Discrétisation automatique de machines à signaux en automates cellulaires

Tom Besson

Centre Val de Loire UNIVERSITÉ Spirit of the Loire Valley

UNIVERSITÉ D'ORLÉANS

UNIVERSITE D'ORLEANS

ÉCOLE DOCTORALE MATHÉMATIQUES, INFORMATIQUE, PHYSIQUE THÉORIQUE ET INGÉNIERIE DES SYSTÈMES

Laboratoire d'Informatique Fondamentale d'Orléans

Thèse présentée par :

Tom BESSON

soutenue le : **10 avril 2018**

pour obtenir le grade de : **Docteur de l'Université d'Orléans**

Discipline/ Spécialité : **Informatique**

Discrétisation automatique de machines à signaux en automates cellulaires

THÈSE DIRIGÉE PAR :

Jérôme DURAND-LOSE	Professeur des Universités, Université d'Orléans

RAPPORTEURS :

Olivier BOURNEZ	Professeur, École polytechnique
Christine EISENBEIS	Directrice de recherche, INRIA

JURY :

Olivier BOURNEZ	Professeur, École polytechnique, président du jury
Christine EISENBEIS	Directrice de recherche, INRIA, rapporteure
Sama GOLIAEI	Assistant Professor, University of Tehran, examinatrice
Jean-Baptiste YUNÈS	Maître de Conférence Habilité, Université Paris Diderot, examinateur
Thierry MONTEIL	Chargé de Recherche CNRS, CNRS/Paris 13, examinateur
Jérôme DURAND-LOSE	Professeur des Universités, Université d'Orléans, directeur de thèse

Remerciements

Exegi monumentum aere perennius, en toute modestie. Mais ce « monument » n'aurait pas pu être achevé sans l'aide et le soutien de nombreuses personnes. Je me dois donc de les remercier ici, en espérant ne pas en oublier, mais sachant pertinemment que ce sera le cas (désolé en avance, promis je penserai à vous si je fais une autre thèse...).

Je souhaite donc remercier tout d'abord les membres de mon jury de thèse : Olivier Bournez qui a accepté de présider ma soutenance et pour son rapport, Christine Einsenbeis, pour son rapport elle aussi, puis finalement Sama Goliaiei, Thierry Monteil et Jean-Baptiste Yunès pour avoir été examinateurs du jury.

Celui qui a le plus motivé cette thèse (en en étant à la source), qui m'a encadré pendant toute sa durée, m'a encouragé, aidé à y voir plus clair et convaincu de m'accrocher dans les moments les plus difficiles mérite bien entendu une reconnaissance toute particulière. Je souhaite donc exprimer ma plus profonde reconnaissance à mon directeur, Jérôme Durand-Lose, sans qui, bien évidemment, cette thèse n'aurait pas été possible.

Ensuite, une thèse c'est aussi un laboratoire, et je souhaite donc remercier tous les membres du LIFO pour m'avoir accueilli et intégré rapidement. Le LIFO est un lieu particulièrement agréable pour la recherche et je suis reconnaissant d'avoir pu y préparer ma thèse. Je tiens d'ailleurs à remercier en particulier Isabelle Renard, Florence Maubert et Brigitte Dupuy qui ont réussi avec brio à faire face à mon inaptitude administrative et à m'aider à naviguer dans les difficultés de la bureaucratie. J'aurais eu bien du mal à faire quoi que ce soit si elles n'avaient pas été aussi efficaces.

Un laboratoire c'est aussi des équipes et je tiens donc à remercier particulièrement les membres de GAMoc pour m'avoir ouvert à d'autres sujets de recherche que le mien et fournit des collègues de premier plan. Merci donc à Nicolas Ollinger, Ioan Todinca, Mathieu Liedloff, Florent Becker, Martin Delacourt et Anthony Perez.

Les doctorants du LIFO ont aussi été à l'origine du cadre agréable du laboratoire (et aussi d'en dehors), et pour ça, ils méritent que je les remercient ici. Merci donc à eux tous pour avoir été présents pour tout un tas de choses, à commencer par de longues discussions dont la pertinence n'était pas toujours évidente mais l'intérêt primordial (sur le moment). Pêle-mêle, et sans les noms pour économiser de la place, je remercie : Kevin, Diego, Vivien, Xavier, Gaëtan, Benjamin, Valentin, Sabrina, Abdel, Romain, Gauthier, Pedro, Victor et si j'en oublie, je suis désolé.

Enfin, impossible de mener à bien un tel projet sans un cercle familial et amical de premier plan. Je remercie donc mes amis de longue date : Vincent, Benjamin, Lilian, Jessy pour m'avoir encouragé chacun à votre manière. Et pour finir, ma famille au sens large mais surtout mes parents, mon frère et ma sœur : vous m'avez supporté (en devant expliquer à tout le monde qu'à mon âge j'étais encore étudiant) tout ce temps et je doute que j'aurais pu finir cette thèse si vous n'aviez pas été aussi brillants pour me motiver à continuer.

Sommaire

Liste des figures

Introduction

État de l'art

Depuis longtemps, la philosophie, les mathématiques et désormais l'informatique tentent de définir l'action de calculer. Si la notion n'a jamais vraiment été clairement définie, les siècles ont vu passer de nombreuses méthodes pour coder et traiter automatiquement (par des procédés algorithmiques et pas forcément par une machine) des informations, qu'elles soient numériques, géométriques ou d'autres natures. Pour l'informaticien, le point d'intérêt majeur de ces recherches est l'invention des machines de Turing par Alan Turing en 1936 [Tur36], qui peut être considérée comme la naissance de l'informatique. La thèse de Church-Turing [Chu36] affirme que celles-ci capturent la notion de calculabilité, dans le sens où tout ce qu'une machine de Turing peut calculer est alors du domaine du calculable, à l'opposé de ce qu'elles ne peuvent faire, qui fait donc partie du domaine de l'incalculable.

Dès lors, les *modèles de calculs* (des modèles mathématiques permettant de calculer) se multiplièrent pour tenter d'étendre cette notion de calculabilité, en cherchant à exprimer le calculable différemment ou en créant des modèles dont le potentiel calculatoire est plus faible mais spécialisé. Ce sont des modèles comme les automates finis [Law04, MP43] et les automates à pile [GGH67]. La recherche se porte aussi sur des modèles exploitant des phénomènes physiques : l'optique [WN05], la mécanique quantique [Fey86], la gravité (avec des dominos) [Ste11]. D'autres encore s'inspirent du vivant : les réseaux de neurone [SS95], l'ADN [Amo97], le comportement des cellules [PR02].

Les modèles de calcul précédents n'exploitent le plus souvent qu'une abstraction des phénomènes réels qui les ont inspirés, en ignorant leurs limitations naturelles. Il devient alors nécessaire de représenter leurs fonctionnements par des concepts abstraits, des expressions arithmétiques formelles ou des constructions géométriques. C'est à ce deuxième type d'abstraction que cette thèse s'intéresse : des modèles qui s'appuient uniquement sur des constructions géométriques pour calculer. Souvent, leurs espaces d'application sont discrets : le domaine spatial et/ou temporel des interactions produisant le calcul est celui des entiers. C'est le cas du modèle de calcul par boules de billard [DL02], des tuiles auto-assemblantes [Win98], ou des très connus et étudiés automates cellulaires [vN66] (tous trois trouvent d'ailleurs leurs origines dans la physique ou la biologie). Ces modèles, bien que fondamentalement discrets, sont souvent représentés par des dessins continus : des droites, demi-droites, segments ou points dans un espace continu, qui se « déplacent », illustrant ainsi de manière naturelle les dynamiques qu'ils décrivent. Ces représentations continues ont été abstraites en des modèles dans des univers continus, avec notamment les évolutions « d'espaces colorés » de G. Jacopini et G. Sontacchi [JS90], qui proposent

une étude fondamentale du comportement d'un tel modèle.

En parallèle de ces modèles de calculs Turing-complet, la notion de calculabilité est étendue. Quelle est exactement cette notion que les machines de Turing et les nombreux autres modèles affirment capturer ? Cette question n'a pas vraiment de réponse universelle, mais la recherche dans ce domaine a vu naître d'autres questions, liées elles aussi à la calculabilité. L'une d'entre elles est de savoir ce que pourrait calculer une machine qui serai *super-Turing*, c'est-à-dire capable de résoudre des problèmes insolubles pour les machines de Turing. De ce questionnement naissent à nouveau de nombreux autres modèles possédant une ou plusieurs propriétés qui leur permet d'atteindre cette puissance. C'est le cas du modèle de Blum-Shub et Smale (BSS) qui propose de manipuler exactement les réels [BSS89], des machines de Turing à temps infini qui effectuent un nombre infini d'étapes de calcul [HL00] ou même simplement du modèle pensé par Turing que sont les machines de Turing à oracle, qui ont la possibilité de résoudre un problème particulier en en demandant le résultat à une autre machine, appelée oracle, qui résoudra toujours exactement ce problème (même si celui-ci est incalculable) [Tur39]. Il est important de garder à l'esprit que bien que ces modèles sont intéressants d'un point de vue théorique, leur utilisation dans notre monde physique est à l'heure actuelle impossible. Les spécificités qu'ils utilisent pour résoudre « l'incalculable » sont en effet toujours peu compatibles avec les lois de la physique tels que nous les connaissons aujourd'hui.

L'étude d'un de ces modèles de calculs géométriques, les automates cellulaires, a donné naissance aux machines à signaux, un nouveau modèle super-Turing, proposé par Jérôme Durand-Lose [DL03]. Basées sur les notions de signal et collision dans les automates cellulaires, ces machines en sont une abstraction continue, capable de manipuler les réels. Ce modèle de calcul géométrique continu a montré posséder un pouvoir de calcul supérieur aux machines de Turing [DL12], et donc la capacité de les simuler, de même que les automates cellulaires dont elles sont issues [DL11]. Par conséquent, et en tant qu'abstraction continue, un des intérêts du modèle est de permettre l'expression simple d'une dynamique discrète d'un automate cellulaire via une représentation continue. Il est donc souhaitable de pouvoir revenir automatiquement aux automates cellulaires à partir des machines à signaux, afin d'utiliser cette création continue pour concevoir facilement des automates cellulaires. Cette thèse propose donc de décrire plusieurs méthodes, dans différents cadres, permettant d'accomplir un tel objectif. Ce procédé est appelé *discrétisation*, l'idée étant que le modèle continu des machines à signaux est transformé en un modèle discret.

La discrétisation est un procédé qui apparaît régulièrement dans de nombreux cadres et est systématiquement problématique (des exemples de ces problèmes sont donnés par la suite). Très souvent, les représentations continues d'un phénomène sont simples à comprendre et permettent une modélisation aisée. Cependant, la « réalité » du phénomène est généralement plus complexe. L'expression continue n'est alors qu'une modélisation, une abstraction de la réalité facile à manipuler mais souvent inexacte dans la pratique (pas obligatoirement fausse mais généralement imprécise). Ce problème est le plus souvent résolu par une construction *ad hoc* : pour chacun des cas intéressants du modèle, une méthode spécifique de discrétisation est proposée, méthode qui ne fonctionnera pas sur n'importe quel autre cas.

Par exemple, la transformation d'une figure géométrique quelconque en un ensemble fini de pixels est une discrétisation et pose déjà des problèmes. La simple conversion d'une droite continue en droite discrète nécessite de faire des choix : la droite discrète est elle

4-connectée ou 8-connectée ? Quelle est sa période (quelle est la forme de la structure se répétant à l'infini) ? Quelle est « l'épaisseur » de chaque point ? Il y a encore bien d'autres questions auxquelles il faudrait répondre, et celles-ci ont bien souvent plusieurs solutions équivalentes en termes de correction de la discrétisation. Certains travaux s'y étant attachés montrent la non-trivialité d'une conversion ayant pourtant l'air simple. L'algorithme de Bresenham [Bre65], permettant de faire la discrétisation d'une ligne de ratio rationnel , en est un exemple. Pour les lignes de ratio irrationnels, la discrétisation est plus complexe : elle passe par l'utilisation de mots Sturmiens pour établir leur trajectoire [MH40].

Un autre exemple de discrétisation plus naturel est celui du passage des réels aux entiers. Comment exprimer π avec des entiers ? Arrondir, tronquer ? En conservant combien de décimales ? Encore une fois, beaucoup de questions et encore plus de solutions. De fait, π est un nombre irrationnel, impossible à exprimer exactement par des entiers seuls, nécessitant donc une formule permettant de le calculer. Une solution pour le discrétiser est donc d'écrire un nombre fini de ses chiffres en calculant, jusqu'à un certain point, ses décimales. La difficulté d'une telle expression provient des multiples façons de le faire : π est calculé depuis l'Antiquité, depuis la méthode d'Archimède jusqu'à des méthodes modernes utilisant toute la puissance de nos ordinateurs [BBB04]. Si ces calculs ne sont pas extrêmement complexes, ils ne fourniront cependant jamais la valeur exacte de π. Et c'est là la faiblesse de la discrétisation : peu importe le procédé, il existe toujours des cas comme celui de π, pour lesquels il n'existe pas d'équivalent discret exact à la valeur continue d'origine.

Avec toutes ces difficultés, la discrétisation *automatique* est bien souvent évitée : de fait, elle est complexe à mettre en œuvre, oblige à l'approximation et à l'erreur, et n'est pas nécessaire dans la plupart des cas. Cependant, il y a parfois des solutions qui sont proposées, comme dans [RR96] qui donne une idéalisation des équivalents discrets de certaines notions continues. Cependant, ces méthodes s'appliquent à un nombre limité de constructions continues et sont très complexes, nécessitant généralement de définir tout un nouveau pan de formalisation pour être utilisable. Dans cette thèse, ces outils ne sont pas utilisés car le cadre est nouveau (les machines à signaux), les quelques méthodes existantes n'y sont donc pas adaptées.

Pour les machines à signaux, discrétiser signifie convertir une machine en un automate cellulaire, et transformer les entrées de la machine en entrées de l'automate cellulaire ainsi obtenu. Pour bien comprendre les enjeux et les difficultés de ce type de discrétisation, il est nécessaire de présenter plus spécifiquement les deux modèles, ainsi que de développer ce que signifie discrétiser dans ce cadre.

Automates cellulaires

Depuis l'introduction des automates cellulaires (AC) par J. Von Neumann dans les années 1940 [vN66], de nombreuses recherches les concernant, depuis l'algorithmique jusqu'à leur pouvoir de calcul [Sar00], ont été poursuivies. Le fonctionnement du modèle est très simple : un ensemble d'automates disposés côte à côte sont dans un état particulier. Elles se mettent à jour toutes en même temps en consultant les états de leurs voisines, le tout dans un espace et un temps discret.

Au sein de ce modèle, et malgré une étude intensive, le problème de la création d'un

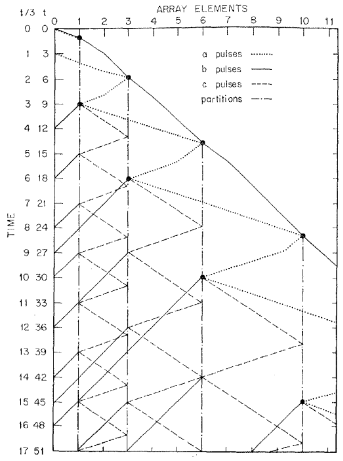
AC pour un objectif ou un comportement précis existe toujours et est récurrent. Des AC spécialisés sont fabriqués par des experts du domaine, de la même manière que programmer un ordinateur était autrefois réservé aux spécialistes. Désormais, coder devient de plus en plus simple, et il est naturel de chercher à savoir pourquoi cela n'est pas le cas des automates cellulaires. L'idée de la simplification, comme dans la programmation conventionnelle, peut passer par la conversion d'un modèle de plus haut niveau manipulant une construction similaire à une existante dans les AC. Dans ce cadre, il s'agit des signaux et collisions discrets observables dans les AC et formant la base du calcul par collision [Ada02]. Couramment, les dynamiques des AC dérivent (par fabrication ou par observation) des particules/signaux et des collisions entre eux. Il y a de nombreux exemples, dans la littérature des AC, de *conversions* implicites depuis une construction continue.

De fait, les signaux et collisions sont des composants fondamentaux des AC et en tant que tels, l'étude des comportements des AC basés sur les notions de particules/signaux et collisions a apporté beaucoup d'informations sur la structure des AC. Un des premiers usages de ce genre d'objets apparaît dans l'engendrement de nombres premiers tel qu'établi par Fischer dans [Fis65]. Les signaux sont couramment utilisés pour : résoudre le problème de synchronisation d'escouade (Firing Squad Synchronization Problem en anglais, ou FSSP) [Got66, Yun07], pour calculer avec la règle 110 [Coo04] ou avec seulement quatre états [OR11], pour comprendre un AC [CMD03]. . . Dans [Ric08] une question importante est le positionnement automatique de signaux discrets. Les signaux discrets peuvent aussi être étudiés pour eux mêmes comme dans [MT99] où Mazoyer et Terrier proposent des constructions pour des signaux avec des ratios spécifiques (qui peuvent être vus comme la vitesse à laquelle ils se déplacent dans le diagramme espace-temps discret).

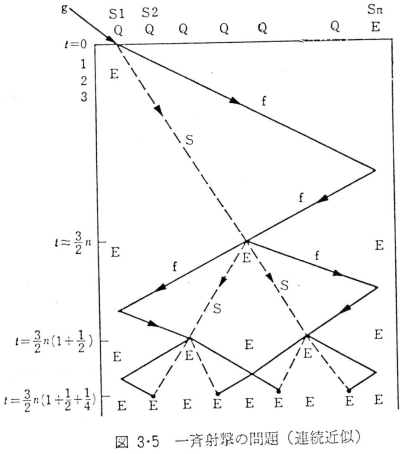
Ces constructions/présentations passent généralement par une abstraction continue, au lieu d'une construction discrète, où les signaux sont représentés comme des droites continues en interaction. Cela est particulièrement évident en observant l'engendrement de nombres premiers de Fischer (Fig. 1(a)) ou certaines solutions du FSSP de Goto et Yunès (Figures 1(b) et 1(c)). Ces constructions sont comprises dans des cadres continus avant d'être implantées dans un espace-temps discret. Les états et fonctions de transitions ne sont que rarement données car la correction de la construction est claire sur la construction par signaux ; de plus, ils peuvent être pénibles à établir, et peu intéressants à lire.

Machines à signaux

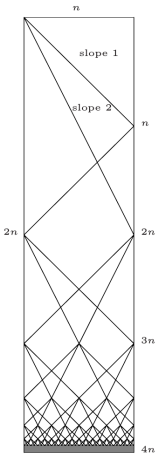
Le modèle des *machines à signaux* (MS) est une abstraction formalisant les représentations continues de signaux dans les AC. Il est l'outil formel pour penser ou concevoir avant de discrétiser mais peut aussi être étudié comme un modèle dynamique autonome. Dans les machines à signaux, les *signaux* sont des points sans dimension se déplaçant à une vitesse constante. Ils sont entièrement décrits par leur position et nature, appelée *méta-signal* et interagissent lorsqu'ils se croisent. Ce modèle est simple : un nombre fini de signaux se déplacent sur l'axe réel, et lorsque deux ou plus se rencontrent, ils sont détruits et de nouveaux signaux sont émis à leur place. Le résultat d'une collision de signaux est donné par un ensemble de *règles de collision*, précisant quels signaux sont émis selon ceux qui sont entrés en collision. La dynamique apparaît sur l'évolution temporelle du système, et peut être représentée comme un diagramme espace-temps où les signaux



(a) la solution de Fischer au problème d'engendrement de nombres premiers [Fis65, Fig. 2]



(b) la solution de Goto au FSSP [Got66, Fig. 3]



(c) une solution de Yunès au FSSP [Yun07, Fig. 1.a]

FIGURE 1 – Exemples de signaux utilisés dans la fabrication d'automates cellulaires.

sont dessinés comme des segments. Cette représentation est le calcul géométrique auquel s'intéresse cette thèse. Jérôme Durand-Lose s'est intéressé aux capacités du modèle, depuis son pouvoir de calcul [DL05, DL11] à sa capacité à simuler d'autres modèles (comme le modèle de Blum, Shub et Smale [BSS89] dans [DL07]).

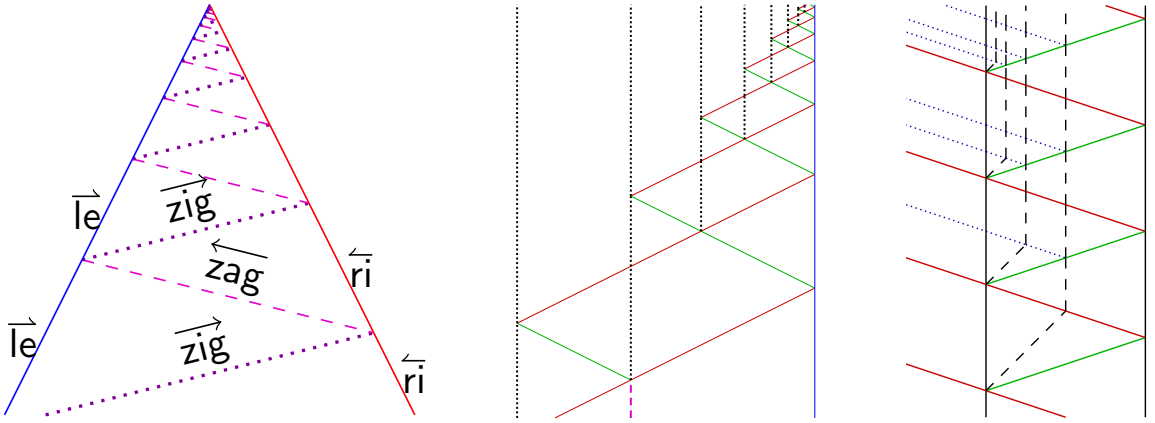
Dans cette thèse, les machines à signaux sont limitées aux rationnels (ce que cela implique est expliqué dans le Chapitre 1). La raison en est simple : le déplacement de signaux discrets de vitesses irrationnelles est un sujet de recherche à lui seul. Par conséquent, pour garantir que la discrétisation automatique soit au centre des recherches (et non pas la construction de signaux discrets à vitesse irrationnelle), le choix de se limiter aux rationnels à été fait. De plus, cette thèse souhaitant proposer un logiciel capable de discrétiser automatiquement une machine à signaux, se limiter à la manipulation de rationnels permet l'implantation des différentes techniques proposées.

Discrétisation

Dans cette thèse, l'objectif est de fournir une méthode capable de discrétiser une machine à signaux en un automate cellulaire. Ce travail se place dans un cadre quasi nouveau, puisque celui-ci n'a été abordé qu'à travers le modèle des machines à signaux discrètes via pré-topologie [LS10], dans lequel un premier pas vers la discrétisation a été fait à travers la conversion partielle des machines à signaux en un modèle discret situé entre les machines à signaux et les automates cellulaires.

Les machines à signaux ayant émergé de l'observation des automates cellulaires, les deux modèles sont donc fortement liés. De fait, d'après [DL11], il est possible de *simuler* un AC avec une MS. Cependant, comme mentionné précédemment, l'inverse n'est pas trivial et est réalisé uniquement pour des cas particuliers, en ignorant souvent les détails techniques. Le problème est le même que pour tout passage d'une construction continue à une discrète : traiter les approximations et faire face aux comportements impossibles

à reproduire. Ces difficultés sont si importantes qu'ils rendent de fait l'objectif de cette thèse irréalisable parce que théoriquement impossible (pour une discrétisation exacte). De la même manière qu'il est impossible d'exprimer exactement tout les réels avec des entiers, la conversion exacte de toutes les machines à signaux est impossible (notamment lorsqu'il faut faire face à des cas comme ceux de la Fig. 2).



(a) cas « d'accumulation » basique à 4 vitesses

(b) cas « d'accumulation » avec irrationalité

(c) densité de signaux illimitée à 4 vitesses

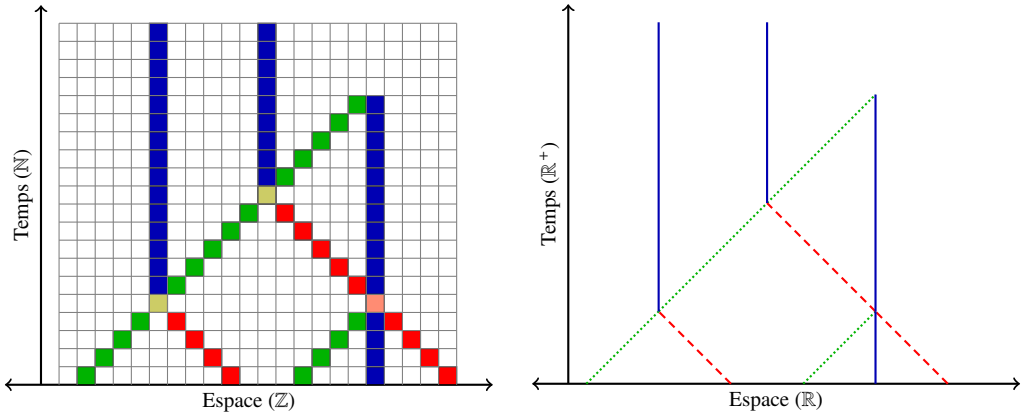
FIGURE 2 – Diagrammes espace-temps non discrétisables (exactement).

Cependant, des solutions peuvent être trouvées :

— il est possible de faire le choix de restreindre les machines à signaux à une sous-classe plus simple et par essence discrétisable,
— ou encore d'ignorer la possibilité d'un comportement purement continu et transformer directement les méta-signaux et règles de collisions en états et règles de transition, avec toutes les limitations que cela pose,
— une autre solution est de proposer une construction permettant d'exprimer des méthodes d'approximation de chacun des cas problématiques.

En plus de ces restrictions, l'objet de cette thèse est de proposer une méthode de discrétisation *automatique*. Il faut donc qu'elle puisse être accomplie par une machine physique existante, avec aussi peu d'influence humaine que possible. Il s'agit de proposer à un spécialiste des automates cellulaires de construire une machine à signaux avec ses règles et sa configuration initiale, de la fournir au système, et que celui-ci produise une identification comme celle de la Fig. 3 pour chaque machine à signaux (avec des approximations si il y a lieu), puis de l'exprimer sous la forme d'un automate cellulaire et de sa configuration initiale.

Les trois possibilités de discrétisation précédentes ont donc donné lieu à trois approches différentes, mais pas complètement dissociées, du problème. La première, l'approche par simplification, propose de s'intéresser uniquement à une sous-classe précise de machines à signaux qui, grâce à une propriété unique, peuvent être exactement discrétisées. La deuxième, l'approche brutale, discrétise en considérant la machine d'origine comme étant parfaitement discrétisable, impliquant donc qu'elle échoue sur toute *structure continue*. La troisième, l'approche par structuration du calcul, utilise une nouvelle méthode de construction des machines à signaux, définie dans cette thèse et intégralement dans le but



(a) une exécution d'un automate cellulaire (b) une exécution d'une machine à signaux

FIGURE 3 – Un exemple d'identification.

de discrétiser, pour fournir à un éventuel utilisateur la possibilité de décrire la discrétisation qu'il souhaite utiliser lors de l'engendrement automatique de l'AC (et donc de fournir une approximation).

Approche par simplification

Dans l'approche par simplification, la sous-classe de MS utilisée est celle des rationnelles à 3 vitesses : seulement trois vitesses différentes sont disponibles pour les méta-signaux, et les vitesses de même que les positions des signaux dans la configuration initiale sont restreintes aux nombres rationnels. Ces machines ne peuvent avoir de comportements extrêmement complexes : les signaux restent sur un réseau régulier et quasi-périodique de collisions appelé *maillage* [BCDL⁺13, DL13]. Ce maillage est essentiellement discret, ce qui permet la conversion vers un AC sans approximation ainsi que la conservation de la dynamique. Cette identification est faite par le biais de *l'ordre de causalité* des collisions (ou d'états qui s'y identifient). Cet ordre résulte en un diagramme dirigé acyclique labellisé (Labeled Directed Acyclic Graph en anglais ou LDAG). De plus, la position de toute collision dans un diagramme peut être linéairement calculée à partir de celle correspondante dans l'autre diagramme. L'existence de tels maillages n'est pas garanti avec quatre vitesses ou plus, ou l'irrationalité.

La construction démarre avec la *normalisation* de la machine à signaux, de sorte que la conversion soit plus simple et en particulier que les vitesses appartiennent à $[-1, 1]$ pour que les AC de rayon 1 soient utilisables. Puis les méta-signaux et les règles de collision sont transformés en états et transitions. La conversion des configurations suit la même méthode : normalisation de la configuration continue, puis création de la configuration discrète.

Cette transformation fonctionne grâce à la propriété de maillage. L'AC est conçu de sorte qu'avec le positionnement correct des états initiaux, un maillage discret correspondant est engendré. Une attention particulière est accordée à la discrétisation pour assurer que les éléments évoluent sur le maillage. La correction du processus provient de la « préservation » du maillage.

Cette conversion est automatique et a été implantée en Java à partir d'une librairie de machines à signaux existante. Son architecture est donnée dans l'Annexe A, de même que quelques diagrammes espace-temps engendrés et des données sur la taille de l'AC.

Approche brutale

Cette approche choisit d'ignorer la possibilité de comportements continus dans les machines à signaux, et peut s'utiliser sur toute machine rationnelle. De ce fait, elle fonctionne parfaitement sur les rationnelles à trois vitesses (qui ne peuvent pas contenir ce type de structure), mais est immédiatement mise en défaut par une construction purement continue. Deux méthodes différentes ont été utilisées pour implanter cette approche, la première ayant été abandonnée à cause de sa complexité technique.

La première méthode est l'engendrement des états et règles au cas par cas. Elle consiste en de nombreux calculs d'états possibles, de transitions possibles, et de toutes les combinaisons possibles. Une fois ces combinaisons calculées, elles sont mises sous forme de règle de transition par un processus *ad hoc*. Bien qu'attrayante, cette méthode demande d'énumérer les états, alors que ceux-ci sont en nombre infini. Par conséquent, son implantation technique, même dans un cadre réduit, est très complexe. Pour cette raison, elle a été abandonnée au profit de la méthode suivante et n'est pas décrite dans cette thèse.

La solution retenue est d'exécuter le comportement de la machine à chaque nouvelle transition inconnue. L'idée est de lancer la discrétisation sans avoir la totalité des états et règles de transition, et à chaque règle inconnue, extraire la configuration continue correspondante aux trois cellules d'entrées, de faire exécuter la machine à signaux à discrétiser sur cette configuration et de récupérer le résultat ainsi obtenu. Cette méthode offre l'avantage de ne pas avoir besoin d'énumérer toutes les possibilités, car elle ne s'intéresse qu'à ce qui se passe effectivement dans le diagramme espace-temps discret. Comme la méthode précédente, elle nécessite cependant de limiter le nombre d'états possibles (celui-ci étant toujours potentiellement infini), et ce faisant d'approximer parfois la position de certains signaux, mais elle est plus efficace et plus simple à implanter que la précédente. Cela dit, de par sa nature, elle échoue lorsqu'elle est utilisée sur une « structure continue ». C'est pour cela qu'elle est utilisée en association avec l'approche de discrétisation par structuration du calcul.

De nouveau cette discrétisation a été implantée en Java et son architecture est donnée dans l'Annexe B.

Approche par structuration du calcul

Dans l'approche par structuration du calcul, les machines manipulées restent rationnelles et le nombre de vitesses n'est pas limité. La rationalité de ces machines est une limite à la puissance de calcul des machines à signaux mais elles restent Turing complètes, et même super-Turing, avec des structurations comme celles de la Fig. 2 [DL03] (Partie 2). Cette approche est en deux étapes : d'abord établir une nouvelle méthode de description d'une machine à signaux, puis l'utiliser pour proposer des outils d'expression d'approximations à un utilisateur extérieur.

La nouvelle représentation d'une machine à signaux est appelée *modularité*. Son concept est simple : découper un diagramme espace-temps continu en zones (rectangulaires) conte-

nants au minimum une collision (et donc au minimum deux signaux). Ces zones, appelées modules, sont ensuite connectées par des signaux (entrants et sortants de modules). Cette vision permet d'abstraire le diagramme espace-temps, non plus en collision et signaux, mais en régions (ou blocs) de calculs, interagissants par le biais de signaux de communication. Ainsi, pour exprimer le diagramme espace-temps d'une machine à signaux sur une configuration initiale, il est possible d'utiliser la modularité et d'ignorer les collisions et certains signaux. En s'appuyant sur cette construction, une nouvelle expression des dynamiques d'une machine à signaux apparaît : la *dynamique modulaire*.

Muni de cette nouvelle vision, il devient possible de proposer une méthode d'expression d'approximations. La nature de chacun de ces modules est exprimée à travers des objets appelés *méta-modules*. Ces méta-modules sont en nombre fini pour chaque dynamique, et pour chacun d'entre eux, il est possible d'exprimer un équivalent discret. Ainsi, une dynamique modulaire *continue* est entièrement convertie en dynamique modulaire *discrète*, en conservant autant que possible l'ordre de causalité des modules. Dès lors, la dynamique modulaire discrète est transformée en un ensemble d'états et de règles de transitions d'automate cellulaire, en plus d'une configuration initiale correspondant à celle de la machine à signaux d'origine.

De la même manière que pour la discrétisation des machines rationnelles à trois vitesses, celle-ci a été implantée en Java à partir de la même librairie de machines à signaux. La structure de cette implantation est donnée dans l'Annexe C.

Cette méthode est aussi suffisamment souple pour pouvoir être simplement altérée ou enrichie. En effet, il existe des dynamiques de machines à signaux que la modularité donnée ici ne peut exprimer complètement. Par conséquent, il convient d'amender la méthode pour y intégrer des solutions *ad hoc* résolvant ces problèmes. Dans le cadre de cette thèse, trois de ces solutions sont proposées.

La première permet de décrire un diagramme espace-temps infini ne présentant pas de « structures continues ». Par une méthode d'expression appropriée, elle permet la description d'une infinité d'éléments par un nombre fini de modules. Elle est notamment utilisée pour permettre la discrétisation de la simulation d'une machine de Turing par une machine à signaux. La deuxième construction *ad hoc* s'applique à une structure fractale simple qui répète infiniment un seul module par itération. Il s'agit d'une construction récurrente dans les machines à signaux, qui est notamment utilisée pour exploiter le potentiel super-Turing du modèle, et qui est donc un problème majeur à la discrétisation. Il est en effet impossible de la discrétiser exactement mais la méthode de description proposée permet d'exprimer une approximation. La troisième et dernière solution donnée s'applique elle aussi à une structure fractale, mais qui cette fois double son nombre de modules à chaque itérations. Cette construction est emblématique de l'utilisation de signaux dans les automates cellulaires, puisqu'il s'agit d'une des solutions au FSSP (voir la section de cette introduction sur les automates cellulaires). Encore une fois, cette structure n'est pas exactement discrétisable, mais une description modulaire étendue permet de stopper le calcul au bon moment et ainsi d'obtenir une expression modulaire discrétisable, correspondant exactement à une solution du FSSP dans un automate cellulaire.

Implantations

Toutes les discrétisations proposées ici ont été implantées en Java (celle du dernier chapitre étant encore très grossière). Ces programmes se basent sur trois bibliothèques existantes, permettant de manipuler les deux modèles.

La première permet de manipuler les machines à signaux rationnelles dans un cadre purement programmatoire : elles fournit les objets et classes nécessaires à la manipulation du modèle. Son utilisation ne peut se faire que par la programmation : elle prend en entrée des objets Java et produit d'autres objets Java. Elle est donc utilisée lors des implantations pour avoir accès à des données directement compréhensible pour le langage, qui peuvent être facilement manipulés, copiés et transformés pour produire des objets correspondant à leur discrétisation.

La deuxième permet de manipuler les machines à signaux comme des fichiers à part, facile à lire et écrire pour l'homme, qui sont ensuite interprétés sous le format précédent. Elle permet donc de créer facilement des machines, de les exécuter, et d'observer leurs calculs. Elle est utilisée pour produire des données d'entrées pour la discrétisation, qui seront ensuite transformées en objets Java avant de subir un des processus de discrétisation.

La troisième a été produite dans le cadre de cette thèse et permet de manipuler les automates cellulaires à une dimension avec le voisinage usuel. Bien qu'il existe déjà de tels programmes, le choix d'en développer un nouveau s'est fait pour deux raisons :

— tout d'abord, les outils existants, bien que très complets, n'étaient pas toujours simples à interfacer avec les machines à signaux en Java sans passer par des fichiers intermédiaires,

— ensuite, avoir le code source à disposition et bien le connaître permet de le modifier facilement pour pouvoir l'inscrire dans le cadre de la discrétisation et d'y coder la notion de signal discret.

Un outil simple à donc été spécifiquement créé et enrichi au fil des implantations du processus de discrétisation.

Les différentes discrétisations implantées l'ont donc été en Java, en s'appuyant sur ces outils, et en suivant les processus théoriques décrits dans le présent manuscrit. Les structurations de ces implantations sont données dans les Annexes A, B et C.

Plan

Ce mémoire se découpe en deux parties.

La première partie couvre trois chapitres, présentant les préliminaires nécessaires à la compréhension, mais aussi les premières méthodes basiques de discrétisation.

Le Chapitre 1 donne les définitions qui seront utilisées tout au long de ce mémoire, notamment celles autour des deux modèles clefs.

Le Chapitre 2 présente une méthode de discrétisation exacte d'une sous-classe des machines à signaux, puis prouve la correction de cette technique.

Le Chapitre 3 s'intéresse à une méthode de discrétisation brutale : ici, pas de restrictions autre que la rationalité des machines à signaux, mais la méthode n'assure aucune forme d'approximation et échoue lorsqu'un tel cas se présente. Elle ne s'intéresse qu'à la transformation directe des méta-signaux et règles de collisions en états et règles de transition.

La deuxième partie, en trois chapitres, propose une approche avancée de la résolution du problème à travers la notion de modularité ainsi que comment elle peut être altérée pour gérer certains cas particuliers.

Le Chapitre 4 introduit la notion de modularité et pose les définitions qui y sont propres, que ce soit dans sa vision continue ou discrète.

Le Chapitre 5 donne les premiers algorithmes implantant la méthode et s'appuyant sur la modularité pour fournir une discrétisation basique. Celle-ci possède encore des défauts dans certains cas problématiques majeurs, notamment les structures fractales.

Le Chapitre 6 manipule la modularité dans deux cadres différents : son utilisation théorique pour prouver l'exactitude d'une discrétisation automatique, mais aussi comment y exprimer une dynamique infinie ou une structure fractale avec ou sans augmentation du nombre de modules.

Enfin, une conclusion viens rappeler les différentes méthodes et les résultats des discrétisations mis en œuvre dans le présent manuscrit et propose des perspectives de recherche sur le sujet.

Première partie

Définitions et premiers résultats

Chapitre 1

Machines à signaux et automates cellulaires

Ce chapitre donne les définitions formelles des modèles de calcul centraux dans cette thèse. Il s'intéresse tout d'abord à exprimer ce que sont les machines à signaux. Il définit ensuite les automates cellulaires, vers lesquels les machines à signaux seront compilés. Enfin, la notion de simulation de machines à signaux par automates cellulaires est précisée, la discrétisation étant la création d'un AC capable d'une telle tâche.

1.1 Machines à signaux

1.1.1 Méta-signaux et signaux

Ce modèle manipule des signaux qui se déplacent et se rencontrent dans l'espace. Ces signaux sont des instances de méta-signaux qui déterminent l'information transportée et la vitesse des signaux.

Définition 1 (Méta-signal) Un méta-signal est une paire information transportée/vitesse. Il est noté μ.

L'information transportée est toujours implicite dans tout le reste de cette thèse. Elle est représentée sur les figures par la couleur et/ou le nom du méta-signal.

La vitesse d'un méta-signal est obtenue par une fonction de vitesse.

Définition 2 (Fonction de vitesse) La fonction de vitesse associe un réel à chaque méta-signal d'une machine à signaux. Elle est notée S.

Il n'y a qu'un nombre fini de méta-signaux pour une machine donnée. Quand cela ne prête pas à confusion, l'information véhiculée est utilisée pour désigner le méta-signal. La vitesse est une valeur réelle qui représente un déplacement par unité de temps. Les signaux ne peuvent se déplacer sans que le temps ne s'écoule (le temps ne peut qu'augmenter). Deux méta-signaux sont parallèles s'ils ont la même vitesse (tout méta-signal est parallèle à lui-même).

Un signal est une instance d'un méta-signal se trouvant à une position donnée (dans l'espace-temps) et se déplaçant.

Définition 3 (Signal) Un signal se définit par son méta-signal et sa position.

La vitesse d'un signal est celle de son méta-signal. Deux signaux sont *parallèles* s'ils ont la même vitesse. Chaque signal se déplace de manière rectiligne uniforme suivant sa vitesse : si un signal de vitesse v se trouve à la position x_0 au temps t_0, il se trouvera au temps t à la position x calculée par : $x = x_0 + v \cdot (t - t_0)$. Un signal de vitesse positive se déplace vers la droite, un signal de vitesse négative vers la gauche, et un signal de vitesse nulle reste à la même position.

La Figure 1.1 donne des exemples de signaux. Dans toutes les figures suivantes, les méta-signaux ne sont pas précisés car il suffit d'observer les signaux pour comprendre les vitesses et informations transportées (la couleur dans la représentation graphique).

Nom	Vitesse
a	-1
b	0
c	1
d	2
e	1/4

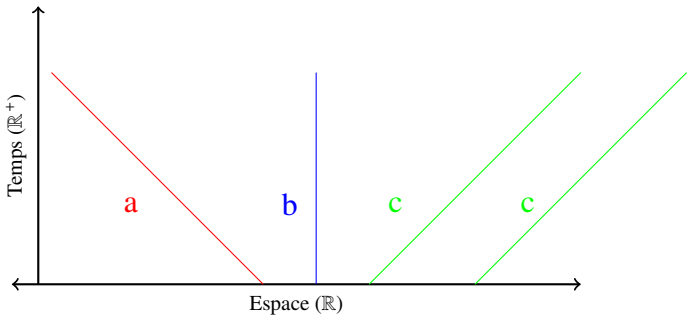


FIGURE 1.1 – Instanciations de méta-signaux.

1.1.2 Collisions

Une collision se produit quand deux ou plus signaux se rencontrent. Il s'agit de rencontres exactes : deux signaux passant près l'un de l'autre n'interagissent que s'ils se rencontrent exactement en un point.

Les signaux étant des instances de méta-signaux à des positions données, le résultat d'une rencontre entre signaux ne dépend que de leurs méta-signaux car on désire une dynamique uniforme dans le temps et l'espace.

Une règle de collision se définit donc par un ensemble d'au moins deux méta-signaux (se rencontrant) et un autre ensemble de méta-signaux (les remplaçant). Les collisions ne sont que des applications des règles.

Définition 4 (Règle de collision) Une règle (de collision) se définit par un ensemble d'au moins deux méta-signaux entrants (ρ^-) et par l'ensemble des méta-signaux sortants (ρ^+), aucun de ces ensembles ne contenant deux méta-signaux distincts parallèles. Elle se note $\rho = \rho^- \rightarrow \rho^+$. La règle est dite *blanche* si $\rho^- = \rho^+$, comme si les signaux se croisaient sans interagir. Elle est dite *annihilante* si $\rho^+ = \emptyset$.

Il ne peut y avoir qu'une règle définie pour le même ensemble de méta-signaux. Le système est déterministe.

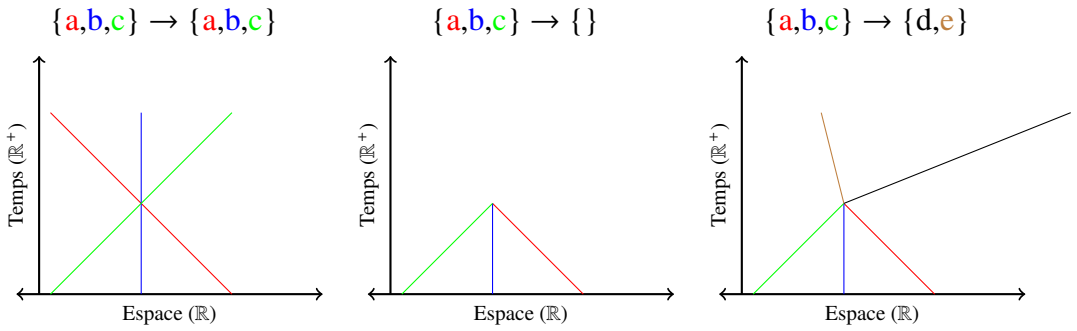
Il ne peut y avoir deux signaux parallèles dans les signaux entrants, car pour être présents à la collision, ils devraient être au même endroit précédemment. De même pour

les signaux sortants, si deux signaux sont de même vitesse, ils seraient instantanément en collision ce qui pourrait provoquer des collisions en chaîne sans que le temps avance.

Les signaux sortants sont des instances des méta-signaux indiqués par la règle de collision, leurs positions initiales étant celle de la collision.

Si des signaux correspondant aux méta-signaux ρ^- se rencontrent à la position p, et s'il existe une règle $\rho^- \to \rho^+$, alors les signaux disparaissent et sont remplacés par des instances des méta-signaux ρ^+ à la même position. Les signaux entrants sont dits *morts*, et les signaux sortants *naissant*.

Si aucune règle n'est définie pour une collision, une règle par défaut est appliquée (généralement une règle blanche ou annihilante). Une collision peut augmenter ou réduire le nombre de signaux, tant qu'il n'y a qu'un nombre fini de collision, un nombre fini de signaux n'engendre qu'un nombre fini de signaux. La Figure 1.2 donne des exemples de collisions.



(a) collision blanche (b) collision annihilante (c) collision générique

FIGURE 1.2 – Exemples de collisions.

1.1.3 Machines à signaux

Une machine à signaux se définit par ses méta-signaux et ses règles.

Définition 5 (Machines à signaux (MS)) Une *machine à signaux* est un triplet (M, S, R) tel que :

— M est un ensemble fini de méta-signaux,
— $S : M \to \mathbb{R}$ est la fonction vitesse,
— R est un ensemble fini de règles de collisions.

À partir de maintenant, $\mathfrak{A}$ (et $\mathfrak{B}$...) désigne toujours une machine à signaux.

Définition 6 (Machine à signaux rationnelle) Une machine à signaux est rationnelle si toutes les vitesses et les positions des signaux sur toute configuration initiale (cette notion est définie dans la suite) sont rationnelles.

Pour toute machine rationnelle, les opérations en jeu garantissent que l'on reste avec des valeurs rationnelles. Une récurrence immédiate montre qu'au bout d'un nombre fini de collisions, celles-ci ont toutes eu lieu à des positions (à coordonnées) rationnelles. Cette thèse ne s'intéresse qu'aux machines rationnelles, car celles-ci sont manipulables exactement par un ordinateur (à l'exception des structures fractales) et rendent donc possible l'implantation des différents procédés proposés.

1.1.4 Configuration

Pour pouvoir décrire l'évolution du système, il faut pouvoir exprimer son état à un instant donné. Cela se fait à travers la notion de *configuration* : une configuration d'une machine à signaux est l'état, en termes de signaux et de collisions présents, de la ligne réelle à un instant précis. Celle-ci étant infinie, pour pouvoir l'exprimer, seules les positions sur lesquelles se trouve un signal ou une collision sont précisées, toutes les autres sont considérées vides. Une machine à signaux est démarrée sur une configuration appelée *configuration initiale*. Formellement, une configuration se définit comme suit :

Définition 7 ((𝔄-)configuration) Une (𝔄-)configuration, $\mathfrak{c}$, est une association de chaque point de la ligne réelle à un méta-signal, une règle de collision ou la valeur $\oslash$ indiquant qu'il n'y a rien là. Il y a un nombre fini de points non-$\oslash$.

Soit μ un méta-signal et v une valeur dans $M \cup R$, $\mu \ltimes v$ signifie « est égal (au méta-signal) ou appartient à l'ensemble des méta-signaux en sortie (de la règle de collision) », i.e. $v = \mu$ ou $v = \rho^- \to \rho^+ \wedge \mu \in \rho^+$.

Un signal est *sans fin* s'il n'entre pas en collision et a donc une durée de vie (son temps d'existence au sein de la machine à signaux) de $+\infty$.

Définition 8 (Cône de lumière) Soient v_{max} et v_{min} les valeurs maximale et minimale prises par la fonction vitesse S. Le cône de lumière du point (x_0,t_0) est défini par l'ensemble des points (x, t) tels que $t_0 < t \cap v_{max} \cdot (t - t_0) < x - x_0 < v_{min} \cdot (t - t_0)$.

1.1.5 Dynamique

Afin d'exprimer la dynamique d'une machine à signaux à partir d'une configuration initiale particulière, simplement observer l'enchaînement des configurations n'est pas possible. En effet, le temps étant continu, le nombre de ces configurations l'est aussi, et il n'est donc pas possible de les manipuler dans leur totalité pour en exprimer la dynamique. Pour ce faire, il est possible de choisir des étapes discrètes, et le meilleur choix pour de telles instants est le moment des collisions. Cela mène à définir les séquences de collisions.

Définition 9 (Séquence de collisions) Soit $\mathfrak{c}$, le *délai à la prochaine collision*, $\Delta(\mathfrak{c})$ est égal au minimum des nombres réels positifs d tels que :

$$\exists x_1, x_2 \in \mathbb{R}, x_1 \neq x_2, \exists \mu_1, \mu_2 \in M \begin{cases} \mu_1 \ltimes \mathfrak{c}(x_1) \ , \\ \mu_2 \ltimes \mathfrak{c}(x_2) \ , \\ x_1 + d \cdot S(\mu_1) = x_2 + d \cdot S(\mu_2) \ . \end{cases}$$

Il est de $+\infty$ si d n'existe pas.

La *séquence de collisions* est définie par : $t_0 = 0$, $t_{n+1} = t_n + \Delta(\mathfrak{c}_{t_n})$ où $\mathfrak{c}_t$ est la configuration au temps t.

Cette séquence est finie s'il existe un n tel que $\Delta(\mathfrak{c}_{t_n}) = +\infty$. Sinon, puisqu'elle est croissante, elle admet une limite. Si la séquence est finie ou sa limite est infinie, alors le diagramme espace-temps est entièrement défini. Sinon, il y a une *accumulation* et la configuration au temps limite reste indéfinie. Dans le cadre de cette thèse, les accumulations sont hors-propos, mais deux cas particuliers sont traités dans le Chapitre 6.

Une exécution (ou orbite) est l'application d'une machine à signaux sur une configuration initiale donnée. La fin du calcul d'une machine à signaux correspond, au choix, à l'apparition d'un signal donné, à ne plus avoir de collisions, à un temps ou nombre de collisions prédéfini... La dernière configuration atteinte par une exécution est appelée *configuration finale*.

Définition 10 (Temps final) Le temps final d'une exécution d'une machine à signaux est celui de sa configuration finale (si elle existe). Il est noté t_∞.

Un *diagramme espace-temps* est la collection des configurations consécutives qui forment une image en deux dimensions.

Définition 11 (Diagramme espace-temps) Un diagramme espace-temps sur une configuration $\mathfrak{c}$ est notée $\mathbb{D}(\mathfrak{c})$ et est une fonction de $\mathbb{R} \times [0, t_\infty] \to \{\varnothing\} \cup M \cup R$ correspondant aux traces des différents signaux.

Dans les diagrammes, le temps se déplace toujours vers le haut comme dans la Figure 3.

1.2 Automates cellulaires

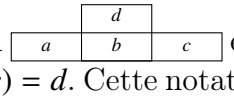
1.2.1 Définition formelle

Le modèle vers lesquelles les machines à signaux sont transformées est celui des automates cellulaires. Celui-ci fonctionne dans le discret et se définit comme suit :

Définition 12 (Automate cellulaire) Un *automate cellulaire* à une dimension est un triplet $(Q, f, \#)$ tel que :
- Q est un ensemble fini *d'états*,
- $f : Q^3 \to Q$ est la *fonction de transition* locale, et
- $\#$ est un état spécial tel que $f(\#, \#, \#) = \#$ (*l'état quiescent*).

Cette définition limite le voisinage aux deux voisins les plus proches pour simplifier la suite. Cela cause une limitation des vitesses qui amènent à normaliser les machines à signaux (donné en fin de chapitre).

Dans la définition « traditionnelle » des automates cellulaires, l'état quiescent n'existe pas forcément. Dans le cadre de cette thèse, l'existence d'un état quiescent est obligatoire (car il sera utilisé pour la discrétisation).

La notation 

est utilisée pour définir la fonction de transition de l'AC et signifie $f(a, b, c) = d$. Cette notation est importante car elle met en valeur la fenêtre utilisée pour calculer le prochain état. Seul ce qui se passe dans cette fenêtre est à considérer.

Une nouvelle *configuration* est obtenue par l'application de la fonction de transition locale à chacune des cellules de l'automate cellulaire. Une nouvelle configuration est engendrée à chaque étape temporelle. Pour pouvoir démarrer ce calcul, la première configuration est fournie à l'automate et est appelée *configuration initiale*.

Dans cette thèse, $\mathcal{A}$ (et $\mathcal{B}$...) désigne toujours un automate cellulaire.

Définition 13 (($\mathcal{A}$-)configuration) Une *($\mathcal{A}$-)configuration*, c, est une association de chaque point de la ligne entière à un état de Q. C'est un élément de $Q^{\mathbb{Z}}$.

Un *site* est une cellule à une itération donnée. Dans l'évolution depuis une configuration c, la valeur de la cellule x à l'étape t est notée c^t_x.

1.2.2 Signaux discrets

L'objectif du présent manuscrit est de proposer une méthode de discrétisation des machines à signaux en automates cellulaires. Or, avec les seuls définitions précédentes, s'il est tout à fait possible de proposer des techniques de discrétisation, l'ajout d'un objet précis et défini au sein des AC ayant le rôle de signal est un avantage important. Ces structures sont ici appelées $\mathfrak{A}$-AC-signaux, μ-AC-signaux ou plus simplement AC-signaux. Ils sont les principaux objets manipulés par le système de discrétisation, et tout AC ainsi obtenu les utilisera donc majoritairement. Ce faisant, la discrétisation obtenue produit des diagrammes espace-temps discrets dont la représentation est graphiquement proche de celle du diagramme espace-temps continu d'origine.

Pour pouvoir définir ces AC-signaux, il convient d'exprimer tout d'abord comment ils représentent des signaux continus, en permettant de trouver la trace d'un diagramme espace-temps d'une MS dans un diagramme espace-temps d'un AC.

Pour le reste de cette sous-section, $\mathfrak{A}$ est une machine à signaux dont les vitesse sont comprises entre -1 et 1 et μ un de ses méta-signaux.

Définition 14 (MS-CA (ou $\mathfrak{A}$-$\mathcal{A}$) relation de représentation) Soit $W = M \cup R$ l'ensemble des méta-signaux et de règles de collisions d'une machine à signaux $\mathfrak{A}$ et Q l'ensemble des états d'un automate cellulaire $\mathcal{A}$. Une $\mathfrak{A}$-$\mathcal{A}$ *relation de représentation* $\mathcal{R}$ relie W et Q de la façon suivante : $\lambda \mathcal{R} q$ si et seulement si λ est *représenté par* q. Tout méta-signal et règle de collision doit être représenté par au moins un état. L'état quiescent correspond à $\oslash$, il ne doit donc rien représenter.

Il s'agit d'une relation : un méta-signal/règle peut être représenté par plusieurs états et un état peut représenter plus qu'un méta-signal/règle.

Très grossièrement, en supposant que la grille $\mathbb{Z}\times\mathbb{N}$ utilisée pour décrire l'évolution d'un automate cellulaire vienne être plaquée sur le diagramme espace-temps d'une machine à signaux, les AC-signaux correspondent aux sites qui se trouvent sur la trace d'un MS-signal qui croise le bas du carré leur correspondant.

Plus formellement, un $\mathfrak{A}$-AC-signal ou μ-AC-signal se définit comme suit :

Définition 15 ($\mathfrak{A}$-AC-signal / μ-AC-signal) Soit μ un méta-signal d'une machine à signaux $\mathfrak{A}$. Un *μ-AC-signal* est défini par (x, b, d, φ) où :

— $x \in \mathbb{Z}$, est la *position de base*,

— $b, d \in \mathbb{N}$ avec $b \leq d$ (d peut être ∞), sont les dates de *naissance* et de *mort*, et
— $\varphi \in [0, 1]$ est la *phase*.

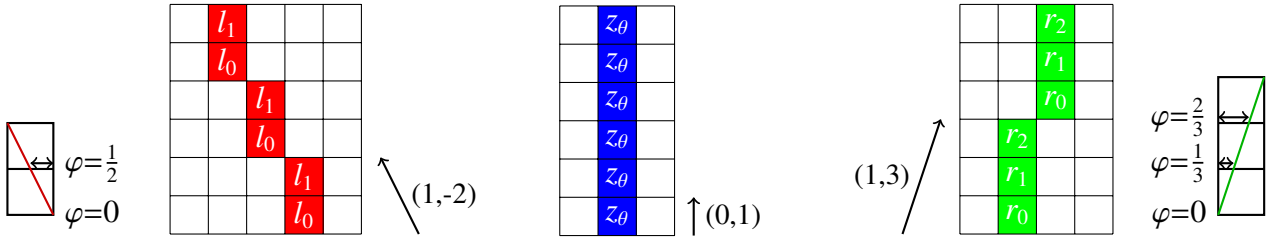
Il correspond à l'ensemble des sites :

— $\{(x + \lfloor \varphi + (t-b).S(\mu) \rfloor, t) \mid t \in [b, d]\}$, si $0 < S(\mu)$ (signal allant vers la droite),
— $\{(x - \lfloor \varphi - (t-b).S(\mu) \rfloor, t) \mid t \in [b, d]\}$, si $S(\mu) < 0$ (signal allant vers la gauche),
— $\{(x + \lfloor \varphi \rfloor, t) \mid t \in [b, d]\}$, sinon (signal stationnaire).

Le AC-signal doit représenter $\mu : \mu \mathcal{R} c^t_x$ pour tout (x, t) dans le μ-AC-signal (c est la configuration initiale du diagramme espace-temps).

La *vitesse* d'un $\mathfrak{A}$-AC-signal est celle de μ. La *période temporelle* d'un μ-AC-signal est le dénominateur minimum de $S(\mu)$.

Tout AC-signal de vitesse $\frac{a}{b}$ a une dynamique périodique simple : il se déplace de a cellules sur le côté (gauche ou droit) en b itérations. La Fig. 1.3 donne des exemples de $\mathfrak{A}$-AC-signals.



(a) signal de vitesse $-\frac{1}{2}$ (b) signal stationaire (vitesse nulle et phase constante) (c) signal de vitesse $\frac{1}{3}$

FIGURE 1.3 – Exemples de $\mathfrak{A}$-AC-signaux et de phases.

Les entiers b et d sont les numérations des itérations qui marquent le début (*birth*) et la fin (*death*) du μ-AC-signal, x est la position spatiale du début du signal. Pour les signaux allant vers la droite, la phase φ est la distance entre le coin bas-gauche de la cellule et l'emplacement réel où le signal (continu) entre dans la cellule par le bas. Cela correspond à la différence entre la position entière dans l'AC et la position continue dans la MS. Si un MS-signal (allant vers la droite) est lancé à (x, t) avec t un entier, alors le $\mathfrak{A}$-AC-signal correspondant débute à $(\lfloor x \rfloor, t)$ avec la phase $x - \lfloor x \rfloor$. Si t n'est pas un entier, alors la position qu'il devrait avoir à $\lfloor t \rfloor$ doit être calculée. Si le signal va vers la gauche, la règle est adaptée en miroir : la phase est la distance au coin bas-droit de la cellule. Cela est illustré dans la Fig. 1.3. Tout les signaux stationnaires ont une phase constante.

1.2.3 États AC-signaux

Les AC-signaux sont donc des ensembles de sites dans des états particuliers, qui marquent la présence d'un signal continu. L'ensemble des états AC-signaux se définit comme suit :

Définition 16 Un *état d'un AC-signal* s est un état d'un automate cellulaire produit par la discrétisation définit par : $\exists \mu_1 \in M$ tel que $\mu_1 \mathcal{R} s$ et $\forall \mu_2 \in M, \neg(\mu_2 \mathcal{R} s)$ avec $\mathcal{R}$ une MS-CA relation de représentation.

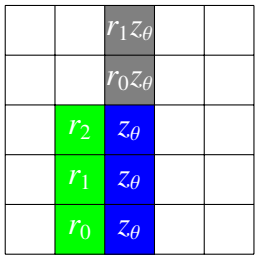
L'ensemble des états AC-signaux d'un automate cellulaire est noté $Q_E \subset Q$.
L'ensemble des états AC-signaux représentant exactement un seul *signal* est noté $Q_E^0 \subset Q_E$.

1.2.4 États composites

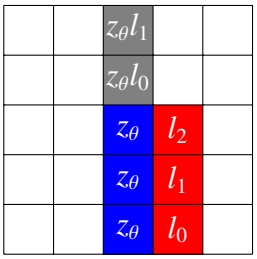
Cette notion d'AC-signaux permet de manipuler une structure discrète s'apparentant aux signaux des machines à signaux. Cependant, par leur nature discrète, ils ne peuvent être vus comme des points de dimension 0 se déplaçant. De fait, ils possèdent une « épaisseur » à la fois spatiale et temporelle, ce qui peut causer la présence de plusieurs AC-signaux sur le même site. Cette situation se résout en définissant la notion d'état composite, qui marque cette information. Plus formellement, il se définit comme suit :

Définition 17 (État composite) Un *état composite* s est un état d'un automate cellulaire produit par la discrétisation définit par : $\exists \mu_1, \mu_2 \in M$ tels que $\mu_1 \neq \mu_2 \wedge \mu_1 \mathcal{R}\, s \wedge \mu_2 \mathcal{R}\, s$, et $\forall \rho \in R$, $\neg(\rho \mathcal{R}\, s)$ avec $\mathcal{R}$ une MS-CA relation de représentation.
L'ensemble des états composite d'un automate cellulaire est noté $Q_C \subset Q$.

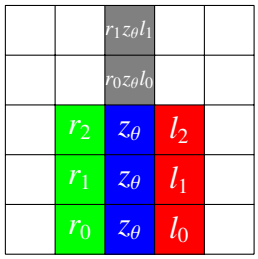
Les états composites sont surtout présents lors des phases pré et post collisions dans les automates cellulaires simulant les machines à signaux. Dans la Figure 1.4 illustrant cette notion, ainsi que dans le reste de ce manuscrit, la couleur grise indique une cellule dans un état composite, alors que les autres sont des cellules ne contenant qu'un seul AC-signal.



(a) état composite de deux AC-signaux, un allant vers la droite et un stationnaire



(b) état composite de deux AC-signaux, un allant vers la gauche et un stationnaire



(c) état composite de trois AC-signaux, allant vers la droite, la gauche et un stationnaire

FIGURE 1.4 – Exemple de l'évolution d'AC-signaux en états composite.

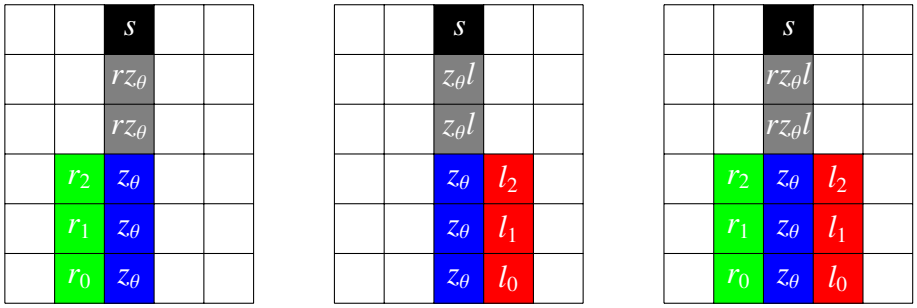
1.2.5 États collisions

Avec les définitions précédentes, les signaux ont désormais un équivalent discret dans les automates cellulaires. Cependant, les machines à signaux manipulent un autre concept très important puisqu'il est à l'origine de leur procédé calculatoire. Il s'agit des collisions entre signaux qui doivent donc elles aussi avoir un équivalent discret. Celui-ci porte le nom d'état collision, et marque un site comme contenant une ou plusieurs collisions de AC-signaux. Il se définit formellement par :

Définition 18 (État collision) Un *état collision* s est un état d'un automate cellulaire produit par la discrétisation définit par : $\exists \rho \in R$ telle que $\rho \mathcal{R} s$ et $\forall \mu_1 \in M, \neg(\mu_1 \mathcal{R} s)$ avec $\mathcal{R}$ une MS-CA relation de représentation. Les états collision, bien que contenant plusieurs signaux continus, sont distincts des états composites. Au sein d'un état collision, le signal discret disparaît.

L'ensemble des états collision d'un automate cellulaire est noté $Q_K \subset Q$.

Les états collisions sont généralement les prédécesseurs ou les suites logiques d'un ou plusieurs états composites, entrant en collision et en émettant d'autres. Dans la Figure 1.5 illustrant cette notion, ainsi que dans le reste de ce manuscrit, la couleur noire indique qu'une cellule est dans un état collision.



(a) collision de deux AC-signaux, un allant vers la droite et un stationnaire

(b) collision de deux AC-signaux, un allant vers la gauche et un stationnaire

(c) collision de trois AC-signaux, allant vers la droite, la gauche et un stationnaire

FIGURE 1.5 – Exemple de l'apparition d'un état collision.

1.2.6 États complexes

Les états AC-signaux, composites ou collisions sont des cas « idéaux », dans lesquels un seul type d'événement se produit (un seul AC-signal, plusieurs AC-signaux ne se rencontrant pas ou une collision), mais il est possible que plusieurs d'entre eux se produisent en même temps, c'est-à-dire un état qui représente à la fois la présence d'une ou plusieurs collisions, et de signaux qui ne font partie d'aucune collisions. Ces états sont appelés états complexes et sont définis comme suit :

Définition 19 (État complexe) Un *état complexe* s est un état d'un automate cellulaire produit par la discrétisation définit par : $\exists \rho \in R$ telle que $\rho \mathcal{R} s$ et $\exists \mu_1 \in M$ tel que $\mu_1 \mathcal{R} s$ avec $\mathcal{R}$ une MS-CA relation de représentation.

L'ensemble des états complexes d'un automate cellulaire est noté $Q_X \subset Q$.

La Table 1.1 résume les différents ensembles d'états en fonction du nombre de collisions et de méta-signaux qu'ils représentent. Le symbole # est la notation utilisée pour l'état quiescent.

Nombre de méta-signaux \ Nombre de collisions	0	1+
0	#	Q_K
1	Q_E	Q_X
2+	Q_C	Q_X

TABLE 1.1 – Résumé des différents types d'états.

1.3 Dynamiques, simulation et normalisation

La dynamique de tout diagramme espace-temps de machine à signaux peut être représentée par un graphe acyclique labellisé (Labeled Directed Acyclic Graph en anglais, ou LDAG). Simuler une machine à signaux revient donc à pouvoir engendrer une copie parfaite de tout les LDAG qu'elle est capable produire, à partir des configurations initiales.

1.3.1 Représentation de diagramme espace-temps sous forme de LDAG

La dynamique dans une machine à signaux correspond à l'ordre dans lequel les collisions s'enchaînent. Ces collisions sont des étapes discrètes liées par des signaux : une collision est juste avant une autre si un signal engendré par la première se termine dans la deuxième. En considérant les collisions comme des nœuds et les signaux comme des arêtes (en ajoutant un nœud pour chaque signal sans fin et signal dans la configuration initiale), un LDAG labellisé avec des règles de collisions et des méta-signaux est créé. Ce LDAG est l'expression de la dynamique du diagramme espace-temps et représente *l'ordre de causalité* de l'exécution. Ainsi, si les LDAG de deux exécutions sont identiques, alors les exécutions sont identiques *du point de vue de la dynamique*.

Définition 20 ($\mathfrak{A}$-DAG ou LDAG) Soit $\mathfrak{A}$ une machine à signaux. Un $\mathfrak{A}$-*DAG* est un graphe acyclique dirigé où les arêtes (resp. nœuds) sont labellisées avec des éléments de M (resp. $R \cup \{\bot, \top\}$). C'est la représentation de l'ordre de causalité d'un diagramme espace-temps de $\mathfrak{A}$. Le LDAG engendré depuis une configuration $\mathfrak{c}$ est noté $\widehat{\mathfrak{c}}$.

Le label $\bot$ est utilisé pour les signaux présents dans la configuration initiale. Le label $\top$ est celui des signaux sans fin.

1.3.2 Simulation de dynamique

Afin de pouvoir exprimer qu'un automate cellulaire simule une machine à signaux, il faut exprimer ce que veut dire simuler une machine à signaux. La première étape est de définir comment une machine peut en simuler une autre. L'expression la plus simple est qu'une MS simule une autre si toute configuration de la deuxième peut être transformée en une configuration de la première tout en engendrant le même LDAG à un renommage près. Cette simulation passe par l'utilisation de fonctions de conversion, de relabellisation et d'association de nœuds. Formellement :

Définition 21 (Simulation de machines à signaux) Soient $\mathfrak{A}$ et $\mathfrak{B}$ deux machines à signaux et $C_\mathfrak{A}$, $C_\mathfrak{B}$ les ensembles de leurs configurations. La machine $\mathfrak{B}$ *simule* $\mathfrak{A}$ s'il existe une *fonction*

de conversion $\zeta : C_{\mathfrak{A}} \to C_{\mathfrak{B}}$ et une *fonction de relabellisation* $\psi : M_{\mathfrak{B}} \cup R_{\mathfrak{B}} \to M_{\mathfrak{A}} \cup R_{\mathfrak{A}}$ telles que :

$$\forall \mathfrak{c} \in C_{\mathfrak{A}}, \widehat{\mathfrak{c}} = \psi(\widehat{\zeta(\mathfrak{c})})$$

où ψ est canoniquement étendue aux LDAG comme relabellisant chaque nœuds et arêtes.

De plus, $\forall \mathfrak{c}_1 \in C_{\mathfrak{A}}, \mathfrak{c}_2 \in C_{\mathfrak{B}}$, il doit exister la fonction d'association de nœuds $\mathcal{N} : \widehat{\mathfrak{c}_1} \to \widehat{\mathfrak{c}_2}$ qui associe tout nœud de coordonnées (x, y) dans $\widehat{\mathfrak{c}_1}$ à un nœud de coordonnées $(ax + b, cy)$ dans $\widehat{\mathfrak{c}_2}$, avec $0 < a$ et $0 < c$.

La simulation est un pré-ordre (réflexive et transitive).

1.3.3 LDAG dans les AC et simulation

D'après la définition 15, chaque μ-AC-signal est une ligne discrète où tout les sites représentent le méta-signal μ (selon une relation définie dans la Déf. 14). Avec celle-ci, il devient possible de définir la notion de LDAG dans les automates cellulaires, et de formaliser la simulation des machines à signaux par ceux-ci. En premier lieu, il convient de définir formellement les LDAG dans les AC :

Définition 22 (LDAG dans les AC) Soient $\mathfrak{A}$ une machine à signaux, $\mathcal{A}$ un automate cellulaire et $\mathcal{R}$ une relation de représentation $\mathfrak{A}$-$\mathcal{A}$. Soient c une configuration de $\mathcal{A}$ et $\mathcal{D}$ le diagramme espace-temps engendré par cette configuration. L'ensemble des LDAG $\widehat{c}^{\mathcal{R}}$ est de la forme suivante :

- les arêtes correspondent à des μ-AC-signaux dans $\mathcal{D}$;
- les sommets sont les sites qui représentent les règles de collisions plus un sommet $\perp$ pour chaque μ-AC-signal présent dans la configuration initiale et un sommet $\top$ pour chaque μ-AC-signal sans fin ;
- les arêtes sont entrantes dans un sommet si le vecteur allant du site le plus haut du signal au site du sommet est $(-1, 1)$, $(0, 1)$, $(1, 1)$ ou $(0, 0)$ (si le site du sommet et le site le plus haut du signal sont confondus), ou un μ-AC-signal sans fin et son sommet $\top$ dédié ;
- les arêtes sont sortantes d'un sommet si le vecteur allant du site le plus bas du signal au site du sommet est $(-1, -1)$, $(0, -1)$, $(1, -1)$ ou $(0, 0)$ (si le site du sommet et le site le plus bas du signal sont confondus), ou un μ-AC-signal présent dans la configuration initiale et son sommet $\perp$ dédié ;
- pour chaque site c_x^t, $\forall \mu \in M_{\mathfrak{A}}$ tel que $\mu \mathcal{R} c_x^t$, c_x^t appartient à un μ-AC-signal, et $\forall \rho \in R_{\mathfrak{A}}$ telle que $\rho \mathcal{R} c_x^t$, il existe un nœud correspondant ;
- les labels correspondent au méta-signal représenté par le μ-AC-signal et les règles de collisions représentées par le sommet.

Maintenant, la Définition 21 peut être étendue pour définir la simulation de machines à signaux par un automate cellulaire :

Définition 23 (Simulation de MS par un AC) Soient $\mathfrak{A}$ une machine à signaux, $\mathcal{A}$ un automate cellulaire et $C_{\mathfrak{A}}$, $C_{\mathcal{A}}$ les ensembles de leurs configurations. $\mathcal{A}$ *simule* $\mathfrak{A}$ s'il existe une *fonction de conversion* $\zeta : C_{\mathfrak{A}} \to C_{\mathcal{A}}$ et une *relation de représentation* $\mathfrak{A}$-$\mathcal{A}$ $\mathcal{R}$ telles que :

$$\forall \mathfrak{c} \in C_{\mathfrak{A}}, \widehat{\mathfrak{c}} = \widehat{\zeta(\mathfrak{c})}^{\mathcal{R}}.$$

De plus, $\forall \mathfrak{c} \in C_{\mathfrak{A}}, c \in C_{\mathcal{A}}$, il existe une fonction d'association de nœuds $\mathcal{N} : \widehat{\mathfrak{c}} \to \widehat{c}$ qui associe tout nœud de coordonnées (x, y) dans $\widehat{\mathfrak{c}}$ à un nœud de coordonnées $\lfloor (ax + b, cy) \rfloor$ dans $\widehat{c}$, avec $0 < a$ et $0 < c$.

La discrétisation du Chapitre 2 assure l'unicité du LDAG discret engendré mais ce n'est pas le cas des discrétisations présentées après.

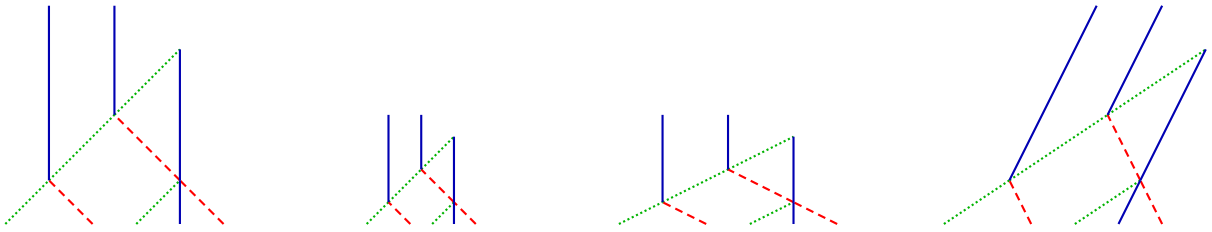
Définition 24 (Automate cellulaire simulant) Un automate cellulaire (à une dimension) est appelé *automate cellulaire simulant* (ACS) s'il simule une machine à signaux et l'ensemble de ses états $Q = \{\#\} \cup Q_E \cup Q_C \cup Q_K \cup Q_X$ tels que l'union de ces ensembles est disjointe. Un automate cellulaire simulant est dit *absolu* si ses règles sont localement capables de simuler le comportement de tout les ensembles possibles de signaux, c'est à dire qu'il peut simuler n'importe quelle interaction de signaux sur l'espace et la durée d'une cellule.

Sachant qu'il faudrait une infinité d'états pour un tel automate, les discrétisations proposées ici utilisent différentes techniques pour forcer un nombre fini d'états et ne sont donc pas absolus.

Le but de cette thèse est de proposer la création automatique d'un ACS à partir d'une machine à signaux, ainsi que la configuration initiale associée. Les automates cellulaires simulant sont utilisés à partir du Chapitre 3.

1.3.4 Normalisation

La normalisation d'une machine à signaux est le procédé de transformation des vitesses des méta-signaux ainsi que des positions des signaux dans la configuration initiale, tout en conservant sa dynamique. Changer linéairement les vitesses des méta-signaux ou leurs positions dans la configuration initiale n'affecte pas les dynamiques de la machine (tant que des coefficients multiplicateurs positifs sont appliqués) [DL03, Chap. 5]. La Fig. 1.6 donne un exemple de ce genre de transformation. Ceci justifie la correction du processus de normalisation, celui-ci n'influençant pas la dynamique de la machine.



(a) original (b) positions divisées par deux (c) vitesses doublées (d) vitesses augmentées de 0.5

FIGURE 1.6 – Exemples de transformations linéaires.

Lemme 25 *Il est toujours possible de transformer les vitesses des méta-signaux d'une machine à signaux rationnelle avec des opérations linéaires de façon à ce qu'elles soient comprises entre* -1 *et* 1 *tout en conservant la dynamique.*

Preuve. Simplement diviser toutes les vitesses par celle dont la valeur absolue est la plus grande suffit à produire des vitesses comprises entre -1 et 1. □

Lemme 26 *Il est toujours possible de transformer les positions de la configuration initiale d'une machine à signaux rationnelle avec des opérations linéaires de façon à ce que toutes les positions non-⊘ soient des entiers naturels tout en conservant la dynamique.*

Preuve. En les multipliant par le plus petit commun multiple des dénominateurs des positions non-⊘, elles deviennent entières. □

Définition 27 (Forme normale) Une machine à signaux rationnelle est *en forme normale* si et seulement si ses vitesses sont comprises entre -1 et 1. Une configuration est *en forme normale* si et seulement si elle correspond à une machine en forme normale et toutes les positions non-⊘ de la configuration initiale sont des entiers.

Ces deux normalisations utilisent uniquement des transformations linéaires sur les vitesses ou les positions dans la configuration initiale, donc elles n'altèrent pas les dynamiques.

Intérêt de la normalisation. Les automates cellulaires à une dimension classiques avec le voisinage usuel ne peuvent contenir de AC-signaux de vitesse supérieure à 1 ou inférieure à -1. C'est pourquoi toutes les vitesses sont fixées entre ces valeurs. Quant à la transformation des positions rationnelles en entiers, elle est directement reliée à la nature des automates cellulaires. De fait, ils fonctionnent sur $\mathbb{Z}$ alors que les signaux continus sont sur des positions rationnelles, cette conversion permet donc de mettre directement des signaux sur la configuration initiale d'un automate cellulaire

Par la suite, il est toujours considéré que les machines à signaux manipulées sont rationnelles et normales.

Chapitre 2

Discrétisation automatique des machines à signaux rationnelles à 3 vitesses

Tout au long de ce chapitre, il est considéré que le résultat d'une règle non précisée est l'état quiescent (#). Ce n'est pas le cas dans le reste de cette thèse.

Ce chapitre s'intéresse à une classe particulière de MS qui se discrétise exactement. Ces machines produisent un diagramme espace-temps régulier prisonnier d'un maillage discret sur lequel s'appuie la discrétisation. Dès que l'on élargit la classe, ce maillage disparaît et la discrétisation exacte devient incertaine. Ces machines se définissent par :

Définition 28 (Machines à signaux rationnelle à 3 vitesses) Une machine à signaux est rationnelle à 3 vitesses (3S-$\mathbb{Q}$) si :

— la machine à signaux est rationnelle, et

— seulement trois vitesses sont disponibles pour les méta-signaux.

Ces machines ont la propriété d'avoir leurs signaux piégés dans un *maillage* (Fig. 2.1). Il s'agit de l'union de demi-droites de $\mathbb{R} \times \mathbb{R}^+$ suivant les trois vitesses disponibles dans une machine 3S-$\mathbb{Q}$. Celles-ci sont arrangées régulièrement de façon à ce que chaque type de demi-droite rencontre les deux autres dans la même collision.

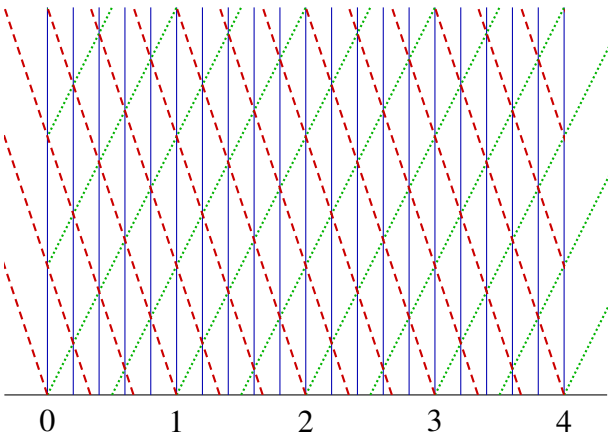


FIGURE 2.1 – Le $(2, 3, 4)$-MS-maillage.

Formellement, un maillage est défini par :

Définition 29 ((p,q,n)-(MS-)maillage) Soit p, q et n trois positions entières (p et q premiers entre eux), le (p,q,n)-*(MS-)maillage* correspond à l'union des demi-droites suivantes de $\mathbb{R}\times\mathbb{R}^+$:

— $\mathfrak{V}_v : 0 \leq t$ et $x = v/(p+q)$ où $v \in \{0, 1, 2, \ldots, n(p+q)\}$,

— $\mathfrak{L}_l : x \leq n$, et $t/q + x = l/q$ où $l \in \{0, 1, 2, \ldots\}$ et

— $\mathfrak{R}_r : 0 \leq x$ et $t/p - x = r/p$ où $r \in \{(-n.p), \ldots, -1, 0, 1, 2, \ldots\}$.

Avec ces maillages, il devient possible d'énoncer la propriété sur laquelle tout le procédé de ce chapitre repose. Elle assure en effet que la discrétisation est toujours possible et exacte pour les machines à signaux rationnelles à 3 vitesses. Cette propriété est la suivante :

Propriété 30 ([BCDL$^+$13, Lem. 1] et [DL13, Lem. 1]) *Pour toute 3S-$\mathbb{Q}$ machine à signaux avec les vitesses $-\frac{1}{q}$, 0 et $\frac{1}{p}$ (p et q premiers entre eux) sur une configuration initiale où les valeurs non-$\oslash$ sont dans $\{0, 1, \ldots, n\}$, les positions non-$\oslash$ (les traces des signaux et collisions) appartiennent au (p,q,n)-(MS-)maillage.*

Les conditions sur les vitesses et les positions non-$\oslash$ peuvent être réduites, mais l'expression du maillage serait plus complexe.

Toute machine à signaux 3S-$\mathbb{Q}$ et configuration initiale peuvent être re-normalisées pour satisfaire ces conditions.

Le nombre de vitesses possibles étant limité à trois par définition, la forme normale d'une machine 3S-$\mathbb{Q}$ possède une propriété supplémentaire sur la forme de ses vitesses. Celle-ci n'existe que pour des raisons pratiques : représenter n'importe quelle vitesse rationnelle entre -1 et 1 sur un automate cellulaire est possible mais la forme choisie permet de diminuer le nombre d'états et de règles de transition nécessaires pour simuler, mais aussi simplifier les algorithmes les engendrant.

Définition 31 (Forme 3S-$\mathbb{Q}$ normale) Une machine à signaux 3S-$\mathbb{Q}$ est *en forme normale* si et seulement si ses vitesses sont $-\frac{1}{q}, 0, \frac{1}{p}$ (avec q et p premiers entre eux). Une configuration est *en forme normale* si et seulement si elle correspond à une machine normalisée et toutes les positions non-$\oslash$ de la configuration initiale sont des entiers naturels.

La normalisation des machines à signaux 3S-$\mathbb{Q}$ est un cas particulier de celle des machines à signaux rationnelles. Se basant sur la même propriété, avec seulement trois vitesses possibles, elle permet de proposer le lemme suivant :

Lemme 32 *Il est toujours possible de transformer les vitesses des méta-signaux d'une machine à signaux 3S-$\mathbb{Q}$ avec des opérations linéaires de façon à ce qu'elles soient $-\frac{1}{q}, 0, \frac{1}{p}$ avec q et p premiers entre eux tout en conservant la dynamique.*

Preuve. En premier lieu, il faut obtenir une vitesse nulle. Pour cela, la vitesse intermédiaire est soustraite aux trois vitesses (il y a trois vitesses rationnelles différentes, donc il y a une maximum, une minimum, et une entre les deux). Ces trois vitesses, $-a$, 0, et b, sont ensuite divisées par a, ce qui les transforment en -1, 0 et $\frac{q}{p}$ ($\frac{q}{p}$ est la forme irréductible de $\frac{a}{b}$). Finalement, elles sont divisées par q, ce qui donne $-\frac{1}{q}, 0, \frac{1}{p}$. □

À partir de maintenant, p et q sont toujours utilisés pour exprimer les dénominateurs des vitesses des méta-signaux de vitesse non-nulle.

Afin de prouver la correction de la discrétisation proposée dans ce chapitre, il est nécessaire de définir la notion de MS-Rencontre. Il s'agit des positions dans le maillage où plusieurs des demi-droites le composant se rencontrent au même point. Formellement, une MS-Rencontre se définit par :

Définition 33 (MS-Rencontre (continue)) Dans le (p, q, n)-(MS-)maillage, la *(MS)-rencontre* de coordonnées (v, r), $\mathfrak{e}_v^r$, correspond à la position dans le diagramme espace-temps à l'intersection de $\mathfrak{V}_v$ et $\mathfrak{R}_r$, c'est-à-dire :

$$\left(\frac{v}{p+q}, r + \frac{p}{p+q}v\right) \ .$$

La rencontre $\mathfrak{e}_v^{r+1}$ *dépend* directement —si elles existent— de $\mathfrak{e}_{v-1}^{r+1}$, $\mathfrak{e}_v^r$, et $\mathfrak{e}_{v+1}^{r-1}$ (ou de la configuration initiale), où les coordonnées (v, r) sont des entiers tels que $0 \leq v \leq n(p+q)$ et $-n.q \leq r$. La rencontre $\mathfrak{e}_v^r$ appartient aussi à $\mathfrak{L}_{r+v}$.

Le terme « rencontre » est utilisé plutôt que « collision » étant donné qu'à cet endroit du diagramme espace-temps il pourrait y avoir une collision, un signal ou rien du tout.

Les rencontres munies de l'ordre basé sur la dépendance forment un ordre bien fondé : l'induction peut y être utilisée.

2.1 Discrétisation formelle

En partant d'une MS normalisée, les états et les règles de transitions sont définis pour assurer les déplacements des signaux lorsqu'ils sont :

— isolés (loin de toute collision),
— proches (avant ou après une collision), et
— sur la même cellule (processus de collision).

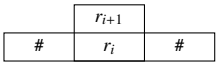
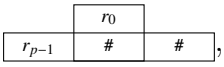
La fin de cette section s'intéresse à la construction de la configuration initiale de l'automate cellulaire, pour que seules les règles de transitions définies aient besoin d'être utilisées.

Pour simplifier la présentation, la convention suivante est utilisée : r et r' représentent tout méta-signal allant vers la droite, l et l' tout méta-signal allant vers la gauche, et z et z' tout méta-signal stationnaires. Les signaux r et r' sont donc de vitesse positive, l et l' de vitesse négative et z et z' de vitesse nulle.

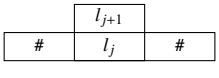
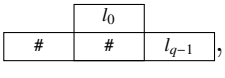
2.1.1 Mouvement des signaux isolés

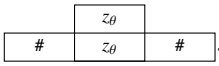
Le déplacement des signaux isolés (c'est-à-dire des signaux à une distance d'au moins deux cellules de tout autre signal) est fait avec quelques états et transitions. Pour avoir des $\mathfrak{A}$-AC-signaux se déplaçant dans le diagramme espace-temps comme dans la Fig. 1.3, les états et transitions suivants sont définis :

— pour chaque méta-signal r : les états r_i avec $0 \leq i < p$ sont définis, ainsi que les transitions suivantes :

$\forall i$ tel que $0 \leq i < p-1$:  et  ,

— pour chaque méta-signal l : les états l_j avec $0 \leq j < q$ sont définis, ainsi que les transitions suivantes :

$\forall j$ tel que $0 \leq j < q-1$:  et  ,

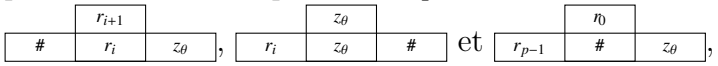
— pour chaque méta-signal z : l'état z_θ est défini, ainsi que  .

Pour les états stationnaires, l'indice est θ et non zéro pour indiquer une phase spécifique : $\theta = \frac{q}{p+q}$. Cela sert à positionner exactement la collision.

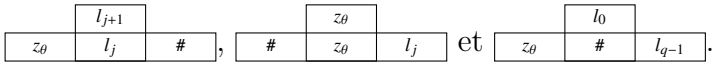
2.1.2 Signaux se rapprochant

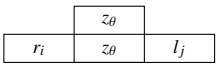
Les $\mathfrak{A}$-AC-signaux proches les uns des autres (à une distance d'une cellule d'un autre signal) nécessitent des transitions spécifiques. Cette situation apparaît quand un $\mathfrak{A}$-AC-signal allant vers la droite (r) ou un allant vers la gauche (l) se rapproche d'un stationaire (z). Pour ces cas, comme illustré dans la Fig. 2.2, les transitions suivantes sont créées :

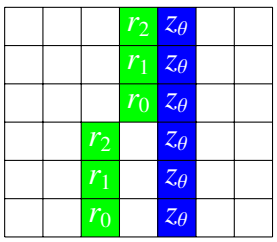
— pour r et z : $\forall i$ tel que $0 \leq i < p-1$:



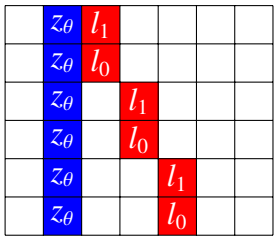
— pour l et z : $\forall j$ tel que $0 \leq j < q-1$:



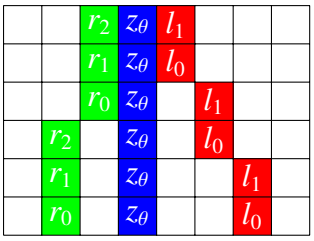
— pour l, z et r : $\forall j$ tel que $0 \leq j < q-1$ et $\forall i$ tel que $0 \leq i < p-1$: 



(a) un signal allant vers la droite et un stationnaire



(b) un signal allant vers la gauche et un stationnaire



(c) les trois vitesses en même temps

FIGURE 2.2 – Exemples de $\mathfrak{A}$-AC-signaux se rapprochant.

2.1.3 Signaux se déplaçant sur la même cellule et gestion des collisions

La phase d'un $\mathfrak{A}$-AC-signal allant vers la droite [1] est la distance entre le coin bas-gauche de la cellule et l'emplacement réel où le signal (continu) entre dans la cellule par le bas.

Quand des signaux entrent dans le même cellule, l'hypothèse suivante sur les phases est supposée :

Hypothèse d'induction 34 *Quand les signaux sont sur la même cellule pour la première fois, les signaux non-stationnaires ont la phase* 0 *et les signaux stationnaires ont la phase* θ.

Quand deux $\mathfrak{A}$-AC-signaux (ou plus) sont sur le point d'entrer en collision, ils sont sur la même cellule avant la collision. À un certain point, cette cellule va changer d'état pour signifier que la collision se produit à cette itération. Pour trouver le moment exact de cette collision, il est nécessaire de revenir aux signaux continus et de le calculer.

1. pour ceux allant vers la gauche, la règle est adaptée en miroir

En utilisant les signaux allant vers la droite et ceux allant vers la gauche, deux droites sont définies :

— $y = px$ représentant la trace d'un signal allant vers la droite ,

— $y = -qx + q$ représentant la trace d'un signal allant vers la gauche.

Résoudre ce système donne :

$$\begin{cases} x = \dfrac{q}{p+q} \\ y = \dfrac{p \cdot q}{p+q} \end{cases} .$$

La valeur de x correspond à la position spatiale de la collision et y à la position temporelle (celle utilisée pour identifier l'itération de la cellule dans laquelle la collision se passe). Ces deux valeurs sont les coordonnées exactes de la collision. Trouver le placement de la collision dans le diagramme espace-temps de l'automate cellulaire revient à obtenir la partie entière de ce point. Cependant, un problème apparaît si une collision peut exister sur la bordure d'une cellule. Cela est impossible, car :

Lemme 35 *Les collisions apparaissent toujours à l'intérieure d'une cellule et jamais sur sa bordure car* $\frac{p \cdot q}{p+q} \notin \mathbb{Z}$ *(et* $\frac{q}{p+q} \notin \mathbb{Z}$*).*

Preuve. Si $q = 1$ ou $p = 1$ alors $p + q = p \cdot q + 1$, donc $\frac{p \cdot q}{p+q} \notin \mathbb{Z}$. Sinon, supposons que $\frac{p \cdot q}{p+q} \in \mathbb{Z}$. Étant donné que $2 < p + q$, $\exists m$ premier et $m \backslash (p + q)$ (m divise $p + q$). Alors $m \backslash pq$. Étant donné que p et q sont premiers entre eux, $m \backslash p$ (ou symétriquement, $m \backslash q$). Étant donné que $m \backslash (p + q)$ et $m \backslash p$ alors $m \backslash q$. Donc $m \backslash p$ et $m \backslash q$. Mais $gcd(p, q) = 1$ et $m > 1$ (car m est premier). C'est impossible, donc $\frac{p \cdot q}{p+q} \notin \mathbb{Z}$. □

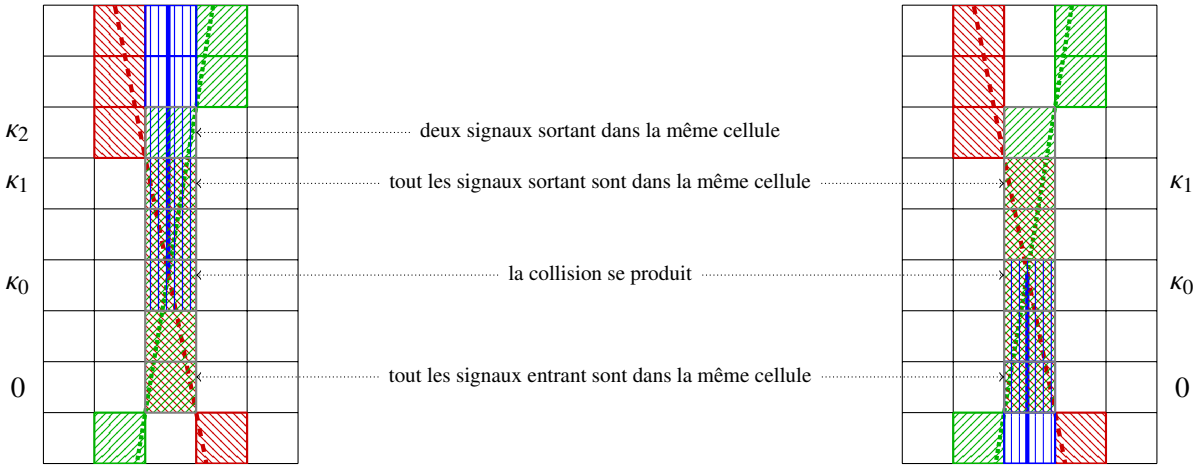
Quand plusieurs signaux sont sur la même cellule, il est nécessaire de définir de nouveaux états et transitions pour qu'ils continuent de se déplacer. C'est aussi pendant cette période qu'ils entrent en collision, donc il faut gérer ce cas. Cette phase nécessite d'identifier le moment exact de la collision (en utilisant la méthode donnée précédemment).

Soit $\rho : \{r, z, l\} \to \{l', z', r'\}$ une règle de collision (tout méta-signal peut être absent, mais au moins deux des r, z, l doivent être présents). Soit $\kappa_0 \in \mathbb{N}$ le moment de la collision. Soient ρ_k les états où deux ou plus signaux sont sur la même cellule. Quand $k < \kappa_0$, ρ_k signifie $(r_k z_\theta l_k)$ où un seul d'entre eux peut être manquant. Quand $\kappa_0 < k$, ρ_k signifie $(l'_k z'_\theta r'_k)$ où plusieurs d'entre eux peuvent manquer.

Soit κ_1 le premier moment où les signaux sur la même cellule se séparent et κ_2 le second. Quand 0 ou 1 signal est émis, κ_1 et κ_2 sont indéfinis. Quand 2 ou 3 signaux sont émis par ρ, κ_1 et κ_2 sont donnés par la Table 2.1. Ces valeurs sont illustrées dans la Fig. 2.3.

Signaux émis	$\{z', r'\}$	$\{l', z'\}$	$\{l', r'\}$	$\{l', z', r'\}$
κ_0	$\left\lfloor \frac{p \cdot q}{p+q} \right\rfloor$			
κ_1	$p - 1$	$q - 1$	$\min(p, q) - 1$	
κ_2	/	/	/	$\max(p, q) - 1$

TABLE 2.1 – Étapes principales d'une collision.



(a) collision émettant 3 signaux (b) collision émettant 2 signaux

FIGURE 2.3 – Illustration de κ_0, κ_1 et κ_2 avec $p = 6$ et $q = 5$.

Sous l'Hyp. 34, il est maintenant possible de créer les transitions qui permettent le mouvement sur la même cellule ainsi que la collision. Les premières transitions à définir sont celles qui permettent aux signaux d'entrer dans la même cellule :

— | # | z_θ | l_{q-1} | (ρ_0) si $\rho^- = \{z, l\}$,

— | r_{p-1} | z_θ | # | (ρ_0) si $\rho^- = \{r, z\}$,

— | r_{p-1} | # | l_{q-1} | (ρ_0) si $\rho^- = \{r, l\}$ et

— | r_{p-1} | z_θ | l_{q-1} | (ρ_0) si $\rho^- = \{r, z, l\}$.

Puis ils se rapprochent jusqu'à la collision exacte :

— $\forall k$ tel que $0 \leq k < \kappa_0$: | # | ρ_k | # | (ρ_{k+1})

À partir de là, il y a plusieurs possibilités :

Si la collision n'émet aucun signal, | # | ρ_{κ_0} | # | (#) est ajoutée[2].

Si la collision émet un seul signal, comme illustré dans la Fig. 2.4(a) :

— si $\rho^+ = \{r'\}$, ajouter : | ρ_{κ_0} | # | # | (r'_0) si $\kappa_0 + 1 = p$, et | # | ρ_{κ_0} | # | (r'_{κ_0+1}) sinon ;

— si $\rho^+ = \{l'\}$, ajouter : | # | # | ρ_{κ_0} | (l'_0) si $\kappa_0 + 1 = q$, et | # | ρ_{κ_0} | # | (l'_{κ_0+1}) sinon ;

— si $\rho^+ = \{z'\}$, ajouter : | # | ρ_{κ_0} | # | (z'_θ).

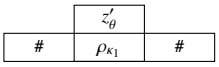
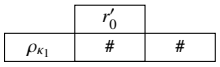
Les transitions définies précédemment s'appliquent à partir de là.

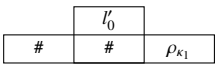
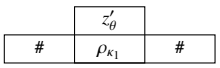
Si la collision émet plus qu'un signal, un ensemble de transitions différent est nécessaire pour que les signaux progressent sur la même cellule jusqu'à séparation :

— $\forall k$ tel que $\kappa_0 \leq k < \kappa_1$: | # | ρ_k | # | (ρ_{k+1}).

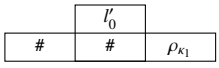
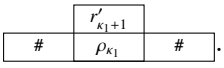
Si la collision émet deux signaux, comme illustré dans la Fig. 2.4(b) :

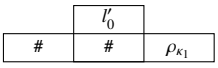
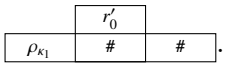
2. déjà couvert par le résultat par défaut de ce chapitre : #

— si $\rho^+ = \{z', r'\}$, ajouter :  et  ;

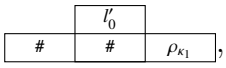
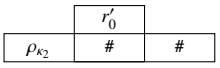
— si $\rho^+ = \{l', z'\}$, ajouter :  et  ;

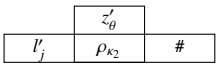
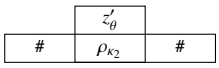
— si $\rho^+ = \{l', r'\}$,

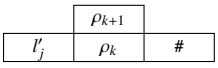
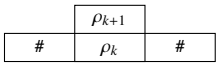
  — si $q < p$ (le cas $p < q$ est symétrique), ajouter :  et  .

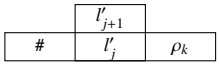
  — si $q = p = 1$, ajouter :  et  .

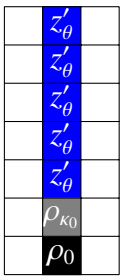
Si la collision émet trois signaux, comme illustré dans la Fig. 2.4(c), et dans le cas $q \leq p$ (le reste est symétrique), ajouter :

—  , et  ;

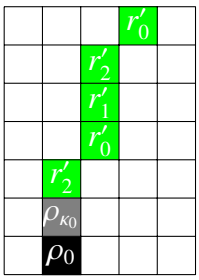
— soit $j = \kappa_2 - \kappa_1 - 1$,  si $0 < j < q$, et  sinon ;

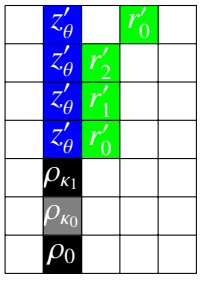
— pour $\kappa_1 < k < \kappa_2$:  pour $j = k - \kappa_1 - 1$ et $j < q$ et  pour $\kappa_1 + q < k$;

— pour $0 \leq j < q - 1$:  pour $k = j + \kappa_1 + 1$ et $k \leq \kappa_2$.

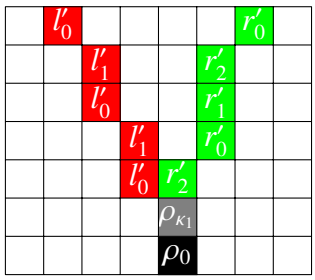


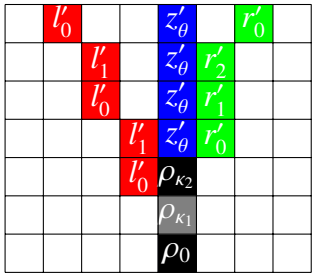
(a) collisions émettant un $\mathfrak{A}$-AC-signal





(b) collisions émettant deux $\mathfrak{A}$-AC-signaux ($\kappa_0 = \kappa_1$ sur la droite)





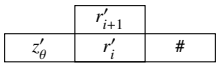
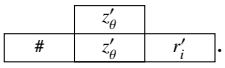
(c) collision émettant trois $\mathfrak{A}$-AC-signaux (ici $\kappa_0 = \kappa_1$)

FIGURE 2.4 – Collisions et $\mathfrak{A}$-AC-signaux s'éloignant les uns des autres avec $p = 3$ et $q = 2$.

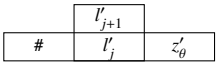
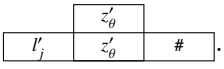
2.1.4 Signaux s'éloignant les uns des autres

Après quelques itérations sur la même cellule puis collision, les $\mathfrak{A}$-AC-signaux vont dans des directions différentes, en s'éloignant des autres. Cela arrive après chaque collision émettant au moins deux $\mathfrak{A}$-AC-signaux, qui sont soit un r' et un z', soit un l' et un z', soit les trois (s'il n'y a que r' et l', ils sont déjà suffisamment éloignés). Cela est géré par les transitions suivantes :

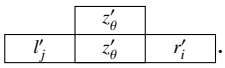
Pour r' et z' :

$\forall i$ tel que $0 \leq i < p - 1$:  et  .

Pour l' et z' :

$\forall j$ tel que $0 \leq j < q - 1$:  et  .

Pour z' entre r' et l' : $\forall i, j$ tels que $0 \leq j < q-1$, $0 \leq i < p-1$ et $i+p = j+q$:



.

Toutes ces transitions sont illustrées dans la Fig. 2.4.

2.1.5 Configurations initiales et échelle de discrétisation

Une échelle assure que le maillage continu est discrétisé à la fois spatialement et temporellement : l'espace entre les signaux stationnaires doit être entier, de même que les moments des collisions. Par la définition des rencontres (Déf. 33), tout multiple (positif) de $p+q$ est suffisant. Ce multiple doit être suffisamment grand pour rendre les signaux visibles.

Définition 36 Soit $\delta = 3(p+q)$ *l'échelle de discrétisation*. Soit $\mathfrak{c}$ la configuration initiale (normalisée) de la machine à signaux (normalisée) à discrétiser. Alors la AC-configuration initiale pour simuler $\mathfrak{c}$, c_0, est définie par :

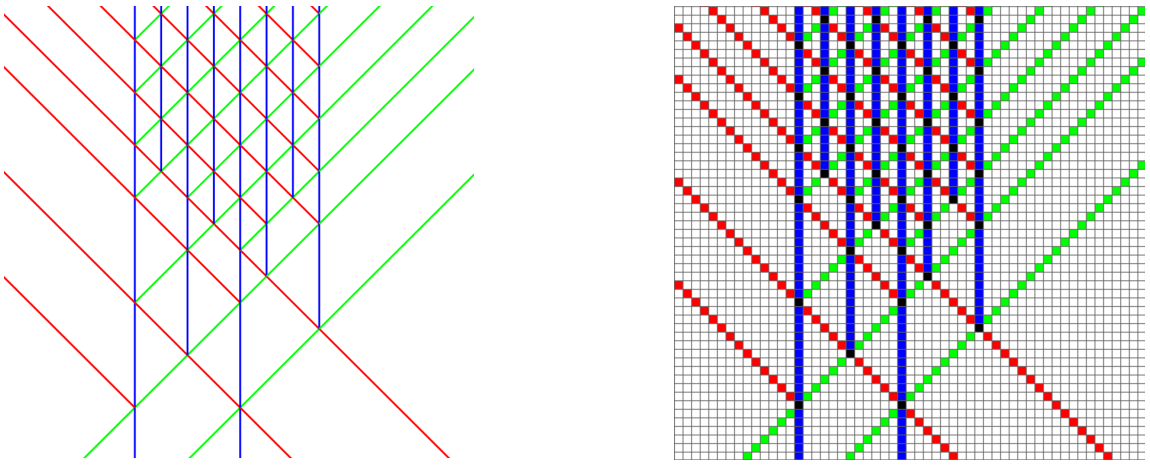
$$\forall i \in \mathbb{Z},\ c_0(i) = \psi(\mathfrak{c}(i/\delta)) \ .$$

où $\psi(\oslash) = \#$, $\psi(z) = z_\theta$ pour tout méta-signal stationnaire, et $\psi(\mu) = \mu_0$ pour tout autre méta-signal.

Toutes les positions non-$\oslash$ sont dans $\mathbb{Z} \times \delta$ de sorte que toute l'information dans $\mathfrak{c}$ est conservée.

Toutes les cellules ne sont pas dans la même phase et si elles étaient transformées en signaux continus, ils ne termineraient pas dans la même position. Ce n'est pas un problème car l'espace est discret et toutes les transitions sont faites pour gérer ce cas exactement. Cette construction assure que l'Hyp. 34 est vérifiée dès la première itération.

Avec l'application de cette échelle, l'objectif est de mettre les $\mathfrak{A}$-AC-signaux sur un maillage discret, comme illustré dans la Fig. 2.5(b). Dans cette figure, le maillage complet est engendré car chaque collision émet un signal de chaque vitesse possible.



(a) cas continu (b) cas discret

FIGURE 2.5 – Formation d'un maillage au sein d'un diagramme espace-temps.

2.2 Correction

Dans cette section, il est prouvé par induction que le LDAG engendré depuis la configuration initiale de la machine à signaux est le même que celui engendré par l'automate cellulaire simulant avec les même positions des collisions, à un décalage linéaire près. Cela est fait en prouvant tout d'abord que l'évolution simulée est aussi contrainte dans un maillage. Puis à partir de ce maillage, le LDAG est récupéré. Seul le cas $q \leq p$ est développé ici, le reste suit par symétrie.

2.2.1 Maillage discret

C'est l'équivalent discret du maillage continu, comme illustré dans la Fig. 2.6.

Définition 37 ((δ, p, q, n)-AC-Maillage) Soient δ, p, q, et n des entier positifs avec p et q premiers entre eux et δ un multiple de $q + p$. Le (δ, p, q, n)-AC-Maillage correspond à l'union des ensembles de $\mathbb{Z} \times \mathbb{N}$ suivants :

— $V_v = \left\{\frac{\delta}{q+p}v\right\} \times \mathbb{N}$ avec $v \in \{0, 1, 2, \ldots, n(q + p)\}$,

— $L_l = \left\{ (x, t) \middle| 0 \leq t \vee x \leq n\delta \vee x = \left\lceil \frac{l\delta - t}{q} \right\rceil \right\}$ avec $l \in \mathbb{N}$, et

— $R_r = \left\{ (x, t) \middle| 0 \leq t \vee 0 \leq x \vee x = \left\lfloor \frac{t - r\delta}{p} \right\rfloor \right\}$ avec $r \in \{(-n.p), \ldots, -1, 0, 1, 2, \ldots\}$.

Chaque ensemble correspond à la trace d'un AC-signal (pas nécessairement provenant de la configuration initiale). Ils peuvent se superposer et être sur la même cellule pendant plusieurs itérations consécutives (comme détaillé par la suite).

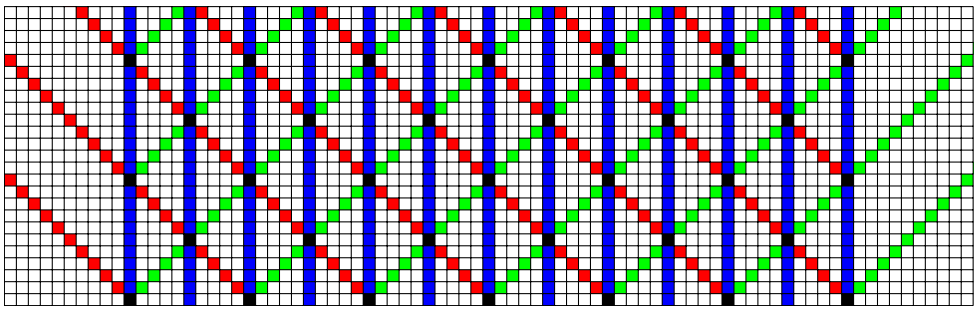


FIGURE 2.6 – Le $(10, 1, 1, 6)$-AC-Maillage.

Avec une hypothèse sur la localisation précise du site κ_0, les collisions illustrées dans la Fig. 2.7 sont prouvées incluses dans le maillage. Cette hypothèse est que ρ_{κ_0} correspond à la discrétisation d'une MS-rencontre $\mathfrak{e}^r_v$.

Définition 38 (AC-rencontre (discrète)) Dans le (δ, p, q, n)-(AC-)maillage, la *AC-rencontre* de coordonnées (v, r), e^r_v, avec $\delta = 3(p + q)$ correspond à :

$$\left(\left\lfloor \frac{\delta}{p+q}v + \frac{q}{p+q} \right\rfloor, \left\lfloor r\delta + \frac{p\delta}{p+q}v + \frac{pq}{p+q} \right\rfloor \right) = (3v, 3pv + r\delta + \kappa_0) \quad .$$

Où les coordonnées (v, r) sont entières et telles que $0 \leq v \leq n(p+q)$ et $-n.q \leq r$.

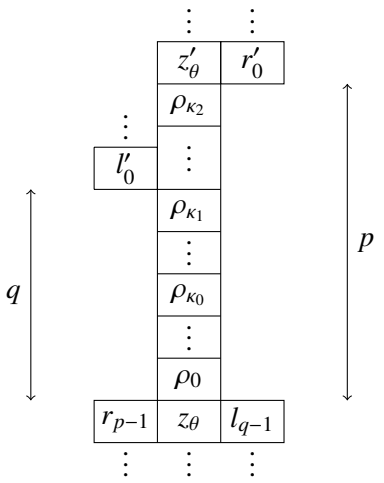


FIGURE 2.7 – Collision générique entière $\rho = \{r, z, l\} \to \{l', z', r'\}$.

Le dernier terme de la formule ci-dessus provient des phases des signaux initiaux (θ pour les signaux stationnaires et 0, mais partant de la droite pour les signaux allant vers la gauche). Au bas de la Fig. 2.7, ρ_0 a les coordonnées : $(3v, 3pv + r\delta)$, tel que :

Fait 39 *À l'AC-rencontre e^r_v :*
- *ρ_0 appartient à V_v, R_r et L_{v+r},*
- *ρ_{κ_2} appartient à R_r, et*
- *ρ_{κ_1} appartient à L_{v+r}.*

Cela fixe les trois AC-signaux sur une période. De cela on peut déduire que la collision entière localisée ici est dans le (δ, p, q, n)-AC-Maillage.

Cela reste vrai si il n'y a pas de collision (zéro ou un AC-signal présent).

Il reste à prouver que toute AC-collision est localisée à une certaine e^r_v. Cela est fait par induction :
- les AC-signaux dans la configuration initiale sont dans les (δ, p, q, n)-AC-Maillage avec les bonnes phases et
- si les AC-signaux entre en collision et sont dans le maillage, il en va de même pour les AC-signaux résultants.

Il n'y a pas d'autres site non-quiescent à prendre en compte. Cela prouve donc que :

Théorème 40 *Tout site non-quiescent de l'exécution depuis la AC-configuration simulante est localisé dans le (δ, p, q, n)-AC-Maillage.*

2.2.2 Engendrer le LDAG pour l'AC et l'identifier à celui de la MS

Il reste à définir la relation de représentation $\mathfrak{A}$-$\mathcal{A}$ pour obtenir le LDAG (il est unique ici). Cela se fait directement à partir de la construction. Soient l, z et r des méta-signaux allant vers la gauche, stationnaires et allant vers la droite. Ce qui suit est défini :

$$\forall i,\ 0{\leq}i{<}q,\ l\,\mathcal{R}\,l_i\ ,\quad z\,\mathcal{R}\,z_\theta\ ,\quad \text{et}\quad \forall i,\ 0{\leq}i{<}p,\ r\,\mathcal{R}\,r_i\ .$$

Soit $\rho = \{r, z, l\} \to \{l', z', r'\}$ une règle.

$$\begin{array}{lll} \forall i, 0 \leq i < \kappa_0, & r\,\mathcal{R}\,\rho_i, & z\,\mathcal{R}\,\rho_i, \text{ et } l\,\mathcal{R}\,\rho_i, \\ & \rho\,\mathcal{R}\,\rho_{\kappa_0}, & \\ \forall i, \kappa_0 < i \leq \kappa_1, & r'\,\mathcal{R}\,\rho_i, & z'\,\mathcal{R}\,\rho_i, \text{ et } l'\,\mathcal{R}\,\rho_i, \\ \forall i, \kappa_1 < i \leq \kappa_2, & r'\,\mathcal{R}\,\rho_i \text{ et } z'\,\mathcal{R}\,\rho_i. & \end{array}$$

Comme illustré dans la Fig. 2.7, la relation de méta-signaux/AC-signaux s'étend en dehors de la collision depuis chaque côté de sorte que les AC-signaux se connectent à la collision. Une induction structurelle sur le maillage prouve que les LDAG des MS et AC sont identiques. De plus, les localisations des intersections dans un maillage peuvent être linéairement calculées à partir de celle de l'autre maillage, de telle manière que :

Théorème 41 *Avec l'AC engendré, toute MS-configuration est associée à une AC-configuration qui la simule avec une conservation linéaire de l'emplacement de la collision.*

Maximalité pour la discrétisation exacte. Cette discrétisation, parfaitement fonctionnelle et exacte, souffre cependant de ne fonctionner que sur les machines à signaux rationnelles à trois vitesses. En effet, si le nombre de vitesses possibles passe à quatre (ou plus), il devient dès lors possible de produire des structures qui ne s'inscrivent pas sur un maillage discret (un exemple de ce problème est traité dans le Chapitre 6). L'autre condition est la rationalité des machines, car si l'on accepte qu'une vitesse ou un positionnement dans la configuration initiale soit irrationnelle, il devient alors possible de produire des structures fractales avec accumulation. Ces deux constructions sont montrées dans [BCDL$^+$13]. Cette classe de machines à signaux est donc maximale pour la discrétisation exacte, puisque l'étendre même légèrement met fin à son exactitude systématique.

Une implantation de cette discrétisation a été faite, et plus d'informations à ce sujet peuvent être trouvées dans l'annexe A.

Chapitre 3

Approche brutale de la discrétisation

Ce chapitre propose de discrétiser des machines à signaux rationnelles sans autres limitations, en transformant directement les méta-signaux et règles de collisions en états et règles de transition. Cette approche ne garantit rien sur la qualité de la discrétisation : elle est incorrecte sur des comportements « continus » ou sur de trop grande quantités d'information par cellules ou encore sur des signaux « mal positionnés ». La méthode de discrétisation proposée ici réduit l'information à une quantité manipulable dans un cadre expérimental. Cette information se trouve dans le « bas » des cellules de l'automate cellulaire, qui sont chacune assimilées à une configuration de machine à signaux. Tout au long de ce chapitre, il est considéré que l'automate cellulaire produit a pour but de simuler une machine à signaux $\mathfrak{A} = (M, S, R)$.

3.1 Constructions des états

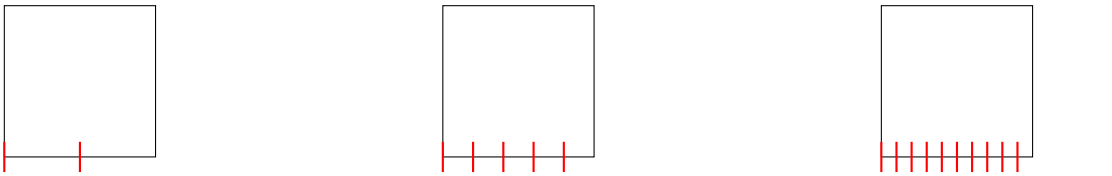
Dans ce chapitre, le système de discrétisation ne dispose d'aucune information structurelle sur la machine à signaux d'origine. Le nombre d'états possibles est donc potentiellement infini. Il est alors nécessaire de trouver une solution pour imposer un nombre fini d'états. Ici, cela s'est porté sur l'introduction d'un *pas de discrétisation* qui limite à un nombre fini les positions possibles de signaux continus dans une cellule. Cela peut être vu comme définissant des emplacements situés à intervalles réguliers dans les entrées (le bas) des cellules. Avec cela, le nombre de signaux par cellule devient alors fini, ce qui permet de n'engendrer qu'un nombre fini d'états. Formellement, ce pas de discrétisation se définit comme suit :

Définition 42 (Pas de de discrétisation) Un *pas de discrétisation* τ d'un ACS (voir la Définition 24) est un entier tel que : $\forall m \in M, \tau \cdot S(m) \in \mathbb{N}$.

Cette définition assure que tout signal continu partant d'un des emplacements sur un site sera toujours sur une position possible d'un site situé après une étape temporelle (s'il n'est pas entré en collision).

La Figure 3.1 donne trois exemples de pas de discrétisation et de positions alors possibles pour les signaux.

Avec un pas de discrétisation, il devient possible de donner une définition formelle d'un état d'un ACS. Cependant, pour des raisons de simplicité d'expression, il faut d'abord définir un ensemble d'éléments particuliers :



(a) emplacements possibles pour $\tau = 2$ (b) emplacements possibles pour $\tau = 5$ (c) emplacements possibles pour $\tau = 10$

FIGURE 3.1 – Exemples de pas de discrétisation dans un site.

Définition 43 (Ensemble des parties à vitesses distinctes) Soit $\mathcal{P}(M)$ l'ensemble des parties de M. Un ensemble de méta-signaux $\mathcal{X}$ appartient à l'*ensemble des parties à vitesses distinctes* $\mathcal{P}_{\nparallel}$ si : $\mathcal{X} \in \mathcal{P}(\mathcal{M})$ et $\forall x \in \mathcal{X}, \nexists y$ tel que $S(x) = S(y)$ et $y \neq x$.

Avec ces définitions, l'ensemble Q des états d'un ACS est vu comme un ensemble de fonctions qui associent à chaque entier compris entre 0 et $\tau - 1$ un ensemble de méta-signaux. Formellement :

Propriété 44 (États d'un ACS) *Soit $\mathcal{A}$ un automate cellulaire et Q l'ensemble de ses états, alors $\mathcal{A}$ est un ACS si $\forall s \in Q$ et $\forall \mu \in M$, $\exists g : [0, \tau - 1]_{\mathbb{N}} \rightarrow \mathcal{P}_{\nparallel}$ telle que $\mu \in g(0) \cup g(1) \cup ... \cup g(\tau - 1) \Rightarrow \mu \mathcal{R} s$ avec $\mathcal{R}$ une relation de représentation telle que définit dans la Définition 14 et donnée par la suite. La fonction g est appelée* fonction de pas de discrétisation.

Le point d'intérêt est maintenant de proposer une définition et une méthode d'engendrement des sous-ensembles de Q, c'est-à-dire l'ensemble des états représentant des méta-signaux sans collisions (exprimé dans la Définition 16 et noté Q_E), l'ensemble des états composites (décrit dans la Définition 17 et noté Q_C), l'ensemble des états collision (donné dans la Définition 18 et noté Q_K) et l'ensemble des états complexes (définit dans la Définition 19 et noté Q_X).

3.1.1 Ensemble des états appartenant à un AC-signal

Dans cette sous-section, l'ensemble des états à l'étude est celui des AC-signaux représentant exactement un seul signal. Il s'agit donc de l'ensemble d'états Q_E^0, et non pas Q_E. Tout les états dans Q_E mais pas dans Q_E^0 seront traités par la suite comme des états composites.

Les états AC-signaux engendrés ici sont ceux dont la fonction de pas de discrétisation associe exactement un méta-signal à exactement une des positions et $\emptyset$ à toutes les autres. Plus formellement :

Définition 45 (État AC-signal (appartenant à Q_E^0)) Soit s un état. Alors, $s \in Q_E^0 \Leftrightarrow \exists! i \in [0, \tau - 1]_{\mathbb{N}}$ tel que $g(i) = \{\mu\}$ avec g la fonction de pas de discrétisation associée à s et μ un méta-signal. De plus, $\forall j \neq i$, $g(j) = \emptyset$.

Si e est un état appartenant à un AC-signal, et μ le méta-signal associé au signal dans cet état, alors $\mu \mathcal{R} e$.

Avec cette définition, un AC-signal est donc un ensemble de sites représentant exactement un seul méta-signal, et dans ce cadre exactement un seul signal. Il existe un état pour chaque position déterminée par le pas de discrétisation. Engendrer ces états est donc très simple : il suffit de créer τ états, numérotés de 0 à $\tau - 1$.

Pour pouvoir les relier et ainsi permettre la propagation de l'information, il faut définir des règles de transition. Ces règles décalent d'une cellule vers la gauche ou la droite l'AC-signal lorsque le signal continu qu'il représente dépasse de la gauche ou de la droite de la cellule. Plus formellement :

Définition 46 (Règles de transition d'un AC-signal) Soit μ un méta-signal tel que $S(\mu) = \frac{p}{q}$ avec p et q minimaux et premier entre eux et soit S l'ensemble des états représentant une seule itération d'un signal de type μ.

Soient $i \in [0, \tau - 1]_{\mathbb{N}}$ et $j = i + \tau \cdot \frac{p}{q}$.

Si $0 \leq j < \tau$, alors $f(\#, S_i, \#) = S_j$.

Si $-\tau \leq j < 0$ et soit $j' = j + \tau$, alors $f(\#, \#, S_i) = S_{j'}$.

Si $\tau \leq j < 2\tau$ et soit $j' = j - \tau$, alors $f(S_i, \#, \#) = S_{j'}$.

Avec f la fonction de transition locale de l'automate cellulaire.

Avec ces définitions, les AC-signaux sont engendrés avec leurs états et leurs règles pour leur permettre de se déplacer isolés dans le vide. Il devient alors nécessaire d'engendrer et de « déplacer » les autres états.

3.1.2 Ensemble des autres états

Engendrer tous les autres états revient à produire tous les sous-ensembles de méta-signaux possibles, dans toutes les positions données par le pas de discrétisation. Il faut donc lister toutes les possibilités d'ensembles de méta-signaux et vérifier si une collision s'y produit. Ainsi, pour chacune des τ positions possibles dans une cellule, il peut s'y trouver n'importe quel élément de l'ensemble des parties corrigé. Tout les ensembles d'états y sont possibles, et il suffit d'en retirer ceux qui sont des méta-signaux uniques pour obtenir tout les états restant. Ils sont en *très grand nombre* : il y en a $|\mathcal{P}(M)|^\tau$ au pire cas.

Pour donner un exemple du nombre de ces états, une machine à 5 méta-signaux de vitesses différentes ($|\mathcal{P}_{\nparallel}(M)| = (2^5) = 32$) et un pas de discrétisation de 10, engendrera donc 32^{10} états dans le pire cas au total, dont il faut retirer le nombre négligeable des AC-signaux isolés. Il s'agit ici du nombre maximal d'états possibles, qu'il faudrait tous engendrer et tester. Si dans un cadre théorique il est tout à fait possible de considérer avoir un tel nombre d'états, il est bien trop grand pour un travail expérimental. De plus, il faut produire les règles pour chacun de ces états, chacune composée d'un triplet d'états. Puisque celui-ci est composé de n'importe lesquels, il y a donc $(|\mathcal{P}_{\nparallel}(M)|^\tau)^3$ règles à définir soit, dans l'exemple donné, $(32^{10})^3$ règles. Devant de tels nombres, il est impossible d'espérer produire une discrétisation fonctionnelle engendrant tous ces états et règles. C'est pourquoi il est nécessaire de s'intéresser à une approche qui limite ces nombres, et c'est ce que propose la section suivante.

Malgré cela, cette approche de création de l'ensemble de ces états a été tentée dans une implantation qui considère un nombre de signaux par cellules limités à trois. Il en va de même pour les collisions qui sont limitées à une seule par cellule. Cette implantation,

au demeurant incomplète, permet d'obtenir quelques résultats intéressants. Celle-ci n'est pas présentée dans ce mémoire.

3.2 Création des règles et états par utilisation de la machine à signaux

Le trop grand nombre d'états et de règles qu'il faudrait engendrer pour pouvoir obtenir un automate cellulaire simulant « absolu » (pouvant simuler ou approximer tous les cas possibles), amène la nécessité de trouver une solution permettant de ne créer qu'un sous-ensemble « suffisant » de ces états. Dans le Chapitre 2, l'outil de discrétisation possède une information sur la structure de la machine à signaux (le maillage), qui permet de fortement limiter le nombre d'états et de règles. Dans cette section, la solution choisie est d'explorer ces ensembles d'états et de règles et de ne créer que ceux qui sont nécessaires à l'automate cellulaire sur une configuration initiale précise. La discrétisation débute avec une configuration initiale discrète utilisant des états AC-signaux et rien d'autre. L'AC est alors exécuté et à chaque apparition d'une nouvelle règle inconnue, la machine à signaux d'origine est utilisée pour définir le résultat. Pour ce faire, les cellules connaissent la configuration continue qu'elles contiennent, et la règle devient alors une configuration de machine à signaux qui sera exécutée puis observée pour obtenir l'état résultat. Il devient donc nécessaire de définir la configuration d'une cellule et d'une règle de transition.

Avec ces définitions, il est alors possible d'appliquer une méthode de discrétisation brutale. Celle-ci a été implantée en Java par Loïc Labourbe, étudiant de Licence 3 Informatique à l'Université d'Orléans, au cours d'un stage co-encadré par l'auteur et Jérôme Durand-Lose dans le cadre de cette thèse. La méthode se déroule en quatre étapes :

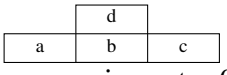
— elle reconstitue la configuration continue d'une règle de transition,
— exécute la machine à signaux sur celle-ci,
— extrait le résultat, et
— créée un nouvel état et une nouvelle règle si besoin.

3.2.1 Définitions

Pour pouvoir créer une configuration de machine à signaux à partir de cellules, il convient de définir comment une telle information pourrait être retrouvée. Cela se fait à travers la définition de la *configuration* d'une cellule (Figure 3.2).

Définition 47 La *configuration* d'une cellule est l'ensemble des signaux présents en entrée de la cellule (au temps 0), ramenés entre 0 (inclus) et 1 (exclus).

Ces définitions s'étendent naturellement aux règles de transition, pour y définir la notion de configuration (Figure 3.3).

Définition 48 La *configuration* d'une règle de transition 

est l'agglomération des configurations des cellules contenant les états a, b et c, comprise entre 0 et 3.

Munis de ces définitions, il est désormais possible de définir et d'utiliser les états engendrés par simulation.

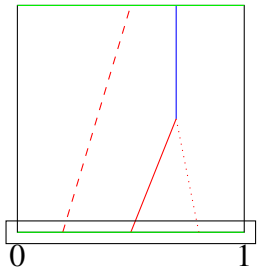


FIGURE 3.2 – Configuration d'une cellule.

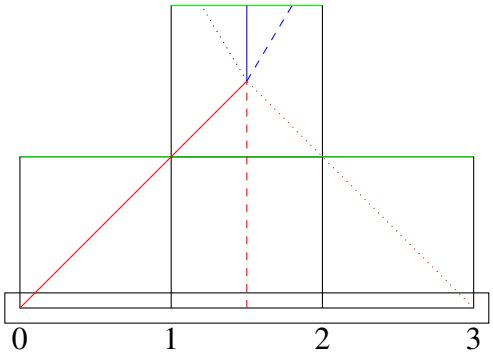
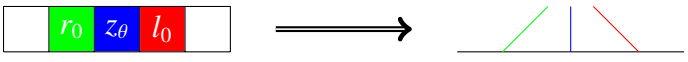


FIGURE 3.3 – Configuration d'une règle de transition.

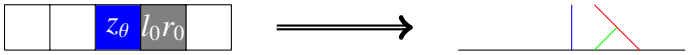
Définition 49 Un *état engendré par simulation* est un état de l'automate cellulaire qui contient l'information de sa configuration, sous la forme d'un ensemble d'associations $< x, \mu >$ avec $x \in [0, 1[_{\mathbb{Q}}$ et $\mu \in M$. Ces positions sont celles données par le pas de discrétisation.

3.2.2 Reconstitution de la configuration continue

Durant l'évolution de l'automate cellulaire, de nouvelles règles et états sont engendrées à la volée. Lorsque cela arrive, leurs configurations sont transformées en configurations de machine à signaux. Pour ce faire, il suffit de récupérer la configuration de chaque cellule et de les agglomérer pour créer la configuration initiale. Cette configuration étant enregistrée dans chaque état, cela se fait aisément. La Fig. 3.4 illustre ce processus.



(a) reconstitution n'impliquant que des AC-signaux isolés



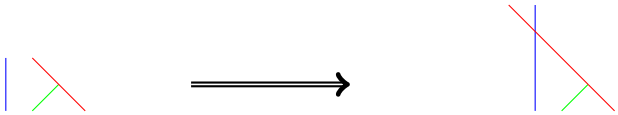
(b) reconstitution impliquant des états calculés précédemment

FIGURE 3.4 – Reconstitution de configurations continues à partir de trois cellules.

3.2.3 Exécution sur la machine à signaux

Une fois la configuration continue reconstituée, elle est exécutée sur la machine à signaux en cours de discrétisation. Elle calcule sur un temps correspondant à la hauteur d'une cellule de l'automate cellulaire, ce qui correspond à une unité de temps dans la machine à signaux. La Figure 3.5 illustre une exécution de ce type (qui n'est rien de plus que celle de la machine à signaux correspondante).

(a) évolution de la configuration extraite de la Fig. 3.4(a)



(b) évolution de la configuration extraite de la Fig. 3.4(b)

FIGURE 3.5 – Exécutions des configurations de la Figure 3.4.

Cette reconstitution peut être associée à une projection d'une cellule vers une configuration de machine à signaux. De plus, lors de cette phase, un comportement purement continu peut apparaître et bloquer le processus d'exécution de la machine à signaux. Ce cas de figure est le point faible de cette méthode, causant l'échec du processus, puisque le système ne connaît pas de technique d'approximation. Si aucun problème n'est rencontré, la configuration résultat est alors extraite.

Le problème que cette phase peut poser, est l'apparition d'un nombre infini de collisions (avec une accumulation) qui rendrait donc impossible la discrétisation. Pour le résoudre, chaque discrétisation est associée à un nombre maximal de collisions par cellule, et si ce nombre est dépassé, le résultat d'une règle causant ce problème est un état fixe indiquant l'erreur.

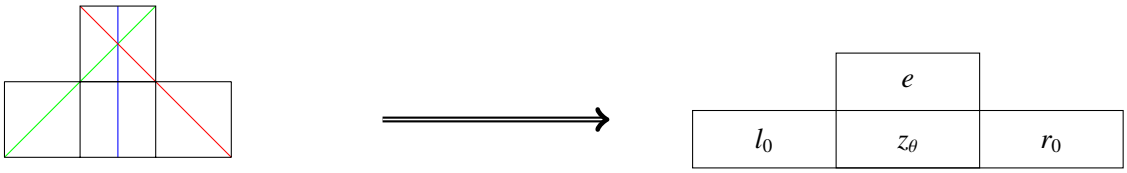
3.2.4 Extraction du résultat et nouvel état

À partir du calcul précédent, la configuration de la cellule résultat de la règle est extraite. Le procédé d'approximation de cette discrétisation apparaît à ce moment : les différents signaux résultats sont positionnés sur la graduation discrète (le pas de discrétisation) de la cellule. Ils peuvent être déplacés ou supprimés s'ils sont dans des positions incorrectes. Par exemple, dans la Figure 3.6, les signaux rouge, orange, rose et marron sont déplacés vers les emplacements disponibles les plus proches, alors qu'un des signaux marron est supprimé.

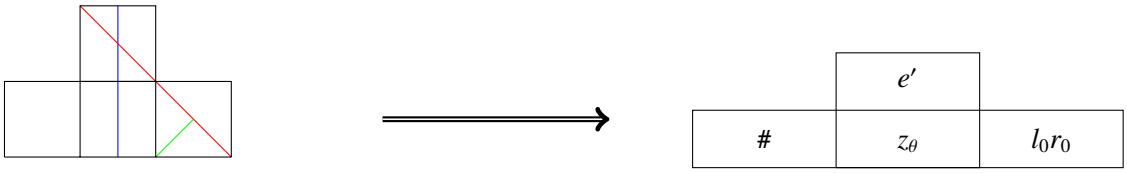
Il existe d'autres règles d'approximation du résultat de règles de transition, dans des cas parfois plus complexes. Cependant, la logique reste la même : ramener autant de signaux que possibles sur les positions disponibles dans les cellules, en perdant le minimum d'information. Ces règles d'approximation sont données dans [Lab17].

FIGURE 3.6 – Approximation d'un résultat d'une règle de transition à 5 emplacements possibles.

Après ce processus d'approximation, un nouvel état enregistrant la configuration obtenue est créé et la règle de transition est ajoutée à l'automate cellulaire. La discrétisation reprend alors son cours jusqu'à l'apparition d'une autre règle au résultat inconnu, ou jusqu'à atteindre la condition d'arrêt de l'automate cellulaire. La Figure 3.7 illustre le processus d'extraction.



(a) extraction du résultat de la Fig. 3.5(a) et règle de transition correspondante



(b) extraction du résultat de la Fig. 3.5(b) et règle de transition correspondante

FIGURE 3.7 – Extraction des règles de transition depuis les résultats de la Figure 3.5

Après cette étape, la relation de représentation $\mathcal{R}$ est complétée : le nouvel état obtenu représente désormais l'ensemble des signaux de sa configuration initiale.

3.2.5 Résultat

Cette méthode est plus performante que la précédente puisqu'elle n'est pas limitée aux machines rationnelles à trois vitesses. De plus, elle permet de n'engendrer qu'un faible nombre d'états et de règles en ne les créant que lorsqu'ils sont nécessaires permettant donc d'être utilisable dans un cadre pratique. Cependant, elle ne propose pas de solution pour traiter les comportements purement continus : son seul travail d'approximation est propre au processus et n'agit aucunement sur le traitement de ces structures particulières, qui font alors apparaître des états « erreur » indiquant un trop grand nombre de collisions dans une cellule. La gestion de cette erreur est alors entre les mains de l'utilisateur (qui peut soit stopper la discrétisation soit suivre une règle définie). Elle rend accessible des structures plus complexes que celles des machines rationnelles à trois vitesses avec par exemple la construction de la Figure 3.8 qui nécessite plus de trois vitesses (cinq dans ce cas).

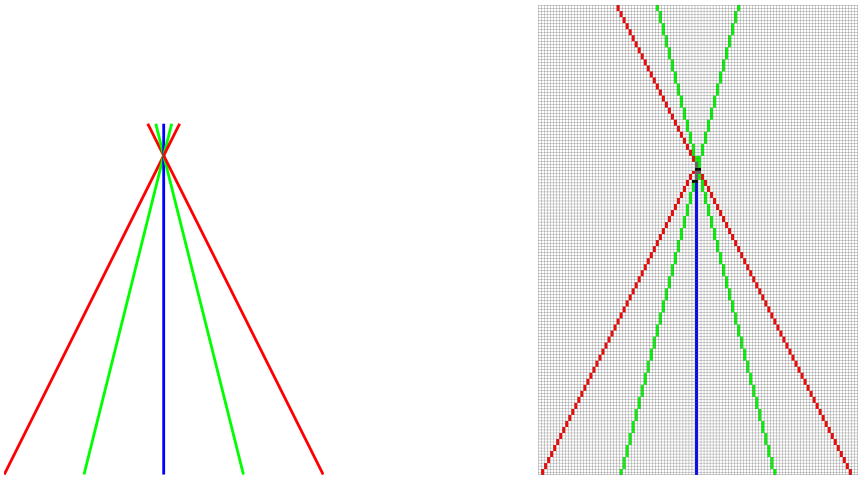
(a) la machine à signaux d'origine

(b) l'automate cellulaire simulant

FIGURE 3.8 – Un résultat de la discrétisation par simulation de la machine à signaux.

Cependant, cette méthode échoue parfois sur des discrétisations qui peuvent se faire de manière exacte. En effet, le système d'approximation embarqué qui permet de limiter le nombre d'états possibles à une valeur finie entraîne des modifications de positionnement des signaux qui peuvent être la source d'erreurs. De fait, décaler un signal continu sans décaler les autres peut modifier la dynamique continue d'origine, et donc faire cette opération lors du procédé de discrétisation engendre évidemment des erreurs. Il est même possible de produire des diagrammes avec peu de signaux qui exhibent cette erreur. C'est par exemple le cas de la Figure 3.9. Par conséquent, il est nécessaire de proposer une méthode qui va assurer de pouvoir éviter ces problèmes de décalage, en proposant une structuration du calcul qui les empêche.

Le problème dans cette figure provient du processus d'approximation. En effet, dans la machine à signaux, les signaux continus ne forment qu'une seule collision (tout les cinq) puis tout les signaux émis s'échappent. Dans la discrétisation, les signaux étant approximés sur des positions incorrectes, *ils déclenchent d'autres collisions*. Si l'une (ou plusieurs) d'entre elles déclenchent la destruction de signaux, alors ces signaux sont détruits parce qu'une collision qui ne devrait pas avoir lieu se déclenche. Il y a ici une erreur d'approximation, entraînant une discrétisation incorrecte.



(a) la machine à signaux d'origine

(b) l'automate cellulaire simulant

FIGURE 3.9 – Une erreur de la discrétisation par simulation de la machine à signaux.

Deuxième partie

Approche avancée : modularité

Introduction à la modularité

La notion de modularité naît de deux volontés :

— trouver une nouvelle méthode de description des dynamiques d'une machine à signaux et

— proposer une solution pour construire un diagramme espace-temps à partir de « briques » élémentaires, s'assemblant comme un puzzle.

Dans le cadre de cette thèse, l'objectif est de permettre d'utiliser ces deux éléments pour proposer une méthode de discrétisation automatique s'appuyant sur des données externes offrants des solutions aux problèmes d'approximations.

L'idée de la modularité est de découper un diagramme espace-temps engendré par une machine à signaux en zones d'intérêt appelées « modules ». Ces modules sont rectangulaires, contiennent au moins une collision (et donc au moins deux signaux) et possèdent un certains nombres de signaux entrants et sortants. Ce découpage se fait dans le but de chercher une nouvelle méthode de description de la dynamique qui abstrait les collisions pour conserver des « zones de calcul ». Après une observation rapide de tels découpages, un concept apparaît : au sein de ces modules, certains contiennent le même type d'information, c'est-à-dire les mêmes signaux qui se rencontrent dans les mêmes collisions et rentrent et sortent du module dans le même ordre. La seule différence entre ces modules est généralement une question d'échelle : certains sont plus grands, d'autres sont plus petits mais leur *sémantique* reste la même. Cette apparition d'information répétée est particulièrement visible dans des cas extrêmes, et notamment dans les structures fractales que sont les accumulations. Un exemple d'un tel découpage d'une multitude de modules sémantiquement identiques peut être trouvé dans la Figure 3.10. Dès lors, la notion de méta-module apparaît : ceux-ci sont pour les modules ce que sont les méta-signaux pour les signaux, la description de leur nature, sans précision de leur position spatio-temporelle ni de leur taille.

Ce couple méta-modules/modules fait émerger une nouvelle forme d'expression de la dynamique d'une machine à signaux : celle-ci n'est plus exprimée à partir des collisions mais à partir des modules. Cela dit, elle reste suffisamment proche de la description usuelle pour pouvoir en utiliser les mêmes concepts en les adaptant : les notions de cônes d'influence, de compositions et d'ordre de causalité sont tous des équivalents modulaires aux cônes de lumière et ordre de causalité des collisions dans l'expression usuelle de la dynamique d'une machine à signaux. L'observation est simple : les modules sont sémantiquement décrits par des méta-modules et lors de leur utilisation dans un diagramme espace-temps, ils sont reliés les uns aux autres par des signaux. Cette nouvelle approche permet d'abstraire la notion de collision du modèle pour n'observer que des modules, des boîtes noires d'entrées/sorties, se composants de différentes manières, entraînant de nouvelles visions des dynamiques d'une machine à signaux.

La finalité de ces constructions est de permettre la discrétisation. Pour ce faire, ces concepts peuvent être aisément transposés dans les automates cellulaires fabriqués dans la discrétisation du type de celle employée dans le Chapitre 3. Ce faisant, il devient possible de concevoir des outils permettant de discrétiser une dynamique modulaire continue (de machine à signaux) en son équivalent discrèt. De plus, la notion de méta-module peut être étendue pour permettre d'approximer dans certains cas particuliers. En effet, si les modules continus peuvent être infiniment petits ou infiniment denses, ce n'est pas le cas

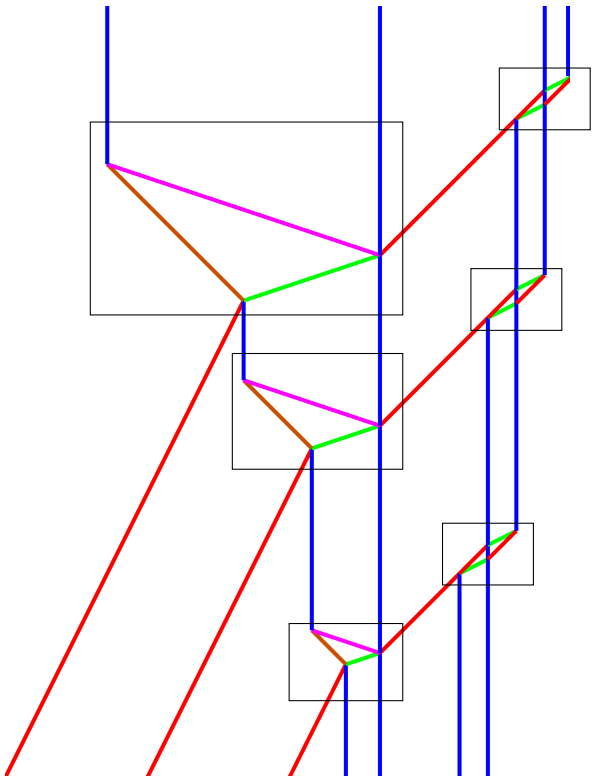


FIGURE 3.10 – Un exemple de découpage modulaire.

des modules discrets. Donc, lorsqu'un tel cas apparaît, cette nouvelle information est appliquée à sa méthode de discrétisation, et permet ainsi d'utiliser une approximation contrôlée par l'utilisateur. Cette solution de discrétisation est donc automatique avec l'apport externe qu'est la solution d'approximation.

Chapitre 4

Définitions de la modularité

Ce chapitre donne toutes les définitions propres à la notion de modularité. Il s'intéresse d'abord à la définition des méta-modules, puis à celle des modules, avant de définir toutes les notions nécessaires pour pouvoir exprimer les dynamiques modulaires fines qui émergent de cette nouvelle approche.

La première section définit les méta-modules continus et leurs pendants discrets. Il s'agit à ce moment de méta-module uniquement rectangulaires. Il est bien sûr possible d'imaginer des méta-modules de toutes formes et tous les traitements *ad hoc*. Cette forme et d'autres choix de simplicité sont pris pour espérer pouvoir *automatiser*. Le pouvoir expressif reste néanmoins très important.

La section suivante s'intéresse à la définition des modules, instanciation des méta-modules. Encore une fois, la version continue est donnée d'abord, puis son pendant discret.

La troisième section définit la notion de *cônes d'influence*, utilisée pour pouvoir exprimer des liens de causalité entre les modules.

Enfin, la dernière section présente la notion de dynamique modulaire, à la fois continue et discrète, qui sera utilisée par la suite pour proposer une méthode de discrétisation.

Ce chapitre ne fait que fournir des définitions permettant de mettre en place la discrétisation par la suite. Par conséquent, il n'apporte pas de résultats mais uniquement des outils permettant de montrer la correction du processus de discrétisation qui est donné dans le Chapitre 5.

De plus, tout au long de ce chapitre et des suivants, il est considéré que la relation de représentation (Définition 14) utilisée dans cette discrétisation modulaire est correcte. Elle est en faite donnée par l'expression des méta-modules discrets représentant les méta-modules continus (ces deux concepts sont définis dans ce chapitre). Par conséquent, il est supposé que l'utilisateur de la discrétisation fournit des informations quant à cette relation de représentation. Si cette hypothèse est bien correcte lors de la discrétisation, cela assure la correction locale de la simulation.

4.1 Méta-modules continu et discret

4.1.1 Méta-module continu

Un méta-module continu est un patron, une description d'un ensemble de modules. Il ne précise pas exactement la taille et la position du module correspondant, ou même

les entrées/sorties exactes. C'est simplement un descriptif indiquant que tout module répondant aux conditions d'entrées aura en sortie les conséquences qu'il précise. Pour faire le parallèle avec la programmation objet, le méta-module est la *classe* et le module une instanciation.

Chaque mention des mots module ou méta-module dans cette sous-section signifie module ou méta-module *continu*.

Un méta-module continu est défini en trois parties :
— des arguments d'entrée,
— des conditions sur ces arguments, et
— des conséquences (résultats) du méta-module.

Arguments Les *arguments* sont des variables typées qui *doivent* être instanciées lors de la création d'un module à partir de ce méta-module. Il existe deux types d'arguments : des arguments *principaux* et des arguments *supplémentaires*.

Principaux Les arguments principaux apparaissent toujours dans la définition d'un méta-module. Il en existe neuf, réparties en cinq catégories :
— $\mathfrak{A} = \{M, S, R\}$ la machine à signaux (méta-signaux, vitesses et règles de collisions),
— I_C, I_L, I_R les patrons des entrées du module (bas, gauche et droite),
— O_C, O_L, O_R les patrons des sorties du module (haut, gauche et droite),
— w la largeur du module, et
— d la durée du module.

Supplémentaires Les arguments supplémentaires sont des variables dont la présence va varier dans chaque méta-module. Ils peuvent être de différents types, mais les plus communs sont des méta-signaux ou des nombres rationnels.

Il n'y a pas d'autres variables (libres ou liées) dans la suite.

Conditions Un méta-module continu définit un certains nombre de *conditions*, qui indiquent si un module est ou non une instanciation correcte de ce méta-module. Ces conditions sont :
— sur la machine,
— syntaxiques (les patrons d'entrées *i.e.* le bas du rectangle et les deux côtés),
— sémantiques (impliquant des calculs sur les rationnels et liant plusieurs variables) dont :
  — des opérations booléennes : $\|, \&\&$,
  — des opérations de comparaison : $!=, ==, <, <=$,
  — des opérations rationnelles : $+, -, *, /, \%$ (modulo) , $/!$ (division entière),
  — le parenthésage,
  — tout cela avec accès aux vitesses et positions des signaux,
— des conditions sémantiques particulières sur la durée, et
— des conditions sémantiques particulières sur la largeur.

Conséquences Les *conséquences* sont des propriétés de la sortie (patrons de sorties, position des signaux, distance entre des signaux, conservation d'un signal entre le début et la fin...), exprimées avec les mêmes outils que les conditions (sémantiques et syntaxiques).

Définition d'un patron d'entrée/sortie continu

Un patron d'entrée/sortie continu exprime l'ordre d'apparition des signaux sur les frontières du module. C'est une expression régulière formée à partir des méta-signaux pour exprimer l'enchaînement syntaxique des signaux. Entre chaque signaux, un caractère indique s'il existe ou non de l'espace entre eux :

— **,** : espace non-nul entre deux signaux
— **.** : aucun espace entre deux signaux (résultat d'une collision)
— **;** : peut y avoir de l'espace ou non entre deux signaux

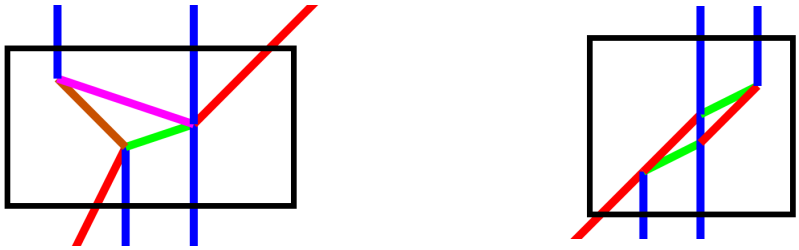
De plus, les signaux peuvent être marquées d'un $\backslash i$ (entrée par le bas) ou $\backslash o$ (sortie par le haut) *puis un entier*, et ces notations peuvent être suivies de $<$ (par la gauche) ou $>$ (par la droite). Par exemple :

— $z\backslash i0.a.c, z\backslash i1$ (patron d'entrée) signifiant qu'un signal z marqué par la valeur 0 entre dans le module par le bas, où il est en collision avec un signal a et un c. Enfin, un signal z marqué 1 entre dans le module par le bas.
— $z\backslash o{<}0, c.b\backslash o2, z\backslash o{<}1$ (patron de sortie de gauche) signifiant qu'un signal z marqué par la valeur 0 sort du module par le côté gauche à distance de deux signaux b et c sortant d'une collision, puis plus loin un signal z marqué 1.

Les numérotations sont utilisées uniquement pour pourvoir reconnaître par la suite quel signal est référencé dans une condition sémantique.

Pour simplifier sa représentation, un méta-module peut être exprimé par un dessin donnant sa forme générale.

L'annexe D donne deux exemples verbatim de méta-modules tels que manipulés par l'implantation, et ils sont illustrés par la Figure 4.1.



(a) élargissement de deux signaux verticaux

(b) décalage de deux signaux verticaux

FIGURE 4.1 – Exemples de méta-modules continus.

4.1.2 Méta-module discret

Un méta-module discret est l'équivalent discret d'un méta-module continu. Il ne précise pas exactement la taille et la position du module correspondant, ou même les entrées/sorties exactes. C'est simplement un descriptif indiquant que tout module répondant aux conditions d'entrées aura en sortie les conséquences qu'il précise.

Cette sous-section décrit le pendant discret des méta-modules continus rectangulaires. De ce fait, elle décrit les méta-modules discrets rectangulaires. De plus, chaque mention des mots module ou méta-module dans celle-ci signifie module ou méta-module *discret*.

Un méta-module discret est défini en quatre parties par :
— des arguments d'entrées,
— des conditions sur ces arguments,
— des conséquences (résultats), et
— des informations de discrétisation.

Les méta-modules discrets portent une information supplémentaire par rapport à leurs pendant continus : ils précisent à quels méta-modules continus ils sont associés.

Arguments Comme pour les méta-modules continus, les arguments sont des étiquettes typées qui *doivent* être instanciées lors de la création d'un module à partir de ce méta-module.

Principaux Les arguments principaux apparaissent toujours dans la définition d'un méta-module. Il y en a neuf, répartis en cinq catégories :
— $\mathfrak{A}$ l'automate cellulaire,
— $\mathfrak{I}_C, \mathfrak{I}_R, \mathfrak{I}_L$ les entrées aux frontières du module (bas, gauche et droite),
— $\mathfrak{O}_C, \mathfrak{O}_R, \mathfrak{O}_L$ les sorties aux frontières du module (haut, gauche et droite),
— $\mathfrak{w}$ la largeur du module, et
— $\mathfrak{d}$ la durée du module.

Dans le cadre de cette thèse, le méta-module discret est fait pour être instancié de façon automatique par l'algorithme de discrétisation, et il n'a donc pas d'automate cellulaire associé. De fait, celui présent dans le méta-module est celui créé à partir de l'ensemble des méta-modules discrets. Cela implique que toute vérification de condition faite sur cet AC sera toujours vraie par construction.

Supplémentaires Les arguments supplémentaires sont des variables dont la présence va varier dans chaque méta-module. Il en existe de plusieurs types mais les plus communs sont des états ou des entiers.

Il n'y a pas d'autres variables (libres ou liées) dans la suite.

Conditions Un méta-module discret définit un certains nombre de *conditions*, qui indiquent si un module peut être associé ou non à ce méta-module. Ces conditions sont :
— sur l'automate cellulaire (règles et états),
— syntaxiques (sur les patrons d'entrées),
— sémantiques (impliquant des calculs sur les entiers et liant plusieurs variables) dont :
  — des opérations booléennes : $\|, \&\&$,
  — des opérations de comparaison : $!=, =, <, <=$,
  — des opérations entières : $+, -, *, \%$ (modulo) , $/!$ (division entière),$pgcd, ppcm$ (plus grand commun diviseur, plus petit commun multiple),
  — le parenthésage,
  — avec accès aux positions des cellules, à la phase, à la période et à la vitesse des CA-Signaux
  — toutes ces conditions sont labellisées avec des informations qui aideront à la discrétisation.
— des conditions sémantiques particulières sur la durée, et

— des conditions sémantiques particulières sur la largeur.

Les conditions sémantiques sont labellisées de sorte à différencier des conditions particulières (les conditions d'échelles, leur utilité est donnée dans le chapitre suivant) des autres.

Conséquences Les *conséquences* sont des propriétés de la sortie (patron de sorties, position des AC-signaux, distance entre les cellules, conservation d'un AC-signal entre le début et la fin...), exprimées avec les mêmes outils que les conditions (sémantiques et syntaxiques).

Informations de discrétisation

Ces informations vont permettre de choisir les méta-modules discrets pour créer l'automate cellulaire. Il y en a deux types :

— des méta-modules continus (ceux qui se discrétisent en ce méta-module discret),

— des états et CA-Signaux qui devront exister dans un automate cellulaire utilisant ce méta-module.

Ces informations sont optionnelles et n'apparaîtront que si le méta-module discret est utilisé dans le cadre de la discrétisation.

Patron d'entrée/sortie discret

Un patron d'entrée/sortie discret exprime l'ordre d'apparition des AC-signaux sur les frontières du module. C'est une expression régulière formée à partir des états pour exprimer l'enchaînement syntaxique des cellules et manipule l'alphabet suivant :

— # : une cellule dans l'état quiescent,

— $\mathfrak{a}_0$: une cellule dans l'état $\mathfrak{a}_0$,

— $\mathfrak{a}.\mathfrak{b}$: une cellule dans l'état $\mathfrak{a}$ puis une cellule dans l'état $\mathfrak{b}$, avec $\mathfrak{a}$ et $\mathfrak{b}$ différents de #.

Les symboles s peuvent avoir en exposant une valeur k qui indique qu'il y a k cellules dans l'état s qui se suivent (*e.g.* : $\#^k$: k cellules quiescentes qui se suivent). Les états peuvent être marqués d'un $\backslash i$ (entrée par le bas) ou $\backslash o$ (sortie par le haut) *puis un entier*, et ces notations peuvent être suivies de $<$ (par la gauche) ou $>$ (par la droite).

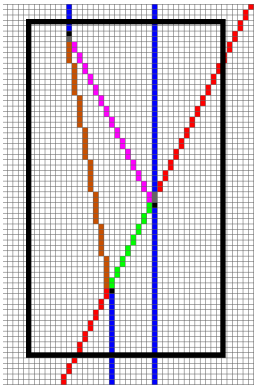
Par exemple :

— $\mathfrak{z}\backslash i0.\#^k.\mathfrak{a}.\mathfrak{c}.\#^l\mathfrak{z}\backslash i1$ (patron d'entrée par le bas) signifiant qu'une cellule dans l'état $\mathfrak{z}$ marqué 0 dans le bas du module est suivies de k cellules dans l'état quiescent puis d'une cellule $\mathfrak{a}$ et d'une $\mathfrak{c}$ après quoi l cellules dans l'état quiescent apparaissent. Enfin, une cellule $\mathfrak{z}$ marquée 1 termine le patron d'entrée par le bas.

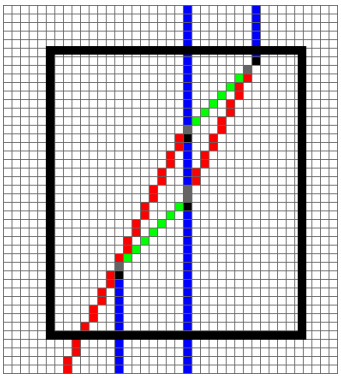
— $\mathfrak{z}\backslash o{>}0.\#^m.\mathfrak{c}.\mathfrak{b}\backslash o{>}2.\#^p.\mathfrak{z}\backslash o{>}1$ (patron de sortie de droite). Ce patron signifie qu'une cellule dans l'état $\mathfrak{z}$ est suivie de m cellules quiescentes puis d'une cellule $\mathfrak{c}$ puis d'une $\mathfrak{b}$ marquée 2 puis de p quiescentes et enfin d'une cellule $\mathfrak{z}$ marquée 1, tout cela dans la sortie de droite.

Les numérotations sont utilisées uniquement pour pouvoir référencer les cellules dans les conditions et conséquences.

Pour compléter sa représentation, un méta-module discret peut être exprimé par un dessin donnant sa forme générale. L'annexe E donne deux exemples de méta-modules discrets tels que manipulés par l'implantation, illustrés par la Figure 4.2.



(a) élargissement entre deux AC-signaux verticaux



(b) décalage de deux AC-signaux verticaux

FIGURE 4.2 – Exemples de méta-modules discrets.

4.2 Modules continu et discret

Un module est une instanciation d'un méta-module. Il existe deux catégories de modules : les flottants et les ancrés. Les modules flottants sont des « briques » élémentaires qui peuvent être utilisées pour créer de toute pièce un diagramme espace-temps. Au contraire, les ancrés sont des modules obtenus à partir de l'observation d'un diagramme existant ou après « ancrage » d'un module flottant.

Dans ce chapitre et les suivants, il est considéré que l'utilisation des modules respectent une hypothèse simple :

Hypothèse 50 *Il n'y a pas de collision en dehors des modules ni sur leurs frontières.*

Cette hypothèse peut se construire facilement en ajoutant des modules « élémentaires » qui ne contiennent qu'une collision seule.

4.2.1 Module continu

Cette section s'intéresse uniquement aux modules continus. Par conséquent, toute mention du mot « module » dans cette section se réfère à leur version continue. De plus, dans les définitions et tout au long de cette sous-section, les structures sont confondues avec l'ensemble de points qu'elles dénotent sur les diagrammes espace-temps auxquelles elles correspondent :

— un signal est confondu avec l'ensemble des points de ce signal sur le diagramme,
— une collision est confondue avec le point d'emplacement de cette collision sur le diagramme,
— un module est confondu avec l'ensemble des points de ce module sur le diagramme,
— une frontière est confondue avec l'ensemble des points de cette frontière sur le diagramme.

Les premiers modules à devoir être définis sont les *flottants*. Leur nom provient du fait qu'ils ne sont associés à aucune coordonnées spatio-temporelles et donc « flottent ».

Définition 51 (Module continu flottant) Un *module continu flottant* est défini par :
— $\mathfrak{A}$ une machine à signaux,
— m un méta-module continu,
— les signaux d'entrées et leurs positions relatives (entre 0 et 1),
— les signaux de sorties et leurs positions relatives (entre 0 et 1), et
— les valeurs éventuelles des arguments supplémentaires du méta-module continu associé.
Les valeurs de 0 et 1 sont relatives et n'expriment pas la même échelle pour les deux dimensions.

Après les flottants, viennent les modules ancrés. Ceux-ci disposent de coordonnées spatio-temporelles et existent donc dans un diagrammes espace-temps. Par conséquent, il s'agit de modules flottants qui sont désormais fixe, et ont donc été « ancrés ».

Définition 52 (Module continu ancré) Un *module continu ancré* est défini par :
— un module flottant,
— w une largeur ,
— d une durée,
— (x_o, t_o) un point d'ancrage (le coin bas gauche du rectangle),
— de plus, les coordonnées des signaux d'entrée et de sortie sont calculés (à l'aide de la largeur, de la durée et du point d'ancrage).

Le processus du passage du module flottant au module ancré est appelé *ancrage*.

Les valeurs d'espace et de temps marquant un module ancré, x^-, x^+, t^-, t^+, sont déduites à partir du point d'ancrage, de la largeur et de la durée, comme illustré dans la Fig. 4.3. Dans les flottants, ces points existent mais leurs valeurs exactes sont inconnues (et sont données lors de l'ancrage du module).

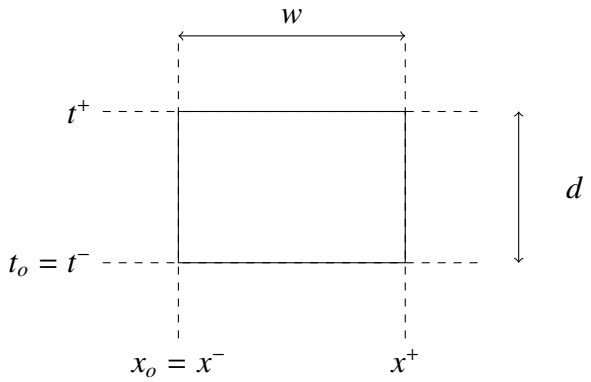


FIGURE 4.3 – Rectangle délimitant un module continu.

À partir de maintenant, x^-, x^+, t^- et t^+ feront toujours référence à ces quatre valeurs, parfois indiquant le module correspondant (x^-_M...). Il en va de même pour w, d, et (x_o, t_o), pouvant aussi indiquer le module (w_M...).

Les *frontières* sont les côtés du rectangle délimitant un module. Formellement :

Définition 53 (Frontières d'un module) Soit M un module, M^N (la frontière nord) est l'ensemble des points (x, t) tels que : $x_o \leq x \leq x_o + w$ et $t = t^+ = t_o + d$. Les autres frontières (sud, est et ouest) sont définit de manière analogue. Elles sont notées M^S, M^E, M^W, respectivement

pour les frontières sud, est et ouest, comme illustré dans la Fig. 4.5(a). M^S est une frontière d'entrée. M^N est une frontière de sortie. M^E et M^W sont des frontières d'entrée et de sortie.

Les notions de signaux entrants et sortants de modules nécessitent de définir davantage d'éléments pour pouvoir les décrire. Ils le sont donc dans la section suivante. Cependant, il est possible de définir ici les entrées et sorties de modules.

Les signaux entrants dans un module forment donc son *entrée* :

Définition 54 (Entrées d'un module) Les entrées sur les trois côtés d'un module sont définies comme des restrictions de $\mathbb{D}$ à l'ensemble des positions (x, t) telles que :

— entrée par la gauche $I_W : t^- < t < t^+$ et $x = x^-$,
— entrée par le bas $I_S : x^- \leq x \leq x^+$ et $t = t^-$, et
— entrée par la droite $I_E : t^- < t < t^+$ et $x = x^+$.

En remplaçant chaque signal entrant par son méta-signal associé et chaque collision par l'ensemble des méta-signaux entrants dans celle-ci et ne provenant pas de l'intérieur du module. Elles sont illustrées dans la Fig. 4.5(b).

Géométriquement, I_W correspond aux positions de M^W, I_S à celles de M^S et I_E aux positions de M^E.

Ces entrées sont alors utilisées pour exprimer la configuration initiale d'un module.

Définition 55 (Configuration initiale d'un module) La *configuration initiale* d'un module continu est la concaténation des signaux de sa frontière Est sur les positions dans lesquelles ils seraient au temps t^-, puis les signaux de la frontière Sud, et enfin ceux de la frontière Ouest (ramenée au temps t^-). Elle est illustrée dans la Figure 4.4.

Cette notation permet de ne manipuler qu'une seule configuration pour représenter les entrées d'un module. Cela simplifie sa représentation lorsqu'un module est utilisé dans un cadre théorique, comme c'est notamment le cas par la suite dans le Chapitre 5.

Il est possible de définir le dual de cette configuration initiale (qui pourrait s'appeler configuration finale) mais ce n'est pas le cas dans cette thèse. En effet, bien qu'il paraisse logique de la définir, cette notion n'est jamais utile par la suite, et donc la proposer n'apporterait rien.

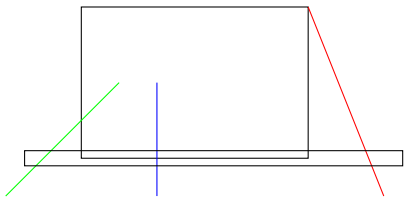


FIGURE 4.4 – Configuration initiale d'un module.

Les signaux sortants d'un module forment sa *sortie* :

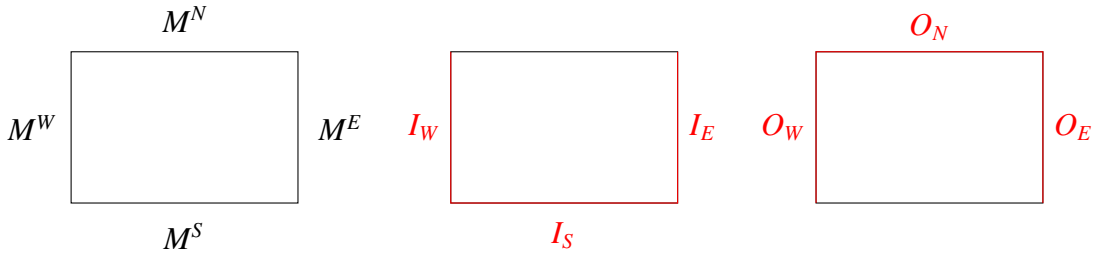
Définition 56 (Sorties d'un module) Les sorties sur les trois côtés d'un module sont définies comme des restrictions de $\mathbb{D}$ à l'ensemble des positions (x, t) telles que :

— sortie par la gauche $O_W : t^- < t < t^+$ et $x = x^-$,
— sortie par le haut $O_N : x^- \leq x \leq x^+$ et $t = t^+$, et

— sortie par la droite O_E : $t^- < t < t^+$ et $x = x^+$.

En remplaçant chaque signal sortant par son méta-signal associé et chaque collision par l'ensemble des méta-signaux sortants de celle-ci et ne provenant pas de l'extérieur du module. Elles sont illustrées dans la Fig. 4.5(c).

Géométriquement, O_W correspond aux positions de M^W, O_N à celles de M^N et O_E aux positions de M^E.



(a) frontières d'un module (b) entrées d'un module (c) sorties d'un module

FIGURE 4.5 – Frontières, entrées et sorties d'un module.

4.2.2 Module discret

Un module discret est une instanciation d'un méta-module discret dans le cadre d'un automate cellulaire (qui est construit à partir d'un ensemble de méta-modules discrets). Il se compose des coordonnées spatio-temporelles exactes du rectangle et des états d'entrées/sorties ainsi que des instanciations des distances de séparation. Les notions de modules flottants et ancrés s'y retrouvent aussi, mais ajoutent de la complexité puisqu'il faut vérifier que les valeurs auparavant comprises entre 0 et 1 sont entières après l'ancrage.

Il diffère malgré tout d'un module continu dans le sens où il est généralement engendré automatiquement à partir des modules continus de la machine à signaux que l'on souhaite discrétiser.

Dans les définitions et tout au long de cette section, les structures sont confondues avec l'ensemble des sites qu'elles dénotent sur les diagrammes espace-temps auxquelles elles correspondent :

— un AC-signal est confondu avec l'ensemble des sites de cet AC-signal sur le diagramme,
— un module est confondu avec l'ensemble des sites de ce module sur le diagramme,
— une frontière est confondue avec l'ensemble des sites de cette frontière sur le diagramme,

Les modules discrets flottants et ancrés se définissent de manière analogue à ceux continus :

Définition 57 (Module discret flottant) Un *module discret flottant* est défini par :

— $\mathfrak{A}$ un automate cellulaire (dans le cadre de la discrétisation, fabriqué à partir des méta-modules),
— $\mathfrak{m}$ un méta-module discret,

— les instanciations relatives des distances entre les cellules dans les états (*e.g.* : la valeur comprise entre 0 et 1 de k dans e^k), et
— les valeurs éventuelles des arguments supplémentaires du méta-module associé.

Toutes ces valeurs sont pour l'instant rationnelles, et donc pas toujours entières. Cependant, lors de l'ancrage des ces modules, les valeurs taille et durée des modules sont choisies de sortes à ce que les valeurs instanciées soient entières.

Définition 58 (Module discret ancré) Un *module discret ancré* est défini par :
— un module discret flottant,
— $\mathfrak{w}$ une largeur,
— $\mathfrak{d}$ une durée,
— $(\mathfrak{x}_o, \mathfrak{t}_o)$ un site d'ancrage (le coin bas gauche du rectangle).
De plus, les coordonnées des états d'entrée et de sortie sont calculées (à l'aide de la largeur, de la durée et du point d'ancrage). Toutes ces valeurs sont entières puisqu'elles sont dans le cadre des automates cellulaires.

Les valeurs d'espace et de temps $\mathfrak{x}^-, \mathfrak{x}^+, \mathfrak{t}^-, \mathfrak{t}^+$ marquant un module discret sont des valeurs entières déduites à partir de l'ancrage, de la largeur et de la durée, comme illustré dans la Fig. 4.6.

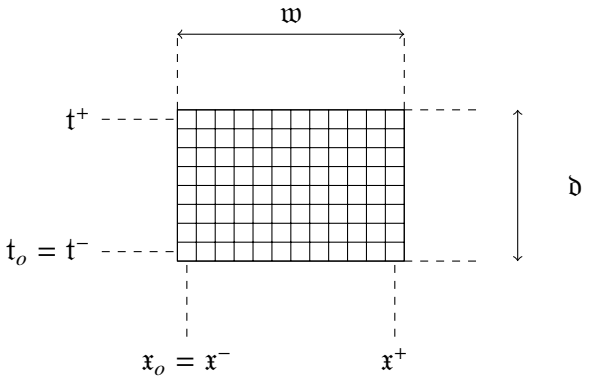


FIGURE 4.6 – Rectangle déterminant un module discret.

À partir de maintenant, $\mathfrak{x}^-$, $\mathfrak{x}^+$, $\mathfrak{t}^-$ et $\mathfrak{t}^+$ feront toujours référence à ces quatre valeurs, parfois indiquant le nom du module correspondant ($\mathfrak{x}^-_{\mathfrak{M}}$...). Il en va de même pour $\mathfrak{w}$, $\mathfrak{d}$, et $(\mathfrak{x}_o, \mathfrak{t}_o)$, pouvant aussi préciser le module ($\mathfrak{w}_{\mathfrak{M}}$...).

Les notions de frontières, d'entrées et de sorties sont définies de manière analogue à leurs versions continues.

Définition 59 (Frontières d'un module discret) Soit $\mathfrak{M}$ un module discret : $\mathfrak{M}^N$ (la frontière nord) est l'ensemble des sites $(\mathfrak{x}, \mathfrak{t})$ tels que : $\mathfrak{x}_o \leq \mathfrak{x} \leq \mathfrak{x}_o + \mathfrak{w} - 1$ et $\mathfrak{t} = \mathfrak{t}^+ = \mathfrak{t}_o + \mathfrak{d} - 1$. Les autres frontières (sud, est et ouest) sont définies de manière analogue. Par conséquent, les cellules dans les coins du module sont dans deux frontières à la fois. Elles sont notées $\mathfrak{M}^S$, $\mathfrak{M}^E$, $\mathfrak{M}^W$, respectivement pour les frontières sud, est et ouest, comme illustré dans la Fig. 4.7(a). $\mathfrak{M}^S$ est une frontière d'entrée. $\mathfrak{M}^N$ est une frontière de sortie. $\mathfrak{M}^E$ et $\mathfrak{M}^W$ sont des frontières d'entrée et de sortie.

Définition 60 (Entrées d'un module discret) Les entrées sur les trois côtés d'un module sont définies comme des restrictions de $\mathfrak{D}$ à l'ensemble des sites (x, t) tels que :

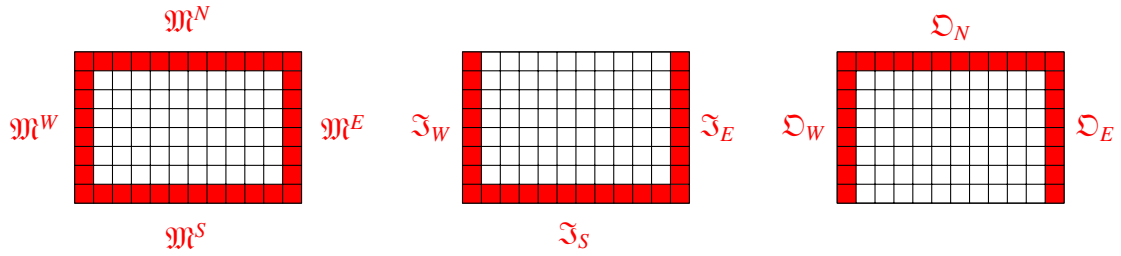
— $\mathfrak{I}_W : t^- < t \leq t^+$ et $x = x^-$ pour les entrées par la gauche,
— $\mathfrak{I}_S : x^- \leq x \leq x^+$ et $t = t^-$ pour les entrées par le bas, et
— $\mathfrak{I}_E : t^- < t \leq t^+$ et $x = x^+$ pour les entrées par la droite.

En remplaçant chaque site par son état associé. Elles sont illustrées dans la Fig. 4.7(b).

Définition 61 (Sorties d'un module discret) Les sorties sur les trois côtés d'un module sont définies comme des restrictions de $\mathfrak{D}$ à l'ensemble des sites (x, t) tels que :

— $\mathfrak{O}_W : t^- \leq t < t^+$ et $x = x^-$ pour les sorties par la gauche,
— $\mathfrak{O}_N : x^- \leq x \leq x^+$ et $t = t^+$ pour les sorties par le haut, et
— $\mathfrak{O}_E : t^- \leq t < t^+$ et $x = x^+$ pour les sorties par la droite.

En remplaçant chaque site par son état associé. Elles sont illustrées par la Fig. 4.7(c).



(a) frontières d'un module (b) entrées d'un module discret (c) sorties d'un module discret

FIGURE 4.7 – Frontières, entrées et sorties d'un module discret.

4.3 Cônes

L'utilité des cônes d'influence reçue et émise est de définir les zones d'influences d'un module dans son diagramme espace-temps, afin de pouvoir en associer plusieurs en une chaîne de causalité. Bien que leur forme géométrique ne soit pas un cône entier, ils conservent ce nom en référence aux cônes de lumières dans le modèle des machines à signaux. Dans les définitions de cette section, un cône est confondu avec l'ensemble des points ou sites de ce cône sur le diagramme.

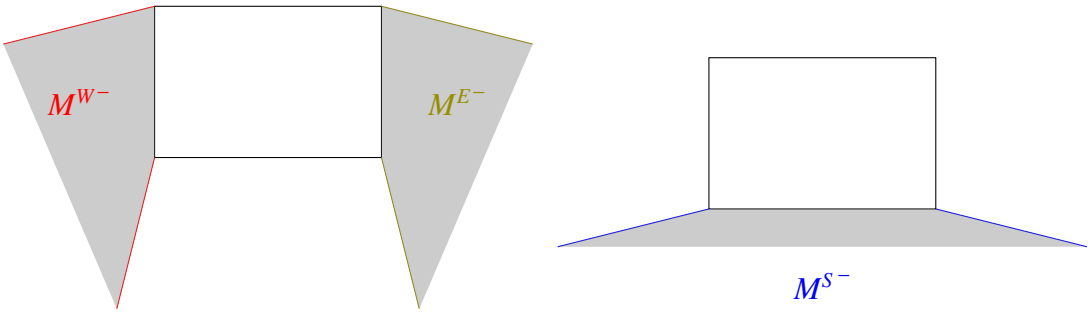
4.3.1 Cônes continus

Cette sous-section exprime les différents cônes d'influences dans le cas des modules continues. Ces définitions nécessitent de nommer quelques vitesses dans la machine à signaux dans laquelle le module est utilisé. Ces vitesses sont :

— v_{min} la plus petite vitesse,
— $v_{max}^{<0}$ la plus grande vitesse strictement négative,
— $v_{min}^{0<}$ la plus petite vitesse strictement positive, et
— v_{max} la plus grande vitesse.

Il existe deux types de cônes : ceux qui reçoivent de l'information et ceux qui en émettent. Les premiers sont les *cônes d'influence reçue* :

Définition 62 (Cônes d'influence reçue) M^{W^-}, le cône Ouest est tout (x,t) tels que : $x^- + (t - t^+)v_{max} \leq x \leq x^- + (t - t^-)v_{min}^{0<}$ et $x < x^-$.
M^{S^-}, le cône Sud est tout (x,t) tels que : $x^- + (t - t^-)v_{max} \leq x \leq x^+ + (t - t^-)v_{min}$ et $t < t^-$.
M^{E^-}, le cône Est est tout (x,t) tels que : $x^+ + (t - t^-)v_{max}^{<0} \leq x \leq x^+ + (t - t^+)v_{min}$ et $x^+ < x$.
Ils sont illustrés par la Figure 4.8.



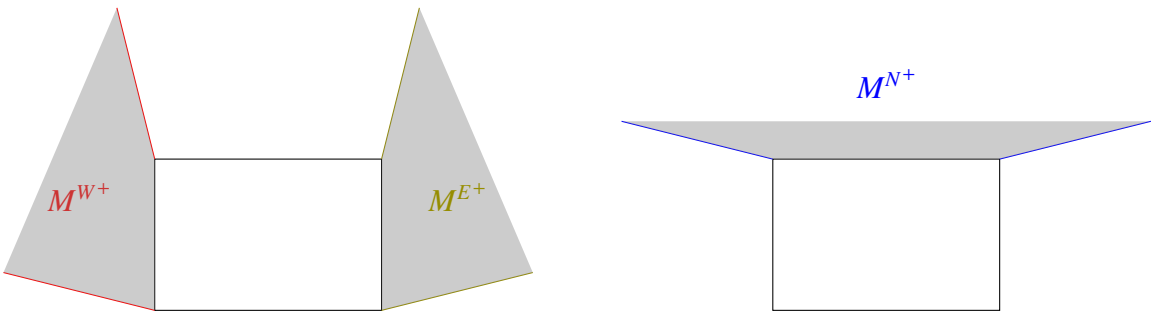
(a) cônes de gauche et droite (b) cône du milieu

FIGURE 4.8 – Cônes d'influence reçue.

Informellement, un cône d'influence reçue contient tout les points tels que si un autre module s'y trouve au moins partiellement, il peut être à l'origine d'un (ou plusieurs) des signaux entrants dans le module associé au cône.

Le deuxième type de cônes sont les *cônes d'influence émise* :

Définition 63 (Cônes d'influence émise) M^{W^+}, le *cône Ouest* est tout (x,t) tels que : $x^- + (t - t^-)v_{min} \leq x \leq x^- + (t - t^+)v_{max}^{<0}$ et $x < x^-$.
M^{N^+}, le *cône Nord* est tout (x,t) tels que : $x^- + (t - t^+)v_{min} \leq x \leq x^+ + (t - t^+)v_{max}$ et $t^+ < t$.
M^{E^+}, le *cône Est* est tout (x,t) tels que : $x^+ + (t - t^+)v_{min}^{0<} \leq x \leq x^+ + (t - t^-)v_{max}$ et $x^+ < x$.
Ils sont illustrés dans la Figure 4.9.



(a) cônes de gauche et droite (b) cône du milieu

FIGURE 4.9 – Cônes d'influence émise.

Informellement, un cône d'influence émise contient tous les points tels que si un autre module s'y trouve au moins partiellement, il peut recevoir un ou plusieurs signaux provenant du module associé au cône.

Pour pouvoir définir l'entrée et la sortie d'un cône, il est d'abord nécessaire de préciser la notion de signal achevé :

Définition 64 (Signal achevé) Un *signal achevé* est l'union de l'ensemble des points d'un signal sur le diagramme, de sa position de naissance et de la position de sa collision de mort si elle existe. Si s est un signal, le signal achevé correspondant est noté $\bar{s}$.

Avec cela, l'entrée dans un cône d'influence est définie :

Définition 65 (Signal entrant par un cône d'influence) Soit s un signal.

Soit M^{E^-} un cône d'influence reçue de droite. Alors s est *entrant* dans M^{E^-} si :

$$|\bar{s} \cap M^E| = 1, |s \cap M^{E^-}| \neq 0 \text{ et } S(s) < 0.$$

Cela est noté $s \triangleright M^{E^-}$ ou $M^{E^-} \triangleleft s$.

La notion d'entrée pour les cônes d'influence reçue de gauche est définie de manière analogue.

Soit M^{S^-} un cône d'influence reçue Sud. Alors s est *entrant* dans M^{S^-} si :

$$|\bar{s} \cap M^S| = 1, |s \cap M^{S^-}| \neq 0 \quad .$$

De plus, si $S(s) > 0$ alors le point d'intersection de s avec le module doit être différent de (x^+, t^-). De la même manière, si $S(s) < 0$ alors le point d'intersection de s avec le module doit être différent de (x^-, t^-). Ces deux restrictions permettent de refuser tout signal « effleurant » simplement le module, à la manière de la Figure 4.10.

Cela est noté $s \triangleright M^{S^-}$ ou $M^{S^-} \triangleleft s$.

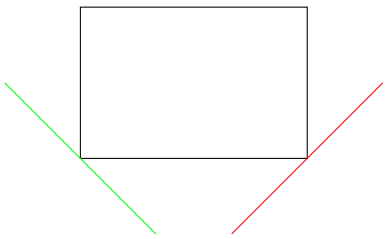


FIGURE 4.10 – Signaux effleurant un module.

Les signaux entrants dans un cône entre du même fait dans un module :

Définition 66 (Signal entrant dans un module) Un signal s est entrant dans un module M si : $s \triangleright M^{E^-}$ ou $s \triangleright M^{S^-}$ ou $s \triangleright M^{W^-}$.

Cela est noté $s \triangleright M$ ou $M \triangleleft s$. La coordonnée spatiale d'entrée de s dans M est notée s_{x_α}. La coordonnée temporelle d'entrée de s dans M est notée s_{t_α}. Le point de coordonnées $(s_{x_\alpha}, s_{t_\alpha})$ est noté s_α.

Les signaux pouvant donc entrer dans un cône, ils peuvent aussi en sortir :

Définition 67 (Signal sortant d'un cône d'influence) Soit M^{E^+} un cône d'influence émise de droite et s un signal. Alors s est *sortant* de M^{E^+} si :

$$|\bar{s} \cap M^E| = 1, |s \cap M^{E^+}| \neq 0 \text{ et } S(s) > 0.$$

Cela est noté $s \lhd M^{E^+}$ ou $M^{E^+} \rhd s$. La notion de sortie pour les deux autres types de cônes d'influence émise est définie de manière analogue .

Et de manière analogue à l'entrée dans un module, les signaux en sortent lorsqu'ils sortent aussi d'un cône :

Définition 68 (Signal sortant d'un module) Un signal s est sortant d'un module M si :
$s \lhd M^{E^+}$ ou $s \lhd M^{N^+}$ ou $s \lhd M^{W^+}$.
Cela est noté $s \lhd M$ ou $M \rhd s$. La coordonnée spatiale de sortie de s dans M est notée s_{x_β}. La coordonnée temporelle de sortie de s dans M est notée s_{t_β}. Le point de coordonnées $(s_{x_\beta}, s_{t_\beta})$ est noté s_β.

Pour finir, les collisions ne peuvent se passer sur une frontière (que ce soit en entrée ou en sortie).

4.3.2 Cônes discrets

Cette sous-section s'intéresse à exprimer les différents cônes d'influences dans le cas des modules discrets. Ces définitions nécessitent de nommer quelques vitesses parmi les vitesses des AC-signaux de l'automate cellulaire dans lequel le module est utilisé. Ces vitesses sont :

— $\mathfrak{v}_{min}$ la plus petite vitesse,
— $\mathfrak{v}_{max}^{<0}$ la plus grande vitesse strictement négative,
— $\mathfrak{v}_{min}^{0<}$ la plus petite vitesse strictement positive, et
— $\mathfrak{v}_{max}$ la plus grande vitesse.

Tout d'abord, le pendant discret des cônes d'influence reçue est défini :

Définition 69 (Cônes d'influence reçue discrets) $\mathfrak{M}^{W^-}$, le *cône Ouest*, est tout site $(\mathfrak{x}, \mathfrak{t})$ tel que : $\mathfrak{x}^- + \lceil(\mathfrak{t} - \mathfrak{t}^+)\mathfrak{v}_{max}\rceil \leq \mathfrak{x} \leq \mathfrak{x}^- + \lfloor(\mathfrak{t} - \mathfrak{t}^-)\mathfrak{v}_{min}^{0<}\rfloor$ et $\mathfrak{x} < \mathfrak{x}^-$.

$\mathfrak{M}^{S^-}$, le *cône Sud*, est tout site $(\mathfrak{x}, \mathfrak{t})$ tel que : $\mathfrak{x}^- + \lfloor(\mathfrak{t} - \mathfrak{t}^-)\mathfrak{v}_{max}\rfloor \leq \mathfrak{x} \leq \mathfrak{x}^+ + \lfloor(\mathfrak{t} - \mathfrak{t}^-)\mathfrak{v}_{min}\rfloor$ et $\mathfrak{t} < \mathfrak{t}^-$.

$\mathfrak{M}^{E^-}$, le *cône Est*, est tout site $(\mathfrak{x}, \mathfrak{t})$ tel que : $\mathfrak{x}^+ + \lceil(\mathfrak{t} - \mathfrak{t}^-)\mathfrak{v}_{max}^{<0}\rceil \leq \mathfrak{x} \leq \mathfrak{x}^+ + \lfloor(\mathfrak{t} - \mathfrak{t}^+)\mathfrak{v}_{min}\rfloor$ et $\mathfrak{x}^+ < \mathfrak{x}$.

Ils sont illustrés par la Figure 4.11.

Les cônes d'influence émise sont eux aussi définis :

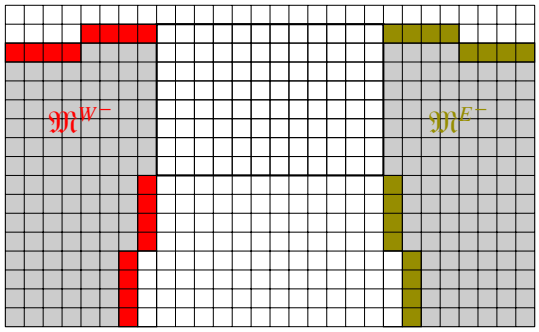
Définition 70 (Cônes d'influence émise discrets) $\mathfrak{M}^{W^+}$, le cône Ouest est tout site $(\mathfrak{x}, \mathfrak{t})$ tel que : $\mathfrak{x}^- + \lfloor(\mathfrak{t} - \mathfrak{t}^-)\mathfrak{v}_{min}\rfloor \leq \mathfrak{x} \leq \mathfrak{x}^- + \lceil(\mathfrak{t} - \mathfrak{t}^+)\mathfrak{v}_{max}^{<0}\rceil$ et $\mathfrak{x} < \mathfrak{x}^-$.

M^{N^+}, le cône Nord est tout site $(\mathfrak{x}, \mathfrak{t})$ tel que : $\mathfrak{x}^- + \lceil(\mathfrak{t} - \mathfrak{t}^+)\mathfrak{v}_{min}\rceil \leq \mathfrak{x} \leq \mathfrak{x}^+ + \lceil(\mathfrak{t} - \mathfrak{t}^+)\mathfrak{v}_{max}\rceil$ et $\mathfrak{t}^+ < \mathfrak{t}$.

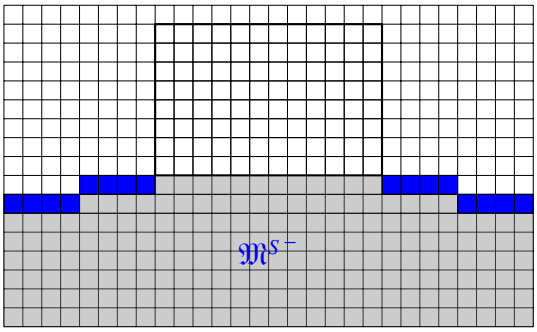
M^{E^+}, le cône Est est tout site $(\mathfrak{x}, \mathfrak{t})$ tel que : $\mathfrak{x}^+ + \lfloor(\mathfrak{t} - \mathfrak{t}^+)\mathfrak{v}_{min}^{0<}\rfloor \leq \mathfrak{x} \leq \mathfrak{x}^+ + \lceil(\mathfrak{t} - \mathfrak{t}^-)\mathfrak{v}_{max}\rceil$ et $\mathfrak{x}^+ < \mathfrak{x}$.

Ils sont illustrés dans la Figure 4.12.

Avec cela, il est possible de définir les notions d'entrées/sorties de cônes et modules.

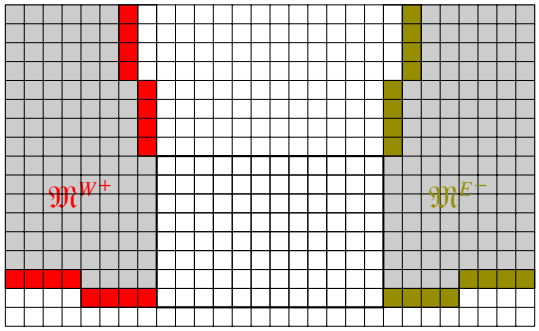


(a) cônes de gauche et droite

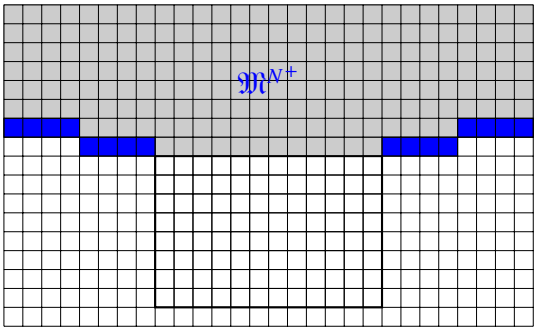


(b) cône du milieu

FIGURE 4.11 – Cônes d'influence reçue.



(a) Cônes de gauche et droite



(b) Cône du milieu

FIGURE 4.12 – Cônes d'influence émise.

Définition 71 (AC-signal entrant dans un cône) Soit $\mathfrak{M}^{E-}$ un cône d'influence reçue de droite et $\mathfrak{s}$ un AC-signal. Alors $\mathfrak{s}$ est *entrant* dans $\mathfrak{M}^{E-}$ si : $|\mathfrak{s} \cap \mathfrak{M}^{E}| = 1$, $|\mathfrak{s} \cap \mathfrak{M}^{E-}| \neq 0$ et la vitesse de $\mathfrak{s}$ est inférieure à 0. Cela est noté $\mathfrak{s} \triangleright \mathfrak{M}^{E-}$ ou $\mathfrak{M}^{E-} \triangleleft \mathfrak{s}$. La notion d'entrée pour les deux autres types de cônes d'influence reçue est définie de manière analogue.

Définition 72 (AC-signal entrant dans un module) Un AC-signal $\mathfrak{s}$ est entrant dans un module $\mathfrak{M}$ si : $\mathfrak{s} \triangleright \mathfrak{M}^{E-}$ ou $\mathfrak{s} \triangleright \mathfrak{M}^{S-}$ ou $\mathfrak{s} \triangleright \mathfrak{M}^{W-}$. Cela est noté $\mathfrak{s} \triangleright \mathfrak{M}$ ou $\mathfrak{M} \triangleleft \mathfrak{s}$. Le site appartenant à la fois à $\mathfrak{s}$ et à une frontière d'entrée de $\mathfrak{M}$ (le « site d'entrée » du AC-signal) est noté $c_{\alpha}^{t_{\bar{M}}}$.

Définition 73 (AC-signal sortant d'un cône) Soit $\mathfrak{M}^{E+}$ un cône d'influence émise de droite et $\mathfrak{s}$ un AC-signal. Alors $\mathfrak{s}$ est *sortant* de $\mathfrak{M}^{E+}$ si : $|\mathfrak{s} \cap \mathfrak{M}^{E}| = 1$, $|\mathfrak{s} \cap \mathfrak{M}^{E+}| \neq 0$ et la vitesse de $\mathfrak{s}$ est supérieure à 0. Cela est noté $\mathfrak{s} \triangleleft \mathfrak{M}^{E+}$ ou $\mathfrak{M}^{E+} \triangleright \mathfrak{s}$. La notion de sortie pour les deux autres types de cônes d'influence émise est définie de manière analogue.

Définition 74 (AC-signal sortant d'un module) Un AC-signal $\mathfrak{s}$ est sortant d'un module $\mathfrak{M}$ si : $\mathfrak{s} \triangleleft \mathfrak{M}^{E+}$ ou $\mathfrak{s} \triangleleft \mathfrak{M}^{N+}$ ou $\mathfrak{s} \triangleleft \mathfrak{M}^{W+}$. Cela est noté $\mathfrak{s} \triangleleft \mathfrak{M}$ ou $\mathfrak{M} \triangleright \mathfrak{s}$. Le site appartenant à la fois à $\mathfrak{s}$ et à une frontière de sortie de $\mathfrak{M}$ (le « site de sortie » du AC-signal) est noté $c_{\beta}^{t_{\bar{M}}}$.

4.4 Compositions et dynamiques modulaires

La dynamique modulaire est l'objectif que toutes les définitions précédentes permettent d'atteindre. C'est une nouvelle expression de la dynamique d'une machine à signaux (pour la version continue) et de celle d'un automate cellulaire utilisant les AC-signaux (pour la version discrète). Ainsi, prouver l'équivalence d'une dynamique modulaire continue et d'une discrète permet d'établir une forme de discrétisation.

4.4.1 Compositions et dynamiques modulaires continues

Cette sous-section définit tout les aspects de la dynamique modulaire dans le cadre continu ds machines à signaux.

Construire la dynamique modulaire passe tout d'abord par exprimer comment les modules sont causalement liés. Ce lien s'exprime avec la notion de composition de modules, qui se décline en deux types. Le premier est la composition qui apparaît le plus naturellement dans ce type de dynamique. Il s'agit de la composition séquentielle :

Définition 75 (Composition séquentielle) Deux modules M_1 et M_2 sont dits composés séquentiellement si : $\exists s$ tel que $M_1 \rhd s \rhd M_2$ et $\nexists s'$ tel que $M_2 \rhd s' \rhd M_1$ avec s et s' des signaux. Cela est noté $M_1 \rightarrow M_2$. La clôture transitive de la composition séquentielle est notée $\rightarrow^+$. Dans les figures, les compostions séquentielles sont représentées par une flèche partant du module source et allant vers le module destination.

Informellement, deux modules sont composés séquentiellement quand un seul d'entre eux émet au moins un signal qui est reçu par l'autre. Deux modules sont liés par la clôture transitive de la composition séquentielle s'il est possible d'en atteindre en suivant une chaîne de compositions partant de l'autre.

Le deuxième type de composition est plus rare. Il s'agit d'une composition qui porte le nom *d'interaction* :

Définition 76 (Interaction) Deux modules M_1 et M_2 sont dits en interaction si : $\exists s$ tel que $M_1 \rhd s \rhd M_2$ et $\exists s'$ tel que $M_2 \rhd s' \rhd M_1$ avec s et s' des signaux. Cela est noté $M_1 \leftrightarrow M_2$. La clôture transitive de l'interaction est notée $\leftrightarrow^+$.

Informellement, deux modules sont en interaction quand tout deux émettent au moins un signal qui est reçu par l'autre. De même, deux modules sont liés par la clôture transitive de l'interaction s'il est possible d'en atteindre un en suivant les interaction de l'autre.

Hypothèse 77 *$\forall M_1, M_2$ des modules, si $M_1 \rightarrow^+ M_2$ alors $\neg(M_2 \rightarrow^+ M_1)$. Autrement dit, $\rightarrow^+$ est antisymétrique.*

Informellement, cela signifie que les constructions localement séquentielles et globalement en interaction sont interdites. Une telle construction est illustrée par la Fig. 4.13.

Sur la figure 4.13, les relations suivantes existent : $M_1 \rightarrow M_2 \rightarrow M_3 \rightarrow M_1$, ce qui est possible mais indésirable car en ce cas $M_1 \rightarrow^+ M_3$ et $M_3 \rightarrow^+ M_1$. Pour éviter ce genre de structure (Hypothèse 77) les modules M_2 et M_3 peuvent être regroupés en un seul module M_4 qui est en interaction avec M_1. Il est aussi possible de découper le module M_1 en deux modules différents qui seront chacun composés séquentiellement avec un des modules M_2 et M_3.

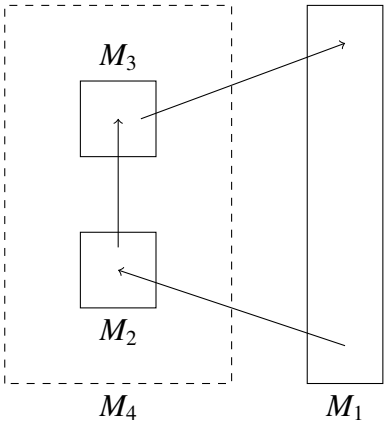


FIGURE 4.13 – Construction incorrecte par rapport à l'Hypothèse 77.

Avec les différentes compositions et l'Hypothèse 77, il est possible d'exprimer la dynamique modulaire continue :

Définition 78 (Dynamique modulaire fine continue) La *dynamique modulaire fine* (continue) est l'ensemble des modules et la description de leurs compositions séquentielles et interactions. Elle peut être représentée par un graphe orienté où les modules en interactions ont une arête non orientée les reliant (voir Figure 4.14). Elle est notée $\mathcal{D}$.

Enfin, exprimer l'équivalence de deux dynamiques modulaires continues permettra par la suite de prouver la correction de certains parties de la discrétisation modulaire :

Définition 79 (Équivalence de dynamiques modulaires fines) Deux dynamiques modulaires fines $\mathcal{D}_1$ et $\mathcal{D}_2$ sont équivalentes s'il y a un isomorphisme f entre chaque éléments de $\mathcal{D}_1$ et $\mathcal{D}_2$, c'est-à-dire :
— $\forall M_1, M_2 \in \mathcal{D}_1,\ M_1 \to M_2 \Leftrightarrow f(M_1) \to f(M_2)$ et
— $\forall M_1, M_2 \in \mathcal{D}_1,\ M_1 \leftrightarrow M_2 \Leftrightarrow f(M_1) \leftrightarrow f(M_2)$.

4.4.2 Compositions et dynamiques modulaires discrètes

Cette sous-section donne les expressions discrètes de toutes les notions précédentes.

Le pendant discret de la dynamique modulaire continue est celle s'appuyant sur les modules discrets. Pour que ceux-ci puissent exister, il faut se placer dans un cadre d'automates cellulaires utilisant uniquement des AC-signaux, et pas dans le cadre classique. De manière analogue à la notion continue, il est alors nécessaire de poser une hypothèse :

Hypothèse 80 *Il n'y a pas d'état collision en dehors des modules ni sur les frontières.*

Dans cet aspect discret, les compositions séquentielles existent aussi :

Définition 81 (Composition séquentielle discrète) Deux modules $\mathfrak{M}_1$ et $\mathfrak{M}_2$ sont dits composés séquentiellement si : $\exists \mathfrak{s}$ tel que $\mathfrak{M}_1 \rhd \mathfrak{s} \rhd \mathfrak{M}_2$ et $\nexists \mathfrak{s}'$ tel que $\mathfrak{M}_2 \rhd \mathfrak{s}' \rhd \mathfrak{M}_1$, avec $\mathfrak{s}$ et $\mathfrak{s}'$ des CA-Signaux. Cela est noté $\mathfrak{M}_1 \dashrightarrow \mathfrak{M}_2$. La clôture transitive de la composition séquentielle

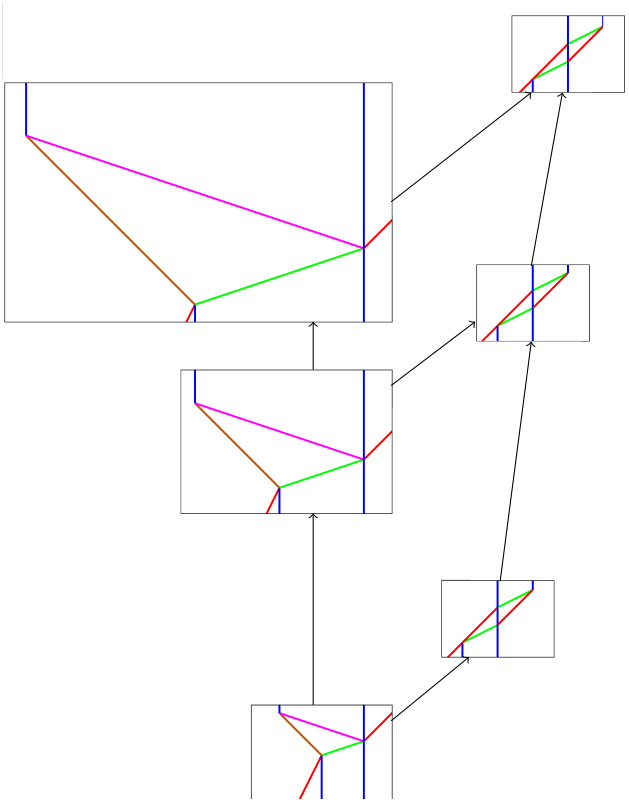


FIGURE 4.14 – Exemple de dynamique modulaire fine continue.

est notée $\dashrightarrow^+$. Dans les figures, les compostions séquentielles sont représentées par une flèche partant du module source et allant vers le module destination.

De nouveau, dans l'aspect discret, les interactions existent :

Définition 82 (Interaction discrète) Deux modules $\mathfrak{M}_1$ et $\mathfrak{M}_2$ sont dits en interaction si : $\exists \mathfrak{s}$ tel que $\mathfrak{M}_1 \rhd \mathfrak{s} \rhd \mathfrak{M}_2$ et $\exists \mathfrak{s}'$ tel que $\mathfrak{M}_2 \rhd \mathfrak{s}' \rhd \mathfrak{M}_1$, avec $\mathfrak{s}$ et $\mathfrak{s}'$ des CA-Signaux. Cela est noté $\mathfrak{M}_1 \nleftrightarrow \mathfrak{M}_2$. La clôture transitive de l'interaction est notée $\nleftrightarrow^+$.

La conséquence des différentes compositions existant au sein du cadre discret est qu'il en va de même pour la dynamique modulaire :

Définition 83 (Dynamique modulaire fine discrète) La *dynamique modulaire fine discrète* est l'ensemble des modules et la description de leurs compositions séquentielles et interactions. Elle peut être représentée par un graphe orienté où les modules en interactions ont une arête non orientée les reliant. Elle est notée $\mathcal{X}$.

Enfin, deux dynamiques modulaires discrètes peuvent bien entendu être équivalentes :

Définition 84 (Équivalence de dynamique modulaire fine discrètes) Deux dynamiques modulaires fines discrètes $\mathcal{X}_1$ et $\mathcal{X}_2$ sont équivalentes s'il y a un isomorphisme f entre chaque éléments de $\mathcal{X}_1$ et $\mathcal{X}_2$, c'est-à-dire :

— $\forall \mathfrak{M}_1, \mathfrak{M}_2 \in \mathcal{X}_1, \mathfrak{M}_1 \dashrightarrow \mathfrak{M}_2 \Leftrightarrow f(\mathfrak{M}_1) \dashrightarrow f(\mathfrak{M}_2)$, et

— $\forall \mathfrak{M}_1, \mathfrak{M}_2 \in \mathcal{X}_1, \mathfrak{M}_1 \nleftrightarrow \mathfrak{M}_2 \Leftrightarrow f(\mathfrak{M}_1) \nleftrightarrow f(\mathfrak{M}_2)$.

La Figure 4.15 donne un exemple de dynamique modulaire fine discrète.

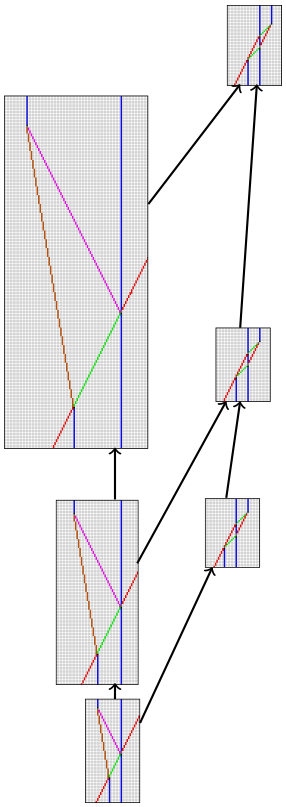


FIGURE 4.15 – Exemple de dynamique modulaire fine discrète.

Avec l'expression des ces deux dynamiques, il est alors possible d'exprimer les modalités d'équivalence continu/discret des dynamiques modulaires :

Définition 85 (Équivalence de dynamiques modulaires fines continues et discrètes) Une dynamique modulaire fine continue $\mathcal{D}$ et une dynamique modulaire fine discrète $\mathcal{X}$ sont équivalentes si les graphes les décrivant sont identiques, à un renommage près et si chaque nœud de ces graphes sont des méta-modules équivalents.

La notion d'équivalence des méta-modules n'est pas donnée ici, il s'agit en fait d'une correspondance donnée par l'utilisateur lors de la discrétisation : celui-ci associe à chaque méta-module continu un équivalent discret dans la dynamique qu'il souhaite discrétiser.

Chapitre 5

Discrétisation des modules et de leurs dynamiques

Ce chapitre traite de la technique de discrétisation utilisée pour transformer une dynamique de machine à signaux en une d'automate cellulaire. Cela passe d'abord par une conversion de la dynamique modulaire fine continue en une discrète. Par la suite, cette construction discrète est utilisée pour produire la configuration initiale, quelques états et règles de transition de l'automate cellulaire simulant la machine à signaux rationnelle dans laquelle la dynamique modulaire continue apparaît. Cela fait, la méthode de discrétisation brutale décrite à la Section 3.2 est alors utilisée pour créer le reste des règles et exécuter l'automate. L'utilisation d'une dynamique modulaire fine discrète correcte permet alors d'affirmer que cette discrétisation brutale se fera sans approximations.

Les techniques décrites ici ne s'appliquent que pour des dynamiques modulaires finies, c'est-à-dire que le nombre de modules les composant est fini. Divers cas d'extension seront traités dans le Chapitre 6. Tout au long de ce chapitre et du suivant, les machines manipulées sont considérées normalisées au sens de la sous-section 1.3.4.

5.1 Discrétisation de la dynamique modulaire continue

Le premier objectif de la discrétisation est d'obtenir la dynamique modulaire fine discrète à partir de celle continue. Cela se fait en cinq étapes :

1. créer les états et règles de transition basiques des AC-signaux à partir des méta-signaux utilisés de la MS à discrétiser pour assurer la propagation des signaux hors module,
2. calculer une *échelle de discrétisation* pour trouver la taille et la position des futurs modules discrets à créer,
3. trouver les méta-modules discrets associés à chacun des méta-modules continus utilisés par les modules de la dynamique modulaire fine continue,
4. créer les modules discrets en utilisant l'échelle puis vérifier leurs corrections par rapport aux méta-modules discrets dont ils sont des instanciations, si cette vérification est fausse, soit retourner au point 1 soit la discrétisation échoue, et enfin
5. composer ces modules pour obtenir la dynamique modulaire fine discrète équivalente à la continue d'origine.

Tout au long de ce chapitre, les différents procédés de discrétisation seront illustrés par des figures les résumant graphiquement. La Figure 5.1 représente une dynamique continue et son équivalent modulaire dont la discrétisation sera donnée en exemple.

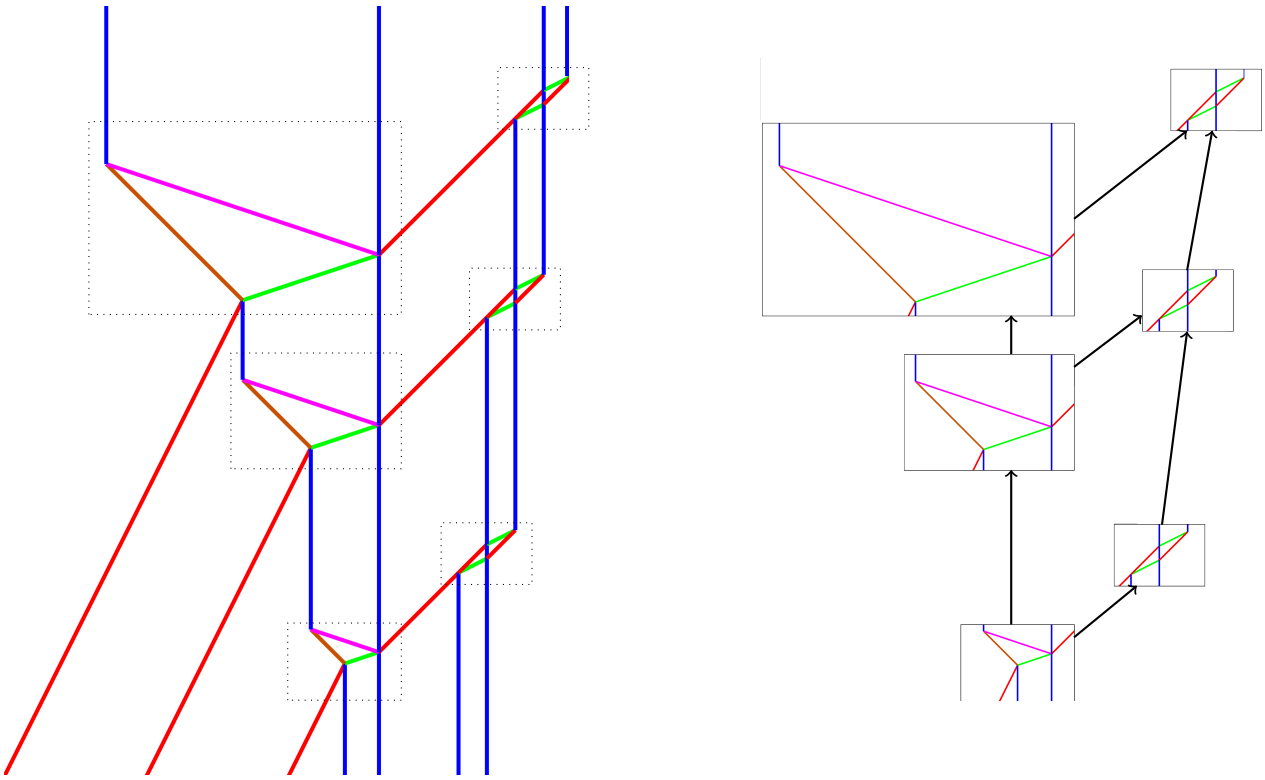


FIGURE 5.1 – Dynamique continue classique et son expression modulaire.

Dans l'expression modulaire, les modules ne sont reliés entre eux que par une seule « flèche ». Ces flèches expriment la présence d'une composition entre deux modules. Cette expression ne s'intéresse donc qu'à la représentation de la dynamique modulaire, à travers ses modules et ses compositions.

5.1.1 Création des états et règles de transition basiques

Il s'agit ici de créer l'ensemble Q_E des états ainsi que les règles de transitions associées pour produire des AC-signaux seulement capables de se déplacer dans le vide (c'est-à-dire entourés de cellules dans l'état quiescent), afin de les manipuler dans les modules, et ainsi créer leurs entrées et sorties. Cet engendrement se fait pour chacun des méta-signaux de la MS. La méthode de production de ces AC-signaux est celle de la sous-section 3.1.1. La Figure 5.2 illustre ce procédé sur l'exemple.

5.1.2 Calcul de l'échelle de discrétisation

La dynamique modulaire fine discrète à engendrer doit donner des coordonnées spatio-temporelles à chacun des modules discrets la composant. Pour cela, une échelle de discrétisation est utilisée : elle permet, à partir des coordonnées rationnelles des modules continus,

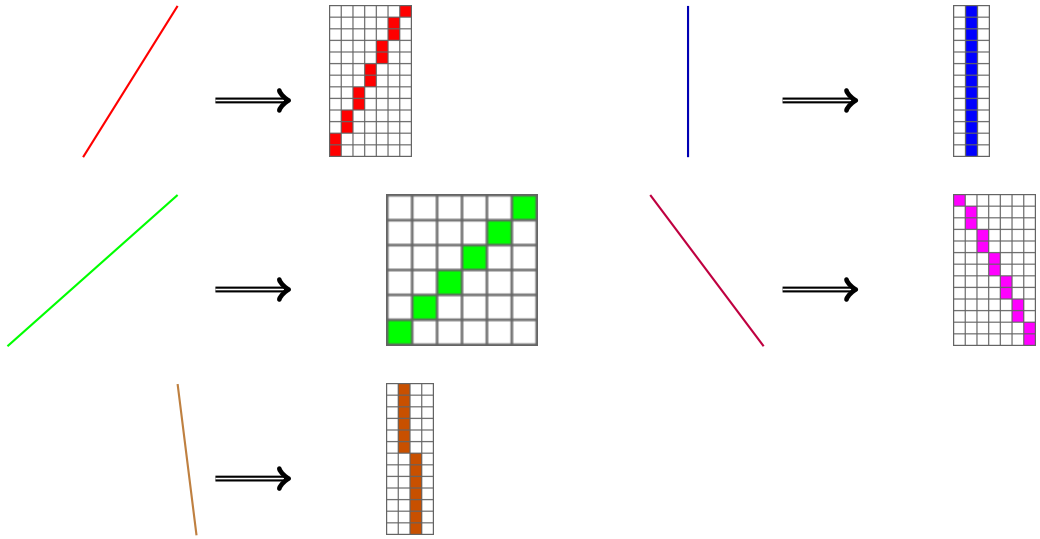


FIGURE 5.2 – États et règles de transitions basiques obtenus avec la MS de la Fig. 5.1.

de calculer celles des modules discrets ainsi que les positions des différents AC-signaux dans ces mêmes modules. Elle se définit comme suit :

Définition 86 (Échelle de discrétisation) Soit $\mathcal{D}$ la dynamique modulaire fine à discrétiser. L'échelle de discrétisation ω est un entier tel que $\forall M \in \mathcal{D}$:

- $\omega \cdot x_M^- \in \mathbb{Z}$
- $\omega \cdot t_M^- \in \mathbb{Z}$
- $\omega \cdot w_M \in \mathbb{Z}$
- $\omega \cdot d_M \in \mathbb{Z}$

De plus, pour tout signal s :

- si $s \triangleright M$, alors $\omega \cdot s_{x_\alpha} \in \mathbb{Z}$ et $\omega \cdot s_{t_\alpha} \in \mathbb{Z}$
- si $M \triangleright s$, alors $\omega \cdot s_{x_\beta} \in \mathbb{Z}$ et $\omega \cdot s_{t_\beta} \in \mathbb{Z}$.

Où les $(s_{x_\alpha}, s_{t_\alpha})$ et $(s_{x_\beta}, s_{t_\beta})$ sont les points d'entrées et de sorties d'un module tels que définit aux Définition 66 et 68. Enfin, ω est minimale.
Puisque toutes ces valeurs sont rationnelles, un tel entier existe toujours.

Informellement, ω est le plus petit commun multiple des dénominateurs minimaux positifs de toutes les positions et distances spatiales et temporelles dans la dynamique continue. Cela inclut les coordonnées des modules, leurs largeur et durée, ainsi que les positions des signaux entrants et sortants dans chacun d'entre eux.

Son utilisation est simple : les positions des éléments discrets sont calculés en multipliant celles du continu par cette échelle, assurant que toutes les positions ainsi obtenues sont entières. Cependant, elle n'empêche pas la création d'un module discret trop petit pour être utilisable.

Définition 87 (Module discret trop petit) Un module discret est dit *trop petit* si il contient un état qui représente un nombre de collisions dépassant un nombre entier donné par l'utilisateur (lors de la mise en place de la discrétisation brutale). Par la suite, ce terme est utilisé pour signifier un module discret dont la taille (largeur et/ou durée) peut faire échouer le procédé de discrétisation.

Ce cas est traité dans la sous-section 5.1.4.

Il est impossible de déterminer de manière automatique si un module discret sera trop petit ou non avant de le créer (puisque cela revient à prévoir le nombre de collisions exactes d'une machine à signaux, ce qui est incalculable [DL05]). Par conséquent, cette information est donné par le créateur des méta-modules discrets, à travers les conditions d'échelle.

À ce stade, il faut faire l'hypothèse suivante :

Hypothèse 88 *Les conditions d'échelle des méta-modules discrets sont correctes et expriment bien un calcul d'échelle suffisamment grand.*

Si ces conditions sont incorrectes, alors le risque est soit d'obtenir des modules trop petits pour être utilisés (causant une erreur de discrétisation), soit de boucler infiniment en ne trouvant jamais une échelle de discrétisation suffisamment grande.

5.1.3 Extraction des méta-modules discrets

Pour pouvoir produire la dynamique modulaire fine discrète, il est nécessaire d'identifier les méta-modules discrets qui la composeront. Pour ce faire, il faut utiliser les informations de discrétisation que contiennent chaque méta-module discret, comme indiqué dans la sous-section 4.1.2. Le principe est simple : pour chacun des modules composant la dynamique modulaire fine continue, récupérer le méta-module continu associé, puis rechercher le méta-module discret qui le discrétise. Cela permet alors d'associer un méta-module discret à chaque module continu et, à l'aide de l'échelle de discrétisation, de créer les modules discrets. La Figure 5.3 illustre ce procédé sur l'exemple.

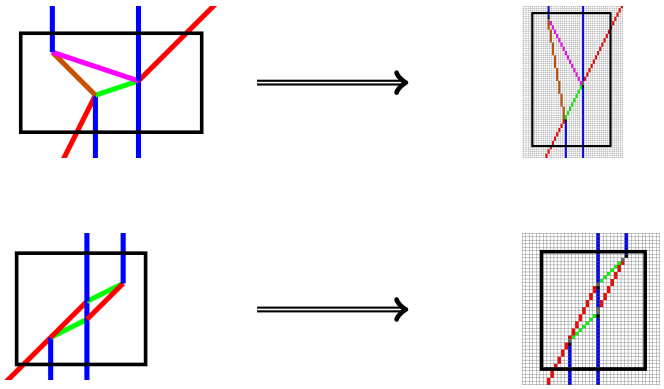


FIGURE 5.3 – Extraction des méta-modules discrets depuis la dynamique de la Fig. 5.1.

5.1.4 Création et vérification des modules discrets

Muni de l'échelle de discrétisation, des méta-modules discrets associés à chacun des modules continus et des états et règles basiques, il devient possible d'engendrer les modules discrets. La première étape est de calculer les coordonnées spatio-temporelles de ces modules, ainsi que leurs largeurs et durées. Cela se fait en multipliant par l'échelle de discrétisation celles des modules continus auxquels chacun des méta-modules discrets ont été associés par l'étape précédente. Formellement :

Algorithme 89 (Coordonnées des modules discret) *Soient M le module continu en cours de discrétisation, $\mathfrak{M}$ le module discrétisé en train d'être engendré et ω l'échelle de discrétisation, alors :*

- $\mathfrak{x}^-_{\mathfrak{M}} = \omega \cdot x^-_M$
- $\mathfrak{t}^-_{\mathfrak{M}} = \omega \cdot t^-_M$
- $\mathfrak{w}_{\mathfrak{M}} = \omega \cdot w$
- $\mathfrak{d}_{\mathfrak{M}} = \omega \cdot d$
- $\forall \mathfrak{s} \rhd \mathfrak{M}$ *et* $s \rhd M$ *tels que* $\mu \mathcal{R} \mathfrak{s}$ *avec* μ *le méta-signal associé à* s, $c^{t^-_M}_{\alpha} = c^{\mathfrak{t}^-_{\mathfrak{M}}}_{\omega \cdot s_{x_\alpha}}$
- $\forall \mathfrak{s} \lhd \mathfrak{M}$ *et* $s \lhd M$ *tels que* $\mu \mathcal{R} \mathfrak{s}$ *avec* μ *le méta-signal associé à* s, $c^{t^-_M}_{\beta} = c^{\mathfrak{t}^-_{\mathfrak{M}}}_{\omega \cdot s_{x_\beta}}$

Avec $c^{t^-_M}_{\alpha}$ *et* $c^{t^-_M}_{\beta}$ *respectivement les sites d'entrées et de sorties de* $\mathfrak{s}$.

L'utilisation de l'échelle telle que définie précédemment force les AC-signaux à être à la phase 0 lorsqu'ils sont sur les frontières (en entrée ou en sortie) d'un module. En effet, par sa définition, la multiplier par la position d'un signal continu donne une valeur entière. Or, dans les AC-signaux, la seule phase correspondant à une coordonnée entière est la phase 0. Par conséquent, lorsque les modules discrets sont créés, les AC-signaux sur les frontières sont mis sur la phase 0. La Figure 5.4 illustre ce procédé sur l'exemple.

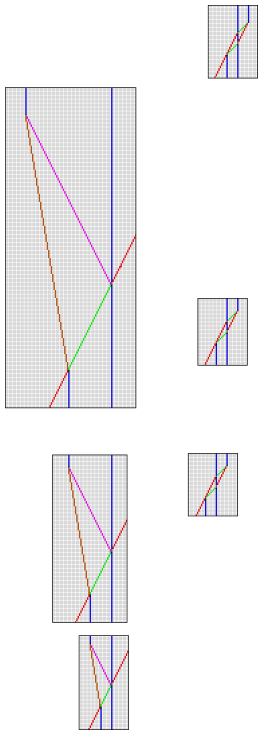


FIGURE 5.4 – Création des modules discrets depuis la Fig. 5.1 et des méta-modules de la Fig. 5.3.

Après la création d'un module, chacune de ses conditions sémantiques et syntaxiques est vérifiée. Si une condition est violée, le label de la condition est observée :

- si c'est une condition d'échelle, celle-ci est agrandie (multipliée par deux) et la discrétisation reprend depuis le début, donc avec un risque de boucler infiniment (avec une condition d'échelle qui n'est jamais satisfaite),

— si ce n'est pas une condition d'échelle, la discrétisation est considérée en échec (dans ce cadre cela signifie que les méta-modules discrets sont incorrects pour cette discrétisation).

Si toutes les conditions d'un module sont correctes, une fonction associant chacun de ses signaux entrants et sortants à un AC-signal est créée.

Définition 90 (Fonction d'appariement) La fonction d'association T d'une discrétisation modulaire est une fonction bijective de l'ensemble des signaux de la dynamique modulaire continue vers l'ensemble des AC-signaux de la dynamique modulaire discrète. Ainsi, $T(s) = \mathfrak{s}$ si s est discrétisé par $\mathfrak{s}$.

Elle sera utilisée par la suite pour composer les modules.

Lorsque tout les modules ont été créés et vérifiés et que la fonction associe tout les signaux inter-modulaires, la discrétisation peut passer à l'étape suivante.

5.1.5 Composition des modules discrets

Avec les modules discrets désormais engendrés, la dernière étape pour produire une dynamique modulaire fine est de les composer. En effet, pour le moment, ces modules sont tous positionnés dans le diagramme espace-temps indépendamment les uns des autres.

Algorithme 91 (Composition des modules discrets) *Soient M_1 et M_2 des modules continus. Soient $\mathfrak{M}_1$ et $\mathfrak{M}_2$ les modules discrets engendrés correspondants. $\forall s$ tel que $M_1 \triangleright s \triangleright M_2$, une composition telle que $\mathfrak{M}_1 \triangleright T(s) \triangleright \mathfrak{M}_2$ est créée.*

Ainsi, pour chacune des compositions dans la dynamique modulaire fine continue, son équivalent discret est créé en utilisant la fonction d'appariement T. La Figure 4.15 est le résultat de ce procédé sur l'exemple.

5.2 Construction de la configuration initiale

Avec la dynamique modulaire fine discrète produite, il est désormais possible de construire une configuration initiale. Cette constructions se fait en deux étapes :

1. extraire les AC-signaux dans la configuration initiale, puis
2. remonter ces AC-signaux pour obtenir leurs positions dans la configuration initiale.

5.2.1 Extraction des AC-signaux de la configuration initiale

La première étape est d'identifier les AC-signaux se trouvant dans la configuration initiale, afin de pouvoir la créer. Identifier ces AC-signaux n'est pas difficile avec l'aide de la dynamique modulaire fine discrète. En effet, avec la connaissance de l'ensemble des compositions de modules, il est possible de retrouver ceux qui appartiennent à la configuration initiale grâce au Lemme suivant :

Lemme 92 *Soit $\mathfrak{s}$ un AC-signal de la dynamique modulaire fine discrète. Si $\mathfrak{s} \triangleright \mathfrak{M}_1$ et $\nexists\, \mathfrak{M}_2$ tel que $\mathfrak{M}_2 \triangleright \mathfrak{s} \triangleright \mathfrak{M}_1$, alors $\mathfrak{s} \in c_0$. C'est-à-dire que tout AC-signal entrant dans un module et n'appartenant pas à une composition provient de la configuration initiale.*

Preuve. Le fait qu'un AC-signal entre dans un module indique qu'il a été émis quelque part, mais comme il n'appartient à aucune composition, cela signifie aussi qu'aucun module n'est responsable de sa création. Par conséquent et par l'Hypothèse 50 (qui suppose qu'il n'y a pas de collisions en dehors des modules), il est obligatoirement présent depuis le début du calcul, donc dans la configuration initiale. □

À noter que les signaux n'entrant ni ne sortant d'un module sont aussi dans la configuration initiale. Cependant, comme ils ne participent pas à la dynamique modulaire, leur présence est négligée.

5.2.2 Création de la configuration initiale

Avec les AC-signaux appartenant à la configuration initiale identifiés, et puisque leurs positions dans les modules et dans le diagramme espace-temps sont connus (la dynamique modulaire fine discrète étant ancrée), engendrer la configuration initiale devient alors simple.

Lemme 93 (Configuration initiale discrète) *Soit* $\mathfrak{s} \in c_0$ *un AC-signal de vitesse* v *et soit* c_x^t *le premier site d'entrée de* $\mathfrak{s}$ *dans un module. La position spatiale de* $\mathfrak{s}$ *dans la configuration initiale est* $\lfloor x + (t \cdot -v) \rfloor$. *L'état à y mettre est celui obtenu en remontant dans le temps les itérations de* $\mathfrak{s}$.

Preuve. Pour obtenir la position d'un AC-signal dans la configuration initiale, il suffit de « remonter » chacun de ces AC-signaux dans le temps, jusqu'à atteindre l'étape temporelle 0 (la première). Il s'agit donc de calculer le déplacement d'un AC-signal allant à rebrousse-temps, et pour cela, il suffit d'appliquer l'inverse de la formule de calcul de déplacement d'un signal. Le résultat est arrondi afin d'assurer que c'est une valeur entière, la position cherchée étant celle d'un automate cellulaire et donc appartenant à $\mathbb{Z}$. La phase de l'état de ces AC-signaux se retrouve de la même manière (en remontant les signaux depuis leurs entrées dans un module). □

La Figure 5.5 illustre ce procédé sur l'exemple et rappelle la configuration initiale continue d'origine.

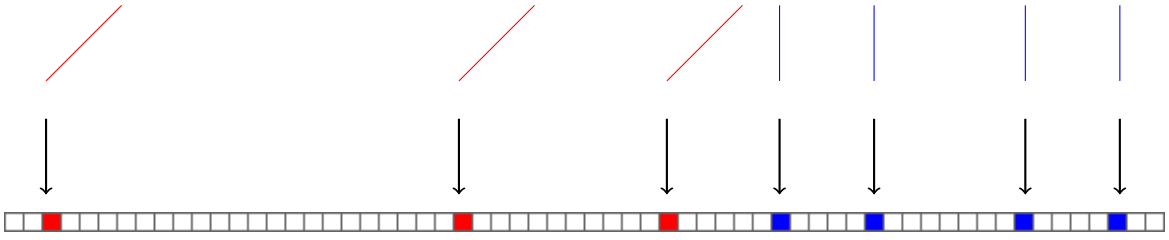


FIGURE 5.5 – Configuration initiale extraite de la dynamique modulaire fine de la Fig. 4.15.

5.2.3 Finalisation de la discrétisation

Une fois la configuration initiale obtenue, il ne reste plus qu'à finaliser la discrétisation de la machine à signaux en utilisant la méthode de discrétisation brutale de la Section 3.2.

La Figure 5.6 est le résultat de tout les traitements précédents, et fournit l'équivalent discret de la dynamique du début de la section précédente.

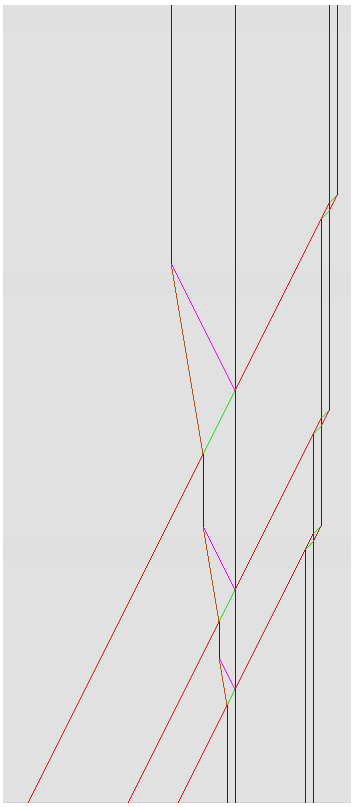


FIGURE 5.6 – Résultat de la discrétisation modulaire de la Figure 5.1.

5.3 Correction de la discrétisation

Le processus de discrétisation étant établi, il faut prouver sa correction. Cette preuve repose sur des hypothèses prises à la fois sur les méta-modules fournis, mais aussi sur la dynamique modulaire fine continue. De ce fait, elle s'appuie donc sur la supposition de correction des données externes au système, et si ce n'est pas le cas, la discrétisation peut être incorrecte ou impossible.

5.3.1 Correction de la dynamique modulaire fine discrète

Cette sous-section s'intéresse à montrer que la dynamique modulaire fine discrète engendrée par les étapes de la Section 5.1 est équivalente à la continue d'origine, au sens de la Définition 85. Elle s'intéresse tout d'abord à la création des modules discrets et à leur positionnement, puis à leurs compositions.

Création des modules discrets et utilisation de l'échelle

La preuve de correction des modules discrets engendrés repose sur l'hypothèse suivante :

Hypothèse 94 *Les méta-modules discrets donnés sont bien des discrétisation des méta-modules continus auxquels ils sont associés.*

En effet, si ce n'est pas le cas, les modules créés en utilisant ces méta-modules seront eux-mêmes faux, entraînant avec eux toute la dynamique modulaire fine discrète engendrée par la suite.

Si cette hypothèse est vraie, alors toute instance d'un de ces méta-modules discrets est une discrétisation d'une instance de son méta-module continu associé, *si le discret a une taille, une durée et une position spatio-temporelle correcte et suffisante*. Ces valeurs sont calculées à l'aide de l'échelle de discrétisation, donc prouver leur correction revient à montrer que le procédé se basant sur celle-ci est correct. Par sa méthode de calcul, l'échelle de discrétisation est une valeur entière positive.

Lemme 95 *Soient $\mathcal{D}_1, \mathcal{D}_2$ des dynamiques modulaires fines continues et ω un entier strictement positif. Si $\forall M_1 \in \mathcal{D}_1, \exists! M_2 \in \mathcal{D}_2$ tel que :*

- $\bar{t_{M_2}} = \omega \cdot \bar{t_{M_1}}$
- $\bar{x_{M_2}} = \omega \cdot \bar{x_{M_1}}$
- $d_{M_2} = \omega \cdot d_{M_1}$
- $w_{M_2} = \omega \cdot w_{M_1}$
- $\forall s_1 \triangleright M_1 ! \exists s_2$ *tels que* $s_2 \triangleright M_2$ *et* $s_{2_\alpha} = \omega \cdot s_{1_\alpha}$
- $\forall s_1 \triangleleft M_1 ! \exists s_2$ *tels que* $s_2 \triangleleft M_2$ *et* $s_{2_\beta} = \omega \cdot s_{1_\beta}$

et pour tout $M_2 \in \mathcal{D}_2$, il existe un tel $M_1 \in \mathcal{D}_1$, alors $\mathcal{D}_1$ est équivalente à $\mathcal{D}_2$ (au sens de la Définition 79. L'équivalence de dynamiques est donc indépendante de la nature des modules.

En d'autres termes, si les tailles, durée, positions spatio-temporelles et distance entre les signaux de chaque module de $\mathcal{D}_2$ sont des multiples de celles de $\mathcal{D}_1$ et d'un seul entier, alors les dynamiques sont équivalentes.

La preuve se découpe en deux parties :
- les dynamiques des modules sont individuellement conservées, et
- les compositions des modules restent correctes et dans le même ordre.

Preuve. Soit c_M la configuration initiale d'un module continu. c_M est donc une configuration de machine à signaux. En la prenant comme configuration initiale de machine, il est possible d'affirmer que multiplier toutes les positions des signaux dans cette configuration par un entier ne modifie pas la dynamique (comme dit dans la sous-section 1.3.4). Par conséquent, multiplier les positions de tous les signaux de c_M par un même entier n'altère pas la dynamique de M. En appliquant ce principe à tous les modules composant une dynamique modulaire continue, alors la dynamique de chacun est individuellement conservée.

Il reste désormais à montrer la correction des compositions.

Les signaux d'entrée et de sortie ayant tous été déplacés sur les frontières des modules par un même entier, les compositions restent correctes et dans le même ordre. En effet, soit s un signal participant à une composition $M_1 \to M_2$ avec (x_0, t_0) ses coordonnées de sortie de M_1 et (x'_0, t'_0) ses coordonnées d'entrée dans M_2 dans la dynamique modulaire continue d'origine et μ son méta-signal associé. Dans ce cas, et puisqu'il n'y a pas de collision en dehors des modules (par l'Hypothèse 50), $x'_0 = x_0 + S(\mu) \cdot (t'_0 - t_0)$. Maintenant,

en multipliant toutes les coordonnées par un entier ω, le nouveau point de sortie de s dans M_1 est $(\omega \cdot x_0, \omega \cdot t_0)$ et celui d'entrée dans M_2 est $(\omega \cdot x'_0, \omega \cdot t'_0)$. Or, si $x'_0 = x_0 + S(s) \cdot (t'_0 - t_0)$, alors $\omega \cdot x'_0 = \omega \cdot x_0 + \omega \cdot S(s) \cdot (t'_0 - t_0)$ car $\omega \cdot x_0 + \omega \cdot S(s) \cdot (t'_0 - t_0) = \omega \cdot (x_0 + S(s) \cdot (t'_0 - t_0))$. Donc en conservant les mêmes signaux mais juste en multipliant leurs positions, les sorties et entrées de modules restent connectées. De plus, comme dit précédemment, l'ordre des signaux reste le même sur les frontières de sorties et les frontières d'entrées des modules, donc l'ordre des compositions reste aussi identique.

Puisque les dynamiques dans les modules restent les mêmes, que les compositions restent dans le même ordre et qu'elles restent correctes dans les entrées et sorties des modules, alors la dynamique continue d'origine et celle obtenue par multiplication par une échelle sont équivalentes. □

Il est donc possible « d'agrandir » une dynamique modulaire fine continue sans la modifier. En faisant cette procédure avec l'échelle de discrétisation calculée telle que dans la sous-section 5.1.2, toutes ces positions ont alors des valeurs entières (puisque ω est le plus petit commun multiple des dénominateurs de celles-ci). À ce moment, la discrétisation créé et place les modules discrets sur ces mêmes positions. Ce procédé conserve la dynamique modulaire fine par le Lemme suivant :

Lemme 96 *Soient $\mathcal{D}$ une dynamique modulaire fine continue et $\mathcal{X}$ une dynamique modulaire fine discrète. Si les largeurs, durées et positions des modules ainsi que les positions des signaux de $\mathcal{D}$ sont les mêmes valeurs entières que celles de $\mathcal{X}$ et qu'aucun des modules de $\mathcal{X}$ n'est trop petit, alors ces deux dynamiques modulaires sont équivalentes.*

Preuve. Si les modules et les compositions de chacune des dynamiques sont aux mêmes emplacements et qu'il n'y a pas de module discret trop petit, alors les graphes produits à partir de chacune de ces dynamiques sont identiques. En effet, les modules étant les nœuds et les compositions les arêtes, et comme ils sont positionnés aux mêmes endroits dans les deux dynamiques modulaires, les sommets du graphe sont dans le même ordre et reliés de la même manière. Ils sont donc les mêmes, à renommage près. Cela correspond à la Déf. 85 donc les deux dynamiques sont équivalentes. □

Tout cela permet de prouver le théorème suivant :

Théorème 97 *Le procédé de discrétisation de la Section 5.1 produit une dynamique modulaire fine discrète équivalente à la dynamique modulaire fine continue d'origine.*

Preuve. D'après l'Hypothèse 94, les modules discrets instanciant les méta-modules continus sont bien des discrétisations, ils sont donc individuellement corrects. De plus, d'après le Lemme 95, il est possible de placer une dynamique modulaire continue sur des valeurs entières uniquement et ce sans l'altérer. Enfin, le Lemme 96 permet d'assurer qu'une dynamique modulaire fine discrète placée sur ces nouvelles positions est équivalente à celle continue d'origine. Donc, la dynamique modulaire fine discrète produite par le procédé de la Section 5.1 est bien équivalente à celle continue d'origine. □

Il reste maintenant à démontrer que la construction de la configuration initiale de l'automate cellulaire permet bien d'engendrer un diagramme espace-temps dont le comportement simule celui de la machine à signaux d'origine.

5.3.2 Correction de la discrétisation

L'extraction des AC-signaux appartenant à la configuration initiale est prouvée dans la sous-section 5.2.1 à travers le Lemme 92. Il reste alors à montrer que l'utilisation de la discrétisation brutale dans ce cadre permet bien d'obtenir une discrétisation correcte dans ce cas d'une exécution « idéale » d'une MS.

Correction de la discrétisation brutale

L'échelle de discrétisation modulaire assure que les AC-signaux en entrées et sorties des modules sont à la phase 0 lorsqu'ils croisent une frontière. De plus, si les méta-modules discrets donnés sont corrects, la dynamique interne d'un module discret conduit toujours au résultat indiqué par son méta-module associé. Par conséquent, cela signifie que l'entrée des AC-signaux à la phase 0 mènera toujours à des sorties à la phase 0. De ce fait, les collisions discrètes ayant émises les AC-signaux sortants représentent des collisions continues se situant de telle manière que les signaux qu'elles émettent se positionnent toujours sur une des positions possibles de la discrétisation brutale (le pas de discrétisation, voir Déf. 42). Une récurrence directe permet d'affirmer que si une collision est placée ainsi, alors celles dont elle est issue le sont aussi, et ainsi de suite jusqu'aux signaux représentés par les AC-signaux en entrée du module. Donc, la discrétisation brutale n'appliquera jamais sa méthode d'approximation (puisque tous les signaux sont sur les pas de discrétisation de la Déf. 42) et sera exacte. Finalement, puisque la configuration initiale est bien un équivalent discret à celle d'origine et que la discrétisation brutale n'approxime pas, le diagramme espace-temps discret obtenu est bien équivalent au continu d'origine.

Chapitre 6

Cas particuliers de discrétisation

Ce chapitre propose des méthodes de discrétisation pour trois structures particulières pouvant apparaître dans les machines à signaux. Ces trois cas présentent tous le point commun de nécessiter d'exprimer une dynamique modulaire contenant un nombre infini de modules. Pour chacune d'entre elles, sa nature et une méthode de discrétisation sont décrites.

6.1 Simulation de machines de Turing

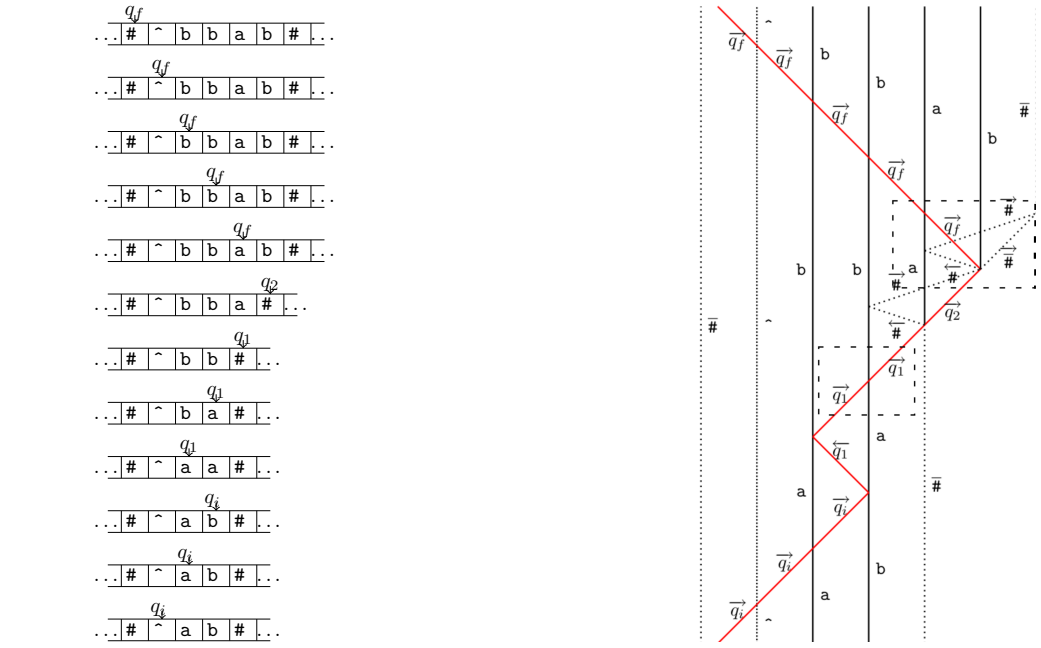
La première construction présentée ici est un ensemble infini de machines qui ont la particularité de simuler le comportement d'une machine de Turing. Dans [DL11], l'auteur propose une méthode de simulation simple, qui suit en fait le mouvement de la tête de lecture. En cherchant à y appliquer la modularité, cela produit un nombre potentiellement infini de modules, mais toujours un nombre fini de méta-modules. Par exemple, la Table 6.1 (Q est l'ensemble des états et Γ l'ensemble des symboles de la machine de Turing) donne les règles de la machine de Turing dont le résultat de la simulation est montré par la Figure 6.1. Dans cette figure, les vitesses des signaux rouge (ceux représentants le déplacement de la tête de lecture) sont 1 et -1, et celles des signaux agrandissant la bande sont 3 et -3.

Table 6.1 – Règles, états et symboles de la machine de Turing.

δ	^	a	b	#
q_i	q_i,^,→	q_i,a,→	q_1,a,←	-
q_1	-	q_1,b,→	-	q_2,a,→
q_2	-	-	-	q_f,b,←
q_f	q_f,^,←	q_f,a,←	q_f,b,←	-

$Q = \{q_i, q_1, q_2, q_f\}$
$\Gamma = \{\text{^}, \text{a}, \text{b}, \#\}$
"–" signifie non définit

Ces constructions sont purement séquentielles (un module après l'autre), et peuvent n'utiliser que des modules très simples (une seule collision, quatre signaux). Même les diagrammes les plus riches qui apparaissent dans ce cadre restent réguliers et faciles à exprimer. En effet, le module le plus complexe qu'elles peuvent contenir (celui qui exprime l'extension de la bande, illustré dans la Fig. 6.1(b)) ne contient que cinq collisions effectives, et toutes se passent à des emplacements fixes et prévisibles. Le problème est que leurs diagrammes espace-temps sont potentiellement infinis (tout comme peut l'être le



(a) exécution d'une machine de Turing (b) sa simulation

FIGURE 6.1 – Simulation de machine de Turing.

calcul d'une machine de Turing) et que cela n'est pas prédictible. Il n'est donc pas toujours possible de l'intégrer en entier dans une dynamique modulaire fine continue telle que celle de la Déf. 78 puisqu'elle requiert un nombre infini de modules. Cependant, la modularité est utilisée ici dans le but de montrer la structuration régulière des diagrammes simulant des machines de Turing, afin d'en déduire l'exactitude de la discrétisation brutale. Pour ce faire, une expression modulaire régulière de ces dynamiques est donnée, et sa discrétisation est prouvée exacte. Cela fait émerger la structuration naturelle de ces diagrammes, permettant d'affirmer l'exactitude de la discrétisation brutale seule.

6.1.1 Expression modulaire

En premier lieu, ces diagrammes sont construits de sorte à ce que l'espace entre deux signaux verticaux adjacents soit toujours le même. En effet, ils sont placés ainsi dans la configuration initiale, et le procédé d'agrandissement de la bande produit aussi un signal vertical respectant cette propriété. De cela découle l'emplacement régulier des collisions : puisque chaque collision non blanche contient un signal vertical entrant (et sortant), elles sont toutes situées à intervalle régulier, avec toujours la même distance entre deux adjacentes. De plus, les signaux représentants la tête de lecture sont tous de la même vitesse absolue (1 et -1), ce qui permet en plus que ces collisions se passent à intervalle de temps régulier. En effet, après une collision, la prochaine sera entre le signal vertical spatialement suivant ou précédent et le signal de la tête de lecture, et puisque l'espace entre deux signaux verticaux est toujours le même, la durée entre deux collisions aussi. La seule exception à cette règle provient des modules agrandissant la bande. De fait, dans ce cadre, l'intervention de signaux de vitesses absolues plus grandes (3 et -3) est nécessaire et al-

tère donc la régularité des collisions. Cependant, ces modules émettent des signaux inscrits sur la structure régulière (ils respectent l'espace fixe entre deux signaux verticaux et les emplacements des collisions). Par conséquent, ces diagrammes simulant des machines de Turing ne manipulent que 6 méta-modules différents, dont 4 ne contiennent qu'une seule collision. De plus, ces modules sont placés à intervalle régulier (en considérant que les signaux verticaux sont sur des positions entières, la distance entre deux modules l'est aussi), à la fois spatialement et temporellement, puisqu'il sont placés soit autour des collisions uniques sur les signaux verticaux, soit autour du groupe de collisions permettant l'agrandissement de la bande. Comme ces collisions uniques ou zones d'agrandissement sont à des positions spatio-temporelles régulières, les modules les contenant peuvent aussi être positionnés ainsi. Cela permet donc de donner la possibilité aux modules mono-collision de tous être de la même largeur. Il en va de même pour les modules d'agrandissement, qui peuvent être tous de la même largeur mais forcément plus grande que celle des modules mono-collision (au moins 3 fois plus). La Fig. 6.2 donne un exemple d'expression modulaire d'une simulation de machine de Turing.

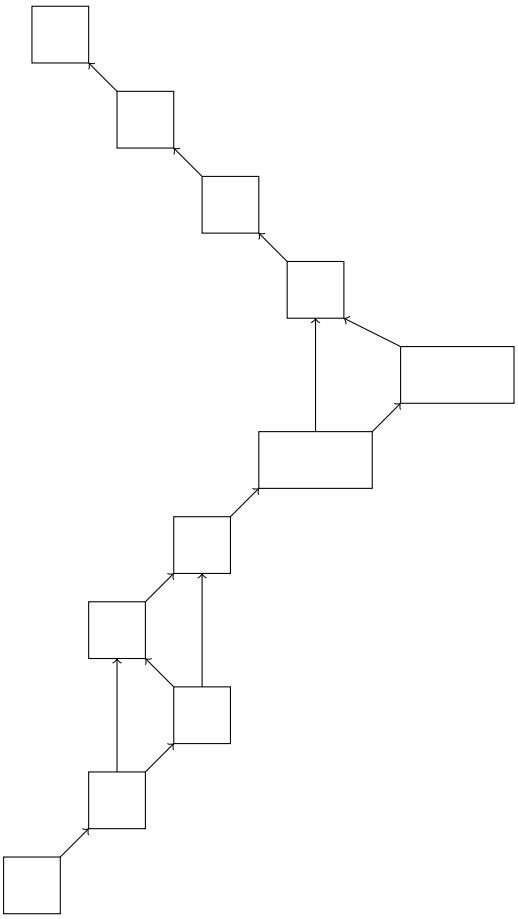


FIGURE 6.2 – Expression modulaire d'une simulation de machine de Turing.

Il est donc possible d'exprimer le diagramme espace-temps d'une machine à signaux simulant une machine de Turing sous la forme d'une dynamique modulaire continue. De plus, cette expression est régulière, au sens de la position spatio-temporelle des modules. Cette régularité permet d'affirmer que tous les modules peuvent avoir une taille entière

et être placés sur des positions entières. Cela permet donc d'affirmer que la discrétisation modulaire sera toujours exacte, puisque les largeurs fixes et entières des modules assurent l'inexistence de structures purement continues (fractales).

6.1.2 Exactitude de la discrétisation brutale

Mieux encore, les positions entières des modules dans la dynamique modulaire assurent que les positions des AC-signaux dans les modules sont eux aussi entiers, à la fois dans l'entrée et la sortie. Par conséquent, les phases des AC-signaux restent toujours correctes, dans le sens où aucun décalage du type de l'exemple présenté dans la Fig. 3.9 ne peut arriver. De plus, le choix de deux tailles possibles pour ces modules rend caduque l'utilisation d'une échelle (puisque ceux-ci seront forcément d'une des ces deux tailles). Par conséquent, puisque les modules sont à des positions entières et à des tailles fixes, tout le processus de discrétisation modulaire (le calcul de l'échelle et le positionnement des modules) est inutile. Il ne reste donc que la discrétisation brutale qui produit la discrétisation. Donc, comme la discrétisation est correcte et que seule la discrétisation brutale est utilisée, celle-ci peut être utilisée en dehors des modules et son résultat sera exact. Cela est illustré par la Fig. 6.3 qui donne le résultat de la discrétisation brutale seule sur une machine à signaux simulant une machine de Turing.

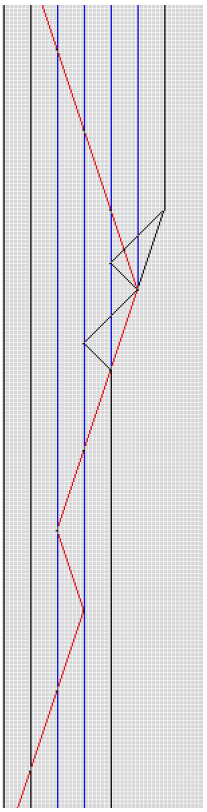


FIGURE 6.3 – Discrétisation d'une simulation de machine de Turing.

Sur cette figure les collisions correspondent aux groupes de deux cellules noires. Ces deux cellules ne représentent en fait qu'une seule et unique collision. Elle apparaissent

ainsi car la collision à lieu exactement sur la bordure d'une cellule, donc dans les deux en même temps. Cette « erreur » est purement liée à l'affichage, la discrétisation est donc tout à fait correcte.

6.2 Structure fractale « zig-zag »

Le deuxième cas particulier de discrétisation est une structure fractale dans laquelle une infinité de signaux et de collisions existent dans un espace et un temps fini. Ce type de structure porte le nom « d'accumulation », l'idée étant qu'une infinité de signaux et collisions « s'accumulent » dans un espace limité. L'existence de ce cas précis d'accumulation nécessite quatre méta-signaux μ_1, μ_2, μ_3 et μ_4 dont les vitesses sont sans importance du moment qu'elles respectent la condition : $S(\mu_4) < S(\mu_2) < S(\mu_3) < S(\mu_1)$.

La machine capable d'engendrer cette structure doit aussi avoir des règles de collision bien précises : $\{\mu_1, \mu_2\} \rightarrow \{\mu_2, \mu_4\}$ et $\{\mu_3, \mu_4\} \rightarrow \{\mu_1, \mu_3\}$.

Enfin, un diagramme espace-temps doit contenir à un moment une configuration de type μ_3 puis (μ_1 ou μ_4) puis μ_2 pour y voir apparaître une fractale « zig-zag ». Ce diagramme contient alors cette structure, illustrée par la Fig. 6.4.

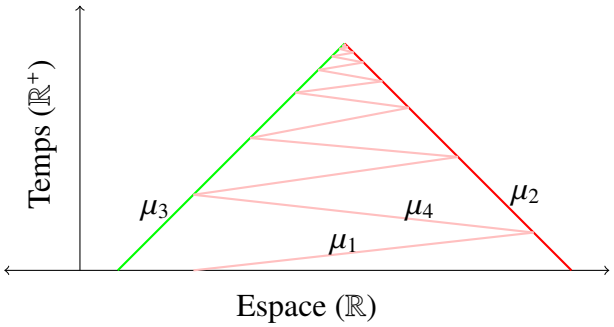
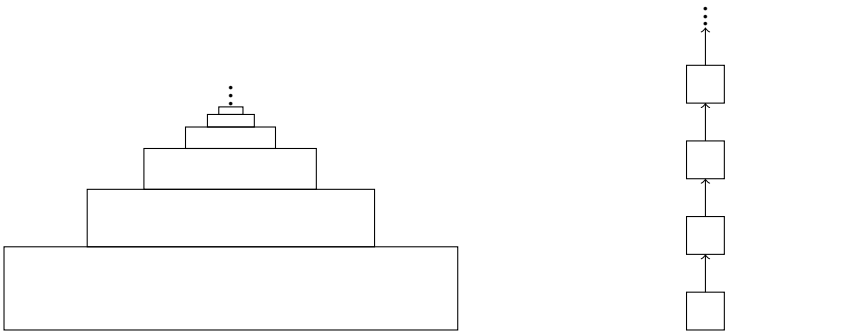


FIGURE 6.4 – Structure fractale « zig-zag ».

Pour l'expliquer simplement, il s'agit de « deux » signaux « rebondissant » entre « deux » autres se rapprochant. Plus le temps avance, plus les deux signaux extérieurs se rapprochent, et plus le nombre de collisions et de signaux est grand, tout cela dans un espace de plus en plus restreint. Ces rebonds se répètent à l'infini, cette structure produit donc une infinité de signaux et de collisions dans un espace et un temps fini (cette fin est marquée par la collision d'une infinité de signaux extérieurs). Il est alors évident qu'il est impossible de discrétiser une telle construction ou alors il faut définir plus avant ce qui est attendu d'un tel phénomène. Une telle structure régulière semble pouvoir s'exprimer facilement par des modules : il s'agit d'instances du même méta-module qui se répètent à l'infini avec un élément symétrique, comme la Figure 6.5 le montre.

Cependant, il est impossible de l'intégrer en entier à une dynamique modulaire fine continue telle que celle de la Déf. 78 puisqu'elle requiert un nombre infini de modules. Par conséquent, il est nécessaire de définir une nouvelle façon d'exprimer ce type de construction, et de l'intégrer aux dynamiques modulaires. La solution choisie ici est d'exprimer cet enchaînement de modules par induction : le premier module (le plus bas) est conservé, tout le reste en découle.



(a) découpage modulaire (b) dynamique modulaire

FIGURE 6.5 – Expressions modulaires de la structure « zig-zag ».

6.2.1 Proto-module

Cette nouvelle structure utilisée dans la dynamique modulaire porte le nom de *proto-module*. C'est un module particulier utilisé pour décrire une construction précise, un ensemble de modules tous issus du même méta-module et qui s'enchaînent à l'infini (ou un nombre de fois donné). Cela permet donc de décrire les structures du types de la fractale « zig-zag » avec un nombre fini d'éléments. Un proto-module est aussi associé à un nombre entier k représentant le nombre minimal d'itérations assurées lors de la discrétisation.

L'objectif est d'en faire une construction discrétisable dans le cadre du procédé modulaire. Cette discrétisation est clairement impossible à faire de manière exacte, puisque les proto-modules peuvent exprimer une structure fractale (comme celle à l'étude ici). Ils doivent donc pouvoir être associés à une *approximation* donnée par l'utilisateur, qui va permettre le traitement automatique d'un cas particulier. Le processus de discrétisation va donc devoir progresser autant que possible de façon exacte, puis lorsqu'il rencontre un problème, utiliser cette approximation donnée pour discrétiser la « pointe » de la strcture « zig-zag ».

6.2.2 Discrétisation

La structure « zig-zag » discrétisée se termine toujours comme cela dans l'automate cellulaire : une cellule (quelque part dans la pointe) finit par contenir trop de collisions et fait donc échouer le processus. Pour discrétiser et approximer lorsque le besoin apparaît, deux structures discrètes sont associées à chaque proto-modules « zig-zag ». La première, D_1, est un méta-module discret normal, qui va simplement représenter les modules se répétant au moins jusqu'à ce qu'un certain nombre d'itérations (dont la valeur est donnée dans le proto-module par k) soit atteint. La deuxième, D_2, est un méta-module discret qui est utilisé si au moins k modules de type D_1 ont été engendrés et que celui en train d'être ajouté viole une condition de taille (il est trop petit). Ce méta-module D_2 est celui qui contient la donnée d'approximation, sous la forme de la règle de transition lorsqu'une ou plusieurs cellules contiennent trop de collisions. L'application du résultat de ce problème donné dans le méta-module D_2 permet donc de continuer la discrétisation automatique en s'appuyant sur le choix fait par l'utilisateur.

La discrétisation d'une dynamique contenant un proto-module se déroule comme celle n'en contenant pas, à une exception près. Cette exception est l'arrivée du proto-module qui est traité de manière particulière :

— tout d'abord, k modules de type D_1 sont engendrés et composés séquentiellement (comme le sont tous les autres modules),
— après cela, cette création continue mais cette fois, si une condition de taille est violée, la construction de modules D_1 est stoppée et un module de type D_2 est engendré et composé avec la dernière itération de D_1 et enfin,
— la lecture de D_2 va permettre d'enregistrer les règles de transition contenant l'état « trop de collisions » dans une cellule.

Pour n'utiliser la règle de trop de collisions que pour cette structure, l'état qui le marque est différent de celui pour le reste du diagramme, et donc sa règle de transition aussi. Cela permet d'avoir plusieurs constructions de ce type dans le même diagramme avec des approximations différentes à chaque fois.

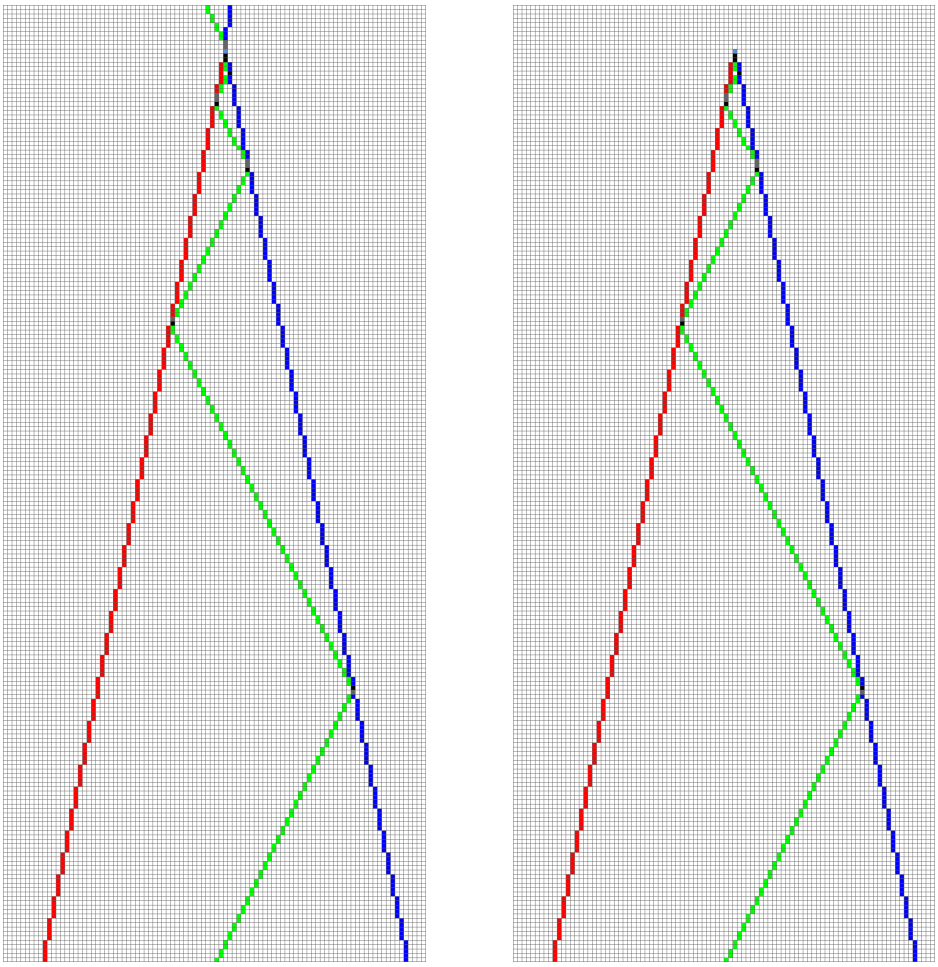
Avec ce procédé, la discrétisation est correcte jusqu'à la première apparition d'un module de type D_2. En effet, jusqu'à ce moment, elle se comporte exactement comme n'importe quelle autre, et donc la preuve de correction précédente peut être utilisée. Après ce moment, la discrétisation approxime et donc ne se comporte plus exactement comme la version continue. Par conséquent, le résultat est un automate cellulaire (et un diagramme espace-temps discret) « proche » (au sens de l'utilisateur) du continu, mais inévitablement incorrect. Cela dit, cette technique permet de proposer une méthode pour discrétiser ces structures « zig-zag », comme l'illustre la Fig. 6.6.

Dans cette discrétisation, le sommet de la structure peut produire deux signaux μ_3 et μ_4 ou rien du tout (qui est le résultat par défaut). Il s'agit ici d'un choix lors de la discrétisation : en effet, dans les machines à signaux, le *point d'accumulation* (le point vers lequel converge les signaux et collisions) a une sortie indéfinie. Cependant, dans le cadre d'une discrétisation et pour les automates cellulaires, il est nécessaire qu'un calcul y ait lieu. L'utilisateur a donc la possibilité d'exprimer ce qu'il souhaite voir se passer dans la « pointe » de la structure, à travers l'expression du résultat du module de type D_2.

6.3 Structure fractale « division par deux »

Le dernier cas étudié ici est une autre structure fractale dont la nature est différente de la précédente. Sa construction est plus complexe de même que sa forme : plutôt que d'avoir la même structure qui se répète, elle se multiplie à chaque étape et l'ensemble des structures répétées est toujours contenu dans le même espace. Ainsi, les signaux se dupliquent infiniment dans un espace fini, prenant de moins en moins de place. Par conséquent, elle est impossible à discrétiser exactement. Cette structure est en fait l'équivalent continu d'une des solutions au problème de synchronisation d'escouade (Firing Squad Synchronization Problem ou FSSP en anglais), et son étude est motivée par cette raison : réussir à la discrétiser automatiquement revient à engendrer automatiquement une solution à ce problème. La Table 6.2 donne les méta-signaux et règles de collisions nécessaires à l'apparition de cette structure, et elle est illustrée par la Fig. 6.8(a).

L'idée est ici que l'espace entre deux signaux verticaux est divisé par deux et cette division est marquée par un nouveau signal vertical. Puis les espaces entre signaux verticaux créés par l'ajout de ce nouveau signal sont à nouveau divisés par deux, et ainsi de



(a) discrétisation avec émission (b) discrétisation sans émission

FIGURE 6.6 – Discrétisation d'une structure zig-zag.

suite, infiniment (puisque les machines à signaux dans le cadre de cette thèse travaillent sur les rationnels, cet espace peut-être infiniment divisé).

Ici, l'expression modulaire est plus complexe : il s'agit de deux méta-modules (un pour le comportement à gauche, un pour celui à droite) dont des itérations se suivent dans un certain ordre (gauche puis droite). À chaque nouvelle répétition de la construction, il y a deux fois plus de modules que dans la précédente. Cette expression modulaire est illustrée par la Fig. 6.7. Cependant, le concept du proto-module peut aussi y être utilisé, mais en altérant son procédé de discrétisation pour l'adapter à la structure de division. Le proto-module continu reste alors de même nature qu'un méta-module classique avec un compteur d'itérations k qui garantit un nombre minimal d'itérations exactes et s'inscrit de la même manière dans une dynamique modulaire continue.

6.3.1 Discrétisation

Pour discrétiser, trois structures sont cette fois utilisées. Deux méta-modules D_L et D_R viennent décrire les comportements des itérations respectivement de gauche et de droite.

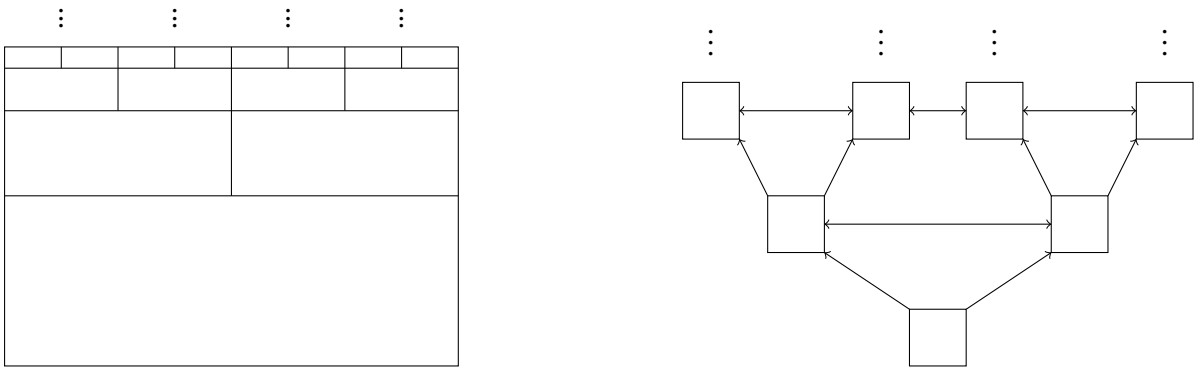
TABLE 6.2 – Méta-signaux et règles de collisions pour la structure « division par deux ».

Méta-signaux :

Nom	Vitesse	Couleur sur la figure
μ_1	0	Rouge
μ_2	1	Vert
μ_3	$\frac{1}{3}$	Bleu
μ_4	-1	Violet
μ_5	$-\frac{1}{3}$	Marron
μ_6	1	Noir

Règles de collisions :

Signaux entrants	Signaux sortants
μ_1, μ_2	μ_1, μ_4
μ_3, μ_4	$\mu_1, \mu_2, \mu_3, \mu_4, \mu_5$
μ_1, μ_6	μ_1, μ_2, μ_3
μ_1, μ_4	μ_1, μ_2
μ_2, μ_5	$\mu_1, \mu_2, \mu_3, \mu_4, \mu_5$



(a) découpage modulaire

(b) dynamique modulaire

FIGURE 6.7 – Expressions modulaires de la structure « division par deux ».

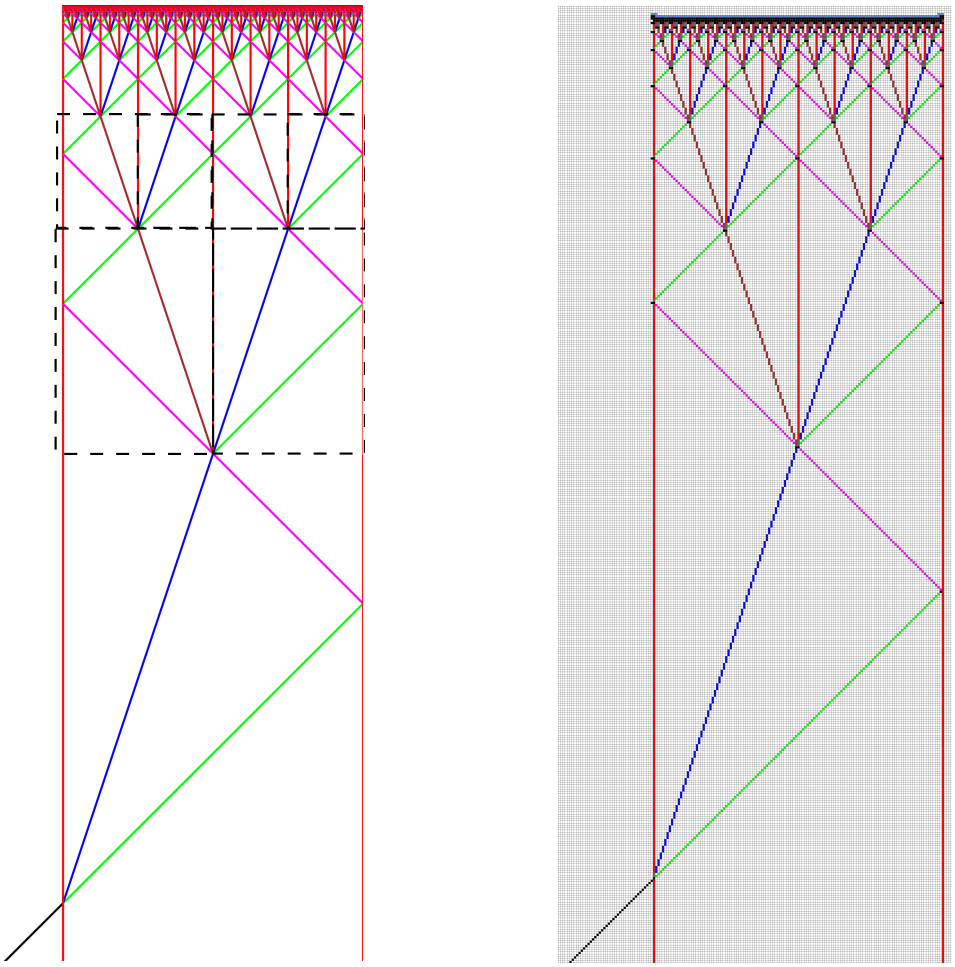
Et à nouveau, un méta-module D_2 prend la place de ces itérations lorsque k divisions sont atteintes. Comme précédemment, ce D_2 donne les règles de transition contenant au moins une cellule avec trop de collisions. En effet, dans cette structure, le problème de la discrétisation apparaît aussi à travers un trop grand nombre de collisions dans une même cellule. La solution d'approximation est donc de préciser le résultat de ce cas. Le méta-module D_2 est celui qui contient la donnée d'approximation, sous la forme de la règle de transition lorsqu'une ou plusieurs cellules contiennent trop de collisions.

La discrétisation d'une dynamique contenant un proto-module, se déroule comme celle n'en contenant pas, à une exception près. Cette exception est l'arrivée du proto-module qui est traité de manière particulière :

— tout d'abord, au moins $2^{(k+1)-1}$ modules de type D_L et D_R sont engendrés et composés séquentiellement,
— lorsqu'un module de type D_L ou D_R en train d'être ajouté viole une condition de taille (il est trop petit), un module de type D_2 est engendré et composé avec les dernières itérations de D_L et D_R,
— enfin la lecture de D_2 permet d'enregistrer le résultat de l'apparition de trop de collisions dans une cellule, qui sera utilisé *uniquement* dans ce cas.

Pour n'utiliser la règle de trop de collisions que pour cette structure, l'état qui le marque est différent de celui pour le reste du diagramme, et donc sa règle de transition aussi. Cela permet d'avoir plusieurs constructions de ce type dans le même diagramme avec des approximations différentes à chaque fois.

Avec ce procédé, la discrétisation est correcte jusqu'à la première apparition d'un module de type D_2. En effet, jusqu'à ce moment, elle se comporte exactement comme n'importe quelle autre, et donc la preuve de correction précédente peut être utilisée. Après ce moment, la discrétisation approxime et donc ne se comporte plus exactement comme la version continue. Par conséquent, le résultat est un automate cellulaire (et un diagramme espace-temps discret) « proche » (au sens de l'utilisateur) du continu, mais inévitablement incorrect. Cela dit, cette technique permet de proposer une méthode pour discrétiser ces structures de division, comme l'illustre la Fig. 6.8.



(a) structure continue (b) sa discrétisation

FIGURE 6.8 – Structure fractale « division par deux » et sa discrétisation.

Bien que cette discrétisation soit fonctionnelle, elle dépend fortement d'un paramètre important sur la largeur du module discret : la distance entre les deux signaux verticaux d'origine doit être exactement divisible par 2^k, avec $k \in \mathbb{N}$. Cela est logique dans le cadre de cette solution au FSSP, puisque le comportement décrit ici est donc une division par deux répétée, donc, si la division n'est pas entière, un des deux côté est plus grand que l'autre, causant alors des erreurs en cascade. Il en est de même pour les solutions historiques de FSSP : tant que la distance discrète est une puissance de deux, tout se passe bien mais si ce n'est pas le cas, la solution passe par de nombreux détails techniques propres aux

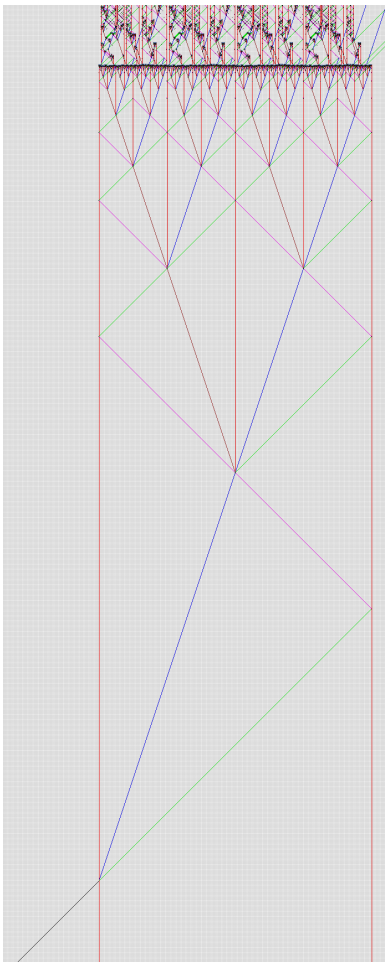


FIGURE 6.9 – Discrétisation et erreur de la division par deux.

automates cellulaires. Dans le cadre de cette thèse, ce travail n'a pas été accompli sur la méthode de discrétisation. La Fig. 6.9 illustre de façon spectaculaire l'erreur de cette discrétisation avec une distance qui n'est pas une puissance de deux.

Conclusion

Résultats

La présente thèse propose trois méthodes de discrétisation des machines à signaux en automates cellulaires.

La première ne fonctionne que sur un sous-ensemble de machines. Ces machines rationnelles à trois vitesses possèdent une propriété qui leur permet d'être discrétisées exactement. Toutes les collisions et signaux de celles-ci se trouvent sur un *maillage*, une structure régulière fondamentalement discrète. De ce fait, leurs discrétisations sont exactes, puisque leurs comportements sont discrets. Le procédé est alors d'établir un ensemble de règles et d'états d'automate cellulaire qui vont simuler le comportement de ces machines. Il devient alors aisé de démontrer son exactitude. Cette méthode a été implantée en s'appuyant sur le travail déjà existant, fournissant ainsi un outil permettant de discrétiser automatiquement ce type de machines.

La deuxième méthode de discrétisation est utilisable sur n'importe quelle machine à signaux mais ne fonctionne pas toujours et ne propose pas de garanties sur la qualité de l'approximation. Elle explore l'ensemble des états possibles et n'utilise que ceux nécessaires à la discrétisation d'une exécution particulière. Pour pouvoir représenter toutes les positions des signaux continus avec des discrets, le nombre d'états nécessaires deviendrait infini avec la définitions des AC-signaux. De ce fait, il est impossible de tous les engendrer et il devient donc nécessaire de limiter le nombre d'états qui seront utilisés. Pour cela, cette méthode exploite la machine à signaux en train d'être discrétisée : chaque cellule représentant un ensemble de signaux continus chacun associé à une position dans cette cellule, il est possible de les utiliser pour recréer une configuration continue de machine à signaux, et ainsi de connaître son comportement sur ce type d'entrées. Avec cela, il devient possible d'obtenir le résultat de trois cellules avec la machine d'origine et donc calculer pour chaque nouveaux groupes de trois cellules son résultat. Cette discrétisation produit donc les règles et états de l'automate cellulaire au fur et à mesure de son exécution. Cependant, cela ne limite pas pour autant le nombre d'états possibles. Pour cela, les positions des signaux continus à l'intérieur des cellules sont limités à des valeurs fixes (calculées pour chaque discrétisation). Ainsi, le nombre d'états sans collision devient fini. Ensuite, pour éviter qu'il puisse y avoir un nombre infini de collisions dans une cellule, ce nombre est limité par une valeur entière qui, lorsqu'elle est atteinte, met en échec la discrétisation ou utilise un état par défaut comme résultat de la règle explorée. Avec cela, le nombre d'états contenant une ou plusieurs collisions est fini, le nombre d'états n'en contenant pas est fini, donc le nombre d'états possibles au total est fini. Cela dit, avec ces comportements, la méthode cause des erreurs (disparition ou déplacement de signaux ou de collisions), notamment lors de l'apparition de comportement purement continus,

puisqu'elle ne propose aucune approximation. Cela mène alors vers l'établissement d'une nouvelle méthode permettant d'exploiter ces bases tout en proposant une solution limitant ces problèmes.

La troisième méthode est donc cela : s'appuyant sur la solution par exploration, elle propose de structurer les diagrammes espace-temps en zones connectées, afin d'utiliser cette structuration dans le cadre d'une discrétisation. Appelé *modularité*, le concept est de définir des zones rectangulaires d'intérêt au sein des diagrammes espace-temps continus, appelées modules, liées entre elles par des signaux continus. Chaque module est une instanciation d'un concept de plus haut niveau, les méta-modules, qui décrivent la nature des modules auxquels ils sont associés. Ainsi, les diagrammes espace-temps de machines à signaux sont décrits en termes de méta-modules et modules dans une construction portant le nom de dynamique modulaire fine continue. Cette nouvelle description de la dynamique d'une exécution d'une machine à signaux peut-être transformée en une version discrète (la dynamique modulaire fine discrète). Cette version discrète peut alors être utilisée pour obtenir un ensemble de signaux discrets et de positions spatio-temporelles qui sont alors utilisés pour produire une configuration initiale d'automate cellulaire qui assure, par la structure modulaire, que la discrétisation sera exacte jusqu'à un certain point bien déterminé.

Cette notion de modularité offre aussi un cadre théorique supplémentaire dans la description de dynamique de machines à signaux, et peut donc être utilisée pour construire différentes preuves. Dans cette thèse, cela est utilisé pour monter l'exactitude de la discrétisation brutale présentée précédemment sur un sous-ensemble de machines à signaux, celles capables de simuler une machine de Turing. Celles-ci présentent une structure régulière particulière qui se décrit facilement par des modules. Cette description modulaire peut alors être utilisée pour une discrétisation et être montrée exacte dans tous les cas. Mieux encore, elle peut aussi être utilisée pour exprimer la structuration de ces machines à signaux et en déduire l'exactitude du processus de discrétisation brutale.

Enfin, la modularité peut être étendue en y ajoutant des éléments et méthodes de discrétisation *ad hoc* pour des structures fractales dans les machines à signaux. Deux de ces solutions sont données dans cette thèse. La première est utilisée pour un type de diagramme problématique contenant une structure fractale simple. Celle-ci se répète infiniment en condensant de plus en plus de signaux dans un espace et un temps de plus en plus faible. Pour résoudre ce problème, la solution passe par la création d'un nouvel élément de description modulaire, le proto-module, avec sa méthode de discrétisation et une technique pour donner son approximation discrète. Cette méthode a été spécialement conçue pour être utilisée sur une structure bien particulière des machines à signaux, connue sous le nom d'accumulation « zig-zag ». La deuxième solution concerne une autre structure fractale, mais celle-ci double le nombre de modules à chaque étape. Elle passe encore une fois par l'utilisation du concept de proto-module mais cette fois adapté à cette situation particulière. Il en va de même pour sa discrétisation et sa méthode d'approximation. La construction pour laquelle cette solution a été conçue correspond à la division infinie par deux dans les machines à signaux. Lorsque discrétisée, elle rappelle une solution du problème de synchronisation d'escouade au sein des automates cellulaires (voir la section de l'introduction sur les automates cellulaires). Ces trois constructions semble indiquer que pour chaque problème de cette nature, une solution *ad hoc* est possible.

Perspectives

Cette thèse propose suffisamment de concepts pour pouvoir être étendue sur de nombreux points. Les perspectives sont de deux natures : des extensions du modèle d'expression modulaire et des améliorations de l'implantation.

Extensions du modèle

Une perspective d'amélioration du modèle de la modularité serait de vérifier la correction de chaque méta-modules donné. Cela signifie prouver formellement que toutes les entrées possibles répondant aux conditions du méta-module engendreront toujours une sortie de la forme donnée dans les conséquences. Dans un premier temps, il est imaginable de faire cette vérification manuellement pour chaque méta-module. Dans un second temps, proposer un outil capable de faire ce travail automatiquement sur chaque nouveau méta-module se présente comme la finalité de ce procédé. Cependant, il est clair qu'il est impossible de faire cette vérification sur tout les méta-modules possibles, certains étant trop complexes pour le pouvoir (un méta-module proposant de résoudre un problème incalculable dans le modèle classique mais pas pour une machine à signaux est évidemment invérifiable). Au final, la meilleure perspective est alors d'assurer la vérification automatique pour des méta-modules suffisamment simples (en définissant formellement ce que cela veut dire).

Une autre possibilité est d'étendre la notion de (méta-)modules, notamment en modifiant leur forme (triangles, hexagones, formes quelconques), ou en proposant de nouvelles conditions d'entrées ou conséquences de sorties. La question qui se pose alors est de savoir si de tels changements pourraient ou non impacter l'expressivité de la modularité, ou si la forme rectangulaire est équivalente à toute autre forme en termes d'expressivité. Dans [Sen13], un chapitre est dédié à une forme antérieure très embryonnaire des modules. Cela dit, l'auteur y propose trois formes pour les modules : rectangulaire, trapézoïdale et hexagonale, et en discute les avantages et inconvénients, ainsi que leur utilisation. Dans la version de la modularité proposée ici, il serait intéressant de savoir si les arguments apportés peuvent aussi s'appliquer ou si la différence de cadre l'en empêche.

Enfin, il peut-être intéressant de proposer davantage de méta-modules permettant de décrire plus de diagrammes espace-temps. Un des objectifs de la modularité étant de permettre de fabriquer un diagramme espace-temps en « assemblant » des modules, proposer des méta-modules pour cela permettrait de faciliter ce travail. De plus, en observant les constructions créées pour les machines à signaux déjà étudiées, il est aisé de constater que certaines structures apparaissent fréquemment, et il est donc naturel de vouloir en proposer des versions modulaires. C'est par exemple le cas de la structure de réduction, un ensemble de signaux capable de réduire l'espace entre plusieurs signaux tout en en conservant l'ordre et sans affecter la dynamique finale. À l'opposé, il existe une construction capable de faire l'inverse : augmenter l'espace entre plusieurs signaux sans altérer la dynamique. Ces deux exemples apparaissent souvent dans des dynamiques continues, mais il existe bien entendu une infinité de méta-modules possibles. Par conséquent, établir des méta-modules pourrait en quelque sorte s'apparenter à l'établissement d'instructions dans un langage de programmation classique. Chaque méta-module est ainsi une « brique élémentaire » d'un langage au nombre d'instructions potentiellement infini. Cela s'appa-

rente à l'utilisation des circuits logiques dans nos ordinateurs modernes ou aux instructions basiques du micro-code des processeurs. En s'inspirant de cette idée, il est possible d'imaginer proposer un ensemble de méta-modules universels dans un certains cadre, qui permettrait de construire n'importe quel diagramme espace-temps.

Amélioration de l'implantation

Bien que les différents éléments proposés dans cette thèse aient été implantés, celle-ci ne remplit que des fonctionnalités basiques et pourrait être étendue et améliorée sur de nombreux aspects.

En premier lieu, l'écriture de méta-modules et modules n'est pas facile d'accès et pourrait donc être simplifiée en proposant notamment une interface graphique capable d'extraire un module à partir d'une partie d'un diagramme espace-temps continu. L'utilisation simple d'un tel outil serait par exemple la possibilité de tracer un rectangle autour de l'expression graphique d'un diagramme, et qu'à partir de là, le module soit construit automatiquement avec ses coordonnées, ainsi que ses entrées et sorties. Pour les méta-modules, proposer un certain nombre de facilités lors de l'écriture (variables implicites, écriture automatique...) permettrait d'en rendre l'accès plus simple. Par exemple, il est aisé d'imaginer que la variable représentant la machine à signaux (ou l'automate cellulaire) dans la description d'un méta-module n'ait plus besoin d'être exprimée et ait un nom par défaut utilisable directement. Ce traitement peut alors être étendu à d'autres variables, et même à certaines conditions ou conséquences. Il est aussi possible d'ajouter des variables locales, utilisées uniquement pour simplifier l'écriture et qui ne rajoute donc aucun pouvoir expressif.

Ensuite, la description des conditions et conséquences telle que proposée actuellement est très basique : elle prend certes la forme d'une expression régulière mais n'en a pas le potentiel expressif. Ainsi, y ajouter l'utilisation des opérateurs *, + ou ? ainsi que le parenthésage et la puissance serait utile à l'expression. De plus, dans son état actuel, ces expressions n'ont pas une utilisation toujours très claire (notamment avec la possibilité de marquer un signal précis) ce qui en rend l'écriture et la lecture parfois difficile. Les simplifier en proposant un autre format ou juste une expression simplifiée permettrait d'écrire plus facilement et donc en quantité plus grande de nouveaux méta-modules.

Toutes ces améliorations et extensions n'ont en fait qu'un but : permettre une utilisation extérieure simple pour toutes personnes souhaitant produire un automate cellulaire à partir d'une machine à signaux. Ainsi, l'étape ultime de ce travail est donc de produire un outil qui pourrait être utilisé dans le cadre de recherches sur les automates cellulaires. De cette manière, la construction de l'expression continue permettrait de produire un automate cellulaire en utilisant le processus de discrétisation automatique. C'est là la finalité de cette thèse : proposer une méthode simple de construction d'un automate cellulaire en utilisant une description continue qui est alors discrétisée. Dans ce cadre, la discrétisation n'aura que rarement besoin des traitement de cas particuliers discutés dans le Chapitre 6, puisque les constructions continues seraient pensées comme des abstractions d'éléments discrets.

Annexes

Annexe A

Implantation de la discrétisation des machines rationnelles à trois vitesses

Cette discrétisation a été implantée et testée en Java à partir de la librairie existante manipulant les machines à signaux (voir la section Implantations de l'introduction). La Figure A.1 fourni un exemple de la conversion d'une machine à signaux rationnelle à trois vitesses en un automate cellulaire.

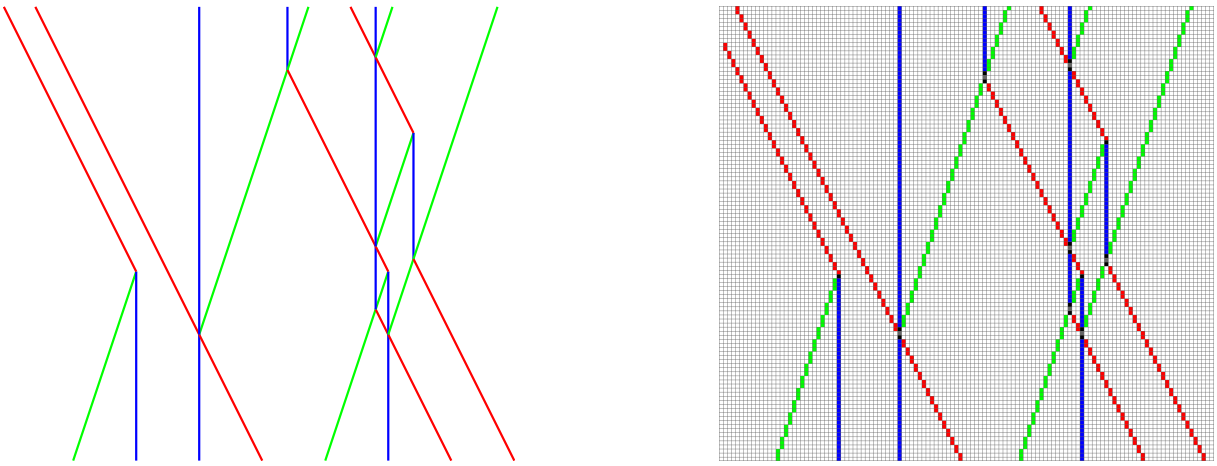


FIGURE A.1 – Un diagramme espace-temps continu et la discrétisation associée.

Chaque étape décrite dans le Chapitre 2 a été implantée. La première est la normalisation et le calcul de l'échelle de discrétisation. Normaliser une machine à signaux se fait en extrayant les vitesses des méta-signaux et les positions des signaux sur la configuration initiale et en appliquant les méthodes décrites dans la Section 2.1. Cela fait, une nouvelle machine est créée, en utilisant les valeurs normalisées. L'échelle de discrétisation peut alors être calculée avec la formule donnée dans la sous-section 2.1.5. La classe Signal_Machine_Normalizing_Scheme est utilisée pour stocker la normalisation de la machine.

La transformation d'une machine à signaux en un automate cellulaire engendre tout les états et transitions tels que décris dans la Section 2.1 (les méta-signaux génériques r, l et z doivent être instanciés avec toutes les combinaisons possibles).

La Figure A.2 donne le diagramme de classe UML de la librairie. Dans ce diagramme, les interfaces du haut appartiennent à la librairie basiques des machines à signaux, la moi-

tié haute présente l'architecture de l'étape de normalisation et la partie basse la conversion en un automate cellulaire.

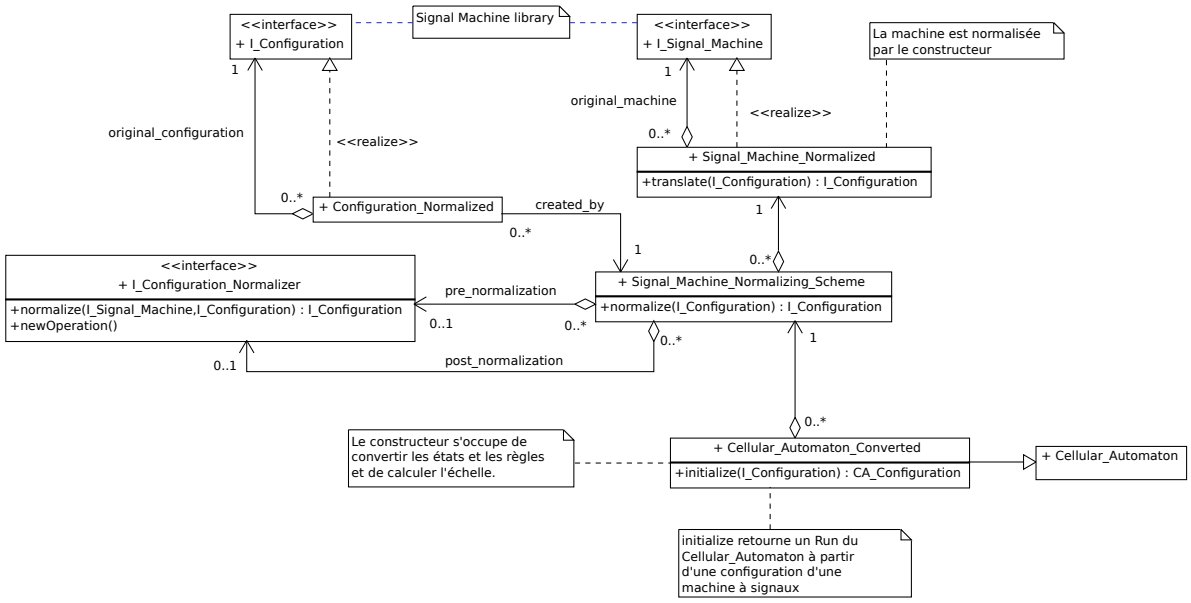


FIGURE A.2 – Le diagramme de classe UML de la librairie.

L'implantation a été testée sur plusieurs MS 3S-$\mathbb{Q}$. Durant ces tests, des données ont été collectées et sont données dans les Figures A.3 et A.4.

Dans la Figure A.3, pour chaque machine, chaque collision émet des signaux des trois vitesses. La partie de gauche d'une cellule indique le nombre d'états produits par la conversion, et la partie de droite le nombre de transitions.

Les Figures A.4(a) et A.4(b) montrent que fixer une vitesse et augmenter le dénominateur de l'autre augmente le nombre de transitions et d'états linéairement, tel qu'attendu. La Figure A.4(c) montre qu'augmenter les dénominateurs des vitesses non-nulles d'une 3S-$\mathbb{Q}$ augmente le nombre d'états linéairement. En fait, elle montre qu'augmenter les deux dénominateurs par 1 ajoute 6 nouveaux états, ce qui peut être expliqué :

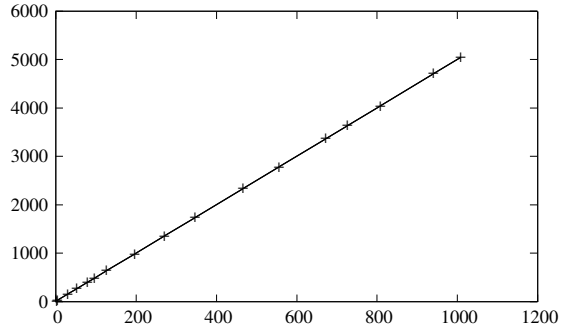
— deux d'entre eux sont pour le mouvement du AC-signal (puisque le dénominateur est augmenté de 1, il est nécessaire d'avoir un nouvel état pour chaque période d'un signal),

— les autres sont ajoutés pour chaque collisions possibles (il y a 3 méta-signaux, donc il y a 4 collisions possibles).

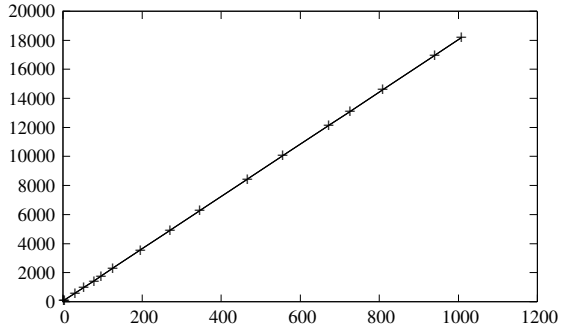
Comme attendu, la Fig. A.4(d) montre qu'augmenter les deux dénominateurs par 1 augmente le nombre de transitions de façon non-linéaires.

p \ q	271	347	467	557	673	727	809	941	1009
271	8 / 25	2008 / 103893	2608 / 137853	3058 / 163383	3638 / 196675	3908 / 212173	4318 / 235707	4978 / 273591	5318 / 293107
347		8 / 25	2684 / 174941	3134 / 207251	3714 / 248895	3984 / 268413	4394 / 298179	5054 / 346095	5394 / 370779
467			8 / 25	3254 / 276611	3834 / 332175	4104 / 358041	4514 / 397319	5174 / 460575	5514 / 493419
557				8 / 25	3924 / 394635	4194 / 425361	4604 / 472019	5264 / 547127	5604 / 585819
673					8 / 25	4310 / 512129	4720 / 568299	5380 / 658719	5720 / 705299
727						8 / 25	4774 / 613119	5434 / 710667	5774 / 760919
809							8 / 25	5516 / 789551	5856 / 845379
941								8 / 25	5988 / 981339
1009									8 / 25

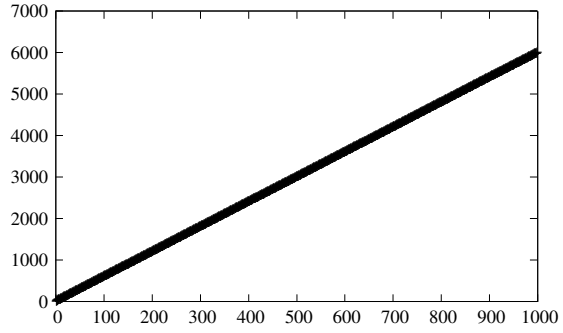
FIGURE A.3 – Nombres d'états (coin haut gauche) et nombre de transitions (coin bas droit) produits par la librairie, selon les vitesses des méta-signaux non-nuls pour certaines valeurs premières (pour assurer l'irréductibilité).



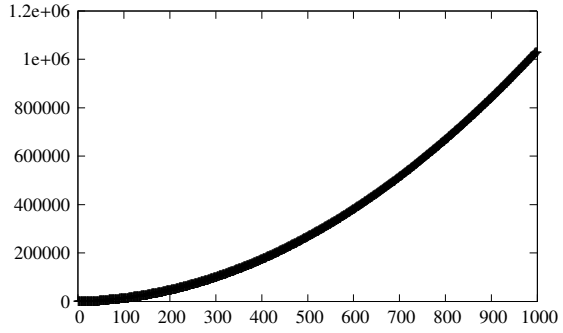
(a) Nombre d'états pour les vitesses $-\frac{1}{2}, 0, \frac{1}{n}$ pour n dans l'ensemble des vitesse données dans les tables précédentes.



(b) Nombre de transitions pour les vitesses $-\frac{1}{2}, 0, \frac{1}{n}$ pour n dans l'ensemble des vitesse données dans les tables précédentes.



(c) Nombre d'états pour les vitesses $-\frac{1}{n}, 0, \frac{1}{n+1}$ pour $n \in \mathbb{N}$ et $n \leq 1000$.



(d) Nombre de transitions pour les vitesses $-\frac{1}{n}, 0, \frac{1}{n+1}$ pour $n \in \mathbb{N}$ et $n \leq 1000$.

FIGURE A.4 – Nombre d'états et de transitions, selon les vitesses des méta-signaux non-nuls.

Annexe B

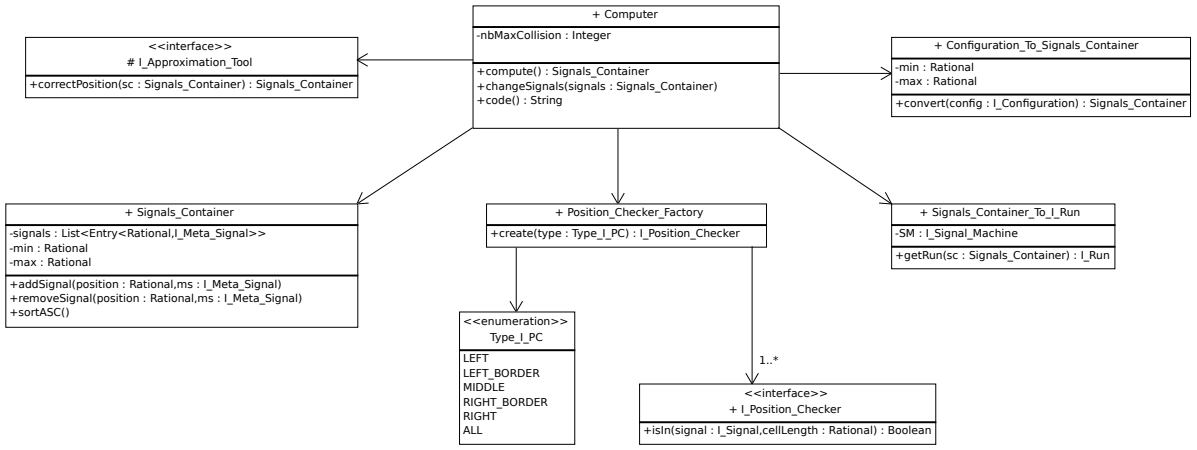
Implantation de la discrétisation brutale

Dans cette implantation, donner le diagramme de classe est sans intérêt. En effet, il ne s'agit que d'une instanciation spéciale d'une bibliothèque dédiées à la manipulation d'AC pour y intégrer l'utilisation de la simulation du comportement d'une machine à signaux sur une certaine configuration puis extraction du résultat.

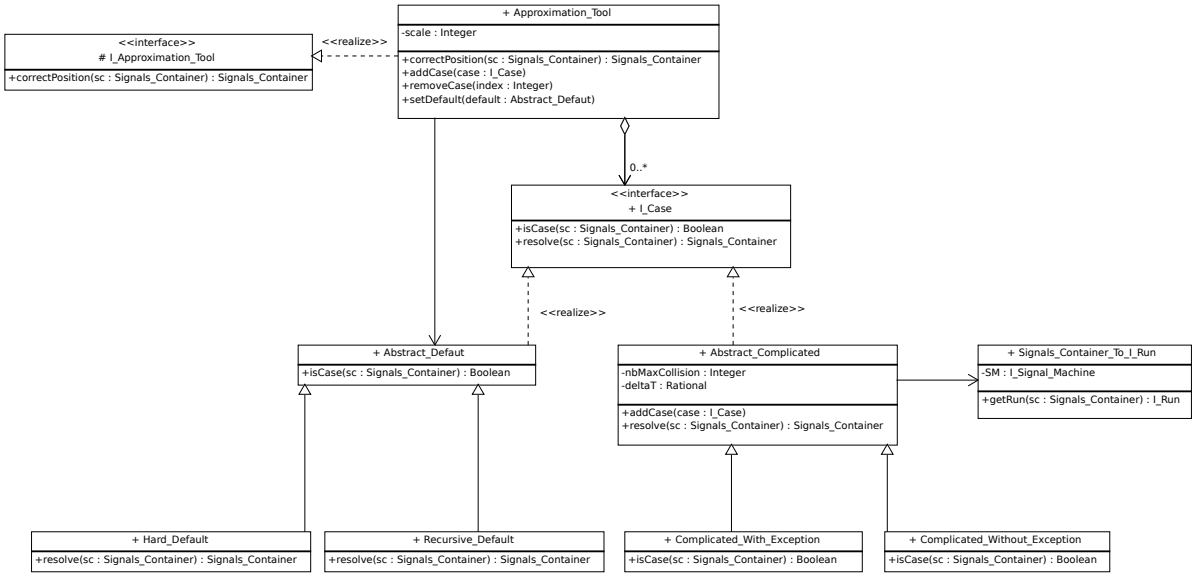
Les seuls diagrammes intéressant et utiles pour qui s'intéresse au code sont donc ceux de [Lab17], qui présentent le détail de cette bibliothèque.

Celle-ci est en deux parties, avec tout d'abord la simulation en elle-même de la machine à signaux, puis l'approximation du résultat quand le besoin se présente. La Fig. B.1 donne ces deux diagrammes.

Le travail d'implantation s'est donc porté sur l'intégration de cette bibliothèque dans un simulateur d'automates cellulaires et ne présente rien de notable. Il a cependant nécessiter un travail important, notamment pour produire un simulateur d'AC pouvant être facilement interfacé avec le calcul des nouveaux états.



(a) simulation par la machine à signaux



(b) méthode d'approximation

FIGURE B.1 – Les diagrammes de classe de l'implantation de la discrétisation brutale, extraits de [Lab17].

Annexe C

Implantation de la discrétisation modulaire

La discrétisation modulaire a aussi été implantée en Java et son implantation est donnée dans la Fig. C.1.

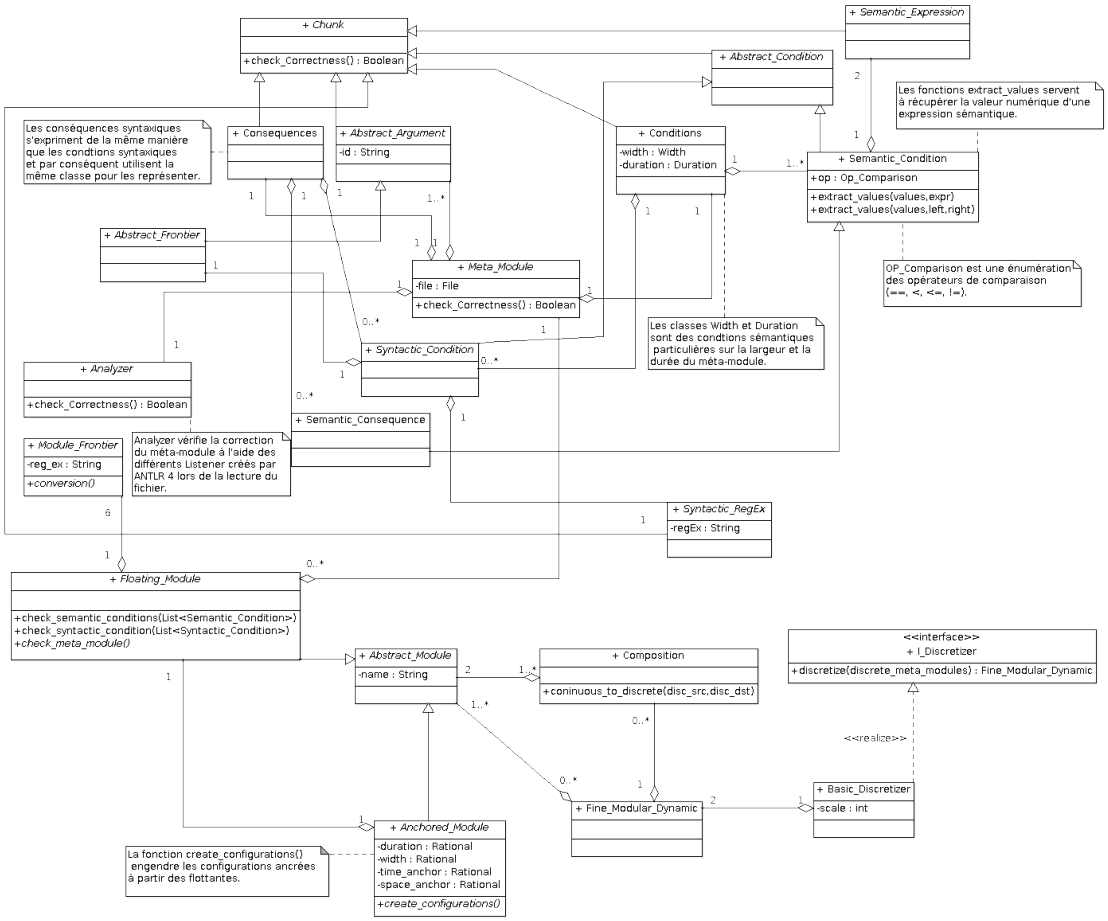


FIGURE C.1 – Le diagramme de classe de l'implantation de la modularité.

Ce diagramme ne donne pas l'implantation de la partie lecture et vérification syntaxique et lexicale du langage de description des fichiers (parser et lexer). Celles-ci sont

faites à l'aide de ANTLR (version 4.5.3), qui engendre automatiquement ces fonctionnalités à partir d'un fichier exprimant la grammaire du langage à compiler.

Il y a de nombreuses classes abstraites dans ce diagramme car il ne s'agit ici que de la vision générique des méta-modules et modules, c'est-à-dire qu'il s'agit d'un squelette qui est implémenté pour les deux types de modularités (continue et discrète). Ainsi, il faudrait en réalité tripler la taille de ce diagramme pour représenter la totalité de l'implantation.

Pour le décrire simplement, il se présente en quatre parties :

— les méta-module (la classe Meta_Module y est centrale),
— les modules (avec la classe Abstract_Module),
— la dynamique modulaire fine (avec notamment la classe Fine_Modular_Dynamic), et
— la discrétisation en elle-même (avec la classe Basic_Discretizer).

Annexe D

Formulation des fichiers décrivant les méta-modules continus

Cette annexe donne des exemples de descriptions de méta-modules continus tels que manipulés par l'implantation de la discrétisation modulaire. Le premier est le méta-module « d'agrandissement » : il multiplie l'espace entre deux signaux verticaux, en laissant un seul des deux à sa position d'origine. Le deuxième est le méta-module de « déplacement » : il décale d'une valeur constante deux signaux verticaux vers la gauche ou la droite. Ces deux méta-modules sont en fait ceux utilisés dans l'exemple de dynamique modulaire du Chapitre 5 (Fig. 5.1).

D.1 Méta-module d'agrandissement

```
**Arguments
***Main
SM : SignalMachine;
IC : CFrontier;
IR : CFrontier;
IL : CFrontier;
OC : CFrontier;
OR : CFrontier;
OL : CFrontier;
d: Duration;
w : Width;
a : MetaSignal in SM;
b : MetaSignal in SM;
c : MetaSignal in SM;
e : MetaSignal in SM;
f : MetaSignal in SM;


**Local variables

**Conditions

***Machine
0 < S(a);
S(b) = 0;
S(a) < S(c);
S(e) < 0;
S(f) < S(d);
{ a , b } --> { c , e } in SM;
{ c , b } --> { a , b , f } in SM;
{ e , f } --> { b } in SM;

***Syntactic
IC ~=  a\i0,b\i1,b\i2;
****Sides
IR ~= Empty;
IL ~= Empty;

***Semantic

****Sides

***Duration
(pos(b\i2) /S(e)) < d ;

***Width
(pos(b\i2) - pos(a\i0)) * 2 < w ;

**Consequences

***Output
OC ~= b,b,a;

****Sides
OR ~= Empty;
OL ~= Empty;

***Properties

****Sides
```

D.2 Méta-module de déplacement

```
**Arguments
***Main
SM : SignalMachine;
IC : CFrontier;
IR : CFrontier;
IL : CFrontier;
OC : CFrontier;
OR : CFrontier;
OL : CFrontier;
d: Duration;
w : Width;
a : MetaSignal in SM;
b : MetaSignal in SM;
c : MetaSignal in SM;
e : MetaSignal in SM;

**Local variables

**Conditions

***Machine
0 < S(a);
S(b) = 0;
S(a) < S(c);
S(e) = S(c) / 2;
{ a , b } --> { c , e } in SM;
{ c , b } --> { b , e } in SM;
{ b , e } --> { b , c } in SM;
{ c , e } --> { b } in SM;

***Syntactic
IC ~=  a\i0,b\i1,b\i2;
****Sides
IR ~= Empty;
IL ~= Empty;

***Semantic

****Sides

***Duration
((pos(b\i2) - pos(a\i0)) + (pos(b\i2) - pos(b\i1)) * S(a) < d ;

***Width
(pos(b\i2) - pos(a\i0)) + (pos(b\i2) - pos(b\i1)) < w ;

**Consequences

***Output
OC ~= b,b;

****Sides
OR ~= Empty;
OL ~= Empty;

***Properties

****Sides
```

Annexe E

Formulation des fichiers décrivant les méta-modules discrets

Cette annexe donne les fichiers exprimant les méta-modules discrets « équivalents » (selon l'utilisateur de la librairie) aux continus exprimés dans l'Annexe D.

E.1 Méta-module d'agrandissement

```
**Arguments
***Main
CA : CellularAutomata;
IC : DFrontier;
IR : DFrontier;
IL : DFrontier;
OC : DFrontier;
OR : DFrontier;
OL : DFrontier;
w : Width;
d : Duration;
a : CASignal;
b : CASignal;

c : CASignal;
e : CASignal;
f : CASignal;

col : State;

k : Rational;
l : Rational;
m : Rational;
o : Rational;
p : Rational;

**Local variables

**Conditions
***Automata
0 < S(a);
S(b) = 0;
S(a) < S(c);
S(e) < 0;
S(f) < S(d);

***Syntactic
IC ~= #^l.a'0\i0.#^k.b'0\i1.#^p.b'0\i2.#^m;

****Sides
IL ~= #^d;
IR ~= #^d;

***Semantic

****Sides

***Duration
(pos(b\i2) /S(e)) < d ;

***Width
(pos(b\i2) - pos(a\i0)) * 2 < w ;

***Normality

**Consequences

***Output
OC ~=  #^l.b'0.#^k.b'0.#^o.a'0.#^m;

****Sides
OL ~= #^d;
OR ~= #^d;

***Properties

****Sides

**Discretization informations

***Continuous meta-modules
resize.mod;
```

E.2 Méta-module de déplacement

```
**Arguments
***Main
CA : CellularAutomata;
IC : DFrontier;
IR : DFrontier;
IL : DFrontier;
OC : DFrontier;
OR : DFrontier;
OL : DFrontier;
w : Width;
d : Duration;
a : CASignal;
b : CASignal;

c : CASignal;
e : CASignal;

col : State;

k : Rational;
l : Rational;
m : Rational;
o : Rational;
p : Rational;

**Local variables

**Conditions
***Automata
0 < S(a);
S(b) = 0;
S(a) < S(c);
S(e) = S(c) / 2;

***Syntactic
IC ~= #^l.a'0\i0.#^k.b'0\i1.#^p.b'0\i2.#^m;

****Sides
IL ~= #^d;
IR ~= #^d;

***Semantic

****Sides

***Duration
((pos(b\i2) - pos(a\i0)) + (pos(b\i2) - pos(b\i1)) * S(a) < d ;

***Width
(pos(b\i2) - pos(a\i0)) + (pos(b\i2) - pos(b\i1)) < w ;

***Normality

**Consequences

***Output
OC ~=  #^l.b'0.#^k.b'0.#^o.#^m;

****Sides
OL ~= #^d;
OR ~= #^d;

***Properties

****Sides

**Discretization informations

***Continuous meta-modules
shift.mod;
```

Bibliographie

[Ada02] A. Adamatzky (éd.) – *Collision based computing*, Springer London, 2002.

[Amo97] M. Amos – « Dna computation », Thèse, University of Warwick, September 1997.

[BBB04] J. Berggren, J. Borwein et P. Borwein – *Pi : A source book*, Springer-Verlag New York, 2004.

[BCDL+13] F. Becker, M. Chapelle, J. Durand-Lose, V. Levorato et M. Senot – « Abstract geometrical computation 8: Small machines, accumulations & rationality », Submitted, 2013.

[Bre65] J. Bresenham – « Algorithm for computer control of a digital plotter », *IBM Systems Journal* **4** (1965), no. 1, p. 25–30.

[BSS89] L. Blum, M. Shub et S. Smale – « On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines », *Bull. Amer. Math. Soc.* **21** (1989), no. 1, p. 1–46.

[Chu36] A. Church – « An unsolvable problem of elementary number theory », *Journal of Symbolic Logic* **1** (1936), no. 2, p. 73–74.

[CMD03] J. P. Crutchfield, M. Mitchell et R. Das – « The evolutionary design of collective computation in cellular automata », *Evolutionary Dynamics—Exploring the Interplay of Selection, Neutrality, Accident, and Function* (J. P. Crutchfield et P. K. Schuster, éds.), New York : Oxford University Press, 2003, p. 361–411.

[Coo04] M. Cook – « Universality in elementary cellular automata », *Complex Systems* **15** (2004), p. 1–40.

[DDS10] D. Duchier, J. Durand-Lose et M. Senot – « Fractal parallelism : Solving SAT in bounded space and time », *Algorithms and Computation - 21st International Symposium, ISAAC 2010, Jeju Island, Korea, December 15-17, 2010, Proceedings, Part I*, LNCS, 2010, p. 279–290.

[DL00] J. Durand-Lose – « Reversible space-time simulation of cellular automata », *Theoret. Comp. Sci.* **246** (2000), no. 1–2, p. 117–129.

[DL02] — , « Computing inside the billiard ball model », *Collision-Based Computing* (London) (A. Adamatzky, éd.), Springer London, 2002, p. 135–160.

[DL03] — , « Calculer géométriquement sur le plan – machines à signaux », Habilitation à Diriger des Recherches, École Doctorale STIC, Université de Nice-Sophia Antipolis, 2003.

[DL05] — , « Abstract geometrical computation : Turing-computing ability and undecidability », *New Computational Paradigms, First Conference on Computability in Europe, CiE 2005, Amsterdam, The Netherlands, June 8-12, 2005, Proceedings*, 2005, p. 106–116.

[DL06a] — , « Forcasting black holes in abstract geometrical computation is highly unpredictable », *Theory and Applications of Models of Computations (TAMC '06)* (J.-Y. Cai, B. S. Cooper et A. Li, éds.), LNCS, no. 3959, Springer, 2006, p. 644–653.

[DL06b] — , « Reversible conservative rational abstract geometrical computation is Turing-universal », *Logical Approaches to Computational Barriers, 2nd Conf. Computability in Europe (CiE '06)* (A. Beckmann et J. V. Tucker, éds.), LNCS, no. 3988, Springer, 2006, p. 163–172.

[DL07] — , « Abstract geometrical computation and the linear Blum, Shub and Smale model », *Computation and Logic in the Real World, 3rd Conf. Computability in Europe (CiE '07)* (B. S. Cooper, B. Löwe et A. Sorbi, éds.), LNCS, no. 4497, Springer, 2007, p. 238–247.

[DL08a] — , « Abstract geometrical computation with accumulations : Beyond the Blum, Shub and Smale model », *4th Conf. Computability in Europe (CiE '08) (abstracts and extended abstracts of unpublished papers)* (Greece) (A. Beckmann, C. Dimitracopoulos et B. Löwe, éds.), University of Athens, 2008, p. 107–116.

[DL08b] — , « Black hole computation : implementation with signal machines », *IW PC* (Wien, Austria) (C. S. Calude et J. F. Costa, éds.), 2008, p. 136–158.

[DL11] — , « Abstract geometrical computation 4: small Turing universal signal machines », *Theoret. Comp. Sci.* **412** (2011), p. 57–67.

[DL12] — , « Abstract geometrical computation 6 : A reversible, conservative and rational based model for black hole computation », *IJUC* **8** (2012), no. 1, p. 33–46.

[DL13] — , « Irrationality is needed to compute with signal machines with only three speeds », *CiE '13, The Nature of Computation* (P. Bonizzoni, V. Brattka et B. Löwe, éds.), LNCS, no. 7921, Springer, 2013, Invited talk for special session *Computation in nature*, p. 108–119.

[Fey86] R. P. Feynman – « Quantum mechanical computers », *Foundations of Physics* **16** (1986), no. 6, p. 507–531.

[Fis65] P. C. Fischer – « Generation of primes by a one-dimensional real-time iterative array », *J. ACM* **12** (1965), no. 3, p. 388–394.

[GGH67] S. Ginsburg, S. A. Greibach et M. A. Harrison – « Stack automata and compiling », *J. ACM* **14** (1967), no. 1, p. 172–201.

[Got66] E. Goto – « Ōtomaton ni kansuru pazuru [Puzzles on automata] », *Jōhōkagaku eno michi [The Road to information science]* (T. Kitagawa, éd.), Kyoristu Shuppan Publishing Co., Tokyo, 1966, p. 67–92.

[HL00] J. D. Hamkins et A. Lewis – « Infinite time Turing machines », *J. Symbolic Logic* **65** (2000), no. 2, p. 567–604.

[JS90] G. Jacopini et G. Sontacchi – « Reversible parallel computation : An evolving space-model », *Theor. Comput. Sci.* **73** (1990), no. 1, p. 1–46.

[Lab17] L. Labourbe – « Discrétisation de machines à signaux vers des automates cellulaires : aspects locaux », Thèse, Université d'Orléans, 2017.

[Law04] M. V. Lawson – *Finite automata*, Chapman and Hall/CRC, 2004.

[LS10] V. Levorato et M. Senot – « Discrete signal machines via pretopology », *Second Workshop on Non-Classical Models for Automata and Applications - NCMA 2010, Jena, Germany, August 23 - August 24, 2010. Proceedings*, 2010, p. 127–140.

[MH40] M. Morse et G. A. Hedlund – « Symbolic dynamics II. Sturmian trajectories », *Amer. J. Math.* **62** (1940), p. 1–42.

[MP43] W. S. McCulloch et W. Pitts – « A logical calculus of the ideas immanent in nervous activity », *The bulletin of mathematical biophysics* **5** (1943), no. 4, p. 115–133.

[MR98] J. Mazoyer et I. Rapaport – « Inducing an order on cellular automata by a grouping operation », *15th Annual Symposium on Theoretical Aspects of Computer Science (STACS '98)*, LNCS, vol. 1373, Springer, 1998, p. 116–127.

[MT99] J. Mazoyer et V. Terrier – « Signals in one-dimensional cellular automata », *Theoret. Comp. Sci.* **217** (1999), no. 1, p. 53–80.

[OR11] N. Ollinger et G. Richard – « Four states are enough ! », *Theoret. Comp. Sci.* **412** (2011), no. 1–2, p. 22–32.

[PR02] G. Păun et G. Rozenberg – « A guide to membrane computing », *Theoretical Computer Science* **287** (2002), no. 1, p. 73 – 100, Natural Computing.

[Ric08] G. Richard – « Systèmes de particules et collisions discrètes dans les automates cellulaires », Thèse, Aix-Marseille Université, 2008.

[RR96] J.-P. Reveillès et D. Richard – « Back and forth between continuous and discrete for the working computer scientist », *Annals of Mathematics and Artificial Intelligence* **16** (1996), p. 89–152.

[Sar00] P. Sarkar – « A brief history of cellular automata », *ACM Comput. Surv.* **32** (2000), no. 1, p. 80–107.

[Sen13] M. Senot – « Modèle géométrique de calcul : fractales et barrières de complexité », Thèse, Université d'Orléans, 2013.

[SS95] H. Siegelmann et E. Sontag – « On the computational power of neural nets », *J. Comput. Syst. Sci.* **50** (1995), no. 1, p. 132–150.

[Ste11] W. M. Stevens – « Computing with planar toppling domino arrangements », Unconventional Computation : 10th International Conference, UC 2011, Turku, Finland, June 6-10, 2011. Proceedings (C. S. Calude, J. Kari, I. Petre et G. Rozenberg, éds.), Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, p. 224–233.

[Tur36] A. M. Turing – « On computable numbers with an application to the entscheidungsproblem proceedings of the london mathematical society¡ », 1936.

[Tur39] — , « Systems of logic based on ordinals† », *Proceedings of the London Mathematical Society* **s2-45** (1939), no. 1, p. 161–228.

[vN66] J. von Neumann – *Theory of self-reproducing automata*, University of Illinois Press, Urbana, Ill. USA, 1966.

[Win98] E. Winfree – « Algorithmic self-assembly of dna », Thèse, California Institute of Technology, June 1998.

[WN05] D. Woods et T. J. Naughton – « An optical model of computation », *Theor. Comput. Sci.* **334** (2005), no. 1-3, p. 227–258.

[Yun07] J.-B. Yunès – « Simple new algorithms which solve the firing squad synchronization problem : a 7-states 4n-steps solution », *Machine, Computations and Universality (MCU '07)* (J. Durand-Lose et M. Margenstern, éds.), LNCS, no. 4664, Springer, 2007, p. 316–324.

Tom BESSON

Discrétisation automatique de machines à signaux en automates cellulaires

Dans le contexte du calcul géométrique abstrait, les machines à signaux ont été développées comme le pendant continu des automates cellulaires capturant les notions de particules, de signaux et de collisions. Une question importante est la génération automatique d'un automate cellulaire *reproduisant* la dynamique d'une machine à signaux donnée. D'une part, il existe des conversions *ad hoc*. D'autre part, ce n'est pas toujours possible car certaines machines à signaux présentent des comportements « continus ». Par conséquent, la discrétisation automatique de telles structures est souvent complexe et pas toujours possible. Cette thèse propose trois manières différentes de discrétiser automatiquement les machines à signaux en automates cellulaires, avec ou sans approximation possible.
La première s'intéresse à une sous-catégorie de machines à signaux, qui présente des propriétés permettant d'assurer une discrétisation automatique exacte pour toute machine de ce type. La deuxième est utilisable sur toutes les machines mais ne peut assurer ni l'exactitude ni la correction du résultat. La troisième s'appuie sur une nouvelle expression de la dynamique d'une machine à signaux pour proposer une discrétisation. Cette expression porte le nom de modularité et est décrite avant d'être utilisée pour discrétiser.
Mots clés : calcul géométrique abstrait, discrétisation automatique, automates cellulaires, machines à signaux

Automatic discretization of signal machines into cellular automata

In the context of abstract geometrical computation, signal machines have been developed as a continuous counterpart of cellular automata capturing the notions of particles, signals and collisions. An important issue is the automatic generation of a cellular automaton *mimicking* the dynamics of a given signal machine. On the one hand, *ad hoc* conversions exist. On the other hand, it is not always possible since some signal machines exhibit "purely continuous" behaviors. Therefore, automatically discretizing such structures is often complicated and not always possible. This thesis proposes different ways to automatically discretize signal machines into cellular automata, both with and without handling the possiblity of approximation.
The first is concerned with a subcategory of signal machines, which has properties ensuring an exact automatic discretization for any machine of this type. The second is usable on all machines but cannot guarantee the exactness and correction of the result. The third is based on a new expression of the dynamics of a signal machine to propose a discretization. This dynamical expression takes the name of modularity and is described before being used to discretize.
Keywords: abstract geometrical computation, automatic discretization, cellular automata, signal machines.

[LIFO - Bâtiment IIIA Rue Léonard de Vinci B.P. 6759 F-45067 ORLEANS Cedex 2]