variations respectives de $\mathbf{V}_a$ et $\mathbf{I}_a$, la fluctuation d'énergie entre l'instant initial et un instant $T < \infty$, est donnée par :

$$w(T) = \int_0^T \mathbf{v}_a(t).\mathbf{i}_a(t).dt \tag{2.1}$$

On dira qu'une cellule est *localement active* à l'instant T si et seulement si $w(T) < 0$. En effet, en supposant nulle l'énergie de la cellule à $t = 0$, une énergie stockée strictement négative à $t = T$ signifie que la cellule a *apporté* de l'énergie au système. Une cellule active agit comme une source de puissance en amplifiant un signal faible. Par exemple, un transistor muni d'une batterie dans un circuit, ou bien un neurone muni d'une réserve de glucose dans un réseau, sont des éléments actifs ; batterie et glucose jouent le rôle d'une réserve d'énergie disponible que la cellule peut utiliser ou distribuer.

La complexité est finalement caractérisée comme la présence d'un motif (statique et non homogène, ou spatio-temporel) dans un médium (discret ou continu) constitué de cellules identiques interagissant avec leurs cellules voisines (sphère d'influence), obéissant aux mêmes lois d'interaction, et avec des conditions initiales et de bord homogènes. On peut alors montrer qu'un médium de *réaction-diffusion* présente de la complexité si, et seulement si, il existe des cellules localement actives dans ce médium.

Bien qu'aucun vocabulaire ne soit disponible au niveau des cellules pour en parler directement, ce travail montre comment des motifs peuvent être représentés par l'ensemble des cellules localement actives. Il s'agit bien là d'un exemple de complexification à rapprocher fortement des considérations développées dans le chapitre 3 sur l'identification des zones actives dans les modélisations MGS.

Une question reste néanmoins en suspens : lorsque les motifs ont des propriétés spatio-temporelles, l'ensemble des cellules localement actives peut varier en fonction du temps, amenant à la problématique du suivi des motifs malgré un support fluctuant. Les travaux d'A.

3. Nous utilisons ici les notations de [28].

Ehresmann, que nous introduisons dans la section suivante, proposent une réponse à cette question.

Complexification structurelle

Nous tâchons ici de présenter, sans entrer dans les détails, le *processus de complexification* introduit par A. Ehresmann et J.-P. Vanbermeersch [11]. Leurs travaux s'intéressent à l'utilisation des *systèmes évolutifs à mémoire* pour la modélisation du vivant, et plus particulièrement de la complexité. L'approche cherche à être la plus générale possible et repose sur une formalisation catégorique des systèmes dynamiques.

Pour plus de clarté, nous nous concentrons ici sur le concept de *Memory Evolutive Neural System* (MENS), une des applications de la méthodologie proposée à la modélisation des interactions neuronales. L'objectif de cette construction est de fournir des arguments en faveur de l'émergence des fonctions cognitives depuis les interactions neuronales, allant jusqu'aux concepts de conscience, de pensée et d'intention. Nous ne nous intéresserons ici qu'au processus de complexification.

Un réseau de neurones est représenté par un graphe Neur_t dont les nœuds, notés $N(t)$, sont des neurones et dont les arcs (orientés), notés $S(t)$, décrivent les liens entre les neurones. Chaque lien est pondéré par un délai de propagation. Le graphe est nécessairement fermé transitivement : étant donnés trois neurones $n_1, n_2, n_3 \in N(t)$, si les arcs (n_1, n_2) et (n_2, n_3) existent, l'arc (n_1, n_3) existent également. Un réseau est un objet dynamique et s'inscrit donc dans le temps (représenté par l'indice t). Au cours de son évolution, des nœuds peuvent apparaître ou disparaître, entraînant l'apparition ou la disparition d'arcs. L'ensemble des réseaux Neur_t forme une collection Neur (voir Fig. 2.4) munie d'une fonction de *transition* k qui, pour chaque couple d'instants $t < t'$, relie les nœuds de $N(t)$ à ceux de $N(t')$ et les arcs de $S(t)$ à ceux de $S(t')$ en préservant l'incidence, propriété assurée par la fonctorialité de k (la définition d'un foncteur est donnée dans la définition 2.4.8), et en vérifiant la transitivité suivante (présentée ici sur les nœuds uniquement) : pour trois instants $t < t' < t''$ et trois nœuds $(n_1, n_2, n_3) \in N(t) \times N(t') \times N(t'')$:

- si n_2 est l'image de n_1 par $k_{t \to t'}$ et n_3 est l'image de n_2 par $k_{t' \to t''}$, alors n_3 est l'image de n_1 par $k_{t \to t''}$; et
- si n_2 est l'image de n_1 par $k_{t \to t'}$ et n_3 est l'image de n_1 par $k_{t \to t''}$, alors n_3 est l'image de n_2 par $k_{t' \to t''}$.

Le processus de complexification est compris dans ce contexte comme la reconnaissance d'un comportement synchronisé, en prenant en compte les délais de communication imposés par les arcs du réseau, d'un groupe de neurones agissant collectivement sur d'autres neurones. Le point important de cette complexification est que ce comportement collectif peut lui-même être représenté par un neurone n. Ainsi, lorsqu'une famille P de neurones agit collectivement avec un neurone n' du réseau, cette action peut être représentée par un lien unique s entre n et n'. Le lien s joue un rôle totalement équivalent à la somme des liens des neurones de P

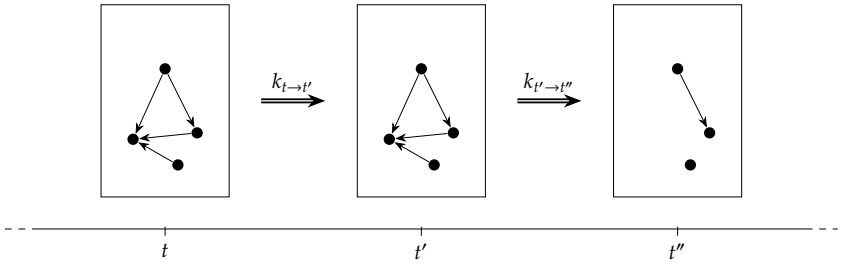


FIG. 2.4: Neur propose un point de vue dynamique sur un réseau de neurones. À chaque instant t, l'état du réseau est vu comme un graphe Neur_t.

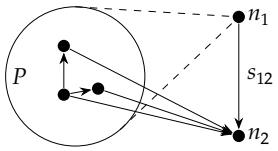


FIG. 2.5: Le cat-neurone N est un neurone abstrait ayant le même comportement fonctionnel que les neurones de la famille P.

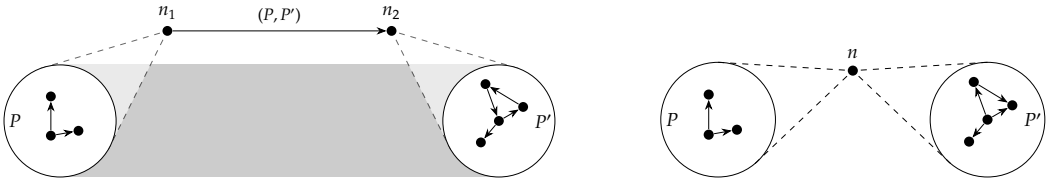


FIG. 2.6: À gauche, un lien simple : si P et P' forment un cluster, symbolisé par une bande grise, alors il existe un lien simple entre N, le binding de Pet N' le binding de P'. À droite, une bascule complexe : un cat-neurone peut être fonctionnellement équivalent à plusieurs familles de neurones.

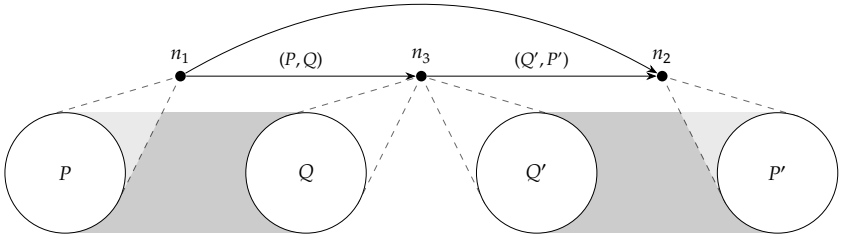


FIG. 2.7: Un lien complexe est la composition de deux liens simples à travers une bascule complexe.

vers n' (voir Fig. 2.5). Formellement, n correspond à une colimite du diagramme défini par P, appelée *binding* dans ce contexte. Le cas intéressant apparaît lorsque le *binding* n'existe pas, c'est-à-dire lorsqu'aucun neurone de Neur ne représente de façon élémentaire le comportement collectif. On propose alors d'enrichir le système initial Neur avec ces nouveaux neurones abstraits, appelés *cat-neurones*, et correspondant aux *bindings* manquants. Une hiérarchie de complexification peut alors être mise en place : les neurones de Neur sont des *cat-neurones* de niveau 0, alors que les *bindings* manquants sont des cat-neurones de niveau 1. Le processus peut être itéré pour obtenir des cat-neurones de niveau l à partir des cat-neurones de niveau $l-1$.

En plus d'être capable de réifier les comportements « émergents » en tant qu'objets de base, ici des neurones, le modèle permet également le suivi au cours du temps de ces comportements. Ce suivi repose sur deux notions :

1. *Lien simple*: un lien simple permet d'associer l'évolution d'un cat-neurone à l'évolution de la famille qui l'engendre. Soient P et P' deux familles de neurones. Un *cluster* entre P et P' est une famille maximale de liens entre les neurones de P et ceux de P' représentant l'évolution des neurones de P en neurones de P' (nous ne rentrerons pas plus dans les détails techniques). Si P et P' admettent n et n' comme *bindings* respectifs, le *cluster* admet également une colimite sous la forme d'un lien entre n et n', appelé *lien simple*. Ce lien représente l'évolution de n en n' (voir Fig. 2.6 à gauche).

2. *Bascule complexe*: un lien simple ne permet pas à lui seul le suivi d'un cat-neurone car la famille P peut disparaître au cours de son évolution. Cependant, d'autres familles que P peuvent engendrer le même cat-neurone. On dira que deux familles de neurones P et P' forment une *bascule complexe* lorsque P et P' engendrent le même *binding* (voir Fig. 2.6 à droite).

On trouvera Fig. 2.7 une illustration de la notion de *lien complexe* montrant comment combiner *lien simple* et *bascule complexe* pour déterminer l'évolution d'un cat-neurone malgré la structure dynamique du réseau.

2.2.4 Bilan

L'ensemble des travaux présentés ci-dessus nous amènent à considérer qu'un outil de modélisation devrait regrouper les propriétés suivantes :

Cadre unifié : Les conclusions de [27] sur leur expérience de multi-modélisation d'une cellule entière de *Mycoplasma genitalium* montrent qu'une uniformisation des descriptions est nécessaire pour produire un outil adapté à l'intégration de données provenant de sources multiples. Les différents points de vue pris sur un même objet, voire la disparité des systèmes étudiés, impliquent une indépendance de l'outil par rapport aux modèles pour permettre l'*identification* des concepts partagés. Un cadre unifiant et générique doit donc être développé de façon agnostique vis-à-vis des sujets d'étude.

Cadre formel : L'analyse d'un modèle, allant de la simple simulation à des techniques mathématiquement plus complexes (*model checking*, analyse abstraite, etc.), repose sur une description précise des modèles. Pour cela, les modèles sont souvent décrits dans un cadre mathématique spécifique, appelé *formalisme*, dont la frontière permet d'assurer l'applicabilité des techniques d'analyse.

Au-delà des facilités mathématiques offertes par les formalismes, un grand nombre d'approches utilisent de façon formelle les propriétés « syntaxiques » des spécifications des modèles afin de les manipuler. Les transformations de modèles illustrent parfaitement cette dépendance à la description des modèles. Les preuves de raffinement imposent que le modèle transformé traduise effectivement les éléments du modèle d'origine.

La prise en compte de plusieurs formalismes va néanmoins à l'encontre du caractère unifiant que nous impose le point précédent. Trouver un sur-formalisme unique à un ensemble de formalismes classiques amène nécessairement à lever les frontières qui faisaient des approches classiques leur intérêt : plus un formalisme est général, moins il pourra nous apprendre sur les modèles étudiés. Il est donc primordial de trouver un point d'équilibre entre ces deux premiers points.

Sémantique : Un modèle ne se résume pas à une spécification mathématique décrite dans un formalisme donné. En effet, si tel était le cas, un modèle se réduirait à un objet symbolique vide de relations au système qu'il décrit. Sa relation avec l'objet d'étude passe par une sémantique, c'est-à-dire, par une structure supplémentaire décrivant l'interprétation des symboles.

La sémantique joue un rôle déterminant dans la compréhension et la manipulation des modèles. Par exemple, le couplage de modèles repose essentiellement sur l'*identification* de concepts communs traités différemment par les modèles. Une telle sémantique agit comme un moteur du couplage : elle permet d'*identifier* au sein de leurs spécifications les éléments en relation avec les mêmes concepts. Un projet tel que l'intégration dans un seul modèle de résultats scientifiques indépendants profiterait d'une description homogène faisant référence à une ontologie partagée, c'est-à-dire une base de concepts communs.

Abstraction : Comme le montrent les techniques de raffinement, il est naturel d'élaborer un modèle à partir d'une spécification grossière qui sera incrémentalement transformée vers une forme plus dense prenant en compte des aspects de plus en plus précis du système étudié. Si un modèle est vu comme une abstraction d'un système, ce type de construction consiste à passer d'une description abstraite vers une autre plus concrète, en respectant la sémantique, dont l'objectif est de s'approcher à la limite du système lui-même.

Les travaux cités précédemment nous laissent penser que ce mécanisme est à la base de toute modélisation. À titre d'exemple, le couplage de modèles peut être vu comme une forme d'utilisation de l'abstraction : le modèle obtenu par couplage peut être vu comme le raffinement le plus abstrait obtenu en combinant les modèles initiaux. Ainsi, Simpop3 est à la fois un raffinement de Simpop2 et un raffinement de SimpopNano.

Multi-niveau : La problématique de la complexification, notamment à travers les travaux de S. Stepney, montre que l'abstraction n'est pas une opération suffisante pour parler des propriétés émergentes. La difficulté vient essentiellement du manque de vocabulaire disponible à certains niveaux de description pour exprimer des concepts présents de façon élémentaire à d'autres niveaux, bien qu'il soit possible de les caractériser qualitativement. La modélisation multi-niveau a pour objectif de mettre en avant cette possibilité d'*identifier* les entités de base d'un niveau comme étant des propriétés observées à un autre niveau. Les travaux de A. Ehresmann et L. Chua proposent d'*identifier* cette fois une propriété émergente à travers le comportement corrélé d'entités, par l'activation commune d'un même cat-neurone dans le premier cas ou par la mesure de l'activité d'une cellule dans le second. Les entités du niveau inférieur enrichissent le niveau supérieur, mais ne le définissent pas entièrement. Un cat-neurone ou une cellule localement active a la capacité d'influer en retour sur le niveau inférieur et c'est l'association de ces deux mécanismes qui détermine les entités des niveaux supérieurs.

Le Bateau de Thésée. Les propriétés précédentes reposent fondamentalement sur différentes formes d'identification, que ce soit entre les symboles d'un modèle et les concepts sémantiques associés, les relations entre un modèle abstrait et son raffinement, ou encore durant un processus de complexification. Cependant, l'identité est une notion polysémique riche dont les différentes compréhensions peuvent aider à préciser certaines problématiques rencontrées en modélisation.

Pour mettre en avant la profondeur du concept d'identité, le paradoxe du bateau de Thésée est souvent employé. Plutarque, dans l'ouvrage « La vie de Thésée » (traduit par D. Ricard [38]), fait état de la conservation de son bateau par les athéniens après la mort de Thésée :

> Le vaisseau sur lequel Thésée s'était embarqué avec les autres jeunes gens, et qu'il ramena heureusement à Athènes, était une galère à trente rames, que les Athéniens conservèrent jusqu'au temps de Démétrios de Phalère. Ils en ôtaient les vieilles pièces, à mesure qu'elles se gâtaient, et les remplaçaient par des neuves qu'ils joignaient solidement aux anciennes. Aussi les philosophes, en disputant sur ce genre de sophisme qu'ils appellent croissant, citent ce vaisseau comme un exemple de doute, et soutiennent les uns que c'était toujours le même, les autres que c'était un vaisseau différent.

La question est donc de déterminer si le bateau de Thésée, malgré sa conservation et son entretien par les athéniens, garde son identité une fois que toutes ses parties ont été remplacées. De plus, si lors des réparations chaque pièce remplacée était utilisée pour reconstruire un bateau complet, ce dernier serait-il le bateau de Thésée ?

Ces questions autour de l'identité qui hantent les philosophes depuis l'antiquité, ont abouti à en donner différentes interprétations. Dans [12], S. Ferret fait un état des lieux de ces interprétations dont voici une brève description :

Identité numérique : Il s'agit de la relation qu'entretient un objet avec lui-même au cours du

temps. Dans l'exemple précédent, l'*identité numérique* du bateau de Thésée est conservée au cours du temps, même si le bateau subit des modifications au fur et à mesure. C'est la seule identité qui résiste au temps.

Identité spécifique : Il s'agit de l'appartenance à une catégorie commune, indépendamment de l'objet concerné. Ainsi, les planches constituant initialement le bateau de Thésée font partie de la même *identité spécifique* et s'opposent aux planches nouvellement installées. D'un point de vue logique, une identité spécifique peut être associée à la propriété séparant les objets qui lui appartiennent de ceux qui ne lui appartiennent pas.

Identité qualitative : Aussi appelée indiscernabilité, il s'agit de l'identification d'objets que l'expérience ne permet pas de distinguer. Pris au sens le plus strict, deux objets ayant les mêmes propriétés (qualités), c'est-à-dire répondant de la même façon à *toutes* les questions qui leurs seraient posées, sont identiques.

Il est intéressant de constater que ces points de vue se mêlent en modélisation. Notamment, la complexification structurelle d'A. Ehresmann permet d'observer les trois identités à l'œuvre au sein d'une même construction. La Fig. 2.8 présente leurs interactions lors de la construction d'un lien complexe : l'identité numérique des (cat-)neurones dont on suit l'évolution au cours du temps, l'identité spécifique des cat-neurones d'un même niveau tous descriptibles de la même façon, et l'identité qualitative des familles de neurones avec *binding* caractérisées par leur comportement synchronisé. On remarquera notamment sur ce dernier point l'indiscernabilité à l'œuvre lorsque des familles d'une bascule complexe – se comportant en tout point de la même façon avec les autres neurones – sont représentées par le même cat-neurone.

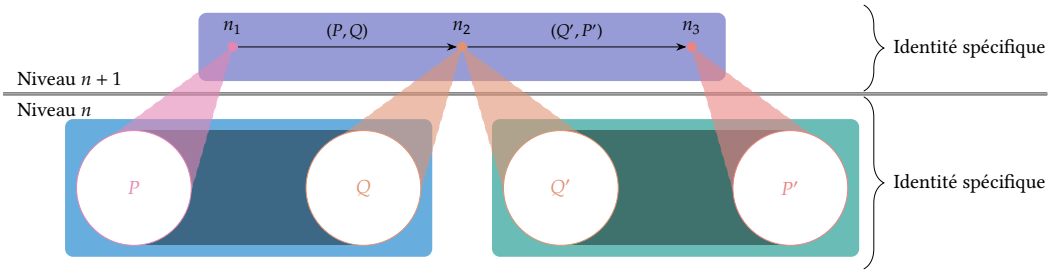


Fig. 2.8: Identités et lien complexe : les identités qualitatives sont données en magenta, en orange et en rouge, les identités numériques sont données en violet, en bleu et en cyan.

2.3 Une notion générale de modèle

Comme nous venons de le voir, le concept de modèle est riche et sa compréhension varie suivant le domaine ou la technique utilisée. Notre objectif étant de s'écarter des cas particuliers pour proposer un outil générique pour comprendre la modélisation multi-niveau, il nous est nécessaire de définir une notion de modèle qui permettra de rendre compte du plus grand nombre de cas d'application. Dans cette section, nous proposons une telle définition.

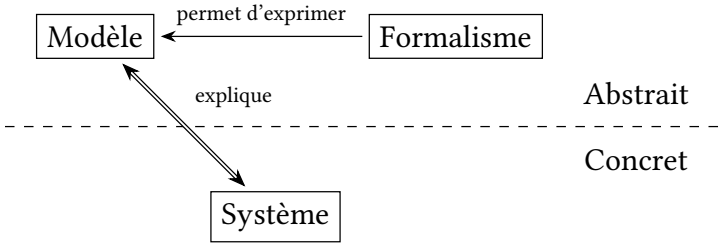


FIG. 2.9: Schéma des relations entre de notions de *système*, *modèle* et *formalisme*.

Nous commençons par fixer une partie du vocabulaire utilisé pour finalement proposer la définition générale qui nous intéresse. Cette généralité étant un frein à l'étude des modèles, nous en proposons ensuite différentes instanciations permettant de retrouver quelques classes de modèles couramment rencontrées. Nous terminons par la présentation de différents modèles d'un même système, un système proie/prédateur.

2.3.1 Vocabulaire

Les trois notions importantes qui nous intéressent dans ce chapitre sont celles de *système*, de *modèle* et de *formalisme*. Bien que, d'un point de vue général, elles sont acceptées de tous, l'usage de ces termes peut varier suivant le domaine. Notre objectif est de lever toute ambigüité en proposant au lecteur notre propre compréhension de ces notions. En outre, nous utiliserons ces définitions comme fondement de notre proposition. Ainsi, la figure FIG. 2.9, résumant les relations que nous avons pu mettre en avant section 2.2, constitue le canevas du développement décrit section 2.4.

Système. On trouve dans de nombreuses disciplines l'utilisation du mot *système*. En voici quelques exemples :

- En informatique, le *système* d'exploitation est un *assemblage* de programmes et de processus chargés de la gestion des périphériques au sens large : la mémoire, les communications via une carte réseau, les entrées utilisateur, les ressources processeur, etc. ;
- En anatomie, le terme *système* désigne un *assemblage* d'organes interagissant au sein d'un organisme dans la réalisation d'une fonction biologique commune : le *système* nerveux, le *système* digestif, etc. ;
- En physique, un *système* est un *assemblage* de points, de solides, constitutif d'un fluide, d'un gaz, ... dont il faut décrire l'évolution au cours du temps ;
- En logique, un *système* formel est un *assemblage* comprenant un vocabulaire, un ensemble d'axiomes et des règles de déduction.

Tous les exemples supplémentaires que nous pourrions trouver reposeraient eux aussi sur deux notions essentielles :

1. Le terme *système* désigne un objet d'étude. Il s'oppose à l'environnement, c'est-à-dire à ce que l'on ne souhaite pas étudier mais qui est potentiellement en interaction avec l'objet d'étude. Suivant le degré d'importance de ces interactions, les termes de *système* ouvert, fermé ou isolé seront plus particulièrement employé.

2. Dans son étude, le *système* est décomposé en parties, et l'étude porte sur la compréhension des relations entre ces parties. D'ailleurs, l'origine étymologique du mot *système* est σύστημα qui, en grec ancien, désigne un tout composé de plusieurs parties ou membres. Cette décomposition a son importance dans notre approche car elle offre le vocabulaire nécessaire pour désigner le système et, nous le verrons par la suite, constitue une base pour la définition d'une sémantique.

Modèle. L'acceptation générale du terme *modèle* fait état d'un objet utilisé comme représentation d'un autre[4]. Suivant la définition que nous venons de donner, l'objet représenté correspond à un *système*, dont un *modèle* a la charge d'en donner une représentation suffisamment fidèle pour procéder à l'étude attendue. Dans certains cas, le *système* est existant et le *modèle* servira de représentation simplifiée en vue d'une analyse souvent explicative du *système*; dans d'autres cas, il correspondra à une construction que nous souhaitons réaliser, le *modèle* permettra alors, dans une approche d'ingénierie, d'étudier la possibilité de réaliser le système, voire d'en guider la construction.

L'aspect dégradé de la représentation offerte par le *modèle* en fait une abstraction du *système*. Elle place les deux concepts dans des mondes *a priori* différents, l'un abstrait et l'autre concret. Par exemple le premier peut être mathématique lorsque la modélisation est formelle et le second réel quand l'objet d'étude existe dans la nature. Cependant, c'est bien la relation entre le *modèle* et le *système* qui fait la modélisation et non le *modèle* seul. En effet, les abstractions apportées par le *modèle* ne font sens que lorsqu'elles se font *par rapport système*. Ainsi, la sémantique associant les abstractions du *modèle* au vocabulaire amené par le *système* complète la relation *modèle-système*. Nous développerons cette idée formellement dans la section 2.4.

Formalisme. Tout *modèle* nécessite un langage comme support d'expression. Ce langage peut être compris au sens large et prend de nombreuses formes. Prenons par exemple la réalisation d'une maquette d'un pont pour en étudier la réponse aux vents en soufflerie. Le support de description dans ce cas n'est autre que le matériel constituant la maquette.

Nous nous limiterons ici aux modélisations dont le langage de description repose sur une construction mathématique. Ces constructions sont appelées *formalismes* et sont la donnée d'une théorie, au sens mathématique, c'est-à-dire un ensemble d'affirmations et un mécanisme de déduction. Cette théorie est accompagnée d'une syntaxe permettant d'en formuler les affirmations.

4. Il existe néanmoins une compréhension opposée de la notion de modèle, que l'on trouve par exemple dans le monde artistique où un *modèle* est utilisé pour ensuite être représenté dans une œuvre (peinture, dessin, littérature, etc.). Dans ce sens, on parle d'une forme d'idéal qu'on cherche à atteindre. La définition à laquelle nous nous réfèrerons est celle du *modèle* scientifique.

Comme évoqué plus haut, la difficulté des approches combinant plusieurs *modèles* d'un même *système* vient de la diversité des *formalismes* utilisés. Il nous semble que la théorie des ensembles offre un cadre commun minimal suffisant pour élaborer notre outil. En effet, la plupart des *formalismes* que nous avons rencontrés se fondent sur une représentation ensembliste. C'est le cas par exemple des algèbres (au sens de l'algèbre universelle) vues comme des ensembles munis de lois de composition internes. Nous verrons section 2.3.2 comment l'enrichissement d'un ensemble par une structure mathématique, c'est-à-dire en se plaçant dans un *formalisme* particulier, permet de retrouver certaines classes de *modèles*. Nous verrons également, section 2.4, comment oublier ces structures permet aux *modèles* de collaborer.

Les systèmes de Lindenmayer. Dans ce paragraphe, nous souhaitons montrer que si, dans une situation particulière, nous savons différencier *système*, *modèle* et *formalisme*, un même objet peut servir suivant le contexte de *système*, de *modèle* ou de *formalisme*.

Les systèmes de Lindenmayer, ou *L-systems*, en sont un bon exemple. Les L-systems sont des grammaires formelles introduites par A. Lindenmayer [10]. Formellement[5], un L-system est un triplet (Σ, w_0, P) où Σ est un ensemble fini de symboles appelé *alphabet*, $w_0 \in \Sigma^*$ est un mot fini initial appelé *axiome*, et $P \subset \Sigma \to \Sigma^*$, appelé l'ensemble des règles, est une fonction associant à chaque lettre de Σ un mot. Un L-system transforme ainsi un mot $a_1 \dots a_n$ en $w_1 \dots w_n$ tel que pour tout i, $w_i = P(a_i)$.

Les L-systems peuvent être observés sous le prisme des trois concepts que nous avons abordés :

1. L-system comme *modèle*: Au même titre que le λ-calcul [8], les machines de Turing [44] ou les grammaires de Chomsky [6], les L-systems ont leur place dans l'étude du calcul en tant que *modèles* de calcul. Dans ce contexte, c'est-à-dire celui de calculabilité, le *système* désigne les fonctions calculables et les modèles énumérés ci-dessus en proposent différents points de vue. En effet, nous pouvons associer un langage à un L-system à travers l'ensemble des mots qu'il génère à partir de l'axiome. Durant les années 70, un important effort de recherche a été entrepris pour comprendre la puissance de calcul proposé par les L-systems [39].

2. L-system comme *système*: Afin de les élever comme modèles de calcul, les L-systems ont fait l'objet d'une étude formelle devenant alors un *système*, suivant le sens évoqué plus haut. Ainsi, nous trouvons également dans [39] des abstractions qui ont permis de mener à bien cette étude. Pour n'en donner qu'un exemple simple, les *fonctions de croissance* permettent à partir d'un L-system d'obtenir la longueur du mot obtenu après n transformations issues de l'axiome initial. Dans notre cas, décrire la fonction de croissance revient à poser un système d'équations mutuellement récursives dénombrant le nombre d'occurrences de chaque lettre à une génération donnée.

3. L-system comme *formalisme*: Il s'agit ici de l'utilisation initiale des L-systems proposée

5. Pour la simplicité du propos, nous restreignons notre présentation au D0-L-system, c'est-à-dire aux L-systems déterministes sans contexte.

par A. Lindenmayer qui avait pour objectif de modéliser le développement des plantes et des bactéries. Dans ce troisième contexte, les L-systems sont utilisés comme *formalisme* pour décrire des modèles biologiques. Par exemple, dans [36] se trouve, parmi de nombreuses autres applications montrant la richesse de l'approche, le L-system $L_{\text{AC}} = (\{r, R, l, L\}, R, P_{\text{AC}})$ comme *modèle* symbolique du comportement de croissance de l'algue *Anabaena catenula*. Cette algue se présente sous la forme d'un filament de cellules. Les cellules sont polarisées (vers la gauche ou vers la droite) et le filament présente un motif de polarisation spécifique. Les règles de P_{AC}, données par

$$\begin{array}{lcl} r & \Rightarrow & R \\ l & \Rightarrow & L \\ R & \Rightarrow & Lr \\ L & \Rightarrow & lR \end{array}$$

décrivent comment naît ce motif. Il repose sur une asymétrie de longueur des cellules filles lors de la division cellulaire. Dans cet exemple, r est l'abstraction d'une cellule courte orientée vers la droite, L d'une cellule longue orientée vers la gauche, etc. La règle $R \Rightarrow Lr$ abstrait la division d'une cellule longue orientée à droite, en deux cellules, respectivement longue et orientée à gauche, et courte et orientée à droite.

2.3.2 Constructions de modèles

Comme nous l'avons présenté précédemment, nous souhaitons nous placer dans le contexte de la théorie des ensembles, dénominateur commun de tous les formalismes formalismes de modélisation introduits jusqu'à présent. Ainsi, nous considérons qu'un modèle est un ensemble de « faits » à propos du système concerné.

Cette simple définition confirme, d'une part, que sans sémantique, un modèle ne nous apprend que peu d'informations sur le système qu'il décrit. Pire encore, en supposant qu'une telle sémantique existe, que l'absence de structure des ensembles ne nous permet pas de relier entre eux les faits que le modèle contient. Le prix à payer pour unifier les formalismes semble trop élevé pour mener à bien notre objectif.

Cependant, un modèle n'est pas n'importe quel ensemble, mais un ensemble issu d'un formalisme bien particulier et qui présente donc les reliquats d'une structure préexistante. Ainsi, tous les modèles correspondant à un formalisme donné exhiberont une forme commune qui peut être utilisée pour classer les modèles. Dans cette section, nous présentons à travers quelques exemples comment enrichir cette vue ensembliste de la modélisation pour retrouver des classes de modèles fréquemment rencontrées.

Dans la suite, nous utiliserons la notation $\mathfrak{M}$ pour désigner un modèle. Afin de distinguer la façon de construire l'ensemble des faits d'un modèle du modèle lui-même, nous noterons $E_{\mathfrak{M}}$ l'ensemble des faits de $\mathfrak{M}$.

Modèle à observables

Dans ce premier exemple, nous considérons que les faits que contiennent les modèles représentent les *états* du système, c'est-à-dire une description des différentes configurations que peut prendre le système. Chaque état correspond à une observation du système et peut être décrit comme un tuple de valeurs, appelées *observables*. En d'autres termes, le modèle correspond à un ensemble de mesures relevées sur le système étudié.

Chaque observable produit une mesure dans un ensemble de valeurs, appelé le *domaine* de l'observable. Un modèle définit donc une *loi d'exclusion* : il distingue parmi l'ensemble de toutes les mesures théoriques possibles la classe des comportements propres au système et exclut celles qui ne le sont pas. Cette approche a été développée par J.C. Willems [46]. L'ensemble des mesures possibles, donné par le produit cartésien des domaines des observables est parfois appelé *universum* dont un modèle est le sous-ensemble. Cela nous amène à la définition formelle suivante.

Définition 2.3.1 (Modèle à observables)**.** *Soit* $\{\mathbb{D}_i\}_{i\in I}$*, une famille d'ensembles. Un* modèle à observables $\mathfrak{M}$ *est un modèle caractérisé par un couple* $\langle \Sigma_{\mathfrak{M}}, \mathfrak{B}_{\mathfrak{M}} \rangle$ *où*

- $\Sigma_{\mathfrak{M}} = \mathbb{D}_1 \times \mathbb{D}_2 \times \mathbb{D}_3 \times \ldots$ *est la* signature *(encore appelé* universum*) du modèle* $\mathfrak{M}$*;*
- $\mathfrak{B}_{\mathfrak{M}} \subseteq \Sigma_{\mathfrak{M}}$ *est le* comportement *du modèle* $\mathfrak{M}$*.*

Les projections $\pi_i : \Sigma_{\mathfrak{M}} \to \mathbb{D}_i$ *sont appelées les* observables *du modèle et* $\mathbb{D}_i$ *est le* domaine *de l'observable* π_i*. Les éléments de* $\mathfrak{B}_{\mathfrak{M}}$ *sont appelés les* états *du modèle.*

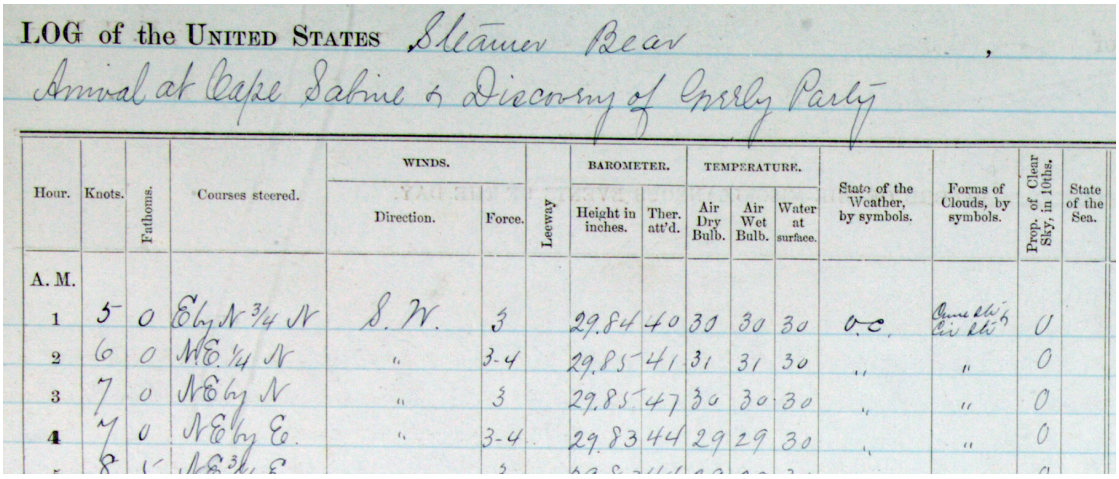
L'ensemble des faits est donné par $E_{\mathfrak{M}} = \mathfrak{B}_{\mathfrak{M}}$*.*

Un modèle à observables pourra être plus ou moins précis dans sa description du système, en présentant plus ou moins d'observables, en reposant sur des domaines plus ou moins fins, ou en dénombrant plus ou moins d'états. Cette précision sera formalisée dans la section suivante par un mécanisme d'abstraction entre modèle.

Exemple 2.3.1. Nous reprenons ici la modélisation des états de l'eau de [46]. Suivant la température, l'eau existe dans différents états solide, liquide ou gazeux. Le modèle $\mathfrak{M}_{eau}$ suivant décrit les états que prend l'eau pour différentes températures, données ici en degrés Celsius (°C), à une pression constante de 1013,25 hPa:

$$\begin{aligned}
\mathbb{D}_{\texttt{Phase}} &= \{\text{solide}, \text{liquide}, \text{gaz}\} \\
\mathbb{D}_{\texttt{Température}} &= [-273, \infty[\\
\Sigma_{\mathfrak{M}_{eau}} &= \mathbb{D}_{\texttt{Phase}} \times \mathbb{D}_{\texttt{Température}} \\
\mathfrak{B}_{\mathfrak{M}_{eau}} &= \{\text{solide}\} \times [-273, 0] \\
&\cup \{\text{liquide}\} \times [0, 100] \\
&\cup \{\text{gaz}\} \times [100, \infty[
\end{aligned}$$

Notons que d'après ce modèle, l'eau à une température de 0 °C peut présenter les deux phases, liquide et solide.



LOG of the UNITED STATES Steamer Bear

Arrival at Cape Sabine & Discovery of Greely Party

Hour.	Knots.	Fathoms.	Courses steered.	WINDS. Direction.	WINDS. Force.	Leeway	BAROMETER. Height in inches.	BAROMETER. Ther. att'd.	TEMPERATURE. Air Dry Bulb.	TEMPERATURE. Air Wet Bulb.	TEMPERATURE. Water at surface.	State of the Weather, by symbols.	Forms of Clouds, by symbols.	Prop. of Clear Sky, in 10ths.	State of the Sea.
A. M.															
1	5	0	Ely N ¾ N	S. W.	3		29.84	40	30	30	30	0.c.	Cum Cir Str	0	
2	6	0	NE ¼ N	"	3-4		29.85	41	31	31	30	"	"	0	
3	7	0	NE by N	"	3		29.85	47	30	30	30	"	"	0	
4	7	0	NE by E.	"	3-4		29.83	44	29	29	30	"	"	0	
	8		NE ¾ E												

FIG. 2.10: Extrait d'une page du journal de bord du bateau à vapeur Bear de l'US Navy, le 22 juin 1884.

Modèle expérimental

Voir un modèle comme un ensemble ne dit rien sur la façon de construire ce modèle. Une première manière de spécifier un ensemble est d'en donner une définition par extension. On parlera de *modèle expérimental* lorsque l'ensemble est défini par extension à partir des résultats d'expériences.

Exemple 2.3.2. Une méthode directe de construction d'un modèle à observables consiste à noter les valeurs mesurées de chaque observable sur une feuille, comme dans le cas d'un journal de bord « maritime », où les observations relevées se limitent à celles qui sont estimées pertinentes pour la navigation (voir FIG. 2.10), et à effectuer autant de relevés que possible. On peut considérer qu'à chaque ligne un *état du système* est décrit et que les valeurs relevées définissent une relation (au sens mathématique) entre les observables. Les colonnes correspondent aux observables (π_i dans la définition ci-dessus) et toutes les valeurs qu'une observable peut prendre forment son domaine ($\mathbb{D}_i$ ci-dessus). Par exemple,

$$\mathbb{D}_{\texttt{WINDS Direction}} \in \{N, NE, E, SE, S, SW, W, NW\}$$

Modèle équationnel

Distinguer un ensemble dans un sur-ensemble de possibilités consiste à donner un prédicat décidant de l'appartenance d'un élément à l'ensemble. Une façon de spécifier un tel prédicat est décrit dans [46] par la donnée d'une équation mettant en relation deux quantités. Ce principe amène à la notion suivante de modèle équationnel.

Définition 2.3.2 (Modèle équationnel)**.** *Un* modèle équationnel $\mathfrak{M}$ *est caractérisé par un couple de fonctions* $\langle f_{\mathfrak{M}}, g_{\mathfrak{M}} \rangle$ *de même signature* $X \to Y$ *pour deux ensembles* X *et* Y *donnés, tel que*

l'ensemble des faits de $\mathfrak{M}$ est donné par

$$E_{\mathfrak{M}} = \{x \in X \mid f_{\mathfrak{M}}(x) = g_{\mathfrak{M}}(x)\}$$

Exemple 2.3.3. Associés à la notion de modèle à observables, les modèles équationnels sont fréquents en physique. À ce titre, l'équation la plus couramment utilisée pour caractériser les gaz parfaits illustre la définition 2.3.2.

D'un point de vue macroscopique, l'état de n moles d'un gaz parfait à l'équilibre thermodynamique est décrit par les trois observables `Volume`, `Pression` et `Température`. On peut construire le modèle $\mathfrak{M}_{gaz}$ des gaz parfaits dont l'ensemble des états est donné par :

$$E_{\mathfrak{M}_{gaz}} = \{(V, P, T) \in \mathbb{D}_{\texttt{Volume}} \times \mathbb{D}_{\texttt{Pression}} \times \mathbb{D}_{\texttt{Température}} \mid PV = nRT\}$$

où $R = 8{,}314\,462\,1\,\mathrm{J\,K^{-1}\,mol^{-1}}$ représente la constante des gaz parfaits. Le modèle $\mathfrak{M}_{gaz}$ est donc un modèle équationnel caractérisé par le couple de fonctions

$$\langle (V, P, T) \mapsto PV, (V, P, T) \mapsto nRT \rangle.$$

Modèle à observables privilégiées

Les classes de modèles que nous avons données jusqu'ici correspondent à la distinction d'un sous-ensemble admissible de faits à partir d'un univers plus grand. Avec cette nouvelle classe, nous commençons à nous intéresser aux modèles dont l'ensemble associé peut être enrichi d'une structure mathématique particulière provenant d'une propriété qu'il exhibe. Dans ce premier exemple, on s'intéresse aux modèles à observables dont le comportement est entièrement déterminé par une partie de ses observables, que nous appellerons *observables privilégiées*.

Définition 2.3.3 (Modèle à observables privilégiées)**.** *Soient $\mathfrak{M} = \langle \Sigma_{\mathfrak{M}}, \mathfrak{B}_{\mathfrak{M}} \rangle$ un modèle avec observables I (*i.e.*, $\Sigma_{\mathfrak{M}} = \prod_{i \in I} \mathbb{D}_i$) et $J \subset I$ un sous-ensemble d'observables. Le modèle $\mathfrak{M}$ est* à observables privilégiées J *si, et seulement si,*

$$\forall b_1, b_2 \in \mathfrak{B}_{\mathfrak{M}} \quad \pi_J(b_1) = \pi_J(b_2) \Rightarrow b_1 = b_2$$

Cette définition n'est pas sans rappeler quelques concepts propres au domaine des bases de données relationnelles, introduites par E. F. Codd dans [9]. Un modèle à observables $\mathfrak{M}$ est très proche d'une relation, au sens de l'algèbre relationnelle, avec sa signature $\Sigma_{\mathfrak{M}}$ pour *schéma* et son comportement $\mathfrak{B}_{\mathfrak{M}}$ comme *extension*, l'ensemble des tuples de la relation. Les observables privilégiées sont elles à rapprocher de la notion de *clé primaire* (ou plus précisément de *super-clé*) : dans un modèle à observables privilégiées, chaque $p \in \prod_{j \in J} \mathbb{D}_j$ identifie de manière unique un tuple de $\mathfrak{B}_{\mathfrak{M}}$ s'il existe. Plus précisément, c'est à la notion de *dépendance fonctionnelle* que correspondent les observables privilégiées. En effet, l'équation de la définition décrit l'existence d'une fonction codée au sein de $\mathfrak{B}_{\mathfrak{M}}$.

Exemple 2.3.4. En reprenant le modèle $\mathfrak{M}_{gaz}$ de l'exemple 2.3.3, il est possible de remarquer que l'équation des gaz parfaits permet de déterminer sans ambiguïté n'importe laquelle de ces

observables à partir des deux autres. Nous pouvons donc montrer qu'à partir de $\mathfrak{M}_{gaz}$, trois modèles à observables privilégiées reposant sur les trois dépendances fonctionnelles peuvent êtres construits : (`Volume`, `Pression` → `Température`), (`Température`, `Pression` → `Volume`) et (`Température`, `Volume` → `Pression`).

Modèle dynamique

L'étude des *systèmes dynamiques* porte sur l'évolution des états du système au cours du temps. Usuellement, les aspects temporels sont modélisés par l'action du temps sur l'état du système. Cette action repose sur la nature monoïdale [16] de l'ensemble utilisé pour représenter le temps. Un *monoïde* est un triplet $\mathbb{T} = \langle D_{\mathbb{T}}, 0_{\mathbb{T}}, +_{\mathbb{T}} \rangle$ (noté ici additivement) tel que $D_{\mathbb{T}}$ est un ensemble arbitraire, $+_{\mathbb{T}}$ est une loi de composition interne binaire *associative* ($\forall x, y, z \in D_{\mathbb{T}}, x +_{\mathbb{T}} (y +_{\mathbb{T}} z) = (x +_{\mathbb{T}} y) +_{\mathbb{T}} z$) et munie d'un élément *neutre* $0_{\mathbb{T}}$ ($\forall x \in D_{\mathbb{T}}, 0_{\mathbb{T}} +_{\mathbb{T}} x = x +_{\mathbb{T}} 0_{\mathbb{T}} = x$). Dans le cadre de la modélisation du temps, les éléments de $D_{\mathbb{T}}$ correspondent à des *durées* et la loi de composition fournit un moyen de cumuler ces *durées*. L'*action* d'un monoïde sur un ensemble arbitraire E est une fonction $\Phi : E \times D_{\mathbb{T}} \to E$ telle que $\forall x \in E, \forall \delta_1, \delta_2 \in D_{\mathbb{T}}$:

$$\Phi(x, 0_{\mathbb{T}}) = x \qquad \Phi(\Phi(x, \delta_1), \delta_2) = \Phi(x, \delta_1 +_{\mathbb{T}} \delta_2)$$

En interprétant les éléments de E comme une modélisation des états d'un système, l'action Φ spécifie une fonction de transition précisant comment, après une durée $\delta \in D_{\mathbb{T}}$, le système passe d'un état $x \in E$ à un nouvel état $\Phi(x, \delta)$.

Cette définition très générale n'impose aucune propriété particulière sur le monoïde utilisé pour modéliser le temps. On retrouve naturellement les modèles de temps classiques, en prenant l'ensemble des réels pour modéliser un temps continu ou l'ensemble des entiers pour modéliser un temps discret. Mais cette définition s'applique également dans des cas moins courants. Par exemple, considérons un automate à états finis, décrivant les transitions d'état en état à la lecture d'un mot construit à partir d'un alphabet, disons $\Sigma = \{a, b\}$. On peut interpréter Σ^*, le monoïde libre engendré par les lettres de l'alphabet, comme une modélisation d'un temps dont les durées sont des accumulations de a et de b. On peut montrer facilement que la fonction de transition de l'automate spécifie une action de monoïde. Ce qui est intéressant dans cette description est que le temps n'est pas ici un temps linéaire où les « instants » s'enchaînent d'une façon totalement ordonnée, mais un temps arborescent dont les branches peuvent être interprétées comme différentes possibilités d'exécutions. Cet exemple rend bien compte de la souplesse de cette définition.

Nous proposons de caractériser les *modèles dynamiques* en s'inspirant de la définition des système dynamiques[6] donnée dans [16].

Définition 2.3.4 (Modèle dynamique)**.** *Un* modèle dynamique *est un modèle* $\mathfrak{M}$ *caractérisé par un triplet* $\langle \mathfrak{B}_{\mathfrak{M}}, \mathbb{T}_{\mathfrak{M}}, \Phi_{\mathfrak{M}} \rangle$ *tel que*

6. On remarquera qu'ici le terme *système* ne correspond pas à la notion que nous avons définie section 2.3.1 mais est plus à rapprocher de celle d'un *formalisme* permettant la *modélisation* de la dynamique d'un *système*.

1. $\mathbb{T}_{\mathfrak{M}}$ *est un monoïde appelé* modèle du temps *et dont les éléments* $\delta \in D_{\mathbb{T}}$ *sont appelés* durées,

2. $\mathfrak{B}_{\mathfrak{M}}$ *est un ensemble non vide appelé l'*espace des états *et dont les éléments sont appelés* états,

3. $\Phi_{\mathfrak{M}}$ *est une action de monoïde de* $\mathbb{T}_{\mathfrak{M}}$ *sur* $\mathfrak{B}_{\mathfrak{M}}$ *appelée* fonction de transition.

L'ensemble des faits de $\mathfrak{M}$ *est donné par :*

$$E_{\mathfrak{M}} = \{(x, \delta, \Phi(x, \delta)) \mid x \in \mathfrak{B}_{\mathfrak{M}}, \delta \in \mathbb{T}_{\mathfrak{M}}\}$$

Dans cette définition, chaque triplet d'un modèle dynamique (x, δ, x') représente la transition du système de l'état x vers l'état x' après une durée δ. Nous remarquerons également que $E_{\mathfrak{M}}$ spécifie exactement la fonction $\Phi_{\mathfrak{M}}$: un modèle dynamique est donc également un modèle à observables privilégiées tel que décrit définition 2.3.3, les observables privilégiées étant ici le temps et les conditions initiales.

Exemple 2.3.5. Le modèle de croissance Malthusien décrit l'évolution de la taille d'une population suivant un taux de natalité r sans aucune autre restriction. Ce type de système présente une croissance exponentielle répondant à l'équation :

$$P(t) = P_0 e^{rt}$$

où t représente le temps, P_0, le nombre d'individus constituant initialement la population, et $P(t)$, la taille de la population après une période de t unités de temps. Cette caractérisation décrit très clairement un système dynamique tel que nous l'avons évoqué ci-dessus et peut être représentée par le modèle dynamique $\mathfrak{M}_{malthus}$ dont l'ensemble des faits est le suivant :

$$E_{\mathfrak{M}_{malthus}} = \{(P, t, Pe^{rt}) \mid P, t \in \mathbb{R}^+\}$$

Le modèle du temps $\langle \mathbb{R}^+, 0, + \rangle$ correspond à la modélisation continue classique utilisant les réels et l'addition standard. L'état du système est également représenté par un élément de $\mathbb{R}^+$ correspondant, cette fois-ci, à un nombre d'individus.

Modèle à base de champ

La notion de champ est particulièrement répandue en modélisation. Un champ associe une donnée à chaque point d'un espace. Lorsque nous parlons de champ, un rapprochement apparaît avec les champs scalaires ou vectoriels de la physique classique où les données représentent des quantités physiques, telles que la température ou la vitesse de déplacement d'un fluide, cependant, suivant les contextes, les champs prennent des formes différentes. D'un point de vue général, on parle de champ lorsque l'on est en présence d'une population d'objets de même nature, comme les cellules d'un automate cellulaire, les agents d'un système multi-agent, les ou les places d'un réseau de Petri, que l'on caractérise individuellement par un certain état, l'état

d'une cellule, l'état d'un agent, l'état d'une place, etc. Cette population décrit également une organisation mettant en relation les objets les uns par rapport aux autres. On parle alors d'un espace et des points de cette espace. Bien souvent, la valeur associée en chaque point est également structurée et on demande au champ de varier *continûment* sur cet espace, c'est-à-dire sans présenter de rupture par rapport à l'organisation des valeurs.

Mathématiquement, ces considérations sont affiliées aux notions d'espaces topologiques et de fonctions continues. Un *espace topologique* $\mathbb{S}$ est la donnée d'un ensemble de *points* $E_{\mathbb{S}}$, d'un ensemble d'*ouverts* $\Omega_{\mathbb{S}} \subset 2^{E_{\mathbb{S}}}$, contenant $E_{\mathbb{S}}$ et $\varnothing$, et clos par union arbitraire et par intersection finie. Une *fonction continue* d'un espace topologique vers un autre est une fonction entre les points qui respectent ces structures : la préimage d'un ouvert doit être un ouvert.

Ces définitions peuvent être interprétées en termes logiques. En tant que sous-ensembles, les ouverts peuvent être compris comme des prédicats offrant un vocabulaire pour classer les points les uns par rapport aux autres. L'union et l'intersection correspondent respectivement aux opérations de disjonction et de conjonction dans cette classification. La continuité impose que le vocabulaire de l'espace d'arrivée corresponde au vocabulaire de l'espace de départ. Ainsi, en comprenant un ouvert comme une variation autour d'un point suivant une certaine propriété, la préimage de cette variation sera une variation autour des préimages du point considéré. L'exemple du champ de température illustre cette idée de variation : considérons l'espace euclidien comme espace de départ et les réels comme espace d'arrivée avec leurs topologies usuelles, c'est-à-dire où une variation autour d'une valeur revient à prendre un intervalle ouvert autour de cette valeur[7]. Ainsi, pour un champ de température donné, si l'on considère l'ensemble P de tous les points de température 20°C, les points de températures comprises entre 19°C et 21°C seront « autour » de ceux de P. Cette idée s'applique également pour un réseau d'interactions entre agents : plus deux agents sont dans des états « proches », c'est-à-dire partageant de plus en plus de propriétés, plus leurs environnements d'interactions seront comparables.

Nous proposons de caractériser les *modèles à base de champ* comme la représentation d'une fonction continue.

Définition 2.3.5 (Modèle à base de champs)**.** *Un* modèle à base de champ *est un modèle* $\mathfrak{M}$ *caractérisé par un triplet* $\langle \mathbb{S}_{\mathfrak{M}}, \mathbb{V}_{\mathfrak{M}}, f_{\mathfrak{M}} \rangle$ *tel que*

1. $\mathbb{S}_{\mathfrak{M}}$ *et* $\mathbb{V}_{\mathfrak{M}}$ *sont des espaces topologiques, et*
2. $f_{\mathfrak{M}} : \mathbb{S}_{\mathfrak{M}} \to \mathbb{V}_{\mathfrak{M}}$ *est une fonction continue.*

L'ensemble des faits de $\mathfrak{M}$ *est donné par :*

$$E_{\mathfrak{M}} = \{(x, f(x)) \mid x \in \mathbb{S}_{\mathfrak{M}}\}$$

L'avantage du point de vue topologique est qu'il autorise tout type de champ. En effet, n'importe quelle fonction entre deux ensembles pourra être rendue continue en utilisant par

7. Pour être exact, il s'agit ici d'une base de la topologie usuelle.

exemple la topologie discrète[8] sur l'espace de départ. La qualité d'un modèle à base de champ viendra du choix judicieux des topologies utilisées. Nous remarquerons enfin que la définition fonctionnelle de $E_{\mathfrak{M}}$ permet d'inclure les modèles à base de champs comme car particulier de modèles à variables privilégiées.

Exemple 2.3.6. En théorie des réseaux sociaux, la séparation, c'est-à-dire la longueur de la plus petite chaîne de connaissances sociales permettant de relier un individu à un autre, est une mesure importante. Elle est par exemple à la base de la théorie des *six degrés de séparation* [43] établissant que toute personne sur la planète est reliée socialement à n'importe quelle autre par une chaîne de longueur au plus 6.

Il est possible de modéliser la séparation comme un champ de distance sur le graphe des liens sociaux. Pour un graphe donné G, le champ de distance à un sommet v_0 associe à chaque sommet v de G la longueur $d(v)$ du plus court chemin entre v_0 et v. Le champ d se présente donc comme une fonction des sommets de G dans $\mathbb{N}$.

Il est possible de considérer la continuité de d en associant au graphe G et à $\mathbb{N}$ leur *topologie digitale*. Pour cette topologie, notée $\Omega_G^{\text{dig.}}$, un ouvert d'un graphe G (vu comme un ensemble de sommets et d'arcs) est défini comme une union d'ouverts de base :

- pour tout arc e, le singleton $\{e\}$ est un ouvert ;
- pour tout sommet v, l'ensemble $\{v\} \cup \{e \in G \mid v \prec e\}$, où $v \prec e$ signifie que e est incident à v dans G, est un ouvert.

Les entiers peuvent être munis de la même topologie en considérant $\mathbb{N}$ comme un graphe dont les sommets sont les entiers et où un arc $\{n, n+1\}$ relie chaque entier n à son successeur.

On peut remarquer que pour toute paire de sommets voisins (v_1, v_2), le champ de distance ne varie qu'au plus de 1 :

$$|d(v_1) - d(v_2)| \leq 1$$

La fonction d peut alors être étendue aux arcs de telle sorte que pour tout arc e entre v_1 et v_2,

$$d(e) = \begin{cases} d(v_1) & \text{si } d(v_1) = d(v_2) \\ \{d(v_1), d(v_2)\} & \text{sinon} \end{cases}$$

Ainsi, la fonction d est continue et peut être utilisée pour constituer un modèle à base de champ $\mathfrak{M}_{sép.}$ de la séparation d'un réseau par :

$$\mathbb{S}_{\mathfrak{M}_{sép.}} = \langle G, \Omega_G^{\text{dig.}} \rangle \qquad \mathbb{V}_{\mathfrak{M}_{sép.}} = \langle \mathbb{N}, \Omega_{\mathbb{N}}^{\text{dig.}} \rangle \qquad f_{\mathfrak{M}_{sép.}} = d$$

Modèle à structure dynamique

Les classes de modèles que nous avons proposées jusqu'ici ne sont pas indépendantes les unes des autres. Comme nous l'avons déjà vu par exemple, les modèles dynamiques et à base

8. La topologie discrète est celle où tout sous-ensemble de points est un ouvert. Il ne faut cependant pas la comprendre comme **la** topologie des modèles discrets.

de champ sont des cas particuliers de modèles à observables privilégiées. Ces deux classes s'intéressent en fait à deux aspects qui reviennent fréquemment dans les modélisations, le *temps* et l'*espace*, et il est classique de retrouver ces deux aspects en même temps au sein du même modèle. L'intersection entre les modèles dynamiques et les modèles à base de champ n'est pas vide, loin s'en faut.

Il est intéressant de noter que l'interdépendance entre temps et espace nous amène en fait à considérer quatre types de modélisation à variables privilégiées spatio-temporelles :

1. $\mathbb{S} \times \mathbb{T} \rightarrow ...$
   L'espace ne varie pas dans le temps et le temps est le même partout dans l'espace. Ainsi, *temps* et *espace* sont indépendants. On retrouve par exemple ce type de modèles en physique classique ou avec les automates cellulaires.

2. $\mathbb{S} \rightarrow (\mathbb{T} \rightarrow ...)$ Le temps varie d'un point de l'espace à un autre. Autrement dit, chaque point à son horloge. Ce cas est représentable par un modèle à base de champs qui associe à chaque point de l'espace un modèle dynamique. Par exemple, une étude de charge dans une grille de calcul amènera à ce type de modélisation : les processeurs tournent chacun à leur propre vitesse alors que la structure du réseau reste fixe.

3. $\mathbb{T} \rightarrow (\mathbb{S} \rightarrow ...)$ L'espace varie au cours du temps : à chaque instant, le domaine spatial est différent. Une représentation reposant sur un modèle dynamique dont l'espace des états sera constitué de modèles à base de champ, sera privilégiée. Nous appelons ces systèmes, des systèmes dynamiques à structure dynamique.

4. $\mathbb{L} \rightarrow ...$
   L'espace et le temps forment ici un tout, $\mathbb{L}$, qui ne peut être dissocié. En physique, la relativité est une parfaite illustration de l'étude de ce type de système. Bien qu'il soit possible de n'observer que l'action du temps sur un point de l'espace en suivant les lignes d'univers, la description qui en résulte n'est pas complète. En effet, un changement d'observation, c'est-à-dire le passage d'une ligne d'univers à une autre, modifie la perception du temps *et* de l'espace sur les autres événements.

Modèle probabiliste

Nous terminons la description de ces quelques classes de modèles en évoquant les systèmes que nous souhaitons modéliser de façon probabiliste. Pour ces modélisations, le formalisme repose essentiellement sur les notions d'espace mesurable et de probabilité. Formellement, un espace mesurable $\mathbb{X}$ est un ensemble $X_{\mathbb{X}}$ muni d'une *tribu* $\mathscr{A}_{\mathbb{X}}$, c'est-à-dire d'un ensemble de sous-ensembles de $X_{\mathbb{X}}$ contenant $X_{\mathbb{X}}$, clos par complémentation et par union dénombrable. Dans le cadre de la théorie de la probabilité, $X_{\mathbb{X}}$ est appelé un *univers*, représentant l'ensemble de toutes les résultats d'une expérience donnée, et $\mathscr{A}_{\mathbb{X}}$ l'ensemble des événements. Une mesure de probabilité $\mathbb{P}$ associe à chaque événement sa probabilité, c'est-à-dire une valeur de $[0, 1]$ telle que $\mathbb{P}(X_{\mathbb{X}}) = 1$ et $\mathbb{P}(\cup_i A_i) = \sum_i \mathbb{P}(A_i)$ pour toutes familles dénombrables d'événements disjoints $\{A_i\}$.

Un modèle peut être la représentation d'une expérience probabiliste : nous parlerons ici de *modèle probabiliste.*

Définition 2.3.6 (Modèle probabiliste)**.** *Un* modèle probabiliste *est un modèle* $\mathfrak{M}$ *caractérisé par un couple* $\langle \mathbb{X}_{\mathfrak{M}}, \mathbb{P}_{\mathfrak{M}} \rangle$ *tel que*

1. $\mathbb{X}_{\mathfrak{M}} = \langle X_{\mathfrak{M}}, \mathscr{A}_{\mathfrak{M}} \rangle$ *est un espace mesurable, et*
2. $\mathbb{P}_{\mathfrak{M}}$ *est une probabilité sur* $\mathbb{X}_{\mathfrak{M}}$.

L'ensemble des faits de $\mathfrak{M}$ *est donné par :*

$$E_{\mathfrak{M}} = \{(A, \mathbb{P}_{\mathfrak{M}}(A)) \mid A \in \mathscr{A}_{\mathfrak{M}}\}$$

On remarquera que cette définition décrit les modèles probabilistes comme des modèles à observables privilégiées particuliers.

Exemple 2.3.7. La croissance de Malthus décrite exemple 2.3.5 peut être modélisée par un processus aléatoire reposant sur une simple règle de reproduction, semblable à une division cellulaire :

$$X \xrightarrow{r} 2X \tag{2.2}$$

où X représente un individu et r est le taux de division des individus. Étant donnée une population composée de N_0 individus, nous cherchons à décrire le temps d'attente avant la prochaine division.

On suppose pour cela que deux divisions ne peuvent être simultanées et que les individus se divisent de façon indépendante. À partir de la règle (2.2), nous posons la probabilité infinitésimale $r\,d\tau$ qu'un individu se divise à l'instant τ, c'est-à-dire durant l'intervalle de temps infinitésimal $[\tau, \tau+d\tau]$. Ces observations nous permettent de déduire les probabilités suivantes :

- $rN\,d\tau$, la probabilité infinitésimale d'une division pour une population de N individus (indépendants par hypothèse) à l'instant τ,
- $e^{-rN_0\tau}$, la probabilité qu'aucune division n'ait lieu durant l'intervalle $[0, \tau]$, et
- $N_0 r e^{-rN_0\tau} d\tau$, la probabilité infinitésimale que la prochaine division ait lieu à l'instant τ.

Cette dernière expression nous amène à considérer la probabilité que la prochaine division se produise durant un certain intervalle de temps, posant ainsi les termes d'un modèle probabiliste $\mathfrak{M}_{div.}$ de la croissance Mathusienne. Pour cela, considérons d'une part l'espace mesurable $\mathbb{X}_{\mathfrak{M}_{div.}}$ comme étant l'ensemble des réels positifs $\mathbb{R}^+$ muni de sa tribu Borélienne classique, c'est-à-dire engendrée par les intervalles fermés $[t_1, t_2]$, et d'autre part la mesure de probabilité :

$$\mathbb{P}_{\mathfrak{M}_{div.}}([t_1, t_2]) = \int_{t_1}^{t^2} rN_0 e^{-rN_0\tau} d\tau = [-e^{-rN_0\tau}]_{t_1}^{t_2} = e^{-rN_0 t_1} - e^{-rN_0 t_2}$$

L'ensemble des faits de $\mathfrak{M}_{div.}$ est alors donné par :

$$E_{\mathfrak{M}_{div.}} = \{([t_1, t_2], \mathbb{P}_{\mathfrak{M}_{div.}}([t_1, t_2])) \mid t_1, t_2 \in \mathbb{R}^+\}$$

2.3.3 Modélisations d'un système proie-prédateur

Nous proposons dans cette dernière section d'illustrer comment les différentes classes de modèles que nous venons d'énumérer permettent plusieurs compréhensions d'un même système. Nous nous intéressons en particulier à la modélisation des systèmes proie-prédateur.

Un système proie-prédateur est un système dans lequel deux populations, celles des proies et celle des prédateurs, varient à cause de leurs interactions, dont la principale est la prédation. Voici deux exemples de ces systèmes :

- La variation du nombre d'individus dans les populations de lynx et de lièvres des neiges, dont un modèle expérimental a été constitué par la Compagnie de la Baie d'Hudson [32].
- La variation du nombre d'individus dans les populations de loups et d'élans, dont un modèle expérimental a été constitué par le parc national d'Isle Royale [34].

En suivant au cours du temps les tailles des deux populations, il est possible d'observer des oscillations couplées plus ou moins régulières. À certaines périodes, un déficit en prédateurs permet aux proies de se développer, alors qu'à d'autres périodes les prédateurs dominent et les proies se font rares.

Ces oscillations sont explicables en considérant le comportement des proies et des prédateurs à partir des trois processus suivants :

Naissance : Les proies se reproduisent en fonction des ressources disponibles, des individus apparaissent avec un certain taux de natalité.

Mort : Les proies et les prédateurs n'étant pas éternels, les individus disparaissent avec un certain taux de mortalité.

Prédation : Il s'agit de l'activité principale des prédateurs et la seule interaction considérée entre les deux populations ; elle est nécessaire à la reproduction des prédateurs et induit un facteur de mortalité supplémentaire pour les proies.

En 1920, A. J. Lotka propose dans [23] une description formelle d'un tel modèle à travers un système de réactions autocatalytiques, à la façon d'un jeu de réaction chimique. L'étude de ce type de systèmes l'amena quelques années plus tard à proposer une analyse de la dynamique de ce système sous la forme d'un système d'équations différentielles couplées [24]. Le même type d'équations furent également proposées indépendamment par V. Volterra à la même période [45]. Depuis, plusieurs autres modèles plus élaborés ont été proposés pour prendre en considération d'autres facteurs environnementaux, ou encore l'incidence de l'organisation spatiale des populations.

Les études des systèmes proies-prédateurs nous offrent ainsi un échantillon de modèles nous permettant d'illustrer notre approche. Nous allons dans la suite de cette section décrire quelques modèles de systèmes proie-prédateur repris des travaux de A.J. McKane [29, 26].

Modèle *à la* Volterra. Dans cette description, nous considérons que le système est décrit par deux observables représentant les densités de population des proies et des prédateurs. En notant, U_V et U_P respectivement ces densités, les trajectoires du système au cours du temps sont décrites par le système d'équations différentielles suivant :

$$\left\{ \begin{array}{rcl} \dfrac{dU_V}{dt} & = & rU_V(1 - \dfrac{U_V}{K}) - gU_PU_V \\ \dfrac{dU_P}{dt} & = & nU_VU_P - \mu U_P \end{array} \right. \tag{2.3}$$

Ce modèle dépend des paramètres r, K, g, n et μ. Comme précisé dans [29], on rencontre souvent sous le nom de modèle de Lotka-Volterra ce même système où le terme U_V/K est absent. Le coefficient K quantifie la capacité limite de l'environnent à accueillir des individus.

Cette formalisation permet de constituer un modèle $\mathfrak{M}_V$ *dynamique à observables*. On définit la signature du modèle par :

$$\Sigma_{\mathfrak{M}_V} = \mathbb{D}_{\texttt{Initial}} \times \mathbb{D}_{\texttt{Temps}} \times \mathbb{D}_{\texttt{Etat}}$$

où $\mathbb{D}_{\texttt{Initial}} = \mathbb{D}_{\texttt{Etat}} = \mathbb{R}^+ \times \mathbb{R}^+$ correspondant à des couples *(densité de proie, densité de prédateur)* et $\mathbb{D}_{\texttt{Temps}} = \mathbb{R}^+$ avec sa structure de monoïde usuelle représentant des durées. L'ensemble des faits est donné comme un sous-ensemble de $\Sigma_{\mathfrak{M}_V}$ par :

$$E_{\mathfrak{M}_V} = \{\, (U_V(0), U_P(0), t, U_V(t), U_P(t)) \mid U_V, U_P \text{ sol. de (2.3)} \,\} \subset \Sigma_{\mathfrak{M}_V}$$

Une *solution de (2.3)* correspond à tout couple de fonctions du temps vérifiant l'équation ; toutes les conditions possibles sont prises en compte. Nous remarquons de plus que la résolution de l'équation se fait une fois les paramètres fixés. Ainsi, à chaque jeu de paramètres correspondra un modèle.

En tant que modèle dynamique, on peut extraire de la définition de $E_{\mathfrak{M}_V}$ une action de monoïde donnée par :

$$\Phi_{\mathfrak{M}_V}([U_V(0), U_P(0)], t) = [U_V(t), U_P(t)]$$

pour chaque élément de $E_{\mathfrak{M}_V}$.

Modèle *à la* Volterra spatialisé. Dans sa version spatialisée, les densités de population sont distribués sur un espace particulier, disons l'espace euclidien de dimension 2 $\mathbb{E}^2$, représentant la région dans laquelle les individus peuvent se déplacer. En d'autres termes, les densités U_V et U_P deviennent des champs, respectivement u_V et u_P, définis sur $\mathbb{E}^2$. Les équations sont modifiées en ajoutant un terme de diffusion modélisant ces déplacements :

$$\left\{ \begin{array}{rcl} \dfrac{du_V}{dt} & = & ru_V(1 - \dfrac{u_V}{K}) - gu_Pu_V + D_V(\nabla^2 u_V + u_V\nabla^2 u_P - u_P\nabla^2 u_V) \\ \dfrac{du_P}{dt} & = & nu_Vu_P - \mu u_P + D_P(\nabla^2 u_P + u_P\nabla^2 u_V - u_V\nabla^2 u_P) \end{array} \right. \tag{2.4}$$

où D_V et D_P sont les coefficients de diffusion respectifs des populations de proies et de prédateurs, et ∇^2 dénote le Laplacien. Suivant les remarques de [26], les termes exprimant une

diffusion croisée en $(u_V \nabla^2 u_P - u_P \nabla^2 u_V)$ entre les deux populations sont peu conventionnels mais correspondent à la considération d'une limitation de la capacité d'accueil de l'environnement. Ces termes disparaissent lorsque cette limite est levée.

Il est possible d'associer à ce nouveau système d'équations une extension du modèle non spatialisé $\mathfrak{M}_V$, un modèle $\mathfrak{M}_{SV}$ *dynamique à observables et à base champs*. L'ensemble des faits du modèle se construit de la même façon que précédemment en prenant en compte l'espace.

$$E_{\mathfrak{M}_{SV}} = \{\, (t,x,y,u_V(0,x,y),u_P(0,x,y),u_V(t,x,y),u_P(t,x,y)) \mid u_V, u_P \text{ sol. de (2.4)} \,\}$$

En tant que modèle dynamique, on peut extraire de cette définition une action de monoïde donnée par :

$$\Phi_{\mathfrak{M}_{SV}}([(x,y) \mapsto u_V(0,x,y),(x,y) \mapsto u_P(0,x,y)],t) = [(x,y) \mapsto u_V(t,x,y),(x,y) \mapsto u_P(t,x,y)]$$

agissant sur l'ensemble des fonctions deux fois différentiables de $\mathbb{E}^2$ dans $\mathbb{R}^+ \times \mathbb{R}^+$, pour chaque élément de $E_{\mathfrak{M}_{SV}}$.

En tant que modèle à base de champs, une fonction continue donnée par :

$$f_{\mathfrak{M}_{SV}}(t,x,y) = (u_V(0,x,y),u_P(0,x,y),u_V(t,x,y),u_P(t,x,y))$$

peut être extraite de la définition de $E_{\mathfrak{M}_{SV}}$ en munissant l'espace de départ $\mathbb{E}^2 \times \mathbb{R}^+$ et l'espace d'arrivée $\mathbb{R}^+ \times \mathbb{R}^+ \times \mathbb{R}^+ \times \mathbb{R}^+$, de leurs topologies habituelles. Une autre approche mettant en avant l'indépendance temps/espace telle que nous l'avons décrite plus haut, est donnée par la définition suivante :

$$f_{\mathfrak{M}_{SV}}(x,y) = ([u_V(0,x,y),u_P(0,x,y)],t) \mapsto [u_V(t,x,y),u_P(t,x,y)]$$

allant de $\mathbb{E}^2$ muni de sa topologie classique vers un espace d'actions de monoïde. La topologie de cet espace est héritée de celle sur $\mathbb{E}^2$.

Modèle *à la* Lotka. Cette approche, radicalement différente des précédentes, est fondée sur les premières descriptions de A. J. Lotka à base de réactions chimiques. On assimile le système à une solution chimique de taille fixe composée de N molécules prises parmi trois types d'éléments différents : V représentant une proie, P pour un prédateur, et E une place vide. Les interactions entre les trois espèces chimiques sont régies par les réactions suivantes :

$$\begin{array}{ccc} & V + E \xrightarrow{b} 2\,V & \\ P + V \xrightarrow{p_1} 2\,P & & P + V \xrightarrow{p_2} P + E \\ V \xrightarrow{d_V} E & & P \xrightarrow{d_P} E \end{array} \tag{2.5}$$

On reconnaît facilement les trois processus décrits en début de section : la naissance d'une proie représentée comme une division cellulaire consommant une place vide, la prédation qui occasionne certaine fois la reproduction d'un prédateur, et la mort des deux types d'individus

donnant de nouvelles places vides. Chaque réaction est paramétrée par une constante, appelée *constante stochastique*[9]. Cette constante représente le taux avec lequel le processus correspondant a lieu. En terme de probabilité, $d_V\ d\tau$ est par exemple la probabilité infinitésimale qu'une proie meurt de sa belle mort (c'est-à-dire sans avoir été chassée) pendant l'intervalle de temps infinitésimal $d\tau$.

D. T. Gillespie a proposé dans [15] une méthode de simulation stochastique exacte de système de réactions chimiques sous hypothèse d'homogénéité. Nous pouvons ainsi considérer un modèle *dynamique probabiliste* des systèmes proie-prédateur $\mathfrak{M}_L$ à partir des trajectoires simulées. De façon informelle, l'ensemble des faits correspond à toutes ces trajectoires. Chacune de ces trajectoires peut être pondérée par une probabilité d'occurrence au sein des trajectoires partageant les mêmes conditions initiales. Ainsi, un sous modèle probabiliste au sens de la définition 2.3.6 peut être établi pour chaque population initiale.

Nous préférons ici mettre l'accent sur les aspects temporels du modèle. Ainsi, nous établissons un ensemble de faits, équivalent à celui que nous venons d'évoquer. Pour se faire, nous considérons d'une part la probabilité $P(n_V, n_P, t | n_V^0, n_P^0)$ que *le système soit composé de n_V proies et n_P prédateurs au temps t considérant une population initiale de n_V^0 proies et n_P^0 prédateurs*, et d'autre part, le domaine *fini* $\mathbb{D}_{\mathsf{Pop}}$ des populations possibles :

$$\mathbb{D}_{\mathsf{Pop}} = \{\ (n_V, n_P) \in \mathbb{N}^2 \mid n_V + n_P \leq N\ \}$$

Considérant que les densités de probabilité sur $\mathbb{D}_{\mathsf{Pop}}$ sont des fonctions $d : \mathbb{D}_{\mathsf{Pop}} \to [0,1]$ telles que

$$\sum_{s \in \mathbb{D}_{\mathsf{Pop}}} d(s) = 1,$$

l'action de monoïde[10] de $\mathbb{R}^+$ est défini sur les densités de probabilité par :

$$\Phi_{\mathfrak{M}_L}(d,t) = s \mapsto \sum_{s' \in \mathbb{D}_{\mathsf{Pop}}} d(s') P(s,t|s')$$

pour finalement obtenir la description de $\mathfrak{M}_L$ comme un modèle dynamique.

Modèle *à la* Lotka spatialisé. L'objet principal présenté dans [26] est une version spatialisée du modèle à base de réactions chimiques $\mathfrak{M}_L$ que nous venons de voir. L'extension consiste en deux points :

1. Le modèle $\mathfrak{M}_L$ est copié sur chaque position d'une organisation spatiale discrète Ω. Afin de se rapprocher du modèle $\mathfrak{M}_{SV}$, on définit Ω comme un ensemble d'indices permettant

9. On reprend ici le vocabulaire de D.T. Gillespie qui discute dans [15] de la différence entre constante stochastique et constante cinétique.

10. On montre facilement qu'il s'agit d'une action de monoïde en remarquant que

$$P(s',0|s) = \delta_s^{s'} \qquad P(s,t_1+t_2|s'') = \sum_{s' \in \mathbb{D}_{\mathsf{Pop}}} P(s,t_2|s') P(s',t_1|s'')$$

d'identifier les positions d'une grille carrée à 2 dimension. Chaque molécule est alors positionnée : V^i, P^i et E^i correspondent respectivement à une proie, un prédateur et une place libre en position $i \in \Omega$.

2. Les réactions chimiques de $\mathfrak{M}_L$ sont également positionnées sur Ω, et deux règles de déplacement sont ajoutées pour simuler le mouvement des individus :

$$\begin{array}{cc} V^i + E^i \xrightarrow{b} 2\,V^i & \\ P^i + V^i \xrightarrow{p_1} 2\,P^i & P^i + V^i \xrightarrow{p_2} P^i + E^i \\ V^i \xrightarrow{d_V} E^i & P^i \xrightarrow{d_P} E^i \\ V^i + E^j \xrightarrow{m_V} E^i + V^j & P^i + E^j \xrightarrow{m_P} E^i + P^j \end{array} \tag{2.6}$$

pour toutes positions *voisines* $i, j \in \Omega$.

Cette extension formalise un nouveau modèle $\mathfrak{M}_{SL}$ qui est à la fois *probabiliste, dynamique et à base de champ.*

Nous n'entrerons pas dans les détails de ces définitions qui restent très proches des constructions que nous avons déjà vues. Cependant, nous pouvons en conclure que les modèles $\mathfrak{M}_V$, $\mathfrak{M}_{SV}$, $\mathfrak{M}_L$ et $\mathfrak{M}_{SL}$ sont proches les uns des autres. Dans [29, 26] se trouvent d'ailleurs les clés de ces relations, expliquant comment établir des ponts formels entre les modèles. Ainsi, on constatera que $\mathfrak{M}_V$ décrit le comportement moyen des trajectoires de $\mathfrak{M}_L$, alors que $\mathfrak{M}_{SV}$ coïncide avec $\mathfrak{M}_V$ lorsque les densités de populations sont spatialement homogènes.

Dans la suite, nous proposons un cadre formel pour décrire ces relations. Notre construction se fonde sur la notion d'abstraction telle que celle que nous avons de mentionné rapidement entre $\mathfrak{M}_V$ et $\mathfrak{M}_L$, ou entre $\mathfrak{M}_V$ et $\mathfrak{M}_{SV}$. L'abstraction autorise également la composition de modèles pour obtenir des descriptions plus précises couplant des travaux initialement indépendants. Ainsi, on pourra poser la question d'un modèle permettant de coupler à la fois $\mathfrak{M}_{SV}$ et $\mathfrak{M}_L$, combinant d'un part une représentation stochastique des interactions individuelles, et la dynamique de la répartition spatiale des populations d'autre part. On pourra également s'intéresser aux relations entre ce couplage et la description proposée par $\mathfrak{M}_{SL}$. La Fig. 2.11 synthétise cette situation.

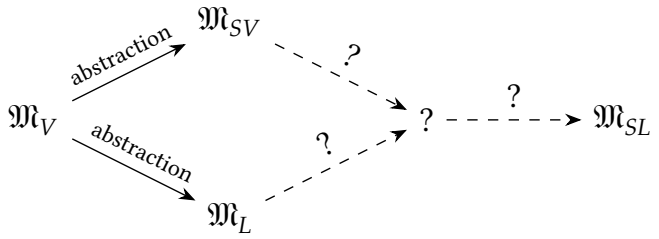


Fig. 2.11: Comment définir le couplage des modèles $\mathfrak{M}_{SV}$ et $\mathfrak{M}_L$? Quel rapport existe-t-il entre ce couplage et $\mathfrak{M}_{SL}$?

2.4 Composition des modèles

Les sections 2.2 et 2.3.2 nous ont amené à voir un modèle comme un ensemble de faits. De ce point de vue, la comparaison entre modèles est limitée à l'étude de leurs propriétés ensemblistes. Comme nous l'avons vu section 2.2, l'abstraction entre modèles est un élément clé de la modélisation multi-niveau. Nous proposons ici d'en faire une interprétation ensembliste permettant de raffiner et coupler différents modèles d'un même système. Ainsi, le passage d'un modèle $\mathfrak{M}_+$ à un modèle $\mathfrak{M}_-$ s'effectue à travers la reconnaissance d'une *flèche d'abstraction* entre ces modèles définie de la façon suivante : *si* $\mathfrak{M}_+$ *est une abstraction de* $\mathfrak{M}_-$*, alors chaque fait de* $\mathfrak{M}_-$ *peut être associé à un fait de* $\mathfrak{M}_+$. L'étude des relations entre les modèles proie/prédateur de la section 2.3.3 illustre ce type de construction ; les trajectoires stochastiques du modèle probabiliste sont associées aux trajectoires moyennes du modèle déterministe.

Cette description nous amène à considérer l'ensemble des fonctions de $\mathfrak{M}_-$ vers $\mathfrak{M}_+$, chacune comprise comme une relation d'abstraction possible entre les deux modèles. Cependant, il est raisonnable de se demander si toutes ces fonctions sont éligibles au titre d'abstraction. Ce n'est définitivement pas le cas au regard de l'exemple des modèles de proie/prédateur. La construction de la flèche d'abstraction repose sur des techniques mathématiques précises qui assurent une cohérence entre les trajectoires stochastiques et les trajectoires déterministes. Cette cohérence vient du fait que les deux modèles développent deux compréhensions différentes du *même* système, et une fonction arbitrairement choisie entre les ensembles de faits de $\mathfrak{M}_-$ et de $\mathfrak{M}_+$ a peu de chance de respecter ce *rapport au système*. Comparer deux modèles d'un système proie-prédateur a du sens, justement car ce sont deux modèles du même système, ils parlent de la même « chose » : les faits de $\mathfrak{M}_-$ concernant les proies sont envoyés vers des faits de $\mathfrak{M}_+$ concernant les proies, les prédateurs de $\mathfrak{M}_-$ vers les prédateurs de $\mathfrak{M}_+$, le temps de $\mathfrak{M}_-$ vers le temps de $\mathfrak{M}_+$, l'espace de $\mathfrak{M}_-$ vers l'espace de $\mathfrak{M}_+$, etc. Il est raisonnable donc d'attendre que les flèches d'abstraction soient les fonctions de $\mathfrak{M}_-$ vers $\mathfrak{M}_+$ qui respectent la *sémantique* imposée par le système.

En se référant aux notations précédentes, il est clair qu'un modèle $\mathfrak{M}$ ne peut être restreint qu'à son seul ensemble de faits $E_{\mathfrak{M}}$. Il est nécessaire de lui associer également une sémantique $\sigma_{\mathfrak{M}}$. La question qui nous est alors posée est comment formaliser cette sémantique. Nous développons ici une proposition fondée sur la définition usuelle des modèles vus comme des *abstractions* du système. La sémantique $\sigma_{\mathfrak{M}}$ serait alors vu comme une flèche d'abstraction au sens que nous venons d'évoquer. L'idée générale consiste donc, pour un système S donné, de considérer un *modèle de référence* $\mathfrak{M}_S$ auquel tout autre modèle $\mathfrak{M}$ de S se rapportera par une fonction $\sigma_{\mathfrak{M}}$ allant de $E_{\mathfrak{M}_S}$ vers $E_{\mathfrak{M}}$.

Dans cette section, nous proposons d'utiliser la théorie des catégories pour élaborer un cadre formel suffisant à la réalisation de cette proposition. Pour plus de clarté, nous commençons par une instanciation informelle dans le cadre des sciences expérimentales. Nous introduisons ensuite les éléments de la théorie des catégories nécessaires à la compréhension de notre formalisation avec laquelle nous clôturerons le chapitre.

2.4.1 Application en sciences expérimentales

La méthodologie que nous proposons est adaptée au cadre des sciences expérimentales où le rapport entre un modèle et le système qu'il décrit est donné par l'expérience. En effet, le contexte expérimental nous invite à considérer comme modèle de référence d'un système S, l'ensemble de tous les résultats d'expériences possibles sur S. Tout modèle sera alors *validé* s'il est en accord avec l'expérience, c'est-à-dire lorsque les faits expérimentaux seront associés aux faits que le modèle décrit. La Fig. 2.12 illustre la situation : le modèle $\mathfrak{M}$ se rapporte au système S *via* le modèle expérimental $\mathfrak{M}_S$. Ce lien est noté ici par une flèche d'abstraction qui correspond à la validation du modèle $\mathfrak{M}$ faisant appel à une démarche expérimentale usuelle :

1. Collecter des données par des mesures, des expériences : peupler $\mathfrak{M}_S$
2. Proposer un modèle permettant de rendre compte de ces mesures : construire le modèle $\mathfrak{M}$
3. *Valider* les prédictions du modèles en les comparant aux mesures directement prises du système étudié : établir la fonction $\sigma_{\mathfrak{M}_S}$ de $\mathfrak{M}_S$ vers $\mathfrak{M}$.

Nous noterons en particulier dans cette figure la frontière claire, représentée en pointillés, entre les modèles et le système : le système S n'est pas un modèle et les flèches d'abstraction et notamment les validations, ne relient que des modèles. On montre ici que la modélisation dépend de la qualité des expérimentations (par exemple de la précision des outils de mesure utilisés) et peut être réévaluée (soit en rejetant un modèle, soit en l'approuvant plus fortement encore) avec l'évolution des méthodes expérimentales, c'est-à-dire en reconsidérant la définition de $\mathfrak{M}_S$.

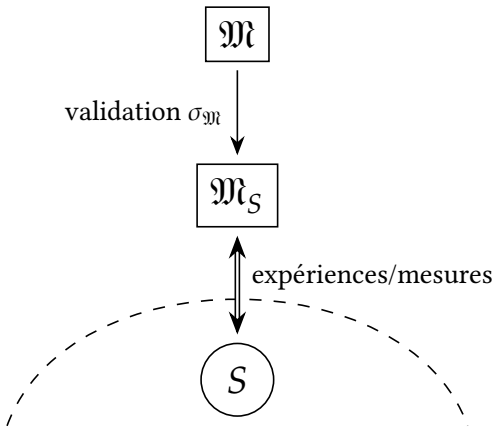


Fig. 2.12: Représentation schématique de la relation entre un modèle $\mathfrak{M}$ et un système S où $\mathfrak{M}_S$ est le modèle de référence.

Pour illustrer nos propos, considérons le système expérimental S suivant : un ballon de baudruche gonflé à l'azote est attaché au fond d'une cuve d'eau maintenue à 20 °C, entièrement

immergé. On cherche à comprendre la force de traction qu'exerce le ballon à son point de fixation.

On remarque après quelques tests que cette force dépend du volume V_e qu'occupe le ballon, sachant que celui-ci éclatera au-delà d'un volume maximal $V_{\max}$.

Deux modélisateurs s'attaquent au problème et cherchent à établir une loi reliant force et volume. Après avoir longuement considéré les données récoltées, ils proposent les modèles suivants :

Modèle $\mathfrak{M}_1$: le premier modélisateur établit que *la relation entre la force et le volume repose sur un rapport de proportionnalité* en accord avec la poussée d'Archimède et propose le modèle équationnel :

$$E_{\mathfrak{M}_1} = \{\, (F, V) \in \mathbb{D}_{\texttt{Traction}} \times \mathbb{D}_{\texttt{Volume}} \mid F = c_0 V \} \uplus \{\bot, \textit{éclaté}\,\}$$

à deux observables `Traction` (en Newton dans $\mathbb{D}_{\texttt{Traction}} = \mathbb{R}^+$) représentant la force de traction verticale et orientée vers le haut au point de fixation P du ballon, et `Volume` (en mètre cube dans $\mathbb{D}_{\texttt{Volume}} = \mathbb{R}^+$), le volume de fluide déplacé. Le coefficient de proportionnalité $c_0 = \rho_E g$ est donné par $\rho_E = 1{,}0\,\mathrm{kg\,m^{-3}}$, la masse volumique de l'eau, et $g = 9{,}81\,\mathrm{N\,kg^{-1}}$, l'accélération de la pesanteur terrestre.

La validation de ce modèle passe par une comparaison entre les mesures faites expérimentalement et les résultats théoriques apportés par la loi proposée. Il s'avère que dans une certaine marge, à savoir lorsque le volume V déplacé est supérieure à un seuil V_a (c'est-à-dire lorsque le ballon est suffisamment gonflé mais qu'il n'a pas encore éclaté), les calculs produisent des résultats très proches de l'observation, l'écart pouvant être attribué à l'appareil de mesure[11]. En dehors de cette marge, l'équation proposée est fausse, et l'on peut se reporter sur le symbole *éclaté* pour les volumes $V > V_{\max}$, ou sur le symbole $\bot$ représentant l'absence d'explication par le modèle $\mathfrak{M}_1$ lorsque $V < V_a$. Ainsi la sémantique de $\mathfrak{M}_1$ revient à associer à une mesure expérimentale V_e du volume du ballon, le calcul théorique lorsque l'on est dans l'intervalle d'acceptation $[V_a, V_{\max}]$ du modèle :

$$\sigma_{\mathfrak{M}_1}(V_e) = \begin{cases} \bot & \text{si } V_e < V_a \\ (c_0 V_e, V_e) & \text{si } V_a \leq V_e < V_{\max} \\ \textit{éclaté} & \text{si } V_e = \textit{éclaté} \end{cases}$$

Modèle $\mathfrak{M}_2$: Dans ce second modèle décrivant le système de manière qualitative, les observables sont données empiriquement par des valeurs discrètes symboliques :

$$E_{\mathfrak{M}_2} = \{\, (\textit{faible, petit}), (\textit{faible, moyen}), (\textit{forte, moyen}), (\textit{forte, gros}), \textit{éclaté}\,\}$$

La sémantique de ce second modèle, moins précis, repose simplement sur une observation visuelle du ballon. Pour ce qui est de la force exercée, lorsque le nœud du ballon

11. Plus précisément, la différence observée peut être expliquée par la non prise en compte dans $\mathfrak{M}_1$ du poids du ballon et de l'azote qu'il contient. On peut en effet rajuster la force s'exerçant au point de contact par l'équation $F = (c_0 - c_1)V - c_2$ où $c_1 = \rho_N g$ avec $\rho_N = 1{,}25\,\mathrm{g\,L^{-1}}$ à la masse volumique de l'azote, et $c_2 = m_B g$ avec m_B la masse du ballon (de l'ordre d'une dizaine de grammes par exemple).

semble tendu, la traction est considérée *forte*, sinon *faible*. La taille du ballon est appréciée sur une échelle à 3 niveaux (*petit*, *moyen*, *gros*), sans compté l'état *éclaté*. En définitif, les mesures expérimentales du volume V_e sont classées en quatre catégories définies par trois valeurs seuils V_k ($k \in \{1,2,3\}$) à travers la sémantique suivante :

$$\sigma_{\mathfrak{M}_2}(V_e) = \begin{cases} (\textit{faible, petit}) & \text{si } V_e < V_1 \\ (\textit{faible, moyen}) & \text{si } V_1 \leq V_e < V_2 \\ (\textit{forte, moyen}) & \text{si } V_2 \leq V_e < V_3 \\ (\textit{forte, gros}) & \text{si } V_3 \leq V_e < V_{\max} \\ \textit{éclaté} & \text{si } V_e = \textit{éclaté} \end{cases}$$

Il est clair que le modèle $\mathfrak{M}_2$ est plus abstrait que $\mathfrak{M}_1$. Formellement, on peut traduire cette abstraction en mettant en relation les faits constituant les deux modèles à travers une fonction a de $E_{\mathfrak{M}_1}$ vers $E_{\mathfrak{M}_2}$. Pour les besoins de l'exemple, on suppose que $V_a < V_1$. La flèche d'abstraction a est définie par :

$$a(s) = \begin{cases} (\textit{faible, petit}) & \text{si } s = \bot \\ (\textit{faible, petit}) & \text{si } s = (F, V) \text{ et } V < V_1 \\ (\textit{faible, moyen}) & \text{si } s = (F, V) \text{ et } V_1 \leq V < V_2 \\ (\textit{forte, moyen}) & \text{si } s = (F, V) \text{ et } V_2 \leq V < V_3 \\ (\textit{forte, gros}) & \text{si } s = (F, V) \text{ et } V_3 \leq V < V_{\max} \\ \textit{éclaté} & \text{si } s = (F, V) \text{ et } V_{\max} \leq V \\ \textit{éclaté} & \text{si } s = \textit{éclaté} \end{cases}$$

Il est facilement de vérifier que la fonction d'abstraction a respecte les sémantiques proposées par :

$$\sigma_{\mathfrak{M}_2} = a \circ \sigma_{\mathfrak{M}_1}$$

Autrement dit, tout fait expérimental (représenté ci-dessus par une mesure V_e) est envoyée vers le même fait de $\mathfrak{M}_2$ en passant soit directement par la sémantique de $\mathfrak{M}_2$, soit indirectement en utilisant la sémantique de $\mathfrak{M}_1$ composée avec la fonction d'abstraction. Cette situation est illustrée Fig. 2.13.

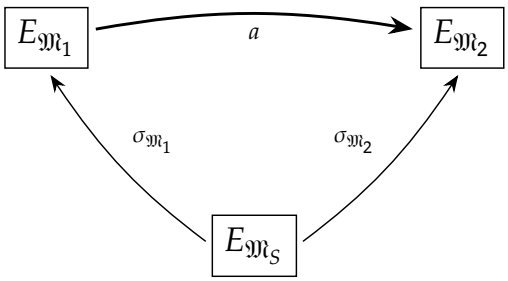


Fig. 2.13: Représentation schématique de la relation d'abstraction entre deux modèles $\mathfrak{M}_1$ et $\mathfrak{M}_2$ assujettis à leur validation par rapport au modèle de référence $\mathfrak{M}_S$. Le modèle $\mathfrak{M}_2$ est une abstraction de $\mathfrak{M}_1$.

2.4.2 Introduction aux catégories

Nous pensons que la théorie des catégories propose un cadre mathématique adapté à la formalisation des notions que nous venons d'évoquer. En particulier, les diagrammes donnés Fig. 2.13 et Fig. 2.11 sont typiques de cette théorie. Dans cette section, nous introduisons les éléments catégoriques nécessaires pour atteindre notre objectif.

Catégorie

Une catégorie est constituée de deux types d'éléments appelés objets et morphismes. Dans une représentation diagrammatique, les premiers sont en général représentés par des sommets et les seconds par des flèches. Une catégorie représente formellement comment les flèches peuvent se composer pour transiter d'objet en objet.

Définition 2.4.1 (Catégorie [3]). *Une* catégorie[12] *est un quadruplet* $\mathbf{K} = (O_\mathbf{K}, hom_\mathbf{K}, id_\mathbf{K}, \circ_\mathbf{K})$ *où*

1. $O_\mathbf{K}$ *est une classe*[13]*, dont les membres sont appelés des* **K**-objets*;*
2. *pour chaque paire d'objets* (A, B) *de* $\mathbf{K}$, $hom_\mathbf{K}(A, B)$ *est un ensemble dont les membres sont appelés des* **K**-*morphismes de* A *vers* B*: on appelle* A*, le domaine de* f *(noté* $dom(f)$*) et* B*, le codomaine de* f *(noté* $cod(f)$*)*[14]*; l'expression* $f \in hom_\mathbf{K}(A, B)$ *est également notée* $A \xrightarrow{f} B$ *ou encore* $f : A \to B$*;*
3. *pour chaque* **K**-*objet* A, $id_\mathbf{K}(A)$ *est un morphisme de* A *vers* A *appelé la* **K**-*identité sur* A*, également noté* id_A*;*
4. $\circ_\mathbf{K}$ *est une loi de composition associant à chaque paire de* **K**-*morphismes* $(f : A \to B, g : B \to C)$ *un* **K**-*morphisme* $(g \circ_\mathbf{K} f) : A \to C$*, appelé la composition de* f *et* g*, assujettie aux conditions suivantes :*
   (a) *la composition est associative : tout triplet de* **K**-*morphismes* $(f : A \to B, g : B \to C, h : C \to D)$ *vérifie l'équation* $hcirc_\mathbf{K}(g \circ_\mathbf{K} f) = (h \circ_\mathbf{K} g) \circ_\mathbf{K} f$*;*
   (b) *les* **K**-*identités sont neutres pour la composition : pour tout* **K**-*morphisme* $f : A \to B$*, on a* $id_B \circ_\mathbf{K} f = f = f \circ_\mathbf{K} id_A$.

La difficulté de la présentation des catégories vient de son caractère désincarné. Voici quelques exemples de catégories que nous avons déjà évoquées et dont nous ferons usage dans la suite.

12. localement petite

13. Dans la théorie des ensembles, les classes ont été introduites afin d'éviter les contradictions issues d'énoncés tels que le paradoxe de Russell (Est-ce que l'ensemble des ensembles n'appartenant pas à eux-même, $x \notin x$, appartient à lui-même ?). Tout comme les ensembles, les classes sont des collections d'éléments pouvant être caractérisées par un prédicat d'appartenance. La différence entre ensembles et classes tient à la limitation imposée sur les prédicats d'appartenance des ensembles par les axiomatisations utilisées (Z, ZF, ZFC, etc.). Ainsi tout ensemble est une classe, mais certaines classes, dites propres, ne sont pas des ensembles.

14. Pour assurer l'unicité du domaine et du codomaine, les ensembles $hom_\mathbf{K}(A, B)$ sont supposés deux à deux disjoints.

Exemple 2.4.1 (Catégorie des ensembles)**.** On note **Set** = $(O_{\mathbf{Set}}, hom_{\mathbf{Set}}, id_{\mathbf{Set}}, \circ_{\mathbf{Set}})$ la catégorie des ensembles où :

1. $O_{\mathbf{Set}}$ est la classe de tous les ensembles,
2. $hom_{\mathbf{Set}}(A, B)$ pour deux ensembles A et B est l'ensemble des fonctions de A dans B,
3. $id_{\mathbf{Set}}(A)$ est la fonction identité pour chaque ensemble A, et
4. $\circ_{\mathbf{Set}}$ est l'opérateur de composition de fonctions usuel en théorie des ensembles.

Exemple 2.4.2 (Catégorie des espaces topologiques)**.** La catégorie des espaces topologiques **Top** a pour objets les espaces topologiques et pour morphismes, les fonctions continues. Pour tout espace $\mathbb{S}$, $id_{\mathbb{S}}$ est la fonction identité (toujours continue). La composition est la composition de fonctions usuelle dont on peut montrer qu'elle respecte la continuité : toute composition de fonctions continues est continue.

Exemple 2.4.3 (Catégorie des monoïdes)**.** La catégorie des monoïdes **Mon** a pour objets les monoïdes et pour morphismes, les *morphismes de monoïdes*, c'est-à-dire les fonctions $h : \mathbb{T}_1 \to \mathbb{T}_2$ respectant les neutres et les lois de composition internes :

$$h(0_{\mathbb{T}_1}) = 0_{\mathbb{T}_2} \qquad h(x +_{\mathbb{T}_1} y) = h(x) +_{\mathbb{T}_2} h(y)$$

Là encore, l'identité et la composition usuelles respectent la définition 2.4.1 et sont utilisées pour compléter la définition de **Mon**.

Exemple 2.4.4 (Catégorie des actions de monoïdes)**.** On définit la catégorie **AMon** ayant pour objets les actions de monoïdes définies précédemment. On définit un morphisme entre deux actions $\Phi : E \times \mathbb{T} \to E$ et $\Phi' : E' \times \mathbb{T}' \to E'$ comme un couple $\langle h, f\rangle$ où h est un morphisme de monoïde de $\mathbb{T}$ dans $\mathbb{T}'$, et f est une fonction de E dans E' tels que pour tout $t \in \mathbb{T}, x \in E$:

$$\Phi'(f(x), h(t)) = f(\Phi(x, t))$$

Il est aisé de montrer que

$$id_{\mathbf{AMon}}(\Phi) = \langle id_{\mathbf{Mon}}(\mathbb{T}), id_{\mathbf{Set}}(E)\rangle \qquad \langle h, f\rangle \circ_{\mathbf{AMon}} \langle h', f'\rangle = \langle h \circ_{\mathbf{Mon}} h', f \circ_{\mathbf{Set}} f'\rangle$$

La théorie des catégories s'attache à comprendre les objets à travers les propriétés de transition qu'exposent les morphismes. La définition de **Top** par exemple nous invite à comprendre les espaces topologiques à travers les fonctions continues. Ce point de vue est illustré par la notion d'*isomorphisme*, qui s'intéresse à savoir si deux objets ont des rôles interchangeables dans un catégorie. La définition qui suit repose uniquement sur les flèches reliant les deux objets concernés.

Définition 2.4.2 (Objets isomorphes)**.** *Soit* **K** *une catégorie. Deux* **K***-objets* A *et* B *sont dits* isomorphes *s'il existe une paire de* **K***-morphismes* $f : A \to B$ *et* $g : B \to A$ *tels que :*

$$id_A = g \circ_{\mathbf{K}} f \qquad \text{et} \qquad id_B = f \circ_{\mathbf{K}} g$$

Les morphismes f *et* g *sont qualifiés d'*isomorphismes.

Les objets extrémaux sont un exemple de propriété, là encore décrits en terme de flèches.

Définition 2.4.3 (Objet initial, objet final)**.** *Soit* **K** *une catégorie. Un* **K***-objet* I *est appelé* objet initial *si pour chaque* **K***-objet* O*, il existe un unique morphisme allant de* I *vers* O*. De même, un* **K***-objet* F *est appelé un* objet final *si pour chaque* **K***-objet* O *il existe un unique morphisme allant de* O *vers* F*.*

À titre d'exemple, **Set** possède un objet initial, l'ensemble vide $\varnothing$, alors que tout singleton constitue un objet final. Nous constaterons d'ailleurs que lorsqu'il existe plusieurs objets initiaux (respectivement finaux), ceux-ci sont tous isomorphes.

Quelques diagrammes classiques

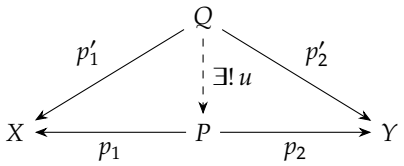
Nous remarquons que les propriétés intéressantes proviennent des multiples façons d'atteindre un objet destination à partir d'un même objet source. C'est par exemple ce qu'exprime la définition d'isomorphisme qui implique qu'il y a un moyen de ne pas « bouger » dans A tout en passant par B. La propriété pour deux chemins d'être égaux après composition s'appellent la *commutativité*. Cette notion est au cœurs de nombreuses constructions catégoriques.

Nous nous intéressons ici en particulier aux notions de *limite* et de *colimite*. De façon informelle, une (co)limite consiste à trouver un objet qui résume au mieux le comportement d'un groupe d'objets et de morphismes. Il s'agit d'une certaine façon de la possibilité de factoriser un grand nombre de flèches en une seule. Nous avons déjà rencontré cette construction section 2.2.3, dans la description des travaux d'A. Ehresmann sur la complexification structurelle : un cat-neurone, défini dans ce formalisme comme une colimite, représente à lui seul le comportement fonctionnel d'une famille de neurones.

Plutôt que de présenter formellement les définitions générales de limite et colimite, nous préférons nous concentrer sur 4 constructions particulières qui nous seront utiles dans la suite.

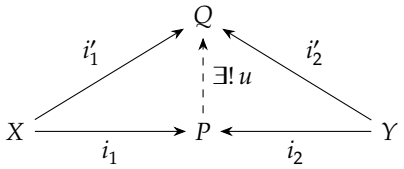
Produit et coproduit. Ce premier groupe de (co)limites cherche à définir un représentant P résumant le comportement de deux objets X et Y sur n'importe quel autre objet Q. On entend ici par comportement deux flèches f et g provenant de (ou tombant sur) Q et tombant sur (ou provenant de) X et Y. S'il existe un représentant P résumant X et Y alors tout couple de flèches (f, g) doit être résumable en une unique flèche entre P et Q. Ces considérations amènent aux définitions suivantes.

Définition 2.4.4 (Produit)**.** *Soient les deux objets* X, Y*. Le* produit *de* X *et* Y*, s'il existe, consiste en un objet* P *et deux morphismes* $p_1 : P \to X$ *et* $p_2 : P \to Y$ *tels que pour tout autre objet* Q *et morphismes* $p'_1 : Q \to X$ *et* $p'_2 : Q \to Y$*, il existe un unique morphisme* $u : Q \to P$ *tel que le diagramme suivant commute*



En renversant les flèches de cette définition, on obtient le *coproduit*, la construction *duale* du produit.

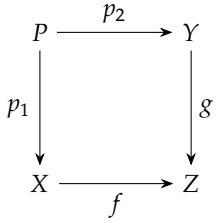
Définition 2.4.5 (Coproduit)**.** *Soient les deux objets X, Y. Le* coproduit *de X et Y, s'il existe, consiste en un objet P et deux morphismes $i_1 : X \to P$ et $i_2 : Y \to P$ tels que pour tout autre objet Q et morphismes $i'_1 : X \to Q$ et $i'_2 : Y \to Q$, il existe un unique morphisme $u : P \to Q$ tel que le diagramme suivant commute*



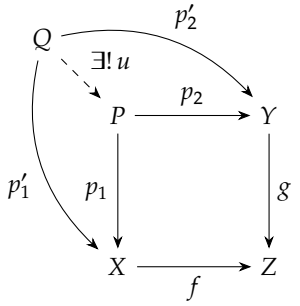
Illustrons ces constructions dans le cas des ensembles en commençant par le produit. En interprétant X et Y comme des ensembles, la construction cherche à définir le meilleur ensemble possible nous permettant de factoriser en une seule fonction u n'importe quel couple de fonctions f, g de même domaine Q. Autrement dit, pour un même élément $q \in Q$, les fonctions produisent deux valeurs $f(q)$ et $g(q)$ respectivement dans X et Y, et l'on cherche à représenter ces deux valeurs en une seule dans P sans perte d'information. Le meilleur ensemble P permettant de satisfaire ces contraintes est le produit cartésien $X \times Y$. Par un raisonnement similaire, on arrive à la conclusion que le coproduit dans **Set** correspond à l'union disjointe $X \uplus Y$.

Produit fibré et somme amalgamée. Ces nouvelles constructions sont proches des précédences. Le produit fibré peut être vu comme une construction comparable à un produit avec une contrainte supplémentaire : deux morphismes vers un troisième objet Z sont utilisés pour représenter une forme de cohérence entre X et Y.

Définition 2.4.6 (Produit fibré)**.** *Soient trois objets X, Y, Z et deux morphismes $f : X \to Z$ et $g : Y \to Z$. Le* produit fibré *des morphismes f, g, s'il existe, consiste en un objet P et deux morphismes $p_1 : P \to X$ et $p_2 : P \to Y$ tel que le diagramme suivant commute*

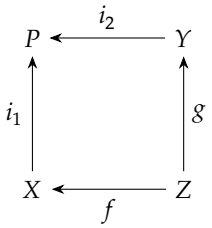


et soit universel, *c'est-à-dire que pour tout autre objet Q et morphismes $p'_1 : Q \to X$ et $p'_2 : Q \to Y$ alors il existe un unique morphisme $u : Q \to P$ tel que le diagramme suivant commute*

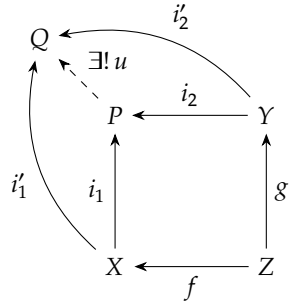


En renversant les flèches de cette définition, on obtient la *somme amalgamée*, la construction *duale* du produit fibré.

Définition 2.4.7 (Somme amalgamée)**.** *Soient trois objets X, Y, Z et deux morphismes $f : Z \to X$ et $g : Z \to Y$. La* somme amalgamée *des morphismes f, g, si elle existe, consiste en un objet P et deux morphismes $i_1 : X \to P$ et $i_2 : Y \to P$ tel que le diagramme suivant commute*



et soit universel, *c'est-à-dire que pour tout autre objet Q et morphismes $i'_1 : X \to Q$ et $i'_2 : Y \to Q$ alors il existe un unique morphisme $u : P \to Q$ tel que le diagramme suivant commute*



Dans **Set**, ces constructions existent toujours. Le produit fibré correspond à l'opération de θ-jointure, c'est-à-dire à la construction du produit cartésien de X et de Y dans lequel ne seront sélectionnés que les couples (x, y) vérifiant l'équation $f(x) = g(y)$. De la même façon, la somme amalgamée peut être vue comme l'union disjointe de X et Y quotientée par la relation $f(z) \equiv g(z)$ pour tout élément $z \in Z$.

Foncteurs

Il est naturel de munir une structure algébrique d'une notion de morphisme permettant de passer d'une algèbre à une autre en préservant la structure. Les catégories ne font pas exception, et l'on appelle *foncteur* la notion de morphisme de catégories.

Définition 2.4.8 (Foncteur)**.** *Soient* $\mathbf{C}$ *et* $\mathbf{D}$ *deux catégories. Un* foncteur[15] *de* $\mathbf{C}$ *vers* $\mathbf{D}$ *est un couple de fonctions* $F = \langle F_0, F_1 \rangle$ *tel que*

- F_0 *associe à chaque* $\mathbf{C}$*-objet* X*, un* $\mathbf{D}$*-objet* $F_0(X)$*,*
- F_1 *associe à chaque* $\mathbf{C}$*-morphisme* $f : X \to Y$*, un* $\mathbf{D}$*-morphisme* $F_1(f) : F_0(X) \to F_0(Y)$ *de sorte que :*
  1. $F_1(id_{\mathbf{C}}(X)) = id_{\mathbf{D}}(F_0(X))$ *pour tout* $\mathbf{C}$*-objet* X*, et*
  2. $F_1(g \circ_{\mathbf{C}} f) = F_1(g) \circ_{\mathbf{D}} F_1(f)$ *pour tous* $\mathbf{C}$*-morphismes* $f : X \to Y$ *et* $g : Y \to Z$*.*

Remarquons que les foncteurs peuvent être composés pour donner de nouveaux foncteurs. Il est d'usage d'utiliser F en lieu et place de F_0 et F_1 pour éviter les indices et alléger les notations. Nous présentons ci-dessous quelques exemples de foncteurs.

Exemple 2.4.5 (Foncteur identité)**.** Pour toute catégorie $\mathbf{C}$, on peut définir son *foncteur identité* qui envoie chaque $\mathbf{C}$-objet sur lui-même et chaque $\mathbf{C}$-morphisme sur lui-même. Avec ce foncteur, tous les ingrédients sont présents pour pouvoir définir la catégorie **Cat** des catégories. Il convient ainsi de noter $id_{\mathbf{Cat}}(\mathbf{C})$ (ou encore $1_{\mathbf{C}}$) le foncteur identité d'une catégorie $\mathbf{C}$.

Exemple 2.4.6 (Foncteur d'oubli)**.** De façon informelle, les *foncteurs d'oubli* permettent d'envoyer les objets d'une catégorie vers ceux d'une autre catégorie possédant moins de contraintes (justifiant le terme d'oubli). Nous intéresserons en particulier au foncteur d'oubli permettant de passer d'une catégorie $\mathbf{C}$ vers la catégorie des ensembles. Par exemple, on peut définir les foncteurs d'oubli suivants :

- $U_{\mathbf{Top}} : \mathbf{Top} \to \mathbf{Set}$ qui envoie tout espace topologique $\mathbb{S} = \langle E_{\mathbb{S}}, \Omega_{\mathbb{S}} \rangle$ vers son ensemble support $E_{\mathbb{S}}$ (en oubliant la structure décrite par les ouverts de $\Omega_{\mathbb{S}}$), et toute fonction continue vers elle-même ;
- $U_{\mathbf{Mon}} : \mathbf{Mon} \to \mathbf{Set}$ qui envoie tout monoïde $\mathbb{T} = \langle D_{\mathbb{T}}, 0_{\mathbb{T}}, +_{\mathbb{T}} \rangle$ vers son ensemble support $D_{\mathbb{T}}$ (en oubliant l'opération de composition entre ses éléments), et tout morphisme de monoïde vers lui-même, en tant que fonction de l'espace support ;
- $U_{\mathbf{AMon}} : \mathbf{AMon} \to \mathbf{Set}$ qui envoie toute action $\Phi : E \times \mathbb{T} \to E$ vers son ensemble support défini par :
$$U_{\mathbf{AMon}}(\Phi) = \{ (x, \delta, \Phi(x, \delta)) \mid x \in E, \delta \in \mathbb{T} \}$$
et tout morphisme d'action $\langle h, f \rangle : \Phi \to \Phi'$ vers la fonction :
$$\begin{array}{rccc} U_{\mathbf{AMon}}(\langle h, f \rangle) : & U_{\mathbf{AMon}}(\Phi) & \to & U_{\mathbf{AMon}}(\Phi') \\ & (x, t, x') & \mapsto & (f(x), h(t), f(x')) \end{array}$$

15. covariant

Constructions catégoriques

Nous terminons cette introduction aux catégories avec deux constructions : l'opposée d'une catégorie et la catégorie des flèches au-dessus (ou au-dessous) d'un objet (ou *slice* catégorie).

Catégorie opposée : Étant donnée une catégorie $\mathbf{C}$, il est toujours possible de considérer son *opposée* $\mathbf{C}^{\mathbf{op}}$ qui possède les mêmes objets que $\mathbf{C}$ mais dont les morphismes sont les flèches de $\mathbf{C}$ inversées. En d'autres termes, entre $\mathbf{C}$ et $\mathbf{C}^{\mathbf{op}}$, domaines et codomaines échangent leurs rôles.

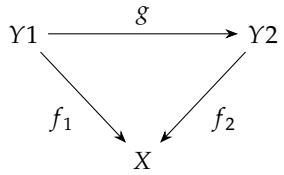
Cette construction est purement symbolique et n'apporte pas de propriétés supplémentaires à $\mathbf{C}^{\mathbf{op}}$ qui n'étaient pas déjà présente chez $\mathbf{C}$. Cependant, travailler dans la catégorie opposée permet souvent de travailler avec des flèches allant dans une direction plus naturelle pour la compréhension. L'un des principaux avantages de cette construction est de mettre en avant la dualité entre limites et colimites. En effet, comme nous l'avons indiqué en exemple plus haut, produit et coproduit sont des notions duales : passer de l'une à l'autre s'effectue en changeant le sens des flèches. Ainsi, un produit dans $\mathbf{C}$ se présentera sous la forme d'un coproduit dans $\mathbf{C}^{\mathbf{op}}$, et vice versa. Il en est de même pour le produit fibré et la somme amalgamée.

Flèches au-dessus d'un objet : Les morphismes étant les éléments de première importance dans les catégories, de nombreuses situations amènent à considérer comment les flèches sont relier les unes aux autres. Il existe plusieurs moyens de formalisation les relations entre flèches avec l'élaboration de 2-catégories qui ajoute un niveau supplémentaire d'éléments (des flèches entre flèches), les transformations naturelles, etc.

Nous nous intéressons, dans notre cas, à une construction spécifique où les flèches qu'il nous intéresse d'étudier possèdent toutes le même domaine ou le même codomaine. On parlera alors de catégories de flèches au-dessus ou au-dessous d'un objet. Pour ces catégories, il est nécessaire de fournir une notion de morphisme, ce que propose la définition suivante.

Définition 2.4.9 (Flèche au-dessus/au-dessous d'un objet)**.** *Soit* $\mathbf{C}$ *une catégorie et* X *un* $\mathbf{C}$*–objet. La* catégorie des $\mathbf{C}$*–morphismes* au-dessus de X*, noté* $\mathbf{C} \downarrow X$*, est la catégorie dont*

- *les objets sont les* $\mathbf{C}$*–morphismes de la forme* $f : Y \to X$*, et dont*
- *les morphismes entre* $f_1 : Y_1 \to X$ *et* $f_2 : Y_2 \to X$ *sont les* $\mathbf{C}$*–morphismes* $g : Y_1 \to Y_2$ *tels que* $f_1 = f_2 \circ_{\mathbf{C}} g$*, c'est-à-dire tels que le diagramme suivant commute :*



L'identité et la composition sont naturellement héritées de $\mathbf{C}$*.*

La construction duale est la catégorie des $\mathbf{C}$*–morphismes* au-dessous de X*, notée* $X \downarrow \mathbf{C}$*.*

2.4.3 Catégorie des modèles

Nous présentons maintenant les bases d'une construction catégorique pour la modélisation multi-niveau. Nous commençons par définir la catégorie $\mathbf{Abs}_S$ des modèles d'un système S. Nous proposons ensuite d'en analyser certaines propriétés fortement liées aux techniques de modélisation que nous avons présentées section 2.2.

Définition de la catégorie des modèles

Dans la suite, nous supposons l'existence d'un système S que l'on cherche à modéliser. Notre objectif est de définir une catégorie $\mathbf{Abs}_S$ dont les objets sont les modèles de S et les flèches représentent les liens d'abstraction entre ces modèles. En suivant l'idée générale que nous avons décrite dans l'introduction de cette section, nous supposons un ensemble particulier noté $E_{\mathfrak{M}_S}$ représentant les faits du système S auxquels tout modèle de S doit se référer. D'une certaine façon, l'ensemble $E_{\mathfrak{M}_S}$ représente le vocabulaire qu'on souhaite pouvoir utiliser pour décrire S. Rappelons que le but d'un modèle de S est d'expliquer le système S en faisant appel à ce vocabulaire. Dans le contexte des sciences expérimentales présenté section 2.4.1, ce vocabulaire consistait en toutes les expériences réalisées/réalisables sur le système étudié.

Définition 2.4.10 (Catégorie des modèles)**.** *Soit S un système représenté par un ensemble de faits de référence $E_{\mathfrak{M}_S}$. La catégorie $\mathbf{Abs}_S$ des modèles de S est définie par :*

$$\mathbf{Abs}_S := (E_{\mathfrak{M}_S} \downarrow \mathbf{Set})^{\mathbf{co}}$$

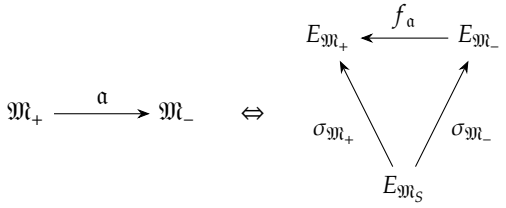
*à savoir, l'*opposée de la catégorie des flèches de **Set** *au-dessous de $E_{\mathfrak{M}_S}$.*

Cette définition mérite quelques éléments de clarification.

Objets de $\mathbf{Abs}_S$. Les objets de $\mathbf{Abs}_S$ représentent les modèles de S en accord avec les faits de référence de $E_{\mathfrak{M}_S}$. Par définition, un modèle $\mathfrak{M}$ est construit à partir d'une flèche de **Set** de domaine $E_{\mathfrak{M}_S}$. En notant cette flèche $\sigma_{\mathfrak{M}} : E_{\mathfrak{M}_S} \to E_{\mathfrak{M}}$, on retrouve ici la définition informelle de la sémantique des modèles que nous avons présentée : une fonction des faits de référence vers les faits du modèle.

On peut remarquer qu'un même ensemble de faits peut être utilisé par deux sémantiques différentes. De façon générale, tout ensemble E de **Set** (hormis l'ensemble vide) peut être atteint à partir de $E_{\mathfrak{M}_S}$. De plus, il apparaîtra dans $\mathbf{Abs}_S$ autant de fois qu'il existe de fonctions dans $hom_{\mathbf{Set}}(E_{\mathfrak{M}_S}, E)$, donnant lieu à autant de modèles différents de S pour ce seul ensemble de faits. La catégorie $\mathbf{Abs}_S$ est donc extrêmement riche en modèles.

Morphismes de $\mathbf{Abs}_S$. Si l'on constate que les objets de $\mathbf{Abs}_S$ formalisent la notion de sémantique des modèles que nous cherchions à mettre en place, ses flèches permettent quant à elles de formaliser la notion d'abstraction. En effet, la définition 2.4.10 amène à considérer une flèche $\mathfrak{a} : \mathfrak{M}_+ \to \mathfrak{M}_-$, signifiant que le modèle $\mathfrak{M}_+$ est une abstraction du modèle $\mathfrak{M}_-$, pour chaque fonction $f_{\mathfrak{a}}$ de $E_{\mathfrak{M}_-}$ dans $E_{\mathfrak{M}_+}$ telle que le diagramme suivant commute :



Nous retrouvons à nouveau la situation décrite informellement en début de section. On remarquera en particulier la correspondance entre ce diagramme et celui illustré Fig. 2.13.

Nous finirons en rappelant que par construction la composition et l'identité sont bien définies dans $\mathbf{Abs}_S$. La composition correspond à la transitivité de l'abstraction qui revient à dire informellement que si $\mathfrak{M}_1$ est une abstraction de $\mathfrak{M}_2$ et que $\mathfrak{M}_2$ est une abstraction de $\mathfrak{M}_3$, alors $\mathfrak{M}_1$ est une abstraction de $\mathfrak{M}_3$ par composition des deux fonctions d'abstraction. Pour ce qui est de l'identité, elle signifie que tout modèle est une abstraction de lui-même. Dans ce cas, la fonction d'abstraction correspond à la fonction identité de l'ensemble des faits du modèle concerné.

Objets extrémaux de $\mathbf{Abs}_S$. Toute catégorie de flèches d'une catégorie $\mathbf{C}$ arbitraire au-dessous d'un $\mathbf{C}$–*objet* X vérifie les deux propriétés suivantes :

1. id_X (qui est bien un $\mathbf{C}$–*morphisme* au-dessous de X) est un objet initial de $X \downarrow \mathbf{C}$;
2. si $\mathbf{C}$ présente un objet terminal F, l'unique $\mathbf{C}$–*morphisme* de X vers F est un objet terminal de $X \downarrow \mathbf{C}$.

En considérant que la définition 2.4.10 considère l'opposée d'une telle catégorie et que les notions d'objets initiaux et finaux sont duales, on déduit aisément que la catégorie $\mathbf{Abs}_S$ possède des objets initiaux et des objets finaux.

Les objet initiaux sont les fonctions constantes de $E_{\mathfrak{M}_S}$ vers les singletons, c'est-à-dire les objets terminaux de **Set**. Par exemple, on associe au singleton $\{\bot\}$, le modèle $\mathfrak{M}_\bot$ correspondant au modèle le plus abstrait possible, celui qui offre le moins d'explication de S en identifiant tout les faits de $E_{\mathfrak{M}_S}$ en un seul, $\bot$. En revanche, l'objet final, à savoir la fonction identité $id_{E_{\mathfrak{M}_S}}$, constitue le modèle $\mathfrak{M}_S$ le plus concret qu'il est possible de construire étant donné le vocabulaire proposé dans $E_{\mathfrak{M}_S}$. Ainsi, $\mathbf{Abs}_S$ se présente comme une *hiérarchie d'abstraction* entre les modèles de S dont les objets initiaux et finaux forment les deux extrémités. On remarquera que, dans le cas particulier où le vocabulaire que l'on s'autorise pour parler de S est vide (c'est-à-dire lorsque $E_{\mathfrak{M}_S} = \varnothing$), la hiérarchie « s'écrase » en un unique objet.

Transformation de modèles

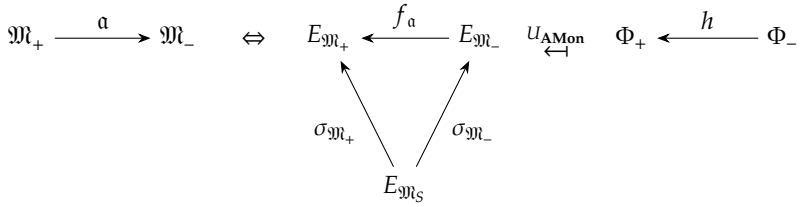
L'opération essentielle que notre cadre formel met en avant est l'abstraction entre modèles. Elle provient des concepts développés dans le cadre des transformations de modèles que nous

avons présentées section 2.2.2 au sein de laquelle référence était faite à deux propriétés des transformations qu'il est bon de confronter à notre proposition.

Transformation endogène et foncteur d'oubli. Se restreindre à la simple définition ensembliste de $\mathbf{Abs}_S$ n'apporte aux modèles aucun pouvoir explicatif. Ce fut d'ailleurs l'objet de la section 2.3.2 que de montrer qu'il est nécessaire de rajouter de la structure sur les ensembles de faits pour en tirer plus d'informations. On peut d'ailleurs remarquer qu'en tant que objet terminal, les faits de référence constituent certes le modèle le plus concret mais, privés de structure, n'apportent aucune description supplémentaire du système.

Pour palier à cette absence de structure, notre proposition consiste à considérer toute catégorie $\mathbf{C}$ munie d'un foncteur d'oubli $U_{\mathbf{C}}$ vers **Set** comme la description d'un formalisme. Par exemple, les modèles dont les ensembles de faits sont des images d'actions de monoïde par le foncteur $U_{\mathbf{AMon}}$, correspondent à la classe des modèles dynamiques explicitée définition 2.3.4.

Il est possible d'aller plus loin dans cette proposition. Nous pouvons considérer les abstractions issues des images de morphismes par *le même* foncteur d'oubli $U_{\mathbf{C}}$: ces cas correspondent exactement aux transformations de modèles endogènes au formalisme décrit par $\mathbf{C}$. Illustrons cela pour les modèles dynamiques. Considérons dans **AMon**, Φ_- et Φ_+ deux actions de monoïde, et $h : \Phi_- \to \Phi_+$ un morphisme entre elles. Le cas qui nous intéresse suppose que $U_{\mathbf{AMon}}$ envoie Φ_- (respectivement Φ_+) sur l'ensemble des faits $E_{\mathfrak{M}_-}$ (respectivement $E_{\mathfrak{M}_+}$) d'un modèle $\mathfrak{M}_-$ (respectivement $\mathfrak{M}_+$) de $\mathbf{Abs}_S$, de telle sorte que l'image de $U_{\mathbf{AMon}}(h) = f_{\mathfrak{a}}$ pour une flèche d'abstraction $\mathfrak{a} : \mathfrak{M}_+ \to \mathfrak{M}_-$:



Cette situation décrit bien une abstraction de modèles dynamiques : passer du modèle $\mathfrak{M}_+$ au modèle $\mathfrak{M}_-$ par $\mathfrak{a}$ est une transformation endogène de $\mathfrak{M}_+$ en $\mathfrak{M}_-$ respectant l'action du temps.

Évidemment, les transformations exogènes sont également à l'honneur. Elles correspondent aux abstractions $\mathfrak{a} : \mathfrak{M}_+ \to \mathfrak{M}_-$ dont les domaines et codomaines sont dans l'image de foncteurs d'oubli *différents*.

Transformation horizontale et isomorphisme. Considérons parmi les transformations de modèles, les transformations verticales et les transformations horizontale. Les premières correspondent à une notion de raffinement entre le modèle initial et le modèle transformé de telle sorte que le premier soit en général plus abstrait que le second. La notion d'abstraction que nous avons formalisée correspond à cette verticalité.

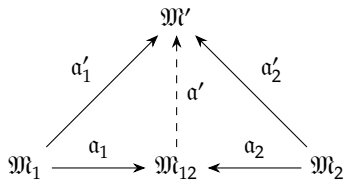
Les transformations horizontales correspondent à des équivalences entre des modèles ayant

au final le même pouvoir d'explication tout en étant décrits de façons différentes, par exemple lorsqu'ils reposent sur des formalismes différents (ce qui est représentable dans notre cadre par l'utilisation de foncteurs d'oubli différents comme nous venons de le voir). Les deux modèles sont donc interchangeables. Comme nous l'avons vu, cette propriété se traduit en termes catégoriques par un isomorphisme entre objets. Soient $\mathfrak{M}_1$ et $\mathfrak{M}_2$, deux modèles de $\mathbf{Abs}_S$. Par définition, l'isomorphisme entre $\mathfrak{M}_1$ et $\mathfrak{M}_2$ se traduit par la présence de deux abstractions $\mathfrak{a}_{12} : \mathfrak{M}_1 \to \mathfrak{M}_2$ et $\mathfrak{a}_{21} : \mathfrak{M}_2 \to \mathfrak{M}_1$ telles que $\mathfrak{a}_{12} \circ \mathfrak{a}_{21}$ et $\mathfrak{a}_{21} \circ \mathfrak{a}_{12}$ donnent respectivement les identités de $\mathfrak{M}_1$ et de $\mathfrak{M}_2$. En dépliant la définition de $\mathbf{Abs}_S$, cela revient à dire que les ensembles de faits de $\mathfrak{M}_1$ et de $\mathfrak{M}_2$ sont isomorphes dans **Set**. Attention, l'inverse est faux : deux modèles dont les ensembles de faits sont en bijection ne sont pas nécessairement isomorphes car il peut ne pas exister de bijection respectant les sémantiques deux modèles simultanément.

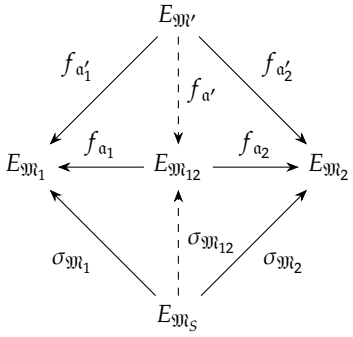
Couplage de modèles

L'opération importante de la modélisation multi-modèle est le couplage de modèles. Nous montrons dans cette dernière section comment les constructions de (co)limites permettent ce couplage.

Le couplage simple. Le couplage de deux modèles peut être décrit en terme d'abstraction : le couplage entre un modèle $\mathfrak{M}_1$ et un modèle $\mathfrak{M}_2$ revient à construire le « meilleur » modèle $\mathfrak{M}_{12}$ dont $\mathfrak{M}_1$ et $\mathfrak{M}_2$ sont des abstractions. En effet, on cherche à retrouver dans $\mathfrak{M}_{12}$ les contributions de $\mathfrak{M}_1$ et $\mathfrak{M}_2$. En cela, $\mathfrak{M}_{12}$ est plus complet, et donc plus concret, que ses parents. De plus, on entend par « meilleur » modèle, le fait que la définition de $\mathfrak{M}_{12}$ se limite uniquement aux contributions de $\mathfrak{M}_1$ et $\mathfrak{M}_2$. En d'autres termes, tout autre modèle $\mathfrak{M}'$ établissant un rapport entre $\mathfrak{M}_1$ et $\mathfrak{M}_2$ serait une spécialisation de $\mathfrak{M}_{12}$. Sous la forme d'un diagramme, cette définition informelle se décrit de la façon suivante :



On reconnaît ici le diagramme d'un coproduit qui invite à poser la définition du couplage dans $\mathbf{Abs}_S$ à travers cette construction. Il faut cependant s'assurer de son existence et, le cas échéant, de sa définition : quels ensemble de faits et sémantique peut-on lui associer ? Pour ce faire, il suffit de considérer l'équivalent du diagramme précédent dans **Set**.

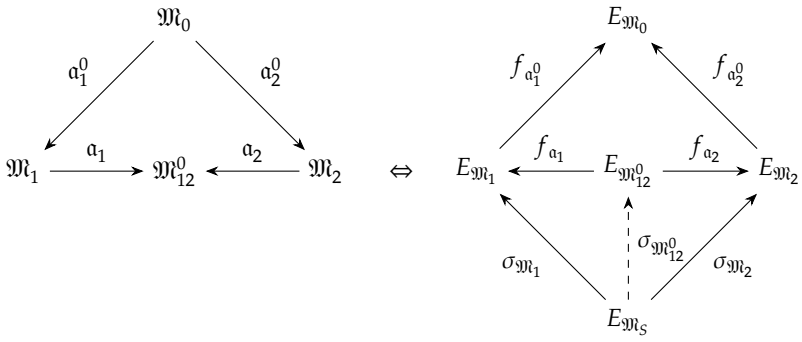


Le coproduit de $\mathbf{Abs}_S$ dérive ici du produit de **Set** dont il hérite les propriétés. Nous en déduisons qu'il est toujours possible de coupler deux modèles ; il suffit pour cela de faire le produit cartésien des ensembles de faits des modèles à coupler. Les fonctions d'abstraction correspondent alors aux fonctions de projection. Nous avons également fait figurer sur le diagramme l'ensemble des faits de référence $E_{\mathfrak{M}_S}$ et les sémantiques des modèles, $\sigma_{\mathfrak{M}_1}$ et $\sigma_{\mathfrak{M}_2}$. L'universalité du coproduit implique alors qu'il existe un unique morphisme de $E_{\mathfrak{M}_S}$ vers $E_{\mathfrak{M}_{12}}$, que nous identifions aisément à la sémantique de $\mathfrak{M}_{12}$. Celle-ci consiste à retourner pour chaque fait de référence, le n-uplet des faits associés par chaque modèle :

$$\begin{array}{rccl} \sigma_{\mathfrak{M}_{12}} : & E_{\mathfrak{M}_S} & \longrightarrow & E_{\mathfrak{M}_{12}} = E_{\mathfrak{M}_1} \times E_{\mathfrak{M}_2} \\ & r & \longmapsto & (\sigma_{\mathfrak{M}_1}(r), \sigma_{\mathfrak{M}_2}(r)) \end{array}$$

Somme amalgamée de modèles. Remarquons qu'en considérant toutes les paires de faits possibles, le produit cartésien construit beaucoup plus de couples que nécessaire. La sémantique joue un rôle de filtre puisque seule l'image de $\sigma_{\mathfrak{M}_{12}}$ n'est au final prise en compte dans la définition des flèches de $\mathbf{Abs}_S$. Cela fait jouer un rôle important aux faits de référence qu'il n'est pas toujours possible de tenir. En effet, si l'on reprend le cas des sciences expérimentales par exemple, la référence est donnée par l'expérience et la réalisation de *toutes* les expériences est en réalité impraticable. Le couplage simple reste théorique lorsque le choix de l'ensemble $E_{\mathfrak{M}_S}$ est une hypothèse de travail.

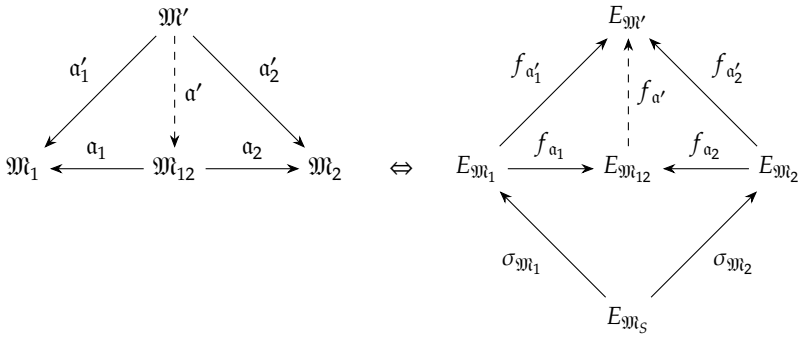
Néanmoins, nous avons vu que le produit fibré correspondait pour les ensembles à un produit cartésien suivi d'une sélection permettant de se ramener à l'ensemble de couples qui nous intéresse. Tout comme le produit de **Set** donne naissance à un coproduit dans $\mathbf{Abs}_S$, le produit fibré de **Set** devient une somme amalgamée dans $\mathbf{Abs}_S$:



Pour cette construction, on suppose, en plus des deux modèles qui nous intéressent, un troisième modèle $\mathfrak{M}_0$ plus abstrait que $\mathfrak{M}_1$ et $\mathfrak{M}_2$ (par les abstractions $\mathfrak{a}_1^0$ et $\mathfrak{a}_2^0$). La somme amalgamée de ce diagramme revient à calculer le produit fibré dans **Set**. Celui-ci correspond, comme attendu, à un produit cartésien entre les faits de $\mathfrak{M}_1$ et $\mathfrak{M}_2$ restreint aux couples envoyés vers un fait identique par $f_{\mathfrak{a}_1^0}$ et $f_{\mathfrak{a}_2^0}$. Les deux abstractions $\mathfrak{a}_1$ et $\mathfrak{a}_2$ correspondent toujours aux projecteurs. La sémantique $\sigma_{\mathfrak{M}_{12}^0}$ est donnée, comme pour le couplage simple, par universalité. Elle est donc unique et correspond à la restriction de la sémantique $\sigma_{\mathfrak{M}_{12}}$ du couplage simple au sous-ensemble du produit cartésien imposé par $\mathfrak{M}_0$. Plus $\mathfrak{M}_0$ sera proche (en terme d'abstraction) de $\mathfrak{M}_1$ et $\mathfrak{M}_2$, plus la restriction sera fine[16]. On remarquera d'ailleurs que lorsque le modèle $\mathfrak{M}_0 = \mathfrak{M}_\perp$, le modèle initial (le plus abstrait), la construction se ramène au couplage simple.

Pour conclure, la somme amalgamée permet de coupler deux modèles en se restreignant à une partie commune identifiée par un troisième modèle.

Construction duale. Si le coproduit permet de spécifier le couplage de modèles, il est naturel de considérer la composition de modèles induite par le produit de $\mathbf{Abs}_S$. Nous procédons comme pour le produit en analysant les diagrammes :

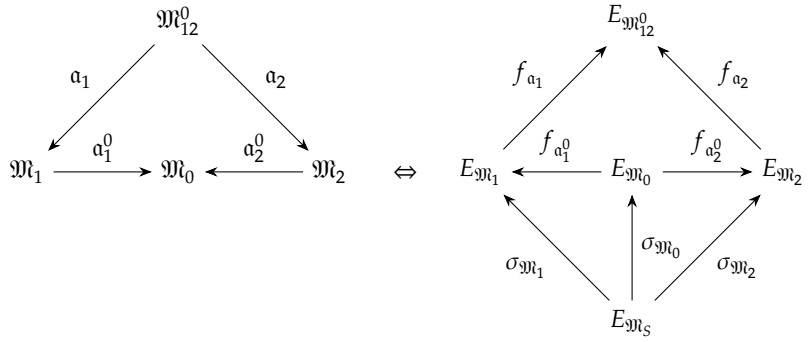


Dans ce cas, et comme on pouvait l'attendre d'une construction duale, le produit dans $\mathbf{Abs}_S$ construit le dénominateur commun aux deux modèles le plus concret possible. Cependant,

16. La meilleure restriction sera d'ailleurs obtenue lorsque $\mathfrak{M}_0$ correspond au produit de $\mathfrak{M}_2$ et de $\mathfrak{M}_2$. Cependant, comme nous le verrons dans le paragraphe suivant ce calcul dépend fortement d'une connaissance explicite de $E_{\mathfrak{M}_S}$.

d'un point de vue catégorique, les calculs sont différents. Le produit de $\mathbf{Abs}_S$ ne résulte pas du coproduit de **Set** mais de la somme amalgamée, avec les faits de référence et les sémantiques pour sommet du diagramme. Ainsi, le modèle $\mathfrak{M}_{12}$ correspond à l'union disjointe des faits de $\mathfrak{M}_1$ et de $\mathfrak{M}_2$ quotientés par la relation d'équivalence regroupant au sein d'une même classe les faits images d'un même fait de référence. D'un point de vue constructif moins intéressant que le coproduit, le produit trouvera un intérêt dans un contexte d'analyse (de code par exemple) afin d'extraire de deux modèles d'un même système la « meilleure » partie commune.

À nouveau, cette construction dépend d'une connaissance approfondie des faits de référence et n'est donc pas applicable dans un certain nombre de cas. On peut envisager de faire endosser ce rôle à un autre modèle à travers un produit fibré dans $\mathbf{Abs}_S$, ce qui revient à considérer le diagramme suivant :



Au final, on obtient un dénominateur commun $\mathfrak{M}_{12}^0$ qui n'est pas le plus concret possible en général, mais qui est le plus concret possible vérifiant la composition induite par la construction de $\mathfrak{M}_0$ à partir de $\mathfrak{M}_1$ et $\mathfrak{M}_2$.

2.5 Conclusion et perspectives

Cette courte section vise à rappeler les principaux résultats obtenus suite à notre approche théorique du multi-niveau ainsi que nos futures pistes de travail sur le sujet.

2.5.1 Conclusion

Dans ce chapitre nous avons posé les fondements d'une description formelle et trans-formalisme pour la description et pour une meilleure compréhension de la modélisation multi-niveau. Nous avons commencé dans la section 2.2 par la définition du terme multi-niveau en présentant différent types de modélisation multi-niveau : couplage de modèle, transformation de modèle et complexification. Nous avons déterminé que la modélisation multi-niveau était un cas particulier du multi-modèle, pratique dans laquelle plusieurs modèles sont assemblés afin de construire des modèles plus conséquents. Dans la modélisation multi-niveau s'ajoute une contrainte sur la représentation de cet assemblage, où les modèles sont rassemblés par niveaux

et où nous pouvons décrire via des méthodes génériques comment assembler ces modèles selon leur niveau de représentation. Comme nous n'avons pas trouvé de cadre général suffisamment expressif pour parler de ces assemblages, nous avons décidé d'élaborer, section 2.3.2, une définition générale de modèle qui puisse englober tous ces cas d'usage. S'intéresser à la notion de modèle nous a amené à différencier les termes de modèle, système et formalisme. Un modèle est, au final, un ensemble de faits, ce que nous décrivons ensuite formellement par un foncteur d'oubli. Le système n'est pas accessible et constitue l'inspiration d'un modèle expérimental, décrit par extension par le biais de mesures sur ce système. Le formalisme est quant à lui le principe organisateur des faits du modèle. Nous avons ensuite défini quelques types de modèles classiques (dynamiques, probabilistes, ...). Enfin, dans la section 2.4, nous nous penchons sur la description de la composition de plusieurs modèles, d'après les différentes méthodes d'assemblage que nous avons présenté plus tôt. Cette présentation nous amène à introduire quelques notions de théorie des catégorie, une théorie s'intéressant à la composition d'éléments opaques appelés objets et morphismes, afin d'introduire la catégorie des modèles. Dans ce cadre, nous avons pu exprimer clairement le rôle du modèle de référence comme garant de la sémantique commune à plusieurs modèles à assembler.

Ce chapitre représente une toute première étape de défrichage qui a permis de mettre en lumière les résultats suivants :

- une définition de modèle, système et formalisme ;
- la mise en évidence d'un rapport de chaque modèle avec sa sémantique, identifiée comme son lien avec un modèle de référence ;
- la clarification de la flèche d'abstraction comme un morphisme de **Abs**;
- l'éclaircissement des transformations de modèles à la lumière de **Abs**: transformation verticale (abstraction) et horizontale (changement de point de vue) ;
- la présentation des différentes manières de composer plusieurs modèles au sein de **Abs**: couplage simple, somme amalgamée de modèles (correspondant à un produit cartésien des faits de chacun des modèles restreint par la sémantique) et sa construction duale (correspondant au « dénominateur commun » entre les deux modèles considérés).

2.5.2 Perspectives

Malgré la présentation de nombreux nouveaux concepts, il reste encore beaucoup à accomplir pour parfaire un cadre théorique pour l'intégration des niveaux d'organisation dans les modèles.

Le premier point est le manque d'applications pratiques décrites avec cet outil. Nous comptons, à court terme, écrire une section sur l'application à la modélisation proie/prédateur, système présenté en milieu de chapitre. De manière générale, nous avons manqué d'exemples à la fois concrets et suffisamment simples. Ainsi une perspective de travail à plus long terme est la présentation, dans le cadre que nous avons introduit, de nombreux autres exemples pouvant ainsi aider à affiner les constructions présentées section 2.4. Parmi ces exemples, la question de la modélisation de l'activité du chapitre 3 et la composition des trois modèles support de

OTB du chapitre 4 ont tout à fait leur place. À terme, nous comptons décrire au moins un cas d'application par type de transformation/couplage :

- isomorphisme (transformation horizontale),
- abstraction/raffinement (transformation verticale),
- couplage simple,
- somme de modèles,
- produit de modèles,

tout en mettant en évidence la classe de modèle utilisée : modèles dynamiques, modèles à champs, modèles probabilistes, etc.

Le second point que nous souhaitons explorer est bien plus prospectif. Une fois quelques exemples traité dans notre formalisme, quels nouveaux outils peut-on développer pour analyser et automatiser la construction de modèles multi-niveau. Avec le recul, nous devrions être capable d'identifier des structures récurrentes dans l'établissement des modèles, suivant les objets d'études ou suivant les disciplines.

Finalement, nous pourrons aborder notre objectif de recherche à long terme : notre outil est-il un cadre suffisant pour exprimer clairement le fonctionnement des systèmes complexes par le biais de mécanismes de complexification. Nous commencerons par décrire les exemples donnés section 2.2.3 avec nos propres outils : le raffinement d'un modèle complexe de F. Polack et S. Stepney peut-il être décrit plus simplement dans notre cadre, peut-on pointer spécifiquement ce qui diffère d'un raffinement classique ? La complexification dans les CNN et la complexification structurelle de A. Ehresmann et J.-P. Vanbermeersch peuvent-ils simplement s'exprimer dans notre cadre ?

Références

[1] HDI Abarbanel, Mikhail Izrailevich Rabinovich, A Selverston, MV Bazhenov, R Huerta, MM Sushchik et LL Rubchinskii. "Synchronisation in neural networks". In : *Physics-Uspekhi* 39.4 (1996), p. 337–362.

[2] LF Abbott et Thomas B Kepler. "Model Neurons: from Hodgkin-Huxley to Hopfield". In : *Statistical mechanics of neural networks*. Springer, 1990, p. 5–18.

[3] Jiří Adámek, Horst Herrlich et George E Strecker. "Abstract and concrete categories. The joy of cats". In : (2004).

[4] Anne Bretagnolle, Eric Daudé et Denise Pumain. "From theory to modelling : urban systems as complex systems". In : *Cybergeo : European Journal of Geography* (2006).

[5] Anthony N Burkitt. "A review of the integrate-and-fire neuron model : I. Homogeneous synaptic input". In : *Biological cybernetics* 95.1 (2006), p. 1–19.

[6] Noam Chomsky. "Three models for the description of language". In : *IRE Transactions on information theory* 2.3 (1956), p. 113–124.

[7] Leon O. Chua. "CNN: A Vision of Complexity". In : *International Journal of Bifurcation and Chaos* 07.10 (1997), p. 2219–2425.

[8] Alonzo Church. "A set of postulates for the foundation of logic". In : *Annals of mathematics* (1932), p. 346–366.

[9] Edgar F Codd. "A relational model of data for large shared data banks". In : *Communications of the ACM* 13.6 (1970), p. 377–387.

[10] Chris G De Koster et Aristid Lindenmayer. "Discrete and continuous models for heterocyst differentiation in growing filaments of blue-green bacteria". In : *Acta Biotheoretica* 36.4 (1987), p. 249–273.

[11] Andrée C. Ehresmann. "MENS: From Neurons to Higher Mental Processes up to Consciousness". In : *Integral Biomathics: Tracing the Road to Reality*. Sous la dir. de L. Plamen Simeonov, S. Leslie Smith et C. Andrée Ehresmann. Berlin, Heidelberg : Springer Berlin Heidelberg, 2012, p. 29–30.

[12] Stéphane Ferret. "Le bateau de Thésée". In : *Le problème de l'identité à travers le temps, Paris, Éditions de Minuit* (1996).

[13] Richard FitzHugh. "Impulses and physiological states in theoretical models of nerve membrane". In : *Biophysical journal* 1.6 (1961), p. 445.

[14] Martin Gardner. "Mathematical games : The fantastic combinations of John Conway's new solitaire game "life"". In : *Scientific American* 223.4 (1970), p. 120–123.

[15] Daniel T Gillespie. "Exact stochastic simulation of coupled chemical reactions". In : *The journal of physical chemistry* 81.25 (1977), p. 2340–2361.

[16] Marco Giunti et Claudio Mazzola. "Dynamical systems on monoids : Toward a general theory of deterministic systems and motion". In : *Methods, models, simulations and approaches towards a general theory of change* (2012), p. 173–185.

[17] David Golomb, John Guckenheimer et Shay Gueron. “Reduction of a channel-based model for a stomatogastric ganglion LP neuron”. In : *Biological cybernetics* 69.2 (1993), p. 129–137.

[18] James B Grace, T Michael Anderson, Eric W Seabloom, Elizabeth T Borer, Peter B Adler, W Stanley Harpole, Yann Hautier, Helmut Hillebrand, Eric M Lind, Meelis Pärtel et al. “Integrative modelling reveals mechanisms linking productivity and plant species richness”. In : *Nature* 529.7586 (2016), p. 390–393.

[19] Diana-Elena Gratie, Bogdan Iancu, Sepinoud Azimi et Ion Petre. *Quantitative model refinement in four different frameworks, with applications to the heat shock response*. Rapp. tech. Technical Report 1067, TUCS, 2013.

[20] JL Hindmarsh et RM Rose. “A model of neuronal bursting using three coupled first order differential equations”. In : *Proceedings of the Royal society of London. Series B. Biological sciences* 221.1222 (1984), p. 87–102.

[21] Jonathan R Karr, Jayodita C Sanghvi, Derek N Macklin, Miriam V Gutschow, Jared M Jacobs, Benjamin Bolival, Nacyra Assad-Garcia, John I Glass et Markus W Covert. “A whole-cell computational model predicts phenotype from genotype”. In : *Cell* 150.2 (2012), p. 389–401.

[22] Anneke G Kleppe, Jos B Warmer et Wim Bast. *MDA explained : the model driven architecture : practice and promise*. Addison-Wesley Professional, 2003.

[23] Alfred J Lotka. “Analytical note on certain rhythmic relations in organic systems”. In : *Proceedings of the National Academy of Sciences* 6.7 (1920), p. 410–415.

[24] Alfred J Lotka. “Elements of physical biology”. In : (1925).

[25] Thomas Louail. “Comparer les morphogénèses urbaines en Europe et aux États-Unis par la simulation à base d’agents–Approches multi-niveaux et environnements de simulation spatiale”. Thèse de doct. Université d’Evry-Val d’Essonne, 2010.

[26] Carlos A Lugo et Alan J McKane. “Quasicycles in a spatial predator-prey model”. In : *Physical Review E* 78.5 (2008), p. 051911.

[27] Derek N Macklin, Nicholas A Ruggero et Markus W Covert. “The Future of Whole-Cell Modeling”. In : *Current opinion in biotechnology* 28 (août 2014), p. 111–115.

[28] Klaus Mainzer et Leon O. Chua. *Local Activity Principle: The Cause of Complexity and Symmetry Breaking*. Imperial College Press, 2013.

[29] Alan J McKane et Timothy J Newman. “Predator-prey cycles from resonant amplification of demographic stochasticity”. In : *Physical review letters* 94.21 (2005), p. 218102.

[30] Tom Mens et Pieter Van Gorp. “A taxonomy of model transformation”. In : *Electronic Notes in Theoretical Computer Science* 152 (2006), p. 125–142.

[31] Catherine Morris et Harold Lecar. “Voltage oscillations in the barnacle giant muscle fiber.” In : *Biophysical journal* 35.1 (1981), p. 193.

[32] Eugene Pleasants Odum et Howard Thomas Odum. *Fundamentals of Ecology.[By] EP Odum... in Collaboration with Howard T. Odum...* Philadelphia & London, 1959.

[33] OMG. *OMG Unified Modeling Language (OMG UML), Version 2.5*. Object Management Group, 2015.

[34] Rolf O Peterson et John A Vucetich. "Ecological studies of wolves on Isle Royale". In : *Annual Report* 98 (1997).

[35] Fiona Polack et Susan Stepney. "Emergent properties do not refine". In : *Electronic Notes in Theoretical Computer Science* 137.2 (2005), p. 163–181.

[36] Przemyslaw Prusinkiewicz et Aristid Lindenmayer. *The algorithmic beauty of plants*. Springer Science & Business Media, 2012.

[37] D Pumain, L Sanders, H Mathian, F Guerin-Pace et S Bura. "SIMPOP: a multi-agents model for urban transition". In : *Recent development in spatial information modelling and processing. Budapest, Geomarket* (1995), p. 71–85.

[38] Dominique Ricard. *La vie des hommes illustres*. fr. T. 1. Paris : Firmin Didiot Frères, 1858.

[39] Grzegorz Rozenberg et Arto Salomaa. *The mathematical theory of L systems*. T. 90. Academic press, 1980.

[40] RB Stein, KV Leung, D Mangeron et MN Oğuztöreli. "Improved neuronal models for studying neural networks". In : *Kybernetik* 15.1 (1974), p. 1–9.

[41] Susan Stepney, David Cooper et Jim Woodcock. "More Powerful Z Data Refinement: pushing the state of the art in industrial refinement". In : 1998, p. 284–307.

[42] Gabriele Taentzer, Karsten Ehrig, Esther Guerra, Juan de Lara, Laszlo Lengyel, Tihamer Levendovszky, Ulrike Prange, Dániel Varró et Szilvia Varró-Gyapay. "Model transformation by graph transformation : A comparative study". In : (2005).

[43] Jeffrey Travers et Stanley Milgram. "The small world problem". In : *Phychology Today* 1 (1967), p. 61–67.

[44] Alan Mathison Turing. "On computable numbers, with an application to the Entscheidungsproblem". In : *Proceedings of the London mathematical society* 2.1 (1937), p. 230–265.

[45] Vito Volterra. "Fluctuations in the abundance of a species considered mathematically". In : *Nature* 118.2972 (1926), p. 558–560.

[46] Jan C Willems. "Paradigms and puzzles in the theory of dynamical systems". In : *Automatic Control, IEEE Transactions on* 36.3 (1991), p. 259–294.

Chapitre 3

L'activité dans la programmation spatiale

Sommaire

3.1 Introduction

Nous commencerons cette partie en introduisant la notion d'activité, ses origines et son rapport à la modélisation multi-niveau. Après une partie concernant MGS et le formalisme qui lui est associé, nous aborderons ses liens avec l'*activité*. Il apparaît que cette notion est intimement liée à la vision de l'espace que promeut le projet MGS et ainsi, notre exploration mettra en évidence des résultats à la fois formels et pratiques.

3.2 Activité

Intuitivement, nous dirons que l'activité est constitué d'un ensemble des choses qui changent, c'est à dire sur lesquels une action a eu lieu, en opposition au reste qui ne change pas. Au delà des différents sens communs accordés à l'activité, on rencontre quelques autres termes qui y sont naturellement liés comme état, énergie, changement ou mouvement. Le concept, plus formel, d'activité est emprunté au domaine de la simulation [27] ; il est reconnu, défini et utilisé dans plusieurs domaines scientifiques, dont la physique, la biologie, l'économie, l'épistémologie et bien naturellement l'informatique. L'article [20] fait la synthèse des différents domaines et propose une définition de l'activité dans chacun de ces contextes.

Activité en simulation

> An activity is an operation that transforms the state of a system over time. It begins with an event and ends by producing another event (linked to the termination of the activity). **Muzy et al.** [20]

Une *activité*, terme dénombrable dans ce cas, est une opération transformant l'état d'un système au cours du temps. Elle commence avec un évènement et termine en produisant un autre évènement (lié à la fin de cette l'activité). Un *évènement* est ce qui change l'état d'un système (éventuellement composé de nombreux composants). Pour finir, on appelle un *processus* une séquence d'activités et d'évènements ordonnés dans le temps. On peut remarquer que le terme « évènement » est central dans cette définition des activités. De plus les activités sont des objets plutôt que la mesure d'une quantité (la quantité d'activité).

Activité dans DEVS Deux définitions de l'activité pour les systèmes à évènements discrets (Discrete-Event Systems ou DEVS) sont proposées :

- L'activité *qualitative* : un système est qualitativement inactif quand aucun évènement n'a lieu et est qualitativement actif sinon.
- L'activité *quantitative* : la somme des activités quantitatives interne et externe est égale à la valeur de l'activité quantitative, pendant un certain temps de simulation. Les évènements discrets peuvent être de deux types : interne ou externe au modèle atomique

(endogène ou exogène). L'*activité quantitative interne* correspond au nombre d'évènements discrets internes, pendant un certain temps de simulation. L'*activité quantitative externe* correspond au nombre d'évènements discrets externes, pendant un certain temps de simulation. L'activité externe fournit une information sur la quantité de messages échangés par les modèles atomiques.

Activité du vivant En biologie, l'activité semble être liée au vivant, à la propriété d'être vivant. Une cellule est dite active si elle est *vivante*, ce qui ramène à la question de la définition du vivant. À l'échelle infra-cellulaire, cette définition fonctionne moins bien : il suffit de considérer chaque partie d'une cellule et tenter de déterminer si elle est vivante ou pas. Pour résoudre ce problème, la définition d'activité que l'on souhaite considérer est celle du point de vue de la chimie, à l'échelle moléculaire :

> [...] any biological object which is the locus of both exchanges (gas, liquids, nutrients, molecules, etc.) and transformations (chemical transformations), which result in maintaining the integrity of the object, is declared active.
> **Muzy et al.** [20]

Un objet biologique est dit *actif* s'il est à la fois un locus d'échanges (gaz, liquides, nutriments, molécules, ...) et de transformations (transformations chimiques) et que ces échanges et transformations assurent la persistance de cet objet.

Avec ces deux définitions arrivent les problèmes de *quantification* de l'activité dans les systèmes biologiques ; les mesures sont indirectes et les observables, c'est à dire les paramètres mesurables, sont nombreuses. En effet, au delà de l'influence même de la mesure sur ces systèmes, parfois seules des mesures de paramètres globaux, comme la quantité de CO_2 et de O_2 échangé entre le système et l'extérieur, sont accessibles. Autre exemple, pour connaître l'activité d'un réseau de neurones dans le cerveau, les chercheurs et les médecins ont recourt à l'imagerie par résonance magnétique fonctionnelle (IRMf) qui permet de connaître les zones du cerveau qui sont irriguées par du sang oxygéné. Dans les deux cas les mesures sont indirectes : elles sont plus ou moins globales et ne permettent pas de connaître l'activité de chacune des parties du système, de chaque cellule ou de chaque partie d'une cellule. De plus, de par la redondance des systèmes biologiques, une même sous-partie peut avoir plusieurs fonctions et une même fonction peut-être assurée par des sous-parties différentes dans un même système. Il est ainsi difficile de déterminer quelle partie du système a contribué au fonctionnement global et dans quelle proportion.

Activité économique L'activité est au cœur des sciences économiques de part l'objet d'étude de cette discipline. En effet l'économie est une science qui se focalise sur la manière dont des ressources rares sont utilisées dans le but de satisfaire les besoins humains. Elle s'intéresse par conséquent aux opérations de production, de distribution et de consommation des biens d'une part, aux institutions et aux structures dont le but est de faciliter ces opérations d'autre part. Autrement dit, l'activité correspond à « ce que les acteurs font » ; l'activité économique est donc un phénomène impactant une ressource rare.

Deux perspectives sur l'activité cohabitent :

- l'économie (néo)classique se penche sur les conséquences de l'activité, c'est à dire sur l'utilité de la situation qui en découle et non pas sur les décisions prises par les acteurs. Ces décisions sont prises à la suite d'un calcul rationnel, soit seul (*substantive rationality*), soit en groupe après délibération (*procedural rationality*), en fonction de la valeur de gain liée à ce choix ;
- l'économie comportementale (*behavioral economics*) est plutôt portée sur la manière dont les acteurs choisissent leurs activités, comment ils choisissent ce qu'ils vont faire. Leurs choix sont dans ce cas dictés par une règle préétablie (*rule rationality*), un mode de comportement qui incorpore toutes les situations concernées par cette règle.

On remarque que dans chacune de ces perspectives on *ignore* l'activité en elle-même ; la valeur de l'utilité, conséquence d'un choix d'activité, est mise en avant dans le premier cas, le choix du type d'activité lui-même est mis en avant dans le second cas. En économie, établir la mesure de l'activité ne suscite que peu d'intérêt et *reste un défi à relever*. Par analogie avec la définition de l'activité quantitative dans le cas des systèmes à évènements discrets, on donne la définition suivante : l'activité est le décompte du nombre d'évènements concernant des ressources rares. Un exemple d'usage prospectif est donné dans la section intitulée « Optimal Control Model of Activity » de [20].

Pour conclure cette section sur l'activité, nous émettons les remarques suivantes :

- Les différentes définitions de l'activité suivant les domaines d'études ne se recouvrent pas exactement, néanmoins dans chaque cas l'activité, ou les activités, nous offrent une mesure sur le système étudié ;
- La mesure d'activité donne une représentation abstraite du système étudié, une mesure agrégée qui *appartient à un autre niveau de représentation que le système* lui-même.

Nous verrons dans la suite de ce chapitre qu'il est possible de définir un nouveau type d'activité, l'activité spatiale, dans le contexte de MGS et que cette activité nous apportera une représentation du système utile à la fois pour la simulation et pour la modélisation.

3.3 Complexe cellulaire abstrait

Les développements présentés dans la suite de ce chapitre reposent sur une classe d'objets mathématiques nommés *complexes cellulaires abstraits*, une description formelle et abstraite de l'espace qui sert de fondement aux collections topologiques de MGS ainsi qu'à la définition de l'activité spatiale. Cette section présente le formalisme et quelques propriétés associées.

Afin d'effectuer des calculs dans l'espace, il peut être pratique de découper cet espace en éléments primordiaux, possédant une dimension, « accolés » les uns aux autres. Nous appelons ces éléments primordiaux des *cellules topologiques* dont l'assemblage combinatoire forme un

complexe cellulaire abstrait (CCA). Une p-cellule est une cellule topologique de dimension p. Par exemple, les 0-cellules sont des points, les 1-cellules sont des arcs, les 2-cellules sont des surfaces, les 3-cellules des volumes, ... Les cellules topologiques sont « jointes » par une relation appelée *relation d'incidence*. Plus formellement, il s'ensuit les définitions correspondantes.

Définition 3.3.1. *Soit S un ensemble de symboles appelés* cellules topologiques *muni d'un ordre partiel localement fini $\cdot \preceq \cdot \subset S \times S$ appelé* relation d'incidence. *Le couple $\mathscr{K} = (S, \preceq)$ est appelé* complexe cellulaire abstrait.

Définition 3.3.2. *Soit $\mathscr{K} = (S, \preceq)$ un complexe cellulaire abstrait. La fonction $dim_{\mathscr{K}} : S \to \mathbb{N}$ telle que pour $\sigma_1, \sigma_2 \in S$, $\sigma_1 \prec \sigma_2 \Rightarrow dim_{\mathscr{K}}(\sigma_1) < dim_{\mathscr{K}}(\sigma_2)$ est appelée fonction de* dimension.

Dans la suite de ce manuscrit, on notera dim au lieu de $\dim_{\mathscr{K}}$ quand il n'y a pas d'ambigüité sur $\mathscr{K}$. De plus, on notera parfois $\sigma \in \mathscr{K}$ au lieu de $\sigma \in S$.

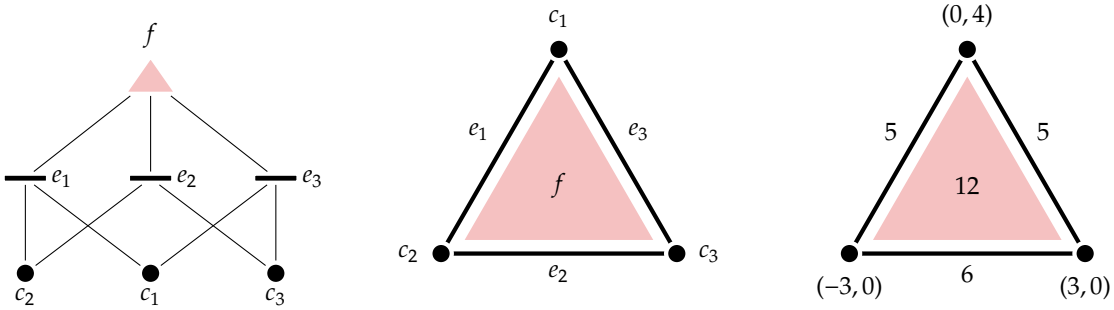


FIG. 3.1: Diagramme de Hasse (à gauche) de la relation d'incidence dans le complexe cellulaire du milieu. Ce dernier est constitué d'une 2-cellule f, de trois 1-cellules (e_1, e_2, e_3) et de trois 0-cellules (c_1, c_2, c_3). Les trois arcs sont les faces de f et par définition, f est une coface de e_1, e_2 et e_3. Ce complexe est décoré par des valeurs arbitraires (à droite) comme des coordonnées pour les points, des distances pour les arcs et une aire pour la surface.

L'opération ensembliste *union* est naturellement étendue aux complexes cellulaires : soient $\mathscr{K}_1 = (S_1, \preceq_1)$ et $\mathscr{K}_2 = (S_2, \preceq_2)$ deux complexes cellulaires, $\mathscr{K}_1 \cup \mathscr{K}_2 = (S_1 \cup S_2, \preceq_1 \cup \preceq_2)$ si $\dim_{\mathscr{K}_1}$ et $\dim_{\mathscr{K}_2}$ coïncident sur $S_1 \cap S_2$.

Définition 3.3.3. *Soient $\mathscr{K}_1 = (S_1, \preceq_1)$ et $\mathscr{K}_2 = (S_2, \preceq_2)$ deux complexes cellulaires abstraits. On dit que $\mathscr{K}_2$ est un sous-complexe de $\mathscr{K}_1$, noté $\mathscr{K}_2 \subset \mathscr{K}_1$, si :*

- $S_2 \subset S_1$
- $\preceq_2 \subset \preceq_1$ *et*
- $dim_{\mathscr{K}_1}$ *coïncide avec* $dim_{\mathscr{K}_2}$ *sur* S_2.

Définition 3.3.4. *Soit $\mathscr{K} = (S, \preceq)$ un complexe cellulaire abstrait et $\sigma \in \mathscr{K}$ une cellule. L'ensemble des* faces *de σ est l'ensemble :*

$$\{\tau \in K \mid \tau \prec \sigma \wedge dim(\tau) = dim(\sigma) - 1\}$$

et σ est appelée la coface *de τ.*

Nous définissons ci-dessous trois opérateurs (fermeture, étoile et liaison) sur les CCA utilisant la relation d'incidence ; ils sont extraits des travaux d'Axen [1]. La fermeture d'un CCA « sélectionne » toutes les cellules adjacentes de dimension inférieure, à l'inverse l'étoile d'un CCA « sélectionne » toutes les cellules adjacentes de dimension supérieure. L'opérateur liaison capture les cellules en lien avec les cellules d'un CCA soit par une cellule de dimension supérieure, soit par une cellule de dimension inférieure. La figure 3.2 présente une vue intuitive du fonctionnement de ces opérateurs. Les définitions suivantes les introduisent plus formellement.

Définition 3.3.5. *Soit $\mathscr{K}$ un complexe cellulaire abstrait et $S \subset \mathscr{K}$. On nomme* fermeture *de S et on note $\overline{S}$ l'ensemble $\{\sigma \in \mathscr{K} \mid \exists \tau \in S, \sigma \preceq \tau\}$. Symétriquement, on nomme* étoile *de $S \subset \mathscr{K}$ l'ensemble $\{\sigma \in \mathscr{K} \mid \exists \tau \in S, \tau \preceq \sigma\}$. On note la fermeture de l'étoile $\overline{St}S = \overline{St\,S}$.*

Définition 3.3.6. *Soit $\mathscr{K}$ un complexe cellulaire abstrait et $S \subset \mathscr{K}$. L'opérateur de* liaison *(*Link *en anglais et noté par conséquent Lk dans les travaux de [1]) est défini ainsi : $Lk\,S = \overline{St\,S} - St\,\overline{S}$, où $-$ est l'opérateur de soustraction classique de la théorie des ensembles.*

3.4 MGS : un langage dédié à la modélisation et à la simulation des (DS)²

MGS est un langage dédié à la modélisation et à la simulation des (DS)². C'est un langage de programmation spatial : un calcul consiste en un déplacement, induit par la relation de voisinage, dans l'espace abstrait de la structure de donnée (appelée *collection topologique*) et aussi en une action sur cette structure (appelée *transformation*) dépendant du calcul. Dans ce but, MGS embarque les concept de collection topologique et de transformation dans le framework d'un langage dynamiquement typé[1] et fonctionnel[2]. Les collections topologiques sont les seules structures de données disponibles dans le langage, c'est à dire qu'il n'existe pas d'autre manière d'agréger des données par rapport à une certaine relation de voisinage dans MGS. Les transformations, définies par une syntaxe spécifique à base de règles de réécriture, sont des fonctions définies par cas agissant sur ces collections.

3.4.1 Topologie des interactions

Pour décrire l'évolution spatiale d'un processus, d'un élément ou d'une sous-partie d'un système, il existe deux alternatives classiques :

- un point de vue spatial, dans lequel on considère ces mouvements par rapport à un espace préexistant, fixé. Dans le cas d'un système proie-prédateur par exemple, chaque

1. Dans notre contexte, *dynamiquement* typé signifie qu'il n'y a pas de vérification statique des types et que les erreurs de type sont détectées pendant l'exécution lors de l'évaluation d'une expression.

2. MGS est un langage de programmation applicatif : les opérateurs combinent les valeurs sur lesquels ils s'appliquent pour donner des nouvelles valeurs, ils ne produisent pas d'effet de bord.

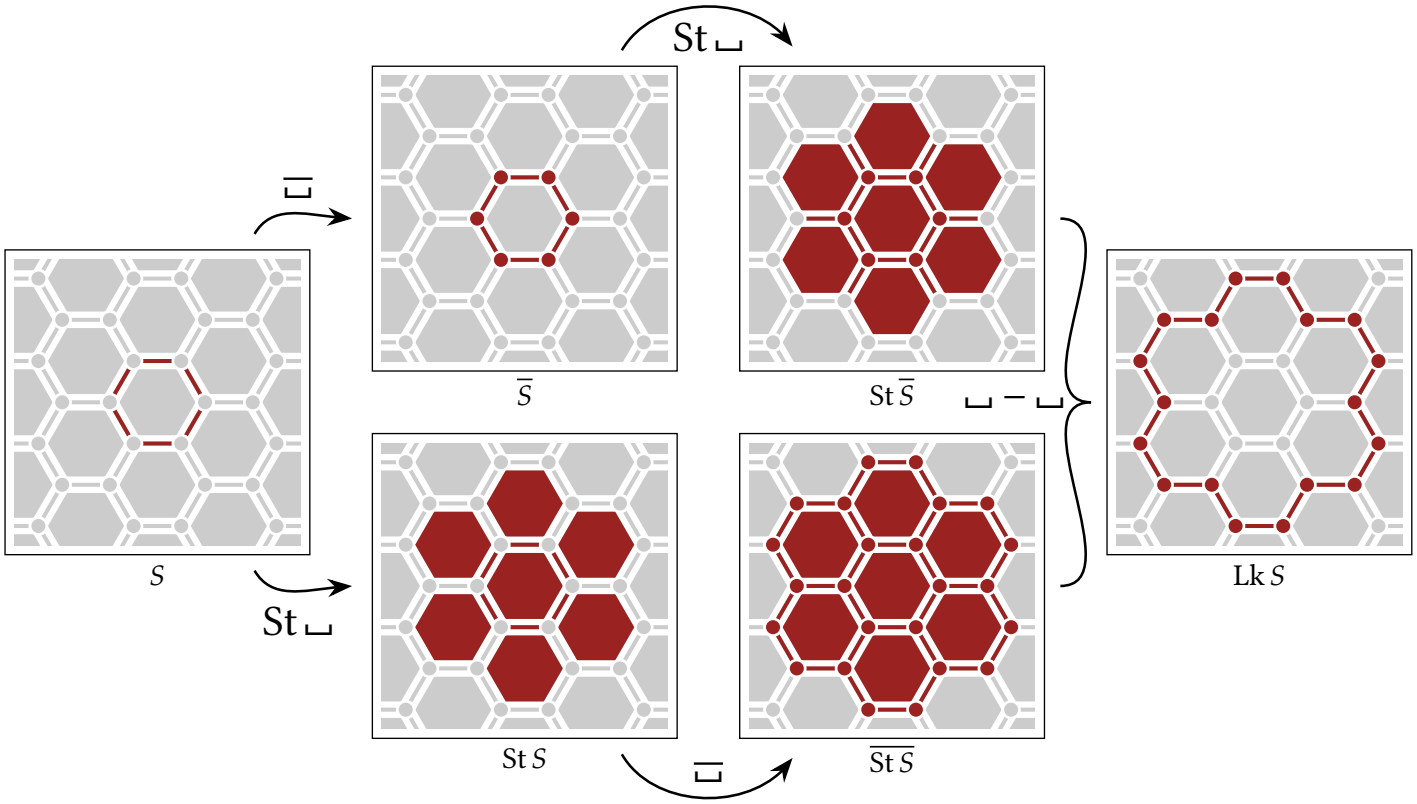


FIG. 3.2: Construction de la liaison (à droite) d'un sous-complexe $S \subset \mathscr{K}$ (à gauche). En passant par le haut, on applique successivement l'opérateur fermeture suivi d'étoile, en passant par le bas c'est d'abord étoile suivi de fermeture. La liaison de S, Lk S, s'obtient par la différence entre $\overline{\text{St}\,S}$ et St $\overline{S}$.

position de l'espace peut-être occupée soit par une proie, soit par un prédateur, soit vide ;

- un point de vue agent, focalisé sur l'individu. Ce dernier émet et reçoit des messages d'autres agents pour interagir avec son environnement.

Cependant, ni l'un ni l'autre de ces points de vue n'est satisfaisant essentiellement parce qu'ils se concentrent chacun sur l'évolution locale d'une unique entité. Par exemple, dans le cas d'un problème de collision de particules dans un automate cellulaire un choix se fait entre une évolution en deux étapes [28] (propagation et collision) et l'utilisation d'un gaz sur réseau [5], une variante d'automate cellulaire qui considère une évolution couplée de plusieurs cellules. Du point de vue agent, le problème de synchronisation entre plusieurs agents amène aussi au développement de stratégies plus flexibles [17, 18, 19]. Le parti pris dans MGS est de dépasser ces deux points de vue en se recentrant sur ce qui créée la dynamique d'un système : les *interactions* entre entités (qu'ils soient agents ou morceaux d'espace). Les interactions représentent l'évolution simultanée d'une sous-partie (souvent petite) des entités composant le système.

Supposons maintenant qu'à chaque instant de l'évolution d'un $(DS)^2$, seules quelques entités interagissent entre elles. Isoler ces groupes d'éléments en interaction revient à partitionner le système en groupes d'interaction unitaires (ou atomes) n'étant pas en relation entre eux *à l'instant courant* comme illustré par la figure 3.3. Cette partition forme un treillis d'atomes qu'il est possible et intéressant de voir comme une topologie : c'est la *topologie des interactions* [7, 8]. Ce point de vue est incarné par les collections topologiques de MGS.

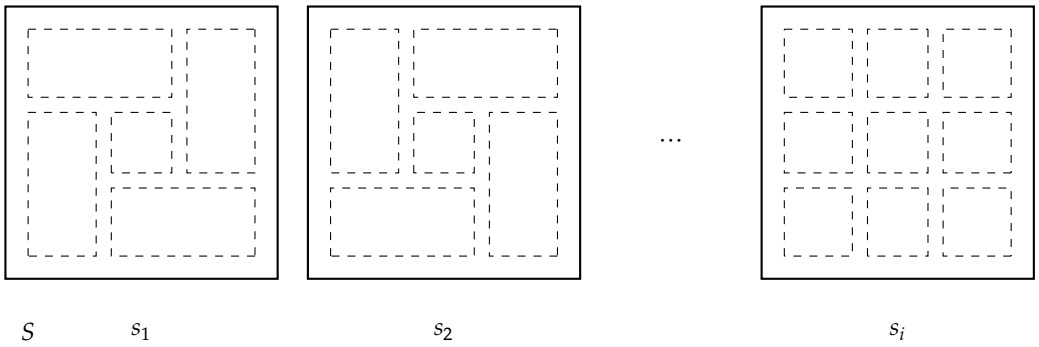


FIG. 3.3: Les partitions successives du système S forment une topologie, c'est la topologie des interactions

3.4.2 Collections topologiques

Une collection topologique est une *structure de données générique* se fondant sur la topologie des interactions. Dans le paradigme adopté par MGS, l'espace où évolue un phénomène, l'espace de modélisation, est spécifié à partir des interactions possibles entre les composantes du système à modéliser. Le fondement formel des collections topologiques est un objet mathématique appelé *complexe cellulaire abstrait* [8], introduit dans la section 3.3. Présentons

d'abord quelques structures de données habituelles et leur implémentation sur des complexes cellulaires abstraits.

Tableau Un tableau est un ensemble compact de cases accolées les unes aux autres par un de leur bord. En 1D un tableau t est définit par sa longueur n et on le note par conséquent $t[n]$. À l'instar du modèle mémoire du langage C, un tableau est une sous-partie de la *mémoire*. La structure de la mémoire peut être représentée par un complexe cellulaire constitué d'un nombre non borné de cellules où chaque « case » est une 0-cellule $\forall i \in \mathbb{N}\ c_i$. Nous représenterons le fait que deux 0-cellules (c_i, c_{i+1}) soient voisines par une 1-cellule $d_{i,i+1}$ de sorte que $\forall i \in \mathbb{N}\ c_i \preceq d_{i,i+1}$ et $c_{i+1} \preceq d_{i,i+i}$. Un tableau est une sous-partie de la mémoire où des 0-cellules consécutives sont décorées par n valeurs différentes de la valeur spéciale *null*.

Dans MGS, cette structure de donnée est directement disponible sous le type `'array`. La syntaxe pour construire un tableau de trois cases contenant les valeurs $1, 2, 3$ est

```
1, 2, 3, ():'array
```

Liste Une liste chaînée est une autre structure de données apparemment proche des tableaux où chaque élément d'une liste a soit un élément successeur, soit *n'en a pas*. Cette structure de données peut être représentée par un complexe cellulaire abstrait où chaque élément i est une 0-cellule c_i. Nous représenterons le fait que deux cellules (c_i, c_j) soient voisines par une 1-cellule $d_{i,j}$ de sorte que $c_i \preceq d_{i,j}$ et $c_j \preceq d_{i,j}$. Dans MGS, cette structure de donnée est appelée séquence, son type est `'seq`. La syntaxe pour construire une séquence de trois éléments de valeurs $1, 2, 3$ est

```
1, 2, 3, ():'seq
```

La différence fondamentale entre liste et tableau tient au fait qu'il n'est pas possible de retirer une case dans un tableau : sa structure est fixée, seules ses valeurs peuvent être changées. Ces deux structures de données ont la même représentation structurelle mais leur comportement diffère quant à la sémantique de l'ajout et du retrait de valeurs décorant les cellules du complexe sous-jacent. Cette remarque prendra du sens dans la section 3.4.3 consacrée aux transformations.

GBF Les *Group-Based Field* (GBF) [9, 24] sont une collection topologique fondée sur les graphes de Cayley, des graphes qui encodent la structure d'un groupe. Plus particulièrement, les GBF sont issus de graphes de Cayley de groupes Abélien, car ils ont un structure de grille. En pratique, ils représentent les espaces où les déplacements sont commutatifs comme les grilles carrées ou hexagonales où chaque générateur de la présentation d'un groupe indique une direction. La syntaxe MGS pour construire une collection de type `'gbf` représentant une grille hexagonale est

```
gbf hexa = < a, b, c | a + b = c >
```

La collection `hexa` possède trois opérateurs de voisinage notés `|a>`, `|b>` et `|c>`, ainsi que leurs opposés `<a|`, `<b|`, `<c|`. Nous utiliserons cette collection plus tard dans le chapitre 3.6 pour les simulations de la propagation d'un feu de forêt.

Chaînes topologiques La collection la plus générale de MGS est la *chaîne topologique*, dans le sens où elle permet la manipulation plus libre de la structure des collections topologiques. On définit une collection de ce type en construisant un arbre de cellules, munies d'une valeur arbitraire, reliées par leur incidence.

```
v1 := new_dvertex(‘a) ;;
v2 := new_dvertex(‘b) ;;
e1 := new_edge(v1,v2) ;;
```

`v1`, `v2` et `e1` sont trois valeurs de type `‘dcell` ; `v1` et `v2` sont des 0-cellules et `e1` est une 1-cellule.

Ces quelques exemples permettent de se faire une première idée sur le fonctionnement des collections topologiques et la manière dont elle sont implémentées. Le lecteur intéressé par une présentation plus formelle pourra se référer au manuscrit de thèse d'Antoine Spicher [23] traitant en détail du sujet.

3.4.3 Transformations topologiques

Une transformation topologique est une fonction définie par cas sur les collections topologiques : elle prend en argument une collection topologique et retourne en sortie une collection topologique du même type en effectuant le remplacement d'une sous-collection de la collection fournie en paramètre. Cette opération, par analogie avec la réécriture de chaînes de caractère, est appelée *réécriture topologique* du fait que modifier une collection topologique permet de modifier aussi bien les données que leur structure. La syntaxe d'une transformation est

```
trans NomDeLaTransformation = {
  motif_1 => expression_1;
  [...]
  motif_i => expression_i;
  [...]
  motif_n => expression_n;
}
```

À chaque motif de filtrage `motif_i` correspond son expression `expression_i` qui sera évaluée, puis « recollée » dans la collection topologique d'entrée. Un motif de filtrage prend la forme d'une succession d'éléments séparés par l'opérateur de voisinage « `,` », dont la sémantique est la même *quelle que soit la collection.*

Le temps du modèle est déterminé par l'ordre et les conditions d'application des règles données dans une transformation. Est-ce que les motifs doivent être filtrés successivement (temps asynchrone), simultanément (temps synchrone) ou de manière probabiliste ? Le modèle détermine quelle stratégie adopter. Il existe dans MGS différentes *stratégies* d'application des règles, dont les suivantes ont été utilisées dans nos travaux :

- `'default` (Maximal-Parallel) : les règles sont appliquées successivement jusqu'à ce que cela ne soit plus possible ;
- `'asynchronous` (Asynchrone ou Séquentielle) : simule une évolution asynchrone, une seule règle est appliquée une fois, et l'ordre des règles fixe la priorité ;
- `'gillespie` (Gillespie) : chaque règle décrit une réaction et se voit accorder une constante de réaction, par exemple $C = 0{,}01$,

  ```
  motif_i ={ C = 0.01 }=> expression_i;
  ```

  et est appliquée en fonction de cette constante en suivant la méthode de calcul stochastique proposée par Gillespie [10].

Comme pour la section précédente, cette courte présentation est fournie pour donner au lecteur une intuition sur le fonctionnement des transformations topologiques. Dans le reste du chapitre, un certain nombre d'exemples plus complets seront fournis. Le manuscrit [23] traite en détail du sujet.

3.5 L'activité dans MGS

Nous nous intéressons au concept d'*activité* dans MGS. Comme nous l'avons vu dans la section 3.5, l'activité est une mesure du nombre d'évènements ou de changement d'état dans une simulation. Cette mesure peut être utilisée pour différents objectifs, comme optimiser une simulation ou bien avoir une meilleur compréhension de sa dynamique. Dans le contexte de MGS, l'activité nous permet

1. de développer un meilleur algorithme de filtrage par motifs et,
2. d'identifier des structures d'ordre supérieur dans un modèle ainsi que de les suivre lors des simulations.

Dans cette section nous présentons une méthode générique pour suivre une région active durant la simulation.

Pour des raisons de simplicité, nous nous restreignons aux motifs ne comportant au maximum que deux éléments en interaction. Nous illustrons cette idée avec un exemple simple mais paradigmatique : un modèle de propagation de feu de forêt [22] où le *feu* se propage au travers d'une *forêt* en laissant derrière lui de la matière calcinée sous la forme de *cendres*. Ce modèle peut être décrit par un automate cellulaire à trois états que l'on peut aisément écrire dans une transformation MGS:

```
trans fire_spread = {
  'Forest as x / member('Fire, neighbors x) => 'Fire;
  'Fire                                     => 'Ashes;
}
```

Les états sont représentés par trois symboles : `'Forest` pour la forêt, `'Fire` pour le feu et `'Ashes` pour les cendres. La première règle spécifie comment un élément de forêt prend feu avec pour

voisin un élément de feu, la seconde règle spécifie comment un élément de feu s'éteint une fois la combustion terminée et devient un élément de cendre. La figure 3.4 montre trois pas successifs de l'évolution de cet automate sur une collection topologique à grille carrée.

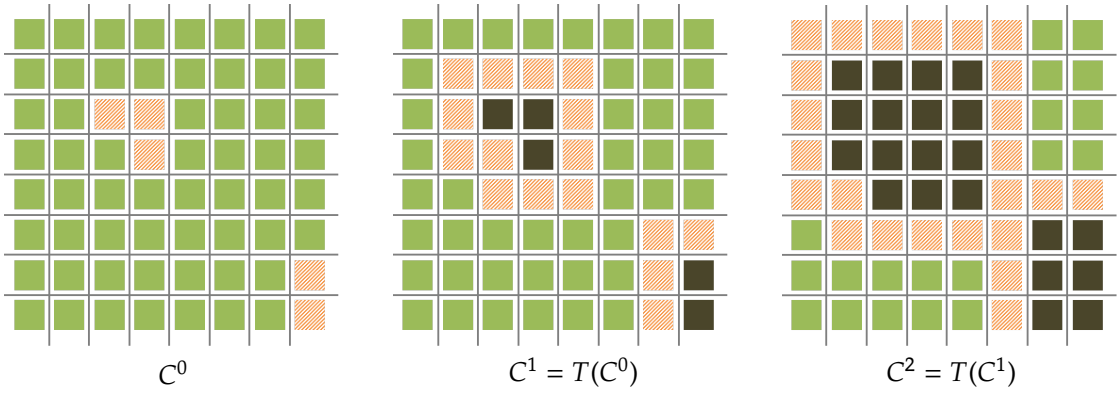


FIG. 3.4: Trois premiers pas de la simulation de la propagation d'un feu de forêt. Les cellules vertes (ou gris foncé) sont des cellules de forêt, les cellules oranges (hachurées) représentent le feu et les cellules foncées représentent les cendres.

Dans cet exemple, l'activité est située sur une sous-partie du domaine – seuls les éléments de feu et leur cellules voisines évoluent – et progresse de proche en proche. Néanmoins, l'algorithme de filtrage de motif de MGS doit itérer sur toute la collection à chaque application de la transformation `fire_spread`. Étant donné le faible nombre de cellules en interaction, le processus de filtrage est ici assez peu efficace. Par la suite, nous chercherons à cibler le filtrage sur les cellules qui évoluent grâce au mécanisme de filtrage. De plus, l'activité met en avant une structure de niveau supérieur importante de la propagation d'un feu de forêt : le *front de propagation*.

Activité et interactions L'interprétation de l'activité dans MGS correspond au nombre d'interactions ayant lieu à chaque pas de la simulation. Puisque les interactions arrivent à chaque fois qu'une règle de la transformation filtre une sous-collection, on constate que l'activité sépare naturellement une collection en deux parties :

- une sous-collection *active* où une interaction peut avoir lieu et
- une sous-collection *quiescente* où aucune interaction ne peut avoir lieu.

Définissons les sous-collections actives et quiescentes d'une manière formelle et générale dans le cadre des collections topologiques. Soit C un ensemble appelé collection topologique et T une transformation. On définit la *fonction de filtrage* $\mathbb{M}_T : C \to \mathscr{P}(C)$ de la transformation T comme la fonction qui à C fait correspondre l'ensemble des sous-collections de C filtrées par un motif de T. En d'autres termes, la fonction de filtrage $\mathbb{M}_T$ représente un appel au processus

de filtrage de motif. La *sous-collection active* A de C est ainsi définie par

$$A = \bigcup_{S \in \mathbb{M}_T(C)} S$$

c'est à dire l'assemblage[3] de toutes les sous-collections filtrées de C. Ainsi, la *sous-collection quiescente* Q corresponds exactement au complémentaire de A dans C :

$$Q = C \setminus A$$

La figure 3.5 montre la décomposition d'une collection en deux sous-collections actives et quiescentes dans le cas de l'exemple de propagation du feu de forêt.

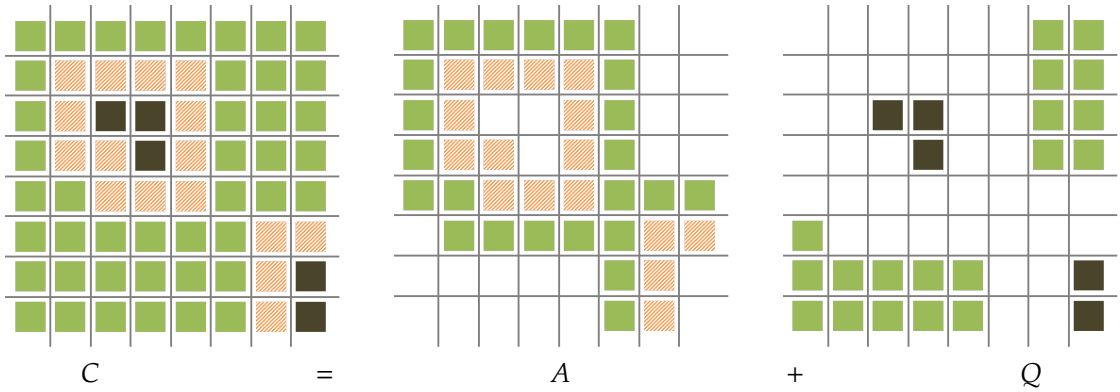


FIG. 3.5: Décomposition d'une collection topologique en deux ensembles de cellules actives et quiescentes. En vert (gris foncé), orange (hachuré) et foncé on représente la forêt, le feu et les cendres, respectivement.

Dynamique de l'activité Afin de suivre une région active au cours de la simulation d'un modèle, nous définissons ci-après une manière de calculer l'évolution d'une région active (agrandissement, rétrécissement, creusement, etc.) reposant sur les propriétés topologiques de la sous-collection active.

La modélisation d'un système naturel implique souvent que des phénomènes apparaissent à des échelles temporelles et spatiales distinctes : chaque niveau est dédié à un phénomène se déroulant dans une fenêtre de temps et d'espace particulière. Ces échelles apparaissent pour des raisons logiques (à une certaine échelle, un système présente des propriétés uniformes et peut être modélisé par des règles homogènes agissant sur des objets pertinents à cette échelle) ou bien pour des raisons d'efficacité, par exemple la simulation réductionniste de l'intégralité du système à partir des premiers principes est difficile d'un point de vue calculatoire alors

3. L'opérateur $\bigcup$ est utilisé à la place de + car certaines sous-collections filtrées par $\mathbb{M}_T$ peuvent avoir des supports qui se recouvrent.

qu'une description plus grossière du modèle est bien souvent satisfaisante. Les modèles et simulations à plusieurs échelles sont considérés lorsque des interactions apparaissent entre différentes échelles. Pour les échelles spatiales, on doit considérer simultanément plusieurs représentations spatiales comme dans le *raffinement de maillage* [4]. Cette méthode repose sur une séquence de « grilles rectangulaires entrelacées » sur lequel une EDP est discrétisée. Il est important de noter que ces sous grilles ne sont pas incorporées dans la grille principale mais superposées pour atteindre le but recherché. Un autre exemple, dans le domaine de la modélisation discrète, est la classe *automate complexe* [13] qui correspond à un « graphe d'automates cellulaires ».

Considérons $(C^0, C^1, \dots)$ la trajectoire des collections par l'application successive de la transformations T, où $C^{i+1} = T(C^i)$ pour $i \geq 0$. En utilisant la décomposition active-quiescente sur $C^i = A^i + Q^i$, on peut remarquer que l'application de la transformation agit seulement sur la partie active et épargne Q^i de telle sorte que le filtrage de motif peut n'être utilisé que sur A^i :

$$C^{i+1} = T(C^i) = T(A^i \mid F^i) + Q^i \tag{3.1}$$

La notation $T(A^i \mid F^i)$ est introduite pour indiquer que le mécanisme de filtrage pourrait nécessiter des informations de Q^i, dans le cas où il est nécessaire de visiter le voisin d'un élément filtré. Par exemple, dans la précédente transformation `fire_spread`, `member('Fire, neighbors x)` n'implique pas que les voisins visités sont dans A^i. Cette information, notée F^i, correspond à la sous-collection dont le support est constitué de tous les éléments de Q^i ayant un voisin dans A^i :

$$F^i = \mathrm{Lk}\, A^i$$

L'opérateur *link* Lk sélectionne par construction l'ensemble de toutes les cellules voisines d'une cellule quelconque σ. Nous utilisons ici son extension naturelle à l'ensemble des cellules actives. La figure 3.6 montre un exemple d'application de Lk sur un complexe cellulaire.

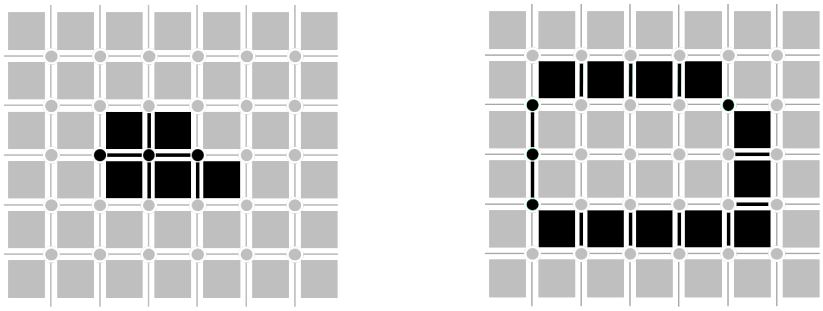


Fig. 3.6: Application de l'opérateur Lk (droite) sur un complexe cellulaire de dimension 2 (gauche).

La décomposition de C^{i+1} proposée dans l'équation (3.1) ne coïncide pas à la définition de A^{i+1} et Q^{i+1}. En fait, quelques cellules de Q^i peuvent devenir actives et *vice versa*. Néanmoins, le comportement de la frontière active-quiescente au cours de la simulation peut être raffiné

en utilisant la sous-collection F^i. En effet, en supposant que la règle de transformation n'implique au maximum que deux éléments voisins, une cellule quiescente, à une certaine itération de la simulation, sans voisins appartenant à la sous-collection active aura un environnement inchangé et par conséquent restera dans son état courant à l'itération suivante. Formellement, on a :

$$Q^i - \mathrm{Lk}\, A^i \subset Q^{i+1} \qquad A^{i+1} \subset T(A^i \mid \mathrm{Lk}\, A^i) + \mathrm{Lk}\, A^i \tag{3.2}$$

La seconde inclusion est obtenue en utilisant la complémentarité et signifie de manière équivalente que la partie active ne peut grandir au delà de F^i. L'équation (3.1) se réécrit donc

$$C^{i+1} = \left[T(A^i \mid \mathrm{Lk}\, A^i) + \mathrm{Lk}\, A^i\right] + \left[Q^i - \mathrm{Lk}\, A^i\right] \tag{3.3}$$

Un filtrage de motif optimisé Considérant l'équation (3.2), il est clair que l'équation (3.3) est une sur-approximation de la décomposition active-quiescente de C^{i+1}. En exemple, sur la figure 3.7, les lignes en gras représentent la limite d'expansion de la partie active au prochain pas de simulation et certaines cellules de cette sous-collection deviennent quiescentes. Ces cellules peuvent être identifiées comme les cellules de $T(A^i \mid \mathrm{Lk}\, A^i) + \mathrm{Lk}\, A^i$ ne pouvant être impliquées dans aucune des interactions au pas $i+1$. Autrement dit, elles ne sont pas sélectionnées par le mécanisme de filtrage :

$$\mathbb{M}_T(C^{i+1}) = \mathbb{M}_T(T(A^i \mid \mathrm{Lk}\, A^i) + \mathrm{Lk}\, A^i)$$

et le calcul de la séquence $(A^i)_{i\in\mathbb{N}}$ suit

$$\begin{cases} A^0 & = & \bigcup_{S\in\mathbb{M}_T(C^0)} S \\ A^{i+1} & = & \bigcup_{S\in\mathbb{M}_T(T(A^i \mid \mathrm{Lk}\, A^i)+\mathrm{Lk}\, A^i)} S \end{cases}$$

Il est important de noter que cette définition de A^i ne se réfère aucunement à la sous-collection quiescente Q^i, car l'opérateur de voisinage Lk sélectionne exactement les cellules nécessaires. En conséquence, suivre l'activité durant la simulation nous permet de nous concentrer sur une partie réduite de la collection lors du filtrage. L'optimisation induite est discutée et illustrée dans les deux parties suivantes sur des exemples plus complexes.

Caractérisation topologique L'étude précédente peut être généralisée aux transformations impliquant plus de deux éléments en interaction en reconsidérant la distance à laquelle la sous-collection active peut s'étendre à chaque itération de la simulation.

Soit r le *rayon* d'interaction, *i.e.*, la distance minimale (en terme de saut) entre éléments en interaction. La restriction précédente consistait à considérer les transformations de rayon $r = 1$; considérons maintenant un rayon arbitraire $r \in \mathbb{N}$.

L'expansion de F^i_r de A^i pour un rayon r est donnée récursivement par

$$\begin{cases} F^i_0 & = & 0 \\ F^i_{r+1} & = & \mathrm{Lk}(A^i + F^i_r) + F^i_r \end{cases}$$

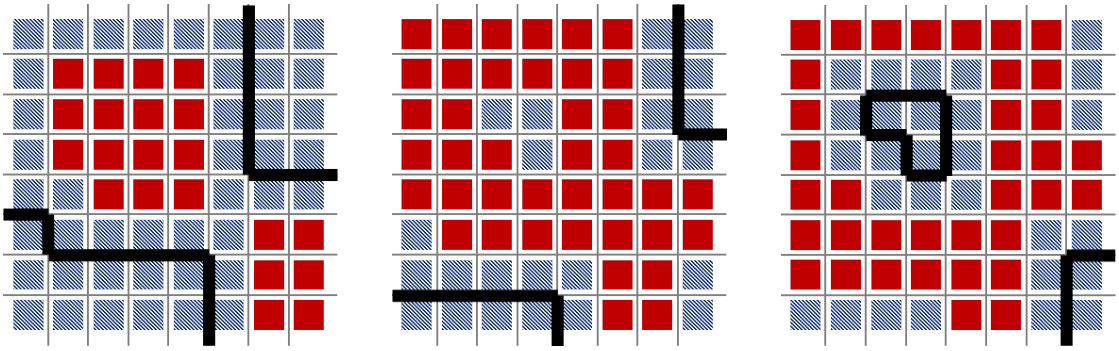


FIG. 3.7: Évolution de la frontière active-quiecsente pour les trois pas de la figure 3.4. Les cellules actives sont en rouge (gris foncé), les cellules quiescentes en bleu clair (hachuré). Les lignes en gras représentent la décomposition induite par l'équation (3.3).

Cette définition est analogue à la définition de l'*opérateur de vague* $W(n)$ dans [1] et révèle la nature topologique de l'activité. L'opérateur de vague est utilisé pour l'élaboration d'une théorie de Morse combinatoire. La théorie de Morse est un outil mathématique pour étudier la topologie des espaces. Brièvement, cette théorie traite des *fonctions de Morse* – une manière d'inonder l'espace avec un « liquide » – et des *points critiques* – où le liquide révèle des bassins, des cols, des îles dans la topographie de l'espace.

3.6 Propagation d'un feu de forêt

Dans cette section, nous illustrons les notions de collection topologique et de transformation sur l'exemple paradigmatique de la propagation d'un feu de forêt.

3.6.1 Un exemple canonique

Les incendies forestiers présentent des effets négatifs pour l'environnement. D'après l'organisation des nations unies pour l'alimentation et l'agriculture[4],

> [...] la destruction du couvert végétal par les incendies incontrôlés aggrave à la fois le réchauffement climatique, la pollution de l'air, la désertification et la perte de biodiversité.

Ces effets nocifs incitent à développer les techniques de lutte contre les incendies ainsi que leur gestion [11]. De nombreux modèles compliqués ont été établi dans le but de simuler la propagation d'un feu de forêt en temps réel comme [14, 21]. Ces modèles incluent des fonctionnalités supplémentaires comme un *système d'information géographique* [29], ...

4. http://www.fao.org/news/story/fr/item/29097/icode/

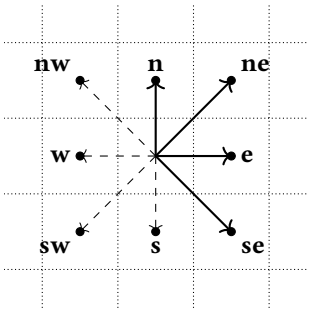


FIG. 3.8: Un GBF définissant un voisinage de Moore, avec quatre générateurs **e**, **n**, **ne** et **se** et deux contraintes **n** + **e** = **ne**, **e** − **n** = **se**. Les directions **w**, **s**, **sw** et **nw** sont définies comme les inverses des générateurs.

De plus, d'après [16], il existe différent types de modèle pour la propagation d'un feu de forêt, fondés soit sur la théorie des probabilités [15], sur une extension d'un modèle stochastique [12], sur le principe de Huygens [3] ou encore sur la simulation de l'écoulement d'un fluide [6, 26]. Remarquons que ces modèles sont soit imprécis soit trop consommateur en ressources [2]. De plus, aucun de ces modèle ne prend en compte l'intégralité des facteurs affectant la propagation d'un feu de forêt.

Afin de représenter l'activité spatiale dans un cadre un plus concret que l'exemple ad hoc précédent, nous avons implémenté un modèle simple, efficace et réaliste pour la propagation d'un feu de forêt utilisant le formalisme des automates cellulaires (AC) [16], avec le langage de programmation MGS. Notre modèle de propagation de feu de forêt prend en compte les principaux effets environnementaux, à savoir le vent (à la fois la force et la direction), le type de carburant et la topographie de la zone de propagation.

Le *front de l'incendie* est la zone séparant la forêt des cendres. Le principal objectif de notre modèle est de déterminer le prochain front de l'incendie en utilisant :

- le front de l'incendie actuel,
- la répartition des vitesses de propagation à chaque point de forêt,
- la force et la direction du vent ainsi que
- l'altitude de chaque point de la zone de propagation.

Représentation des états avec MGS

Chaque configuration de l'AC est représenté par une collection topologique de type GBF de MGS avec un voisinage de Moore. Par exemple, afin de définir une grille avec un voisinage de Moore, on utilise quatre générateurs nord (**n**), est (**e**), nord est (**ne**) et sud est (**se**) :

```
gbf Land = < n, ne, e, se ; 100 n = 0, 100 e = 0, ne = e + n, se = e - n > ;;
```

Les générateurs `s = <n|`, `sw = <ne|`, `w = <e|` et `nw = <se|` sont automatiquement définis car ils correspondent aux déplacements inverses sur la présentation d'un groupe abélien.

On associe ensuite une valeur à chaque cellule d'un GBF. Cette valeur est un dictionnaire contenant l'avancée de sa combustion (une valeur réelle), la direction du vent (un générateur de GBF), la vitesse de combustion (une valeur réelle) et l'altitude (une valeur entière) :

```
record cell = {
  burned_out     : float,
  wind_direction : position,
  burning_rate   : float,
  altitude       : int
} ;;
```

L'*avancée de la combustion* correspond à la surface de la cellule déjà consumée. Lorsque cette valeur atteint 1.0, cela signifie que l'intégralité de la cellule a été consumée et donc qu'elle est transformée en cendres.

Une cellule a sa propre vitesse de combustion qui peut être vue comme une mesure agrégée des caractéristiques du carburant comme son type, sa densité, etc.. L'effet de la vitesse de combustion est ensuite renforcé ou bien inhibé par d'autres facteurs environnementaux. La *direction du vent* et sa force peuvent accentuer ou inhiber la propagation du feu en fonction de la position du voisin et la direction du vent : si le voisin est dans la direction opposée au vent alors la cellule brûlera plus vite, sinon si le voisin est dans la direction du vent alors la cellule brûlera moins rapidement. Dans notre simulation, le vent peut soit être fort (effet marqué) ou faible (effet plus marginal). L'*altitude* des voisins peut jouer un rôle similaire : si le voisin est plus élevé alors son effet sur la vitesse du propagation du feu sera inhibé alors que s'il est plus bas alors son effet sera accentué (le feu a tendance à monter).

Spécification dynamique en MGS

Voici la fonction de transition MGS représentant l'évolution locale de chaque cellule de l'AC :

```
  x => (
    let b = neighborsfold_indexed((\p,y,acc.(
      x.burning_rate        *
      distance_effect(p,^x) *
      wind_effect(p,^x)     *
      slope_effect(y,x)     *
      y.burnt               + acc
    )),x.burnt,x) in
      x + { burnt = min(1.0,b) }
  );

} ;;
```

L'extrait de code précédent présente une transformation MGS constitué d'une unique règle de réécriture pour mettre à jour l'état de chaque cellule. La fonction `neighborsfold_indexed` est une fonction `fold` classique conçue pour itérer sur chaque voisin d'une cellule. Elle prend en entrée une fonction à trois paramètres, qui sont `p` la position de la cellule voisine, sa valeur `y` et un accumulateur `acc` des valeurs déjà réduites. Ainsi, cette simple fonction de transition

itère sur chaque cellule en mettant à jour l'avancée de la combustion à chaque fois en prenant en compte l'effet de la distance aux voisins, la force et la direction du vent et l'altitude des voisins. Aussi, de part leur plus grande distance, les cellules en diagonale ont un effet moins conséquent sur leurs voisines, limité à 83% (voir [16]).

3.6.2 Description de la propagation d'un feu de forêt du point de vue de l'activité

Nous avons réécrit l'exemple précédent afin que le mécanisme de filtrage ne s'applique que dans la zone active.

En accord avec l'approche théorique présentée un peu plus tôt, l'espace des cellules de l'AC est partagé entre les cellules actives qui prennent part à l'évolution de leurs voisines et les cellules quiescentes ne jouant aucun rôle à l'itération courante. Comme présenté à la fin de la section 3.5, l'activité se propage comme une vague, à l'instar du feu de forêt.

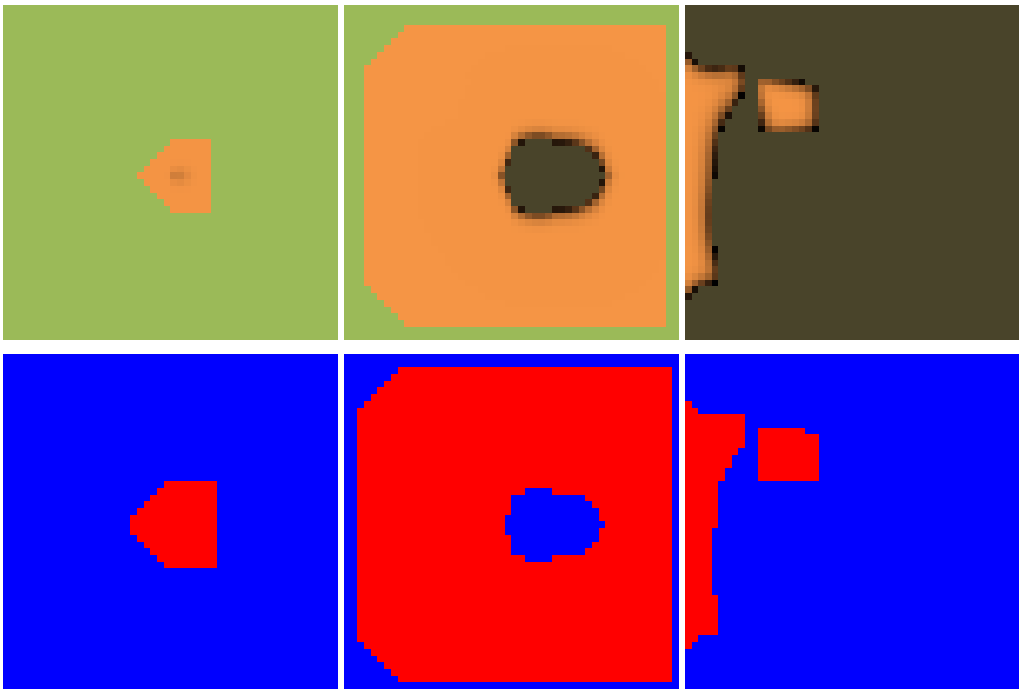


FIG. 3.9: Propagation du feu présentée à 3 itérations différentes, de gauche à droite, dans une simulation où les cellules enflammées sont initialement au centre d'une grille de taille 50×50. Dans la première série, les cellules de forêt sont en vert (gris clair), les cellules enflammées sont en orange (gris clair) et les cellules entièrement consumées sont en gris foncé. Dans la seconde série (bas) les cellules actives sont en rouge (gris clair) et les cellules quiescentes en bleu (gris foncé).

Nous avons développé un algorithme générique nous permettant de suivre la zone active,

dans laquelle des changement arrivent, au cours de la simulation. Ce dernier est exécuté à chaque itération.

Commençons par introduire quelques points de notation. L'ensemble A est choisi pour garder la trace des cellules actives durant la simulation et nous noterons A^i avec $i \in \{0, \dots, n\}$ l'ensemble des cellules actives à l'itération i de la simulation, n étant le dernier pas de la simulation. Au départ, A^0 contient toutes les cellules. L'ensemble Q contient toutes les cellules quiescentes et $Q^0 = \varnothing$. L'ensemble F^i permet de suivre les cellules pouvant devenir actives à l'itération suivante et correspond exactement à notre définition précédente d'un front d'incendie. Au début de la simulation, $F^0 = \varnothing$.

Mise à jour de l'état des cellules La première partie de l'algorithme concerne la mise à jour des cellules actives. Elle résulte de l'application des règles d'une transformation T sur les cellules de A plutôt que sur toutes les cellules. Une fois effectuée, une nouvelle ségrégation entre cellule active et quiescente apparaît et doit être calculée : certaines cellules deviennent actives, d'autres deviennent quiescentes.

```
1 forall px ∈ A^i do          // Mise à jour de l'état de toutes les cellules actives
2     if syn < i then
3         x ← state.(px).cur;
4     else
5         x ← state.(px).backup;
6     state.(px) ← {cur = T(x), backup = state.(px).cur, syn = i};
```

Mise à jour de la zone active Afin de trier entre cellules active et quiescentes, nous utilisons un prédicat d'activité P pour déterminer quelles cellules sont actives, soit restant active dans A^i ou devenant active dans F^i.

```
fun P(state, px) = (
  let x = state.(px).cur in
    (x.burnt != 1.0)
    &&
    exists((\y.(y.cur.burnt != 0.0)), neighbors(state,px))
) ;;
```

Dans un premier temps, on rempli un ensemble FQ comme stockage temporaire. Les éléments

de cet ensemble seront répartis, plus tard, soit dans F^{i+1} soit dans Q^{i+1}.

```
1  forall a ∈ A^i do                                   // Mise à jour de la zone active
2      if P(a) then
3          A^{i+1} ← a;
4      else
5          FQ ← a;
6  forall f ∈ F^i do
7      if P(f) then
8          A^{i+1} ← f;
9      else
10         FQ ← f;
11 forall q ∈ Q^i do
12     if not(HasNeighbor(q, F^i)) then
13         Q^{i+1} ← q;
14     else
15         FQ ← q;
```

Les cellules qui, soit restent actives, soit le deviennent, sont stockées dans A^{i+1}. Toutes les autres sont quiescentes, si elles sont suffisamment distantes de la zone active, ou dans F^{i+1}, si elles ont un voisin dans A^{i+1} car elles peuvent devenir actives à la prochaine itération.

```
1  forall c ∈ FQ do                                    // Séparation de FQ
2      if HasNeighbor(c, A^{i+1}) then
3          F^{i+1} ← c;
4      else
5          Q^{i+1} ← c;
```

3.6.3 Mesure de performance

En utilisant l'algorithme de filtrage de motif classique, l'intégralité de la collection doit être itérée à chaque mise à jour. Pour un AC sur une grille carrée de côté n, l'algorithme doit parcourir n^2 cellules ce qui résulte en un pas d'itération en temps constant (voir figure 3.10), tandis que seul un nombre fluctuant de cellules voient leur état changer (ce qui correspond à une fraction de la grille).

Dans cette section, nous utilisons l'activité pour réduire le coût du filtrage. Nous avons réécrit l'exemple de la propagation d'un feu de forêt précédent pour que le filtrage n'ait lieu que dans la région active en ignorant la région quiescente. À l'instar de la partie théorique de la section 3.5, l'espace de l'AC est scindé entre cellules actives prenant part à la transformation

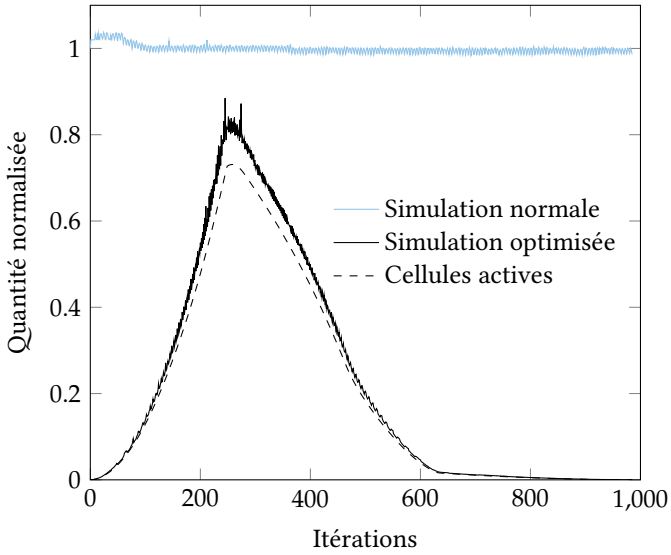


FIG. 3.10: Nombre de cellules actives et temps de calcul pour l'algorithme classique et optimisé par itération sur une grille de taille 500×500. Les valeurs sont normalisées selon la taille totale (250 000 cellules) en ce qui concerne le nombre de cellules, et le temps moyen de calcul d'une simulation classique en ce qui concerne le temps de calcul.

de leurs voisines et cellules quiescentes ne jouant aucun rôle dans l'itération courante. Il en résulte une amélioration notable du temps de calcul sur l'ensemble de la simulation.

La figure 3.10 montre le nombre de cellules actives à chaque itération ainsi que le temps de calcul par itération pour une simulation classique et optimisée. Le temps est normalisé par rapport au temps de calcul de la simulation classique de telle sorte qu'il soit aisé d'apprécier le gain induit par l'algorithme optimisé. Les pics et les fluctuations du temps de calcul proviennent de la variabilité des ressources disponibles lors du test.

De la figure 3.9, on peut déduire qu'au début de la simulation, très peu de cellules sont actives. Leur nombre s'accroit dramatiquement avec la progression du feu pour ne diminuer qu'une fois la forêt consumée et transformée en cendres. Pour l'AC le temps de calcul est linéaire avec le nombre de cellules : plus il y a de cellules, plus le temps de calcul est élevé tout en restant constant par cellule. De la figure précédente, on peut remarquer que le temps de calcul pour l'algorithme classique est le même à chaque itération : le mécanisme de filtrage doit explorer chacune des 250 000 cellules pour vérifier si un motif peut s'appliquer ou non. Dans ce cas, qu'un motif corresponde ou non ne change pas le temps de calcul nécessaire à l'application d'une règle de transformation.

On remarque également que le nombre de cellules actives et le temps de calcul de l'algorithme optimisé coïncident : ce dernier dépend effectivement du nombre de cellules à explorer, d'où le fait que la forme des deux courbes soit quasi identique. Les cellules ne sont jamais toutes actives au même moment, c'est pourquoi, même quand un grand nombre de cellules

sont actives, le temps de calcul par itération pour l'algorithme optimisé reste plus faible que pour sa contrepartie classique. Le coût de la maintenance des ensembles des cellules actives et quiescentes peut se lire à partir de la différence entre le temps de calcul pour l'algorithme optimisé et le nombre de cellules actives. Cette optimisation dépend du problème étudié et le graphe en sortie est totalement dépendant de la configuration des données initiales. Ainsi, on ne peut pas parler en toute généralité d'une optimisation en temps de calcul. Le cas suivant donne des résultats sensiblement différents.

3.7 Agrégation limitée par diffusion

Dans cette section, nous considérons un exemple plus élaboré : nous avons choisi d'implémenter le modèle d'un système présentant une *agrégation limitée par diffusion* (ALD). Contrairement au modèle de propagation d'un feu de forêt, le modèle d'une ALD présente de réelles interactions entre deux particules au lieu du changement d'état d'une cellule en fonction de l'état de ses voisines. En MGS, cela se traduit par le filtrage d'une paire de cellules au lieu d'un seule. Nous verrons néanmoins que cela n'affecte pas la propagation en vague de l'activité. Les résultats suivants découlent de la section 3.5, où apportée par l'activité est utilisée pour accélérer le temps de simulation en n'appliquant les règles de transformation *qu'à la seule* région active.

3.7.1 L'exemple d'une simulation d'une ALD

Le phénomène d'ALD est un phénomène physique dans lequel on observe un ensemble de particules s'agrégeant pour former des arbres Brownien [30]. Un modèle de ce phénomène a été implémenté par [25] sur un AC avec MGS.

Représentation des états avec MGS

Le treillis régulier d'un AC est représenté par un GBF décrivant un voisinage de Moore sur une grille carrée cyclique en deux dimensions.

En MGS, cette collection est spécifiée à partir de la présentation du groupe des déplacements. Comme dans l'exemple précédent, la définition d'une grille au voisinage de Moore nécessite une base de quatre mouvements, nord (**n**), est (**e**), nord-est (**ne**) et sud-est (**se**) :

```
gbf Grid = < n, ne, e, se ; 50 n = 0, 50 e = 0, ne = e + n, se = e - n > ;;
```

Les quatre équations additionnelles indiquent que la grille est cyclique et de taille 50×50, et définit les relations entre les directions horizontale, verticale et diagonales tandis que les directions inverses sont générées automatiquement. Ensuite, chaque cellule du GBF est étiqueté par son état, qui varie au cours des itérations de la simulation :

```
type cell = ‘empty | ‘mobile | ‘static ;;
```

L'étiquette `'mobile` désigne une cellule contenant une particule pouvant se déplacer dans une direction aléatoire à l'itération suivante ; l'étiquette `'static` est utilisée pour les cellules contenant une particule qui restera fixe jusqu'à la fin de la simulation. Finalement l'étiquette `'empty` est utilisée pour les cellules vides, ne contenant aucune particule, et pouvant accueillir une particule à l'itération suivante.

Spécification dynamique en MGS

Spécifier la dynamique d'une ALD dans une transformation revient à rédiger deux règles appliquées suivant une stratégie *maximal-parallel*:

```
trans evolution = {
  'mobile, 'static => 'static, 'static ;
  'mobile, 'empty =>  'empty,  'mobile ;
} ;;
```

La première règle de la transformation est sélectionnée si deux cellules voisines sont dans les états `'mobile` et `'static`, et les remplace par deux cellules dont l'état est `'static`. La seconde règle de la transformation s'applique si deux cellules voisines sont dans les états `'mobile` et `'empty` auquel cas elles échangent leurs états pour simuler un *saut* d'un particule de la première vers la seconde. On remarque que la priorité est donnée à la règle d'agrégation ce qui permet à une cellule étiquetée `'mobile` entourée de plusieurs `'empty` et d'au moins une `'static` de devenir `'static` plutôt que de continuer sa marche aléatoire. Cette fonction de transition itère sur toutes les cellules et mets à jour leur état. La figure 3.11 présente la simulation du modèle à différentes itérations.

3.7.2 L'activité dans une simulation d'ALD

Comme on pourrait s'y attendre, la nature d'une simulation d'ALD est bien différente de l'exemple précédent. On décèle moins bien la vague d'activité que dans la propagation d'un feu de forêt. Néanmoins, les mêmes principes sont utilisés sur l'exemple de l'ALD et les résultats sont présentés juste après.

L'algorithme utilisé pour suivre l'évolution de l'activité doit prendre en compte l'interaction entre deux cellules voisines : il doit d'abord en filtrer une puis une seconde dans le voisinage de la première. Le mécanisme de filtrage et de mise à jour de l'état d'une cellule est au cœur de l'algorithme de filtrage de motif de MGS. Mis à part la mis à jour de l'état d'une cellule, le reste de la procédure est identique : il faut répartir les cellules itérées dans A^{i+1}, F^{i+1} et Q^{i+1}.

3.7.3 Mesure de performance

Dans cette section, nous utilisons l'information fournie par l'activité pour réduire le coût du filtrage. L'exemple d'une ALD a été réécrit pour que les règles de la transformation ne s'appliquent qu'aux cellules des régions actives en omettant les cellules quiescentes. À l'instar du point de vue théorique présenté dans la section 3.5, l'espace de l'AC est partagé entre cellules

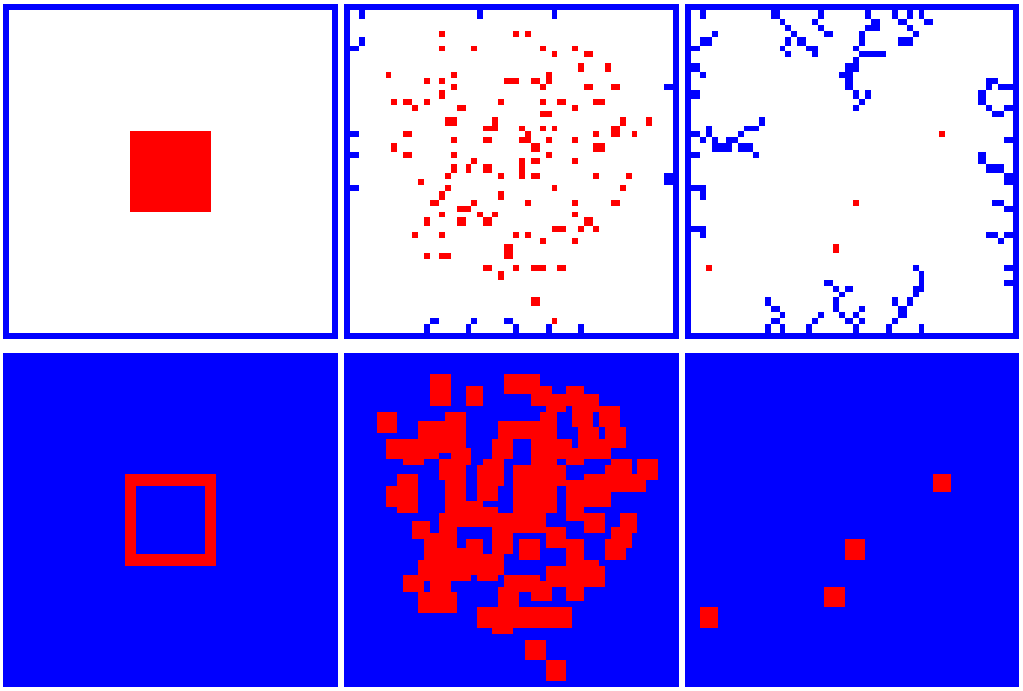


FIG. 3.11: ALD à trois itérations différentes, de gauche à droite, d'une simulation comportant des particules mobiles au centre et des particules statiques aux bords sur une grille de 50×50. Dans la série du haut, les cellules `'mobile` sont en rouge, les cellules `'static` sont en bleu et les `'empty` sont en blanc. Dans la série du bas, les cellules actives sont en rouge et les cellules quiescentes sont en bleu.

actives prenant par à la transformation et cellules quiescentes ne jouant aucun rôle à l'itération courante. La simulation s'en retrouve accélérée, mais dans de moins larges proportions par rapport à l'exemple précédent.

La figure 3.12 présente un graphe du nombre de cellules actives pour chaque itération et le temps de calcul par itération pour l'algorithme optimisé. Les fluctuations occasionnelles sont causées par une variation des ressources disponibles dans l'environnement de test.

Comme montré sur la figure 3.11, au début de la simulation, très peu de cellules sont actives ; il n'y a que la *couronne* autour du carré initial de particules mobiles représentant les particules ayant la possibilité de bouger. Leur nombre croît dramatiquement avec la marche aléatoire des particules remplissant l'espace et décroît dès qu'une particule mobile s'attache à une particule fixe. Comme dans le cas d'un AC classique, le temps de calcul est linéaire avec le nombre de cellules : plus leur nombre est important, plus le temps nécessaire au calcul est important par itération. De plus, le temps de calcul pour l'algorithme classique est quasiment identique à chaque itération : le mécanisme de filtrage doit parcourir l'ensemble des 10 000 cellules pour vérifier si un motif convient. Le fait qu'une règle s'applique ou non ne change que marginalement le temps de calcul nécessaire par cellule.

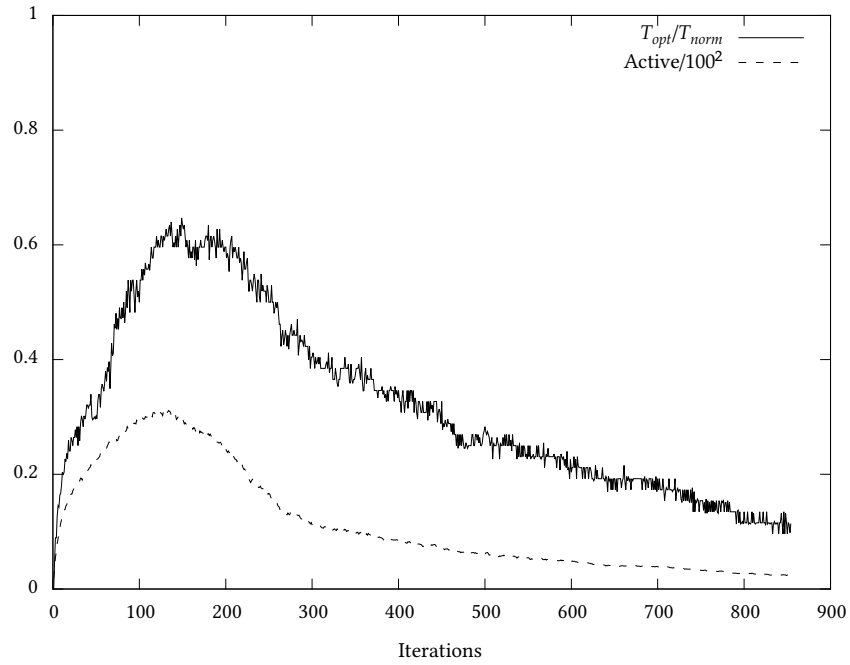


FIG. 3.12: Nombre de cellules actives, et temps de calcul pour l'algorithme optimisé par itération sur une grille de 100×100. Les valeurs ont été normalisées en utilisant la taille totale de la grille (10 000 cellules) et le temps de calcul par itération pour l'algorithme classique respectivement.

Du graphe, nous remarquons que l'activité et le temps de calcul pour l'algorithme optimisé coïncident : en effet ce dernier dépend de la taille de la région active. Il semble donc naturel que l'allure générale de la courbe soit identique à celle de la variation du nombre de cellules actives.

Les cellules ne sont pas toutes actives au même moment. En fait, pour la simulation présentée, le nombre maximum de cellules actives est très faible et reste en dessous de 30% du total. Par conséquent, même à son pic, l'algorithme optimisé requiert moins de temps de calcul par itération que l'algorithme classique. Le coût de la maintenance de la zone active peut être lu à partir de la différence entre le temps de calcul pour l'algorithme optimisé et le nombre de cellules actives. Dans notre cas, il est assez élevé, peut être parce que le mécanisme d'optimisation lié à l'activité a été implémenté comme un programme MGS et n'est pas directement inclus dans le langage lui-même. Cette optimisation est dépendante des données, cela se voit en comparant les différences de résultats par rapport à l'exemple précédent. De manière générale, il n'y a aucune garantie de l'optimisation en temps des calculs mais il ne peut y avoir de ralentissement car le pire scénario revient à devoir filtrer entièrement une collection topologique et calculer les collections A, F et Q, ce qui est le cas par défaut. Concernant l'ALD, l'accélération de la simulation est aussi dépendante du type d'implémentation ; dans le cas d'un modèle à agent, le suivi de l'activité nous a permis de nous focaliser sur les particules mobiles et ainsi

d'obtenir un temps de simulation optimal.

3.8 Conclusion et perspectives

Cette courte section vise à rappeler les principaux résultats obtenus suite à nos travaux sur l'activité spatiale ainsi que nos futures pistes de travail sur le sujet.

3.8.1 Conclusion

Nous avons, dans ce chapitre, présenté quelques points clef du formalisme de MGS, en particulier : les collections topologiques et les transformations. Nous avons aussi succinctement décrit deux modèles formels, les complexes cellulaires abstraits et la réécriture topologique, servant de fondement au langage. Nous avons d'abord utilisé un automate à trois états minimal pour introduire le concept d'activité (section 3.5) dans le domaine du calcul spatial en définissant une zone active comme étant les régions de l'espace filtrées, et où seules les cellules filtrées sont mises à jour. Bien que ce concept ait été introduit sur un exemple de propagation de feu de forêt (section 3.6), le fait que l'activité progresse comme une vague n'est pas restreint à ce seul cas et s'étend naturellement à d'autres cas comme le modèle d'agrégation limitée par diffusion (section 3.7). La propagation d'un feu de forêt semble être un cas particulier où l'activité est en adéquation presque parfaite avec la simulation où la sur-approximation induite par le suivi de l'activité est minimale.

De nos précédentes observations, nous avons obtenus deux résultats principaux :

1. un résultat formel : la description générique du calcul de la zone active, indépendamment de toute connaissance sur la zone quiescente, et
2. un résultat expérimental : la mise à jour de la zone active uniquement nous a apporté un gain en vitesse d'exécution inversement proportionnel au nombre de cellules actives, pour les deux applications.

Notre motivation à l'exploration de l'activité dépasse la simple optimisation de temps des simulations avec MGS. Elle découle de notre approche de la modélisation multi-niveau que nous avons introduit dans le chapitre 2. À ce stade de nos travaux, au niveau du langage, les zones actives ne sont pas des objets de première classe : elles sont calculées à chaque pas de la simulation et il n'existe pas encore de lien entre ces régions entre deux pas de simulation différents. Par cet aspect, l'activité trace un chemin vers une définition générique et une utilisation du *front* actif, de la *réification* des régions actives en des objets de première classe. Reformulé en terme d'identité, notre implémentation permet par identification qualitative de découvrir les zones actives, qui sont des objets d'un autre niveau de modélisation. L'identité numérique de ce front actif n'est pas encore inclus au niveau du langage. D'autres questions demandent à être abordées avant de pouvoir manipuler les zones actives comme des objets de premier ordre. Nous les présentons dans la section suivante.

3.8.2 Perspectives

Nos travaux futurs s'articulent autour des grands axes suivants, conditions nécessaires à la *réification* des fronts de vague d'activité et à leur inclusion en tant qu'objets de première classe dans MGS:

Identifier Nous avons déjà établi qu'il était possible, sans modifier le langage, d'identifier la zone spatiale active dans une simulation MGS: cette zone est décrite par les trois ensembles de cellules A, Q et F. Notre prochaine étape est d'étendre les tests effectués sur des ensembles de cellules plus grands, d'une part, et sur des modèles implémentés sur d'autres collections topologiques, d'autre part, afin de confirmer nos observations. Schématiquement, en prenant C_i comme la collection topologique au pas i et A_i comme l'ensemble des cellules actives au pas i, alors nous avons :

$$\begin{array}{c} A_i \\ \uparrow \\ C_i \end{array}$$

De plus, nous nous posons la question de la structure de donnée la mieux adaptée à l'ensemble des cellules actives : est-ce la collection topologique du modèle ? Ou bien d'autres candidats sont-ils plus appropriés ?

Suivre Nous avons également mis en œuvre un suivi rudimentaire de la zone active. Cependant, ce suivi n'existe pas encore du point de vue de MGS. Aussi, il est nécessaire de déterminer une correspondance directe, à deux pas successifs i et $i+1$ de la simulation, entre A_i et A_{i+1}. Autrement dit, il nous faudrait établir une transformation agissant directement sur A_i, indépendamment du modèle sous-jacent. Prenons C_i comme l'ensemble des cellules de la collection topologique du modèle au pas i et C_{i+1} au pas $i+1$, si le diagramme suivant commute, alors cette transformation est une abstraction :

$$\begin{array}{ccc} A_i & \longrightarrow & A_{i+1} \\ \uparrow & & \uparrow \\ C_i & \longrightarrow & C_{i+1} \end{array}$$

Différencier Nous avons constitué les ensembles A, Q et F à partir de l'intégralité des règles d'une transformation. Nous souhaitons enquêter sur le sens de la distinction entre différents sous-ensembles d'activité engendrés par une ou plusieurs règles d'une transformation. Par exemple, dans le cas de la transformation de l'ALD, page 90, nous avions deux règles, la première décrivant la capture par accrétion d'une particule, et l'autre décrivant leur déplacement brownien. Quel serait le sens, l'usage, la portée et l'implémentation du front actif d'accrétion ou du front actif de déplacement ? Qu'en est-il de leurs interactions ?

Une fois les réponses à ces questions clarifiées, nous pourrons alors proposer une syntaxe spécifique à la manipulation des fronts de vague d'activité au niveau du langage MGS.

À l'avenir, il sera intéressant de préciser les relations entre le contexte original de l'opérateur de vague, emprunté à la théorie de Morse, et notre exploration de l'activité spatiale.

Références

[1] Ulrike Axen. "Topological Analysis Using Morse Theory and Auditory Display". Thèse de doct. Champaign, IL, USA : University of Illinois at Urbana-Champaign, 1998.

[2] J.H. Bardley et A.B. Clymer. "Difficulties in the simulation of wildfires". In : *Proceedings of the 1993 Simulation Multiconference on the International Emergency Management and Engineering Conference*. 1993, p. 161–171.

[3] T. Beer. "The Australian National bushfire model". In : *Proceedings of 8th Biennial Conference and Bushfire Dynamics Workshop*. Canberra, Australia, 1989, p. 568–572.

[4] Marsha J Berger et Joseph Oliger. "Adaptive mesh refinement for hyperbolic partial differential equations". In : *Journal of computational Physics* 53.3 (1984), p. 484–512.

[5] Bastien Chopard et Michel Droz. "Cellular automata modeling of physical systems". In : *Cellular automata modeling of physical systems* (1998).

[6] O Delémont et J-C Martin. "Application of computational fluid dynamics modelling in the process of forensic fire investigation : Problems and solutions". In : *Forensic science international* 167.2 (2007), p. 127–135.

[7] Jean-Louis Giavitto et Olivier Michel. "Data Structure as Topological Spaces". In : *Unconventional Models of Computation: Third International Conference, UMC 2002 Kobe, Japan, October 15–19, 2002 Proceedings*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2002, p. 137–150.

[8] Jean-Louis Giavitto, Olivier Michel, Julien Cohen et Antoine Spicher. "Computations in space and space in computations". In : *Unconventional Programming Paradigms*. Springer, 2005, p. 137–152.

[9] Jean-Louis Giavitto, Olivier Michel et Jean-Paul Sansonnet. "Group-based fields". In : *Parallel Symbolic Languages and Systems*. Springer Berlin Heidelberg, 1996, p. 209–214.

[10] Daniel T Gillespie. "Exact stochastic simulation of coupled chemical reactions". In : *The journal of physical chemistry* 81.25 (1977), p. 2340–2361.

[11] RB Good et RHD McRae. "The challenges of modeling natural area ecosystems". In : *Proceedings of 8th Biennial Conference and Bushfire Dynamics Workshop*. Canberra, Australia, 1989, p. 475–484.

[12] F. Hirabayashi et Y. Kasahara. "Fire-spread simulation using an extended dynamic percolation process model". In : *NEC research and development* 89 (1988), p. 111–122.

[13] Alfons G Hoekstra, Eric Lorenz, Jean-Luc Falcone et Bastien Chopard. "Towards a Complex Automata formalism for multi-scale modeling". In : *Int. J. Mult. Comp. Eng* 5.6 (2007), p. 491.

[14] X. Hu, Y. Sun et L. Ntaimo. "DEVS-FIRE: design and application of formal discrete event wildfire spread and suppression models". In : *SIMULATION* 88.3 (oct. 2011), p. 259–279.

[15] T. N. Ivanilova. "Set probability identification in forest fire simulation". In : *Annual Review in Automatic Programming* 12, Part 2 (1985), p. 185–188.

[16] Ioannis Karafyllidis et Adonios Thanailakis. “A model for predicting forest fire spreading using cellular automata”. In : *Ecological Modelling* 99.1 (1997), p. 87–97.

[17] Yoann Kubera, Philippe Mathieu, Sébastien Picault et al. “Interaction-Oriented Agent Simulations: From Theory to Implementation.” In : *ECAI*. 2008, p. 383–387.

[18] Marco Mamei et Franco Zambonelli. *Field-based coordination for pervasive multiagent systems*. Springer Science & Business Media, 2006.

[19] Marco Mamei, Franco Zambonelli et Letizia Leonardi. “Co-fields : Towards a unifying approach to the engineering of swarm intelligent systems”. In : *Engineering Societies in the Agents World III*. Springer, 2003, p. 68–81.

[20] Alexandre Muzy, Franck Varenne, Bernard P Zeigler, Jonathan Caux, Patrick Coquillard, Luc Touraille, Dominique Prunetti, Philippe Caillou, Olivier Michel et David RC Hill. “Refounding of the activity concept ? Towards a federative paradigm for modeling and simulation”. In : *Simulation* 89.2 (2013), p. 156–177.

[21] Lewis Ntaimo, Xiaolin Hu et Yi Sun. “DEVS-FIRE: Towards an Integrated Simulation Environment for Surface Wildfire Spread and Containment”. In : *SIMULATION* 84.4 (2008), p. 137–155.

[22] Martin Potier, Antoine Spicher et Olivier Michel. “Topological computation of activity regions”. In : *SIGSIM Principles of Advanced Discrete Simulation*. Montreal, Québec, Canada, mai 2013, p. 337–342.

[23] Antoine Spicher. “Transformation de collections topologiques de dimension arbitraire : application à la modélisation de systèmes dynamiques”. Thèse de doct. Evry-Val d’Essonne, 2006.

[24] Antoine Spicher, Olivier Michel et Jean-Louis Giavitto. “A topological framework for the specification and the simulation of discrete dynamical systems”. In : *Sixth International conference on Cellular Automata for Research and Industry (ACRI’04)*. T. 3305. LNCS. Springer, 2004, p. 238–247.

[25] Antoine Spicher, Nazim A Fatès, Olivier Simonin et al. “From Reactive Multi-Agent models to Cellular Automata - Illustration on a Diffusion-Limited Aggregation model”. In : *Proceedings of ICAART’09* (2009).

[26] S. R. Tieszen. “On the Fluid Mechanics of Fires”. In : *Annual Review of Fluid Mechanics* 33 (2001), p. 67–92.

[27] K. D Tocher. *The Art of Simulation*. English Universities Press, 1967.

[28] Tommaso Toffoli et Norman Margolus. *Cellular automata machines : a new environment for modeling*. Cambridge : MIT press, 1987.

[29] W. Wagner, A. Ullrich, T. Melzer, C. Briese et K. Kraus. “From single-pulse to full-waveform airborne laser scanners : potential and practical challenges”. In : *International Archives of Photogrammetry and Remote Sensing* 35.B3 (2004), p. 201–206.

[30] T. A. Witten et L. M. Sander. “Diffusion-Limited Aggregation, a Kinetic Critical Phenomenon”. In : *Phys. Rev. Lett.* 47.19 (nov. 1981), p. 1400–1403.

Chapitre 4

Travelling Bacteria

Sommaire

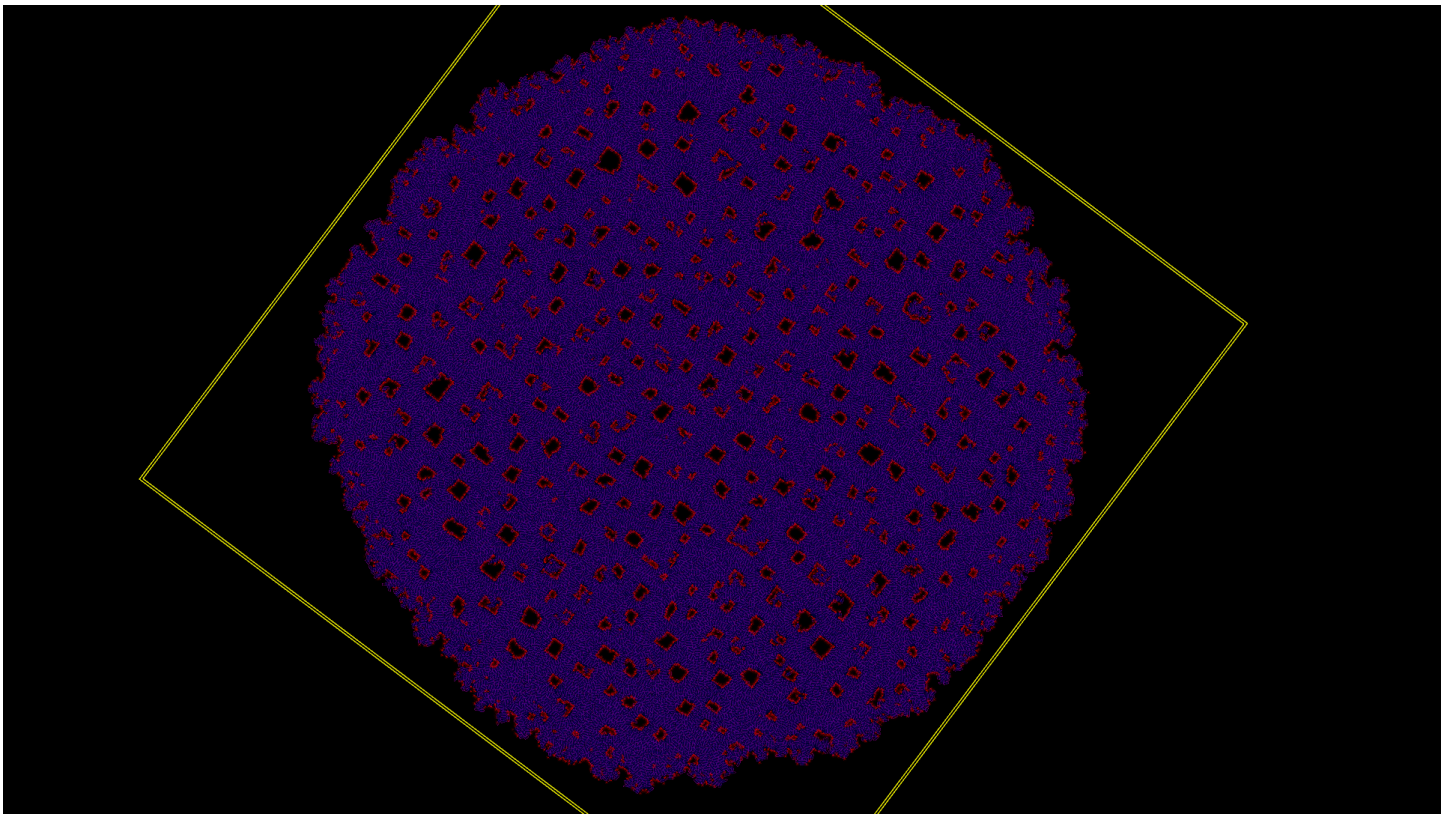


FIG. 4.1: Fenêtre de rendu du logiciel OTB lors d'une simulation comportant un grand nombre de bactéries. Les cadres jaunes indiquent l'espace actuellement occupé par les bactéries et leur support

Préliminaires

Voici quelques notations utilisées dans ce chapitre :

- $(G, +)$ est le groupe engendré par l'ensemble G et la loi $+ : G \times G \to G$.
- $\langle G \rangle$ est le groupe libre engendré par l'ensemble des générateurs G.
- $\mathbb{N}_+$ est l'ensemble des entiers naturels privé de 0.
- $\lfloor x \rfloor = \max\{m \in \mathbb{Z} \mid m \leq x\}$ est la fonction partie entière de $x \in \mathbb{R}$.
- $\vec{u}$ est un vecteur,
- $\|\vec{u}\|$ est la norme euclidienne du vecteur $\vec{u}$,
- $k \cdot \vec{u}$ est le produit scalaire entre un scalaire k et un vecteur $\vec{u}$,
- $\vec{u} \cdot \vec{v}$ est le produit scalaire entre deux vecteurs $\vec{u}$ et $\vec{v}$,
- $\vec{u} \wedge \vec{v}$ est le produit vectoriel entre deux vecteurs $\vec{u}$ et $\vec{v}$.
- $[x]_{x_{min}}^{x_{max}} = \begin{cases} x_{max} & \text{si } x > max \\ x_{min} & \text{si } x < min \\ x & \text{sinon} \end{cases}$ est la fonction *clamp*.

4.1 Simuler la morphogenèse des populations

En tant que science expérimentale, la recherche en biologie cellulaire et moléculaire nécessite d'important moyens matériels afin de tester *in vivo* les hypothèses formulées par les chercheurs. Notamment, la préparation et l'exécution d'une expérience est…

- consommatrice en temps. Il faut parfois attendre que les cultures d'un organisme particulier atteignent l'état désiré[1];
- consommatrice en ressources. Maintenir les conditions de pression, d'humidité, de température, d'acidité, d'oxygénation, etc. nécessite du matériel onéreux ;

D'un côté, de par les moyens engagés, il est difficile d'envisager de tester des hypothèses très prospectives ou bien de se faire une idée du résultat probable d'une expérience quand beaucoup d'entités interagissent. D'un autre, un plan d'expérience comprend idéalement un nombre minimum nécessaire d'expérimentations pour valider ou infirmer une hypothèse du modèle, ce nombre pouvant néanmoins rester élevé au regard des moyens à disposition. La simulation informatique permet, à partir d'un modèle donné, de confirmer ou de rejeter rapidement une hypothèse, ce qui peut aider à choisir quelles expérimentations effectuer. Dans ce cas c'est un outil d'aide à la décision. Si la simulation est suffisamment précise, comme dans la cas de la simulation de la motilité de quelques bactéries *Escherichia Coli* [4], elle pourrait même se substituer en partie à l'expérimentation *in vivo*; on parlera dans ce cas d'expérimentation *in silico*.

Nous nous intéressons au problème de la *morphogénèse* c'est à dire à l'ensemble des lois qui déterminent les étapes de la forme, de la structure des tissus, des organes et des organismes.

1. cf. le projet Synbiotic

Les organismes multicellulaires sont le fruit d'un grand nombre de divisions cellulaires successives et qui donnent lieu à l'apparition d'une forme et de fonctionnalités caractéristiques et reproductibles. Comprendre la morphogénèse est un enjeu fondamental pour la biologie synthétique (voir à ce propos la section 1.1).

Pour répondre à ce problème et dans le cadre du projet Synbiotic, nous avons développé OTB, un simulateur des interactions d'une population de bactéries *E. Coli* au cours de sa croissance. OTB permet de suivre une cellule ou une sous partie de cette population au cours du temps de manière similaire aux conditions expérimentales et ainsi de tester des hypothèses à l'échelle de la population dans le domaine de la biologie synthétique.

4.1.1 Propriétés attendues

Dans le but de fournir aux biologistes le moyen d'expérimenter *in silico* la validité d'un modèle concernant l'évolution d'une population de bactéries *E. Coli*, nous avons dressé une liste de propriétés nécessaires.

Sorties Le simulateur permet de recueillir des résultats qualitatifs via un affichage graphique, reprenant la métaphore de la boîte de Petri qui permettra d'identifier la forme de la population en croissance. Il est possible de marquer une sous-partie de la population pour la suivre plus facilement. Le simulateur permet de recueillir des résultats quantitatifs, tel que le nombre et la position des bactéries dans un fichier texte pour traitement ultérieur, par exemple pour la reconnaissance automatique de forme, ou des calculs statistiques ;

Entrées Les conditions initiales de la simulation sont décrites par un fichier de configuration dans un langage dédié ; au delà des conditions initiales de l'expérience, il permettra au moins de fixer quelques paramètres du rendu graphique ou textuel, comme la couleur de représentation des morphogènes et des bactéries, et de sélectionner quelles informations quantitatives seront écrites pendant la simulation ;

Précision Le simulateur reprend aussi fidèlement que possible les connaissances actuelles sur l'aspect mécanique (croissance et collision des bactéries) et sur l'aspect chimique (réaction et diffusion de morphogènes). Une étape de calibrage sera donc nécessaire par rapport à des comptes rendus d'expériences *in vivo*;

Efficacité Le simulateur est suffisamment efficace pour être utilisable sans matériel spécifique ; par exemple, passer d'une bactérie à une population de l'ordre de 10^4 bactéries, soit un temps réel d'environ de 4 h30 min, nécessitera un temps de simulation correspondant de l'ordre de l'heure sur un ordinateur portable contemporain, moins encore sur une machine spécialisée ;

Autonomie Les résultats sont accessibles sans la nécessité de faire fonctionner le simulateur sur une grille de calcul, un tel matériel ne servira qu'à réduire le temps de simulation afin d'effectuer un plan d'expérience plus rapidement ;

Scalabilité Malgré les points précédents, OTB passe à l'échelle et peut s'adapter à une machine de calcul dédiée ou à une grille de calcul distribuée ; plusieurs instances du simulateur peuvent être lancées de manière concurrente ;

Adaptabilité Il est possible d'ajouter à OTB de nouvelles fonctionnalités avec peu d'effort : l'architecture du simulateur est modulaire et vient avec une documentation claire ;

Stabilité Comme une simulation complexe peut prendre plusieurs heures, il est important que le simulateur ne produise pas d'erreur en cours de fonctionnement, autrement dit OTB est stable ; d'une part le langage utilisé apportera des garanties de part des vérifications du système de type, d'autre part des tests fonctionnels permettront de valider le fonctionnement correct de chaque partie du simulateur ;

Ouverture Le simulateur est un programme informatique ouvert, son code source peut-être téléchargé en ligne, compilé, amélioré et redistribué, modifié ou non ;

4.1.2 OCaml

Pour répondre aux critères *d'adaptabilité* et de *stabilité*, nous avons choisi d'implémenter OTB avec le langage OCaml, un langage de programmation fonctionnel, statiquement typé, modulaire et interfaçable avec le langage C. OCaml est à la fois un projet académique avec de solides bases théoriques, notamment utilisé comme langage d'implémentation de l'assistant de preuve Coq[2], et aussi un langage industriel ayant à son actif de belles réussites comme l'analyseur statique Astrée en usage chez Airbus.

Langage de programmation fonctionnel La programmation fonctionnelle est issue du λ-calcul, un modèle de calcul mathématique reposant sur des variables, des applications et des abstractions regroupées sous le nom de λ-*termes*. Dans le modèle du λ-calcul, la règle centrale pour le calcul des λ-termes est la *substitution* dont les deux principaux représentants sont l'α-conversion et la β-réduction. Le calcul de nouveaux λ-termes s'effectue par un système de réécriture. C'est ce modèle de calcul qui a donné lieu au paradigme de la programmation fonctionnelle. Par extension, les langages de programmation fonctionnelle purs ont la particularité de considérer la programmation d'un algorithme comme une composition de fonctions s'appelant les unes les autres plutôt que comme une suite d'instructions qui agissent par effet de bord, c'est-à-dire en modifiant l'état de la mémoire. Ils se dissocient de la vision de la machine à état comme support de programmation en adoptant un style plus mathématique dans leur écriture, proche de la réécriture des λ-termes. OCaml dispose ainsi

- de structures de contrôle comme le filtrage par motif,
- de clôtures, qui sont des définitions de fonction accompagnées de leur environnement lexical, analogues des λ-termes et

2. cf. la page « à propos » du projet Coq `https://coq.inria.fr/about-coq`

- permet de définir des fonctions d'ordre supérieur, c'est-à-dire aux fonctions pouvant avoir des fonctions en argument et pouvant retourner des fonctions comme résultat.

Un paradigme de programmation influe sur la syntaxe et la sémantique du langage, et comme OCaml est un langage multiparadigme, empruntant aux paradigmes de la programmation fonctionnelle, de la programmation impérative et de la programmation orientée objet, il hérite de leurs aspects. OCaml dispose d'un ramasse-miette efficace qui gère la désallocation de la mémoire quand celle-ci n'est plus utilisée sans l'intervention de l'utilisateur. Ceci fait de OCaml un langage de programmation de haut niveau. Comme nous le présentons ci-après, OCaml est de plus un langage fonctionnel fortement et statiquement typé.

Typage fort et typage statique Les types ont d'abord été introduit en logique combinatoire et dans le λ-calcul dans les années 1930. Ils ont depuis eu un fort impact en informatique : ils sont utilisés comme une abstraction d'un programme et permettent d'en vérifier certaines propriétés, comme par exemple garantir que le résultat d'une fonction appartient à un ensemble d'éléments connus. Dans le langage C, un langage bas niveau proche de l'expression d'un processeur, les types décrivent la taille de la représentation machine et les règles de conversion d'une valeur du langage. En OCaml, les types sont plutôt utilisés pour modéliser les données du programme. OCaml permet de déclarer des types algébriques récursifs permettant par exemple de décrire la structure de listes

```
type 'a list =
    Nil
  | Cons of 'a * 'a list
```

ou d'arbres binaires

```
type 'a tree =
    Leaf of 'a
  | Node of 'a tree * 'a tree
```

OCaml présente un typage fort. On distingue un typage fort d'un typage faible par la manière dont s'effectue la transformation d'un type en un autre : plus elle est explicite plus c'est un typage fort, moins elle explicite moins c'est un typage faible. Autrement dit, plus un langage est fortement typé, plus il est difficile de transformer un type en un autre. Dans le langage C, un opérateur appelé *cast* permet aisément de transformer presque n'importe quel type en un autre. En OCaml, cette transformation n'est pas possible dans le cadre de fonctionnement normal du langage, donc OCaml est plus fortement typé que C et le système de type d'OCaml est par conséquent plus riche. Le code source d'un programme OCaml est typé statiquement, c'est à dire que toutes les annotations de type sont connues avant la compilation. À l'opposé, dans le cas d'un typage dynamique, certaines informations de type peuvent ne pas être connues avant la compilation, peuvent rester après la compilation et peuvent évoluer durant l'exécution du programme.

OCaml utilise un typage statique et fort et les annotations de type sont utilisées à la compilation pour permettre au compilateur de détecter les erreurs de types avant l'exécution du programme, ce qui apporte quelques garanties concernant le comportement. Par exemple, les

types nous permettent d'apporter des informations supplémentaires que le compilateur peut prendre en compte. Déclarer les types

```
type nom = Nom of string
type prenom = Prenom of string
```

permet au compilateur de distinguer deux chaînes de caractères identiques, mais dont le sens est différent, ce qui est représenté par deux types différents.

Modularité native Lors de la rédaction d'un programme, il est vite nécessaire de « compartimenter » le code source, d'abord parce que certaines fonctionnalité sont génériques et peuvent être utilisées plusieurs fois dans des contextes différents, mais aussi pour des raisons de relecture et de segmentation de l'espace de noms. La *modularité* consiste à rassembler les fonctionnalités d'un programme dans des modules *indépendants* et *interchangeables*. En OCaml, la modularité est une caractéristique native du langage dans la gestion du code source et repose sur trois constructions syntaxiques : les *structures*, les *interfaces* et les *modules*. Les structures contiennent l'implémentation des modules, les interfaces décrivent les valeurs qui en sont accessibles — les valeurs dont l'implémentation n'est pas exposée sont des valeurs abstraites, et celles qui n'apparaissent pas du tout dans l'implémentation du module sont inaccessibles. Un module peut avoir plusieurs interfaces — du moment qu'elles sont toutes compatibles avec les types de l'implémentation — et plusieurs modules peuvent vérifier une même interface. Enfin, OCaml dispose de modules paramétriques nommés *foncteurs* qui sont des structures paramétrées par d'autres structures.

Interface avec le langage C Un aspect important pour les besoins du cahier des charges (voir 4.1.3) est de pouvoir communiquer avec des objets compilés C via une interface dédiée, typiquement des bibliothèques écrites et compilés dans ce langage. OCaml propose une interface claire avec des bibliothèques logicielles écrites en C, permettant d'empaqueter des valeurs dans des types OCaml conservant les avantages d'un typage fort, de dépaqueter des valeurs typées de OCaml vers un type C arbitraire et surtout de profiter de la gestion automatique de la mémoire par le ramasse-miette de OCaml.

4.1.3 Calcul parallèle

Notre simulateur calcule l'évolution d'un grand nombre d'objet de même nature simultanément. Pour répondre aux critères *d'efficacité* et *d'autonomie* nous avons le choix d'utiliser un GPU, le processeur dédié des cartes graphiques, ou bien d'utiliser un cluster : une grappe de serveurs interconnectés sur le réseau local. En prenant le nombre d'opérations en virgule flottante par seconde, notée FLOPS, comme référence de la vitesse d'un ordinateur, utiliser le parallélisme embarqué dans les GPU semble être la bonne manière d'atteindre des vitesses de calcul élevées. Nous nous appuyons sur le fait que plus de la moitié des 500 supercalculateurs les plus rapides au monde sont construit sur NVidia Kepler[3], un GPU. Les GPU grand public

3. https://www.top500.org/statistics/list/

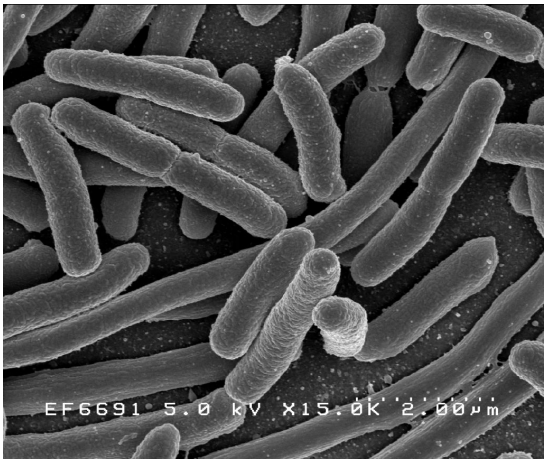
sont accessibles en terme de prix, leur consommation est suffisamment faible pour fonctionner sur batterie dans un PC portable et les plus puissants atteignent les 10 000 GFLOPS[4].Enfin, le parallélisme des GPU semble être plus efficace énergétiquement par rapport aux CPU en comparant leur GFLOPS par Watt [10].

SIMD La taxinomie de Flynn [12] classifie différentes architectures d'ordinateur et comporte quatre catégories selon le type d'organisation du flux de données et du flux d'instructions. Le modèle de programmation le plus répandu repose sur l'architecture de Von Neumann, ce qui correspond à la catégorie *Single Instruction Single Data* (SISD). Les ordinateurs de cette catégorie n'exploitent aucun parallélisme et sont purement séquentiels, tant au niveau des instructions qu'au niveau de la mémoire. Ce modèle de programmation ne correspond pas à la réalité des processeurs contemporains qui exploitent le parallélisme des données (plusieurs banques de mémoire RAM) ou d'instructions grâce à plusieurs processeurs à plusieurs cœurs. À l'extrême opposé de la classification se trouvent les architectures *Multiple Instruction on Multiple Data* (MIMD), puis les architectures *Multiple Instructions on Single Data* (MISD). Dans le cas des MISD, des problèmes de concurrence à la lecture–écriture apparaissent et il est d'usage d'avoir recours à des barrières de synchronisation, des sémaphores ou des Mutex. SPMD, *Single Program on Multiple Data*, est la technique la plus courante pour utiliser le parallélisme des ordinateurs multiprocesseurs contemporain, où chaque processeur exécute un programme distinct et possède sa propre mémoire, en plus de partager une mémoire globale, la RAM. Finalement, dans les ordinateurs contemporains, il existe, en complément d'un CPU et d'une mémoire partagée, des cartes de calcul dédiées, comme des cartes graphiques, spécialisées dans le calcul vectoriel, capable d'effectuer des opérations rapides sur des grandes matrices. L'architecture d'une carte graphique, munie d'un GPU et d'une mémoire RAM entre dans la catégorie *Single Instruction on Multiple Data* (SIMD). Ces processeurs sont spécialisés dans le traitement en parallèle d'un grand nombre de données. Depuis la taxinomie de Flynn, d'autres types de Nous avons choisi de tirer parti du parallélisme disponible sur l'ordinateur où est exécuté OTB, qu'il provienne de plusieurs CPU ou d'une carte graphique.

OpenCL Afin d'accéder aux périphériques présent sur la plateforme de simulation et de les programmer, nous utilisons OpenCL (Open Computing Language), un framework pour l'écriture de programmes s'exécutant sur des plateformes hétérogènes : microprocesseurs (CPU), processeurs graphique (GPU), processeurs de signal numérique (DSP), réseau de portes programmables in situ (FPGA) ou autre carte accélératrice. OpenCL spécifie un langage de programmation, inspiré de C99, pour la programmation de ces périphériques ainsi que des interfaces de programmation (API) pour le contrôle de la plateforme et l'exécution des programmes sur les périphériques. OpenCL fournit une interface standard pour la programmation parallèle utilisant le parallélisme de tâche et le parallélisme de données. Il existe d'autres framework pour la programmation des GPU, comme CUDA[5], cependant l'avantage de OpenCL est qu'il

4. https://www.nvidia.com/en-us/geforce/products/10series/titan-x-pascal
Par exemple, la carte graphique NVidia Titan X possède 3584 cœurs de calculs, cadencés à 1417 MHz pour un prix avoisinant les 1200 euros

5. http://www.nvidia.com/object/cuda_home_new.html



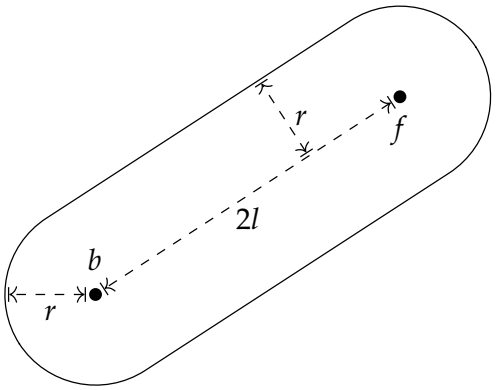


FIG. 4.2: Une image d'une microscopie électronique à balayage d'une culture de *E. Coli* (à gauche) et une représentation simplifiée en deux dimensions d'une bactérie *E. Coli* (à droite)

est le seul langage pour le GPGPU (General Purpose computing on Graphics Processing Unit) à ne pas être restreint à une seule plateforme. En effet, il fonctionne sur une large gamme de matériel et, au contraire de CUDA, ne se limite pas à un seul constructeur, ni à un seul type de périphérique. OpenCL est un standard ouvert maintenu par Khronos Group[6], un consortium technologique à but non lucratif. De nombreux constructeurs fournissent une implémentation officielle, comme AMD, Intel et Nvidia les principaux constructeurs de CPU et de cartes graphiques grand public.

OpenGL Pour répondre au point « Sortie », nous avons choisi d'utiliser OpenGL (Open Graphics Library), une API multiplateforme, multilangage utilisée pour programmer un GPU spécifiquement pour un rendu graphique. Dans le modèle de programmation de OpenGL, un ensemble de vecteurs dans l'espace en trois dimensions sont successivement transformés pour obtenir une projection en deux dimension rastérisée. Le programmeur écrit des programme de traitement, des shader, qui viennent se brancher à différents points de la chaîne de traitement graphique. À l'inverse de OpenCL, OpenGL est optimisé pour le calcul vectoriel et n'est pas approprié pour effectuer des calculs généraux sur les GPU. Cette bibliothèque logicielle nous permet de proposer une ou plusieurs fenêtre de rendu en temps réel pour présenter des informations qualitatives sur la progression d'une simulation.

4.1.4 Modélisation d'une cellule

Escherichia Coli est une bactérie bacillaire, c'est à dire en forme de bâtonnet, du genre *Escherichia*. C'est une bactérie vivant habituellement dans l'intestin des animaux à sang chaud, dont elle constitue 0,1% de la flore intestinale. Découverte en 1885 par Theodor Escherich et étudiée

6. https://www.khronos.org/about

intensivement depuis 60 ans, c'est un organisme très utilisé dans le domaine de l'ingénierie biologique et de la microbiologie industrielle, notamment pour sa disponibilité et sa facilité de culture. La souche K-12, adaptée à l'étude en laboratoire, est également un des premiers organisme dont le génome a été séquencé [2].

Elle est à l'origine du domaine de la biotechnologie en étant choisie comme premier organisme génétiquement modifié contenant de l'ADN recombiné, c'est à dire de l'ADN dont la séquence des nucléotides à été altérée pour modifier l'expression des gènes de la bactérie. Elle est ensuite utilisée comme hôte pour la production de protéines hétérologues (provenant d'un autre organisme), notamment pour la production industrielle d'insuline humaine [14] mais aussi de vaccins. *E. Coli* est utilisée en bioremédiation pour la neutralisation de déchets toxique ou la dépollution des sites contaminés et pour la production d'enzymes. Les efforts de recherche se concentrent actuellement sur la synthèse par *E. Coli* de protéines plus complexes comprenant par exemple des ponts bisulfure ou nécessitant une modification post-traductionnelle pour être stables ou bien juste fonctionnelles [16].

Paramètre	**Symbole**	**Valeur**	**Unité**
Longueur	$L = 2(l + r)$	2,5(6)	µm
Largeur ou Diamètre	$d = 2r$	0,88(9)	µm
Volume	V	1,4	fL
Vitesse rectiligne	v	29(6)	$\mu m\, s^{-1}$
Temps de génération	G	20 — 30	min

TAB. 4.1: Quelques ordres de grandeur associés aux bactéries *Escherichia Coli* [6, 7][11]

La chimiotaxie de *Escherichia Coli* Comme la plupart des bactéries bacillaires, *E. Coli* possède une ciliature péritriche : elle est assortie de longs flagelles elliptiques, quatre en moyenne, recouvrant sa surface qui sont utilisés comme moteur de son déplacement. Ces flagelles sont fixés à leur base à un moteur rotatif réversible entraîné par un flux de protons et tournent sur eux-mêmes à une fréquence de l'ordre de 100 Hz. Suivant le sens de leur rotation, sens direct (CCW pour Counter ClockWise) et sens indirect (CW pour ClockWise), ils sont à l'origine de deux modes de déplacement :

- le mode « run », les moteurs tournent dans le sens direct et les flagelles se regroupent pour propulser la bactérie dans une direction rectiligne. Il dure en moyenne 1 s;
- le mode « tumble », au moins un des moteurs tourne dans le sens indirect. Les flagelles se désolidarisent ce qui a pour effet moyen de faire tourner la bactérie sur elle-même dans le sens horaire [9]. Ce mode dure en moyenne 0,1 s.

Ces deux modes de déplacement alternent successivement et, en fonction de l'environnement de la bactérie, les amènent à migrer vers des environnements favorables, riches en nutriments et à fuir les environnements toxiques et délétères [1]. Ce phénomène s'appelle la *chimiotaxie*. Il résulte de la transmission de signaux extracellulaires, détectés par les récepteurs de la membrane, aux moteurs des flagelles qui contrôlent le comportement de la cellule.

D'un point de vue macroscopique, *E. Coli* effectue une marche aléatoire biaisée vers une direction particulière. Elle détecte la variation *dans le temps* de la concentration d'un chimioeffecteur. Ainsi, lorsque la bactérie se dirige vers un milieu plus favorable, les réorientations par tumble deviennent plus rares et la bactérie peut remonter le gradient de la concentration d'une molécule qui se diffuse dans l'environnement. Entre deux run, *E. Coli* peut complètement changer de direction, avec un angle allant jusqu'à 180°, le plus souvent entre 0° et 90° [35]. D'un point de vue microscopique, *E. Coli* peut détecter différents acides aminés, sucres et dipeptides ainsi que l'acidité, la température et l'état d'oxydoréduction.

Les récepteurs principaux, comme ceux de l'aspartate (Tar) et de la sérine (Tsr), sont très nombreux et sont aux nombre de quelques milliers par cellule. Les récepteurs secondaires, comme ceux spécifiques aux dipeptides (Tap), ribose et galactose (Trg), et au potentiel d'oxydoréduction (Aer), sont bien moins nombreux avec seulement une centaine de copies par cellule. Ces récepteurs se regroupent en amas, ce qui induit des dynamiques complexes dans la détection de chimioeffecteurs [30], impliquant soit une plus grande sensibilité ou au contraire une plus grande diversité de détection.

Dans le réseau de régulation génétique, parmi les voies de signalisation de la cellule, la voie de réponse chimiotaxique consiste donc en un ensemble de protéines réceptrices transmembranaires sus-citées et les produits de six gènes chimiotaxiques : CheR, CheB, CheW, CheA, CheY et CheZ. L'information sur la liaison de ligants bénéfiques ou nocifs aux récepteurs est transportée jusqu'au moteur des flagelles ce qui leur permet de changer le sens de rotation du moteur. Il existe plusieurs modèles stochastiques capables de simuler en détail la cascade d'évènements entre la liaison d'un chimioeffecteur à un récepteur jusqu'au biais des moteurs [28, 24].

Modèle de croissance *E. Coli* suit un modèle de croissance linéaire [36, 21] par élongation et bien que son diamètre décroisse avec l'élongation, cette variation reste négligeable quand L est suffisamment grande [33, 29]. Nous considérerons donc que seule sa longueur L (voir Fig. 4.1) varie lors de la croissance d'une bactérie. Comme toutes les cellules procaryotes, *E. Coli* se reproduit par scissiparité. Ce cycle de division implique :

1. La réplication de la molécule d'ADN circulaire au sein de la cellule ;
2. La migration après duplication des deux brins d'ADN à chacun des pôles de la cellule ;
3. La croissance de la cellule pour laisser la place à un matériel génétique plus volumineux ;
4. Le plan équatorial de la cellule se resserre et fini par scinder la membrane en deux, de sorte que chaque nouvelle cellule fille ait le même matériel génétique.

L'équation 4.1 présente le calcul du temps de génération pour une population de bactéries dont la croissance s'effectue par division binaire, où t est le temps en minutes, n est le nombre de générations, B est le nombre de bactéries au début de la période de temps mesurée et b est le nombre de bactéries à la fin de la période de temps mesurée. Cela permet aux biologistes de retrouver le temps moyen d'un cycle cellulaire pour la population observée.

$$G = \frac{t}{n} = \frac{t}{3,3 \times ln(b/B)} \tag{4.1}$$

Vieillissement et mort cellulaire Il semble que les bactéries se scindant en deux bactéries filles en tout point identiques atteignent une sorte d'immortalité, au moins fonctionnelle. Néanmoins quand elle n'est pas tuée par des antibiotiques, par manque de nutriments ou par le système immunitaire d'organisme multicellulaires, même dans des conditions idéales *E. Coli* vieillit [31]. Ce vieillissement résulte d'une asymétrie entre les parois membranaires des pôles de la cellule. En effet, un de ces pôle est celui de la cellule mère, alors que le second provient du plan équatorial de scission de la cellule mère et a été fraîchement créé. En partant d'une unique cellule mère, à chaque génération deux cellules possèdent les parois membranaires des pôles de la cellule mère susmentionnée. Avec les générations, le matériel cellulaire diffusant peu et ayant une longue demi-vie s'accumule dans ces « vieux » pôles. Les cellules aux pôles plus âgés produisent moins de cellules filles et ont une probabilité de décès spontané plus élevée. On peut ainsi donner un âge à chaque pôle prenant en compte le nombre de divisions qu'il a subit. Cela permet d'attribuer un âge à chaque cellule, qui est définit comme l'âge du pôle le plus âgé. En conséquence, OTB inclus un modèle de vieillissement des bactéries pouvant causer le décès spontané (nommé apoptose) d'une cellule si sa lignée dépasse une certaine valeur, paramétrable en début de simulation.

4.1.5 Organisation logicielle du simulateur

Pour décrire l'organisation logicielle du simulateur, il nous faut parler des architectures de OpenCL et de OpenGL avant de présenter l'architecture de OTB afin de comprendre comment ces différents composants communiquent pour simuler l'évolution d'une population de bactéries.

Architecture OpenCL

Nous présentons ici une version simplifiée de l'architecture de OpenCL[7]. Un programme OpenCL est construit en deux parties : le programme hôte et les *kernel*. Le programme hôte s'exécute normalement sur CPU. Dans notre cas, c'est la partie du simulateur gérée par OCaml. Les kernels sont des fonctions écrites en C99 destinées à être exécutées sur les périphériques gérés par OpenCL, comme une carte graphique. Le modèle d'exécution d'OpenCL définit comment ces kernels sont instanciés sur les différentes unités de calcul disponibles.

Lorsque l'hôte soumet un kernel à exécuter à OpenCL, un espace d'indices est définit et, pour chaque indice de cet espace, une instance du kernel est exécutée. Cette instance est appelée un *work-item*. Les work-items sont regroupés en *work-group*, une division par blocs de l'espace d'indices. Les work-items ont accès à quatre types de mémoire : la mémoire globale, la mémoire constante, la mémoire locale et la mémoire privée. Chacune de ces mémoires est manipulable

7. http://www.khronos.org/registry/cl/specs/opencl-1.1.pdf

soit par l'hôte, soit par un work-item, soit par les deux. Par exemple, la mémoire privée est réservée à un work-item, la mémoire local est accessible à tous les work-items du même work-group, les mémoires constantes et globales sont accessibles à l'hôte comme aux work-items.

L'hôte peut également demander des transferts de données entre sa mémoire principale, la RAM, et les mémoires du device, dans les deux sens, afin de fournir des données initiales aux work-items et de récupérer leur travail.

Architecture OpenGL

Le framework OpenGL définit une architecture client-serveur : le client fonctionnant sur l'hôte, côté CPU, et le serveur s'exécutant sur la carte graphique, côté GPU. La pièce maîtresse du serveur est le *rendering pipeline* (chaîne de traitement du rendu), une succession de tâches permettant de transformer par étapes un ensemble de vertex, définis dans un espace en trois dimensions, en un tableau de pixels en deux dimensions.

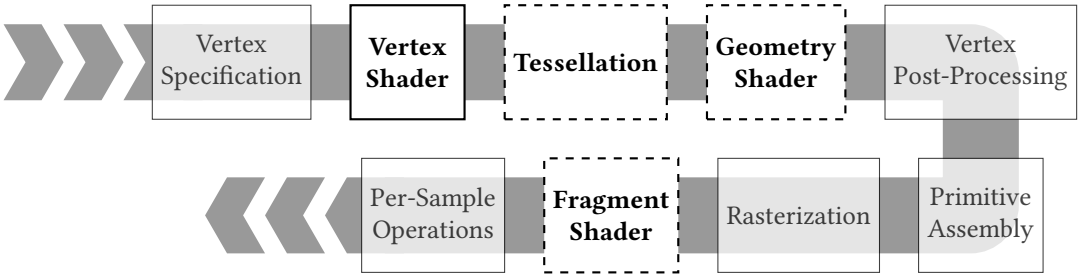


FIG. 4.3: Vue de haut niveau de la chaîne de traitement de OpenGL. Les shaders en gras peuvent être définis par l'utilisateur, les shaders en pointillé sont optionnels.

Ces étapes peuvent peuvent pour certaines être programmées à l'aide d'un langage appelé *GLSL* (OpenGL Shading Language), une variante de C, et sont appelées des *shaders* (voir FIG. 4.3). Voici une description succinte des shaders programmable utilisés dans OTB, avec leurs entrées, leurs sorties et une description de leur rôle :

Vertex Shader

- Entrée : un vertex ;
- Sortie : un vertex ;
- Description : ce shader modifie un vertex, par exemple ses attributs de position `vec4 gl_Position` ou de taille de point `float gl_PointSize`.

Geometry Shader

- Entrée : un vertex, une ligne, des triangles ;
- Sortie : zéro, un ou plusieurs vertex, ligne ou triangles ;
- Description : ce shader permet d'effectuer des calculs géométriques sur les primitives qui lui sont données, soit pour modifier leur attributs, soit pour générer de

nouveaux points, de nouvelles lignes ou de nouveaux triangles. Par exemple, dans OTB, ce shader est utilisé pour construire les vertex nécessaire à la représentation d'une bactérie, à partir de trois vertex donnés en entrée.

Fragment Shader

- Entrée : un fragment ;
- Sortie : une couleur ;
- Description : ce shader permet d'attribuer une couleur à un pixel sur l'écran à partir des informations calculées par le shader de Rasterization. C'est ici par exemple que la représentation de chaque bactérie obtient ses couleurs.

Architecture de OTB

L'architecture de OTB est organisée en modules OCaml. L'usage de modules a été une méthode intuitive pour rapidement prototyper le fonctionnement du simulateur. Les modules de OTB peuvent être découpés en deux catégories : les modules haut niveau, utilisés dans la programmation de la logique du simulateur et les modules bas niveau utilisés pour fournir des outils concrets aux modules de haut niveau, comme par exemple communiquer avec les périphériques d'entrée/sortie via des binding vers des bibliothèques externes. Les modules sont écrits avec OCaml, mais l'accès à OpenGL et OpenCL ne peut se faire que via le langage C, ainsi les modules de bas niveau `Viewer` et `OpenCL` sont en partie dédiés à la traduction entre l'interface C et OCaml.

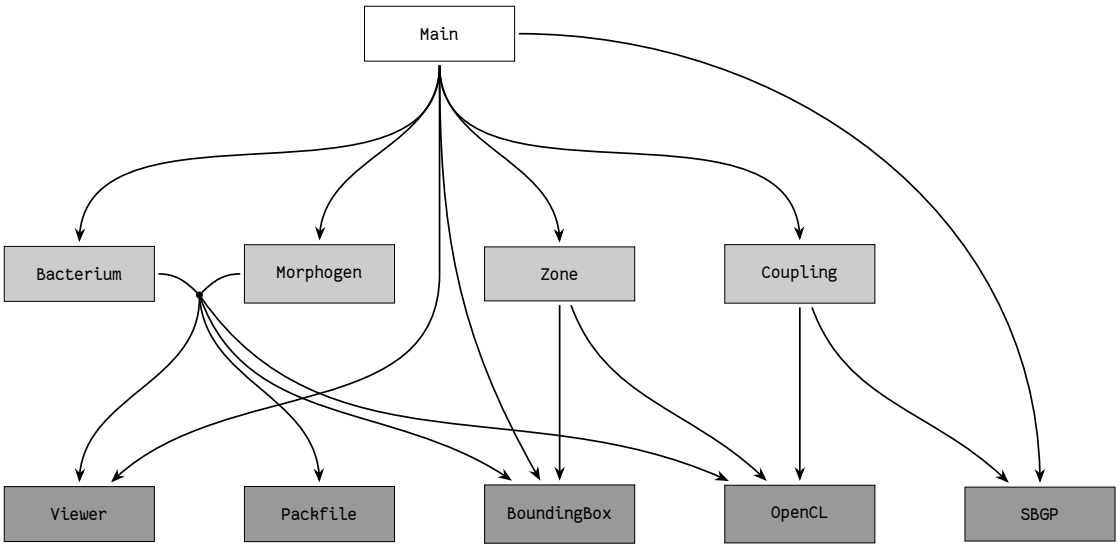


Fig. 4.4: Organisation des dépendances entre les modules de OTB. Les modules de bas niveau sont organisés sur le rang inférieur, les modules de haut niveau sont sur le rang supérieur.

Entrée Le point d'entrée par défaut du programme, à savoir la fonction d'entrée dans le programme, se trouve dans le module `Main`. Ce dernier fait appel aux autres modules pour construire une simulation et l'exécuter.

Modules de haut niveau Les modules de haut niveau regroupent les définitions des fonctions dédié au travail des modèles : d'une bactérie et d'une population de bactéries, d'un environnement de morphogènes, d'une zone délimitant les bords physiques de l'environnement de simulation et enfin de l'assemblage des trois modèles précédents. Comme indiqué par les flèches de la Fig. 4.5, un module de haut dépend d'un ou plusieurs modules bas niveau. Le module `Bacterium` décrit le modèle d'une bactérie ainsi que l'automate cellulaire support d'une population de bactéries. Sa fonction la plus notable, `Bacterium.run`, prépare et lance le calcul d'un pas de simulation d'une population de bactéries sur le GPU. Le module `Morphogen`, assez symétrique au module `Bacterium`, décrit l'automate cellulaire modélisant la réaction-diffusion des morphogènes. Sa fonction principale, `Morphogen.run` initialise et lance le calcul d'un pas de simulation d'une grille de morphogènes sur le GPU. Ces deux modules font appel à quatre autres modules de bas niveau qui sont décrits dans le paragraphe suivant. `Zone` est un module de haut niveau modélisant une boîte de Petri de taille dynamique en fonction de la répartition des individus qu'elle contient. Ses bords sont représentées par un cadre jaune dans la Fig. 4.1. Finalement, le module `Coupling` décrit le modèle complet obtenu par l'interaction entre les bactéries et leur environnement. C'est ce modèle qui permet aux bactéries de déposer des morphogènes dans leur environnement et de consommer les morphogènes qui y sont présent.

Modules de bas niveau Le module `Packfile` regroupe les fonctions de sérialisation nécessaire à la sauvegarde dans un fichier de l'état d'une simulation. Le module `BoundingBox` contient les types et les fonctions permettant de définir et d'effectuer des calculs courants sur des boîtes encadrantes, une représentation rectangulaire d'une zone géométrique.

Les modules `OpenCL` et `Viewer` sont des modules permettant d'accéder à des bibliothèques externes. En effet, OTB communique avec la carte graphique *via* le langage et framework OpenCL pour les calculs généraux. Dans un premier temps, nous avons fait appel à un module OCaml, appelé SPOC [3], fournissant un binding OpenCL et une gestion automatique de la mémoire de la carte graphique. Au delà d'une simple interface entre OCaml et OpenCL, SPOC abstrait la programmation GPGPU et donne un accès transparent à d'autres framework. Cependant, SPOC posant des problèmes de stabilité et possédant bien plus de fonctionnalités que nous avions besoin, nous avons finalement écrit notre propre interface bas niveau pour OpenCL, car nous souhaitons manipuler explicitement le transfert entre l'hôte et la carte. Cette interface a pour intérêt d'être simple, stable et de donner une grande latitude au développeur OCaml quant à la gestion de la mémoire entre l'hôte et la carte graphique. Ce binding est contenu dans le module `OpenCL`. OTB communique aussi avec la carte graphique *via* OpenGL. De la même manière, il existe plusieurs bindings OpenGL pour OCaml. Toutefois, nous n'utilisons qu'un faible sous-ensemble des fonctions de l'API d'OpenGL et nous avons écrit notre propre interface bas niveau, minimale et spécifique à l'utilisation de OTB. `Viewer` contient le binding vers OpenGL, glew et glfw, trois bibliothèques utilisées dans la création de l'interface graphique du

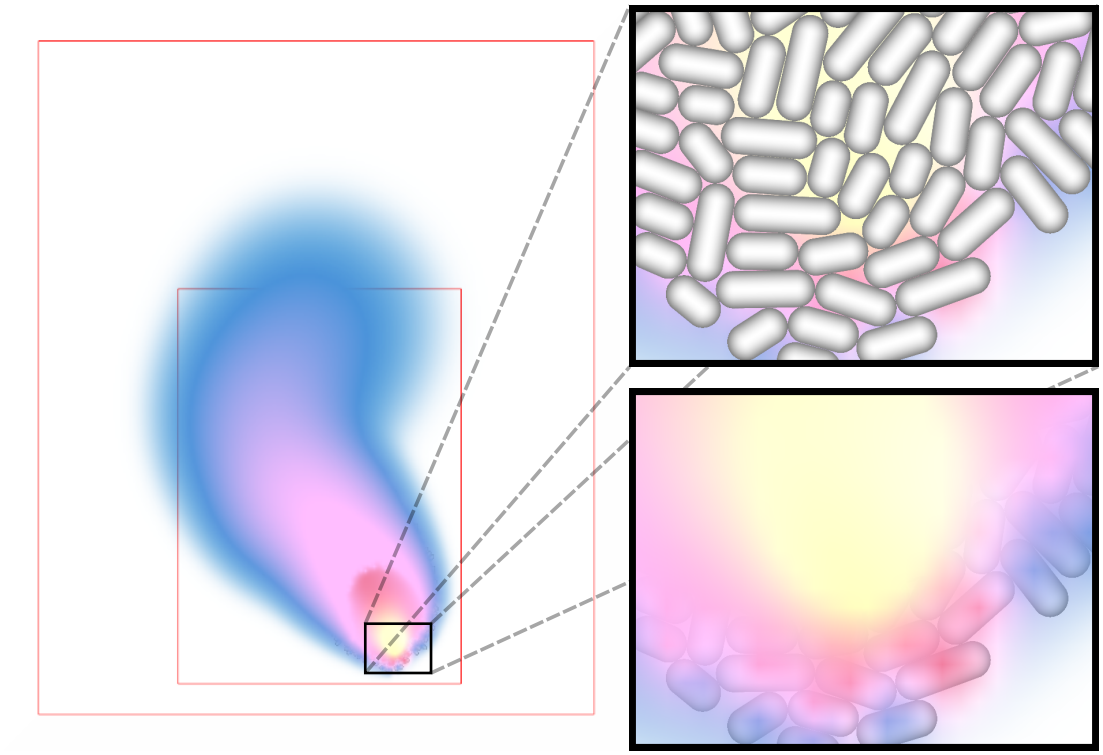


FIG. 4.5: Trois vues d'une même simulation avec 1174 bactéries présentant, à gauche, une vue globale montrant les zones dynamiques (délimitées en rouge, la zone intérieure est celle des bactérie, la zone extérieure est celle des morphogènes). À droite sont présentés des vues agrandies de la simulation mettant en évidence, en haut, le comportement physique des bactéries et, en bas, la diffusion des morphogènes.

simulateur.

4.1.6 Le langage SBGP

Pour répondre au point « Entrée » du cahier des charges, nous avons développé notre propre langage afin de faciliter le paramétrage d'une simulation [27] avec OTB. SBGP (Synthetic Biology Genetic Programming) est un langage déclaratif décrivant les options de configuration d'une simulation OTB. Un fichier de configuration de la simulation écrit avec SBGP donne une représentation abstraite du génome de la bactérie sous la forme d'une machine à états finis et inclus des informations supplémentaires pour paramétrer la simulation, comme les couleurs de rendu.

Syntaxe Un fichier SBGP est formaté en JSON [5]. Un génome SBGP est un objet JSON contenant les blocs suivants :

- `signals`: la liste des signaux en usage dans la simulation, chaque signal est définit par un triplet : nom du signal, coefficient de diffusion et coefficient de dégradation ;
- `reactions`: la liste des réactions, si elles existent. Chaque réaction est définit par un triplet : (r, p, τ) où r est une liste des réactants, p une liste des produits et τ le taux de réaction ;
- `type`: la liste des types identifiés par une étiquette unique ;
- `behavior`: le comportement, un objet donnant pour chaque type la liste des actions à exécuter à chaque pas de temps ;
- `transition`: une matrice de taille $card(\texttt{type}) \times card(\texttt{type})$ qui représente la table de transition d'état entre chaque types. La matrice donne l'identifiant de la transition si il existe.
- `cond_transition`: la transition conditionnelle identifiée par son étiquette ; chaque étiquette dans la table de transition d'état doit être défini dans ce bloc

Le type de chaque bloc est le suivant :

```
{
  "signals"         : [[Sid_i, Deg_rate_i, Diff_rate_i], [...]],
  "reactions"       : [[[Pid_i, Pid_j], [Rid_i], Rate_i], [...]],
  "type"            : [T0, ..., Tn],
  "behavior"        : {T0 : [{InstructionID : [Pid_1, ..., Pid_n]}]}
  "transition"      : [type x type] of Condition_name,
  "cond_transition" : [{Condition_name_i : ConditionID}, ..., {...}]
}
```

Déclaration des signaux Les signaux sont déclarés dans une liste. Chaque signal est un triplet contenant le nom du signal sous forme de chaîne de caractères, et les coefficients de diffusion et de dégradation sous forme de nombres réels.

```
  "signals" : [
    ["alpha", 0.05, 0.01],
    ["beta", 1.5, 1]
  ]
```

Déclaration des réactions Les réactions sont déclarés dans une liste. Chaque réaction consiste en un triplet : la liste des réactants, la liste des produits et la constante cinétique de la réaction.

Par exemple, le bloc SBGP suivant est utilisé pour décrire la réaction $2\alpha + \beta \xrightarrow{0.15} \gamma$:

```
  "reactions" : [
    [
      ["alpha", "alpha", "beta"],
      ["gamma"],
      0.15
    ]
  ]
```

Déclaration des types Les types de bactéries légaux sont déclarés dans une liste d'identifiants. Le bloc suivant donne la liste des types impliqués dans le génome :

```
"type" : [
  "QUIESCENT",
  "LEADER",
  "FLESH",
  "SKIN",
  "DEAD"
]
```

Déclaration du comportement Les comportements spécifiques aux types sont déclarés dans une liste associative ; une liste d'instructions est donnée pour chaque étiquette déclaré dans le bloc type. Par exemple, le code suivant donne le comportement d'une bactérie de type `LEADER`:

```
"behavior" : {
  "LEADER" : [ { "EmitSignal" : ["alpha", "150"] },
               { "StopDivide" : [] },
               { "Accumulate" : ["lambda","1"] }
  ]
}
```

Une instruction est décrite conformément à la structure suivante :

```
{ "InstructionID" : ["param1", ... ,"paramN"] }
```

La valeur du champ `InstructionID` doit être pris parmi l'ensemble des instructions SBGP suivantes :

- `EmitSignal(s,q)`: dépose une quantité `q` du morphogène `s` à la position courante de la bactérie dans l'environnement ;
- `AbsorbSignal(s,q)`: consomme une quantité `q` du morphogène `s` à la position courante de la bactérie dans l'environnement ;
- `Differentiate(A)`: force un changement de type de la bactérie vers le type `A` (il doit exister dans la liste des types) ;
- `Die()`: détruit la bactérie. Elle est en pratique marquée comme inactive et réinitialisée ;
- `Freeze()`: place le marqueur `immobile` sur une bactérie. Dans cet état une bactérie est immobile et ne croît pas ;
- `Unfreeze()`: retire le marqueur `immobile`;
- `Divide(r)`: définit le taux de croissance de la bactérie à la valeur particulière `r`;
- `StopDivide()`: définit le taux de croissance de la bactérie comme nul, l'empêchant ainsi de se diviser ;
- `Accumulate(p,q)`: incrémente le compteur de morphogènes `p` avec la valeur `q`;
- `Deplete(p,q)`: décrémente le compteur de morphogènes `p` avec la valeur `q`.

Déclaration des transitions Le bloc transition est une matrice encodant le graphe de différentiation du génome. La taille de la matrice dépend du nombre de types. Lorsque qu'une transition existe entre deux types, une étiquette est définie dans la case de la matrice correspondante, sinon, le mot-clé `NA` est spécifié. Le code SBGP suivant présente le graphe de différenciation d'un cœur homéostatique à deux couches : la population croît autour d'une bactérie mère en formant deux anneaux concentriques constitués d'une quantité invariante de bactéries.

```
"transition" : [
  ["NA", "CL", "C1" , "C2" , "NA" , "NA" ],
  ["NA", "NA", "NA" , "NA" , "NA" , "NA" ],
  ["NA", "NA", "NA" , "C12", "NA" , "NA" ],
  ["NA", "NA", "C21", "NA" , "C23", "NA" ],
  ["NA", "NA", "NA" , "NA" , "NA" , "C3D"],
  ["NA", "NA", "NA" , "NA" , "NA" , "NA" ]
],
```

Déclaration des conditions de transition Une condition est une expression booléenne renvoyant `true` ou `false`. La valeur du champ `ConditionID` doit être prise parmi l'ensemble des instructions SBGP suivantes :

- `AND(Cid1,Cid2)`, `OR(Cid1,Cid2)`: `AND` et `OR` sont les opérations classiques de la logique booléenne ;
- `EqThreshold(s,q)`: ce prédicat n'est vérifié que si la concentration du morphogène `s` est égale à `q`;
- `LessThreshold(s,q)`: ce prédicat n'est vérifié que si la concentration du morphogène `s` est inférieure à `q`;
- `GreaterThreshold(s,q)`: ce prédicat n'est vérifié que si la concentration du morphogène `s` est supérieure à `q`;
- `LessEqThreshold(s,q)`: ce prédicat n'est vérifié que si la concentration du morphogène `s` est inférieure ou égale à `q`;
- `GreaterEqThreshold(s,q)`: ce prédicat n'est vérifié que si la concentration du morphogène `s` est supérieure ou égale à `q`;
- `BetweenThreshold(s,q1,q2)`: ce prédicat n'est vérifié que si la concentration du morphogène `s` est comprise strictement entre `q1` et `q2`;
- `InternalProtEqCond(p,q)`: ce prédicat n'est vérifié que si le compteur de morphogènes `p` est égal à la valeur `q`;
- `InternalProtLessCond(p,q)`: ce prédicat n'est vérifié que si le compteur de morphogènes `p` est inférieur strictement à la valeur `q`;
- `InternalProtGreatCond(p,q)`: ce prédicat n'est vérifié que si le compteur de morphogènes `p` est supérieur strictement à la valeur `q`;
- `InternalProtLessEqCond(p,q)`: ce prédicat n'est vérifié que si le compteur de morphogènes `p` est inférieur ou égal à la valeur `q`;
- `InternalProtGreatEqCond(p,q)`: ce prédicat n'est vérifié que si le compteur de morphogènes `p` est supérieur ou égal à la valeur `q`;

- `MidPlane(a,b,t)`: ce prédicat n'est vérifié que si la valeur de la concentration du morphogène `a` moins la concentration du morphogène `b` est comprise entre `-t` et `t`.

4.2 Propagation Parallèle à la Margolus

Nous avons présenté l'architecture de haut niveau de OTB, ainsi que son langage de configuration, cette section présente l'algorithme de Propagation Parallèle à la Margolus (PPM). Cette algorithme forme le socle théorique que nous implémentons dans les kernels OpenCL. Nous commençons par un rappel sur les automates cellulaires, les automates à blocs, puis nous présentons l'algorithme de bounding-box dynamique et enfin PPM.

4.2.1 Automates cellulaires et voisinage de Margolus

Notre travail se situe dans le contexte des automates cellulaires. Un automate cellulaire est un modèle de calcul discret, introduit par John Von Neumann et Stanislaw Ulam dans les années 1960 [26]. C'est un modèle de calcul Turing complet qui a d'abord été objet de recherche académique sur les processus d'auto réplication, ensuite popularisé par le « Game of Life » de John Conway[8] où il est devenu un modèle pour les processus biologiques. Les automates cellulaires sont un outil des paradigmes de programmation non conventionnels, d'un côté permettant de formaliser les processus biologiques observés, et de l'autre de les modéliser et de les simuler.

Définition 4.2.1 (Configuration)**.** *Soit $\mathscr{C}$ un ensemble d'états, G un groupe, $c_{\mathscr{C}} : G \to \mathscr{C}$ est une* configuration *et $C_{\mathscr{C}} := \{c | c : G \to \mathscr{C}\}$ est* l'ensemble de toutes les configurations $c_{\mathscr{C}}$*. Quand le contexte lève tout ambigüité, on pourra omettre l'indice $\mathscr{C}$.*

Définition 4.2.2 (Automate cellulaire)**.** *Un automate cellulaire est un tuple $\mathscr{A} = (M, \mathscr{C}, N, f)$ dont :*

- $(M, +)$ *est un groupe abélien libre appelé* maillage des cellules*;*
- $\mathscr{C}$ *est* l'ensemble des états*, aussi parfois appelé alphabet,*
- $N \in M^k$, $k \in \mathbb{N}$ *est* l'indice de voisinage*, dont les éléments sont uniques deux à deux et $N = (n_1, n_2, \dots, n_k)$;*
- $f : \mathscr{C}^k \to \mathscr{C}$ *est la* fonction de transition locale

f induit une fonction de transition globale F, $\forall c \in C_{\mathscr{C}}$ et $m \in M$:

$$F(c)(m) := f(c(m + n_1), c(m + n_2), \dots, c(m + n_k))$$

La mise à jour d'une configuration se fait de manière locale, par l'application de f sur chaque cellule en fonction du voisinage, et *synchrone* avec des pas de temps discrets. Dans le cas d'un voisinage classique ou de Von Neumann, à gauche dans la Fig. 4.6, les voisins directs sont pris

8. lors de sa publication dans les pages du Scientific American en octobre 1970

en compte pour le calcul du nouvel état de chaque cellule. Du point de vue de MGS, un automate cellulaire est constitué d'un ensemble C de collections topologiques et d'une transformation T jouant le rôle de la fonction de transition globale de l'automate. Quelques restrictions s'appliquent sur chaque $c_i \in C$ et T: c_i doit être une collection newtonienne de type GBF et T doit utiliser la stratégie de réécriture `maximal-parallel`, afin que la mise à jour soit synchrone.

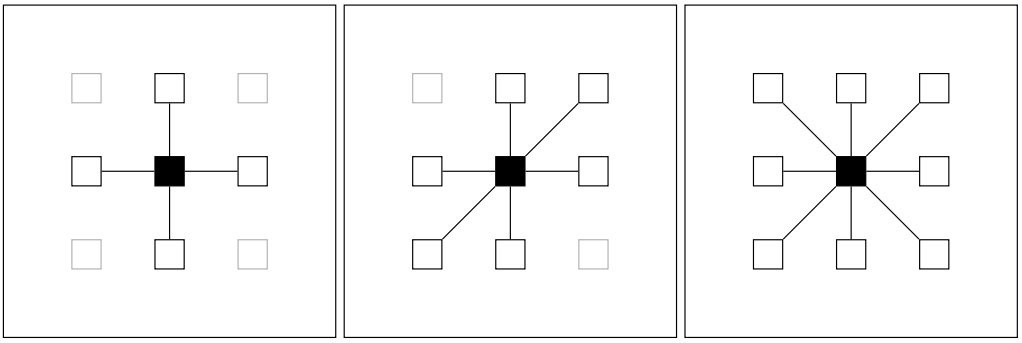


FIG. 4.6: Trois exemples de voisinages sur un grille carrée en 2D, à gauche un voisinage de Von Neumann, au centre un voisinage hexagonal et à droite un voisinage de Moore.

En tant que modèle de calcul, un automate cellulaire fonctionne de manière fortement parallèle : la même fonction est appliquée sur chaque cellule de la grille, ou autrement dit le même algorithme est appliqué à de multiples données ce qui est la définition d'une architecture SIMD. On pourrait donc tirer parti d'une architecture fortement parallèle pour implémenter un automate cellulaire. Néanmoins, la dépendance entre les données par la relation de voisinage entre les cellules pose encore problème. Une solution classique revient à utiliser une mémoire intermédiaire (un buffer) depuis lequel on va lire les données et écrire les résultat dans un second buffer, cassant ainsi la dépendance en écriture sur les cellules concernées. Cependant, il reste une *dépendance en lecture* sur les cellules voisines causant des collisions de lecture et pouvant encore porter atteinte à l'efficacité d'une implémentation parallèle. Pour résoudre ces problèmes, nous nous sommes tournés vers le voisinage de Margolus et de manière plus générale, le modèle de calcul des automates à blocs.

S'intéressant au développement d'ordinateurs simulant des systèmes physiques, Toffoli et Margolus introduisent dans [32, 22] les automates cellulaire à blocs (*block cellular automata*), ou automate à partitions (*partitioning automata*), pour modéliser le comportement dynamique de l'écoulement d'un fluide en respectant le principe de *réversibilité* des lois physiques. En thermodynamique, durant un processus réversible l'entropie n'augmente pas et le système est en état d'équilibre thermodynamique. On appelle *automate cellulaire réversible* les automates cellulaire dont on retrouve la configuration d'origine en « inversant » la flèche du temps. Un automate cellulaire dont chaque cellule est mise à jour en inversant sa valeur précédente et en ignorant son voisinage est trivialement réversible. Cependant, au delà d'une dimension, la réversibilité d'un automate cellulaire est indécidable [19]. L'automate gaz sur réseau (lattice gas automaton) [17], la règle « Tron » [32], l'ordinateur à boules de billard (billiard-ball computer) [13] sont des exemples d'automates cellulaires réversibles. Toffoli et Margolus montrent

également que la classe des automates à partitions est équivalente à celles des automates cellulaires, et que ces deux modèles peuvent se simuler l'un l'autre.

Définition 4.2.3 (Automate cellulaire à blocs)**.** *Un automate cellulaire à blocs, aussi appelé automate à partitions, est un tuple* $B = (M, \mathscr{Q}, p, \text{shift}, f)$ *avec :*

- $(M, +)$ *le maillage des cellules. C'est le groupe abélien* libre *engendré par l'ensemble fini* $A = \{g_1, ..., g_n\}$*;*
- $\mathscr{Q}$ *un ensemble d'états ;*
- $p : M \to M$ *la fonction de pavage qui pour* $\{c_1, ..., c_n\} \subset \mathbb{Z}^n$ *et* $\{k_1, ..., k_n\} \subset \mathbb{N}_+^n$ *associe à chaque point de* M *un* bloc*:*

$$p(\sum_{i=1}^{n} c_i\, g_i) = \sum_{i=1}^{n} \left\lfloor \frac{c_i}{k_i} \right\rfloor g_i$$

  La fonction inverse $p^{-1}(m) = \{m \in M | p(m) \in M\}$ *permet d'obtenir l'ensemble des éléments du bloc indicé par* m*.*
- shift *est la liste d'éléments* $(m_1, ..., m_n)$ *de* M *.*
- $f : Q^k \to Q^k$ *est la fonction de transition locale, avec* $k = \prod_{i=1}^{n} k_i$*.*

La fonction de transition locale f *induit une fonction de transition globale, dont la définition est décrite par l'algorithme en posant :*

```
1 forall s ∈ shift do
2 |   forall m ∈ M do
3 |   |   M ← f(c_1(T^s β(m))) ;
```

- T^s *l'opérateur de décalage sur les groupes Abéliens,* $T^s f(g) = f(g + s)$ *en prenant un groupe Abélien* G*,* $g \in G$ *et* $f : G \to G$*;*
- $\beta(m) = (p^{-1} \circ p)(m)$ *l'ensemble des éléments appartenant au bloc de la cellule* m *et*
- $c_1(B) : M \to Q^k$ *la configuration d'un ensemble de cellules* indexées par leur élément correspondant *dans le groupe. Nous n'expliciterons pas la version formelle ici pour ne pas alourdir la définition inutilement.*

Nous avons choisi d'utiliser la structure de groupe abélien libre pour représenter un maillage de cellule. C'est également le point de vue utilisé dans la définition des GBF à la différence que nous nous restreignons aux groupes libres, afin de pouvoir définir un pavage naturel. De ce point du vue, il semblerait qu'un automate cellulaire à bloc ne puisse pas avoir de maillage hexagonal, avec la présentation de groupe $H = \langle a, b, c \mid a + b = c \rangle$, car ce n'est pas un groupe libre. Néanmoins, nous pourrons considérer qu'un maillage hexagonal est un maillage carré, de présentation $H = \langle a, b \rangle$ sur lequel le voisinage entre cellules est spécifique comme présenté sur la Fig. 4.6 au centre. Par conséquent, c'est la fonction de transition locale qui déterminera quel voisinage est utilisé.

La fonction $\lfloor x/k \rfloor$ se comporte « naturellement » autour de 0: chaque image par cette fonction possède exactement k préimages. Cette caractéristique permet, par exemple, de partitionner $\mathbb{Z}$ en sous-ensembles de cardinalité k, d'où son usage dans la définition de p.

Pour la définition de la fonction de pavage, nous avons choisi une approche similaire à la définition d'une partition de l'espace passant par les $\mathbb{Z}$-modules comme introduit dans [25] à la section VI.1.1. Notre approche correspond mieux à l'idée intuitive d'une sous division de l'espace : à chaque cellule du maillage on fait correspondre une cellule du même maillage, ce qui expose immédiatement la structure et la régularité du pavage.

Le voisinage de Margolus est la règle de partitionnement du maillage d'un automate à blocs en une grille composée de blocs de 2 cellules (ou en carrés de 2×2 cellules en deux dimensions, en cubes de 2×2×2 cellules en trois dimension, ...) avec un décalage d'une « case » dans toutes les directions. C'est donc un cas particulier d'automate cellulaire à blocs, avec les restrictions suivantes (en reprenant le contexte de la définition des automates à blocs) :

- $p : M \to M$ la fonction de pavage qui pour $\{c_1, ..., c_n\} \subset \mathbb{Z}^n$ associe à chaque point de M un bloc de côté 2 :

$$p(\sum_{i=1}^{n} c_i\, g_i) = \sum_{i=1}^{n} \left\lfloor \frac{c_i}{2} \right\rfloor g_i$$

et

- shift est l'élément $\sum_{i=1}^{n} 1\, g_i \in M$.

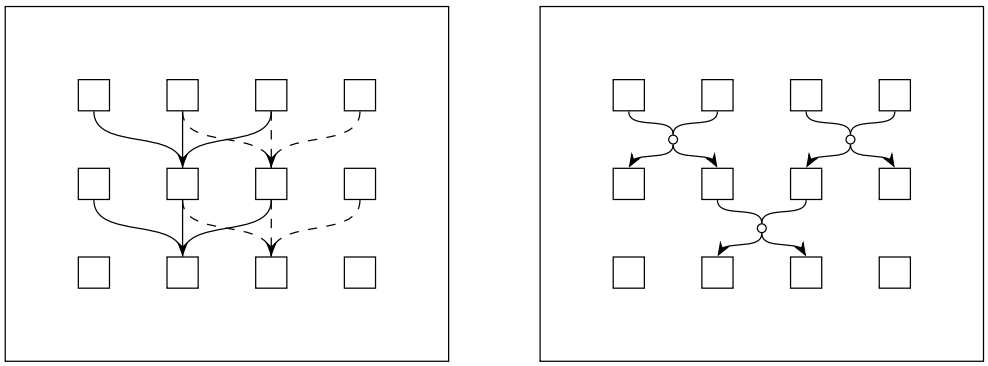


Fig. 4.7: Voisinage classique (à gauche) d'un automate cellulaire à une dimension et voisinage de Margolus (à droite). Pour le voisinage classique, chaque ligne correspond à un pas dans la simulation de l'automate, de haut en bas. Pour le voisinage de Margolus, un seul pas de simulation est présenté, la ligne du milieu est présentée afin de montrer la correspondance entre les deux automates.

4.2.2 Bounding box et activité

Dans OTB, l'environnement dans lequel évoluent les bactéries n'a pas *a priori* de taille fixe et subit une expansion ou une contraction en fonction de l'espace occupé par les bactéries, ou par leur support. Nous appelons *bounding box* la boîte englobante dont l'intérieur forme l'environnement dans lequel évoluent les bactéries, ou leur support. Cette bounding box a une taille rectangulaire libre. La bounding box est construite *a posteriori*: elle est déterminée par la taille de la grille d'automate qu'elle représente. De cette manière, il est possible d'accéder rapidement à la taille d'une population de bactéries et d'effectuer des calculs de collision entre bounding box et des opérations de séparation et de fusion. Ces opérations, bien que planifiées, ne sont que partiellement implémentées à l'heure de l'écriture de ce mémoire.

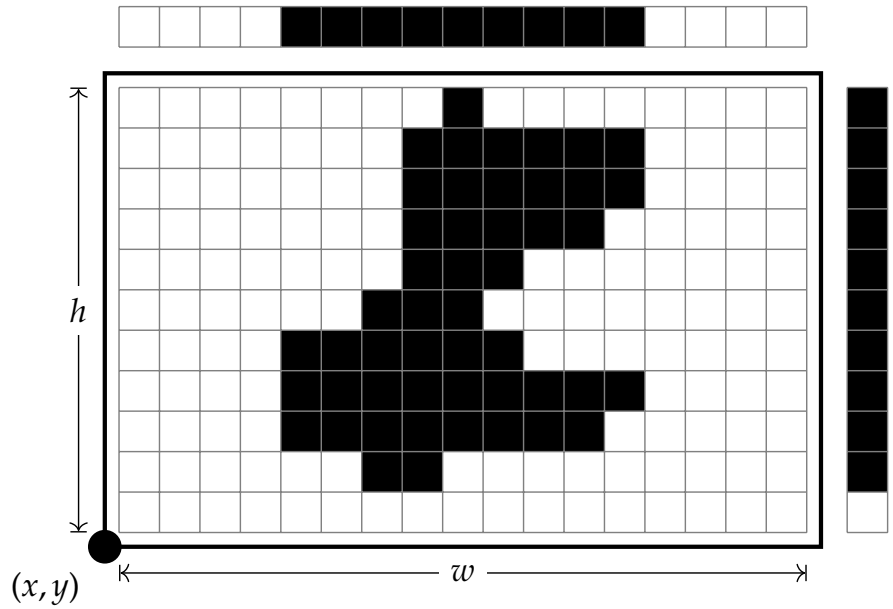


FIG. 4.8: Grille d'un automate cellulaire, ses indicateurs vertical et horizontal et sa Bounding Box (en gras)

Nous utilisons une mesure d'activité spatiale pour déterminer s'il est nécessaire d'agrandir ou de diminuer la taille de l'environnement. Chaque cellule de l'automate contenant une bactérie est considérée comme active. Si une cellule active se trouve à une distance inférieure à d_c d'un des bords, alors l'environnement est agrandi. À l'inverse, si la distance la cellule active la plus proche d'un bord à ce bord est supérieure à d_f alors, l'environnement est rétréci.

En pratique, une bounding box est définie par un triplet (p, w, h) où $p = (x, y)$ est un point dans le plan représentant le coin inférieur gauche de la boîte, w est sa largeur et h est sa hauteur (Fig. 4.8). Nous mesurons l'occupation pour chaque ligne et chaque colonne avec deux tableaux appelés *indicateur*: un pour l'occupation horizontale, l'autre pour l'occupation verticale. Ces deux tableaux sont suffisant pour déterminer s'il est nécessaire d'agrandir ou de rétrécir l'environnement. L'algorithme suivant est utilisé par OTB pour déterminer une nouvelle bounding

box optimale, w, h sont la largeur et la hauteur respectives de la bounding box de départ, c_w, c_h sont les tableaux indicateurs sur la largeur et la hauteur respectivement.

La première partie de cet algorithme détermine la position de coupe par rapport à chacun des quatre bords nord, sud, est et ouest de la bounding box. Il suffit pour cela de déterminer la position du premier élément non nul pour chaque extrémité de chaque indicateur, ce qui nous fournit le quadruplet (posN, posS, posE, posW) de positions de coupe au nord, au sud à l'est et à l'ouest respectivement.

La seconde partie de cet algorithme détermine la nouvelle dimension de la bounding box, en prenant en compte les deux marges notée d_c et d_f et en fonction des positions de coupe déterminées auparavant. La fonction `CropBoundingBox()` est chargée de la mise en œuvre du découpage de l'ancienne bounding box et en retourne une nouvelle en fonction du quadruplet (n, s, e, w) indiquant les corrections de marge à chacune des extrémité. Par exemple, n = 2 agrandi de 2 unités la taille de la bounding box au nord, alors que e = −4 rétrécie de 4 unités la taille de la bounding box à l'est. Cette fonction est implémenté dans le module de bas niveau `BoundingBox`.

```
   input  : Bounding box actuelle oldBB et posN, posS, posE, posW
   output : Une nouvelle bounding box ajustée
 1 if posN < d_c then                                          // Marge nord
 2 |   n ← d_c − posN;
 3 else
 4 |   if posN > d_f then
 5 |   |   n ← d_c − posN;
 6 |   else
 7 |   |   n ← 0;
 8 // Exactement la même opération pour sud, est et ouest
 9 return cropBoundingBox(oldBB, n, s, e, w)
```

Algorithme 1 : Création d'un nouvelle bounding box aux *bonnes* dimensions

4.2.3 Propagation Parallèle à la Margolus

L'algorithme PPM est inspiré du voisinage de Margolus, à droite dans la Fig. 4.7, en deux dimensions et l'utilise comme structure de données. Il effectue la simulation d'un automate cellulaire à blocs sur une machine parallèle d'architecture SIMD, nous l'avons implémenté sous forme d'un kernel OpenCL. Pour faciliter sa lisibilité, en voici une description un peu plus haut niveau que notre implémentation.

Cellule Chaque cellule est constituée de quatre emplacements : son propre état et l'état de trois de ses voisines comme présenté dans la Fig. 4.9. Ces quatre emplacements seront notés dans l'algorithme suivant par abus de la notation classique de tableau, avec les indices spéciaux 1, 2, 3 et 4. Ainsi, $m[a1]$ désigne le premier emplacement de la cellule A. La Fig. 4.9 (a) donne

une représentation visuelle de la position des cellules et de leurs emplacements en accord avec l'algorithme.

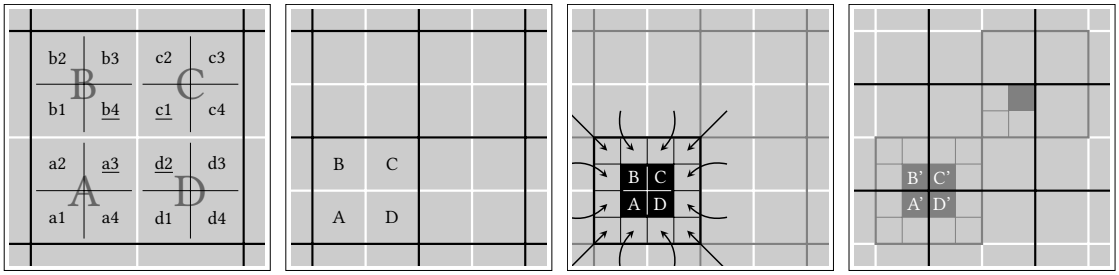


FIG. 4.9: *Général.* Quatre configurations successives d'un automate cellulaire en deux dimensions dont les cellules sont en gris. *(a).* Les coordonnées utilisées dans l'algorithme pour une cellule de Margolus. *(b).* Les cellules A,B,C et D font partie d'une cellule du voisinage de Margolus (en noir, en gras). *(c).* Chacune des cellules stocke quatre états : la copie de l'état de trois de ses voisines et le sien (en blanc sur fond noir). Le positionnement de ces états est présenté par rapport au voisinage de Margolus courant. *(d).* Après une étape de calcul, les états de A, B, C et D sont mis à jour en A', B', C' et D' et le voisinage de Margolus est décalé.

Algorithme PPM prend en entrée un tableau de cellules `cellSpace[w][h]` en deux dimensions de largeur `w` et de hauteur `h` et un indicateur `shift` déterminant le décalage du voisinage. La fonction d'évolution locale `evol(c,n1,n2,n3,n4,n5,n6,n7,n8)` prend en argument l'état courant `c` ainsi que l'état des 8 cellules voisines `n1,...,n8`.

La fonction de mise à jour est appelée en parallèle sur *le même tableau* de cellules et doit donc être paramétrée pour indiquer sur quelle cellule elle travaille : c'est le rôle des deux variables spéciales gc_x et gc_y indiquant la position globale en abscisse et en ordonnée du bloc courant. C'est donc la coordonnée d'une cellule du voisinage de Margolus qui est fournie. Ces deux variables sont les paramètres propres à un work-item OpenCL, qui permet à chaque instance du kernel d'avoir un comportement différent.

La cellule localement créée n est au cœur du fonctionnement de PPM. Il pourrait paraître en premier lieu étrange qu'elle soit recopiée à l'identique dans les quatre cellules de `cellSpace` constituant l'unique cellule de Margolus en cours de mise à jour. Néanmoins, en s'aidant de la FIG. 4.9 et en prenant en compte que le prochain décalage de la grille de Margolus se fait sur une cellule en diagonale, une symétrie apparaît.

La création d'un tableau local m de 2×2 est en revanche moins importante, cependant elle reste utile pour deux raisons : elle améliore sa lisibilité de l'algorithme et l'utilisation de variables locales en général nous permet de limiter le nombre d'accès à la mémoire globale en faisant un usage plus intensif de la mémoire privée (voir le paragraphe Architecture OpenCL page 110).

L'algorithme PPM travaille sur un bloc et, dans le voisinage de Margolus, chaque bloc est

input : Une configuration cellSpace d'un automate cellulaire en deux dimension, un indice de décalage shift valant 0 ou 1 et les coordonnées globales (gc_x, gc_y) du bloc actuel
output : Aucune (le changement est effectué par effet de bord)

1 $p_x \leftarrow 2 * gc_x +$ `shift` ; `// Coordonnées de la cellule`
2 $p_y \leftarrow 2 * gc_y +$ `shift`;
3 $i_x \leftarrow p_x + 1$;
4 $i_y \leftarrow p_y + 1$;
5 Nouveau tableau local de 2×2 cellules : m[2][2];
6 m[0][1] $\leftarrow$ `cellSpace`[$N * iy + px$];
7 m[1][1] $\leftarrow$ `cellSpace`[$N * iy + ix$];
8 m[0][0] $\leftarrow$ `cellSpace`[$N * py + px$];
9 m[1][0] $\leftarrow$ `cellSpace`[$N * py + ix$];
10 Nouvelle cellule locale : n;
11 n[1] $\leftarrow$ evol($m[a3]$, `// Cellule courante` A
12 $m[a1], m[a2], m[b1], m[b4], m[c1], m[d2], m[d1], m[a4])$;
13 n[2] $\leftarrow$ evol($m[b4]$, `// Cellule courante` B
14 $m[a2], m[b1], m[b2], m[b3], m[c2], m[c1], m[d2], m[a3])$;
15 n[3] $\leftarrow$ evol($m[c1]$, `// Cellule courante` C
16 $m[a3], m[b4], m[b3], m[c2], m[c3], m[c4], m[d3], m[d2])$;
17 n[4] $\leftarrow$ evol($m[d2]$, `// Cellule courante` D
18 $m[a4], m[a3], m[b4], m[c1], m[c4], m[d3], m[d4], m[d1])$;
19 cellSpace[$N * iy + px$] $\leftarrow n$;
20 cellSpace[$N * iy + ix$] $\leftarrow n$;
21 cellSpace[$N * py + px$] $\leftarrow n$;
22 cellSpace[$N * py + ix$] $\leftarrow n$;

Algorithme 2 : PPM

indépendant, ainsi il n'y a pas de problème de concurrence ni à l'accès au données ni à l'écriture. Chaque cellule embarque un quart de son voisinage et comme un bloc est constitué de quatre cellules, ce bloc contient toutes les informations nécessaires pour effectuer une mise à jour localement. Ainsi, alors que les algorithmes usuels de simulation d'automates cellulaires utilisent deux copies de la configuration actuelle pour éviter les collisions de lecture, PPM n'en a pas besoin.

4.3 Implémentation de OTB

Dans cette section nous décrivons les trois modèles (appelés moteurs) qui gouvernent le fonctionnement du simulateur : le moteur chimique, le moteur physique et le moteur de décision.

4.3.1 Moteur chimique : réaction et diffusion

Dans cette section nous présentons la fonction de transition de l'automate cellulaire en charge de la simulation chimique du milieu dans lequel évolueront les bactéries. Ce milieu a comme propriété de diffuser plusieurs morphogènes et de les faire réagir. L'équation de réaction-diffusion est un modèle mathématique qui permet, entre autres, de modéliser l'évolution de la concentration dans le temps et dans l'espace d'espèces chimiques diffusant et réagissant entre elles. Il a notamment été utilisée par A. Turing [34] a qui nous empruntons le terme de *morphogène* pour décrire une espèce chimique dans notre automate. Exprimée pour un seul morphogène dans une dimension d'espace, l'équation de réaction-diffusion est appelé l'équation de Kolmogorov-Petrovsky-Piskounov (KPP) [20].

Nous partirons d'une version généralisée de l'équation KPP en deux dimensions d'espace, avec un nombre $n \in \mathbb{R}_+$ de morphogènes :

$$\frac{\partial \vec{\varphi}}{\partial t} = \vec{\mathrm{X}}(\vec{\varphi}) + \vec{\mathrm{D}}^{\mathrm{T}} \cdot \left(\frac{\partial^2 \vec{\varphi}}{\partial x^2} + \frac{\partial^2 \vec{\varphi}}{\partial y^2} \right) \tag{4.2}$$

L'équation 4.2 donne la variation des concentrations de plusieurs espèces chimiques, ou morphogènes, dénotée $\vec{\varphi} = (\varphi_1, \dots, \varphi_n)$, au cours du temps et dans un espace à deux dimensions où $\vec{\mathrm{D}}^{\mathrm{T}} = (\mathrm{D}_1, \dots, \mathrm{D}_n)$ sont les coefficients de diffusion et $\vec{\mathrm{X}}^{\mathrm{T}} = (\mathrm{X}_2, \dots, \mathrm{X}_n)$ est un champ de vecteur décrivant la cinétique locale du système, autrement dit les réactions entre morphogènes.

Discrétisation Lors d'une simulation numérique, le temps et l'espace sont discrétisés et il existe différentes méthodes pour approcher par des méthodes discrètes la solution de l'équation 4.2. L'implémentation actuelle du simulateur utilise la méthode d'Euler, la méthode d'intégration numérique la plus simple :

$$\varphi_{k,i,j}^{t+\Delta t} = \Delta t \, \mathrm{X}_1(\varphi_{1,i,j}^t, \dots, \varphi_{k,i,j}^t) + \frac{\mathrm{D}_k \Delta t}{(\Delta x)^2}(\varphi_{k,i-1,j}^t + \varphi_{k,i+1,j}^t + \varphi_{k,i,j-1}^t + \varphi_{k,i,j+1}^t - 4\varphi_{k,i,j}^t) \tag{4.3}$$

où les indices (i, j) dénotent la position sur une grille carrée en 2D. Dans [8], les auteurs fournissent une classe de méthodes pour l'intégration numérique d'équations de diffusion et de réaction-diffusion approchant la solution exacte de l'équation de diffusion continue. En l'absence d'une validation plus poussée de notre outil, nous ne pouvons pas encore nous prononcer sur l'impact de l'erreur introduite par la discrétisation sur automate. Cependant, d'un point de vue qualitatif, le moteur chimique s'est suffisamment bien comporté pour les quelques tests que nous avons menés. En choisissant un Δt suffisamment petit, en l'occurrence $\Delta t = 0{,}01$, nous avons pu reproduire des réactions chimiques classiques, dont une réaction BZ (voir section 4.4.1).

Fonction de transition Chaque cellule de l'automate a pour état la concentration de chacun des n morphogènes de la simulation :

$$(conc_1, conc_2, \dots, conc_n)$$

Une table des réactions $RTable[i][j]$ est définie comme un tableau en deux dimensions avec selon i les n morphogènes et selon j les r réactions. Une liste des taux de réaction $RRate[j]$ donne le coefficient de vitesse pour la réaction j. Ces deux valeurs sont définies avant la simulation et sont donc accessibles à la fonction de transition locale du moteur chimique. Par exemple, dans le cas d'une simulation à trois morphogènes a, b, c où deux réactions r, s sont possibles

$$RTable = \begin{bmatrix} -1 & -2 & 2 \\ 0 & 2 & -1 \end{bmatrix} \qquad RRate = (\alpha, \beta)$$

désigne les réactions

$$a + 2b \xrightarrow{\alpha} 2c \qquad \text{et} \qquad c \xrightarrow{\beta} 2b$$

```
1 Fonction evol (c : cell, n1: cell, ..., n8: cell) : cell as
2     for i from 0 to n − 1 do                    // Diffusion de chacun des morphogènes
3         c'[i] ← c[i] + Δt ∗ D[i] ∗ 1/(Δx)² ∗ (n2[i] + n4[i] + n6[i] + n8[i] − 4c[i]);
4         for j from 0 to r − 1 do                // Résultat des réactions entre morphogènes
5             c'[i] ← c'[i] + Δt ∗ RTable[i][j] ∗ RRate[j];
6     return c';
```

Algorithme 3 : Fonction de transition locale du moteur chimique

Dans le cas de la diffusion sans réaction, la diffusion sur automate à bloc à voisinage de Margolus a été traité dans la littérature [23] et montre que le processus de diffusion est exactement simulé par un automate cellulaire booléen bloc synchrone sur voisinage de Margolus.

4.3.2 Moteur physique : collisions

Dans cette section, nous présentons la construction de la fonction de transition de l'automate cellulaire en charge de la simulation physique des bactéries. Le moteur physique représente

certains aspects de la mécanique du solide, notamment les translations sur le plan, la rotation et la collision entre objets. Nous utilisons un modèle de mécanique newtonien avec des simplifications spécifiques à l'échelle à laquelle les bactéries évoluent. Le modèle suivant est décrit dans un plan vectoriel en deux dimensions.

Modèle physique au repos Une bactérie est modélisée par une masse m, un centre de masse $p = (p_x, p_y)$, une longueur l et un rayon r donnant son épaisseur (voir Fig. 4.2 droite). Son orientation est représentée de manière équivalente, soit par un angle t, soit par un vecteur directeur $d = (d_x, d_y)$ en prenant $d_x = cos(t)$ et $d_y = sin(t)$. La longueur totale d'une bactérie est notée L et sa surface S. Nous ajoutons deux points $f = (f_x, f_y)$ et $b = (b_x, b_y)$ représentant respectivement l'avant et l'arrière de la bactérie

$$f_x = p_x + l \times cos(t) \qquad b_x = p_x - l \times cos(t)$$
$$f_y = p_y + l \times sin(t) \qquad b_y = p_y - l \times sin(t)$$

Du point de vue cinétique, une bactérie possède une vitesse de déplacement de norme v_b colinéaire à $\vec{d}$ et une vitesse de rotation de norme ω_b autour de p. Elle a également un taux de croissance g faisant varier sa masse, sa longueur et sa largeur.

La quantité de mouvement, aussi appelé moment linéaire, d'une bactérie est $\vec{p} = m\vec{v}_b$ et son moment angulaire, aussi appelé moment cinétique, autour de son centre d'inertie est $\vec{L} = I\vec{\omega}_b$ où le moment d'inertie I est similaire au moment d'inertie d'une tige de densité de masse uniforme, tournant autour de son centre de masse, de la même longueur que la bactérie, à savoir :

$$I = \frac{1}{12} \times m(2r + 2l)^2 = \frac{1}{6} \times m(r + l)^2$$

Du point de vue dynamique, nous considérons que les bactéries évoluent dans un milieu dont la viscosité est très élevée. Au centre de masse de chaque bactérie s'applique la somme des forces extérieures d'accélération linéaire F et accélération angulaire T transmises lors de collisions avec d'autres objets.

Collision entre deux bactéries En général, déterminer si deux objets sont en collision, c'est à dire déterminer si l'intersection des deux objets est non vide, n'est pas trivial, et ce pour deux raisons : les objets ont une forme potentiellement complexe et il peut y avoir un grand nombre d'objets. Pour les ceux qui ont une forme complexe, un objet de forme proche et plus simple, comme un disque ou un rectangle, appelé *bounding box*[9] est utilisée pour faciliter le calcul d'intersection. Dans le cas où un grand nombre d'objet peuvent entrer en collision, l'espace est divisé en petites sections afin que le test de collision n'ait lieu que pour un faible nombre d'objets.

Commençons par déterminer la fonction de collision pour deux bactéries. Cette fonction prend en entrée deux bactéries et retourne vrai s'il y a collision, avec la distance de collision la

9. différent de la bounding box que nous avons présenté avant, celle-ci est autour d'une unique bactérie, notre bounding box est autour d'une population de bactérie. Leurs fonctions sont toutefois similaires.

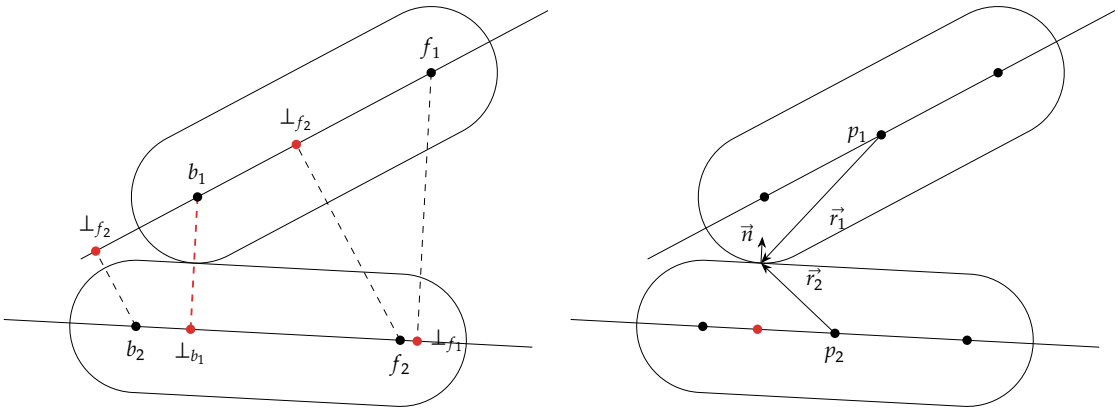


FIG. 4.10: La détection d'une collision a lieu lorsque l'un des pôles d'une bactérie est proche du corps d'une autre bactérie (à gauche). Quelques vecteurs utilisés dans le calcul de l'impulsion (à droite)

plus courte, les vecteurs de collision, les positions des collisions, ou faux s'il n'y a pas collision. Chaque bactérie est déterminée par un quadruplet comprenant le vecteur position du centre de masse, le vecteur directeur (unitaire), la longueur et le rayon respectivement $b_1 = (\vec{p_1}, \vec{d_1}, l_1, r_1)$ et $b_2 = (\vec{p_2}, \vec{d_2}, l_2, r_2)$. À partir de ces informations, nous pouvons obtenir les coordonnées des extrémités

$$\vec{f_1} = \vec{p_1} + l_1 \cdot \vec{d_1} \qquad \vec{f_2} = \vec{p_2} + l_2 \cdot \vec{d_2}$$
$$\vec{b_1} = \vec{p_1} - l_1 \cdot \vec{d_1} \qquad \vec{b_2} = \vec{p_2} - l_2 \cdot \vec{d_2}$$

Le critère de collision est simple. Grâce à la forme de la bactérie, il nous faut juste vérifier qu'aucun point du segment $[f_1, b_1]$ n'est à une distance inférieure ou égale à $d_{\min} = r_1 + r_2$ d'un point du segment $[f_2, b_2]$. Aussi, nous commencerons par construire les projections orthogonales restreintes à un segment des pôles $\vec{f_i}, \vec{b_i}$ d'une bactérie i sur le corps d'une bactérie j.

$$\vec{f_1^2} = \vec{p_2} + \left[\vec{d_2} + (\vec{f_1} - \vec{p_2})\right]_{-l_2}^{l_2} \cdot \vec{d_2} \qquad \vec{f_2^1} = \vec{p_1} + \left[\vec{d_1} + (\vec{f_2} - \vec{p_1})\right]_{-l_1}^{l_1} \cdot \vec{d_1}$$
$$\vec{b_1^2} = \vec{p_2} + \left[\vec{d_2} + (\vec{b_1} - \vec{p_2})\right]_{-l_2}^{l_2} \cdot \vec{d_2} \qquad \vec{b_2^1} = \vec{p_1} + \left[\vec{d_1} + (\vec{b_2} - \vec{p_1})\right]_{-l_1}^{l_1} \cdot \vec{d_1}$$

Restreindre la projection a un segment reflète le fait qu'une bactérie a une longueur *limitée*. Si on omet cette restriction, on obtient les projections de la FIG. 4.10, où la distance la plus courte n'est pas nécessairement indicatrice d'une collision. Il nous reste à obtenir les 4 vecteurs de collisions

$$\vec{n}_{f_{12}} = \vec{f_1^2} - \vec{f_1} \qquad \vec{n}_{f_{21}} = \vec{f_2^1} - \vec{f_2}$$
$$\vec{n}_{b_{12}} = \vec{b_1^2} - \vec{b_1} \qquad \vec{n}_{b_{21}} = \vec{b_2^1} - \vec{b_2}$$

dont les normes nous donnent bien la distance entre les corps des bactéries. Si $\|\vec{n}_{f_{12}}\| \leq d_{\min}$ ou $\|\vec{n}_{b_{12}}\| \leq d_{\min}$ ou $\|\vec{n}_{f_{21}}\| \leq d_{\min}$ ou $\|\vec{n}_{b_{21}}\| \leq d_{\min}$, alors il existe au moins une collision entre les deux bactéries. Le point d'impact se trouve sur le segment entre le pôle de la première bactérie et son projeté sur le corps de la seconde. Supposons que la collision ait lieu lorsque les corps des deux bactéries sont au contact, alors les points de collisions peuvent être les suivants

$$\vec{c}_{f_{12}} = \frac{r_1 \cdot \vec{f}_1 + r_2 \cdot \vec{f}_1^2}{r_1 + r_2} \qquad \vec{c}_{f_{21}} = \frac{r_2 \cdot \vec{f}_2 + r_1 \cdot \vec{f}_2^1}{r_1 + r_2}$$
$$\vec{c}_{b_{12}} = \frac{r_1 \cdot \vec{b}_1 + r_2 \cdot \vec{b}_1^2}{r_1 + r_2} \qquad \vec{c}_{b_{21}} = \frac{r_2 \cdot \vec{b}_2 + r_1 \cdot \vec{b}_2^1}{r_1 + r_2}$$

Modèle physique post-collision Pour calculer le résultat des échanges de forces lors d'un impact entre une bactérie i et une bactérie j, il est nécessaire de déterminer le type d'impact que nous souhaitons représenter. Dans le cas d'un impact parfaitement élastique, toute l'énergie transmise à la bactérie est restituée et l'énergie cinétique du système est conservée. Dans le cas d'un impact inélastique, une partie de l'énergie est dissipée et seule une partie de l'énergie totale est restituée. En une dimension, les vitesses post-collision de deux objets a et b sont données par les deux équations suivantes

$$v_a = \frac{Cr\ m_b(u_b - u_a) + m_a u_a + m_b u_b}{m_a + m_b} \qquad v_b = \frac{Cr\ m_a(u_a - u_b) + m_a u_a + m_b u_b}{m_a + m_b}$$

où

- $Cr \in [0, 1]$ est le *coefficient de restitution*, 0 pour une collision parfaitement inélastique et 1 pour une collision parfaitement élastique ;
- u_a, u_b sont les vitesses avant collision des objets a, b respectivement ;
- m_a, m_b sont les masses des objets a, b respectivement.

Dans le cas d'une collision en 2D, nous pourrions appliquer ces deux équations sur chacune des deux composantes des vecteurs vitesse, mais il nous manquerait les informations sur le moment angulaire de chaque bactérie. Nous utilisons un modèle un peu différent, ajusté à la collision de corps en 2D dont voici la description.

Les vecteurs vitesse *au point de collision* pré-collision sont

$$\vec{v}_{ac1} = \vec{v}_1 + \vec{\omega}_1 \wedge \vec{r}_1 \qquad \vec{v}_{ac2} = \vec{v}_2 + \vec{\omega}_2 \wedge \vec{r}_2$$

Notre objectif est de trouver les nouvelles valeurs $\vec{v}_i'$ et $\vec{\omega}_i'$ des deux bactéries après la collision. Écrivons d'abord les vecteurs vitesse au point de collision post-collision

$$\vec{v}_{pc1} = \vec{v}_1' + \vec{\omega}_1' \wedge \vec{r}_1 \qquad \vec{v}_{pc2} = \vec{v}_2' + \vec{\omega}_2' \wedge \vec{r}_2$$

Les vitesses relatives au point d'impact $\vec{v}_{ac12}, \vec{v}_{pc12}$ pré-collision et post-collision respectivement sont données par

$$\vec{v}_{ac12} = \vec{v}_{ac1} - \vec{v}_{ac2} \qquad \vec{v}_{pc12} = \vec{v}_{pc1} - \vec{v}_{pc2}$$

Dans notre modèle, nous faisons l'hypothèse que la vitesse à laquelle les deux bactéries se séparent est proportionnelle à la vitesse à laquelle les deux bactéries se rapprochent.

$$\vec{v}_{pc12} \cdot \vec{n} = -Cr\, \vec{v}_{ac12} \cdot \vec{n}$$

Nous retrouvons ici le coefficient de restitution $Cr \in [0,1]$ et $\vec{n}$ est le *vecteur normal unitaire* ($\|\vec{n}\| = 1$) au point de collision pointant vers l'extérieur de la seconde bactérie.

La collision en elle-même est modélisée par une *impulsion*. Pour une bactérie quelconque, rencontrer un obstacle équivaut à se voir appliquer une force au point de collision, dont la durée d'application est suffisamment courte pour qu'elle soit considérée *constante*. Cette application s'effectue *sans frottements* ce qui implique qu'elle soit dirigée selon $\vec{n}$, la normale de collision. Une impulsion $\vec{J} = j \cdot \vec{n}$ est l'intégration de cette force sur $\Delta_t = t' - t$ le temps d'application

$$\vec{J} = \int_t^{t'} \vec{F} dt$$

Comme d'après la seconde loi de Newton, une force est la dérivée temporelle d'une quantité de mouvement et que l'on suppose Δ_t suffisamment court pour que $\vec{F}$ ne varie pas, une impulsion est une variation entre les quantités de mouvement entre les temps t et t' et a donc la dimension d'une quantité de mouvement. La collision modifie les vitesses des deux bactéries selon $\vec{n}$ avec l'intensité j. En divisant le tout par la masse, on retrouve la dimension d'une vitesse. L'impulsion s'applique selon $\vec{n}$ pour la première bactérie et selon $-\vec{n}$ pour la seconde.

$$\vec{v}_1' = \vec{v}_{ac1} + j\vec{n}/m_1 \qquad \vec{v}_2' = \vec{v}_{ac2} - j\vec{n}/m_2$$

Les vitesses angulaires sont aussi modifiées de manière symétrique en prenant en compte la distance au centre de rotation des bactéries — si cette distance est nulle, la vitesse angulaire est nulle — et en divisant cette fois par le moment d'inertie de chacune des bactéries

$$\vec{\omega}_1' = \vec{\omega}_{ac1} + (\vec{r}_1 \wedge j\vec{n})/I_1 \qquad \vec{\omega}_2' = \vec{\omega}_{ac2} - (\vec{r}_2 \wedge j\vec{n})/I_2$$

Après résolution des équations précédentes pour j, on obtient

$$j = \frac{-(1+Cr)\, \vec{v}_{ac12} \cdot \vec{n}}{1/m_1 + 1/m_2 + (\vec{r}_1 \wedge \vec{n})^2/I_1 + (\vec{r}_2 \wedge \vec{n})^2/I_2}$$

et il est possible de déterminer le quadruplet $(\vec{v}_1', \vec{\omega}_1', \vec{v}_2', \vec{\omega}_2')$ post-collision. Plus généralement, l'impulsion résultant d'une collision entre une bactérie a et une bactérie b se notera j_{ab}.

Dans le cas de plusieurs collisions simultanées de la bactérie 1 avec les bactéries $(2, \dots, N)$, dans la mesure ou chaque bactérie est indépendante, les vitesses linéaires et angulaires post-collision de la bactérie 1 sont

$$\vec{v}_1' = \vec{v}_{ac1} + \sum_{k=2}^{N} \frac{j_{1k}\vec{n}}{m_1} \qquad \vec{\omega}_1' = \vec{\omega}_{ac1} + \sum_{k=2}^{N} \frac{\vec{r}_1 \wedge j_{1k}\vec{n}}{I_1}$$

l'impulsion étant le seul terme dépendant de la collision, il est différent pour chacune des collisions.

Fonction de transition locale La méthode naïve pour détecter les collisions entre un nombre arbitraire de bactéries est d'effectuer un test de collision pour chaque paire de bactéries, c'est à dire $\binom{n}{2}$ combinaisons, ce qui n'est pas atteignable pour une population de l'ordre de 10^4 individus avec les contraintes que nous nous sommes fixées. Pour résoudre ce problème, nous utiliserons le fait que, comme nous effectuons une simulation physique, nous partons du principe que tout ce qui cause un changement d'état du système a un *effet local* et se *propage* de proche en proche. C'est bien le chemin que nous avons emprunté en choisissant les automates cellulaires comme support de la simulation. Nous devons déterminer comment effectuer notre recherche de collision localement pour qu'elle soit correcte à l'échelle de la population. Nous allons donc restreindre la recherche de collision à un petit groupe de bactéries, contenu dans une cellule de l'automate. Le nombre de bactéries par cellule *nbpc* dépend de plusieurs paramètres :

- l'aire minimum d'une bactérie S_{min} (dépendant de l_{min} et r_{min}) ;
- l'aire d'une cellule C_p de diffusion du moteur physique.

Nous donnons une dimension physique concrète aux cellules car nous allons devoir superposer les deux automates cellulaires des moteurs physique et chimiques pour savoir où les bactéries déposent et consomment des morphogènes. De plus, nous gardons un rapport entier entre les aires des cellules C_p du moteur physique et C_c du moteur chimique afin que les deux grilles soient toujours alignées.

$$nbpc = \left\lfloor \frac{C_p}{S_{min}} \right\rfloor \qquad \text{et} \qquad C_p = n_r \times C_c$$

Le rapport entre C_p et C_c, noté n_r, est un entier.

L'espace ainsi segmenté en cellules est une *configuration*. Afin que notre automate cellulaire soit complet, il nous reste à spécifier la fonction de transition locale à utiliser pour mettre à jour cette configuration.

La fonction `getCollisions` effectue les calculs de collision présentés ci-dessus et retourne `true` si une collision est détectée et `false` sinon. Les variables $\vec{c}$, $\vec{n}$ et $\vec{j}$ sont passées par adresse et sont définies seulement dans le cas d'un collision.

```
1  Fonction evol (c : cell, n_1 : cell, ..., n_8 : cell) : cell as
2      for i from 0 to nbpc − 1 do
3          // Calcul de collision
4          v ← 0, ω ← 0;
5          forall c_n ∈ {c, n_1, ..., n_8} do
6              for j from 0 to nbpc − 1 do
7                  if c[i] ≠ c_n[j] and getCollisions(c[i], c_n[j], c⃗, n⃗, j⃗) then
8                      v ← v + j⃗/c[i].m, ω ← ω + ((c⃗ − c[i].p⃗) ∧ j⃗)/c[i].I;
9          // Mise à jour de position
10         updateBacteria(v, ω);
11     return c′;
```

Algorithme 4 : Fonction de transition locale du moteur physique

4.3.3 Moteur de décision : bactéries et morphogènes réunis

La coordination entre moteur chimique et moteur physique est le dernier problème à résoudre. En effet, pour être complet, les deux parties du simulateur doivent communiquer :

- les bactéries peuvent *lire* des valeurs des concentrations en morphogène *à leur position* et
- les bactéries peuvent *consommer* ou *déposer* des morphogènes à leur position.

Lire, consommer et déposer des morphogène implique que la fonction d'évolution des bactéries doit avoir connaissance de la configuration de l'automate chimique et doit pouvoir la modifier. Il n'est pas souhaitable que les bactéries écrivent directement dans les cellules de la configuration de l'automate chimique car ces valeurs dépendent du temps et sont intégrées numériquement. Une meilleure solution est de faire déposer par les bactéries une contribution ϵ qui sera ajoutée lors de l'intégration numérique pendant la mise à jour de l'automate chimique. Nous reprenons l'équation 4.3 auquel nous ajoutons un nouveau terme :

$$\varphi_{k,i,j}^{t+\Delta t} = \Delta t \, \mathrm{X}_1(\varphi_{1,i,j}^t, \dots, \varphi_{k,i,j}^t) + \frac{\mathrm{D}_k \Delta t}{(\Delta x)^2}(\varphi_{k,i-1,j}^t + \varphi_{k,i+1,j}^t + \varphi_{k,i,j-1}^t + \varphi_{k,i,j+1}^t - 4\varphi_{k,i,j}^t) + \Delta t \, \epsilon_{\varphi_k,i,j}$$

Ensuite, les deux configurations des automates physique et chimique doivent se superposer dans l'espace afin de lier la position d'une bactérie à la concentration en morphogènes à cette position. À la position du centre d'une bactérie (p_x, p_y) correspond une cellule de la configuration de l'automate cellulaire du moteur chimique

$$\left(\frac{p_x}{\Delta x}, \frac{p_y}{\Delta x}\right)$$

où Δx est la taille du côté d'une cellule de cet automate et à condition qu'à la position $(0, 0)$ corresponde la cellule de coordonnées $(0, 0)$.

La contrainte suivante concerne la taille des configurations. Nous avons indiqué dans la section 4.2.2 que les dimensions des configurations variaient en fonction de leur contenu. Comme une bactérie doit toujours pouvoir lire, écrire ou consommer des morphogènes, cela implique que la partie de l'espace S_c couvert par la configuration du moteur chimique doit inclure la partie de l'espace S_p couvert par la configuration du moteur physique, soit $S_p \subset S_c$.

Finalement, lorsque les bactéries ont accès aux concentrations de morphogènes, elles peuvent s'en servir pour modifier leur comportement ce qui est l'objet du moteur de décision. Chaque bactérie a un état prédéfini en début de simulation qui regroupe les différents variables normales et booléennes (dont la valeur par défaut est donnée entre parenthèses) suivants

- Division (vrai) : si vrai, la bactérie croît et se divise quand sa masse a doublé ;
- Décès (faux) : si vrai, la bactérie décide d'entrer en apoptose et disparaît de la simulation ;
- TumbleRun (vrai) : si vrai, la bactérie tourne sur elle-même, sinon la bactérie se déplace selon une trajectoire rectiligne uniforme vers l'avant ;
- Production/Consommation du morphogène φ_i: la bactérie dépose une certaine quantité de morphogènes à sa position dans l'environnement. Cette quantité est déterminée par le modélisateur en début de simulation.

Les transitions entre les états sont également donnés en début de simulation dans un fichier de description de simulation SBGP.

Le fonctionnement du moteur de décision est donné par la fonction de transition décrite dans l'algorithme 5. Cette dernière est appelée sur chaque cellule de la configuration de l'automate du moteur physique directement, sans passer par PPM car il n'y a pas conflit en lecture ou en écriture et peut être parallélisée naïvement. Cette fonction a de plus à sa disposition la configuration de l'automate du moteur chimique H ainsi qu'une configuration E des contributions aux concentrations des morphogènes de la même taille que H. Nous considérerons une simulation où il y a N morphogènes. Dans l'algorithme 5, toutes les valeurs précédées de `init` sont issues de l'initialisation de la simulation.

En prenant en compte toutes les contraintes précédentes, l'exécution d'un pas de simulation s'effectue de la façon suivante :

1. un pas du moteur physique,
2. un pas du moteur chimique,
3. redimensionnement des configurations et
4. un pas du moteur de décision.

```
1  Fonction evol (c : cell, H : configuration, E : configuration) : cell as
2      for i from 0 to nbpc − 1 do
3          // Indice de la cellule de H correspondant à la bactérie courante
4          j ← projection(H, (c[i].px, c[i].py));
5          // Mise à jour de la bactérie pour chaque morphogène
6          for k from 0 to N − 1 do
7              E[i][k] <- E[i][k] + init.behavior.(c[i].type).EmitSignal[k].value
8          // Transition vers un nouveau type
9          for t in init.transisition.(c[i].type) do
10             if init.cond_transition.t then
11                 c[i].type ← t;
12                 break;
13     return c′;
```

Algorithme 5 : Fonction de transition locale du moteur de décision

4.4 Mise en œuvre

Dans cette section nous présentons trois exemples de simulation avec OTB. Pour chacun de ces exemple nous fournissons le rendu de l'interface graphique et le fichier SBGP qui l'a engendré. Ces trois exemples mettent en évidence le fonctionnement de chacun des moteurs (chimique, physique et de décision).

4.4.1 Une réaction de Belooussov-Jabotinski

Les réactions de Belooussov-Jabotinski (BZ) sont une classe de réactions chimiques dans laquelle les concentrations de plusieurs réactants oscillent périodiquement jusqu'à épuisement de l'un d'entre eux. Leur importance réside dans le fait que l'état d'équilibre n'est pas atteint immédiatement mais après une série d'oscillations pouvant prendre plusieurs minutes. Quand elle est effectuée dans une boîte de Petri, il est possible d'observer des vagues de réactions.

Dans le cas de notre moteur chimique, nous pouvons facilement spécifier les réactions et les couleurs associés à chacun des morphogènes. Nous construisons donc cette réaction avec le minimum de réactifs et de réactions nécessaires, soit deux réactifs *bleu* et *vert* et les trois réactions suivantes :

$$bleu + vert \xrightarrow{0.05} 2vert$$
$$bleu \xrightarrow{0.05} 2bleu$$
$$vert \xrightarrow{0.05} .$$

La configuration de la simulation décrite en SBGP est la suivante

```
{
  "signals"         : [ ["bleu", 0.0, 1.0], ["vert", 0.0, 1.0] ],
  "reactions"       : [ [["bleu","vert"], ["vert","vert"], 0.05],
                        [["bleu"],        ["bleu","bleu"], 0.05],
                        [["vert"],        [],              0.05] ],
  "type"            : [],
  "behavior"        : {},
  "transition"      : [],
  "cond_transition" : []
}
```

où les quatre derniers champs sont vides car il n'y a pas de bactéries à simuler. Durant la simulation, il est possible d'observer une alternance entre les concentrations des deux morphogènes (en vert et en bleu) au cours du temps. Comme les morphogènes diffusent, ne s'évaporent pas et que l'environnement est ouvert, les concentrations locales baissent et il devient de plus en plus difficile de distinguer les deux morphogènes au centre. Par contre, des vagues de concentrations alternant bleu puis vert se forment à la périphérie conformément aux réactions BZ effectuées en boîte de Petri.

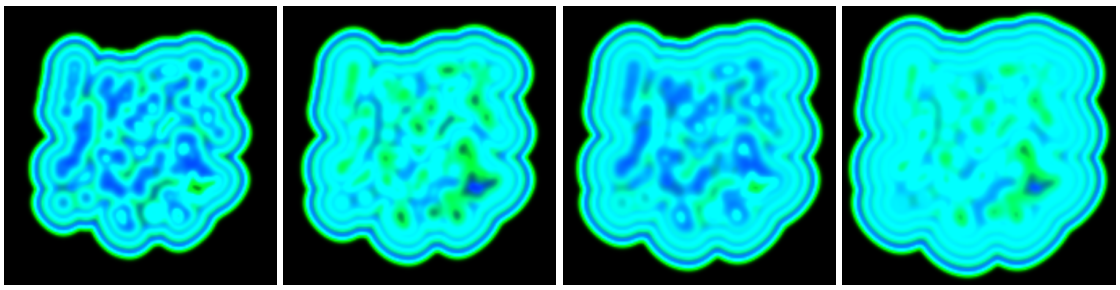


FIG. 4.11: Quatre états d'une simulation de réaction BZ ordonnés en fonction du temps de gauche à droite. Il est possible de noter l'alternance entre le vert et le bleu au cours du temps

4.4.2 Sectorisation d'une population de *E. Coli*

Pour la validation du moteur physique, nous avons reproduit une expérience de la littérature [15]. Dans cet article, les auteurs étudient la ségrégation génomique à la frontière entre deux populations de bactéries en croissance.

Protocole expérimental Les auteurs utilisent deux souches de bactéries *Escherichia Coli* de génotypes exactement identique, à l'exception d'une unique mutation du même gène promoteur induisant

- la première souche à produire une protéine fluorescente cyan (CFP) et
- la seconde souche à produire une protéine fluorescente jaune (YFP).

On répartit de façon homogène ces deux souches pour obtenir des cultures de différentes proportions. Une gouttelette de cette culture est posée au centre d'une gélose pour dénombrement dans un milieu contenant un médium de croissance enrichi afin d'obtenir une croissance rapide des deux populations. Comme les bactéries utilisées sont immobiles, le développement de la population s'effectue aux bords. Dans le cas d'un mélange équilibré contenant 50% de chaque souche, il est possible d'observer des secteurs monochromes dans la zone d'expansion de la population.

D'après les auteurs de [15], la Fig. 4.12 à gauche : « [...] is a visible manifestation of random genetic drift acting at the leading edge of a range expansion ». La dérive génétique (genetic drift) est l'évolution d'une population ou d'une espèce causée par des phénomènes aléatoires, impossible à prévoir. Du point de vue génétique, c'est la modification de la fréquence d'un allèle, ou d'un génotype, au sein d'une population, indépendamment des mutations, de la sélection naturelle et des migrations.

Simulation avec OTB Dans notre simulation de cette expérience, nous avons utilisé deux états différents ayant le même comportement pour les deux souches de bactéries afin de pouvoir colorer les deux sous populations a posteriori. Voici le protocole que nous avons suivi :

1. nous avons initialement laissé croître une population initiale issue d'une unique bactérie jusqu'à 1054 individus, pour obtenir une goutte de l'ordre du dixième de millimètre ;
2. nous avons marqué chaque individu par l'état A avec une probabilité de 50%, les autres ont été marqué par l'état B;
3. Nous avons repris la croissance de cette population jusqu'à atteindre une population de 271 958 individus (voir Fig. 4.12) après vingt minutes de calcul sur un serveur muni d'une carte de calcul Tesla[10];
4. nous avons fait émettre aux individus de type A un morphogène de couleur verte et aux individus type B un morphogène de couleur rouge, afin de pouvoir observer qualitativement la répartition des bactéries.

Bien qu'idéalement il nous faudrait une population de l'ordre de 10^6 individus pour être dans les mêmes conditions que le protocole expérimental, nous avons obtenu un résultat qualitativement très proche des photos présentées dans l'article original. Étonnamment, nos bactéries rudimentaires, par les seules contraintes mécaniques du moteur physique, se répartissent en secteurs lors de la croissance de la population.

10. NVidia Tesla est une carte de calcul dédié munie de plusieurs GPU

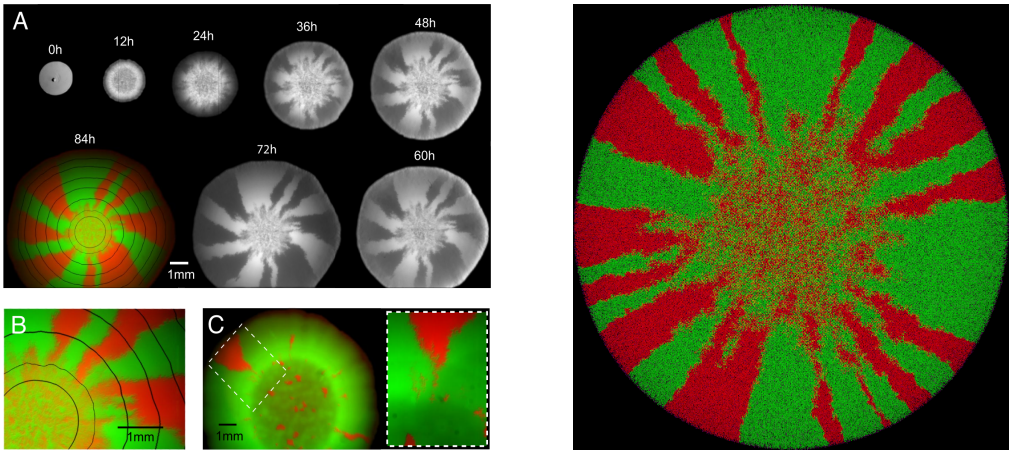


FIG. 4.12: Deux population de bactéries *E. Coli* croissant librement dans un milieu riche en nutriment. La figure de droite est issue de la fenêtre de rendu du simulateur OTB. La figure de gauche est issue de [15] pour comparaison

4.4.3 Une population stable ?

Les bactéries croissent, se divisent et entrent en apoptose. Nous avons tenté dans cette simulation d'obtenir une population stable ou au moins croissant linéairement. Intuitivement, l'idée que avons suivie est de faire en sorte que les bactéries croissent en cercles concentriques autour de la bactérie mère initiale. Cette dernière dépose un morphogène, appelé « Vert » dans le modèle, et se divise en deux bactéries : une nouvelle bactérie mère et une bactérie fille qui change d'état immédiatement. Cette nouvelle bactérie fille dépose un morphogène « Rouge » dans l'environnement. L'ensemble des bactéries filles forme alors un premier anneau autour de la bactérie mère. Si ces dernières ne perçoivent pas un taux de morphogène « Vert » suffisamment grand, signifiant qu'elles sont trop éloignées, alors elles changent à nouveau d'état.

Voici le comportement que nous avons spécifié en SBGP :

```
{
  "signals"          : [ ["Vert",  0.01,  2.5],
                         ["Rouge", 0.001, 1.5],
                         ["Bleu",  0.001, 1.5] ],
  "reactions"        : [],
  "type"             : [ "LEADER",
                         "CDAUGHTER",
                         "FDAUGHTER",
                         "DEAD" ],
  "behavior"         : {
    "LEADER" : [ { "Divide" : [] },
                 { "Tumble" : [] },
                 { "EmitSignal" : [ "Vert", 50 ] } ],
    "CDAUGHTER" : [ { "Divide" : 0.005 },
                    { "Tumble"     : [] },
                    { "EmitSignal" : [ "Rouge", 50 ] } ],
```

```
    "FDAUGHTER" : [ { "Divide" : 0.005 },
                    { "Tumble"      : [] },
                    { "EmitSignal" : [ "Bleu", 50 ] } ],
    "DEAD" : [ { "Die" : [] } ]
  },
  "transition"       : [
    ["NA", "NA", "NA", "NA"],
    ["CD", "NA", "NA", "NA"],
    ["NA", "CF", "NA", "NA"],
    ["NA", "NA", "CA", "NA"]
  ],
  "cond_transition" : [
    {"CD" : "IsDaughter()"},
    {"CF" : "LessThreshold(Vert,0.1)"},
    {"CA" : "And(LessThreshold(Rouge,0.1),LessThreshold(Bleu,10.0))"}
  ]
}
```

Les bactéries peuvent être de quatre types :

1. `LEADER` est la première bactérie de la simulation. Elle dépose le morphogène vert dans l'environnement ;
2. `CDAUGHTER` sont les bactéries filles directes de la bactérie de type `LEADER`. Elles déposent le morphogène `Rouge` dans l'environnement ;
3. `FDAUGHTER` sont les bactéries filles qui sont assez éloignées de la bactérie de type `LEADER` d'après la concentration en morphogène `Vert`. Elles déposent le morphogène `Bleu` dans l'environnement ;
4. `DEAD` sont les bactéries filles qui se sont trop éloignées de la bactérie de type `LEADER` et qui meurent quand les concentrations en morphogènes `Vert` et `Bleu` sont trop faibles.

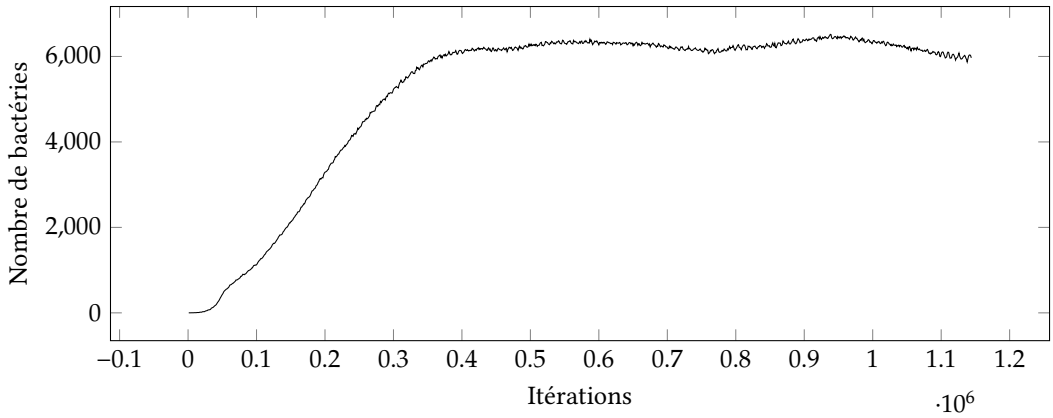


FIG. 4.13: Graphique de la simulation « Une population stable ? » présentant le nombre de bactérie en fonction du temps de simulation. Après une phase de croissance linéaire, la population se stabilise autour de 6000 individus

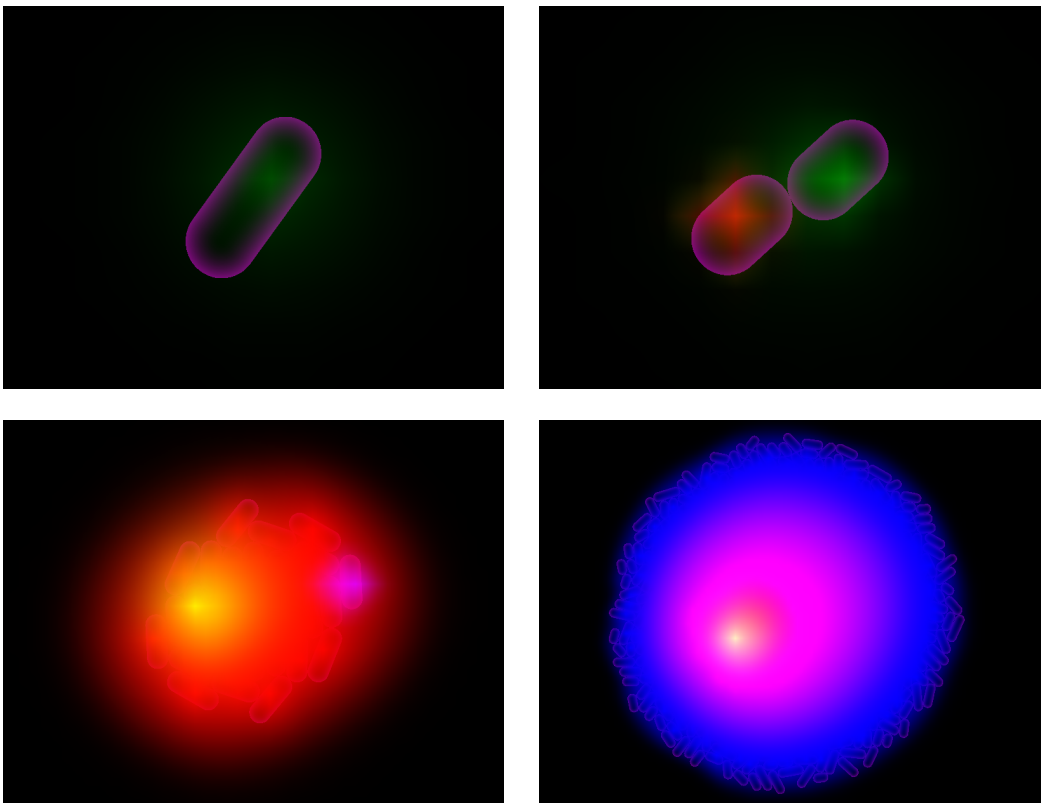


FIG. 4.14: Différentes étapes de la simulation « Une population stable ? ». La bactérie mère (en haut à gauche) se divise une première fois et sa fille change d'état (en haut à droite). Après un certain nombre de divisions, une des bactérie est suffisamment éloignée de la bactérie mère et change d'état (en bas à gauche). Plus tard dans la simulation, les bactéries « bleues » trop éloignées de la bactérie mère meurent à la périphérie.

4.5 Conclusion et perspectives

Cette courte section vise à rappeler les principaux résultats obtenus suite à nos travaux sur le simulateur de bactéries OTB ainsi que nos futures pistes de travail sur le sujet.

4.5.1 Conclusion

Dans ce chapitre nous présentons OTB, un simulateur d'une population de bactéries *Escherichia Coli*. Nous débutons ce chapitre en présentant nos motivations pour le développement d'OTB ainsi que les choix techniques que nous avons fait : l'utilisation de OCaml, de OpenCL et OpenGL. Nous avons ainsi décidé de tirer partie du parallélisme présent dans les ordinateurs grand public, ce qui nous a amené à développer PPM, un algorithme inspiré des travaux des chapitres précédents (section 4.2). PPM est conçu de sorte que le contexte nécessaire à l'exécution de chaque kernel lui soit immédiatement disponible. Ainsi le travail de chaque kernel est effectué de manière autonome et en parallèle sans accès concurrent à la mémoire. PPM est utilisé à la fois pour l'exécution du moteur chimique et pour l'exécution du moteur physique. Ces deux moteurs étant indépendants, la conception de OTB repose sur un dernier moteur de décision permettant aux bactéries et à leur support de communiquer par le couplage des moteurs chimique et physique. Ces trois moteurs sont présentés dans la section 4.3. Nous clôturons le chapitre par trois simulations permettant de rendre compte du fonctionnement du simulateur (section 4.4), montrant le fonctionnement des moteurs chimique, physique et de décision.

De nos travaux sur OTB nous tirons deux résultats :

- un résultat théorique : l'algorithme PPM est générique et peut-être adapté à d'autres outils travaillant sur les automates cellulaires ;
- un résultat pratique : le simulateur OTB est un exécutable utilisable librement sur différentes plateformes, principalement pour l'étude de la morphogénèse dans les populations de *E. Coli*. Il est modulaire et extensible et pourrait donc aisément être adapté à d'autres types de population.

4.5.2 Perspectives

Nos travaux futurs s'articulent autour des grands axes suivants, dont l'objectif principal est la pérennisation d'OTB comme outil de simulation :

Calibration Nous projetons de comparer les résultats de simulation avec des résultats d'expériences *in vivo* connus afin d'étalonner au mieux les moteurs physique et chimique. Si cela s'avère nécessaire, nous changerons l'algorithme d'intégration numérique pour une version plus précise, au prix de la performance. Nous envisageons d'implémenter bien plus d'exemples de modèles qu'actuellement. Nous considérons dans ce but l'établissement d'une collaboration avec des scientifiques des sciences du vivant s'intéressant à la morphogénèse ou plus largement aux dynamiques d'une population de bactéries.

Performance La performance actuelle du simulateur OTB provient principalement d'une planification en amont de sa conception. Il est encore possible d'améliorer la vitesse d'exécution du simulateur. Pour cela, il sera nécessaire d'effectuer des mesures quantitatives afin d'identifier les parties du programme les plus coûteuses en temps de calcul et les optimiser au mieux, par rapport à l'environnement dans lequel le simulateur est utilisé et en fonction des ressources en espace disponible sur le périphérique de calcul. Il pourra ensuite être envisageable de comparer les performances de OTB par rapport à d'autres simulateurs de population de bactéries comme GRO [18].

Interface de sortie L'interface utilisateur de OTB est à ce jour rudimentaire. Pour qu'un plus grand nombre d'utilisateurs puissent utiliser le simulateur au quotidien il est impératif de lui ajouter quelques fonctionnalités : un meilleur contrôle à la souris, l'ajout d'obstacles et de flux dans l'environnement afin de tester l'influence de contraintes mécaniques sur une population en croissance ou encore présenter plusieurs vues en simultané de la même population. Il est envisageable d'ajouter d'autres modes de sortie, comme des mesures quantitatives agrégées sous forme de graphes en fonction du temps de la simulation comme, par exemple, l'orientation moyenne des bactéries, leur nombre, leur taille moyenne, leur volume moyen, etc.

Interface d'entrée La description d'une simulation par un fichier SBGP peut encore être améliorée. Il est d'abord nécessaire d'optimiser le contenu de ce fichier en fonction des informations réellement nécessaire à une simulation OTB ce qui sera possible une fois le point « Calibration » effectué et après discussion avec les futurs utilisateurs d'OTB. Il est aussi envisageable de contrôler OTB à partir d'un programme externe, par exemple une interface graphique regroupant les interfaces d'entrée et de sortie pour un usage plus intuitif.

Formes de bactérie De part sa conception, OTB est modulaire et extensible. Une piste de travail futur est l'exploration de la simulation d'autre formes de bactéries du genre bacillus (comme *E. Coli*), diplobacillus (des couples de bâtonnets), streptobacillus (des chaînes de bacillus), coccus (des sphères), coccobacillus (des ovales), etc. Changer de forme de bactérie implique de mettre à jour l'algorithme de collision qui a été développé spécifiquement pour les bactéries bacillaires. Ces développements se feront également en lien avec le point « Calibration » en fonction des besoins.

Grille hexagonale Une propriété des grilles hexagonales est que toutes les cellules ont un bord en commun et leur centres sont équidistants (ce qui n'est pas le cas sur une grille carrée, les centres des cellules en diagonale sont plus éloignés). Nous souhaitons explorer le fonctionnement de PPM sur ce support afin de déterminer s'il est possible d'obtenir une diminution des erreurs de la méthode d'intégration actuelle du moteur chimique et du calcul de collision du moteur physique.

MGS Dans le point « Interface d'entrée » nous avons indiqué qu'il était envisageable qu'un programme externe serve d'interface de contrôle pour OTB. MGS est un bon candidat pour cette tâche. En effet, nous pouvons considérer que OTB définit une collection topologique dont les éléments sont les bactéries. Cette collection est paramétrée par les

réactions entre les morphogènes. Chaque élément de cette collection a pour valeur la spécification d'un réseau de régulation génétique sous la forme d'un automate avec nombre fini d'états.

Références

[1] Howard C Berg, Douglas A Brown et al. "Chemotaxis in Escherichia coli analysed by three-dimensional tracking". In : *Nature* 239.5374 (1972), p. 500–504.

[2] Frederick R Blattner, Guy Plunkett, Craig A Bloch, Nicole T Perna, Valerie Burland, Monica Riley, Julio Collado-Vides, Jeremy D Glasner, Christopher K Rode, George F Mayhew et al. "The complete genome sequence of Escherichia coli K-12". In : *Science* 277.5331 (1997), p. 1453–1462.

[3] Mathias Bourgoin, Emmanuel Chailloux et Jean-Luc Lamotte. "SPOC: GPGPU programming through stream processing with OCaml". In : *Parallel Processing Letters* 22.02 (2012), p. 1240007.

[4] Dennis Bray, Matthew D Levin et Karen Lipkow. "The chemotactic behavior of computer-based surrogate bacteria". In : *Current biology* 17.1 (2007), p. 12–19.

[5] Tim Bray. "The javascript object notation (json) data interchange format". In : (2014).

[6] Hans Bremer et Patrick P. Dennis. "Modulation of chemical composition and other parameters of the cell by growth rate". In : (1996).

[7] Nicholas C. Darnton, Linda Turner, Svetlana Rojevsky et Howard C. Berg. "On torque and tumbling in swimming Escherichia coli". In : *Journal of bacteriology* 189.5 (2007), p. 1756–1764.

[8] Rui Dilão et Joaquim Sainhas. "Validation and calibration of models for reaction–diffusion systems". In : *International Journal of Bifurcation and Chaos* 8.06 (1998), p. 1163–1182.

[9] Willow R. DiLuzio, Linda Turner, Michael Mayer, Piotr Garstecki, Douglas B. Weibel, Howard C. Berg et George M. Whitesides. "Escherichia coli swim on the right-hand side". In : *Nature* 435.7046 (2005), p. 1271–1274.

[10] Tingxing Dong, Veselin Dobrev, Tzanio Kolev, Robert Rieben, Stanimire Tomov et Jack Dongarra. "A step towards energy efficient computing : Redesigning a hydrodynamic application on CPU-GPU". In : *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*. IEEE, 2014, p. 972–981.

[11] Britanica Online Encyclopedia. *Bacteria article*. 2016.

[12] M. J. Flynn. "Some Computer Organizations and Their Effectiveness". In : *IEEE Transactions on Computers* C-21.9 (sept. 1972), p. 948–960.

[13] Edward Fredkin et Tommaso Toffoli. "Conservative logic". In : *International Journal of Theoretical Physics* 21.3 (1982), p. 219–253.

[14] David V Goeddel, Dennis G Kleid, Francisco Bolivar, Herbert L Heyneker, Daniel G Yansura, Roberto Crea, Tadaaki Hirose, Adam Kraszewski, Keiichi Itakura et Arthur D Riggs. "Expression in Escherichia coli of chemically synthesized genes for human insulin". In : *Proceedings of the National Academy of Sciences* 76.1 (1979), p. 106–110.

[15] Oskar Hallatschek, Pascal Hersen, Sharad Ramanathan et David R Nelson. “Genetic drift at expanding frontiers promotes gene segregation”. In : *Proceedings of the National Academy of Sciences* 104.50 (2007), p. 19926–19930.

[16] Mee-Jung Han et Sang Yup Lee. “The Escherichia coli Proteome: Past, Present, and Future Prospects”. eng. In : *Microbiology and Molecular Biology Reviews* 70.2 (juin 2006), p. 362–439.

[17] J Hardy, Yv Pomeau et O De Pazzis. “Time evolution of a two-dimensional classical lattice system”. In : *Physical Review Letters* 31.5 (1973), p. 276.

[18] Seunghee S Jang, Kevin T Oishi, Robert G Egbert et Eric Klavins. “Specification and simulation of synthetic multicelled behaviors”. In : *ACS synthetic biology* 1.8 (2012), p. 365–374.

[19] Jarkko Kari. “Reversibility and surjectivity problems of cellular automata”. In : *Journal of Computer and System Sciences* 48.1 (1994), p. 149–182.

[20] Andrei N Kolmogorov, IG Petrovsky et NS Piskunov. “Etude de l’équation de la diffusion avec croissance de la quantité de matière et son application à un problème biologique”. In : *Moscow Univ. Math. Bull* 1.1-25 (1937), p. 129.

[21] HE Kubitschek. “Cell volume increase in Escherichia coli after shifts to richer media.” In : *Journal of bacteriology* 172.1 (1990), p. 94–101.

[22] Norman Margolus. “Physics-like models of computation”. In : *Physica D : Nonlinear Phenomena* 10.1 (1984), p. 81–95.

[23] Yu Medvedev. “Multi-particle cellular-automata models for diffusion simulation”. In : *Methods and tools of parallel programming multicomputers*. Springer, 2010, p. 204–211.

[24] Bernardo A. Mello et Yuhai Tu. “Quantitative modeling of sensitivity in bacterial chemotaxis : The role of coupling among different chemoreceptor species”. en. In : *Proceedings of the National Academy of Sciences* 100.14 (juil. 2003), p. 8223–8228.

[25] O. Michel. “Représentations dynamiques de l’espace dans un langage déclaratif de simulation”. Thèse de doct. Université de Paris-Sud, centre d’Orsay, déc. 1996.

[26] John Von Neumann. *Theory of Self-Reproducing Automata*. Sous la dir. d’Arthur W. Burks. Champaign, IL, USA : University of Illinois Press, 1966.

[27] Jonathan Pascalie, Martin Potier, Taras Kowaliw, Jean-Louis Giavitto, Olivier Michel, Antoine Spicher et René Doursat. “Developmental Design of Synthetic Bacterial Architectures by Morphogenetic Engineering”. In : *ACS Synthetic Biology* 5.8 (2016), p. 842–861.

[28] Thomas S. Shimizu, Sergej V. Aksenov et Dennis Bray. “A Spatially Extended Stochastic Model of the Bacterial Chemotaxis Signalling Pathway”. en. In : *Journal of Molecular Biology* 329.2 (mai 2003), p. 291–309.

[29] Kirsten Skarstad, Harold B Steen et Erik Boye. “Cell cycle parameters of slowly growing Escherichia coli B/r studied by flow cytometry.” In : *Journal of Bacteriology* 154.2 (1983), p. 656–662.

[30] Victor Sourjik. “Receptor clustering and signal processing in E. coli chemotaxis”. In : *Trends in microbiology* 12.12 (2004), p. 569–576.

[31] Eric J Stewart, Richard Madden, Gregory Paul et François Taddei. “Aging and Death in an Organism That Reproduces by Morphologically Symmetric Division”. In : *PLoS Biol* 3.2 (fév. 2005), e45.

[32] Tommaso Toffoli et Norman Margolus. *Cellular automata machines : a new environment for modeling*. Cambridge : MIT press, 1987.

[33] Frank J Trueba et Conrad L Woldringh. “Changes in cell diameter during the division cycle of Escherichia coli.” In : *Journal of bacteriology* 142.3 (1980), p. 869–878.

[34] A. M. Turing. “The Chemical Basis of Morphogenesis”. In : *Philosophical Transactions of the Royal Society of London B : Biological Sciences* 237.641 (1952), p. 37–72.

[35] Linda Turner, William S. Ryu et Howard C. Berg. “Real-Time Imaging of Fluorescent Flagellar Filaments”. en. In : *Journal of Bacteriology* 182.10 (mai 2000), p. 2793–2801.

[36] A Zaritsky, NB Grover, J Naaman, CL Woldringh et RF Rosenberger. “Growth and form in bacteria”. In : *Comments Mol. Cell. Biophys* 1 (1982), p. 237–260.

Chapitre 5

Conclusion

Dans ce très court chapitre, nous résumons les chapitres précédents et présentons nos contributions au projet de recherche ANR Synbiotic.

5.1 Résumé de nos travaux

Modèles et modélisation multi-niveau Dans ce chapitre, nous présentons la définition d'un formalisme unifiant dans le but de permettre la spécification commune et la classification de modèles. En partant du constat que modéliser revenait à définir une loi d'exclusion, c'est à dire distinguer les comportements acceptés des comportements rejetés par le modèle, nous avons donné une définition d'un modèle comme un couple $(\Sigma, \mathfrak{B})$ constitué de la signature du modèle Σ et de son comportement $\mathfrak{B}$. Un modèle peut être construit suivant différentes méthodes : par extension, dans le cas d'un modèle expérimental par exemple, ou bien par intention. Après avoir fourni quelques exemples de reformulation de modèles classiques dans notre formalisme, nous nous sommes attaqués à l'expression des relations pouvant exister entre différents modèles. Il est apparu nécessaire de considérer un modèle de référence pour définir une flèche d'abstraction entre deux modèles : le modèle de référence se présente comme le lien au système étudié, et peut être un modèle expérimental. Nous avons également étudié l'expression formelle d'une composition entre deux modèles. Ces définitions nous ont permis d'établir une construction cohérente constituée de plusieurs modèles en relation les uns avec les autres fournissant aussi une première partie concrète de réponse au problème de la construction de modèles constitués de sous-modèles en relation les uns avec les autres.

L'activité dans la programmation spatiale Dans ce chapitre, nous abordons une nouvelle technique de modélisation fondée sur l'activité spatiale dans le cadre d'un langage informatique pour la modélisation et la simulation spatiale nommé MGS. MGS est un langage de programmation non-conventionnel qui met l'accent sur la place centrale de l'espace dans la modélisation. Spécifier un modèle avec MGS revient à choisir une structure de donnée adaptée, appelée collection topologique, et une fonction de transition, la transformation. Une simulation

d'un modèle écrit avec MGS correspond à la réécriture successive de la collection topologique choisie. Grâce à MGS, nous avons introduit et mis en situation un nouvel algorithme de calcul d'un front d'activité : premièrement cet algorithme généralise des optimisations déjà connues sur les automates cellulaires à toute collection topologique, deuxièmement, notre algorithme rend accessible comme donnée de premier ordre les zones spatialement actives pendant la simulation d'un modèle écrit avec MGS.

Travelling Bacteria Dans ce chapitre, nous présentons OTB, un simulateur du comportement d'une population de bactéries *Escherichia Coli* écrit en OCaml et en C. Ce simulateur repose sur un modèle du comportement des bactéries *E. Coli* dans leur environnement qui est le fruit de la composition de trois modèles :

- le *moteur physique*, dédié au comportement physique des bactéries ;
- le *moteur chimique*, dédié à la réaction et à la diffusion des morphogènes dans l'environnement des bactéries, et
- le *moteur de décision*, dédié à l'interaction de ces deux environnements, décrivant les interactions mutuelles entre les deux moteurs précédents, il régit le comportement de chaque bactérie et fait appel aux deux modèles précédents.

Les deux moteurs physiques et chimiques reposent sur les automates cellulaires en deux dimensions pour lesquels nous avons développé une technique de simulation originale, l'algorithme de Propagation Parallèle à la Margolus (PPM), dans le but de pouvoir simuler une population de l'ordre de 10^5 individus sur des ordinateurs munis de cartes graphique grand public.

5.2 Contribution à Synbiotic

Ce manuscrit de thèse est un livrable du projet ANR Synbiotic. Dans ce document nous avons contribué spécifiquement à deux work-packages, WP2 (chapitre 3) et WP3 (chapitre 4), ainsi qu'au projet de recherche en général (chapitre 2) :

Contributions à WP2 Ce WP est dédié à l'élaboration d'un langage, nommé L1, de programmation spatiale dont la fonction est de décrire les exemples déterminés dans le WP1. Le langage MGS, introduit dans le chapitre 3, est un candidat idéal pour ce rôle. Tout d'abord, MGS est construit sur les collections topologiques, une structure de donnée dans laquelle l'espace est traité explicitement. Il dispose d'une primitive spatiale s'appliquant à toute collection topologique : l'opérateur de voisinage « , ». Cet unique opérateur (et ses restrictions) permet de spécifier, indépendamment de la collection, le modèle de chaque exemple du WP1. Ensuite, dans MGS, le temps est modélisé par les altérations successives d'une collection topologique au moyen d'une fonction définie par cas sur des motifs de filtrage d'une collection topologique appelée transformation. Cette fonction permet de traiter différents aspects temporels (asynchrone, synchrone, déterministe, stochastique, etc.) de l'évolution et se trouve particulièrement bien adapté à le description des exemples du WP1. Finalement, la mise en évidence de l'activité spatiale dans

MGS nous permet de simuler les exemples, dans certain cas, plus efficacement en restreignant le filtrage de motif à la partie active de la collection topologique, c'est à dire celle qui se trouvera modifiée au prochain pas de simulation.

Contributions à WP3 Ce WP est dédié à l'élaboration d'un second langage, nommé L2, de programmation prenant en entrée les concepts utilisés en sortie de L1, et en les instanciant dans les individus (des bactéries). Le simulateur OTB, muni du langage de description SBGP, introduit dans le chapitre 4, est un candidat idéal pour ce rôle. Tout d'abord, le langage de description permettant d'instancier le comportement des bactéries est SBGP. Il décrit l'équivalent d'un Réseau de Régulation Génétique (RRG) d'une bactérie, c'est à dire l'action de l'environnement sur l'expression de son code génétique. Dans OTB, ce RRG est décrit sous la forme d'un automate à états finis embarqué dans chacune des bactéries, ce qui permet de retrouver des comportements de population observés en laboratoire. Ensuite, nous avons fait en sorte que OTB soit le plus efficace possible, en utilisant le parallélisme à disposition dans les cartes graphiques grand public. Nous pouvons atteindre en quelques minutes une population de l'ordre des 10^5 bactéries interagissant les unes avec les autres. Finalement, et même si cet objectif n'est pas encore atteint au moment de l'écriture de ce manuscrit, nous estimons que OTB peut-être vu comme une collection topologique particulière de MGS, ce qui nous permettra de lier les langages L1 et L2 directement au niveau de MGS. Ainsi, les primitives spatiales définies dans L1 pourront avoir une traduction explicite directement dans MGS et servir à définir le RRG incorporé dans les bactéries de OTB.

Contributions générales Le projet Synbiotic a pour objectif de décrire le passage d'un comportement de haut niveau, à l'échelle d'une population d'entités, à un comportement de bas niveau, à l'échelle de l'individu, par le passage le long d'une tour de langages, similaire au processus de compilation d'un programme informatique. Pour commencer, nous constatons que chacun de ces étages de cette tour de compilation peuvent être vu comme des niveau de modélisation. À chaque étage, un langage de programmation manipule des objets d'une certaine manière, avec une certaine préférence, par exemple, dans MGS, l'accent est mis sur la description explicite de l'espace. Dans le cadre du chapitre 2, nous pourrions voir cet étage comme un niveau de description où les modèles appartiennent tous à la classe des modèles à champ, en y ajoutant certainement quelques contraintes pour mieux coller à MGS, les champs offrent une description plus abstraite que les opérateurs de MGS nous permettent. Ensuite, le cadre formel développé dans le chapitre 2, nous donne un outil concret pour aller caractériser les classes de modèles spécifiques à chaque étages et à expliciter les liens entre chacun des étages de cette tour de langages. Nous sommes désormais en mesure, dans des travaux futurs, de déterminer quel type de flèche il existe entre chacun des niveaux de description du projet. De plus, la description de l'activité spatiale dans le chapitre 3 nous donne déjà une voie vers une représentation intermédiaire entre L0, le langage de haut niveau pour la description des exemples du WP1 et L1. Finalement, les travaux effectués chapitre 2 devraient apporter un éclairage bienvenu sur les travaux des autres WP, en proposant un support à l'étude de la calculabilité des modèles de chacun des niveaux pour le WP5, et un cadre de des-

cription général permettant d'étudier au mieux le respect des spécifications dans le cadre du WP6.

Nos perspectives de recherche globales sont bien sûr plus nombreuses que la somme des perspectives de recherche de chacune des parties. Est-ce bien le cas ? Nous le saurons bien assez tôt.

Résumé

La description et la compréhension d'un système passe souvent par la construction d'un modèle mathématique. Ce dernier constitue un point de vue particulier sur le système (structurel, dynamique, etc.). Constituer des modèles plus complets, c'est-à-dire multi-point-de-vue, atteint rapidement les limites des formalismes qui les supportent. Une solution alternative passe par le couplage de plusieurs modèles « simples ». Dans le cas où chaque modèle correspond à un niveau de description du système, comme le niveau de la molécule, le niveau de la cellule, le niveau de l'organe, pour un système biologique, nous parlerons de modélisation multi-niveau. Ces niveaux sont organisés et interagissent. Nous pensons que la modélisation multi-niveau ouvre une voie prometteuse pour l'étude des systèmes complexes, traditionnellement durs à modéliser. Nous explorons trois voies pour la compréhension du fonctionnement de ces modèles en nous restreignant à la question de la relation entre global et local, c'est à dire entre l'individu et la population. La première voie est formelle et passe par la définition mathématique de « modèle » indépendamment du formalisme qui le supporte, par la présentation des différents types de modèles que l'on peut construire et par la définition explicite des relations qu'ils entretiennent. La seconde voie est portée par l'activité, définie dans le cadre de MGS, un langage de programmation spatiale, dont le modèle de calcul est fondé sur la réécriture des collections topologiques au moyen de transformations. Nous fournissons une méthode constructive pour l'obtention d'une description de plus haut niveau (une abstraction) des systèmes étudiés en déterminant automatiquement quelle est la sous-collection active sans la nécessité de faire référence à la sous-collection quiescente. La dernière voie est pratique, elle passe par la programmation de OTB, un outil de simulation parallèle pour l'étude de la morphogénèse dans une population de bactéries *E. Coli*. Pour OTB, nous avons conçu un algorithme générique de calcul parallèle d'un automate cellulaire en deux dimensions, adapté aux cartes graphiques grand public. Le modèle embarqué dans OTB correspond au couplage de trois modèles correspondant chacun à un niveau de description du système : le modèle physique, qui décrit la dynamique des collisions entre bactéries, le modèle chimique, qui décrit la réaction et la diffusion des morphogènes, et le modèle de prise de décision, qui décrit l'interaction entre les bactéries et leur support.

Abstract

We often build mathematical models to describe and understand what a system does. Each model gives a specific point of view on the system (structure, dynamics, etc.). Building more comprehensive models that encompass many different points of view is limited by the formalism they are written in. Coupling "simple" models to form a bigger one is an alternative. If each model corresponds to a level of description of the system, e.g., the molecular level, the cellular level, the organ level in biology, then we call this technique multi-level modelling. Levels of description are organized and interact with each other. We think that multi-level modelling is a promising technique to model complex systems, which are known to be difficult to model. We have opened three distinct research tracks to investigate the link between local and global properties, for instance between those of an entity and its population — a classical opposition in complex systems. On the first track, we give precise definitions of a model — independently of its underlying formalism, of a system and of some of the relations models have (validation, abstraction, composition). We also introduce different classes of models and show how they relate to some classical definitions (dynamic models, spatial models, etc.) On the second track, we look at MGS, a spatial programming language based on the rewriting of topological collections by means of transformation functions. We present a constructive method giving us access to a higher level of description of the system (an abstraction). This method automatically computes the active sub-collection of a model, without any knowledge about the quiescent sub-collection, and follows it for each time step. Finally, on the third track, we present OTB, a parallel simulator for the study of morphogenesis in a population of *E. Coli* bacteria. We provide a generic algorithm for the parallel simulation of two-dimensional cellular automata on general-purpose graphics cards. OTB itself is built around a multi-level model for the population of bacteria. This model is the result of the coupling of three "simple" (base) models: a physical model, describing how bacteria collide, a chemical model, describing how morphogenes react and diffuse, and a decision model, describing how bacteria and their environment interact.