Modèle géométrique de calcul : fractales et barrières de complexité

Maxime Senot

UNIVERSITÉ
D'ORLÉANS

ÉCOLE DOCTORALE MIPTIS
MATHÉMATIQUES, INFORMATIQUE, PHYSIQUE THÉORIQUE ET INGÉNIEURIE DES SYSTÈMES

Laboratoire d'Informatique Fondamentale d'Orléans

THÈSE

présentée par :

Maxime Senot

soutenue le : **27 juin 2013**

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

Discipline/Spécialité : **Informatique**

Modèle géométrique de calcul : fractales et barrières de complexité

THÈSE DIRIGÉE PAR :

Jérôme Durand-Lose	Professeur des Universités, Université d'Orléans

RAPPORTEURS :

Véronique Terrier	Maître de Conférence HDR, Université de Caen
Jean-Baptiste Yunès	Maître de Conférence HDR, Université Paris 7

JURY :

Denys Duchier	Professeur des Universités, Université d'Orléans – *examinateur*
Jérôme Durand-Lose	Professeur des Universités, Université d'Orléans – *directeur de thèse*
Géraud Sénizergues	Professeur des Universités, Université Bordeaux 1 – *président du jury*
Véronique Terrier	Maître de Conférence HDR, Université de Caen – *rapporteur*
Jean-Baptiste Yunès	Maître de Conférence HDR, Université Paris 7 – *rapporteur*

à mes parents

Remerciements

Le jour de la soutenance de thèse, après s'être vu décerner le titre de docteur (ce qui est plutôt sympa), on se trouve un peu pris de court pour le discours et les remerciements à chaud (sauf pour ceux qui les ont suffisamment préparés avant, ce qui n'était pas mon cas). Les remerciements du manuscrit permettent de remédier à cela : il est plus facile avec le recul et à tête reposée de penser à toutes les personnes qui nous ont soutenus et aidés, d'une manière ou d'une autre, à mener la thèse jusqu'à son terme. Sauf qu'en fait, ce n'est pas plus facile du tout.

Tout d'abord, j'adresse mes sincères remerciements à tous les membres de mon jury de thèse pour avoir accepté de s'investir dans cette tâche et d'y avoir montré le plus grand intérêt : à Géraud Sénizergues, qui m'a fait l'honneur de présider ma soutenance ; à Véronique Terrier et Jean-Baptiste Yunès, qui ont accepté d'être mes rapporteurs et dont l'investissement, la méticulosité et le sérieux des relectures et commentaires m'ont permis d'améliorer ce manuscrit et de réfléchir à de nouvelles problématiques ; à Denys Duchier, avec qui j'ai eu la chance de collaborer et qui a accepté d'être examinateur.

Il n'y a pas de thèse possible sans directeur de thèse pour travailler, guider, discuter, apprendre le métier de chercheur, conseiller ou jouer à la coinche. Donc merci à Jérôme, qui a fait preuve d'une patience d'ordre ω, ayant toujours un moment à me consacrer malgré toutes ses responsabilités administratives, familiales... et qui a su me recadrer lorsque je m'éloignais de mon sujet de thèse.

Je remercie l'équipe GAMoC, qui m'a accueilli et m'a fourni un cadre de travail idéal et scientifiquement très enrichissant, ainsi que tous mes collègues et collaborateurs ; merci en particulier au chef de l'équipe, Ioan, ainsi qu'à Vincent, Florent, et Mathieu. Le LIFO dans son intégralité constitue lui aussi un environnement de travail à la fois agréable et stimulant, et je souhaite à tous les doctorants de bénéficier d'un tel accueil et cadre. Il en a été de même au sein du département informatique et de toute l'équipe pédagogique, et je remercie en particulier Ali et Catherine, pour la confiance qu'ils m'ont accordée concernant l'enseignement ; Isabelle et Florence, qui étaient toujours là pour rattraper mes retards et bourdes concernant tout ce qui est administratif ; et tous les collègues avec qui j'ai eu le plaisir d'enseigner.

Les autres doctorants (et aussi les jeunes docteurs, pour qui la galère de fin de thèse est encore fraîche en mémoire) sont plus aptes que toutes autres personnes à comprendre et partager les diverses périodes que l'on connaît durant la thèse. Ces rencontres et moments de partage entre doctorants orléanais (en info, en sciences, ou en divers) ont été nombreuses et prolifiques en bons moments, et ce, à diverses occasions (ADSO, week-ends ski, doctoriales, manifestations scientifiques, soirées, foot, pétanque, événements d'assos de doctorants ou d'étudiants...). Je remercie tous les doctorants du LIFO, pour la bonne ambiance qu'ils ont créée au sein du labo,

et j'ai aussi une pensée pour l'équipe de foot du LIFO (dans laquelle ne figurent pas que des doctorants, mais également des anciens), dont l'esprit d'équipe et l'amour du beau jeu a permis de gagner le tournoi interlabos (et deux fois de suite en plus). Comme c'est le but du jeu des remerciements, je me risque à une liste exhaustive * et je remercie donc en particulier : Mathieu, Matthieu, Nicolas, Sylvain, Yohan, Anthony, J-H, Julien, Simon, Yannick, Hélène, Bastien, Thang, Joeffrey, Romain, Abdel, Irénée. Toutes les personnes que j'ai oubliées se reconnaîtront : ce sont celles avec qui j'ai travaillé, discuté, ri, fait une activité quelconque mais distrayante, ou juste passé des soirées et des bons moments. Je finis cette liste par les amis d'ici et là (Jura, Besançon, Paris) et de bien avant la thèse, mais qui y ont quand même contribué indirectement : Raza, Marion, Teddy, François, Pierre, et Antoine.

Merci à Claire, qui était à mes côtés dès le début de ma thèse : elle y est restée à tout moment, que ce soit les moments de joie comme ceux de doute. Son soutien a grandement contribué à la réussite de cette thèse, et je lui en suis reconnaissant.

Enfin et surtout, surtout, un grand grand merci à toute ma famille, au sens le plus large. Merci en particulier à Martine et Alain, pour m'avoir hébergé quelques temps au début de ma thèse et avant, et pour être venu à ma soutenance. Merci à Fabrice, Anthony, Julien, Edwige, Virginie, Léa, Robtoine et Romane. Et enfin, merci à mes parents, qui ont fait preuve d'une patience infinie (au moins d'ordre non dénombrable), et cela depuis plus de trente ans. En plus du soutien indéfectible et des encouragements qu'ils m'ont apportés depuis toujours, ils ont dû pendant ces quatre dernières années expliquer à l'entourage que non seulement j'étais encore étudiant, mais qu'en plus, ce que j'étudiais ne servait à rien. Donc un grand merci à eux, sans qui je n'en serais littéralement pas là.

Maxime

∗. Conditions générales : (1) En fait cette liste n'est pas du tout exhaustive. (2) En la lisant, vous renoncez à toute réclamation ou plainte en cas d'oubli de votre prénom dans la liste sus-citée. (3) En cas d'oubli de votre prénom et sur présentation d'arguments justifiant d'un cas exceptionnel, une bière peut éventuellement être envisagée en guise de réparation.

Sommaire

Liste des Figures

Liste des Tableaux

Introduction

Quand on parle de géométrie, on pense aux constructions à la règle et au compas telles que celles enseignées au collège et au lycée, c'est-à-dire à la *géométrie euclidienne dans le plan*. Bien que n'étant qu'un cas particulier parmi les diverses géométries utilisées en mathématiques modernes, la géométrie euclidienne garde encore la grande importance qu'elle a acquise dans la Grèce antique. En tant que modèle de raisonnement axiomatique, elle contribua largement au formidable développement des mathématiques abstraites il y a plus de 2 000 ans.

L'essor de cette nouvelle science se fit aussi à travers l'arithmétique, conjointement à la géométrie euclidienne. La notion d'*algorithme* a été essentielle dans ces deux domaines, qui présentent des similarités au niveau des raisonnements et techniques utilisés. C'est le cas du principe d'itération, qui trouve des application à la fois en arithmétique avec l'algorithme d'Euclide (calcul du pgcd par itération de divisions euclidiennes) et en géométrie avec le premier exemple connu de *fractale* : les cercles d'Appolonius (pavage d'un triangle curviligne par des cercles en itérant la construction de cercles tangents). Il est clair que, par nature, l'arithmétique permet le calcul. Il se trouve en fait que la géométrie euclidienne, *a priori* éloignée des manipulations de symbole nécessaires au calcul, fournit aussi des outils permettant d'appliquer des méthodes effectives et des algorithmes, prenant la forme d'*algorithmes géométriques*. Plusieurs questions naturelles se posent : peut-on effectuer autant de calculs avec des algorithmes géométriques qu'avec des algorithmes classiques ? peut-on calculer *plus efficacement* ? et si oui, comment procéder ?

La géométrie euclidienne dispose pour cela d'une caractéristique importante : un *espace continu*. La présence d'un espace fournit un cadre pour calculer de manière *parallèle* (plusieurs calculs menés simultanément), contrairement à l'arithmétique où les calculs sont exécutés séquentiellement (une opération après l'autre). De plus, la nature continue de l'espace permet une technique d'itération infinie, qui engendre facilement des structures régulières et auto-similaires : des *fractales*. De nombreuses fractales contiennent implicitement la notion de parallélisme, et leur déploiement dans le plan euclidien repose sur la répétition simultanée de simples motifs. Il apparaît donc naturel d'essayer de tirer profit de ces propriétés dans un contexte de calcul géométrique, et d'étudier les liens entre calculs parallèles et fractales dans le cadre de la géométrie euclidienne. C'est ce que nous cherchons à faire ici, avec entre autres comme objectif la possibilité de mener du *calcul fractale* dans le plan euclidien. Il faut pour cela avant tout définir un cadre d'utilisation du plan euclidien et le munir d'une dynamique, indispensable à la notion de calcul. Une manière simple de procéder est illustrée par la Fig. 1 : différents segments de droite y sont simultanément prolongés de manière uniforme vers le haut.

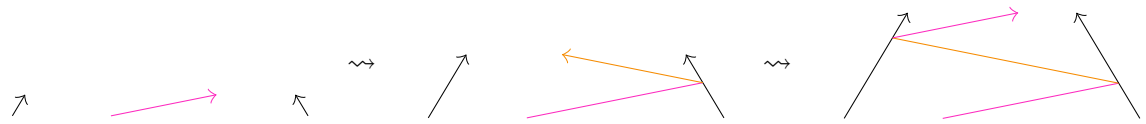


Figure 1 – Prolonger des segments colorés en suivant certaines règles.

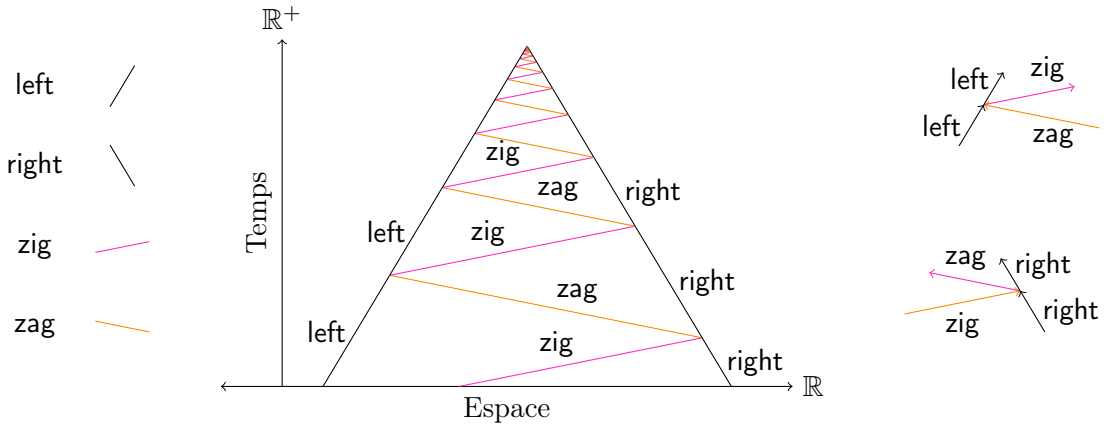
C'est dans ce contexte que nous nous plaçons : nous nous intéressons à des constructions comme celle de la FIG. 1. De telles figures géométriques impliquent uniquement des segments rectilignes colorés qui se prolongent dans le plan euclidien. Les segments initialement présents (tout en bas) sont prolongés en même temps, de manière uniforme et suivant une direction privilégiée (vers le haut). À l'intersection entre plusieurs de ces segments colorés, les segments arrivant sont remplacés par d'autres segments, en suivant certaines contraintes. La FIGURE 1 illustre les étapes d'une telle construction à trois instants différents. Ce procédé simple produit des figures naturellement munies d'une *dynamique* uniforme (donnée par la prolongation des segments vers la haut). De plus, il engendre naturellement des constructions aux propriétés de régularité et d'auto-similarité déjà mentionnées, les fractales. La FIGURE 1 suggère déjà cette aptitude à générer des fractales en répétant un motif simple (un zig-zag dans ce cas) à différentes échelles.

Les fractales ont joué un rôle important dans l'histoire des mathématiques et dans le développement de l'analyse réelle, notamment pour la compréhension de concepts comme la continuité ou la dérivabilité. Depuis, elles n'ont cessé de trouver des applications dans de nombreux domaines différents (mathématiques, modélisation en physique, en finance...). De nature géométrique itérative, elles peuvent jouer un rôle particulier dans la conceptualisation du « parallélisme géométrique ». Mais à notre connaissance, à part certaines constructions d'automates cellulaires, l'étude et l'utilisation méthodique de fractales à des fins de programmation géométrique parallèle sont inexistantes. La perspective d'étude des fractales à des fins d'algorithmique géométrique pourrait définir de nouvelles problématiques qui nous semblent importantes pour la compréhension du *calcul géométrique abstrait*, notamment en ce qui concerne le rôle des fractales dans le découpage automatique d'un espace géométrique et son utilisation pour y mener des calculs de manière distribuée.

Les machines à signaux. Une telle étude nécessite une cadre formel permettant d'exprimer à la fois les notion de calcul géométrique, calcul parallèle et structures fractales. Un *modèle de calcul géométrique abstrait* se distingue particulièrement : le modèle des *machines à signaux*. En permettant d'interpréter des figures géométriques comme la FIG. 1 en termes de machine, calcul, algorithme, les machines à signaux apparaissent très adaptées pour l'étude de la notion de parallélisme géométrique et de calcul fractale : nous allons montrer dans cette thèse qu'elles le sont effectivement.

Ce modèle introduit dans [Durand-Lose 2003] est inspiré du modèle des automates cellulaires et forme de ce fait un modèle naturellement parallèle. Il constitue, contrairement à ces derniers, un système dynamique à temps et espace continus. Les machines à signaux permettent de décrire formellement, à l'aide de primitives de calcul de type géométrique, les constructions géométriques décrites précédemment : les objets de base y sont les *signaux* (traçant des segments de droite) dont la vitesse (la pente) et le type (la couleur) correspondent à des *méta-signaux*. L'opération élémentaire qui permet le calcul par signaux est l'interaction entre ces signaux, modélisée sous forme de *collisions* (les intersections des segments) dont le comportement est décrit de façon déterministe par des *règles de collisions*. Une machine à signaux sera formellement donnée par de tels ensembles finis : un ensemble de méta-signaux (chacun étant muni d'une vitesses) et un ensemble de règles de collisions. La construction géométrique proposée en FIG. 1 se modélise ainsi par la machine à signaux définie par la FIG. 2 : les méta-signaux y sont listés (FIG. 2(a)) et correspondent aux quatre types de segments de couleurs et de pentes différentes ; ce qui passe à l'intersection de deux segments colorés est dicté par deux règles de collisions (FIG. 2(c)). La FIGURE 1 est alors reprise et complétée (FIG. 2(b)), et sa dynamique correspond maintenant à l'évolution d'une machine à signaux. Nous appelons la représentation dans le plan euclidien de cette évolution un *diagramme espace-temps*.

Les diagrammes espace-temps correspondent aux constructions géométriques décrites précé-



(a) Méta-signaux. (b) Diagramme espace-temps. (c) Règles de collisions.

FIGURE 2 – Formalisation de la FIG. 1 sous forme d'une machine à signaux.

demment, dans lesquelles deux directions dans le plan sont privilégiées : la verticale, donnant l'axe du temps (l'ensemble $\mathbb{R}^+$), et l'horizontale, correspondant à l'espace (l'ensemble $\mathbb{R}$) de la machine. Le plan euclidien ainsi décomposé permet de décrire complètement l'évolution dans le temps d'une machine à signaux à une seule dimension spatiale c'est-à-dire avec des signaux se propageant sur la droite réelle.

Remarquons que le processus de zig-zag dans la FIG. 2 se répète indéfiniment, produisant ainsi une figure qui comporte une infinité de segments mais qui reste néanmoins de hauteur finie. Ce phénomène est appelé une *accumulation*, et est une signature des espaces continus. L'essence des accumulations peut être comprise par le biais d'un des paradoxes de Zénon, le paradoxe de la dichotomie : une partie continue peut être découpée une infinité de fois en parties de plus en plus petites, sans que cela nécessite toutefois un temps infini. Une infinité d'étapes de calcul pourrait ainsi être réalisée en une durée finie grâce à la continuité de l'espace. Certaines accumulations se distinguent par leur régularité et leur auto-similarité : elles correspondent en fait à des figures évoquées précédemment, les fractales. Les structures fractales sont en effet un cas spécial d'accumulations, dans lesquelles un motif (généralement simple) se répète à l'identique une infinité de fois.

Le modèle des machines à signaux est récent, mais sa puissance de calcul est déjà bien cernée : il existe déjà de nombreuses comparaisons avec d'autres modèles de calcul,plus anciens et aux propriétés mieux connues. Mais la plupart de ces simulations traitent de modèles séquentiels, et nous renseignent peu sur le parallélisme inhérent aux machines à signaux, ainsi que sur la manière d'utiliser l'espace continu pour y implémenter des calculs parallèles. En particulier, les phénomènes d'accumulations, dont une première formalisation dans le cadre des machines à signaux a été fournie dans [Durand-Lose 2003], ont déjà été utilisées dans des constructions par machines à signaux, mais surtout de manière séquentielle et rarement dans l'optique de calculs parallèles.

Problématique. À notre connaissance, l'étude et l'utilisation méthodique de fractales à des fins de programmation géométrique parallèle sont inexistantes. De nombreuses fractales contiennent implicitement la notion de parallélisme, et leur caractère géométrique trouve dans les machines à signaux un cadre d'étude naturel. Ainsi, la perspective d'étude des accumulations et des fractales à des fins d'algorithmique géométrique définit de nouvelles problématiques qui nous semblent importantes à la fois pour la compréhension des machines à signaux et pour celle du calcul géométrique en général. Les études menées dans le cadre du calcul géométrique abstrait

ont pour le moment été concentrées sur des constructions séquentielles, et l'aptitude des machines à signaux à mener efficacement des calculs massivement parallèles, notamment à travers une utilisation méthodique de certaines fractales, reste inconnue. C'est sur cette problématique de calcul fractale dans le contexte des machines à signaux que se fonde cette thèse, dont l'objectif est d'éclaircir les liens encore mal cernés entre parallélisme, calculs distribués et fractales. Une telle étude se doit de déterminer dans quelles mesures les accumulations et fractales peuvent être générées et utilisées algorithmiquement par les machines des signaux : c'est ce que nous cherchons à accomplir ici en se concentrant sur le découpage et l'utilisation méthodique de l'espace à travers les fractales. L'un des moyens pour comprendre et illustrer cette puissance de calcul fractale, est d'implémenter par signaux des méthodes de résolution de problèmes algorithmiquement difficiles, et d'étudier les techniques mises en jeu et la complexité des calculs. Un problème classique s'impose naturellement pour une telle étude, car il est représentatif de la classe **NP**, considérée comme une classe de problèmes algorithmiquement difficiles : il s'agit de **SAT**, le problème de satisfaisabilité de formules booléennes. Ce problème, déjà connu pour admettre des résolutions efficaces sur modèles parallèles, tout comme ses variantes telles que **Q-SAT**, fournit donc une cible de choix pour tester le pouvoir de calcul parallèle et fractale des machines à signaux.

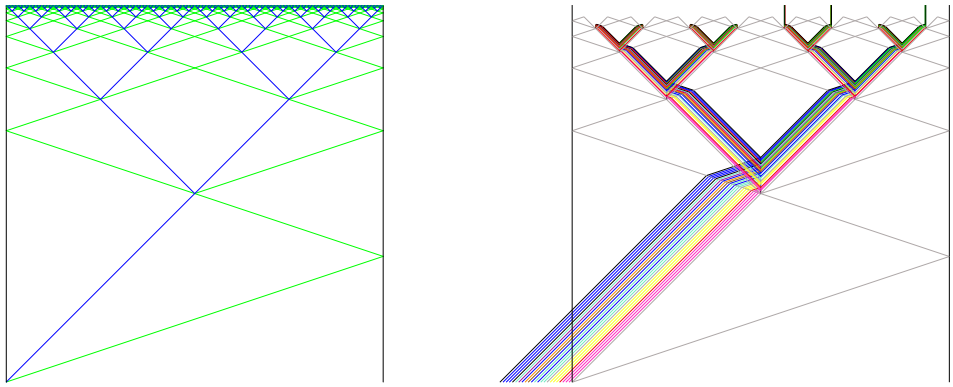
Contributions. Dans le domaine du calcul géométrique par signaux et calcul géométrique fractale, le travail présent apporte trois nouvelles contributions : une étude des accumulations et des fractales, démontrant de nouvelles propriétés ; une méthode de programmation géométrique modulaire, particulièrement adaptée à la nature parallèle des machines à signaux ; et une résolution de problèmes algorithmiques difficiles, combinant l'approche modulaire et l'utilisation de fractales pour mener des calculs massivement parallèles.

La première contribution porte donc sur l'étude de ces phénomènes d'accumulations, en particulier sur le lien entre accumulations et « petites » machines à signaux (c'est-à-dire des machines avec peu de signaux différents, comme celle générant l'accumulation en Fig. 2) ; nous avons dégagé entre autres une *condition nécessaire sur les vitesses (et les positions initiales dans le cas de trois vitesses) d'une machine pour qu'elle puisse engendrer des accumulations.* Certaines « petites » machines permettent, par itération d'un motif géométrique simple impliquant peu de signaux différents, de générer des structures fractales, dont un exemple est fourni en Fig. 3(a). Nous donnons ici différents exemples de fractales générées par signaux, et illustrons sur un exemple deux manières différentes pour approximer une fractale : par une méthode *spécifique* et par une méthode *générique*. L'approximation d'une fractale est ici motivée par la discrétisation automatique de l'espace continu, préparant ainsi la résolution de problèmes combinatoires.

Afin d'utiliser méthodiquement les approximations de fractales et le découpage de l'espace qui en résulte, nous définissons une méthode de *programmation géométrique modulaire* : elle constitue le deuxième apport. À partir de la formalisation dans le cadre des machines à signaux de concepts classiques tels que le codage d'information, la résolution d'un problème, ou la complexité d'un calcul, nous développons la notion de *module géométrique* de calcul. Grâce à celle-ci, des algorithmes géométriques (et les machines à signaux correspondantes) peuvent être conçus à partir de composants de bases : l'assemblage de ces structures élémentaires suivant certains motifs géométriques permettent de construire de nouvelles structures modulaires plus complexes. Parmi ces notions de *composition de modules*, celles de composition parallèle et de composition itérative se trouvent particulièrement adaptées pour la génération et l'utilisation des fractales.

La troisième contribution est constituée par plusieurs résolutions par signaux de problèmes combinatoires classiques, considérés comme algorithmiquement difficiles : les problèmes de satisfaisabilité de formules booléennes. Ces solutions résultent de l'utilisation combinée des fractales et de l'approche modulaire. L'approximation d'une certaine fractale, l'arbre binaire infini de la Fig. 3(a), permet de *fractaliser* l'espace pour y construire une grille de calcul ; cette grille per-

met par le biais de l'utilisation de différents modules de distribuer et d'effectuer des calculs dans l'arbre fractale, comme cela est illustré par la Fig. 3(b).



(a) Un arbre binaire fractale [Durand-Lose 2003, Fig. B.1(b)].

(b) Un calcul se propageant dans l'arbre.

Figure 3 – Distribuer un calcul grâce à une structure fractale.

Cette méthode s'applique particulièrement bien à la résolution des problèmes **SAT** et **Q-SAT**, les problèmes respectifs de satisfaisabilité de formules booléennes et booléennes quantifiées. Nous donnons ici deux méthodes de résolution différentes, qui ont fait l'objet des publications [Duchier *et al.* 2010a] et [Duchier *et al.* 2012], correspondant respectivement à une résolution spécifique de **SAT** et à une résolution générique de **Q-SAT**. La modularité de ces constructions permet également d'en dériver facilement des solutions à certaines variantes comme #**SAT** et **Max-SAT**. Nous démontrons grâce à l'approche modulaire la correction de ces constructions, mais aussi leur efficacité. Munis de mesures de complexité naturellement induites par la nature du modèle, nous pouvons évaluer l'efficacité relative de ces constructions : le problème **Q-SAT**, connu pour être **PSPACE**-complet, est ainsi résolu en *profondeur de collisions* polynomiale, c'est-à-dire avec un algorithme géométrique en temps polynomial.

Structure du manuscrit. Le manuscrit est ordonné de la manière suivante. Un état de l'art est proposé dans le Chapitre 1. Le Chapitre 2 regroupe les définitions et des propriétés des machines à signaux. Le Chapitre 3 fournit les premiers exemples d'algorithmes géométriques et introduit les notions nécéssaires pour l'étude des algorithmes géométriques. Les résultats obtenus sur les accumulations et la génération de fractales par machines à signaux sont détaillés respectivement dans les Chapitres 4 et 5. L'approche de programmation géométrique modulaire est ensuite formalisée dans le Chapitre 6, par les définitions de compositions de machines et de modules géométriques de calcul. Les résultats des deux chapitres précédents sont ensuite appliqués dans les Chapitres 7 et 8, présentant les résolutions respectivement spécifique et générique des problèmes **SAT** et **Q-SAT** ; le Chapitre 9 aborde la résolution de quelques variantes de problème de satisfaisabilité. Enfin, les conclusions et perspectives sont regroupées dans le dernier chapitre.

Chapitre 1

État de l'art : origines et originalités du modèle

Nous proposons ici un état de l'art en trois parties, ordonnées depuis un cadre général (les modèles de calcul) vers un modèle particulier (les automates cellulaires), en finissant par le contexte spécifique qui nous intéresse dans le manuscrit : les machines à signaux. Pour débuter ce chapitre, nous évoquerons divers modèles de calcul, classiques et non-classiques, en prenant soin de détailler quelques uns des différents modèles géométriques de calcul existants dans la littérature. Ensuite, nous ciblerons un modèle de calcul en particulier, essentiel pour comprendre l'origine et la motivation des machines à signaux puisqu'elles en sont inspirées : les automates cellulaires. Après avoir rappelé leur définition, nous illustrerons l'importance des signaux — continus et discrets — dans le monde des automates cellulaires. Le chapitre sera clos par un état de l'art spécifique aux machines à signaux, dans lequel seront rappelées les propriétés déjà connues du modèle, en particulier les comparaisons entre la puissance de calcul des machines à signaux et celles de quelques modèles déjà évoqués en première partie de chapitre.

Par ailleurs, un état de l'art succinct et quelques références aux travaux concernant les accumulations, les fractales, et les résolutions de problèmes difficiles sur différents modèles de calcul (surtout non-classiques) seront donnés de manière spécifique dans les introductions des chapitres traitant ces notions (respectivement les Chapitres 4, 5 et 7). Dans le cas des fractales, nous insisterons particulièrement sur quelques travaux concernant l'étude et la génération de fractales sur des modèles géométriques tels que les automates cellulaires ou l'auto-assemblage.

1.1 Les modèles de calcul

1.1.1 Modèles classiques

Un modèle de calcul définit ce qu'est « être calculable » pour ce modèle donné, selon certaines opérations de base autorisées. Cette notion de primitive de calcul est importante, puisque c'est elle qui conditionne la forme que prend le calcul dans le modèle considéré.

Les notions intuitives d'opération élémentaire et de méthode effective, ainsi que celles qui en découlent comme « être calculable », ont été fortement étudiées dans les années 1930. À cette époque et presque simultanément, Alan Turing, Alonzo Church, Kurt Gödel et Stephen Kleene proposèrent et utilisèrent des modèles de calcul et des formalismes permettant de définir des notions de calculabilité et d'effectivité, respectivement en termes de machines de Turing [Turing 1936], de λ-calcul [Church 1933] et de fonctions récursives [Gödel 1931 ; Kleene 1936] (on trouvera ces articles dans [Davis 2004], qui compile des articles fondateurs de la théorie de la calculabilité). Bien qu'*a priori* différentes, ces notions se sont avérées être par la suite toutes équivalentes entre elles : elles définissent les mêmes fonctions calculables et permettent de résoudre exactement

les mêmes problèmes. Ce constat est à l'origine de la thèse de Church-Turing, que l'on peut formuler par « tout ce qui est effectivement calculable l'est par une machine de Turing ». Cette thèse est vérifiée par de nombreux autres modèles de calcul imaginés par la suite pour cerner la notion de calculabilité effective, comme les systèmes de réécriture [Post 1943], les automates cellulaires [von Neumann 1966], les automates à compteurs [Minsky 1967]... Notons que bien que ces modèles définissent la même classe de fonctions calculables, ils utilisent des primitives de calcul et opérations élémentaires différentes pour obtenir ces fonctions : transition en fonction d'un état et d'un symbole, λ-réduction, composition ou minimisation de fonctions, transition en fonction du voisinage...

La thèse de Church-Turing originale peut être vue comme une thèse sur la puissance de ces systèmes formels. Elle admet cependant des variantes, comme la version physique nommée « thèse M » dans [Gandy 1980], portant sur le monde réel : « ce qui peut être calculé par une machine physique peut l'être par une machine de Turing »,ou comme les versions de Church-Kalmár-Turing : « toute fonction réalisable par un système artificiel de calcul (resp. par une expérience de pensée) est calculable par machine de Turing » [Etesi et Németi 2002].

1.1.2 Modèles non-conventionnels

La thèse de Church-Turing a été confortée par toutes les implémentations physiques de modèles de calcul réalisées jusqu'à aujourd'hui. Il est néanmoins possible d'imaginer des modèles qui dépassent le cadre de la thèse de Church-Turing : de tels modèles de calcul sont appelés *modèles de calcul non-conventionnels*. Ceux-ci peuvent s'écarter du cadre classique pour plusieurs raisons : ils peuvent être définis à partir de primitives de calcul totalement différentes de celles classiquement utilisées ; ils peuvent être de nature à manipuler des valeurs réelles ou continues (contrairement aux modèles classiques qui ont vocation à ne travailler qu'avec des valeurs discrètes) ; ou ils peuvent tirer profit de caractéristiques spéciales, telles que l'accès à des informations supplémentaires ou bien l'utilisation de ressources non-classiques. Pour un tour d'horizon sur les modèles continus et un aperçu des différents modèles « super-Turing » (c'est-à-dire calculant plus qu'une machine de Turing), nous renvoyons respectivement à [Bournez et Campagnolo 2009] et [Stannett 2006].

Nous donnons ici quelques exemples de tels modèles parmi toute la variété existante, que nous regroupons de manière subjective en deux classes : les modèles purement théoriques et les modèles inspirés du monde réel. Nous détaillons à part les modèles géométriques du fait de leur proximité avec les travaux présentés dans ce manuscrit. Ces modèles géométriques sont essentiellement purement théoriques.

Modèles non-conventionnels théoriques. Alan Turing fut l'un des premiers à envisager, en proposant les machines à oracles [Turing 1939], la possibilité de « calculer plus » ou de « calculer autrement » que ce que permettent les modèles classiques, c'est-à-dire qu'il envisagea de sortir du cadre de la thèse de Church-Turing. Les machines de Turing à oracle permettent de résoudre des problèmes non résolubles par une machine de Turing standard : elles ont pour cela accès à un oracle qui peut donner les valeurs de fonctions non-calculables ou des solutions de problèmes normalement indécidables. D'autres modèles ont été étudiés depuis pour leur capacité à être « super-Turing » ou pour leur capacité à manipuler des valeurs réelles ou continues, et non plus seulement des valeurs discrètes comme le font les modèles classiques. On citera entre autres : les machines de Turing de type 2 et l'analyse récursive [Weihrauch 2000] qui étendent la définition des machines à Turing (travaillant sur des objets finis) pour l'adapter aux réels (représentables par des objets infinis) ; les machines de Blum, Shub et Smale (modèle BSS) [Blum *et al.* 1989] manipulant des réels de manière exacte ; les fonctions $\mathbb{R}$-récursives [Moore 1996], version continue des fonctions récursives classiques dans lesquelles les opérateurs discrets sont remplacés par des

opérateurs continus comme l'intégrale ; les machines à ordinaux [Lafitte 2001 ; Hamkins et Lewis 2000], qui définissent des machines de Turing à ruban et à temps indexés par des ordinaux, fonctionnant comme des machines classiques mais avec des transitions supplémentaires pour les ordinaux limites ; les machines accélérantes [Copeland 2002 ; Potgieter 2006], pouvant être définies comme des machines de Turing effectuant leurs transitions successives de manière accélérée. Ces dernières, déjà évoquées par Ralph Blake et Bertrand Russell dans les années 1930, calculent avec une accélération du type du paradoxe de Zénon : elles opèrent leurs étapes successives de calcul de plus en plus rapidement, de telle sorte que l'exécution d'une infinité d'étapes de calcul ne demande qu'un temps fini. Elles permettent entre autres de résoudre des problèmes récursivement énumérables, qui sont indécidables sur les modèles classiques, tels que le problème de l'arrêt.

S'inspirer du réel pour calculer. Les modèles évoqués ci-dessus ont des propriétés qui sortent du cadre de la calculabilité au sens de Church-Turing, mais ils restent tous des modèles théoriques, basés essentiellement sur des propriétés mathématiques. D'autres modèles, inspirés du monde réel, possèdent également certaines de ces propriétés non classiques mais avec des hypothèses « plus réalistes ».

Parmi les modèles inspirés du monde réel, certains sont inspirés de la physique : le GPAC (General Purpose Analog Computer) [Shannon 1941], qui utilisent la possibilité de calculer physiquement (soit mécaniquement, soit électriquement) des intégrales et des dérivées ; les différents modèles de machines quantiques [Feynman 1986 ; Deutsch 1985], basés sur les propriétés de la mécanique quantique ; les modèles relativistes [Hogarth 1994 ; Etesi et Németi 2002 ; Németi et Dávid 2006], qui, dans le cadre des théories de la relativité, utilisent les singularités de l'espace-temps (telles que les trous noirs) pour accélerer des calculs ; les divers modèles de machines optiques [Naughton et Woods 2005 ; Goliaei et Foroughmand-Araabi 2012 ; Reif *et al.* 1990], basés soit les propriétés mécaniques de la lumière, soit sur ses propriétés ondulatoires ; les bouliers et les balances [Arulanandham 2005] et les arrangements de dominos [Stevens 2011], modèles plus anecdotiques qui permettent de calculer au sens de Turing par la simple utilisation de la gravité. On notera qu'il existe un lien entre les modèles relativistes et les machines accélérantes, précédemment évoquées : les modèles relativistes de calcul utilisent en effet les trous noirs et le phénomène de « gel du temps » pour accélérer les étapes de calcul d'une machine de Turing à la manière des machines accélérantes c'est-à-dire selon le même principe que le paradoxe de Zénon.

D'autres formalismes sont directement basés sur des processus issus de la biologie et du monde du vivant comme les réseaux de neurones [Siegelmann et Sontag 1995], inspirés par le fonctionnement d'un cerveau humain ; le calcul par ADN [Amos 1997 ; Păun *et al.* 1998], utilisant les composants de l'ADN et les mécanismes correspondant pour y coder des informations et des instructions ; le calcul par membranes, molécules et les P-systèmes [Păun 2000 ; Păun et Rozenberg 2002], utilisant comme opérations de base les différents comportements possibles des cellules biologiques comme la dissolution, la mitose, la fusion, *etc.*

Ces différentes façons de définir et d'envisager le calcul d'un point de vue inspiré par des processus du monde réel peuvent se regrouper sous la dénomination de *calcul naturel* ou « natural computing ». Nous renvoyons à [Gramß *et al.* 1998] pour une présentation de quelques uns des modèles de calcul non-standard évoqués ci-dessus, ainsi qu'à [Handbook of Nat. Comp. 2012] pour une introduction plus exhaustive et détaillée au calcul naturel, et à [UCNC 2012] pour un aperçu de quelques thèmes de recherche actuelle autour du calcul non-conventionnel et naturel.

L'étude de ces différents modèles, conventionnels ou non, permet de mieux comprendre des notions théoriques fondamentales comme la calculabilité et la complexité des problèmes algorithmes (c'est-à-dire leur « coût »). Dans le cas des modèles non-conventionnels, elle peut également

permettre de mieux cerner certaines propriétés du monde réel et ses descriptions scientifiques, notamment les connexions qui existent entre les mathématiques, l'informatique fondamentale et la physique théorique. L'une d'elle concerne la thèse de Church-Turing, qui peut être relativisée à certaines hypothèses physiques, et son statut dépend du cadre et de la théorie physique dans laquelle se place son étude : considérer la thèse de Church-Turing comme vraie peut ainsi infirmer ou confirmer certaines théories physiques. Nous renvoyons à [Ziegler 2009] pour une discussion sur ces liens entre physique et informatique théorique.

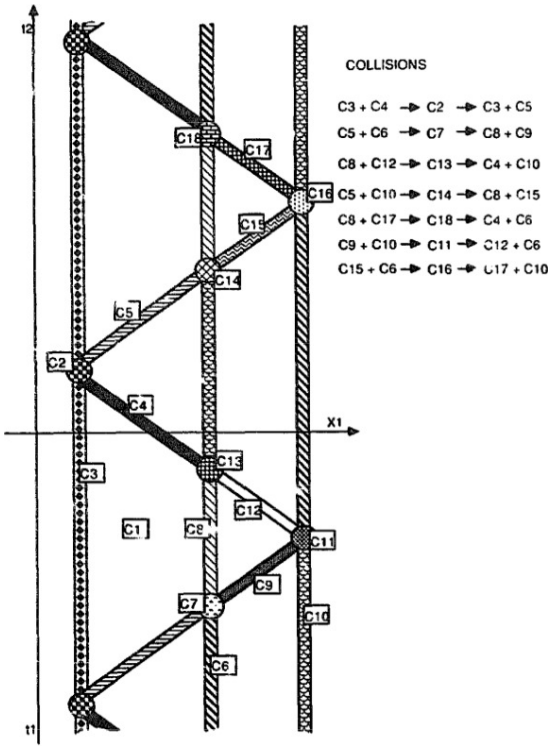
Modèles géométriques. Parmi ces modèles non-conventionnels, nous allons particulièrement nous intéresser aux modèles de nature géométrique, qu'ils soient définis à partir de primitives de calcul géométriques ou bien qu'ils se prêtent à une interprétation géométrique seulement à partir d'un niveau plus global.

Comme nous l'avons évoqué dans l'introduction, il est possible de calculer dans le cadre de la géométrie euclidienne, en utilisant uniquement la règle et le compas. Une telle démarche a été formalisée dans [Huckenbeck 1989, 1991] et dans [Mycka *et al.* 2006], respectivement en termes d'*automates géométriques* et de *machines euclidiennes abstraites*. Dans les deux cas, les primitives de calcul sont essentiellement « tracer une droite » et « tracer un cercle ». Il est alors possible d'écrire des programmes dont les instructions correspondent à ces primitives de calcul. De tels algorithmes permettent non seulement d'exprimer les constructions classiques de géométrie euclidienne telles que le tracé d'une médiatrice, d'une bissectrice... mais ils permettent également de calculer de larges classes de fonctions. Différents types d'automates géométriques sont considérés dans [Huckenbeck 1989, 1991] (avec ou sans instruction de type `jump`, test de parallélisme de droites...), ainsi que les classes de fonctions calculables par ces automates géométriques. Dans [Mycka *et al.* 2006], il est démontré que de telles machines géométriques bénéficient d'un pouvoir de calcul Turing-universel par leur aptitude à simuler des machines à registres (machines RAM) uniquement en traçant des droites, des cercles, en les étiquetant et en testant si un point est dans un cercle. Néanmoins, certaines fonctions restent non-calculables par ces constructions, conformément aux théorèmes de géométrie euclidienne classique, comme la fonction trisectrice de l'angle.

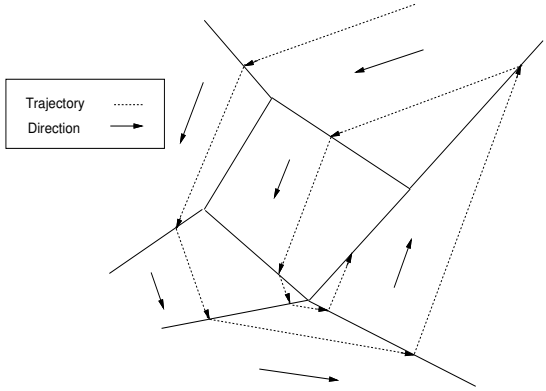
Il est également possible de considérer des modèles géométriques en s'abstrayant des outils classiques de la règle et le compas. C'est le cas par exemple des *univers colorés et automates de Mondrian* introduits dans [Jacopini et Sontacchi 1990]. Ces univers sont des espaces continus (de dimension finie quelconque) dans lesquels sont assemblés des polyèdres. Les jonctions de polyèdres sont définies comme valides si elles respectent certaines contraintes de superposition locale exprimées en termes de voisinages topologiques. Ces contraintes déterministes sur les voisinages peuvent être interprétées comme des intersections entre les polyèdres, et sont ensuite reformulées en termes de boules colorées dénommées collisions. Un exemple d'évolution d'un tel univers coloré est donné par la Fig. 1.1(a). L'étude de ce modèle est motivée par des contraintes d'ordre physique sur le calcul, notamment sur la communication entre processus dans l'espace et sur la réversibilité des opérations. Les auteurs ont caractérisé des classes de fonctions calculables par ce modèle, en particulier dans le cas d'univers colorés réversibles : le modèle général permet facilement de simuler des machines classiques, dont des machines de Turing, et est donc Turing-universel.

Un autre modèle de calcul à forte interprétation géométrique est le modèle des *systèmes à dérivées constantes par morceaux*, ou *PCD systems* (pour « piece-wise constant derivative systems »), introduit dans [Asarin et Maler 1995]. Ces systèmes dynamiques à temps et espace continus sont définis en partitionnant un espace euclidien en un nombre fini de polyèdres convexes, chaque région polyédrale étant affectée d'un vecteur fixe. Lorsqu'un point pénètre dans une région polyédrale, sa trajectoire est définie en lui appliquant le vecteur associé à cette région, jusqu'à

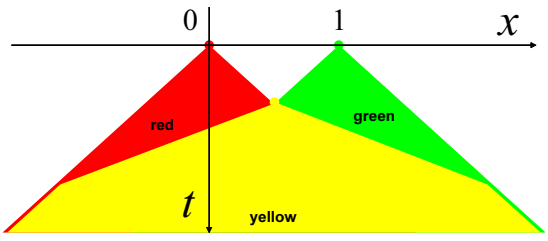
ce qu'il la quitte et entre dans une autre région, munie d'un autre vecteur fixe. Pour chaque point initial, on obtient en appliquant ce processus une trajectoire constituée de segments mis bout-à-bout dans l'espace euclidien. La trajectoire globale ainsi obtenue est affine par morceaux (c'est-à-dire à dérivée constante par morceaux) : la FIGURE 1.1(b) illustre un tel système en dimension deux, ainsi que la trajectoire d'un point. Ce modèle est dit hybride, mêlant à la fois étapes continues (les trajectoires dans une région) et étapes discrètes (les changements à la jonction de deux régions). En identifiant certaines régions polyédrales à des ports d'entrées et de sorties, les systèmes PCD permettent de définir des notions de calcul. Il est alors possible de simuler une machine de Turing, et même de décider des problèmes indécidables sur machines de Turing. En effet, n'importe quel niveau de la hiérarchie arithmétique peut être décidé par des systèmes PCD rationnels (*i.e.* dont tous les paramètres sont rationnels), pourvu qu'ils aient une dimension spatiale assez grande [Asarin et Maler 1995 ; Bournez 1997], L'étude de systèmes PCD généraux (autorisant des paramètres réels) a montré qu'ils peuvent même décider des problèmes hyper-arithmétiques [Bournez 1999]. Dans les deux cas, une version PCD du paradoxe Zénon est implémentée afin d'accélérer le calcul, de telle sorte qu'il est toujours possible de connaître la région finalement atteinte par une trajectoire infinie. Notons que les PCD sont essentiellement séquentiel (les trajectoires considérées ne le sont que pour un seul point à la fois).



(a) Univers colorés [Jacopini et Sontacchi 1990, Fig. 3].



(b) Systèmes à dérivées constantes par morceaux [Bournez 1999, Fig. 2].



(c) Systèmes de transitions sur espace-temps continus [Hagiya 2005, Fig. 1].

FIGURE 1.1 – Exemples de modèles géométriques de calcul.

Un autre modèle à espace et temps continus est constitué par les *systèmes de transitions sur des espaces-temps continus* étudiés par [Takeuti 2005 ; Hagiya 2005]. On peut situer ces systèmes à mi-chemin entre des automates cellulaires rendus infiniment petits et des équations aux dérivées partielles. Les travaux concernant ce type de modèle sont inspirés par l'« amorphous computing », dont le domaine d'étude est l'utilisation d'un très grand nombre d'entités élémentaires ne pouvant communiquer que localement.

La notion importante dans ces systèmes de transitions sur des espaces-temps continus est la propagation d'état à des vitesses différentes, qui est une conséquence des règles de transitions dictant l'évolution de l'état d'un point donné de l'espace. Les règles de transition sont définies à partir de formules du type « le point x passe dans l'état q à l'instant t s'il existe dans le voisinage de (x, t) un point (x', t') dans l'état q' », où chaque voisinage dépend d'une vitesse v et où tous dépendent d'une distance dont l'espace est muni. On peut représenter l'évolution de tels systèmes par des diagrammes espace-temps proches de ceux des machines à signaux : la Fig. 1.1(c) en donne un exemple.

À notre connaissance, les travaux sur ces systèmes portent uniquement sur les propriétés de la trajectoire d'un point c'est-à-dire les états successivement pris par un point de l'espace. Des contraintes sur ces trajectoires ont été formulées à l'aide de formules de logiques modales et de systèmes de règles d'inférences respectivement dans [Takeuti 2005] et [Hagiya 2005], et engendrent une notion de réalisabilité des trajectoire. Dans les deux cas, des conditions sont dégagées pour assurer l'unicité de telles trajectoires.

Nous évoquons pour finir un autre type de modèle géométrique de calcul, les systèmes d'auto-assemblage et les pavages. Nous passons rapidement sur les pavages, dont l'étude d'un point de vue modèle de calcul prend source dans les travaux [Wang 1960 ; Berger 1966], où sont données des simulations globales de machines de Turing par des pavages du plan. Les pavages sont bien évidemment fortement géométriques, mais ils n'admettent *a priori* pas de dynamique.

Un type particulier de pavages se distingue d'un point de vue modèle de calcul : il s'agit des systèmes d'auto-assemblages. Initialement introduits en tant que pavages par Hao Wang, ces systèmes se définissent par des jeux de tuiles planes aux arêtes colorées et sont munis d'une dynamique [Winfree 1998]. À partir de tuiles disposées dans le plan euclidien, de nouvelles tuiles viennent se coller de manière asynchrone, tout en respectant les couleurs des arêtes correspondantes (des contraintes supplémentaires simples peuvent également être ajoutées, comme des contraintes de « colle » et « température »). Ce modèle trouve une motivation dans la biologie et l'étude du vivant : il permet ainsi de formaliser de manière abstraite les repliements et assemblages complexes de certaines structures moléculaires (ADN, enzymes. . .) Plusieurs variantes de ces modèles et notions correspondantes de calculs existent, mais de manière générale il est possible de calculer de nombreux ensembles en les auto-assemblant, et de facilement simuler une machine de Turing avec des jeux de tuiles auto-assemblantes [Rothemund et Winfree 2000 ; Lathrop *et al.* 2011].

On remarque tout particulièrement une utilisation de *signaux continus* dans certains travaux sur l'auto-assemblage [Becker 2008, 2009]. Ces signaux dans le plan euclidien sont utilisés pour contraindre des dynamiques dans l'auto-assemblage d'une structure : ils permettent d'obtenir par leur discrétisation des jeux de tuiles s'auto-assemblant suivant des formes données. Par exemple, des systèmes de signaux permettent d'expliciter des jeux de tuiles de Wang pour auto-assembler en temps optimal des carrés de toutes tailles, tout en utilisant un faible nombre de tuiles.

En lien avec les pavages, le modèle des automates cellulaires génère des diagrammes espace-temps discrets qui se prêtent à diverses interprétations géométriques : nous détaillons ce modèle de calcul dans la section suivante.

1.2 Le monde des automates cellulaires

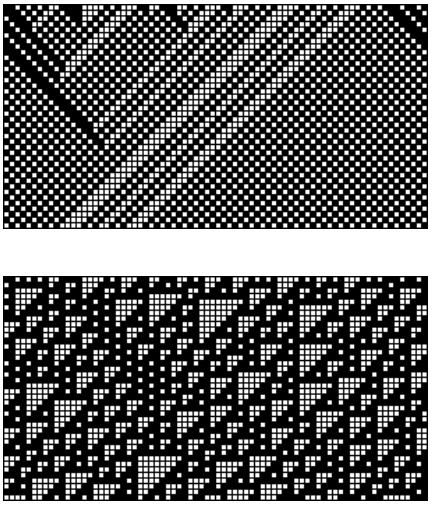
Nous donnons ici plus de détails sur les particularités du modèles des automates cellulaires. Après une brève présentation du modèle et de quelques unes de ses propriétés, nous mettrons en avant le rôle et l'importance des signaux — à la fois continus et discrets — dans l'étude des automates cellulaires.

1.2.1 Introduction des automates cellulaires

Après avoir grandement participé à la théorie et à la construction des premières machines à calculer dans les années 1940, John von Neumann chercha à modéliser le concept de machine capable d'auto-reproduction. Souhaitant en plus que cette machine soit non triviale c'est-à-dire capable d'effectuer des calculs complexes, il proposa sur une idée de Stanislaw Ulam, le modèle des *automates cellulaires (AC)* [von Neumann 1966].

Un automate cellulaire est défini de manière simple : il est défini sur un espace discret composé de cellules juxtaposées les unes à côté des autres. Chaque cellule peut prendre un état parmi un ensemble fini d'états. Toutes les cellules se mettent à jour de manière synchrone (toutes en même temps) selon des étapes de temps discret, et en suivant une règle de transition locale indiquant à chaque cellule le nouvel état à prendre en fonction de son état et de celui de ses cellules voisines. Ainsi, à chaque étape de temps, l'automate met à jour sa configuration c'est-à-dire son état global, défini par les états de chacune de ces cellules.

L'exemple le plus simple d'AC est celui des automates cellulaires élémentaires (ACE). Ceux-ci sont définis sur des lignes de cellules (ils ont donc une seule dimension spatiale), et chaque cellule ne peut prendre qu'un seul état parmi deux. En utilisant la deuxième dimension du plan comme axe du temps, il est possible de représenter l'évolution d'AC de dimension un par un diagramme espace-temps : les configurations successives y sont représentées les unes sous les autres, de telle sorte que le diagramme espace-temps fournit l'historique des états pris par les cellules. La FIGURE 1.2(a) fournit quelques exemples de diagrammes espace-temps d'ACE (dont la règle est donnée selon la numérotation de Wolfram). Ici, tous les diagrammes espace-temps d'AC sont donnés avec l'espace représenté horizontalement et le temps représenté verticalement de haut en bas.



(a) ACE règles 226 et 110.

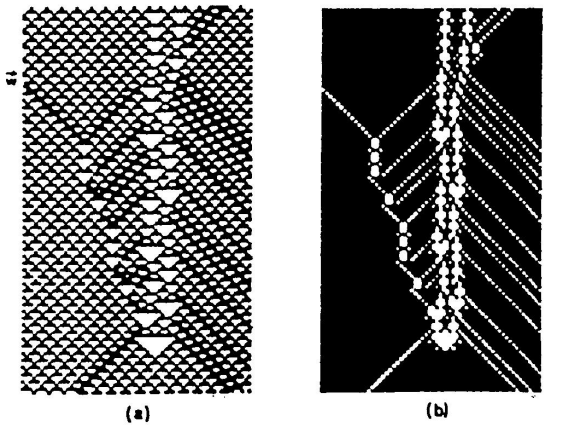


FIG. 7. Rule 54. (a) Annihilation of the radiating particle. (b) The same as (a) with the mapping defined in Fig. 6.

(b) Signaux bruts et filtrés dans l'ACE règle 54 [Boccara *et al.* 1991, Fig. 7].

FIGURE 1.2 – Exemple de diagrammes d'ACE.

Importance et utilisation des automates cellulaires. Depuis leur introduction, les automates cellulaires bénéficient d'un statut particulier dans les sciences. En effet, un AC se définit de manière très simple, mais il peut avoir une dynamique et un comportement extrêmement complexes. Depuis plusieurs dizaines d'années, le modèle des AC a été à la fois un outil et un exemple privilégié pour l'étude des systèmes dynamiques discrets [Hedlund 1969 ; Čulik II *et al.* 1990], dont il est devenu l'un des modèles de références en informatique théorique. D'autre part,

par leur définition même, les automates cellulaires forment un modèle de calcul fortement parallèle [Delorme et Mazoyer 1999]. Ils ont ainsi été utilisés pour modéliser de manière simple les premières agencements parallèles de processeurs. Ils trouvent des applications, en nombre de plus en plus grand, en modélisation de phénomènes physiques, biologiques, sociologiques, tels que des simulations de trafic routier, propagation d'épidémie... [ACRI 2012].

Les signaux dans les AC. Malgré l'apparente simplicité de ce modèle, les automates cellulaires permirent à von Neumann de concevoir une machine à la fois Turing-universelle et capable de s'auto-reproduire. Pour pouvoir être capable à la fois de mener des calculs complexes (qui sont codés et décodés grâces à certains états donnés) et de s'auto-répliquer, la structure de von Neumann utilise de nombreux gadgets techniques dont l'un d'eux est fondamental dans cet automate : il s'agit de la mise en place et de l'utilisation de *fils* pour permettre à l'information de se propager à travers l'espace cellulaire. Ces fils sont essentiels pour transmettre les informations nécessaires à la machine pour procéder aux calculs et à la construction d'autres machines (y compris elle-même). L'information qui se propage dans les fils se présente alors sous forme de *signal rectiligne* se déplaçant dans la structure.

En plus de l'utilisation historique des signaux dans l'automate de von Neumann, il a été constaté depuis que dans la majorité des études empiriques concernant les AC, en particulier des ACE [Wolfram 2002], les diagrammes et le comportement des AC ayant des propriétés intéressantes pouvaient être interprétés et expliqués grâce à des notions — d'abord informelles — de *particules*, *signaux* et *collisions*. C'est le cas de nombreux diagrammes d'ACE (automates à une dimension) dont les diagrammes espace-temps à deux dimensions font clairement apparaître des signaux se propageant à travers l'espace cellulaire au cours du temps. Un exemple en est donné par la Fig. 1.2(b), dans laquelle figurent un diagramme espace-temps d'ACE et la version du diagramme obtenu en appliquant une fonction qui efface les motifs périodiques d'arrière-plan, laissant ainsi clairement apparaître les particules et les collisions du diagramme initial.

1.2.2 L'importance des signaux discrets

Les signaux apparaissent de manière naturelle dans les AC, et sont utilisés à de multiples fins. Nous donnons ici des exemples de leurs utilités ; nous montrons également l'importance de raisonner avec des signaux continus pour la compréhension et la conception d'AC. Nous renvoyons le lecteur à [Adamatzky 2002 ; Adamatzky et Durand-Lose 2012], consacrés au « collision-based computing », dont une grande partie est dédiée à l'utilisation des signaux et des collisions dans les AC et dans d'autres modèles proches tel que le « billard ball model » (modèle boule de billard) [Fredkin et Toffoli 1982 ; Durand-Lose 2002].

Comprendre des comportements. Les premières études expérimentales sur les ACE mirent en évidence différents types de structures apparaissant dans leurs diagrammes espace-temps. Ces structures permirent de proposer une classification des ACE, en distinguant entre autres celles qui impliquaient des particules et des collisions, alors définies de manière informelle [Wolfram 1984].

Nous citons ici un autre exemple de l'utilisation des particules pour mieux appréhender le fonctionnement d'AC : il s'agit de l'évaluation de la performance d'AC engendrés par des algorithmes évolutionnistes [Das *et al.* 1994, 1995]. Dans ces travaux, les signaux sont utilisés pour sélectionner les AC les plus efficaces dans l'accomplissement d'une tâche donnée telle que la synchronisation (voir la Fig. 1.3) ou la classification de densité. Dans ces travaux, les AC impliquant des signaux dans leur comportement ont été constatés comme les plus performants, dégageant ainsi un critère de sélection pour les générations suivantes. Cette méthode a permis de découvrir des AC difficilement concevables manuellement et performants dans l'accomplissement

de certaines tâches : c'est le cas de celui de la FIG. 1.3, dont le comportement peut être compris et prouvé à l'aide de particules qu'une opération de filtrage fait clairement apparaître.

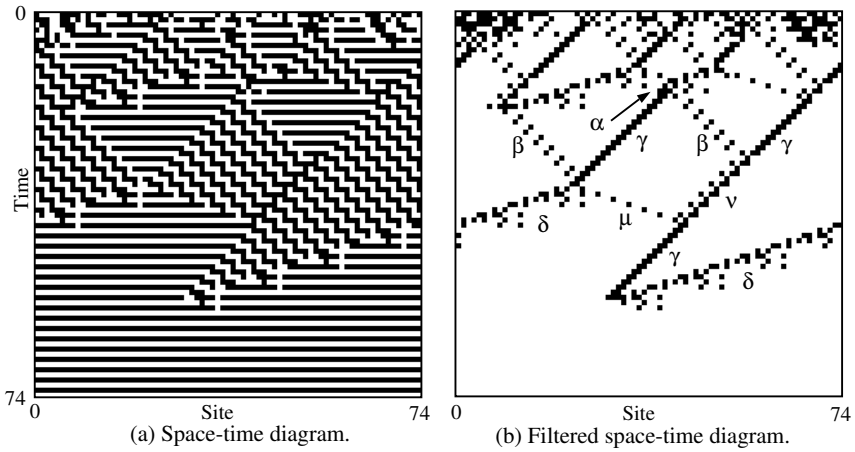


FIGURE 1.3 – Apparitions de signaux à partir d'algorithmes évolutionnistes sur les AC [Das *et al.* 1995, Fig. 1].

Prouver l'universalité. On sait depuis les travaux de von Neumann que les AC ont la capacité de calculer au sens de Turing. Néanmoins, la grande taille de l'AC de von Neumann le rendait difficilement manipulable pour calculer ou pour servir à d'autres constructions. Une quête de l'automate universel minimal s'est donc lancée très tôt, dans laquelle les signaux ont été et sont encore d'une grande importance. Ils ont permis des simulations par AC de modèles de calcul universels soit en concevant des automates simples [Lindgren et Nordahl 1990] comme celui présenté en FIG. 1.4(a) ; soit en tirant le meilleur profit des signaux naturellement engendrés par des ACE (l'ACE 110 en l'occurence) [Cook 2004] et prouvant par conséquent que les AC peuvent être Turing-universel avec seulement deux états et un voisinage minimal.

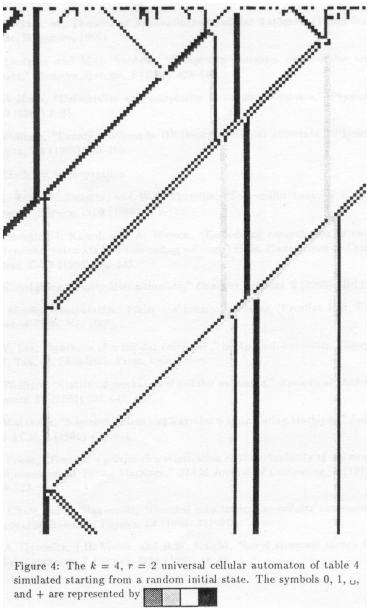
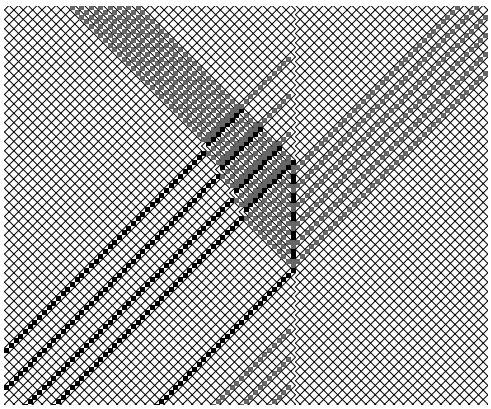


Figure 4: The $k = 4$, $r = 2$ universal cellular automaton of table 4 simulated starting from a random initial state. The symbols 0, 1, ␣, and + are represented by

(a) AC universel de rayon 2
[Lindgren et Nordahl 1990, Fig. 4].



(b) AC intrinsèquement universel de rayon 1
[Ollinger et Richard 2011, Fig. 3].

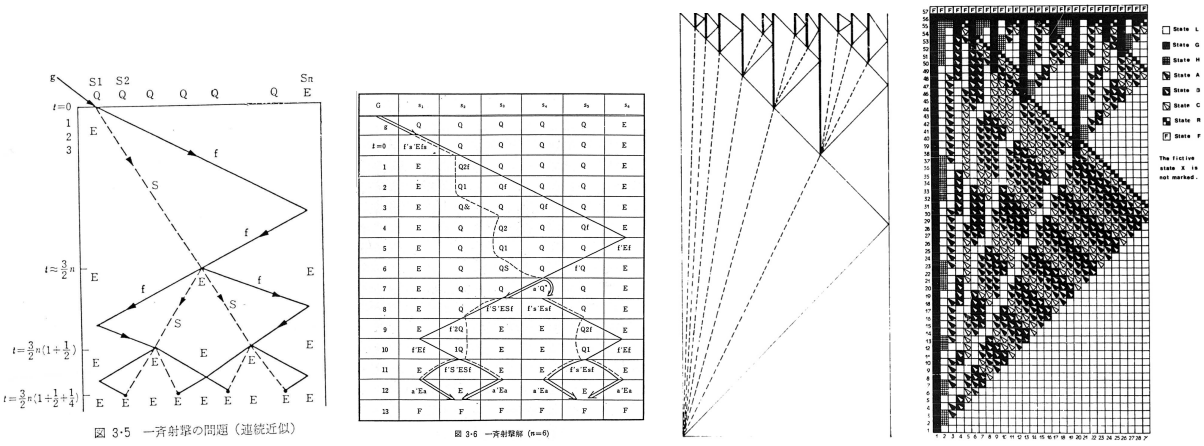
FIGURE 1.4 – Construction par signaux d'AC (intrinsèquement) universels.

Un autre problème faisant l'objet de recherches actives et impliquant la notion d'universalité est le problème de l'universalité intrinsèque, ou la recherche d'AC pouvant simuler tout autre AC. Dans cette quête aussi, les signaux et les collisions ont été d'une grande importance, fournissant entre autres un AC réversible intrinsèquement universel pour les AC réversibles [Durand-Lose 1997] et un AC à peu d'états (quatre) intrinsèquement universel pour les AC de dimension un [Ollinger et Richard 2011] : la Fig. 1.4(b) en donne un exemple.

Concevoir des automates complexes. Nous rappelons ici le problème du fusilier, ou « Firing Squad Synchronization Problem (FSSP) », afin d'illustrer la richesse de l'approche par signaux continus dans la conception d'AC complexes. Le FSSP est le problème consistant à synchroniser une ligne finie de cellules (c'est-à-dire un AC fini de dimension un). Mise à part une des cellules extrémales (la cellule « général »), toutes les autres cellules (les cellules « soldat ») se trouvent initialement dans un même état. Le but est de faire entrer simultanément toutes les cellules pour la première fois dans un état donné (l'état « feu »).

Afin de construire de tels automates synchronisant n'importe quelle ligne de cellules (c'est-à-dire synchronisant une ligne de n cellules pour tout n), de nombreuses méthodes ont été envisagées [Goto 1966 ; Mazoyer 1987 ; Yunès 1994, 2007]. Quasiment toutes sont basées sur l'utilisation de signaux continus : le schéma global du fonctionnement est d'abord conçu avec des signaux continus, puis ce schéma est discrétisé afin d'obtenir l'AC à proprement parler. La Figure 1.5 illustre de telles méthodes, notamment les solutions respectives de Eiichi Goto et Jacques Mazoyer, qui font clairement apparaître le rôle des signaux continus dans leur conception.

Nous insistons en particulier sur la Fig. 1.5(a), la version continue de la solution de Goto, dont la méthode consiste en un *procédé dichotomique* : l'utilisation de trois signaux de pentes différentes permet de diviser la ligne en son milieu, et quatre signaux permettent d'itérer ce processus de manière symétrique. Ce processus initialement appliqué à une ligne de cellules se répète jusqu'à atteindre des divisions de même taille que celle des cellules, qui deviennent alors synchronisées. Alors que dans la construction de Goto, cette méthode est discrétisée pour en faire un AC, nous étudierons dans ce manuscrit cette structure d'*arbre* dans le cadre des signaux continus sur l'espace réel.



(a) La solution continue de Goto et sa version discrétisée [Goto 1966, Fig. 3].

(b) La solution continue de Mazoyer et son implémentation [Mazoyer 1987, Fig. 3 et 12].

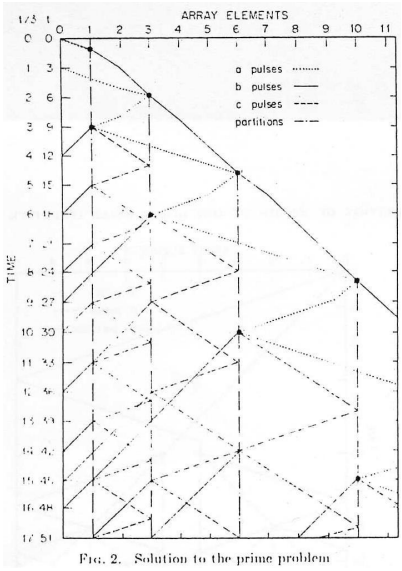
Figure 1.5 – Solutions au problème du fusilier.

Notons qu'en plus d'avoir fourni les premiers automates solutions au FSSP, l'approche par signaux permet de les optimiser et d'obtenir de nouvelles solutions au FSSP et à ses variantes, restant ainsi l'outil canonique pour l'étude de ce genre de problème.

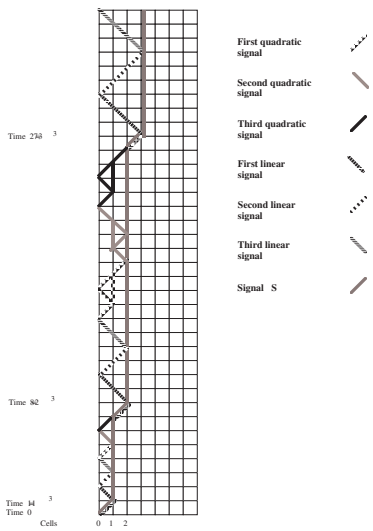
Implémenter des calculs. Il est également possible de réaliser à l'aide des signaux des calculs non triviaux sur des AC. Un des exemples les plus connus est l'automate de Patrick Fischer qui génère les nombres premiers [Fischer 1965]. Cet AC est conçu uniquement en termes de signaux, et ceux-ci sont utilisés pour implémenter une version géométrique du crible d'Érathostène. La FIGURE 1.6(a) donne le principe de fonctionnement de cet automate : les signaux verticaux codent (par leur espacement) des entiers (tous les entiers à partir de 2) dont on veut marquer les multiples stricts ; des signaux sont utilisés pour générer ces entiers ; et enfin des signaux permettent d'atteindre la cellule initiale, laquelle reçoit un signal au temps $3t$ lorsque t est un nombre composé. On peut alors se ramener à un AC tel que la première cellule est dans un certain état au temps t si et seulement si t est un nombre premier.

De même, Jacques Mazoyer, Véronique Terrier et Marianne Delorme ont explicité des AC permettant de construire des classes de fonctions [Mazoyer et Terrier 1999 ; Delorme et Mazoyer 2002]. Ils ont distingué plusieurs types de fonctions et méthodes de constructions, toutes basées sur l'utilisation de signaux. Ceux-ci permettent d'engendrer d'une part des fonctions « Fischer-constructibles » (la première cellule est dans un état donné pour la n-ième fois au temps t si et seulement si $f(n) = t$), et d'autre part des fonctions dont la pente (discrète) globale est directement représentée dans le diagramme espace-temps. Une large classe de fonctions peut ainsi être construite : les fonctions puissances entières (voir la FIG. 1.6(b) pour le cas de n^3), exponentielles, la fonction factorielle...

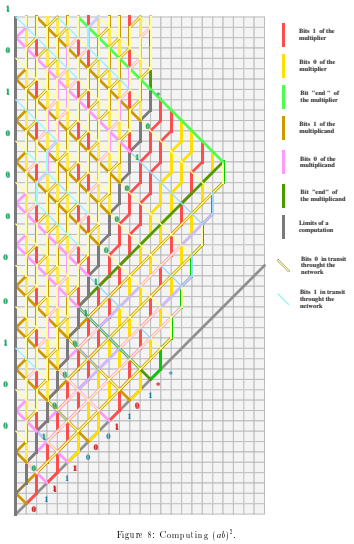
Il existe également des méthodes pour construire des grilles de calcul plus générales [Mazoyer 1996 ; Mazoyer et Yunès 2012 ; Yunès 2012], qui peuvent ensuite être utilisées pour y implémenter des calculs comme l'exemple de la multiplication donnée en FIG. 1.6(c).



(a) Générer les nombres premiers [Fischer 1965, Fig. 2].



(b) Calculer la fonction n^3 [Mazoyer et Terrier 1999, Fig. 5].



(c) Calculer $(a \cdot b)^2$ [Mazoyer 1996, Fig. 8].

FIGURE 1.6 – Exemples d'implémentations de calcul par signaux en AC de dimension 1.

Formalisations des particules et collisions discrètes. À notre connaissance, l'une des formalisations les plus abouties concernant les signaux et les collisions discrètes a été fournie dans [Richard 2008 ; Ollinger et Richard 2009]. Les diagrammes espace-temps y sont formellement décomposés en fond, motif et particules, puis formalisés avec des schémas de ligature, dont l'expressivité peut être réduite à la résolution de formules existentielles dans l'arithmétique de Presbuger. Cette approche a permis d'obtenir des résultats sur la conception et la minimisation

d'AC universels, notamment avec la conception d'un AC intrinsèquement universel à quatre états et voisinage minimal [Ollinger et Richard 2011].

Une autre approche consiste en *systèmes abstraits de collisions discrètes* [Ito *et al.* 2008]. Un système abstrait de collisions est défini par un ensemble de « balles » (équivalent aux états d'un AC), et par une une famille de collisions entre ces balles. Une fonction de collision associant une configuration sortante à toute collision appartenant à la famille, complète la définition d'un système abstrait de collisions. Ces systèmes formalisent de manière simpliste les collisions discrètes et ne permettent pas de retrouver toute la richesse des signaux discrets utilisés par les AC, notamment parce qu'un signal y est modélisé par un unique état (une balle). De plus, la définition des collisions prenant en compte les positions spatiales de la collision, la famille de collisions est définie par des schémas de collisions et peut donc être infinie. Néanmoins, ce modèle permet facilement de simuler d'autres modèles de calcul tels que les systèmes de réécriture cycliques (cyclic tag systems) et le modèle boule de billard (billard ball model).

Signaux discrets et signaux continus. Bien qu'utilisés pour concevoir des AC, par nature discrets, les signaux continus diffèrent de leur version discrète sur plusieurs points. En effet, les signaux continus en sont une version idéalisée : ils n'ont ni épaisseur ni période, contrairement aux signaux discrets qui en ont (à la fois temporellement et spatialement). Du fait de leur épaisseur, le décalage spatial (ou déphasage) entre deux signaux discrets a une importance dans le résultat de la collision discrète entre ces deux signaux. La FIGURE 1.7 illustre les six sorties possibles pour une collision impliquant à chaque fois les deux mêmes particules : les résultats possibles proviennent des décalages possibles entre ces deux particules au moment de la collision.

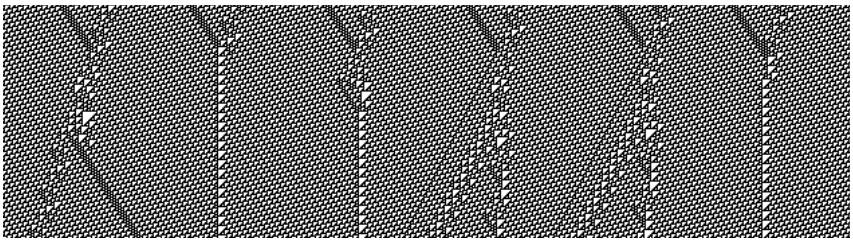


FIGURE 1.7 – Résultats possibles pour une même collision dans l'ACE 110 [Cook 2004, Fig. 6].

1.3 Les machines à signaux en tant que modèle de calcul

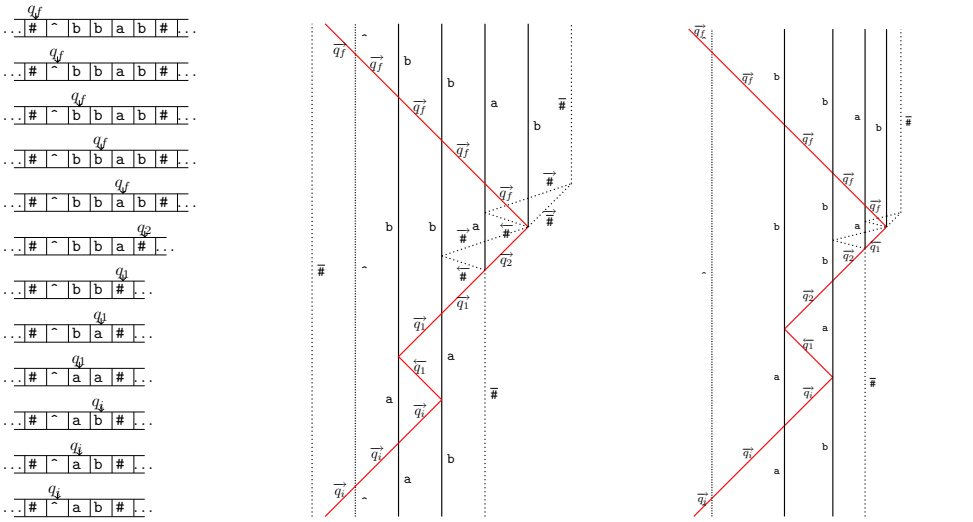
Nous donnons ici plusieurs propriétés sur les machines à signaux d'un point de vue général, qui peuvent être comprises sans les définitions formelles concernant le modèle, lesquelles seront énoncées dans le CHAP. 2. Nous discuterons également des similitudes et distinctions qui existent entre les modèles à signaux et les modèles géométriques proches dont nous avons connaissance.

1.3.1 Propriétés des machines à signaux

Nous énonçons ici des caractéristiques connues des machines à signaux qui concernant leur pouvoir de calcul. Les quelques diagrammes espace-temps qui seront donnés pour illustrer la simulation de modèles de calcul par des machines à signaux peuvent être compris de manière intuitive, sans recourir à leurs définitions formelles.

Turing-univeralité. L'une des toutes premières propriétés qui a été démontrée concernant les machines à signaux est leur *Turing-universalité* [Durand-Lose 2003]. La capacité des machines à signaux à calculer au sens de Church-Turing peut être démontrée par des simulations

directes de différents modèles de calcul qui sont équivalents aux machines de Turing : automates à deux compteurs, automates celllulaires, cyclic tag systems (systèmes cycliques de réécriture) et machines de Turing (on trouvera dans [Durand-Lose 2011a] les simulations de plusieurs de ces modèles). La FIGURE 1.8 illustre des simulations directes d'une machine de Turing : une simulation avec des cellules de même taille est donnée FIG. 1.8(b) et une simulation en espace (continu) constant en FIG. 1.8(c) (la FIG. 1.8(a) donnant l'exécution correspondante sur machine de Turing classique).



(a) Une exécution de MT. (b) Simulation par signaux. (c) Simulation en espace borné.

FIGURE 1.8 – Simulations d'une machine de Turing [Durand-Lose 2011a, Fig. 4].

Les simulations de machines de Turing par des signaux peuvent se faire en utilisant uniquement des machines avec des signaux initiaux à positions rationnelles et des collisions de coordonnées rationnelles : de telles machines sont appelées *machines rationnelles*. De plus, les diagrammes correspondant aux simulations ne contiennent pas d'accumulation. La simulation réciproque est possible : la possibilité de simuler des diagrammes de machines à signaux rationnelles sans accumulation par une machine de Turing découle de l'existence d'un programme capable de générer de tels diagrammes [1].

De la Turing-universalité des machines à signaux découlent beaucoup de propriétés d'indécidabilité sur leur dynamique, telles que les problèmes de finitude d'un diagramme espace-temps, d'apparition d'un signal donné, d'apparition d'une accumulation... [Durand-Lose 2003].

Calculs analogiques. Les machines à signaux étant définies à partir d'espace et temps continus (respectivement $\mathbb{R}$ et $\mathbb{R}^+$), il semble naturel de penser qu'elles puissent effectuer des calculs analogiques. Il a effectivement été montré que certains modèles de calcul manipulant des valeurs réelles sont simulables par machines à signaux. Parmi les modèles évoqués en SOUS-SEC. 1.1.2, l'analyse récursive et le modèle BSS ont été implémentés avec des signaux, respectivement dans [Durand-Lose 2011b] et [Durand-Lose 2008]. Ces deux simulations prouvent que les machines à signaux sont au moins aussi puissantes que ces deux modèles. Elles nécessitent cependant l'utilisation séquentielle d'accumulations afin d'opérer une infinité d'étapes de calcul en un temps fini. En effet, les machines à signaux sans accumulation ne sont pas aussi puissantes que le modèle BSS complet puisqu'elles ont été montrées équivalentes au modèle BSS linéaire [Durand-Lose 2007], une version plus faible des machines BSS. Elles deviennent équivalentes au modèle BSS

1. Un tel logiciel a été implémenté en `java` par Jérôme Durand-Lose et a servi à générer la plupart des diagrammes présents dans ce manuscrit.

complet lorsque l'utilisation d'accumulations est autorisée.

Capacités super-Turing. Parmi les propriétés non-conventionnelles des machines des signaux, la possibilité de résoudre des problèmes récursivement énumérables en fait un modèle super-Turing, et de ce fait un modèle puissant. Cette aptitude à résoudre des problèmes indécidables par machines de Turing a été prouvée en implémentant par signaux et signaux rationnels le modèle « trou noir » [Durand-Lose 2004, 2006a], qui est également le phénomène de type Zénon utilisé par les machines accélérantes et les modèles relativistes. Ce modèle consiste à exécuter une infinité d'opérations en un temps fini, rendant décidable le problème de l'arrêt d'une machine de Turing. Pour être implémentée sur des machines à signaux, cette accélération s'inspirant du paradoxe de Zénon requiert la continuité de l'espace et nécessite l'utilisation d'une accumulation, dans laquelle est insérée la simulation d'une machine de Turing. Dans cette simulation, les accumulations sont utilisées de manière séquentielle : une seule accumulation peut apparaître dans le diagramme, et uniquement à la limite de (l'éventuelle) infinité d'opérations de la machines de Turing.

Parallélisme massif. Comme les automates cellulaires, les machines à signaux bénéficient également d'un parallélisme massif intrinsèque [Duchier *et al.* 2010b]. Les AC étant simulables avec des signaux, comme illustré par FIG. 1.9, la puissance de calcul parallèle des machines à signaux égale au moins celle des AC. Une utilisation de ce parallélisme massif constitue la problématique de cette thèse et sera illustrée dans les chapitres CHAP. 7, CHAP. 8 et CHAP. 9.

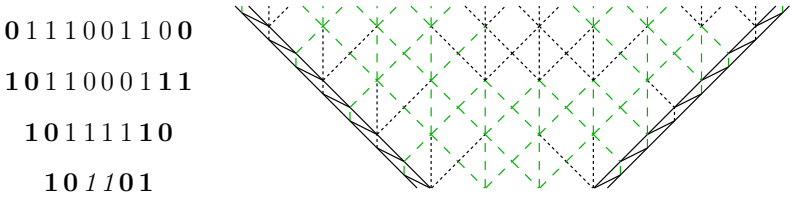


FIGURE 1.9 – Évolution (bornée par les motifs respectifs $^{\omega}(10)$ à gauche et $(011)^{\omega}$ à droite) et simulation de l'ACE 110 [Durand-Lose 2011a, Fig. 7].

1.3.2 Positionnement des machines à signaux

Nous comparons ici les machines à signaux avec les approches similaires dont nous avons connaissance, et nous clorons le chapitre en initiant une discussion sur les différents aspects physiques du modèle.

Approches similaires. Nous reprenons ici quelques uns des modèles géométriques décrits dans la SOUS-SEC. 1.1.2 et dont les contextes sont proches de celui des machines à signaux. Les machines euclidiennes abstraites, les pavages, les systèmes d'auto-assemblage et les AC ne sont pas abordés ici. Nous pensons que par définition, leurs caractéristiques en font d'emblée des modèles de natures géométriques très différentes de celle des machines à signaux. La notion de calcul (purement séquentiel) des machines euclidiennes ou le type d'espace-temps et de dynamique considérés (discret et/ou sans dynamique uniforme) dans le cas des pavages, des systèmes auto-assemblants et des AC constituent des points distinctifs fondamentaux pour le type d'étude considérée ici.

Parmi les autres contextes de calcul géométrique déjà évoqués, le modèle des univers colorés de [Jacopini et Sontacchi 1990] semblent être le modèle qui se rapproche le plus des machines à signaux. Les deux sont définis sur des univers continus de même nature (des espaces euclidiens), et leurs études sont essentiellement restreintes au cas d'objets rationnels.

Nous notons cependant des différences. D'une part, les motivations ne semblent pas être tout à fait les mêmes : elles sont dans le cas des machines à signaux l'étude abstraite de simples objets de base — les signaux — alors que l'étude des univers colorés est motivée par des considérations de type physique comme la réversibilité. Les structures polyédrales considérées dans [Jacopini et Sontacchi 1990] résultent d'ailleurs d'hypothèses sur la nature de l'espace (vitesse bornée, isotropie...), et les signaux et les collisions constatés en sont une conséquence, et non les objets élémentaires qui fondent l'étude. D'autre part, les collisions considérées dans les univers colorés sont des boules et ont donc des dimensions non nulles, néanmoins non bornées inférieurement. L'une des différences majeures reste le traitement des accumulations : elles sont explicitement exclues des univers colorés, et ni elles et encore moins les fractales ne sont considérées dans les constructions.

Concernant les systèmes à dérivées constantes par morceaux (systèmes PCD), les principales différences avec les machines à signaux sont à nos yeux : la non-isotropie de l'espace (due au partionnement de l'espace en régions polyédrales à vecteurs constants), la séquentialité du calcul et le recours à plusieurs dimensions spatiales pour obtenir des propriétés intéressantes. Trois dimensions spatiales sont en effet nécessaires aux systèmes PCD pour être Turing-universel, alors qu'une seule suffit aux machines à signaux. La puissance de calcul de ces systèmes a déjà été caractérisée, en fonction de la dimension des systèmes considérés. De plus, les systèmes PCD à paramètres réels ont été démontrés strictement plus puissants que ceux à paramètres rationnels. Une telle comparaison entre cas réel et cas rationnel n'est pas encore totalement explicitée dans le cadre des machines à signaux, dont les propriétés et pouvoir de calcul ont surtout été étudiés dans le cas rationnel et uniquement en dimension un.

Contrairement aux machines à signaux, les systèmes de transitions sur des espaces-temps continus ne constituent pas une approche géométrique mais proposent un formalisme basé sur des règles d'inférences et la logique modale. De plus, les notions de base du modèle sont la transition d'état d'un point et la diffusion d'information à travers l'espace. Cette notion de propagation d'information est proche de la notion de signal mais ne semble toutefois pas y être équivalente : elle permet l'apparition de « membranes » lors de la propagation d'un état à travers un autre, comme illustré par la Fig. 1.1(c). S'il est *a priori* possible de définir certaines classes de machines à signaux dans le formalisme des systèmes de transition à temps et espace continus, il semblerait que certaines machines (notamment les machines contenant des règles de collisions produisant trois méta-signaux ou plus) ne soient pas formalisables dans ces systèmes de transition. De plus, l'interprétation en termes de machine de calcul y est beaucoup moins intuitive, et les travaux réalisés dans [Takeuti 2005 ; Hagiya 2005] ne traitent que de la réalisabilité de la trajectoire d'un point (*i.e.* les états successivement pris par un point de l'espace) et non de dynamiques ou d'aspects globaux.

Enfin, le paradoxe de Zénon est plus délicat à gérer dans ce contexte, la suite d'états pris par un point de l'espace pouvant être *a priori* infini dans les deux sens. Ce type de phénomène est explicitement exclu par le système de règles d'inférences utilisé pour décrire les trajectoires réalisables.

Considérations physiques sur les machines à signaux. Les travaux concernant les machines à signaux ont été et sont encore surtout motivés par les aspects théoriques du modèle, dont l'étude apporte un point de vue différent sur les notions classiques liées au calcul. Néanmoins, nous pouvons reformuler leurs caractéristiques essentielles et énoncer les postulats physiques qui sous-tendent ce modèle de calcul purement abstrait.

Si certains types de signaux et collisions ont déjà été utilisés de multiples façons pour modéliser des phénomènes physiques [Adamatzky 2002], le modèle des machines à signaux n'a pour le moment aucune application en termes de modélisation. Cela tient sans doute aux définitions

idéalisées des objets considérés : espace et temps continus, particules sans dimensions se déplaçant sur espace à une seule dimension et à vitesse constante, collisions ponctuelles. Ces définitions constituent des hypothèses physiques fortes, qui rendent l'implémentation physique du modèle peu probable, et sur laquelle nous n'avons *a priori* aucune perspective.

Les machines à signaux n'en respectent pas moins les postulats de base de certaines théories physiques majeures comme celle de la relativité. En effet, la vitesse de propagation de l'information y est bornée (le principe de causalité est donc respecté), la densité d'information reste bornée (dans le cas de diagrammes sans accumulation) et l'espace-temps considéré est homogène (en l'absence d'objets), l'espace étant de plus isotrope. Ces caractéristiques sont à recouper avec les principes formulés par Robin Gandy sur la réalité physique d'une machine de calcul : nature discrète, vitesse d'information bornée, densité finie d'information, homogénéité de l'espace et du temps, état quiescent presque partout (*i.e.* partout sauf dans une partie finie de l'univers). Il a été démontré dans [Gandy 1980] qu'il est impossible pour une machine satisfaisant ces principes de calculer strictement plus qu'une machine de Turing. Mais les machines à signaux ne respectent pas la première hypothèse sur la nature discrète d'une machine physique de calcul. De plus, le modèle est connu pour être super-Turing lorsque les accumulations sont considérées, et la présence d'accumulation viole l'un des principes de Gandy : celui de la densité finie d'information.

On notera que certaines classes de machines à signaux ont été étudiées avec des considérations motivées par la physique [Durand-Lose 2006b, 2012a] : c'est respectivement le cas des machines réversibles (possibilité de connaître la totalité du diagramme uniquement à partir d'une configuration à un instant donné) et des machines conservatives (conservation d'une énergie à chaque collision). Ces cas particuliers de machines respectent tous deux le principe de conservation de l'énergie, à la base de la physique des particules et de la thermodynamique.

Chapitre 2

Machines à signaux

Nous présentons dans ce chapitre le modèle formel des machines à signaux, fournissant ainsi un cadre rigoureux à des figures géométriques comme la Fig. 1 de l'introduction, construites à partir d'énoncés du type « quand un segment bleu de pente 1 et un segment vert de pente 1/3 s'intersectent, un nouveau segment rouge de pente 2 les remplace ». Pour en donner une interprétation en termes de machines, nous formalisons les segments de droites colorées sous forme d'objets abstraits (*les méta-signaux*) et nous décrivons leurs intersections par des règles (les *règles de collisions*). À partir de ces objets de base et de leurs instanciations en *signaux* et *collisions*, il devient possible d'étudier le calcul géométrique abstrait sous divers angles, que nous utilisons ici pour présenter les machines à signaux sous quatre aspects différents : machine, topologique, dynamique et géométrique. La multiplicité de ces points de vue provient du fait que l'on considère ici une dynamique de calcul sur un espace-temps euclidien ($\mathbb{R} \times \mathbb{R}^+$), qui est naturellement muni d'une topologie (en tant qu'espace métrique) et d'une structure géométrique (en tant qu'espace affine).

Nous partirons des définitions de base déjà existantes et énoncées dans [Durand-Lose 2003], que nous rappellerons dans les deux premières sections : après avoir donné la définition de machines à signaux à partir des primitives du modèle, nous formulerons de manière topologique les diagrammes espace-temps engendrés par ces machines. Cette approche topologique des diagrammes fournit un cadre formel pour la définition et l'étude des accumulations. Nous associerons ensuite une dynamique aux diagrammes espace-temps, dont nous décrirons l'évolution en considérant la suite des temps de collisions. Nous finirons le chapitre par un point de vue géométrique, en énonçant certaines propriétés d'invariance par transformations affines et en introduisant les notions de *supports* de machines et de diagrammes. Ces derniers concepts, dédiés à l'étude de la structure globale des diagrammes espace-temps, seront ensuite utilisés pour démontrer certaines propriétés des accumulations et fractales dans le Chap. 4.

2.1 Point de vue machine

Nous définissons ici les machines à signaux de la manière qui semble la plus intuitive : une machine à signaux sera constituée par un ensemble de *méta-signaux* et de *règles de collisions* définissant ce qui se passe lorsque plusieurs de ces méta-signaux se rencontrent. Nous commençons ici par détailler les primitives de calcul du modèle, qui nous serviront dans un second temps à définir les machines à signaux sous forme d'un triplet.

Nous ne présentons ici que des machines à une seule dimension spatiale (les signaux se propagent sur la droite réelle $\mathbb{R}$), l'objectif principal de cette thèse étant d'utiliser des fractales à des fins de parallélisme massif dans le plan (la seconde dimension correspondant alors au temps).

Cependant, la définition peut facilement se généraliser à plusieurs dimensions [1] .

2.1.1 Les objets et notions de base du modèle

Comme mentionné dans SOUS-SEC. 1.1.2, les primitives d'un modèle de calcul définissent les objets de base et les opérations élémentaires autorisées, et conditionnent la forme que prend le calcul dans ce modèle. Dans le cas des machines à signaux, les primitives de calcul sont de nature géométrique : ce sont des segments colorés et leurs intersections. Nous les présentons ici d'une manière plus abstraite, afin de les utiliser pour définir formellement les machines à signaux.

Les objets élémentaires du calcul par signaux sont de deux types : d'une part les *signaux* et d'autre part les *collisions*. Les signaux et les collisions sont des objets « concrets », dans le sens où ce sont ceux qui sont utilisés lors d'une exécution de machine à signaux et dans le diagramme espace-temps résultant. Les signaux et collisions sont en fait des instances d'objets abstraits, respectivement les *méta-signaux* et les *règles de collisions*, dont ils héritent des caractéristiques.

Méta-signaux et signaux. Les méta-signaux et les signaux constituent les objets fondamentaux des machines à signaux. La relation entre ces deux types d'objets est de nature *abstraction/instanciation* : les méta-signaux sont les éléments abstraits qui définissent les types de signaux qu'une machine peut utiliser. On peut les comparer aux états utilisés par des machines classiques, comme une machine de Turing. Un signal est une version instanciée d'un méta-signal qui prend place dans l'espace-temps : un signal a donc une date de création et une date de fin (il peut également se propager indéfiniment). Il peut y avoir plusieurs signaux de même type existant simultanément dans une machine à signaux. Comme dans le cas des AC, les signaux peuvent également être vus comme étant la trace dans l'espace-temps de *particules* se déplaçant dans l'espace. Leur vitesse étant constante, les traces résultantes de leurs mouvements sont des segments de droite, signaux et particules constituent donc deux formalisations différentes de la même notion (celle d'information), mais les signaux sont des objets unidimensionnels tandis que les particules sont des points sans dimension. Nous utiliserons dans ce manuscrit exclusivement le terme de signal. Un méta-signal sera identifié par son nom et on parlera indifféremment de signal et de méta-signal : par abus de langage, on dira « un signal μ » au lieu de dire « un signal de type μ ».

Chaque méta-signal est muni d'une vitesse à valeur réelle, donnant la distance qu'il parcourt en une unité de temps. La dynamique de propagation d'un signal (tant qu'il ne rencontre pas d'autre signal) se calcule alors simplement à l'aide de sa vitesse. Si un signal de vitesse ν est présent en position x_0 au temps t_0, sa position x au temps $t \geq t_0$ sera donnée par :

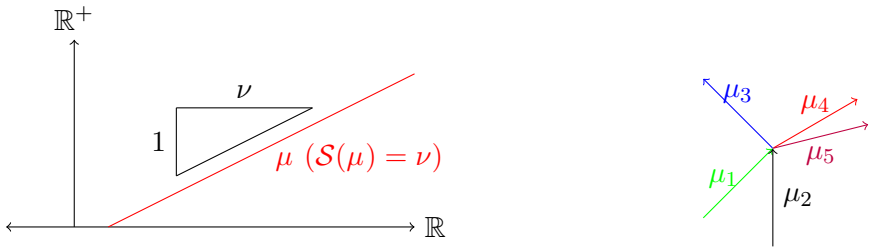
$$x = x_0 + \nu \cdot (t - t_0) \tag{2.1}$$

Vitesse d'un méta-signal. On remarquera que, comme l'espace et le temps sont respectivement représentés horizontalement et verticalement, la vitesse d'un signal ne correspond pas à sa pente dans un repère orthonormé classique. En effet, comme cela est représenté par la FIG. 2.1(a), la vitesse définit la distance parcourue par un signal en une unité de temps et la pente du segment correspondant est donnée par l'inverse de la vitesse : un signal de vitesse 2 est représenté par un segment de pente 1/2. Deux signaux de même vitesse sont donc parallèles car étant de même pente. Nous ne raisonnerons qu'en termes de vitesse, cette notion étant ici plus intuitive que celle de pente. En effet, l'espace dans lequel se propagent les signaux étant une droite, nous pouvons facilement obtenir la nouvelle position d'un signal partant d'une position donnée grâce à l'EQ. (2.1).

1. Nous donnerons d'ailleurs un bref exemple de machine à signaux à deux dimensions spatiales dans le CHAP. 5.

Plus un signal est rapide, plus sa pente est faible. Un signal de grande vitesse est ainsi représenté par un segment de droite proche de l'horizontale. Notons qu'il est impossible d'avoir un signal horizontal : cela signifierait en effet que sa vitesse est infinie, ce que nous excluons explicitement du modèle. Un signal peut toutefois être vertical : il correspond alors à une vitesse nulle et est *stationnaire*.

Nous considérerons par la suite des ensembles de méta-signaux, et leurs vitesses seront données par une fonction $\mathcal{S}$ — la *fonction vitesse* — qui associera à chaque méta-signal la vitesse réelle correspondante : on aura $\mathcal{S}(\mu) = \nu$ si ν est la vitesse du méta-signal μ.



(a) Propagation d'un signal de type μ et de vitesse ν. (b) Règle de collisions $\mathcal{R}(\{\mu_1, \mu_2\}) = \{\mu_3, \mu_4, \mu_5\}$.

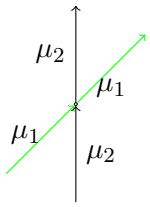
FIGURE 2.1 – Les objets de base du modèle.

Règles de collisions. Les règles de collisions définissent ce qu'il advient lorsque plusieurs signaux interagissent, c'est-à-dire qu'elles décrivent le comportement adopté aux intersections de signaux. Un ensemble de règles de collisions peut être vu comme l'équivalent d'une table (d'un ensemble, d'une fonction...) de transitions d'une machine classique du type machine de Turing. L'ensemble de règles décrivant les interactions d'un ensemble de méta-signaux est ici donné sous la forme d'une fonction $\mathcal{R}$ — la *fonction de collision* — qui associe à tout sous-ensemble de méta-signaux un ensemble de méta-signaux : une règle s'écrit donc $\mathcal{R}(\{\mu_1, \dots \mu_n\}) = (\{\mu'_1, \dots \mu'_m\})$. L'ensemble antécédent $\{\mu_1, \dots \mu_n\}$ est appelé ensemble de *méta-signaux entrants* et l'ensemble image $\{\mu'_1, \dots \mu'_m\}$ est l'ensemble des *méta-signaux sortants* associé à cette collision. La fonction $\mathcal{R}$ est une fonction partielle : il existe en effet des contraintes sur les ensembles entrants et sortants. Il faut au moins deux méta-signaux pour provoquer une collision et des signaux de même vitesse ne peuvent être en collision (car ayant des trajectoires parallèles). De même, on impose que deux signaux sortant d'une même collision aient des vitesses différentes (sinon, ils rentreraient immédiatement en collision, pouvant éventuellement provoquer ainsi une chaîne infinie de collision s'enchaînant de manière continue). La FIG. 2.1(b) fournit la représentation graphique d'une règle de collisions : les signaux entrants arrivent par le bas, les sortants partent vers le haut (les segments représentant les signaux sont également munis de flèches indiquant leur direction).

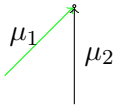
De la même façon qu'un signal est une instance concrète d'un méta-signal, une collision est un objet qui prend place dans l'espace-temps et qui est entièrement caractérisée par la règle de collisions correspondante. Sur un point de collision, tous les signaux rentrants sont immédiatement remplacés par les signaux sortants, et cela en accord avec les règles de collisions. Dans le contexte des machines à signaux, les caractéristiques importantes des collisions sont leur aspect déterministe (des collisions avec les mêmes signaux en entrée produiront les mêmes signaux en sortie) et leur aspect ponctuel (une collision est un point sans dimension).

On distingue certaines formes particulières de règles de collisions qui correspondent à des fonctionnalités souhaitées pour certains signaux (voir FIG. 2.2). Nous en énumérons ici quelques unes. Une règle dont les signaux entrants et sortants sont exactement les mêmes sera qualifiée de *règle blanche* : les signaux se croisent sans aucune interaction (voir FIG. 2.2(a)). Formellement, une règle blanche s'écrit $\mathcal{R}(\{\mu_1, \dots \mu_n\}) = (\{\mu_1, \dots \mu_n\})$. Cette forme est considérée ici comme

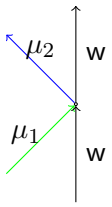
la forme par défaut : toutes les règles de collisions non spécifiées d'une machine seront des règles blanches. On dira d'une règle non blanche qu'elle est une *règle significative*. On appellera *règle d'annihilation* une règle du type $\mathcal{R}(\{\mu_1, \dots \mu_n\}) = \emptyset$, c'est-à-dire telle que l'ensemble des signaux sortants est vide (voir FIG. 2.2(b)). On appellera *règle de rebond* (ou plus simplement *rebond*) toute règle $\mathcal{R}(\{\mu, \mathsf{w}\}) = \{\mu', \mathsf{w}\}$, avec $\mathcal{S}(\mu_2) < \mathcal{S}(\mathsf{w}) < \mathcal{S}(\mu_1)$ ou $\mathcal{S}(\mu_1) < \mathcal{S}(\mathsf{w}) < \mathcal{S}(\mu_2)$(voir FIG. 2.2(c)). Un signal w (pour wall) tel qu'aucun signal ne peut le traverser par définition des règles de collisions, sera qualifié de *signal mur* (un signal arrivant sur mur disparaît ou rebondit). Un signal c (pour clean) est dit *signal de nettoyage* si toutes les règles de collisions (sauf éventuellement une pour stopper le processus) dans lesquelles il apparaît en entrée sont de la forme $\mathcal{R}(\{\mathsf{c}, \mu_1, \dots \mu_n\}) = \{\mathsf{c}\}$: le seul signal sortant est c (voir FIG. 2.2(d)).



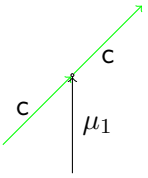
(a) Règle blanche $\mathcal{R}(\{\mu_1, \mu_2\}) = \{\mu_2, \mu_1\}$.



(b) Règle d'annihilation $\mathcal{R}(\{\mu_1, \mu_2\}) = \emptyset$.



(c) Règle de rebond $\mathcal{R}(\{\mu_1, \mathsf{w}\}) = \{\mu_2, \mathsf{w}\}$.



(d) Règle de nettoyage $\mathcal{R}(\{\mathsf{c}, \mu_1\}) = \{\mathsf{c}\}$.

FIGURE 2.2 – Quelques formes utiles de règles de collisions.

2.1.2 Machines à signaux

Nous pouvons maintenant énoncer la définition d'une machine à signaux. Une machine à signaux sera composée d'un ensemble fini de méta-signaux et de deux fonctions définies à partir de cet ensemble : l'une fournissant la vitesse de chaque méta-signal et l'autre décrivant les interactions entre méta-signaux. Formellement :

DÉFINITION 1 (Machine à signaux)
Une machine à signaux $\mathfrak{M}$ *est un triplet* $\mathfrak{M} = (\mathcal{M}, \mathcal{S}, \mathcal{R})$ *tel que :*

(i) $\mathcal{M}$ *est un ensemble fini de méta-signaux ;*

(ii) $\mathcal{S} : \mathcal{M} \to \mathbb{R}$ *est la fonction vitesse qui associe à chaque méta-signal* μ *une vitesse* $\mathcal{S}(\mu)$ *et*

(iii) $\mathcal{R} : \mathcal{P}(\mathcal{M}) \to \mathcal{P}(\mathcal{M})$ *est la fonction (partielle) de collision qui associe à chaque ensemble de méta-signaux* $\rho^- \in \mathcal{P}(\mathcal{M})$ *satisfaisant* $card(\rho^-) \geq 2$ *et* $\mathcal{S}_{\restriction \rho^-}$ *est injective, un ensemble de méta-signaux* ρ^+ *tel que* $\mathcal{S}_{\restriction \rho^+}$ *est injective.*

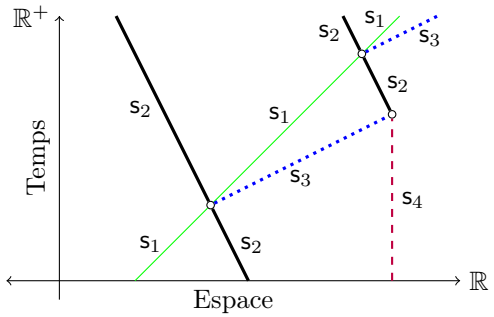
Les conditions sur la définition de $\mathcal{R}$ définissent les contraintes sur les collisions que nous avons déjà évoquées : d'une part, des signaux peuvent rentrer en collision seulement si leurs vitesses sont différentes, et une collision implique au moins deux signaux ; et d'autre part, les signaux résultant d'une collision doivent également avoir des vitesses différentes. Par la suite, nous identifierons la fonction de collision avec son graphe, dont nous identifierons les éléments à des règles de collisions : on notera donc $\rho^- \to \rho^+$ au lieu de $\mathcal{R}(\rho^-) = \rho^+$, afin de garder l'intuition de notion de règle. Nous noterons également parfois $\mathcal{R}^{sign}$ l'ensemble des règles significatives (*i.e.* non-blanches) : $\mathcal{R}^{sign}$ désignera l'ensemble $\{\rho \in \mathcal{R} \mid \rho^- \neq \rho^+\}$. En général, on se contentera de définir $\mathcal{R}^{sign}$ pour définir l'ensemble des règles d'une machine, les autres règles étant considérées comme blanches par défaut.

La FIGURE 2.3 fournit un exemple complet de machine simple : l'ensemble des méta-signaux, ainsi que les valeurs correspondantes de la fonction $\mathcal{S}$ sont donnés en FIG. 2.3(a) ; l'ensemble $\mathcal{R}$

des règles de collisions est donné en FIG. 2.3(c). Une évolution de cette machine est représentée en FIG. 2.3(b).

Méta-signal	Vitesse
s_1	1
s_2	$-1/2$
s_3	3
s_4	0

(a) Méta-signaux.



(b) Diagramme espace-temps.

$\{ s_2, s_1 \} \to \{ s_2, s_1, s_3 \}$

$\{ s_3, s_4 \} \to \{ s_2 \}$

(c) Règles de collisions.

FIGURE 2.3 – Exemple de machine à signaux et évolution à partir d'une configuration initiale.

Configurations et diagrammes. On appellera *configuration* l'« état » du système à un instant donné, c'est-à-dire la liste des objets existant à ce moment et leurs dispositions dans l'espace (la droite réelle). Nous donnerons une telle description sous forme d'une fonction définie de l'ensemble des valeurs valides pour la machine : les méta-signaux (les valeurs des signaux) et les règles de collisions (les valeurs des collisions), auxquelles s'ajoute la valeur $\oslash$ (la *valeur vide*) pour désigner l'absence d'élément. On note $\mathbb{V} = \mathcal{M} \cup \mathcal{R} \cup \{\oslash\}$ l'ensemble des valeurs prises par la machine. Une configuration est donc une fonction définie de l 'espace $\mathbb{R}$ sur $\mathbb{V}$.

Tout signal ou collision dans une configuration doit être spatialement isolé : la seule valeur arbitrairement proche est $\oslash$. Nous ne considérons pas encore de valeur pour les positions qui ont des signaux ou des collisions arbitrairement proches. Ces positions sont des *points d'accumulation* et prendront la valeur $\maltese$. De tels cas seront pris en compte par la formulation topologique dans la SEC. 2.2.

DÉFINITION 2 (Configuration)
On appelle configuration *toute fonction* $c : \mathbb{R} \to \mathbb{V}$ *telle que tous les signaux et collisions sont isolés :* $\forall x \in \mathbb{R},\ c(x) \in \mathcal{M} \cup \mathcal{R} \ \Rightarrow \exists \varepsilon > 0,\ \forall y,\ 0 < |x - y| < \varepsilon \Rightarrow c(y) = \oslash$.
On appelle support de la configuration c *l'ensemble* $support(c) = \{x \in \mathbb{R} \mid c(x) \neq \oslash\}$.

La condition de signaux et collisions isolés implique que le support d'une configuration est au plus dénombrable : il y a au plus un nombre dénombrable de méta-signaux et de collisions sur l'espace $\mathbb{R}$. Nous ne travaillerons par la suite qu'avec des configurations finies c'est-à-dire des configurations à support fini.

On notera c_t pour désigner une configuration au temps $t \in \mathbb{R}$. Le support d'une configuration c, que l'on notera $support(c)$, est l'ensemble des positions de valeur non vide pour cette configuration. Lorsque $c(x) = v$ où $x \in \mathbb{R}$ et $v \in \mathcal{M} \cup \mathcal{R}$, on notera $v@x$ (cette notation implique donc que la valeur en x est non vide).

Nous définissons les configurations initiales comme des configurations munies d'une restriction de finitude du support. Le temps étant homogène, on pourra toujours se ramener au temps $t = 0$ comme origine pour le temps, et une configuration initiale sera toujours notée c_0.

DÉFINITION 3 (Configuration initiale)
Une configuration initiale *est une configuration* $c_0 : \mathbb{R} \to \mathbb{V}$ *à support fini c'est-à-dire l'ensemble* $\{x \in \mathbb{R} \mid c_0(x) \neq \oslash\}$ *est fini.*

Une configuration initiale est complètement définie par la donnée d'un nombre fini de signaux, chacun étant associé à sa position spatiale sur la droite réelle (toutes les autres positions ayant alors la valeur vide). Nous expliciterons donc une configurations initiale $c_0 : \mathbb{R} \to \mathcal{M} \cup \oslash$ par un ensemble fini de la forme $c_0 = \{\mu_i @ x_i \mid \mu_i \in \mathcal{M},\ x_i \in \mathbb{R}\}_{i \in I}$ où $I \subset \mathbb{N}$ est fini, ou plus simplement par $c_0 = \{\mu_1 @ x_1, \dots \mu_n @ x_n\}$ ($\mu_i @ x_i$ signifiant que le signal μ_i est initialement placé en position $x_i \in \mathbb{R}$ c'est-à-dire que $c_0(x_i) = \mu_i$). On notera également parfois $(\mu_1\, \mu_2 \dots \mu_n)$ pour désigner une configuration telle que les positions x_i respectives des signaux μ_i vérifient $x_1 \leq x_2 \leq \dots \leq x_n$.

La présence de collisions dans une configuration initiale correspond à la possibilité d'avoir plusieurs signaux à une même position (ces signaux sont de vitesses différentes, conformément à la définition d'une règle de collisions). Pour rendre les notations plus claires, on s'autorisera à remplacer une collision dans la configuration initiale par l'ensemble des signaux sortants, placés à la même position que la collision. Ainsi, pour une collision $\{\mu_1, \dots, \mu_p\} \to \{\mu'_1, \dots, \mu'_q\}$ initialement placée en position x, on utilisera la notation $[\mu'_1, \dots, \mu'_q]@x$ ou même $\mu'_1 @ x, \dots, \mu'_q @ x$. Avec cette notation, une configuration initiale peut s'exprimer par un ensemble fini d'éléments de la forme $\mu @ x$, comme cela est illustré par la FIG. 2.4.

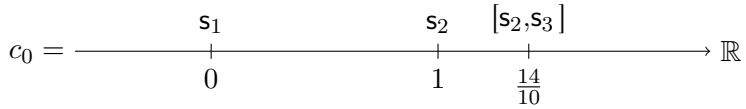


FIGURE 2.4 – Configuration initiale correspondant au diagramme donné par la FIG. 2.3(b) : $c_0 = \{\mathsf{s}_1@0,\ \mathsf{s}_2@1,\ [\mathsf{s}_2, \mathsf{s}_3]@\frac{14}{10}\} = \{\mathsf{s}_1@0,\ \mathsf{s}_2@1,\ \mathsf{s}_2@\frac{14}{10}, \mathsf{s}_3@\frac{14}{10}\}$.

Remarque 2. Nous insistons sur le fait que $[\mu'_1, \dots, \mu'_q]@x = \mu'_1 @ x, \dots, \mu'_q @ x$ est juste une notation pour simplifier l'écriture des collisions quand on exprime une configuration initiale par un ensemble de valeurs non vides. Mais par définition, une configuration c_0 reste une fonction bien définie de $\mathbb{R}$ dans $\mathbb{V}$: pour tout $x \in \mathbb{R}$, la valeur $c(x)$ est unique (et est soit un méta-signal, soit une collision, soit $\oslash$). Nous utiliserons également cette notation pour les collisions dans des configurations quelconques (et pas seulement initiales). Dans ce cas, $\{\mu'_1 @ x, \dots, \mu'_q @ x\}$ devra être interprété par « toute collision en position x produisant l'ensemble $\{\mu'_1, \dots, \mu'_q\}$ en sortie », c'est-à-dire $c(x) = \rho^- \to \rho^+$ avec $\rho^+ = \{\mu'_1, \dots, \mu'_q\}$.

On appelle *diagramme espace-temps d'une machine à signaux* la figure géométrique à deux dimensions obtenue par l'évolution de cette machine à partir d'une configuration initiale. Nous entendons ici par évolution d'une machine l'application des règles définies à chaque collision et la propagation des signaux présents sur la droite réelle (suivant l'EQ. (2.1) décrivant la dynamique d'un signal). L'évolution peut être complètement décrite par l'historique des collisions se produisant au cours du temps sur la droite réelle. Nous formulerons les diagrammes espace-temps de manière topologique en SEC. 2.2, en les considérant dans leur globalité c'est-à-dire sans les munir de dynamique.

Vocabulaire. Nous introduisons ici le vocabulaire qui sera utilisé pour décrire les configurations. Le *support de la configuration c* est l'ensemble des réels dont l'image par c est non vide c'est-à-dire $support(c) = \{x \in \mathbb{R} \mid c(x) \neq \oslash\}$. On appelle *partie significative d'une configuration* (ou *intervalle d'une configuration*) le plus petit segment contenant son support : $\mathcal{I}_c$ est la partie significative de c si pour tout $x \in support(c)$, on a $x \in \mathcal{I}_c$ et si pour tout J vérifiant cette propriété, on a $\mathcal{I}_c \subseteq J$.

À partir des notions de support et partie significatives, nous pouvons définir les notions de cardinal, taille, bornes et extrémaux d'une configuration. On appelle *cardinal d'une configuration* le cardinal de son support : $card(c) = card(support(c))$. On dira que c est une *configuration finie* si elle est de support fini. La *taille d'une configuration*, $|c|$, est donnée par la longueur de sa partie

significative : $|c| = longueur(\mathcal{I}_c)$. La *borne inférieure et supérieure de c* sont définies comme les bornes respectivement inférieure et supérieure de $\mathcal{I}_c$. Lorsque ces bornes sont de valeur non vide, nous parlerons de *minimum* et de *maximum* de la configuration c. Nous noterons ces positions respectivement x^c_{inf}, x^c_{sup}, x^c_{min} et x^c_{max}.

Remarque 3. On peut toujours considérer qu'une configuration s'écrit $c = \{\mu_i@x_i \mid \mu_i \in \mathcal{M},\ x_i \in \mathbb{R}\}_{i\in I}$ avec $I \subseteq \mathbb{N}$. En effet, par définition des configurations, la condition d'isolement des signaux et des collisions implique qu'il ne peut pas y avoir plus d'un nombre dénombrable de valeurs non vides dans une configuration. Le support d'une configuration c pourra donc toujours s'écrire $support(c) = \{x_i \in I \mid I \subseteq \mathbb{N}\}$.

La notion de support permet d'exprimer de nombreuses caractéristiques de configuration à partir de son support ou de sa partie significative. Par exemple, le fait qu'une configuration c ne contienne pas de collision est équivalent à $support(c) \cap \mathcal{R} = \emptyset$, et le fait que le signal le plus à gauche dans c est un signal de type μ s'exprime par $c(min\{support(c)\}) = \mu$.

Configurations rationnelles et machines rationnelles. Les machines à signaux sont définies sur $\mathbb{R}$ (vitesses à valeurs réelles et positions réelles), mais beaucoup de constructions ne nécessitent en fait que peu de vitesses, que l'on peut souvent choisir rationnelles, et même entières. De même, des positions initiales rationnelles permettent de générer la plupart des constructions présentées ici. Les définitions suivantes formalisent de telles *machines rationnelles* :

Définition 4 (Configuration rationnelle)
On dit que c est une configuration rationnelle *si et seulement si* $\forall x \in \mathbb{R}\ c(x) \neq \oslash \Rightarrow x \in \mathbb{Q}$.

Définition 5 (Machine à signaux rationnelle)
Une machine à signaux $\mathfrak{M} = (\mathcal{M}, \mathcal{S}, \mathcal{R})$ *est une* machine rationnelle *si et seulement si* $\mathcal{S}(\mathcal{M}) \subseteq \mathbb{Q}$ *et toute configuration initiale de* $\mathfrak{M}$ *est rationnelle.*

C'est-à-dire qu'une machine est rationnelle si toutes les vitesses ont des valeurs rationnelles et si toutes les positions initiales de valeur non vide considérées pour $\mathfrak{M}$ sont aussi des nombres rationnels. Les positions des collisions sont solutions de systèmes d'équations rationnelles linéaires (données par les équations des mouvement des signaux, qui ont une vitesse constante). Donc dans le cas d'une machine rationnelle, tant qu'il n'y pas d'accumulation, les coordonnées de toutes les collisions resteront rationnelles, car $\mathbb{Q}$ étant un corps, toute solution d'équations du premier degré à coefficients dans $\mathbb{Q}$ est encore dans $\mathbb{Q}$.

Bien que restreintes, les machines à signaux rationnelles gardent un fort pouvoir de calcul : elles sont encore Turing-universelles [Durand-Lose 2006b]. Toutes les machines considérées dans ce manuscrit sont des machines rationnelles (la plupart ayant même leur fonction vitesse à valeur dans $\mathbb{Z}$), à l'exception de celles du Chap. 4 où nous démontrerons des propriétés sur les accumulations qui dépendent de la (non) rationalité relative entre certaines grandeurs (et non de leur rationalité absolue).

Notations et conventions d'écriture. Dans tout le manuscrit, nous adopterons les notations et conventions d'écriture suivantes.

Pour rendre plus claires les définitions de signaux, leurs noms seront surmontés de flèche (à la manière d'un vecteur) pour indiquer le signe de leur vitesse : une flèche droite (resp. gauche) indiquera une vitesse strictement positive (resp. strictement négative). Ainsi, un signal $\overrightarrow{\mu_1}$ sera de vitesse positive alors que $\overleftarrow{\mu_2}$ sera de vitesse négative et se déplacera vers la gauche. Le nom d'un signal sans flèche, comme μ_0, désignera en général un signal de vitesse nulle c'est-à-dire un signal stationnaire. Toutefois, on trouvera dans certains contextes, des noms de signaux qui ne seront pas surmontés d'une flèche, mais qui pourront avoir des vitesses non-nulles (ce sera le

cas lorsque la vitesse d'un signal, et donc son signe, nous est inconnue). Notons que $\overrightarrow{\mu}$ et $\overleftarrow{\mu}$ désignent bien deux méta-signaux différents : l'un allant vers la droite, l'autre vers la gauche (leur vitesse étant différente, ils constituent des entités différentes). L'utilisation du symbole sous la flèche (μ ici) indiquera que $\overrightarrow{\mu}$ et $\overleftarrow{\mu}$ ont des rôles similaires, c'est-à-dire qu'ils auront les mêmes fonctionnalités dans une construction, l'un étant simplement la version droite et l'autre la gauche. Nous utiliserons également parfois une double flèche lorsque nous aurons besoin d'un signal se dirigeant dans la même direction qu'un signal de même type mais plus rapidement, c'est-à-dire ayant une vitesse de même signe, mais de valeur absolue strictement supérieure. Dans ce cas, $\overrightarrow{\mu}$ désignera une version « lente » de μ se dirigeant à droite et $\overrightarrow{\overrightarrow{\mu}}$ désignera une version « rapide ».

On confondra souvent signaux et méta-signaux, que l'on notera μ de manière générale. Cependant, dans certains contextes, on désignera plutôt les signaux par σ, soulignant ainsi le fait qu'ils appartiennent à un diagramme et sont des instances de méta-signaux (et sont donc munis de dates de création et d'une date de fin qui peut éventuellement être infinie). Pour les méta-signaux particuliers, qui ont une utilisation bien définie dans le cadre d'une machine particulière, leur nom sera donné avec une police sans empattement (police de type « sf »). Leur nom explicitera leur rôle ou fonction, soit de manière explicite, soit en abrégé : $\overrightarrow{\mathsf{start}}$ et add désigneront des méta-signaux (et les signaux correspondants) dont les rôles respectifs sont d'initier un processus et de procéder à une addition.

On utilisera ρ pour désigner une règle de collisions de manière générale et ρ^- et ρ^+ seront les ensembles de signaux respectivement entrant et sortant, de telle sorte que ρ s'écrira $\rho = \rho^- \to \rho^+$. On notera parfois une collision par C pour insister sur le fait qu'elle est instanciée (et admet des coordonnées) dans un diagramme espace-temps. On écrira préférentiellement les ensembles de signaux entrants et sortants ordonnés respectivement par vitesses décroissantes et par vitesses croissantes. L'ordre d'écriture suivra ainsi l'ordre d'apparition des signaux dans une collision dans le diagramme espace-temps, lorsque celui-ci est parcouru de la gauche vers la droite.

Les machines seront notées avec des lettres gothiques, éventuellement munies d'indices et d'exposants indiquant respectivement la fonction et les paramètres de la machine. Par exemple, $\mathfrak{M}$ désignera une machine dans un contexte général et $\mathfrak{M}^R_{add}$ désignera la machine pour effectuer une addition binaire vers la droite. Les ensembles de méta-signaux seront toujours désignés par $\mathcal{M}$, les ensembles de règles par $\mathcal{R}$ et les fonctions vitesses par $\mathcal{S}$.

Une configuration sera toujours notée c et les diagrammes espace-temps seront notés par $\mathcal{D}$. De plus, le temps t d'une configuration pourra être précisé en indice et des paramètres de la configuration pourront être donnés en exposants : c_1^x désigne ainsi une configuration au temps $t = 1$ et utilise un paramètre x dans sa description (pour l'une de ses positions par exemple). L'ensemble des configurations d'une machine sera noté $\mathcal{C}onfig$. Lorsque plusieurs machines seront considérées, on rajoutera éventuellement en indice le nom de la machine ou l'ensemble des valeurs des configurations *i.e.* on notera $\mathcal{C}onfig_{\mathfrak{M}}$ ou $\mathcal{C}onfig_{\mathbb{V}}$ pour désigner l'ensemble des configurations de la machine $\mathfrak{M}$ ou l'ensemble des configurations à valeurs dans $\mathbb{V}$. On notera $\mathcal{C}onfig^{fini}_{\mathfrak{M}}$ (ou tout simplement $\mathcal{C}onfig^{fini}$) l'ensemble des configurations finies de la machine $\mathfrak{M}$.

Concernant l'évolution d'une machine, on désignera par $\mathfrak{M}(c_0)$ le diagramme complet obtenu par l'évolution de la machine $\mathfrak{M}$ exécutée à partir de la configuration c_0. Lorsque nous voudrons considérer uniquement la configuration obtenue par l'évolution de la machine, nous spécifierons alors une durée en indice : la notation $\mathfrak{M}_d(c_0)$ désignera ainsi la configuration obtenue après l'évolution de la machine $\mathfrak{M}$ pendant la durée d et à partir de c_0.

2.2 Point de vue topologique

On fournit ici une définition purement topologique des diagrammes espace-temps. Cette définition fournit un cadre naturel pour exprimer la notion d'accumulation, puisqu'elle définit les accumulations dans le cadre des machines à signaux à partir de la notion d'accumulation topologique.

Espace-temps et topologie considérée. Un diagramme espace-temps peut être reformulé en des termes topologiques, à partir de la topologique classique sur $\mathbb{R}^2$ dont les ouverts sont ceux générés par la distance euclidienne c'est-à-dire la distance d définie par :
$\forall (x,t),(x',t') \ \in \mathbb{R}\times\mathbb{R}^+ \ \ d((x,t),(x',t')) = \sqrt{(x-x')^2+(t-t')^2}$.

Cette distance génère une famille d'ouverts dont les ouverts de base sont des boules euclidiennes : la boule de rayon ε et centrée en (x,t) est $\mathcal{B}_\varepsilon(x,t) = \{(x',t') \mid d((x,t),(x',t')) < \varepsilon\}$. Cette topologie est la topologie produit *i.e.* un ouvert de $\mathbb{R}\times\mathbb{R}^+$ est le produit d'un ouvert de $\mathbb{R}$ et d'un ouvert de $\mathbb{R}^+$ (tous deux au sens de la topologie usuelle sur $\mathbb{R}$). Nous pourrons également raisonner en termes de voisinages topologiques.

Configurations étendues et accumulations. Nous commençons par donner une définition plus large des configurations pour prendre en compte la notion d'accumulation avec le symbole ✺. Ces *configurations étendues* sont définies de manière que les configurations définies en Déf. 2 mais la valeur ✺ est ajoutée aux valeurs possibles, et est attribuée à une position x qui est un point d'accumulation pour un ensemble de positions de valeurs non vides. On désigne par $\mathbb{V}^+$ l'*ensemble des valeurs étendues* : $\mathbb{V}^+ = \mathbb{V}\cup\{✺\} = \mathcal{M}\cup\mathcal{R}\cup\{\oslash, ✺\}$.

On rappelle qu'un point $x \in X$ (où X est un ensemble topologique) est un *point d'accumulation* (au sens topologique) de l'ensemble $E \subseteq X$ si pour tout voisinage $\mathcal{V}_x$ de x, l'ensemble $E \cap \mathcal{V}_x$ est non-vide. Avec $X = \mathbb{R}$ et sa topologie classique, la condition devient : $\forall\varepsilon,\ card(E\cap(x-\varepsilon, x+\varepsilon)) > 1$.

Définition 6 (Configuration étendue)
Une configuration étendue *est une fonction* $c : \mathbb{R} \to \mathbb{V}^+$ *telle que :*

(i) tous les signaux et collisions sont isolés :
$\forall x \in \mathbb{R},\ c(x) \in \mathcal{M}\cup\mathcal{R} \ \Rightarrow \exists\varepsilon > 0,\ \forall y,\ 0 < |x-y| < \varepsilon \Rightarrow c(y) = \oslash$;

(ii) tout x qui est un point d'accumulation de $c^{-1}(\mathbb{V}\setminus\{\oslash\})$ vérifie $c(x) = ✺$.

Nous utiliserons le terme de configuration à la fois pour désigner des configurations simples (à valeurs dans $\mathbb{V}$) et des configurations étendues (à valeurs dans $\mathbb{V}^+$) ; l'ensemble des valeurs sera précisé quand le contexte l'imposera.

Un point $x \in \mathbb{R}$ vérifiant la condition *(ii)* de la Déf. 6 est appelé *accumulation spatiale* (ou *accumulation statique*). Une accumulation spatiale en x correspond en fait à une accumulation topologique du support de la configuration (définie sur l'espace de la machine) : le point x est un point d'accumulation de $support(c) \subseteq \mathbb{R}$ et vérifie donc $\forall\varepsilon,\ 1 < card(support(c)\cap(x-\varepsilon, x+\varepsilon))$.

Un autre type d'accumulation peut être envisagé : des *accumulations temporelles* ou *accumulation dynamique*. Elles correspondent également à des accumulations au sens topologique, mais sur l'espace-temps et non plus seulement sur l'espace. Pour définir ces accumulations, nous utilisons la notion de *passé causal* :

Définition 7 (Passé causal)
Soit ν_{max} (vitesse maximale droite) et ν_{min} (vitesse maximale gauche) les valeurs maximale et minimale prises par la fonction vitesse $\mathcal{S}$. On définit le passé causal (ou cône de lumière) au point (x_0,t_0) *par :* $\Gamma^-(x_0,t_0) = \left\{ (x,t) \,\middle|\, t_0 < t \wedge \nu_{max}\cdot(t-t_0) < x-x_0 < \nu_{min}\cdot(t-t_0) \right\}$.

La FIGURE 2.5 illustre la notion de cône de lumière : la valeur de la position (x_0, t_0) dans le diagramme espace-temps dépend uniquement des valeurs prises dans la zone blanche inférieure. De même, la valeur en (x_0, t_0) ne pourra avoir d'influence que sur les positions situées dans la zone blanche supérieure.

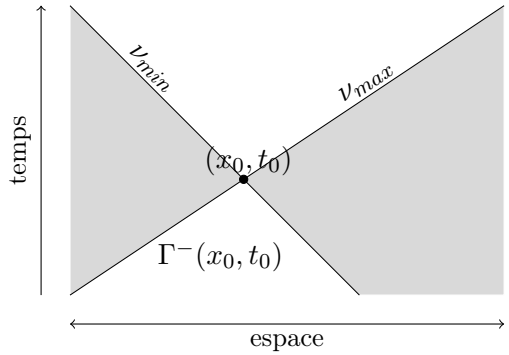


FIGURE 2.5 – Passé causal et cône de lumière.

Remarque 4. La valeur V définie par $V = max\{|\nu_{max}|, |\nu_{min}|\}$ représente la vitesse maximale de propagation de l'information dans la machine à signaux considérée, et peut être vue comme l'équivalent de la vitesse de la lumière dans la théorie de la relativité restreinte.

La notion de passé causal permet de définir des accumulations ne pouvant être provoquées que d'une manière dynamique : les valeurs s'accumulent dans le passé causal et non plus seulement dans l'espace comme c'est la cas pour les accumulations statiques :

DÉFINITION 8 (Accumulation dynamique)
On dit que le point (x_0, t_0) *est une* accumulation dynamique *si pour tout voisinage* $\mathcal{V}_{(x_0,t_0)}$ *de* (x_0, t_0)*, il y a une infinité de signaux ou collisions dans* $\mathcal{V}_{(x_0,t_0)} \cap \Gamma^-(x_0, t_0)$.

Diagrammes espace-temps topologiques. Les diagrammes espace-temps de machines à signaux peuvent être reformulé en termes topologiques, à partir de la distance euclidienne sur $\mathbb{R}^2$ (et plus généralement à partir d'ouverts). La définition topologique d'un diagramme implique que l'espace et le temps sont considérés dans un même objet unique — une structure d'espace-temps — *a priori* dénué de dynamique. Formellement :

DÉFINITION 9 (Diagramme espace-temps topologique)
Un diagramme espace-temps topologique associé à une machine à signaux $\mathfrak{M} = (\mathcal{M}, \mathcal{S}, \mathcal{R})$ *est une application* $\mathcal{D}$ *définie sur* $\mathbb{R} \times [0, T]$ *où* $[0, T] \subset \mathbb{R}^+$ *est un intervalle de temps (*T *peut être infini) sur* $\mathbb{V}^+$ *l'ensemble des valeurs étendue de* $\mathfrak{M}$*, telle que :*

(i) *l'ensemble* $\{x \in \mathbb{R} \mid \mathcal{D}(x, 0) \neq \oslash\}$ *est fini et* $\forall x \in \mathbb{R},\ \mathcal{D}(x, 0) \neq \text{✺}$;

(ii) *si* $\mathcal{D}(x, t) = \mu \in \mathcal{M}$ *alors* $\exists t_i, t_f \in [0, T]$ *avec soit* $t_i < t < t_f$ *soit* $0 = t_i = t < t_f$ *soit* $t_i < t = t_f = T$ *tels que :*
- $\forall t' \in]t_i, t_f[\ \mathcal{D}(x, t') + \mathcal{S}(\mu)(t - t')) = \mu$,
- $t_i = 0$ *ou* $\mathcal{D}(x_i, t_i) \in \mathcal{R}$ *et* $\mu \in (\mathcal{D}(x_i, t_i))^+$ *avec* $x_i = x + \mathcal{S}(\mu)(t_i - t)$,
- $t_f = T$ *ou* $\mathcal{D}(x_f, t_f) \in \mathcal{R}$ *et* $\mu \in (\mathcal{D}(x_f, t_f))^-$ *avec* $x_f = x + \mathcal{S}(\mu)(t_f - t)$;

(iii) *si* $\mathcal{D}(x, t) = \rho^- \to \rho^+ \in \mathcal{R}$ *alors* $\exists\ \varepsilon > 0,\ \forall t' \in [t - \varepsilon, t + \varepsilon],\ \forall x' \in [x - \varepsilon, x + \varepsilon]$
- $\mathcal{D}(x', t') \in \rho^- \cup \rho^+ \cup \{\oslash\}$,
- $\forall \mu \in \rho^- : (\ \mathcal{D}(x', t') = \mu\ ) \Leftrightarrow (\ t < t'$ *et* $x' = x + \mathcal{S}(\mu)(t' - t)\ )$,
- $\forall \mu \in \rho^+ : (\ \mathcal{D}(x', t') = \mu\ ) \Leftrightarrow (\ t' < t$ *et* $x' = x + \mathcal{S}(\mu)(t' - t)\ )$;

(iv) *si* $\mathcal{D}(x, t) = \text{✺}$ *alors*

- *soit* $\forall\ \varepsilon > 0,\ \{x' \in \mathbb{R} \mid \mathcal{D}(x',t) \in \mathcal{M} \cup \mathcal{R}\} \bigcap [x-\varepsilon; x+\varepsilon] \neq \emptyset$;
- *soit* $\exists\ \varepsilon > 0,\ \forall (x',t') \notin \Gamma^-(x,t)$ *tel que* $|x-x'| < \varepsilon,\ |t-t'| < \varepsilon$ *on a* $\mathcal{D}(x',t') = \oslash$ *et* $\forall\ \varepsilon > 0,\ card(\{(x',t') \in \Gamma^-(x,t) \mid t-\varepsilon < t' < t \wedge \mathcal{D}(x',t') \in \mathcal{R}\}) = \infty$.

Comme $\mathbb{V}^+ = \mathcal{M} \cup \mathcal{R} \cup \{\oslash\} \cup \{\text{✺}\}$, on a $\mathcal{D}(x,t) = \oslash$ lorsque le point (x,t) ne valide aucune des conditions de la DÉF. 9 (toutes les valeurs de $\mathbb{V}^+$ sont exclues sauf $\oslash$). La condition *(i)* signifie que la base du diagramme est une configuration initiale telle que définie en DÉF. 3, qui en plus ne contient pas d'accumulation. La condition *(ii)* implique que pour tout point (x,t) tel que $\mathcal{D}(x,t) = \mu$, il existe une boule ouverte centrée sur ce point telle que la seule valeur non vide dans cette boule est μ. La FIGURE 2.6 illustre pour chaque valeur possible l'existence de boules ouvertes telles que la valeur prise dans une boule est entièrement définie par une des conditions de la DÉF. 9. Les deux cas de la condition *(iv)* correspondent aux deux notions d'accumulation, respectivement temporelle et spatiale.

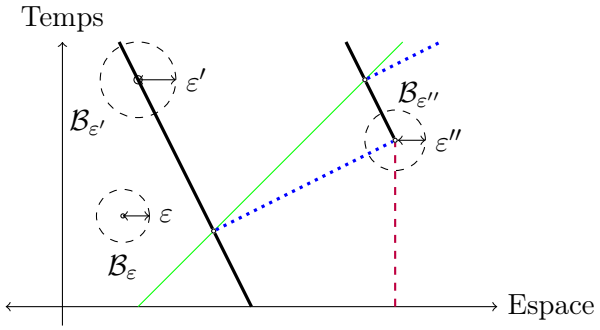


FIGURE 2.6 – Boules centrées sur des valeurs vide ($\mathcal{B}_\varepsilon$), signal ($\mathcal{B}_{\varepsilon'}$) et collision ($\mathcal{B}_{\varepsilon''}$).

L'application $\mathcal{D} : \mathbb{R} \times [0,T] \to \mathbb{V}^+$ peut également être identifié à l'application définie de $[0,T]$ dans l'ensemble des configurations étendues *i.e.* $\mathcal{D} : t \in [0,T] \mapsto c_t$. Cette définition équivalente revient à « découper » le diagramme en sections correspondant aux temps $t \in [0,T]$: le diagramme est alors formé d'une succession de configurations c_t avec $t \in [0,T]$ et on a $c_t = x \mapsto \mathcal{D}(x,t)$. On utilisera toujours la notation $\mathcal{D}$ quand il s'agit d'un diagramme espace-temps (défini sur $\mathbb{R} \times \mathbb{R}^+$) et on gardera la notation c pour désigner une configuration (définie sur $\mathbb{R}$).

L'évolution d'une machine à signaux à partir d'une configuration initiale finie correspond (jusqu'à l'éventuelle apparition d'accumulations) à un diagramme espace-temps topologique qui respecte la DÉF. 9 (voir [Durand-Lose 2003]).

Remarque 5. On pourrait définir des diagrammes restreints en ajoutant à la DÉF. 9 une condition de finitude des configurations : $\forall t \in [0,T]$ l'ensemble $\{x \in \mathbb{R} \mid \mathcal{D}(x,t) \neq \oslash\}$ est fini. Dans ce cas, les accumulations sont nécessairement dynamiques (car une accumulation spatiale au temps t implique que le support de c_t soit infini).

Diagrammes équivalents. On définit l'équivalence de deux diagrammes espace-temps : intuitivement, deux diagrammes sont *équivalents* s'ils ont la même structure, c'est-à-dire les mêmes liens de causalité entre leurs collisions et signaux respectifs, indépendamment de leurs positions et du nom des méta-signaux. Nous modélisons l'invariance de structure grâce à la notion d'homéomorphisme : un diagramme obtenu par déformation continue gardera la même structure topologique.

DÉFINITION 10 (Diagrammes espace-temps équivalents)
Soient $\mathcal{D}$ *et* $\mathcal{D}'$ *deux diagrammes espace-temps de machines à signaux respectivement à valeurs dans* $\mathbb{V}^+ = \mathcal{M} \cup \mathcal{R} \cup \{\oslash, \text{✺}\}$ *et* $\mathbb{V}^{+\prime} = \mathcal{M}' \cup \mathcal{R}' \cup \{\oslash, \text{✺}\}$. *On dit que* $\mathcal{D}$ *et* $\mathcal{D}'$ *sont des* diagrammes équivalents *si :*

(i) il existe un homéomorphisme $h : \mathbb{R} \times \mathbb{R}^+ \to \mathbb{R} \times \mathbb{R}^+$, *et*

(ii) il existe une bijection $\Phi : \mathbb{V}^+ \to \mathbb{V}^{+\prime}$ *telle que* $\Phi[\mathcal{M}] = \mathcal{M}'$, $\Phi[\mathcal{R}] = \mathcal{R}'$, $\Phi(✺) = ✺$ *et* $\Phi(\oslash) = \oslash$ *et telle que* $\forall \mu \in \mathcal{M}, \forall \rho \in \mathcal{R},\ \mu \in \rho^- \ (resp.\rho^+) \Rightarrow \Phi(\mu) \in \Phi(\rho)^- \ (resp.\Phi(\rho)^+)$,

qui vérifient $\forall (x,t) \in \mathbb{R} \times \mathbb{R}^+,\ \mathcal{D}'(x,y) = \Phi(\mathcal{D}(h(x,y)))$.

La deuxième condition signifie que Φ induit une bijection de l'ensemble des méta-signaux de $\mathcal{D}$ sur les méta-signaux de $\mathcal{D}'$ et de l'ensemble des règles de $\mathcal{D}$ dans celles de $\mathcal{D}'$ telle que les rôles des méta-signaux dans les règles sont conservés. Notons aussi qu'il est possible d'induire une application $\Phi : \mathbb{V}^+ \to \mathbb{V}^{+\prime}$ à partir d'une bijection $\phi : \mathcal{M} \to \mathcal{M}'$. Il suffit pour cela de poser $\Phi(\mu) = \phi(\mu)$ (cas d'un méta-signal), $\Phi\left(\{\mu_i^-\}_{1\leq i\leq n} \to \{\mu_j^+\}_{1\leq j\leq m}\right) = \{\phi(\mu_i^-)\}_{1\leq i\leq n} \to \{\phi(\mu_j^+)\}_{1\leq j\leq m}$ (cas d'une règle), $\Phi(✺) = ✺$ et $\Phi(\oslash) = \oslash$. La condition *(ii)* de la Déf. 10 est alors équivalente à l'existence d'une bijection ϕ de $\mathcal{M}$ dans $\mathcal{M}'$ telle que Φ, l'application induite par ϕ de $\mathbb{V}^+$ dans $\mathbb{V}^{+\prime}$, est une bijection.

La définition implique que si $\mathcal{D}$ et $\mathcal{D}'$ sont équivalents, alors le diagramme de Fig. 2.7(a) commute. L'application de l'homéomorphisme h déforme l'espace-temps de manière continue et la structure du diagramme sera préservée lors de cette transformation. On peut également remplacer la condition « *h homéomorphisme* » par la condition équivalente « *h bijection continue* ».

Si les deux diagrammes ont le même ensemble de valeurs c'est-à-dire si ce sont deux diagrammes d'une même machine, alors seule l'existence de l'homéomorphisme h est requise, une bijection de $\mathbb{V}^+$ dans $\mathbb{V}^{+\prime} = \mathbb{V}^+$ étant trivialement donnée par l'identité. La condition devient alors : $\forall (x,t) \in \mathbb{R} \times \mathbb{R}^+\ \ \mathcal{D}'(x,y) = \mathcal{D}(h(x,y))$, et on peut simplifier le diagramme commutatif en celui de la Fig. 2.7(b).

(a) Cas général. (b) Cas de diagrammes de mêmes valeurs.

Figure 2.7 – Diagramme commutatif de diagrammes espace-temps équivalents.

Les deux diagrammes $\mathcal{D}$ et $\mathcal{D}'$ étant associés à des machines, ils vérifient tous deux la Déf. 9. Il est alors immédiat que pour tout $(x,t) \in \mathbb{R} \times \mathbb{R}^+$, pour tout $v \in \mathbb{V}^{+\prime}$, on a : $\mathcal{D}'(x,t) = v$ vérifie l'une des conditions de la Déf. 9 si et seulement si $\mathcal{D}(h(x,t)) = \Phi^{-1}(v)$ vérifie la même condition. En effet, la fonction h étant un homéomorphisme, l'existence d'ouverts est préservée.

Des diagrammes équivalents vérifieront les mêmes propriétés structurelles, comme l'enchaînement et la causalité entre les collisions, la finitude du diagramme,*etc*. Nous verrons en Sec. 2.4 des moyens de générer des diagrammes équivalents à un diagramme donné.

2.3 Point de vue dynamique

Nous présentons ici l'évolution d'une machine en termes de dynamique, en explicitant la construction de diagramme espace-temps en fonction des temps des collisions.

Suite des temps de collisions et front de collisions. Les collisions correspondent aux opérations élémentaires de calcul dans le modèle, et l'historique de leurs coordonnées permet de connaître un diagramme espace-temps. Nous nous focalisons ici sur les temps de collisions car ils correspondent aux changements significatifs dans le diagramme espace-temps, la dynamique

entre deux temps successifs de collisions correspondant à la propagation uniforme et linéaire des signaux. Pour formuler la suite des temps de collisions, nous utilisons la notion de *délai jusqu'à la prochaine collision* :

Définition 11 (Délai jusqu'à la prochaine collision)
À partir d'un configuration c, le délai jusqu'à la prochaine collision $\Delta(c)$, *est défini comme le plus petit réel positif d tel que :*

$$\exists x_1, x_2 \in \mathbb{R}, \exists \mu_1, \mu_2 \in \mathcal{M} \begin{cases} x_1 + d{\cdot}\mathcal{S}(\mu_1) = x_2 + d{\cdot}\mathcal{S}(\mu_2) \\ c(x_1) = \mu_1 \vee (c(x_1) = \rho^- \to \rho^+ \wedge \mu_1 \in \rho^+) \\ c(x_2) = \mu_2 \vee (c(x_2) = \rho^- \to \rho^+ \wedge \mu_2 \in \rho^+) \end{cases} .$$

Cette valeur est $+\infty$ s'il n'existe pas de tel d.

Étant donné un instant t, plusieurs collisions peuvent se produire de manière simultanée à l'instant $t + \Delta(c_t)$ (en effet, la définition n'impose pas l'unicité des signaux rentrant en collision après la durée d). On parlera dans ce cas de *front de collisions au temps t*. De même, la collision considérée pour définir $\Delta(c)$ peut se produire entre plus de deux signaux entrants. À partir du délai entre deux collisions successives, nous définissons :

Définition 12 (Suite des temps de collisions)
La suite des temps de collisions $(t_n)_n$ *est définie par :*

$$\begin{cases} t_0 = 0 \\ t_{n+1} = t_n + \Delta(c_{t_n}) \end{cases}$$

Cette suite est finie s'il existe $n \in \mathbb{N}$ tel que $\Delta(c_{t_n}) = +\infty$. Sinon, la suite est infinie et elle est strictement croissante par définition, elle admet une limite : soit ∞, soit une limite finie que l'on appelle $\tilde{t}$. La Figure 2.8 donne un exemple de diagramme simple sur lequel sont positionnées les premières valeurs de la séquence des temps de collisions.

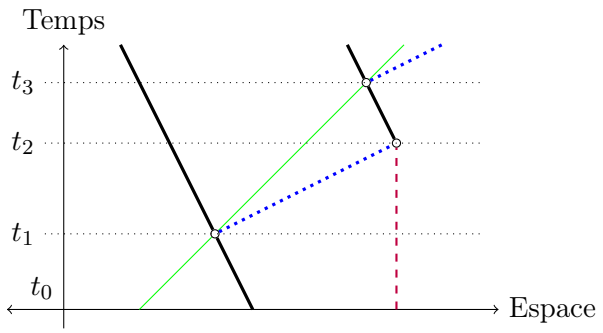


Figure 2.8 – Un diagramme et les premiers temps de collisions.

Dynamique du calcul. Nous explicitons maintenant la *construction dynamique d'un diagramme espace-temps.*

Définition 13 (Dynamique)
Soit c_t une configuration au temps t. On définit pour tout $t' \in [t; \Delta(c_t)[$ la configuration à l'instant t' de la manière suivante. D'abord, les signaux sont disposés de telle sorte que $c_{t'}(x') = \mu$ si et seulement si $c_t(x) = \mu \vee (c_t(x) = \rho^- \to \rho^+ \wedge \mu \in \rho^+)$ où $x = x' + (t{-}t'){\cdot}\mathcal{S}(\mu)$. Il n'y a aucune collision puisque $t' < t + \Delta(c_t)$. Ensuite les accumulations (statiques) sont placées : $c_{t'}(x') = \text{✺}$

si et seulement si x' est un point d'accumulation de $c_{t'}^{-1}(\mathcal{M})$. Toutes les autres positions ont une valeur nulle : pour tout autre x, on a $c_{t'}(x) = \oslash$.

Pour la configuration à l'instant $t' = t + \Delta(c_t)$, les collisions sont placées en premier : $c_{t'}(x') = \rho^- \to \rho^+$ si et seulement si pour tout $\mu \in \rho^-$, $c_t(x_\mu) = \mu \vee (c_t(x_\mu) = \rho^- \to \rho^+ \wedge \mu \in \rho^+)$ avec $x_\mu = x' + (t-t')\cdot\mathcal{S}(\mu)$. Puis les signaux sont placés de la même manière que précédemment, ainsi que les accumulations (statiques).
Si la suite $(t_n)_n$ est finie ou si sa limite est infinie, alors le diagramme espace-temps est entièrement défini. Dans le cas d'une limite finie $\tilde{t}$, la configuration au temps $\tilde{t}$ est définie de la manière suivante. Les accumulations (dynamiques) sont d'abord placées : $c_{\tilde{t}}(x) = \text{✺}$ si et seulement si $\forall \varepsilon > 0\ \exists x' \in \mathbb{R},\ t' \in \mathbb{R}^+$ tels que $|x - x'| < \varepsilon$, $\tilde{t} - \varepsilon < t' < \tilde{t}$ et $c_{t'}(x') \in \mathcal{M} \cup \mathcal{R}$. Ensuite, les collisions : $c_{\tilde{t}}(x) = \rho^- \to \rho^+$ si et seulement si $\forall \mu \in \rho^-$, $\exists \varepsilon$, $\forall \varepsilon'$, $0<\varepsilon'<\varepsilon$ tel que $c_{\tilde{t}-\varepsilon'}(x' - \varepsilon'\cdot\mathcal{S}(\mu)) = \mu$. Puis les méta-signaux sont placés : $c_{\tilde{t}}(x) = \mu$ si et seulement si $\exists \varepsilon$, $\forall \varepsilon'$, $0<\varepsilon'<\varepsilon$, alors $c_{\tilde{t}-\varepsilon'}(x' - \varepsilon'\cdot\mathcal{S}(\mu)) = \mu$. Enfin, les accumulations statiques sont placées.

À chaque instant, une valeur ne peut être associée à une position (selon la définition précédente) uniquement si aucune valeur n'a déjà été fixée pour cette position. Toutes les positions non encore définies à la fin prennent la valeur $\oslash$. La dynamique est uniforme à la fois en temps et en espace.

Le programme `java` de Jérôme Durand-Lose qui permet de calculer et tracer les diagrammes espace-temps, utilise un principe similaire à celui énoncé dans la DÉF. 13, sans toutefois gérer les valeurs d'accumulations. À partir d'une configuration donnée, toutes les collisions possibles sont calculées, fournissant ainsi le temps du prochain front de collisions (donné par la date de collision de plus petite valeur). La nouvelle configuration au temps de collisions suivant est alors mise à jour en respectant les règles de collisions et les nouvelles positions des signaux. Puis le processus est réitéré jusqu'à atteindre un nombre de fronts de collisions initialement fixé.

Remarque 6. Cette définition ne définit pas toujours une extension au calcul. En effet, quand il y a une infinité de signaux présents à l'instant t dans la configuration c_t, $\Delta(c_t)$ n'est plus défini comme un minimum mais comme une borne inférieure qui peut valoir 0.

2.4 Propriétés géométriques

Nous rappelons ici quelques propriétés géométriques des machines à signaux, notamment l'application de transformations géométriques au calcul. Nous introduisons ensuite de nouvelles notions, les *supports* et l'*inclusion* de diagrammes, qui constituent des outils adaptés à l'étude de la structure des diagrammes espace-temps.

2.4.1 Transformations par fonctions affines

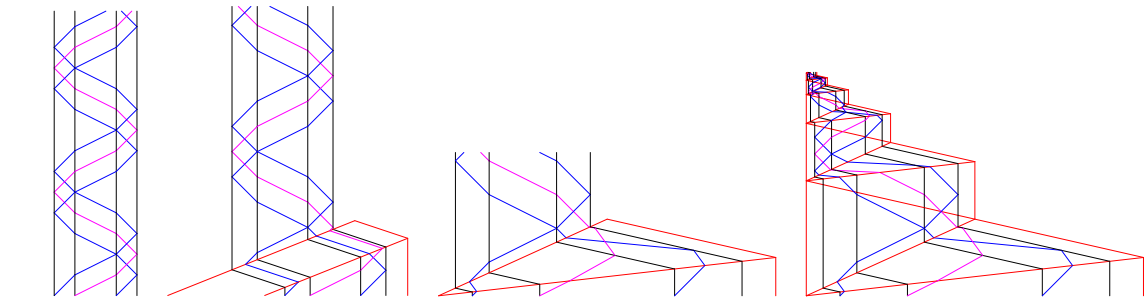
Nous rappelons ici quelques outils géométriques de base, qui ont été pour la plupart définis et utilisés dans [Durand-Lose 2003].

Transformations géométriques du calcul. Les transformations affines usuelles permettent de définir de nouvelles primitives sur le calcul, à niveau plus élevé que ceux des signaux et collisions. En appliquant des transformations simples aux diagrammes, il devient possible de « geler », de déplacer ou de contracter/dilater le calcul respectivement par deux signaux parallèles, par une translation et par une homothétie.

Quelques signaux et règles simples (ajoutés à la machine) suffisent pour transformer les diagrammes selon des applications affines de partie linéaire positive (qui peuvent s'écrire comme composées d'une translation et d'une homothétie de rapport positif). La FIGURE 2.9(b) illustre les effets de telles transformations sur l'exemple de diagramme en forme de bande donné en

FIG. 2.9(a) : une translation et une contraction (*i.e.* une homothétie de rapport $0 < r < 1$) lui sont appliquées respectivement en FIG. 2.9(b) et FIG. 2.9(c).

Par ailleurs, il est possible de ramener tout diagramme dans une zone d'espace-temps triangulaire. Pour cela, des contractions comme celle présentée en FIG. 2.9(c) sont itérées de telle sorte que le diagramme soit toujours inclus dans cette structure contractante bornée par un triangle : la FIG. 2.9(d) illustre le diagramme résultant d'une telle construction. On remarque que sur cet exemple (ainsi que sur tout exemple donné à partir d'un diagramme infini), comme le diagramme de référence donné en FIG. 2.9(a) contient une infinité de collisions et qu'il est ici inséré dans la structure itérative, celle-ci contiendra également une infinité de collisions, produisant ainsi une accumulation tout haut de la structure triangulaire.



(a) Référence. (b) Translation. (c) Contraction. (d) Contractions itérées.

FIGURE 2.9 – Exemples de transformations géométriques [Durand-Lose 2003, Fig. 6.2, 7.6 et 7.9].

Transformations des vitesses. Nous montrons ici qu'il est possible de changer les vitesses d'une machine à signaux sans changer la nature des calculs qu'elle effectue, c'est-à-dire tout en gardant des diagrammes espace-temps équivalents.

Appliquer une fonction affine de ratio strictement positif aux valeurs de vitesses d'une machine ne change pas la structure du diagramme espace-temps engendré par la machine :

LEMME 1 (Invariance du calcul par transformation affine des vitesses)
Soit $\mathfrak{M}$ une machine à signaux et $f : \mathbb{R} \to \mathbb{R}$ une fonction affine de ratio strictement positif. Soit $\mathfrak{M}_f$ la machine à signaux obtenue en appliquant f à toutes les vitesses de $\mathfrak{M}$ c'est-à-dire que la fonction vitesse de $\mathfrak{M}_f$ est égale à $f \circ \mathcal{S}$ où $\mathcal{S}$ est la fonction vitesse de la machine $\mathfrak{M}$. Alors $\mathfrak{M}_f$ génère des diagrammes espace-temps équivalents à ceux générés par la machine $\mathfrak{M}$.

Démonstration. Soit $a \in \mathbb{R}^+$ et $b \in \mathbb{R}$ tels que $f(x) = a \cdot x + b$ pour tout $x \in \mathbb{R}$. Soit $\mathcal{D}$ un diagramme espace-temps de $\mathfrak{M}$ et soit $\mathcal{D}'$ le diagramme généré par $\mathfrak{M}_f$ sur la même configuration initiale que $\mathcal{D}$.

Ajouter la constante b à toutes les vitesses dévie progressivement toutes les positions spatiales mais garde les dates de collisions inchangées. On vérifie facilement que $(x,t) \in \mathbb{R} \times \mathbb{R}^+$, $\mathcal{D}'(x,t) = \mathcal{D}(x - b \cdot t, t)$. Dans le diagramme $\mathcal{D}$, après une durée Δt, la position x_1 d'un signal μ situé en x_0 au temps t_0 est donnée par $x_1 = \Delta t \cdot \mathcal{S}(\mu) + x_0$ (tant qu'il n'entre pas en collision). Dans le diagramme $\mathcal{D}'$, sa nouvelle position est donnée par : $x'_1 = \Delta t \cdot (\mathcal{S}(\mu) + b) + x_0 = \Delta t \cdot \mathcal{S}(\mu) + x_0 + \Delta t \cdot b = x_1 + \Delta t \cdot b$ et on a donc bien $c'(x,t) = c(x - b \cdot t, t)$. Dans le cas d'une collision se produisant aux coordonnées (x,t) entre deux signaux μ_1 et μ_2 (si la collision implique plus de deux signaux entrants, on peut ramener le système d'équations décrivant le mouvement des signaux concernés à une seule équation concernant l'intersection des trajectoires de seulement deux de ces signaux), on sait qu'il existe un temps t_0 tel que t est solution de l'équation $(t - t_0) \cdot \mathcal{S}(\mu_1) + x_1 = (t - t_0) \cdot \mathcal{S}(\mu_2) + x_2$, où x_1 (respectivement x_2) est la position

spatiale de μ_1 (respectivement μ_2) au temps t_0. Ajouter b aux vitesses de μ_1 et μ_2 ne change pas l'équation (car le terme $(t - t_0) \cdot b = \Delta t \cdot b$ apparaît dans les deux membres de l'égalité) donc le temps t_1 solution de l'équation est le même dans le diagramme $\mathcal{D}'$. La position de la collision dans le diagramme $\mathcal{D}'$ est donnée par $x' = (t - t_0) \cdot (\mathcal{S}(\mu_1) + b) + x_1 = x + \Delta t \cdot b$, et on obtient également dans le cas d'une collision $\mathcal{D}'(x, t) = \mathcal{D}(x - b \cdot t, t)$. Comme les positions de tous les signaux et de toutes les collisions sont déviées uniformément, la position d'une accumulation sera également déviée de la même façon : si $\mathcal{D}(x, t) = ✺$ alors $\mathcal{D}'(x + b \cdot t, t) = ✺$. En effet, si l'accumulation est statique, les valeurs s'accumulant dans l'espace verront leurs positions déviées de la même façon et donc la position de l'accumulation sera également déviée. Si l'accumulation est dynamique, son cône de lumière est dévié en même temps que toutes les positions contenues dans celui-ci : son passé causal contiendra donc encore une infinité de collisions.

On montre de la même façon que multiplier toutes les vitesses par la constante positive a modifie toutes les dates mais conserve toutes les positions spatiales (car $a > 0$). On a pour tout $(x, t) \in \mathbb{R} \times \mathbb{R}^+$, $\mathcal{D}'(x, t) = \mathcal{D}(x, a \cdot t)$.

Finalement, appliquer f aux vitesses est équivalent à les multiplier par a et à leur ajouter b et donc pour tout $(x, t) \in \mathbb{R} \times \mathbb{R}^+$, $\mathcal{D}'(x, t) = \mathcal{D}(x - b \cdot t, a \cdot t)$. En particulier, les collisions simultanées, les signaux s'intersectant et les signaux parallèles dans $\mathcal{D}$ le sont encore dans $\mathcal{D}'$.

La fonction $h : \mathbb{R}\times\mathbb{R}^+ \to \mathbb{R}\times\mathbb{R}^+$ définie par $h(x, t) = (x-b\cdot t, a\cdot t)$ est un homéomorphisme (les deux composantes de h sont continues et la bijectivité se vérifie facilement). On a donc montré qu'il existe un homéomorphisme h tel que pour tout $(x, t) \in \mathbb{R} \times \mathbb{R}^+$, $\mathcal{D}'(x, t) = \mathcal{D}(h(x, t))$, c'est-à-dire d'après la Déf. 10 que $\mathcal{D}$ et $\mathcal{D}'$ sont équivalents. □

Normalisations des vitesses. Étant donnés deux réels c et d tels que $c < d$, ce lemme permet de transformer toute machine $\mathfrak{M}$ dont les vitesses incluent les valeurs a et b (avec $a < b$) en une machine équivalente $\mathfrak{M}_f$ telle que les vitesses de $\mathfrak{M}_f$ incluent les valeurs c et d. En effet, la fonction $f : \mathbb{R} \to \mathbb{R}$ définie par $f(x) = \frac{d-c}{b-a} \cdot x + \frac{cb-ad}{b-a}$ est une fonction affine de coefficient strictement positif (car $c < d$ et $a < b$) qui vérifie $f(a) = c$ et $f(b) = d$. Le Lemme 1 nous assure alors que $\mathfrak{M}$ et $\mathfrak{M}_f$ génère des diagrammes espace-temps équivalents.

Par exemple, toute machine à signaux ayant au moins deux vitesses différentes, peut être transformée en une machine équivalente dont l'ensemble des vitesses inclut les valeurs 0 et 1 (ou tout autre paire de nombres réels distincts).

Transformations des configurations initiales. Le calcul est également préservé lorsqu'on applique des transformations affines aux configurations initiales :

Lemme 2 (Invariance du calcul par transformation affine des configurations initiales)
Soit $f : \mathbb{R} \to \mathbb{R}$ une fonction affine de ratio strictement positif et $\mathcal{D}$ un diagramme engendré par une machine $\mathfrak{M}$ à partir d'une configuration initiale c_0. Alors le diagramme $\mathcal{D}'$ engendré par $\mathfrak{M}$ à partir de la configuration c'_0 définie par $c'_0(x) = c_0(f(x))$ pour tout $x \in \mathbb{R}$, est équivalent à $\mathcal{D}$.

Démonstration. Si $f(x) = a \cdot x + b$ ($a > 0$) pour tout $x \in \mathbb{R}$, alors on montre de la même façon que pour le Lem. 1 que pour tout $(x, t) \in \mathbb{R} \times \mathbb{R}^+$, on a $\mathcal{D}'(x, t) = \mathcal{D}(x - b, \frac{1}{a} \cdot t)$. On en déduit directement que les diagrammes $\mathcal{D}$ et $\mathcal{D}'$ sont équivalents. □

2.4.2 Supports et inclusions de diagrammes

On introduit ici des notions permettant de décrire et de comparer les structures des diagrammes, en ne considérant que les positions de valeur non vide, et ce, indépendamment de la valeur exacte de la position du moment qu'elle est non vide.

Notion de supports. Afin d'étudier et de « borner » la structure des diagrammes de machines à signaux, nous définissons une notion de *support*, à la fois pour les machines et les diagrammes. Intuitivement, la *machine support* $\widehat{\mathfrak{M}}$ de la machine à signaux $\mathfrak{M}$ est définie à partir de l'ensemble des vitesses différentes de $\mathfrak{M}$. Les règles de collisions sont définies pour produire le maximum de signaux en sortie, de telle sorte que toute règle de $\mathfrak{M}$ soit « incluse » dans une règle de la machine support.

On appelle *machine à signaux à n vitesses* toute machine ayant exactement n valeurs distinctes pour les vitesses des méta-signaux la composant, *i.e.* telle que $card(\mathcal{I}m(\mathcal{S})) = n$.

Soit $\mathfrak{M} = (\mathcal{M}, \mathcal{S}, \mathcal{R})$ une machine à signaux. On définit sur $\mathcal{M}$ une relation binaire $\sim$ telle que pour tous $\mu, \sigma \in \mathcal{M}$, $\mu \sim \sigma \Leftrightarrow \mathcal{S}(\mu) = \mathcal{S}(\sigma)$. La relation $\sim$ est clairement une relation d'équivalence. Une classe d'équivalence pour $\sim$ contient exactement tous les méta-signaux de $\mathcal{M}$ ayant la même vitesse. $\mathcal{M}$ étant fini, chaque classe d'équivalence l'est aussi. On choisit un système de représentants $\{\mu_i\}_{1 \leq i \leq n}$, où n est le nombre de classes c'est-à-dire le nombre de vitesses distinctes de $\mathfrak{M}$. On écrira $[\mu]_\sim$ pour désigner la classe d'équivalence du méta-signal μ.

Définition 14 (Machine support)
Soit $\mathfrak{M} = (\mathcal{M}, \mathcal{S}, \mathcal{R})$ une machine à signaux. La machine support $\widehat{\mathfrak{M}}$ de $\mathfrak{M}$ *est la machine à signaux $\widehat{\mathfrak{M}} = (\mathcal{M}', \mathcal{S}', \mathcal{R}')$ telle que :*

(i) $\mathcal{M}' = \mathcal{M}/\sim$;

(ii) $\mathcal{S}' : \mathcal{M}' \to \mathbb{R}$ *définie par* $\mathcal{S}'([\mu]_\sim) = \mathcal{S}(\mu)$;

(iii) $\mathcal{R}' = \{\ \rho^- \to \rho^+ |\ \rho^- \subseteq \mathcal{M}', card(\rho^-) \geq 2$ *et* $\rho^+ = \mathcal{M}'\ \}$.

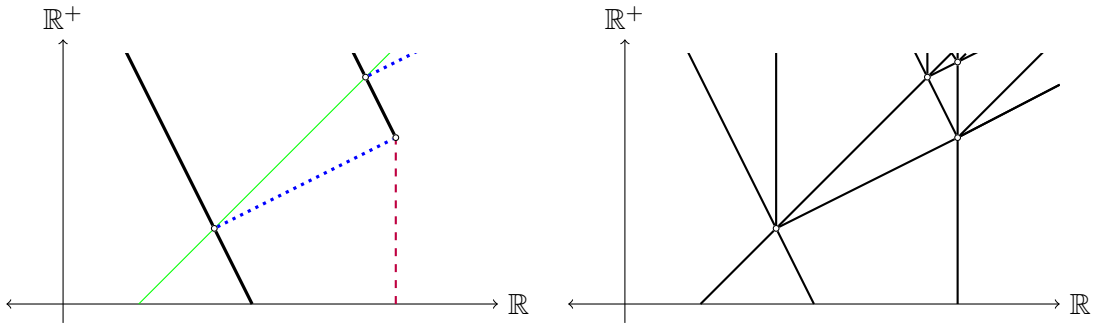
Pour chaque valeur distincte de vitesse, on choisit un unique méta-signal ayant cette vitesse dans la machine originale $\mathfrak{M}$. L'ensemble des règles de collisions est défini de la façon suivante. Pour chaque collision possible, l'ensemble des signaux sortants est $\mathcal{M}'$ (l'ensemble de tous les méta-signaux). Cette définition est possible car tous les méta-signaux dans $\mathcal{M}'$ ont des vitesses deux à deux distinctes.

On étend ensuite la surjection canonique $\mu \mapsto [\mu]_\sim$ en une surjection $\Pi : \mathcal{R} \to \mathcal{R}'$. Pour tout $\rho = \{\mu_i^-\}_{i \in I} \to \{\mu_j^+\}_{j \in J} \in \mathcal{R}$, on définit $\Pi(\rho) = \rho' \in \mathcal{R}'$ par $\rho' = \{[\mu_i^-]_\sim\} \to \mathcal{M}'$ (l'ensemble des méta-signaux sortants de C' est l'ensemble de tous les méta-signaux $\mathcal{M}'$, et ρ' appartient bien à $\mathcal{R}'$ définie ci-dessus). On étend également cette surjection aux configurations : étant donnée une configuration c de la machine $\mathfrak{M}$, on note $\widehat{c}$ la configuration c dans laquelle chaque signal $\mathcal{M}$ (resp. chaque collision C) est remplacé par un signal $[\mu]_\sim$ (resp. par une collision $\Pi(C)$), toutes les positions restant inchangées. Clairement, $\widehat{c}$ est une configuration pour la machine $\widehat{\mathfrak{M}}$. On peut maintenant définir la notion *diagramme espace-temps support* :

Définition 15 (Diagramme support)
Soit $\mathcal{D}$ diagramme espace-temps de la machine à signaux $\mathfrak{M}$ généré à partir de la configuration initiale c_0. On définit $\widehat{\mathcal{D}}$ le diagramme support du diagramme $\mathcal{D}$ *comme le diagramme espace-temps généré par $\widehat{\mathfrak{M}}$ à partir de la configuration initiale $\widehat{c_0}$.*

Le diagramme support du diagramme utilisé dans les exemples précédents est donné en Fig. 2.10. Attention à bien distinguer cette notion de *diagramme support* (resp. configuration support) de celle de *support d'un diagramme* (resp. support d'une configuration) : un diagramme support d'un diagramme est un diagramme $\widehat{\mathcal{D}}$ alors que le support d'un diagramme est un ensemble (donné par $support(\mathcal{D}) = \{(x,t) \in \mathbb{R} \times \mathbb{R}^+ \mid \mathcal{D}(x,t) \neq \oslash\}$).

La notion de machine support sera particulièrement utile pour étudier les accumulations, notamment parce qu'intuitivement, les accumulations apparaissent plus facilement lorsque les collisions génèrent beaucoup de signaux, et dans le cas d'une machine support, tous les signaux possibles sont générés à chaque collision.



(a) Un diagramme espace-temps. (b) Son diagramme support.

FIGURE 2.10 – Le diagramme support du diagramme de la FIG. 2.8.

Notions d'inclusion. On définit ici une notion d'inclusion de configurations et de diagrammes. Cette notion sera utilisée conjointement avec celles de configurations et diagrammes supports pour étudier au CHAP. 4 la structure des diagrammes et des accumulations n'impliquant qu'un faible nombre de vitesses différentes.

DÉFINITION 16 (Inclusion de diagrammes)
Soient $\mathcal{D}$ et $\mathcal{D}'$ deux diagrammes de machines, respectivement définis sur $\mathbb{R} \times [0;T]$ et $\mathbb{R} \times [0;T']$. On dit que $\mathcal{D}'$ est inclus dans le diagramme *$\mathcal{D}$ (ou que $\mathcal{D}'$* est supporté par *$\mathcal{D}$) si $support(\mathcal{D}') \cap \mathbb{R} \times [0; inf(T,T')] \subseteq support(\mathcal{D}) \cap \mathbb{R} \times [0; inf(T,T')]$.*

On notera alors $\mathcal{D}' \subseteq \mathcal{D}$. La restriction des supports à $\mathbb{R} \times [0; inf(T,T')]$ est nécessaire afin de comparer les supports des diagrammes sur la même portion d'espace-temps. On définit de la même façon la notion d'*inclusion de configurations* : une configuration c' est incluse dans la configuration c si $support(c') \subseteq support(c)$.

On remarquera que ces définitions imposent uniquement que toute position de valeur non vide pour le premier diagramme (resp. la première configuration) soit également non vide pour le deuxième diagramme (resp. la deuxième configuration). Elle n'impose pas que la valeur prise soit exactement la même pour les deux diagrammes (resp. pour les deux configurations [2]).

Remarque 7. Lorsque deux diagrammes $\mathcal{D}$ et $\mathcal{D}'$ sont équivalents, l'homéomorphisme h tel que $\forall (x,t) \in \mathbb{R} \times \mathbb{R}^+$, $\mathcal{D}'(x,y) = \Phi(\mathcal{D}(h(x,y)))$ (et où Φ est définie dans DÉF. 10) fournit également le lien entre les supports de $\mathcal{D}$ et $\mathcal{D}'$: on a $support(\mathcal{D}) = h(support(\mathcal{D}'))$. En effet, pour tout (x,y), on a $\mathcal{D}'(x,y) \neq \oslash$ si et seulement si $\Phi(\mathcal{D}(h(x,y))) \neq \oslash$. Comme Φ est une bijection telle que $\Phi(\oslash) = \oslash$, on en déduit que $\mathcal{D}'(x,y) \neq \oslash$ si et seulement si $\mathcal{D}(h(x,y)) \neq \oslash$. On a donc $(x,t) \in support(\mathcal{D}')$ si et seulement si $h(x,t) \in support(\mathcal{D})$, c'est-à-dire $support(\mathcal{D}) = h(support(\mathcal{D}'))$.

Nous montrons maintenant que les diagrammes supports « bornent » les structures des diagrammes dans le sens où tout diagramme est inclus dans son diagramme support :

LEMME 3
Pour tout diagramme $\mathcal{D}$, on a $\mathcal{D} \subseteq \widehat{\mathcal{D}}$.

Démonstration. Soient $\mathcal{D}$ et $\widehat{\mathcal{D}}$ respectivement définis sur $\mathbb{R} \times [0;T]$ et $\mathbb{R} \times [0;T']$. Passer au diagramme support ne supprime aucun objet présent dans le diagramme initial mais peut seulement ajouter de nouveaux objets. Pour chaque signal (resp. collision) présent au point (x,t) dans $\mathcal{D} \cap \mathbb{R} \times [0; inf(T,T')]$, il existe un signal (resp. collision) au point (x,t) dans $\widehat{\mathcal{D}} \cap \mathbb{R} \times [0; inf(T,T')]$. En effet, pour une collision $C = \{\mu_i^-\}_{i \in I} \to \{\mu_j^+\}_{j \in J}$ telle que $\mathcal{D}(x,t) = C$,

2. Ce cas fera l'objet de la DÉF. 18 dans la sous-section suivante.

on a $\widehat{\mathcal{D}}(x,t) = \Pi(C) = \{[\mu_i^-]_\sim\}_{i\in I} \to \mathcal{M}'$. $\Pi(C)$ aura les mêmes coordonnées (x,t) que C, mais elle produira plus de signaux, correspondant à tous les méta-signaux de la machine support. Pour un signal μ tel que $\mathcal{D}(x,t) = \mu$, soit $t = 0$ et on a $\widehat{\mathcal{D}}(x,t) = \widehat{\mathcal{D}}(x,0) = [\mu]_\sim$, soit μ a été crée dans une collision C en (x_0,t_0). Dans ce dernier cas, on a $\widehat{\mathcal{D}}(x_0,t_0) = \Pi(C)$ et l'ensemble des signaux sortants de $\Pi(C)$ est $\mathcal{M}'$, donc $\Pi(C)$ génère en particulier le signal $[\mu]_\sim$. Comme μ et $[\mu]_\sim$ sont tous les deux générés au point (x_0,t_0) de leurs diagrammes respectifs et que leurs vitesses sont égales par définition de $[\mu]_\sim$, ils ont exactement les mêmes équations de mouvement et passent par les mêmes positions spatiales aux mêmes moments. En particulier $\widehat{\mathcal{D}}(x,t) = [\mu]_\sim$. Donc dans les deux cas, si $\mathcal{D}(x,t) = \mu$ alors $\widehat{\mathcal{D}}(x,t) = [\mu]_\sim$. Supposons que $\mathcal{D}$ contienne une accumulation au point (x,t): $\mathcal{D}(x,t) = ✺$. Il y a une infinité de signaux et collisions présents dans $\Gamma^-(x,t)$ (le passé causal de (x,t) in $\mathcal{D}$). Comme pour chaque signal (resp. chaque collision) présent dans $\Gamma^-(x,t)$, il existe un signal (resp. une collision) ayant exactement la même position dans le diagramme support $\widehat{\mathcal{D}}$, il y aura également une infinité de signaux et collisions dans $\widehat{\Gamma^-}(x,t)$, le passé causal de (x,t) dans $\widehat{\mathcal{D}}$. On a donc $\widehat{\mathcal{D}}(x,t) = ✺$.

Finalement, on a $\forall(x,t)\ \mathcal{D}(x,t) \neq \oslash \Rightarrow \widehat{\mathcal{D}}(x,t) \neq \oslash$ c'est-à-dire $support(\mathcal{D}) \subseteq support(\widehat{\mathcal{D}})$. □

Comme nous n'avons pas défini de règles de continuation des diagrammes après les accumulations, la condition sur la restriction des supports à $\mathbb{R} \times [0; inf(T,T')]$ dans la DÉF. 16 est particulièrement importante dans le cas des diagrammes supports. En effet, les diagrammes supports, par leur définition, ont tendance à produire rapidement et beaucoup d'accumulations, et ils peuvent notamment en contenir bien avant la première accumulation des diagrammes dont ils sont les supports. Le LEMME 3 peut être généralisé en ajoutant dans la définition des diagrammes supports qu'une accumulation produit tous les signaux possibles. Dans ce cas, le support d'un diagramme $\mathcal{D}$ est intégralement inclus (sans restriction sur les intervalles de temps) dans celui de son diagramme support *i.e.* $support(\mathcal{D}) \subseteq support(\widehat{\mathcal{D}})$.

De manière plus générale, on peut montrer que la relation d'inclusion de diagramme préserve l'existence d'accumulation :

LEMME 4
Soient $\mathcal{D}$ et $\mathcal{D}'$ deux diagrammes espace-temps tels que $\mathcal{D}' \subseteq \mathcal{D}$.
Alors : $\forall(x,t) \in \mathbb{R}\times\mathbb{R}^+\ \mathcal{D}(x,t) \neq ✺ \Rightarrow \forall(x,t) \in \mathbb{R}\times\mathbb{R}^+\ \mathcal{D}'(x,t) \neq ✺$.

C'est-à-dire : si $\mathcal{D}'$ est inclus dans $\mathcal{D}$ et si $\mathcal{D}$ ne contient pas d'accumulation, alors $\mathcal{D}'$ ne contient pas non plus d'accumulation.

Démonstration. On a $support(\mathcal{D}') \subseteq support(\mathcal{D})$. Comme $\mathcal{D}$ ne contient pas d'accumulation, on a $\forall(x,t) \in \mathbb{R}\times\mathbb{R}^+$, $\mathcal{D}(x,t) \neq ✺$. Pour tout (x,t) et pour tout voisinage $\mathcal{V}_{(x,t)} \subseteq \mathbb{R}\times\mathbb{R}^+$, il n'y a qu'un nombre fini de positions de valeur non vide dans $\mathcal{V}_{(x,t)}$, c'est-à-dire que $\mathcal{V}_{(x,t)} \cap support(\mathcal{D})$ est fini. Comme $support(\mathcal{D}') \subseteq support(\mathcal{D})$, on en déduit que $\mathcal{V}_{(x,t)} \cap support(\mathcal{D}')$ est fini aussi. Donc pour tout $(x,t) \in \mathbb{R}\times\mathbb{R}^+$, $\mathcal{D}'(x,t) \neq ✺$ *i.e.* $\mathcal{D}'$ ne contient pas d'accumulation. □

Remarque 8. Le LEMME 4 implique en fait que le nombre d'accumulations dans $\mathcal{D}$ est au moins égal au nombre d'accumulations dans $\mathcal{D}'$. Mais $\mathcal{D}$ peut contenir strictement plus d'accumulations que $\mathcal{D}'$.

La combinaisons des LEM. 3 et 4 implique le corollaire suivant :

COROLLAIRE 5
Soit $\mathcal{D}$ un diagramme espace-temps d'une machine $\mathfrak{M}$. Si $\widehat{\mathcal{D}}$ ne contient pas d'accumulation, alors $\mathcal{D}$ n'en contient pas non plus.

2.4.3 Opérations sur les configurations

Nous introduisons des concepts permettant de ramener l'étude d'une configuration à celle d'une ou plusieurs autres configurations. Nous définissons pour cela diverses opérations sur les configurations.

Normalisations de configurations. Il est possible d'appliquer diverses opérations aux configurations sans en altérer la structure c'est-à-dire sans altérer l'ordre des signaux et leurs distances relatives. De la même façon qu'appliquer une fonction affine aux vitesses ne change pas la structure des diagrammes espace-temps correspondants, appliquer une fonction affine à une configuration préserve également le diagramme généré sur cette configuration, tel que démontré en Lem. 2. L'utilisation de fonctions affines permet de définir des *normalisations de configuration* :

Définition 17 (Normalisation d'une configuration)
On définit $c^{\hookrightarrow [a;b]}$ *la* normalisation de la configuration c dans l'intervalle $[a;b]$ *en posant pour tout* $x \in \mathbb{R}$ *:* $c^{\hookrightarrow [a;b]}(x) = c(\frac{b-a}{x_{sup}-x_{inf}} \cdot x + \frac{ax_{sup}-bx_{inf}}{x_{sup}-x_{inf}})$*, et où* x_{sup} *et* x_{inf} *désignent respectivement la borne supérieure et la borne inférieure de la configuration* c*.*
On définit $c^{trans(b)}$ *la* translation par b de la configuration c *par* $c^{trans(b)}(x) = c(x-b)$ *pour tout* $x \in \mathbb{R}$*.*

La normalisation d'une configuration dans un intervalle permet donc de ramener la partie significative d'une configuration à un intervalle donné, tout en conservant les rapports de distances (les distances absolues ne sont pas préservées). La translation d'une configuration est en fait un cas particulier de normalisation qui permet de ramener la première valeur non vide d'une configuration à une position fixée. Par exemple, on pourra ainsi considérer des configurations initiales avec un premier signal en position 0. Lors de la normalisation d'une configuration dans un intervalle $[a;b]$, les positions a et b seront de valeurs non vide si et seulement si les bornes supérieure et inférieure de c sont non vides (et peuvent donc être remplacées par x_{max} et x_{min}).

Restrictions et sous-configurations. Il est parfois plus approprié de raisonner sur des portions de configurations ou bien sur des positions ayant certaines valeurs dans une configuration. Nous définissons pour cela les notions de *restrictions* et de *sous-configuration* d'une configuration :

Définition 18
Soit c *une configuration à valeurs dans* $\mathbb{V}$*. On définit la* restriction de c à un intervalle $I \subseteq \mathbb{R}$ *par la restriction classique i.e.* $c_{\restriction I} : I \to \mathbb{V}$ $\quad x \mapsto c(x)$ *.*
On définit la restriction de c à l'ensemble de valeurs $\mathbb{V}' \subseteq \mathbb{V}$ *comme la restriction de l'ensemble image à* $\mathbb{V}'$ *i.e.* $c_{\restriction \mathbb{V}'} : R \to \mathbb{V}'$ *.*

$$x \mapsto \begin{cases} c(x) & si\ c(x) \in \mathbb{V}' \\ \oslash & sinon \end{cases}$$

On dit que c' *est une* sous-configuration de c *si* $support(c') \subseteq support(c)$ *et pour tout* $x \in support(c')$ *on a* $c'(x) = c(x)$*. On notera dans ce cas* $c' \sqsubseteq c$*.*
On dit que c' *est une* sous-configuration connexe de c *si* $c' \sqsubseteq c$ *et s'il n'existe pas de* x *dans la partie significative de* c' *tel que* $x \in support(c) \setminus support(c')$*.*

Unions de configurations. Il est possible de définir de nouvelles configurations à partir d'opérations d'union de configurations. Nous proposons ici de telles opérations pour superposer,

concaténer et joindre deux configurations. De manière intuitive, ces opérations d'unions correspondent respectivement à une union simple, une union bout-à-bout et une union bout-à-bout avec identification des deux extrémités mises bout-à-bout.

Définition 19 (Superposition de configuration)
Soient c et c' deux configurations respectivement à valeurs dans $\mathbb{V}$ et $\mathbb{V}'$. On suppose que pour tout $x \in support(c) \cap support(c')$, on a $c(x) = c'(x)$. On définit $c \cup c'$, la superposition (ou union) des configurations c et c', *par* $c \cup c' : \mathbb{R} \rightarrow \mathbb{V} \cup \mathbb{V}'$.

$$x \mapsto \begin{cases} c(x) & si\ x \in support(c) \\ c'(x) & sinon \end{cases}$$

On définit la *concaténation de configurations à distance ε* par l'union bout-à-bout des configurations telle que le début de la deuxième configuration est situé à une distance ε de la fin de la première configuration :

Définition 20 (Concaténation de configurations)
Soient c et c' deux configurations respectivement à valeurs dans $\mathbb{V}$ et $\mathbb{V}'$. Soit $\varepsilon > 0$.
On définit $c^{\frown_\varepsilon}c'$ la concaténation de c et c' à distance ε *par :*
$c^{\frown_\varepsilon}c' : \mathbb{R} \rightarrow \mathbb{V} \cup \mathbb{V}'$.

$$x \mapsto \begin{cases} c(x) & si\ x \in support(c) \\ c'(x') & si\ x = x^c_{sup} + \varepsilon + x' - x^{c'}_{inf},\ avec\ x' \in support(c') \\ \oslash & sinon \end{cases}$$

Dans une concaténation, lorsque la position maximale de la première configuration et la position minimale de la deuxième sont bien définies et que leurs valeurs concordent, il est possible d'opérer une concaténation à distance $\varepsilon = 0$, fusionnant ainsi les deux extrémités concernées. Nous parlons dans ce cas de *jonction de configurations* :

Définition 21 (Jonction de configurations)
Soient c et c' deux configurations respectivement à valeurs dans $\mathbb{V}$ et $\mathbb{V}'$. On suppose que x^c_{max} et $x^{c'}_{min}$ sont définis et vérifient $c(x_{max}) = c'(x_{min})$. On définit $c^{\urcorner}c'$ la jonction de c et c' par :
$c^{\urcorner}c' : \mathbb{R} \rightarrow \mathbb{V} \cup \mathbb{V}'$

$$x \mapsto \begin{cases} c(x) & si\ x \in support(c) \\ c'(x') & si\ x = x^c_{sup} + x' - x^{c'}_{inf},\ avec\ x' \in support(c') \\ \oslash & sinon \end{cases}$$

On parlera également de jonction suivant une valeur : celle donnée par $c(x_{max}) = c'(x_{min})$. La superposition de configurations suppose une origine commune aux configurations (les positions spatiales de même valeurs sont superposées). Au contraire, la concaténation et la jonction de configurations n'impose pas une origine commune, et ces opérations correspondent en fait à une superposition de configurations après que la deuxième configuration a subi une translation automatique (translation de valeur $x^c_{sup} + \varepsilon + x' - x^{c'}_{inf}$).

Chapitre 3

Algorithmique géométrique

Nous introduisons dans ce chapitre des définitions afin de formaliser dans le cadre des machines à signaux les notions de codages d'entrée/sortie, de calcul, de résolution de problème et de complexité. Ces définitions complètent celles du Chap. 2, donnant ainsi tous les outils nécessaires pour parler d'*algorithmique géométrique*.

Nous commencerons par quelques exemples simples d'algorithmes géométriques, qui nous serviront à introduire de manière intuitive la notion de calcul par signaux et quelques méthodes pour représenter des données avec des signaux et pour prouver la correction des constructions. Nous formaliserons ensuite ces notions : d'abord celle de codages de valeurs discrètes et continues, puis celle de *calcul géométrique abstrait* et enfin celle de résolution de problèmes sur machines à signaux, que l'on distinguera en deux sous-concepts, celui de *résolution spécifique* et celui de *résolution générique*. Nous définirons également des relations entre configurations, afin de les manipuler plus facilement en tant que représentations des entrées/sorties sur machines à signaux. Enfin, après une brève discussion sur les différentes façons de mesurer la complexité d'un calcul géométrique abstrait, nous définirons deux nouvelles mesures de complexité d'un diagramme de machine à signaux. Ces nouvelles mesures prenant en compte les spécificités du modèle et ses primitives de calcul, se nomment la *profondeur de collisions* et la *largeur de collisions*. Elles correspondent respectivement aux mesures en temps et en espace du calcul, et se définissent comme les longueurs d'une chaîne maximale et d'une anti-chaîne maximale.

3.1 Exemples de calculs géométriques

Afin d'illustrer les définitions du Chap. 2 et de donner l'intuition des concepts d'algorithmique géométrique, nous proposons ici une étude de trois exemples simples de calculs géométriques : celui du milieu d'un segment, celui de la soustraction et celui d'une addition binaire. Nous montrons comment interpréter les diagrammes correspondants en termes de calcul, et donnons les preuves de corrections correspondantes.

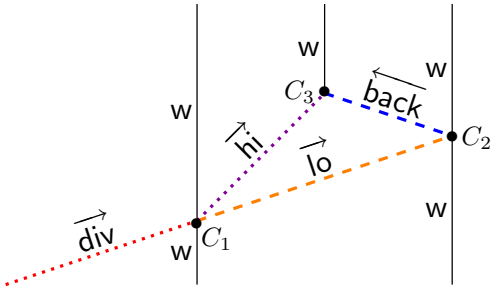
3.1.1 Calcul géométrique du milieu

Nous montrons ici comment la position exacte du milieu d'un segment réel peut être calculé par une machine à signaux simple. La machine $\mathfrak{M}_{mid}$ utilisée est définie par les Fig. 3.1(a) et 3.1(c) ; la Fig. 3.1(b) fournit l'exemple d'un diagramme représentant un calcul géométrique du milieu.

On se donne comme configuration initiale $c_0 = \{\overrightarrow{\mathsf{div}}@p, \mathsf{w}@a,\ \mathsf{w}@b\ \}$, où $p < a < b$. Le segment $[a; b]$ est directement encodé dans cette configuration : il est marqué par les deux signaux w. Décrivons de manière intuitive l'évolution de la machine $\mathfrak{M}_{mid}$ à partir de la configuration c_0. Après la première collision, les signaux $\overrightarrow{\mathsf{hi}}$ et $\overrightarrow{\mathsf{lo}}$ se propagent en direction du signal w de

Méta-signal	Vitesse
w	0
$\overrightarrow{\mathsf{div}}$	3
$\overrightarrow{\mathsf{hi}}$	1
$\overrightarrow{\mathsf{lo}}$	3
$\overleftarrow{\mathsf{back}}$	−3

(a) Méta-signaux.



(b) Diagramme espace-temps.

$\{\mathsf{w}, \overrightarrow{\mathsf{div}}\} \to \{\mathsf{w}, \overrightarrow{\mathsf{hi}}, \overrightarrow{\mathsf{lo}}\}$
$\{\overrightarrow{\mathsf{lo}}, \mathsf{w}\} \to \{\overleftarrow{\mathsf{back}}, \mathsf{w}\}$
$\{\overrightarrow{\mathsf{hi}}, \overleftarrow{\mathsf{back}}\} \to \{\mathsf{w}\}$

(c) Règles de collisions.

FIGURE 3.1 – La machine $\mathfrak{M}_{mid}$: calcul géométrique du milieu d'un segment.

droite. Or, par le choix des vitesses, $\overrightarrow{\mathsf{lo}}$ se déplace trois fois plus vite que $\overrightarrow{\mathsf{hi}}$. Donc le temps que $\overrightarrow{\mathsf{lo}}$ parcourt une distance d, $\overrightarrow{\mathsf{lo}}$, devenu $\overleftarrow{\mathsf{back}}$ après le rebond sur w, aura parcouru une distance trois fois plus grande et égale à $3d$. La distance parcourue par $\overleftarrow{\mathsf{back}}$ est donnée par $3d-(b-a)$ et à la rencontre avec $\overrightarrow{\mathsf{hi}}$, on aura $d = 3d-(b-a)$, et donc $d = \frac{b-a}{2}$, c'est-à-dire que la collision entre $\overrightarrow{\mathsf{hi}}$ et $\overleftarrow{\mathsf{back}}$ se produit exactement au milieu du segment $[a;b]$.

Nous allons maintenant démontrer de manière calculatoire et formelle (en utilisant les équations de mouvements des signaux) qu'à partir de la configuration initiale c_0, on obtient bien une configuration finale telle que le milieu du segment $[a;b]$ est marqué par un signal w. Pour cela, nous calculons l'historique des collisions, en calculant les coordonnées de la prochaine collision puis en recommençant à partir de la configuration obtenue en appliquant la règle correspondante à cette collision.

Évolution de la machine $\mathfrak{M}_{mid}$. Comme les deux signaux w sont parallèles, la première collision a forcément lieu entre $\overrightarrow{\mathsf{div}}$ (qui se déplace vers la droite car étant de vitesse positive) et le premier signal w. Les coordonnées (x_1, t_1) de la collision C_1 sont solutions du système d'équations linéaires suivant, donné par les équations des mouvements des signaux $\overrightarrow{\mathsf{div}}$ et w donnant leur position x au temps t :

$$\begin{cases} x &= 3t+p & \textit{(pour } \overrightarrow{\mathsf{div}}\textit{)} \\ x &= a & \textit{(pour le premier } \mathsf{w}\textit{)} \end{cases}.$$

Le temps t_1 de la première collision est donc $t_1 = \frac{a-p}{3}$. La position correspondante x_1 est trivialement donnée par la position stationnaire de w: $x_1 = a$. En appliquant la règle $\{\mathsf{w}, \overrightarrow{\mathsf{div}}\} \to \{\mathsf{w}, \overrightarrow{\mathsf{hi}}, \overrightarrow{\mathsf{lo}}\}$, on obtient au temps t_1 la configuration $c_{t_1} = \{\mathsf{w}@a, \overrightarrow{\mathsf{hi}}@a, \overrightarrow{\mathsf{lo}}@a, \mathsf{w}@b\}$. Les signaux $\overrightarrow{\mathsf{hi}}$ et $\overrightarrow{\mathsf{lo}}$ se déplacent tous deux vers la droite en direction du signal stationnaire w mais comme $\overrightarrow{\mathsf{lo}}$ (de vitesse 3) est plus rapide que $\overrightarrow{\mathsf{hi}}$ (de vitesse 1), la collision suivante aura lieu entre $\overrightarrow{\mathsf{lo}}$ et le signal w de droite.

En appliquant le même calcul que ci-dessus, on déduit des équations de mouvement de $\overrightarrow{\mathsf{lo}}$ et w de droite, respectivement données par $x = 3(t-t_1)+a$ et $x = b$, que les coordonnées de la deuxième collision C_2 sont $t_2 = \frac{b-p}{3}$ et $x_2 = b$. Après application de la règle $\{\overrightarrow{\mathsf{lo}}, \mathsf{w}\} \to \{\overleftarrow{\mathsf{back}}, \mathsf{w}\}$, la nouvelle configuration est $c_{t_2} = \{\mathsf{w}@a, \overrightarrow{\mathsf{hi}}@\frac{b+2a}{3}, \overleftarrow{\mathsf{back}}@b, \mathsf{w}@b\}$: en effet, la nouvelle position du signal $\overrightarrow{\mathsf{hi}}$ qui se trouvait en $x = a$ au temps t_1 est donnée par $x = (t_2-t_1)+a = \frac{b-p}{3} - \frac{a-p}{3} + a = \frac{b-a}{3} + a = \frac{b+2a}{3}$.

De même, à partir de la configuration c_{t_2} et des mouvements des signaux $\overrightarrow{\mathsf{hi}}$ et $\overleftarrow{\mathsf{back}}$ respectivement donnés par $x = (t - t_1) + a$ et $x = -3(t - t_2) + b$, on en déduit que la date de la troisième collision C_3 est $t_3 = \frac{3b-a-2p}{6}$ et sa position est $x_3 = \frac{a+b}{2}$.

En appliquant la dernière règle de collisions $\{\overrightarrow{\mathsf{hi}}, \overleftarrow{\mathsf{back}}\} \to \{\mathsf{w}\}$ à la collision C_3, on obtient la configuration $c_{t_3} = \{\mathsf{w}@a,\ \mathsf{w}@\frac{a+b}{2},\ \mathsf{w}@b\}$. La position du signal w intermédiaire est $\frac{a+b}{2}$ et correspond donc bien au milieu du segment $[a; b]$. Comme seuls des signaux w apparaissent dans cette dernière configuration, plus aucune collision ne se produit car tous les signaux sont parallèles.

Dans cet exemple, la fin du calcul peut être définie de deux façons : soit par une durée d'évolution de la machine fixée en avance (la durée $\frac{3b-a-2p}{6}$), soit par l'apparition d'une configuration stationnaire (c_{t_3}), qui ne permet plus aucune dynamique dans la machine.

Remarque 9. Le signal $\overrightarrow{\mathsf{div}}$ n'est utilisé que pour démarrer le processus, et n'est pas indispensable au calcul du milieu proprement dit. Il est possible de s'en passer en considérant comme configuration initiale la configuration obtenue après la collision entre $\overrightarrow{\mathsf{div}}$ et w et donnée par $c = \{\ [\mathsf{w}, \overrightarrow{\mathsf{hi}}, \overrightarrow{\mathsf{lo}}]@a,\ \mathsf{w}@b\ \}$ (et égale à la configuration c_{t_1} précédemment décrite). Dans ce cas, la durée du calcul (*i.e.* la hauteur de la construction) est $\frac{b-a}{2}$.

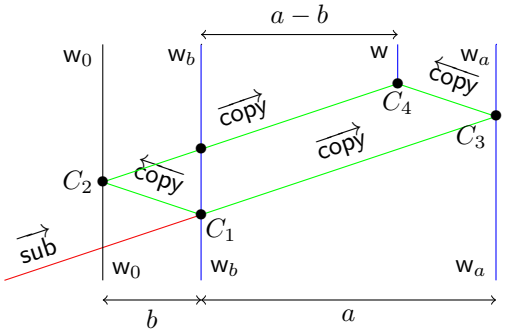
Cet algorithme géométrique de calcul du milieu fournit également une machine $\mathfrak{M}_{div}$ qui permet de calculer la moitié d'une grandeur réelle positive. En effet, la distance d entre les deux premiers signaux de la configuration finale c_{t_3} est égale à $d = \left|a - \frac{a+b}{2}\right| = \frac{b-a}{2}$, c'est-à-dire la moitié de la distance entre les deux signaux w de la configuration initiale.

3.1.2 Algorithme de soustraction et de calcul du modulo

Nous donnons ici un autre exemple d'algorithme géométrique simple : la soustraction de deux valeurs réelles a et b. Comme en SOUS-SEC. 3.1.1, nous montrons ici qu'en encodant deux réels a et b dans la configuration $c_0 = \left\{\overrightarrow{\mathsf{sub}}@s,\ \mathsf{w}_0@x_0,\ \mathsf{w}_b@x_b,\ \mathsf{w}_a@x_a\right\}$, avec $s < x_0 < x_b < x_a$ et telle que $a = x_a - x_b$ et $b = x_b - x_0$ l'évolution de la machine à signaux $\mathfrak{M}_{sub}$, dont les méta-signaux et règles sont donnés par les FIG. 3.2(a) et 3.2(c), aboutit à une configuration finale de la forme $c = \{\mathsf{w}_0@x_0,\ \mathsf{w}_b@x_b,\ \mathsf{w}@x,\ \mathsf{w}_a@x_a\}$, qui encode le résultat de la soustraction $a - b$ par la distance entre w_b et w, c'est-à-dire $x - x_b = a - b$. Une telle évolution est illustrée par le diagramme espace-temps de la FIG. 3.2(b).

Méta-signal	Vitesse
$\mathsf{w}_0,\ \mathsf{w}_a,\ \mathsf{w}_b$	0
$\overrightarrow{\mathsf{sub}},\ \overrightarrow{\mathsf{copy}}$	3
$\overleftarrow{\mathsf{copy}}$	-3

(a) Méta-signaux.



(b) Diagramme espace-temps.

$\{\overrightarrow{\mathsf{sub}},\ \mathsf{w}_b\} \to \{\overleftarrow{\mathsf{copy}},\ \mathsf{w}_b,\ \overrightarrow{\mathsf{copy}}\}$
$\{\mathsf{w}_0,\ \overleftarrow{\mathsf{copy}}\} \to \{\mathsf{w}_0,\ \overrightarrow{\mathsf{copy}}\}$
$\{\overrightarrow{\mathsf{copy}},\ \mathsf{w}_a\} \to \{\overleftarrow{\mathsf{copy}},\ \mathsf{w}_a\}$
$\{\overrightarrow{\mathsf{copy}},\ \overleftarrow{\mathsf{copy}}\} \to \{\mathsf{w}\}$

(c) Règles de collisions.

FIGURE 3.2 – La machine $\mathfrak{M}_{sub}$: soustraction géométrique.

Évolution de la machine $\mathfrak{M}_{sub}$. Commençons par une remarque : comme pour l'exemple du calcul du milieu, le signal $\overrightarrow{\mathsf{sub}}$ est utilisé uniquement pour démarrer la soustraction avec plus

de clarté. Sa position exacte s n'est pas importante du moment que s est la position la plus à gauche. On ne considérera pas ici la toute première collision, se produisant entre $\overrightarrow{\mathsf{sub}}$ et w_0 (qui est une collision blanche), mais on prendra sa date comme instant initial.

Nous nous plaçons donc directement au niveau de la collision C_1 qui se produit entre $\overrightarrow{\mathsf{sub}}$ et w_b et dont les coordonnées (x_1, t_1) vérifient $x_1 = 3t_1 + x_0$ (mouvement de $\overrightarrow{\mathsf{sub}}$, en x_0 à l'instant $t_0 = 0$) et $x_1 = x_b$ (pour w_b). On en déduit facilement que $t_1 = \frac{x_b - x_0}{3}$ (et de manière triviale $x_1 = x_b$). L'application de la règle $\{\overrightarrow{\mathsf{sub}},\ \mathsf{w}_b\} \to \{\overleftarrow{\mathsf{copy}},\ \mathsf{w}_b,\ \overrightarrow{\mathsf{copy}}\}$ produit une nouvelle configuration, dont une étude rapide montre que les deux collisions suivantes C_2 et C_3 auront lieu respectivement entre le w_0 et $\overleftarrow{\mathsf{copy}}$ pour une collision et $\overrightarrow{\mathsf{copy}}$ et w_a pour l'autre. Les coordonnées (x_2, t_2) de la collision C_2 entre w_0 et $\overleftarrow{\mathsf{copy}}$ sont $x_2 = x_0$ et $t_2 = 2t_1 = 2\frac{x_b - x_0}{3}$. On calcule de même les coordonnées de la collision C_3 entre $\overrightarrow{\mathsf{copy}}$ et w_a: $t_3 = \frac{x_a - x_b}{3} + t_1 = \frac{x_a - x_0}{3}$, et $x_3 = x_a$.

Nous pouvons désormais en déduire les coordonnées (x_4, t_4) de la dernière collision C_4 se produisant entre $\overrightarrow{\mathsf{copy}}$ (produit par la collision C_2) et $\overleftarrow{\mathsf{copy}}$ (produit par C_3) (la règle de collisions entre $\overrightarrow{\mathsf{copy}}$ et w_b étant blanche, la collision entre $\overrightarrow{\mathsf{copy}}$ et $\overleftarrow{\mathsf{copy}}$ aura bien lieu). Les positions des signaux $\overrightarrow{\mathsf{copy}}$ et $\overleftarrow{\mathsf{copy}}$ étant respectivement données par $x = 3(t - t_2) + x_0$ et $x = -3(t - t_3) + x_a$, on obtient que $t_4 = \frac{1}{6}(3t_3 + 3t_2 + x_a - x_0) = \frac{x_a + x_b - 2x_0}{3}$ et $x_4 = -3(t_4 - t_3) + x_a = -(x_a + x_b - 2x_0) + (x_a - x_0) + x_a = x_a - x_b + x_0$.

Après application de la règle $\{\overrightarrow{\mathsf{copy}},\ \overleftarrow{\mathsf{copy}}\} \to \{\mathsf{w}\}$ correspondant à la collision C_4, on obtient la configuration stationnaire finale $c_{t_4} = \{\ \mathsf{w}_0@x_0,\ \mathsf{w}_b@x_b,\ \mathsf{w}@x_a - x_b + x_0,\ \mathsf{w}_a@x_a\ \}$. La distance y entre w_b et w est donc égale à $y = x_a - x_b + x_0 - x_b$, c'est-à-dire en reprenant le codage initial des réels a et b que $a = x_a - x_b$ et $b = x_b - x_0$, $y = a - b$.

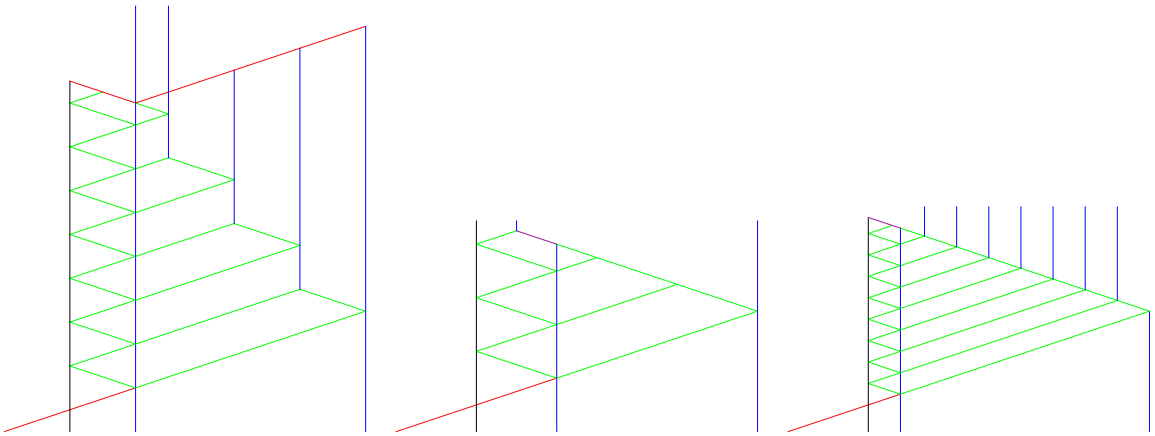
La largeur est trivialement donnée par $x_a - x_0 = (x_a - x_b) + (x_b - x_0) = a + b$. La hauteur de la construction, et donc la durée totale Δ, est donnée par $\Delta = t_4 = \frac{x_a + x_b - 2x_0}{3} = \frac{(x_a - x_b) + (x_b - x_0) + (x_b - x_0)}{3} = \frac{a + 2 \cdot b}{3}$. Le diagramme comprend en tout (et indépendamment des valeurs a et b) six collisions (dont deux blanches), quatre signaux en configuration initiale et quatre en configuration finale.

Remarquons que la hauteur de la construction dépend du choix des vitesses. Si nous avions choisi des vitesses -1 et 1 au lieu des vitesses -3 et 3, la construction aurait eu en vertu du LEM. 1 une hauteur trois fois plus grande et égale à $a + 2b$. Ce choix de vitesses permet donc d'avoir des diagrammes moins hauts et est justifié par les calculs du modulo et de la division euclidienne : ces constructions utilisent des soustractions successives et seront illustrées dans le prochain paragraphe. En modifiant la machine $\mathfrak{M}_{sub}$, il est également possible de concevoir une machine produisant des configurations de sortie ne contenant que deux signaux verticaux dont l'espacement encode le résultat de la soustraction.

Remarque 10. Il est parfois plus convenable de se contenter de preuves intuitives, et il est possible pour des petits exemples de déduire les propriétés recherchées à partir d'arguments simples. Par exemple, dans le diagramme de la FIG. 3.2(b) vu comme une figure géométrique dans le plan euclidien, les signaux $\overleftarrow{\mathsf{copy}}$ et $\overrightarrow{\mathsf{copy}}$ forment un parallélogramme, et les triangles formés par les signaux $\overleftarrow{\mathsf{copy}}$ (resp. $\overrightarrow{\mathsf{copy}}$), w_0 (resp. w_a) et l'horizontale sont isométriques. Cela permet de déduire directement que la distance entre w et w_a au sommet de la construction est égale à la distance entre w_0 et $\overrightarrow{\mathsf{copy}}$ à la base du diagramme. La distance entre w_b et w est donc bien égale à la différence entre a et b.

Le modulo et la division euclidienne. À partir de l'exemple de la soustraction, on obtient facilement le calcul du modulo et celui de la division euclidienne sur $\mathbb{R}$ (les notions de division euclidienne et modulo sur les réels seront précisées en SEC. 4.3). Par définition, le modulo peut se calculer en itérant des soustractions : il suffit donc de modifier la machine $\mathfrak{M}_{sub}$ pour qu'elle réitère la copie de la distance b grâce aux signaux $\overleftarrow{\mathsf{copy}}$ et $\overrightarrow{\mathsf{copy}}$. Le calcul se termine lorqu'il n'est plus possible de soustraire b : après nettoyage grâce à des signaux $\overleftarrow{\mathsf{clean}}$ et $\overrightarrow{\mathsf{clean}}$,

le résultat peut alors être représenté par l'espacement entre deux signaux verticaux, comme cela est illustré par la FIG. 3.3(a). Les règles correspondant à ces deux étapes (itération et arrêt) sont respectivement $\{\overrightarrow{\mathsf{copy}},\ \mathsf{w}_b\} \to \{\overleftarrow{\mathsf{copy}},\ \mathsf{w}_b,\ \overrightarrow{\mathsf{copy}}\}$ et $\{\mathsf{w}_b,\ \overleftarrow{\mathsf{copy}}\} \to \{\overleftarrow{\mathsf{clean}},\ \mathsf{w}_b,\ \overrightarrow{\mathsf{clean}}\}$. Il est également possible de concevoir une variante plus « compacte » (c'est-à-dire avec un diagramme moins haut tout en ayant les mêmes vitesses) : la FIG. 3.3(b) en fournit un exemple. De même, en itérant la soustraction et en émettant un signal vertical à chaque fois que l'on soustrait b à la valeur a, on obtient une construction géométrique calculant la division euclidienne et produisant le résultat codé en unaire, dont un exemple est donné par la FIG. 3.3(c).



(a) Calcul de 7/2 mod 1.

(b) Calcul de 5/2 mod 1 (version compacte).

(c) Division euclidienne de 31 par 4 avec résultat en unaire.

FIGURE 3.3 – Calculer le modulo et la division euclidienne à partir de la soustraction géométrique.

Les exemples des calculs géométriques du milieu et de la soustraction montrent comme il est peut être fastidieux de prouver de manière analytique (*i.e.* avec les équations de propagation des signaux) la correction de constructions géométriques pourtant simples. Nous allons illustrer dans la sous-section suivante comment la correction d'un algorithme géométrique de type combinatoire peut se réduire à une preuve combinatoire. Une méthode plus générale de preuve de correction sera fournie au CHAP. 6 : elle permettra entre autres de simplifier les preuves de constructions complexes en les ramenant à des preuves de fonctionnalités plus basiques, comme celles effectuées dans cette section.

3.1.3 Additionneur binaire géométrique

L'algorithme géométrique considéré ici est une construction permettant de calculer l'addition de deux nombres binaires. Cet algorithme fonctionne avec des valeurs discrètes (des entiers) fournies en entrée, et est donc de type combinatoire.

Les entiers n et m sont codés sous forme de deux faisceaux binaires $[\overrightarrow{bin}(n)]$ et $[\overleftarrow{bin}(m)]$ (respectivement en représentation *big endian* et *little endian*), et un signal $+^R$ placé entre ces deux faisceaux déclenche l'addition. L'addition bit à bit est alors portée par un signal d'addition zig-zaguant entre les deux faisceaux. La FIGURE 3.4 illustre le principe de fonctionnement de la machine $\mathfrak{M}_{add}$ en donnant le diagramme pour l'addition de 6 et 3.

LEMME 6 (Correction de l'additionneur binaire)
Soient n, m deux entiers. Le diagramme engendré par $\mathfrak{M}_{add}$ à partir de la configuration $c_0^{n,m} = \left\{ [\overrightarrow{bin}(n)] +^R [\overleftarrow{bin}(m)] \;\middle|\; (n,m) \in \mathbb{N} \times \mathbb{N} \right\}$, vérifie :

(i) il y a toujours au plus un signal de type $+$ entre les deux signaux end ;

(ii) le signal $+$ disparaît avant la collision entre $\overrightarrow{\mathsf{end}}$ et $\overleftarrow{\mathsf{end}}$;

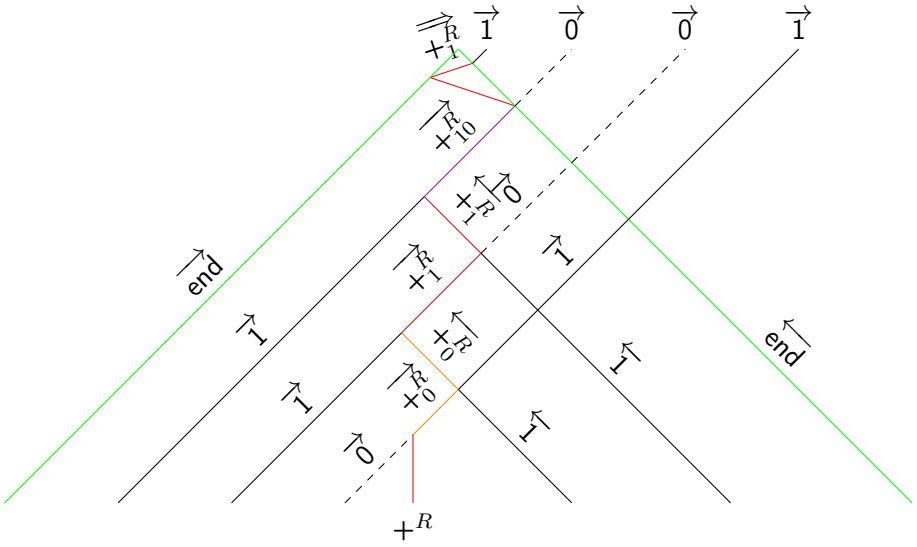


FIGURE 3.4 – Addition binaire géométrique de 6 et 3 par la machine $\mathfrak{M}_{add}$.

(iii) *si un signal + rencontre un bit gauche puis un droit, alors le bit sortant de la deuxième collision est la somme binaire (XOR) des deux bits correspondants et l'éventuelle retenue est propagée aux bits suivants ;*

(iv) *le faisceau obtenue en sortie est* $[\overrightarrow{bin}(n+m)]$.

Démonstration. Les différents points du lemme viennent de l'observation des règles de $\mathfrak{M}_{add}$ fournies par le TAB. 3.1.

(i) Seules les règles ayant un signal de type + en entrée peuvent produire un signal de type +, et en produisent au maximum un seul. Comme il n'y a qu'un seul signal de type + en configuration initiale, il y aura toujours au maximum un seul signal +. De plus, les règles entre les signaux end et + sont des règles de rebond, ce qui garantit que + reste toujours entre les signaux end.

(ii) Le nombre de bits rentrants diminue strictement à chaque collision avec +. Lorsqu'il ne reste plus de bits dans les faisceaux d'entrée, une règle fait disparaître + lors d'une collision avec end (sauf si un nouveau bit doit être créé par un signal $+^{\overrightarrow{\overrightarrow{R}}}_{10}$, auquel cas il disparaîtra après un aller-retour supplémentaire entre les deux signaux end).

(iii) Il suffit de constater que les règles du TAB. 3.1(c) suivent les règles d'une addition binaire classique. La retenue est portée par le signal de type $\overleftarrow{+}$ (qui est soit $+^{\overleftarrow{R}}_0$, soit $+^{\overleftarrow{R}}_1$).

(iv) On remarquera tout d'abord que les règles ne sont pas symétriques entre les versions droite et gauche des signaux. En effet, le résultat étant envoyé vers la droite, il est nécessaire pour additionner les deux bits correspondants de toujours commencer par le bit de droite. Le résultat de l'addition de ces deux bits sera émis lors de la collision entre + et le deuxième bit (le bit arrivant de gauche). Lorsque la toute première collision ne se produit pas entre + et un bit gauche, nous ramenons le début de l'addition vers le premier bit du faisceau gauche en utilisant le signal $\overleftarrow{\overleftarrow{+_{shift}}}$et les règles du TAB. 3.1(b), comme illustré par la FIG. 3.5(a). Le TABLEAU 3.1(b) fournit également les règles pour gérer le cas d'une triple collision initiale. Le point *(iv)* se montre par induction. Le point précédent constitue l'étape d'induction d'addition bit à bit. Il faut considérer maintenant ce qu'il se passe lorsqu'un faisceau a été épuisé et qu'il reste des bits dans l'autre. Lorsqu'un faisceau est épuisé, le signal de type + continue de zig-zaguer, mais entre les bits du faisceau restant et le signal end opposé. On

définit l'équivalent des règles d'addition de bits à bits, mais avec des rebonds sur end (avec l'éventuelle création d'un nouveau bit) : les règles utilisées sont données par le TAB. 3.1(d). En commençant par un bit de gauche, les bits sont successivement additionnés deux à deux, et le bit de sortie émis correspond bien aux deux bits à additionner. Lorsque le processus prend fin, il ne reste en sortie qu'un faisceau de bits et le signal $\overrightarrow{\mathsf{end}}$. La configuration finale est bien de la forme $[\overrightarrow{bin}(n+m)]$: l'entier $n+m$ est représenté en binaire en sortie.

□

(a) Méta-signaux et vitesses.

$\overrightarrow{0}$, $\overrightarrow{1}$, $\overrightarrow{\mathsf{end}}$	1
$\overleftarrow{0}$, $\overleftarrow{1}$, $\overleftarrow{\mathsf{end}}$, $\overleftarrow{\mathsf{end}}_R$	−1
$\overrightarrow{+}^R_0$, $\overrightarrow{+}^R_1$, $\overrightarrow{+}^R_{10}$	1
$\overleftarrow{+}^R_0$, $\overleftarrow{+}^R_1$, $\overleftarrow{+}^R_{10}$	−1
$\overrightarrow{\overrightarrow{+}}^R_0$, $\overrightarrow{\overrightarrow{+}}^R_1$, $\overrightarrow{\overrightarrow{+}}^R_{10}$	3
$\overleftarrow{\overleftarrow{+}}^R_0$, $\overleftarrow{\overleftarrow{+}}^R_1$, $\overleftarrow{\overleftarrow{+}}^R_{10}$	−3
$+^R$	0
$\overleftarrow{\overleftarrow{+}}_{shift}$	−3

(b) Démarrer l'addition.

$\{\overrightarrow{0}, +^R\} \to \{\overrightarrow{+}^R_0\}$
$\{\overrightarrow{1}, +^R\} \to \{\overrightarrow{+}^R_1\}$
$\{\overleftarrow{0}, +^R\} \to \{\overleftarrow{0}, \overleftarrow{\overleftarrow{+}}_{shift}\}$
$\{\overleftarrow{1}, +^R\} \to \{\overleftarrow{1}, \overleftarrow{\overleftarrow{+}}_{shift}\}$
$\{\overrightarrow{0}, \overleftarrow{\overleftarrow{+}}_{shift}\} \to \{\overrightarrow{+}^R_0\}$
$\{\overrightarrow{1}, \overleftarrow{\overleftarrow{+}}_{shift}\} \to \{\overrightarrow{+}^R_1\}$
$\{\overrightarrow{0}, +^R, \overleftarrow{0}\} \to \{\overleftarrow{+}^R_0, \overrightarrow{0}\}$
$\{\overrightarrow{0}, +^R, \overleftarrow{1}\} \to \{\overleftarrow{+}^R_0, \overrightarrow{1}\}$
$\{\overrightarrow{1}, +^R, \overleftarrow{0}\} \to \{\overleftarrow{+}^R_0, \overrightarrow{1}\}$
$\{\overrightarrow{1}, +^R, \overleftarrow{1}\} \to \{\overleftarrow{+}^R_1, \overrightarrow{1}\}$

(c) Additionner les bits.

$\{\overrightarrow{0}, \overleftarrow{+}^R_0\} \to \{\overrightarrow{+}^R_0\}$
$\{\overrightarrow{1}, \overleftarrow{+}^R_0\} \to \{\overrightarrow{+}^R_1\}$
$\{\overrightarrow{0}, \overleftarrow{+}^R_1\} \to \{\overrightarrow{+}^R_1\}$
$\{\overrightarrow{1}, \overleftarrow{+}^R_1\} \to \{\overrightarrow{+}^R_{10}\}$
$\{\overrightarrow{+}^R_0, \overleftarrow{0}\} \to \{\overleftarrow{+}^R_0, \overrightarrow{0}\}$
$\{\overrightarrow{+}^R_0, \overleftarrow{1}\} \to \{\overleftarrow{+}^R_0, \overrightarrow{1}\}$
$\{\overrightarrow{+}^R_1, \overleftarrow{0}\} \to \{\overleftarrow{+}^R_0, \overrightarrow{1}\}$
$\{\overrightarrow{+}^R_1, \overleftarrow{1}\} \to \{\overleftarrow{+}^R_1, \overrightarrow{0}\}$
$\{\overrightarrow{+}^R_{10}, \overleftarrow{0}\} \to \{\overleftarrow{+}^R_1, \overrightarrow{0}\}$
$\{\overrightarrow{+}^R_{10}, \overleftarrow{1}\} \to \{\overleftarrow{+}^R_1, \overrightarrow{1}\}$

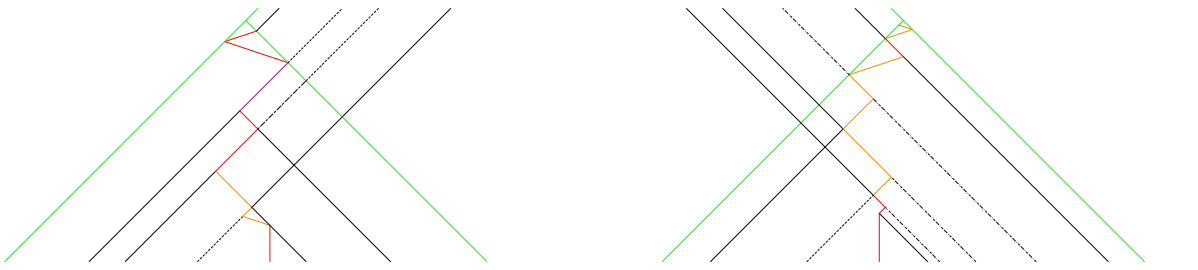
(d) Cas des bits seuls.

$\{\overrightarrow{+}^R_0, \overleftarrow{\mathsf{end}}\} \to \{\overleftarrow{\overleftarrow{+}}^R_0, \overleftarrow{\mathsf{end}}, \overrightarrow{0}\}$
$\{\overrightarrow{+}^R_1, \overleftarrow{\mathsf{end}}\} \to \{\overleftarrow{\overleftarrow{+}}^R_0, \overleftarrow{\mathsf{end}}, \overrightarrow{1}\}$
$\{\overrightarrow{+}^R_{10}, \overleftarrow{\mathsf{end}}\} \to \{\overleftarrow{\overleftarrow{+}}^R_1, \overleftarrow{\mathsf{end}}, \overrightarrow{0}\}$
$\{\overleftarrow{+}^R_0, \overrightarrow{\mathsf{end}}\} \to \{\overrightarrow{\mathsf{end}}, \overrightarrow{\overrightarrow{+}}^R_0\}$
$\{\overleftarrow{+}^R_1, \overrightarrow{\mathsf{end}}\} \to \{\overrightarrow{\mathsf{end}}, \overrightarrow{\overrightarrow{+}}^R_1\}$

(e) Finir l'addition.

$\{\overrightarrow{\overrightarrow{+}}^R_0, \overleftarrow{\mathsf{end}}\} \to \{\overleftarrow{\mathsf{end}}_R\}$
$\{\overrightarrow{\overrightarrow{+}}^R_1, \overleftarrow{\mathsf{end}}\} \to \{\overleftarrow{\mathsf{end}}_R, \overrightarrow{1}\}$
$\{\overrightarrow{\mathsf{end}}, \overleftarrow{\mathsf{end}}_R\} \to \{\overrightarrow{\mathsf{end}}\}$

TABLEAU 3.1 – La machine $\mathfrak{M}_{add}$: méta-signaux et règles.



(a) Calcul de $6+3$, avec recalage du signal initial.

(b) Calcul de $17+2$, avec résultat envoyé à gauche.

FIGURE 3.5 – Autres exemples d'additions binaires géométriques.

Tailles de la construction. Soit l la distance entre les deux signaux initiaux $\overrightarrow{\mathsf{end}}$ et $\overleftarrow{\mathsf{end}}$. Alors la hauteur h est donnée par $h = \frac{l}{2}$ (car $\overrightarrow{\mathsf{end}}$ et $\overleftarrow{\mathsf{end}}$ de vitesses 1 et −1 et tout est

fini après leur collision). Le nombre de collisions total est en $\mathcal{O}(\log_2(n) \cdot \log_2(m))$ et le plus grand nombre de collisions enchaînées est en $\mathcal{O}(max(\log_2(n), \log_2(m)))$. Le nombre maximal de signaux présents à un instant donné est égal au nombre de signaux présents en entrée c'est-à-dire $(\log_2(n)+1)+(\log_2(m)+1)+1$ (car aucune des règles de collisions ne créé de signaux en plus).

3.2 Calcul géométrique abstrait

Sur un modèle de calcul non conventionnel, les primitives de calcul et les objets manipulés sont différents de ceux utilisés en calculabilité classique, et il faut donc redéfinir les notions classiques de codages de données, de problème, de solution avant de pouvoir parler d'algorithmique intrinsèque au modèle. Nous introduisons ici plusieurs notions de résolution de problèmes par machines à signaux, ainsi que les notions en lien comme le codage/décodage des entrées et sorties, l'interprétation d'un calcul... afin de formaliser ce que l'on entend par « résoudre un problème avec des signaux ». Ces notions seront reprises et approfondies au CHAP. 6 dans un contexte particulier, celui des *modules géométriques*.

3.2.1 Représentations de l'information

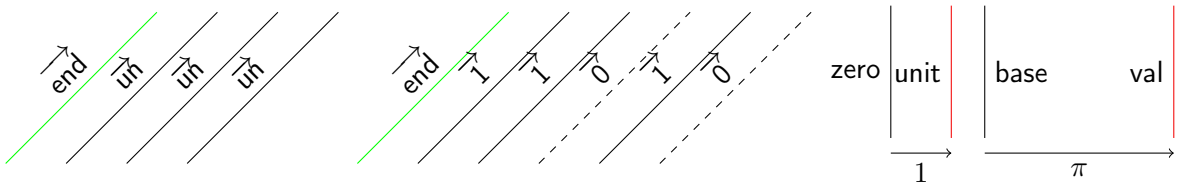
Dans le modèle des machines à signaux, les objets basiques sont les signaux et les collisions, et ce sont en fait les seuls objets disponibles. La représentation d'informations et de données doit donc se faire uniquement par ces objets. Il existe de nombreuses manières de coder des données par des signaux et des collisions. Nous en présentons ici quelques unes uniquement basées sur l'utilisation de signaux : une valeur sera codée par des signaux disposés dans une configuration selon un schéma donné. Nous différencions deux méthodes principales d'encodage, suivant le type de valeur à représenter. En effet, comme dans les modèles classiques, un réel ne se représente pas de la même façon qu'un entier.

Représentation de valeurs discrètes. Il est aisé de représenter des valeurs discrètes (finies) sur machines à signaux : il suffit d'utiliser des méta-signaux pour représenter des symboles. Ainsi, la représentation d'objets définis à partir d'un alphabet de taille k nécessitera k méta-signaux différents, plus éventuellement quelques signaux supplémentaires (par exemple pour marquer le début ou la fin de la représentation). De plus, il est important que ces méta-signaux soient de même vitesse, afin de garantir la stabilité du codage dans l'espace.

Exemple 1 : codage d'entiers en unaire.
Les entiers peuvent se représenter de manière unaire en utilisant deux méta-signaux : le méta-signal $\overrightarrow{\mathsf{un}}$ pour représenter le symbole « | » et le méta-signal $\overrightarrow{\mathsf{end}}$ pour marquer la fin de la représentation. Ces deux méta-signaux sont surmontés d'une flèche vers la droite, dénotant des vitesses positives : ils ont en fait la même vitesse, que l'on choisi égale à 1 pour simplifier. Ainsi, l'entier n que l'on représente habituellement par une concaténation de n symboles « | » (*i.e.* $||\ldots|$) se code sur le même principe avec des signaux et sera représenté par la suite de signaux ($\overrightarrow{\mathsf{end}}$ $\overrightarrow{\mathsf{un}}$ $\ldots$ $\overrightarrow{\mathsf{un}}$), que l'on notera également ($\overrightarrow{\mathsf{end}}$ $\overrightarrow{\mathsf{un}}^n$). La FIGURE 3.6(a) fournit la représentation de l'entier 3, avec des signaux régulièrement espacés et se propageant vers la droite.

Comme les méta-signaux $\overrightarrow{\mathsf{end}}$ et $\overrightarrow{\mathsf{un}}$ sont de vitesses 1, la séquence de signaux ainsi définie est uniquement constituée de signaux parallèles et se propageant vers la droite. Nous pouvons définir de la même façon une représentation unaire d'un entier dans laquelle l'information se propage vers la gauche. Nous ne nous intéressons pas pour le moment aux positions exactes de ces signaux dans l'espace, mais à la séquence qu'ils définissent, sous forme d'une configuration.

Exemple 2 : codage d'entiers en binaire.
Il est possible de définir sur le même principe que l'EX. 1 un codage binaire des entiers. La



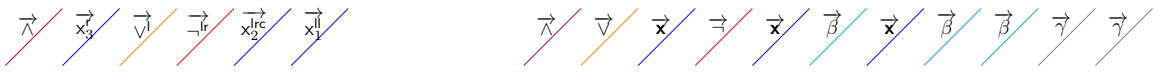
(a) Codage unaire de 3. (b) Codage binaire de 26. (c) Codage du réel π.

FIGURE 3.6 – Exemples de codage de valeurs numériques par signaux.

représentation binaire nécessite trois méta-signaux de même vitesses (deux pour les symboles binaires 0 et 1, et un signal pour marquer la début ou la fin du codage) : $\overrightarrow{0}$, $\overrightarrow{1}$ et $\overrightarrow{\mathsf{end}}$. Pour un entier n, les signaux seront disposés en suivant l'écriture binaire de n en représentation *big endian*, le bit de poids fort étant le plus proche du signal $\overrightarrow{\mathsf{end}}$. La FIGURE 3.6(b) fournit un exemple de représentation binaire de l'entier 26, encodé par la séquence de signaux (disposés les uns à la suite des autres, de gauche à droite) ($\overrightarrow{\mathsf{end}}$ $\overrightarrow{1}$ $\overrightarrow{1}$ $\overrightarrow{0}$ $\overrightarrow{1}$ $\overrightarrow{0}$). Les bits de poids fort sont aux débuts de la séquence, ceux de poids faibles à la fin, c'est-à-dire tout à droite. De manière générale, si l'écriture binaire d'un entier n est donnée par $b_{\lfloor \log_2(n) \rfloor} \dots b_1 b_0 = \sum_{i=0}^{i=\lfloor \log(n) \rfloor} b_i \cdot 2^i$ avec $b_i \in \{0, 1\}$, alors n sera représenté par la suite de signaux : $[\overrightarrow{bin}(n)] = (\overrightarrow{\mathsf{end}}\ \overrightarrow{\sigma_{\lfloor \log_2(n) \rfloor}} \dots \overrightarrow{\sigma_1}\ \overrightarrow{\sigma_0})$ où $\overrightarrow{\sigma_i}$ est un signal $\overrightarrow{1}$ (resp. $\overrightarrow{0}$) si $b_i = 1$ (resp. $b_i = 0$). La position exacte de chaque signal n'est pas importante (cependant deux signaux ne peuvent avoir la même position au même instant, puisque les signaux utilisés ici sont tous de même vitesse), seul l'ordre des signaux est important. On définit de la même façon un faisceau binaire se déplaçant vers la gauche (*i.e.* de vitesse -1), correspondant à une représentation *little endian*.

Il est possible d'envisager de multiples variantes de ces encodages d'entiers : nous pouvons « encapsuler » complètement l'encodage entre deux signaux de type $\overrightarrow{\mathsf{end}}$ ou au contraire s'en passer si l'on impose un espacement régulier entre les signaux (la fin de l'encodage étant alors marqué par l'absence de signal à la prochaine position).

D'autres types d'encodages existent pour représenter des valeurs discrètes non numériques : la FIG. 3.7 donne l'exemple de deux méthodes différentes de codage d'une formule booléenne. La première méthode (FIG. 3.7(a)) consiste à utiliser des signaux spécifiques à l'objet à encoder, tandis que la deuxième méthode (FIG. 3.7(b)) permet, pour un type d'objet donné, de représenter n'importe lequel de ces objets à partir d'une famille finie de signaux fixés. Ces deux méthodes, respectivement qualifiées de *spécifique* et *générique*, seront détaillées et utilisées dans le cas de formules booléennes dans les CHAP. 7 et 8.



(a) Codage spécifique de φ. (b) Codage générique de φ.

FIGURE 3.7 – Exemples de codages d'une formule booléenne : cas de $\varphi = (x_1 \vee \neg x_2) \wedge x_3$.

Représentation de valeurs continues. Les machines à signaux formant un modèle de calcul à temps et espace continus, elles permettent naturellement de manipuler des valeurs réelles ou continues. En tant qu'ensemble de nombres réels, l'espace utilisé ($\mathbb{R}$) contient déjà implicitement des valeurs réelles. Il faudra néanmoins fixer des normes pour un encodage de valeurs continues :

elles pourront par exemple être directement représentées par des positions de signaux donnés, ou bien être assimilées à des distances entres signaux.

Exemple 3 : codage de nombre réels.
Comme illustré par la FIG. 3.6(c), un réel peut se représenter très simplement par la distance entre signaux donnés : dans l'exemple, le nombre π est représenté par la distance (relative) entre les signaux base à val. Les réels négatifs peuvent également être représentés de cette façon, soit en considérant la mesure algébrique de la distance base à val (le réel encodé sera négatif si val est à gauche de base), soit en imposant que base soit toujours à gauche et en utilisant des signaux val^+ et val^- pour indiquer le signe du réel encodé.

L'espace ne disposant pas d'échelle absolue, il faut fixer une unité de mesure, en utilisant par exemple deux signaux zero et unit dont l'espacement fournira la longueur unité. On peut également déduire l'unité de mesure à partir de l'encodage d'une première valeur.

L'exemple de la FIG. 3.6(c) est donné avec des signaux stationnaires, mais il est possible de considérer des signaux de vitesses non nulles (les signaux utilisés doivent néanmoins être parallèles).

Remarque 11. Remarquons qu'il est également possible de représenter des données à l'aide des règles de collisions ou des vitesses des signaux. Nous verrons au CHAP. 7 comment coder des formules booléennes avec des signaux et des règles de collisions (dont les définitions suivront la structure syntaxique de la formule) et au CHAP. 8 comment le faire en utilisant uniquement des signaux. De même, les vitesses étant des nombres réels, il est possible de stocker un nombre r réel ou rationnel en assignant la valeur r comme vitesse d'un méta-signal donné. Le nombre r peut alors être récupéré dans un calcul en tant que distance : ce sera la distance parcourue par le signal de vitesse r pendant une unité de temps. Un tel procédé sera utilisé au CHAP. 4 pour créer une distance irrationnelle à partir d'un méta-signal de vitesse irrationnelle et d'une configuration rationnelle.

Faisceaux de signaux. Les trois exemples de codage donnés en FIG. 3.6 partagent une même structure commune, bien que la nature des objets qu'ils encodent soit différente (discrète et continue). Tous peuvent être vu comme des *faisceaux* propageant une information dans l'espace. Les valeurs y sont codées par un ensemble de signaux tous parallèles c'est-à-dire de même vitesse : vitesse 1 pour FIG. 3.6(a) et FIG. 3.6(b) et vitesse 0 pour FIG. 3.6(c). Malgré leur apparente simplicité (que des signaux parallèles, pas de collisions, dynamique très simple), les faisceaux permettent de représenter l'information et de la propager de manière homogène et sans altération dans l'espace.

On appellera *configuration-faisceau* toute configuration $c : \mathbb{R} \to \mathbb{V}$ qui ne contient que des signaux, tous de même vitesse : $\mathcal{I}m(support(c)) \subseteq \mathcal{M}$ et il existe $\nu \in \mathbb{R}$ tel que pour tout $x \in support(c)$, on a $\mathcal{S}(c(x)) = \nu$. On désignera par *faisceau* soit une configuration-faisceau, soit l'évolution d'une configuration-faisceau jusqu'à une éventuelle altération du faisceau. Notons qu'un faisceau peut également être constitué d'un unique signal.

Taille des représentations. On remarque que dans les EX. 1 et 2, le nombre de signaux constituant le faisceau dépend de la valeur à coder : il faudra $n + 1$ signaux pour coder l'entier n en unaire et $\lfloor \log_2(n) \rfloor + 2$ en binaire. La distance séparant deux signaux successifs ne joue aucun rôle dans le codage de la valeur, et peut même être rendue aussi petite que l'on veut (du moment qu'elle reste strictement positive). Pour le codage d'un réel tel qu'illustré par FIG. 3.6(c), il n'y a toujours que deux signaux (sans compter les deux signaux fixés pour définir une unité de distance), et ce, indépendamment du réel x à coder. Par contre, la distance entre ces deux signaux dépend de x, puisque c'est justement cette distance qui encode x de manière directe.

Ce constat permet de différencier deux types de *mesures de la représentation d'information* : pour des valeurs discrètes ou entières, le nombre d'objets utilisés donnera une mesure de la taille du codage, alors que la taille du codage de valeurs continues ou réelles sera donnée par des distances ou des grandeurs spatiales.

3.2.2 Calculs géométriques abstraits

Un diagramme espace-temps $\mathcal{D}$ peut être interprété comme un calcul de multiples façons. Afin d'assimiler un diagramme à un calcul, il convient de préciser l'interprétation considérée.

Un *calcul géométrique abstrait (CGA)* est constitué d'un diagramme espace-temps généré par une machine à signaux, une condition de fin de calcul et une fonction d'interprétation du résultat.

Définition 22 (Calcul géométrique abstrait)
Un calcul géométrique abstrait *(CGA) est un triplet* $\gamma = (\mathcal{D}, \Phi_H, \mathcal{I})$ *tel que :*

(i) $\mathcal{D}$ *est un diagramme espace-temps ;*

(ii) Φ_H *est une condition de fin de calcul donnée par une fonction définie de l'ensemble des configurations de* $\mathcal{D}$ *dans l'ensemble* $\{\mathsf{vrai}, \mathsf{faux}\}$*;*

(iii) $\mathcal{I}$ *est une fonction d'interprétation définie de l'ensemble des configurations vérifiant* Φ_H *(i.e. les configurations* c *telles que* $\Phi_H(c) = \mathsf{vrai}$*) dans un ensemble de valeurs, appelé ensemble des résultats du calcul.*

L'interprétation du diagramme sera donnée par l'interprétation d'une configuration, et non par une interprétation du diagramme global. Le diagramme est vu ici comme une succession de configurations, la fin et le résultat du calcul étant donnés par la « première » configuration qui vérifie la condition d'arrêt : la fonction $\mathcal{I}$ nous fournit alors l'interprétation du résultat.

Plusieurs types de condition d'arrêt peuvent être envisagés : durée d'évolution fixée à l'avance, apparition d'un signal donné (ou d'une règle prédéfinie), plus d'évolution possible (présence de signaux tous parallèles ou de signaux se propageant dans des directions opposées), apparition d'un cycle de configurations, *etc.*

Un même diagramme peut donner lieu à plusieurs interprétations différentes en termes de calcul, ce que nous illustrerons plus loin. En effet, changer la condition d'arrêt et/ou la fonction d'interprétation donnera en général des résultats qui peuvent s'avérer très différents. On pourra également définir une notion de *temps d'arrêt* $\mathcal{T}_H$ à partir de la borne inférieure des temps t tels que $\Phi_H(c_t)$ est vraie.

Cette définition est très générale, et ne dit rien sur la façon d'obtenir les diagrammes espace-temps ou sur les fonctions d'interprétations. De plus, les calculs géométriques abstraits ainsi définis ne correspondent pas forcément au calcul d'une fonction donnée ou à la résolution d'un problème. Nous allons maintenant formaliser ces CGA dans le cadre de la résolution de problème : les CGA considérés dépendront de l'instance d'entrée du problème considéré.

3.2.3 Résolution de problèmes par machines à signaux

Nous formalisons ici la notion d'*algorithme géométrique* : l'idée est d'associer un calcul géométrique abstrait à chaque instance d'un problème.

Avant de définir ce que nous entendons par « résoudre géométriquement un problème » dans le contexte des machines à signaux, nous formalisons la notion de représentation d'un ensemble de valeurs dans une machine à signaux :

Définition 23
La représentation (ou codage) d'un ensemble X dans la machine à signaux $\mathfrak{M}$ *est un ensemble* $\mathcal{C}onfig$ *de configurations de* $\mathfrak{M}$ *tel qu'il existe une fonction calculable injective* $R : X \to \mathcal{C}onfig$*.*

En général, la fonction R sera simple : elle sera donnée par des schémas ou des algorithmes polynomiaux, et même très souvent linéaires c'est-à-dire que la configuration représentant une entrée sera générée en temps polynomial en la taille de l'entrée. Notons qu'il n'est possible d'encoder au maximum que des ensembles X de cardinal inférieur au continu $2^{\aleph_0}$ c'est-à-dire des ensembles X tels que $card(X) \leq 2^{\aleph_0}$. En effet, comme $card(\mathcal{C}onfig) = 2^{\aleph_0}$ (car une configuration est une fonction de $\mathbb{R}$ dans un ensemble fini), la condition d'injection de X dans $\mathcal{C}onfig$ implique que $card(X) \leq card(\mathcal{C}onfig)$.

Lors de la résolution d'un problème algorithmique par machines à signaux, il convient de séparer deux parties bien distinctes : d'un côté, une partie « algorithmique classique » (définie dans un modèle de calcul classique comme les machines de Turing), et d'un autre côté, une partie « algorithmique géométrique » qui constitue la partie intrinsèque au modèle des machines à signaux. La partie classique est constituée par les fonctions de codages d'une instance du problème dans une machine à signaux et une configuration initiale. Ces fonctions seront également appelées *fonctions de représentation* (ou *fonction de codage*) des entrées. La partie intrinsèque au modèle correspond à l'évolution à partir de la configuration initiale de la machine à signaux utilisée, qui résulte en un diagramme espace-temps correspondant à l'instance considérée. L'interprétation de ce diagramme se fera via une *fonction d'interprétation* (ou *fonction de décodage*).

La *résolution d'un problème par machines à signaux*, ou *algorithme géométrique pour un problème*, sera donc constituée par ces deux parties (classique et intrinsèque). Elle sera donnée d'une part par les algorithmes générant la machine et la configuration initiale à partir de l'entrée du problème, et d'autre part par l'exécution de cette machine à signaux munie de la configuration initiale correspondante.

Résolution par machine unique et par famille de machines. Nous distinguons ici deux cas de résolutions géométriques : soit la machine générée dépend de l'instance du problème, soit elle est indépendante, et dans ce cas, seule la configuration initiale dépend de l'instance du problème. On rappelle que $\mathfrak{M}(c_0)$ désigne le diagramme complet (*i.e.* maximal) obtenu par l'évolution de la machine $\mathfrak{M}$ à partir de la configuration c_0.

Définition 24 (Résolution spécifique)
On dit qu'un problème $\mathbf{P}$ *est* résoluble par une famille de machines à signaux *s'il existe :*

(i) une fonction calculable injective $\mathsf{représentation}$ *définie de l'ensemble des instances de* $\mathbf{P}$ *dans l'ensemble des couples* $(\mathfrak{M}, c_0)$ *(où* $\mathfrak{M}$ *est une machine et* c_0 *est une configuration pour* $\mathfrak{M}$*) telle que* $x \mapsto \mathsf{représentation}(x) = (\mathfrak{M}^x, c_0^x)$ *;*

(ii) une fonction calculable surjective $\mathsf{décodage}$ *définie de l'ensemble des diagrammes de* $\mathfrak{M}$ *dans l'ensemble des réponses pour* $\mathbf{P}$*,*

telles que pour toute instance x *du problème* $\mathbf{P}$*, on a* $\mathbf{P}(x) = \mathsf{décodage}(\mathfrak{M}^x(c_0^x))$.

On dira également que le couple $(\mathsf{représentation}, \mathsf{décodage})$ est une *résolution spécifique* du problème $\mathbf{P}$. La fonction $\mathsf{représentation}$ peut donc se décomposer en deux fonctions : une qui associe une machine à l'instance x, et une autre qui code x dans une configuration initiale. À chaque instance est donc associé un couple (machine, configuration initiale), et la résolution du problème est donc donnée par une famille $\{(\mathfrak{M}^x, c_0^x)\}_{x \in \{instances\ de\ \mathbf{P}\}}$. Cette famille définit la famille $\{\mathfrak{M}^x(c_0^x)\}_{x \in \{instances\ de\ \mathbf{P}\}}$ des diagrammes espace-temps obtenus par évolution de la machine $\mathfrak{M}^x$ sur la configuration c_0^x. Munie de la fonction $\mathsf{décodage}$, elle constitue alors la famille de CGA associée au problème $\mathbf{P}$.

Il est parfois possible d'utiliser la même machine pour toutes les instances du problème $\mathbf{P}$: dans ce cas, la fonction de représentation de la machine est constante, et seule la configuration initiale dépend de l'instance du problème. La fonction $\mathsf{représentation}$ se réduit alors à la fonction de codage de la configuration initiale. On parlera dans ce cas de résolution générique :

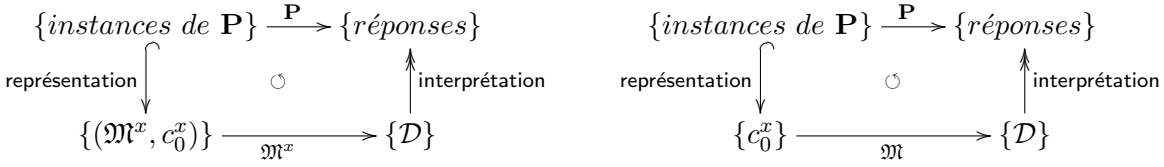
Définition 25 (Résolution générique)
On dit qu'un problème $\mathbf{P}$ *est* résoluble par une machine à signaux *s'il existe :*

(i) une fonction calculable injective représentation *et une machine à signaux* $\mathfrak{M}$ *telles que* $x \mapsto$ représentation$(x) = (\mathfrak{M}, c_0^x)$ *;*

(ii) une fonction calculable surjective décodage *définie de l'ensemble des diagrammes de* $\mathfrak{M}$ *dans l'ensemble des réponses pour* $\mathbf{P}$*,*

telles que pour toute instance x *du problème* $\mathbf{P}$*, on a* $\mathbf{P}(x) =$ décodage$(\mathfrak{M}(c_0^x))$.

Pour insister sur le fait qu'une seule machine est utilisée, on peut alors réécrire le couple (représentation, décodage) en un triplet ($\mathfrak{M}$, représentation, décodage), où la fonction représentation est uniquement à valeurs dans $\mathcal{C}onfig$. La Figure 3.8 montre les diagrammes correspondant aux deux types de résolution de problèmes par machines à signaux. La commutativité de ces diagrammes provient des conditions données dans les deux définitions, à savoir $\mathbf{P}(x) =$ décodage$(\mathfrak{M}^x(c_0^x)$ et $\mathbf{P}(x) =$ décodage$(\mathfrak{M}(c_0^x))$.



(a) Résolution spécifique de $\mathbf{P}$. (b) Résolution générique de $\mathbf{P}$.

Figure 3.8 – Les deux types de résolution d'un problème $\mathbf{P}$ sur machines à signaux.

La résolution spécifique considérée ici se rapproche de la notion de résolution de problème par circuits booléens [Vollmer 1999] : une solution par circuits booléens est donnée par une famille de circuits (un circuit pour chaque taille d'entrée possible). En général, les familles considérées sont *uniformes* : la famille peut être générée par une machine de Turing en temps polynomial c'est-à-dire qu'il existe une machine de Turing fournissant la description du circuit pour les entrées de taille n en temps polynomial en la taille de n. Nous considérerons ici aussi des familles uniformes, mais à la différence des circuits booléens, une machine à signaux sera définie pour chaque instance (et pas non seulement pour chaque taille d'entrée possible).

La résolution générique suit plus l'intuition de solution algorithmique à un problème : dans les modèles classiques, l'algorithme ne dépend jamais de l'entrée, et fonctionne pour les toutes les instances du problème.

Dans les définitions précédentes, on impose que les fonctions représentation et décodage soient calculables : en fait, ces fonctions ne doivent pas être trop compliquées. Elles ne doivent pas en effet cacher la difficulté du problème dans le codage des entrées et des sorties. De même, pour l'étude des complexités d'algorithmes géométriques, les fonctions devront être de complexité moindre que celle du problème considéré.

Nous utiliserons une présentation légèrement différente au Chap. 6, en préférant des fonctions surjectives de décodages des entrées et de sorties, que nous appellerons fonctions de *représentations des entrées et des sorties.*

Correction d'algorithmes géométriques. Un algorithme géométrique sera correct pour la résolution du problème $\mathbf{P}$ si l'une des Déf. 24 ou 25 est vérifiée. Une preuve de correction de cet algorithme géométrique sera donc donnée par une preuve que pour toute instance x, on a bien $P(x) =$ décodage$(\mathfrak{M}^x(c_0^x))$.

Il existe diverses façons de prouver la correction d'un algorithme géométrique. Pour des exemples simples, on peut se convaincre de manière informelle par un argument géométrique

naïf, sans toutefois tomber dans le « ça se voit sur le dessin ». Par exemple, l'utilisation de signaux de même vitesse fait apparaître des parallélogrammes, lesquels permettent de voir que des durées ou des distances entre signaux sont les mêmes.

Les preuves que l'on peut naturellement utiliser sont « analytiques » et suivent le principe de la dynamique énoncé dans la DÉF. 13 : l'historique de l'évolution de la machine est calculé à partir des équations de propagation des signaux et des règles de collisions. À partir d'une configuration donnée, la résolution de systèmes d'équations linéaires (dont les équations décrivent les positions futures de chaque signal présent dans la configuration en fonction de sa vitesse et de sa position actuelle) permet d'obtenir les coordonnées de la prochaine collision. Puis ce calcul est réitéré à partir de la configuration obtenue en mettant à jour la machine au temps de collision ainsi calculé. Cette méthode permet de connaître l'historique complet des collisions, mais est déjà longue et fastidieuse même pour des exemples simples.

Il est parfois possible de raisonner sans tenir compte de la valeur exacte des positions, mais juste avec l'ordre des signaux et des collisions : c'est notamment le cas pour des diagrammes correspondant à des calculs de types discrets. Dans ce cas, il est possible de formuler une preuve combinatoire de la correction du diagramme : nous en avons déjà fourni un exemple dans la SOUS-SEC. 3.1.3.

Nous introduirons dans le CHAP. 6 une autre méthode de preuve de correction d'algorithmes géométriques : nous utiliserons pour cela une *approche modulaire*. Le principe est de découper un diagramme en modules géométriques aux fonctionnalités bien définies et déjà prouvées, puis d'assembler ces « boîtes » de telle sorte à composer les preuves de fonctionnalités. Ainsi, les résultats du CHAPITRE 6 permettront de composer des diagrammes complexes à partir d'algorithmes géométriques simples de telle façon que la preuve globale de correction se déduira des preuves des algorithmes de base utilisés.

3.2.4 Les configurations comme entrées/sorties

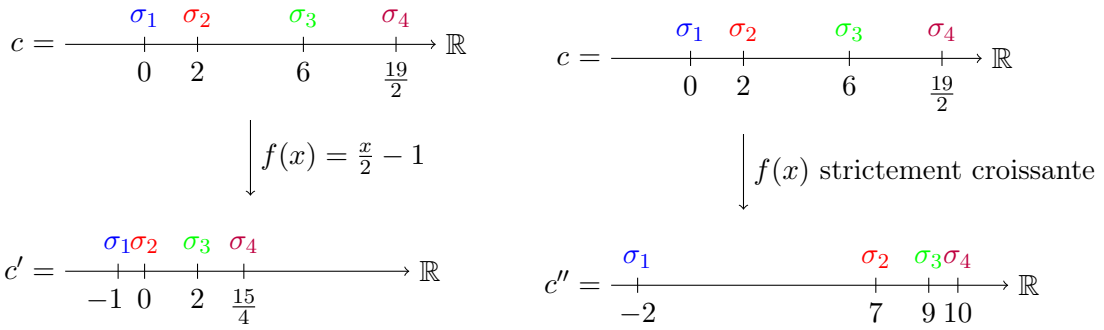
Les entrées/sorties sur machines à signaux sont définies par des ensembles de configurations, mais afin de pouvoir considérer les fonctions de représentations associées nous définissons ici des relations d'équivalences sur les configurations. Les ensembles quotients correspondant constitueront les ensembles des entrées et des sorties, pour lesquels les fonctions de représentations des entrées et des sorties seront respectivement injective et surjective.

Configurations équivalentes. Nous définissons ici des relations d'équivalence sur les configurations. L'intérêt de telles relations est de s'intéresser à la structure d'une configuration, indépendamment des positions exactes des objets la composant. Les deux principales structures utiles pour caractériser une configuration sont les *distances relatives* entre les positions de valeur non vide et l'*ordre* de ces objets.

DÉFINITION/PROPOSITION 26 (Configurations équivalentes)
Soit $\mathcal{C}onfig$ l'ensemble des configurations d'une machine à signaux. On définit une relation binaire $\sim_{aff}$ sur $\mathcal{C}onfig$ telle que pour toutes $c, c' \in \mathcal{C}onfig$ on a $c \sim_{aff} c'$ si et seulement s'il existe une fonction $f : \mathbb{R} \to \mathbb{R}$ affine de ratio strictement positif telle que $c = c' \circ f$. Alors $\sim_{aff}$ est une relation d'équivalence.

On définit une relation binaire $\sim_{aff}$ sur $\mathcal{C}onfig$ telle que pour toutes $c, c' \in \mathcal{C}onfig$ on a :
$c \sim_{ord} c'$ si et seulement s'il existe $f : \mathbb{R} \to \mathbb{R}$ une bijection croissante telle que $c = c' \circ f$. Alors $\sim_{ord}$ est une relation d'équivalence.

Si $c \sim_{aff} c'$, on dira que c et c' sont *affinement équivalentes*. Deux configurations sont donc affinement équivalentes s'il est possible de passer de l'une à l'autre par une translation de valeur



(a) Deux configurations affinement équivalentes.

(b) Deux configurations à ordres équivalents.

FIGURE 3.9 – Configurations équivalentes

b, suivie d'une homothétie de rapport positif a. Dans ce cas, la fonction f correspondante est donnée par $f(x) = a \cdot x + b$. Lorsque $c \sim_{ord} c'$, on dit que c et c' sont à *ordres équivalents*.

Démonstration. La relation $\sim_{aff}$ définit une relation d'équivalence : elle est trivialement réflexive (prendre $f = id$ la fonction identité) et transitive (car la composée de deux fonctions affines de ratios positifs est encore une fonction affine de ratio positif). De plus, $\sim_{aff}$ est symétrique car l'inverse d'une fonction affine de ratio positif existe et est également affine de ratio positif. Donc $\sim_{aff}$ est bien une relation d'équivalence.

De même, la relation $\sim_{ord}$ est réflexive, transitive (car la composée de deux bijections croissantes est encore une bijection croissante) et symétrique (car l'inverse d'une bijection croissante est une bijection croissante). Donc $\sim_{ord}$ est également une relation d'équivalence. □

Remarque 12. La condition « de ratio strictement positif » dans la définition de la relation d'équivalence affine de configurations est indispensable si l'on souhaite conserver non seulement les rapports de distance et l'ordre des signaux, mais surtout l'ordre des collisions à venir et la structure du diagramme.

Il est également possible de définir de la même manière une relation d'équivalence par translations, qui est un cas particulier de l'équivalence affine de configurations. On notera cette relation par $\sim_{trans}$ et on dira de deux configurations en relation par $\sim_{trans}$ qu'elles sont *équivalentes à translation près* ou *à translation équivalente*. Cela permettra entre autres d'encoder de manière injective des valeurs réelles par des distances, indépendamment de toute position exacte.

Deux configurations affinement équivalentes sont également à ordre équivalent puisque toute fonction affine de ratio strictement positif est également une bijection croissante. En fait, pour toutes c et c', on a : $c \sim_{trans} c' \Rightarrow c \sim_{aff} c' \Rightarrow c \sim_{ord} c'$.

Ensembles de configurations équivalentes. De nombreuses machines ne nécessitent pas de considérer des configurations avec les positions exactes : la distance relative des signaux, ou même juste leur ordre dans la configuration, suffit parfois pour déduire le comportement général de la machine sur toutes les configurations d'un certain type. Nous définissons de tels ensembles de configurations à partir des relations d'équivalences précédemment introduites :

DÉFINITION 27 (Ensembles de configurations équivalentes)
*Soit $\mathfrak{M}$ une machine à signaux. On définit $\mathcal{C}onfig_{\mathfrak{M}}^{aff}$ l'*ensemble des configurations à transformation affine près *(resp. $\mathcal{C}onfig_{\mathfrak{M}}^{ord}$ l'*ensemble des configurations à ordre près *et $\mathcal{C}onfig_{\mathfrak{M}}^{trans}$ l'*ensemble des configurations à translation près pour $\mathfrak{M}$*) par $\mathcal{C}onfig_{\mathfrak{M}}/_{\sim_{aff}}$ (resp. par $\mathcal{C}onfig_{\mathfrak{M}}/_{\sim_{ord}}$ et $\mathcal{C}onfig_{\mathfrak{M}}/_{\sim_{trans}}$).*

Remarque 13. Un calcul et son interprétation dépendront des ensembles de configurations choisis comme ensembles d'entrées et de sortie. Par exemple, en prenant pour sorties l'ensemble des configurations de la forme $\{\mathsf{w}@x,\ \mathsf{w}@y,\ \mathsf{w}@z \mid x < y < z\}$, la machine $\mathfrak{M}_{mid}$ présentée en SOUS-SEC. 3.1.1 permet de :

- calculer le milieu du segment $[a; b]$ si l'ensemble des entrées est $\left\{\mathsf{w}@a,\ \overrightarrow{\mathsf{hi}}@a,\ \overrightarrow{\mathsf{lo}}@a,\ \mathsf{w}@b\right\}$;
- calculer $a/2$ si l'ensemble des entrées est $\left\{\mathsf{w}@0,\ \overrightarrow{\mathsf{hi}}@0,\ \overrightarrow{\mathsf{lo}}@0,\ \mathsf{w}@a\right\}/_{\sim_{trans}}$;
- générer un nouveau signal vertical entre deux autres si l'ensemble des entrées est $\left\{\mathsf{w}@0,\ \overrightarrow{\mathsf{hi}}@0,\ \overrightarrow{\mathsf{lo}}@0,\ \mathsf{w}@a\right\}/_{\sim_{ord}}$.

La notion d'ordre équivalent pour les configurations, associée à la notion d'équivalence de diagramme donnée par la DÉF. 10, permet de formaliser une notion intuitive de robustesse :

DÉFINITION 28 (Machine robuste)
Une machine à signaux $\mathfrak{M}$ est dite robuste *si pour toutes configurations c_0, c'_0 telles que $c_0 \sim_{ord} c'_0$, les diagrammes engendrés par $\mathfrak{M}$ sur c_0 et c'_0 sont équivalents.*

Une machine à signaux $\mathfrak{M}$ est dite relativement robuste *s'il existe un segment $I \subseteq \mathbb{R}$, non réduit à un point, tel que pour toutes configurations c_0, c'_0 vérifiant $c_{0\restriction I} \sim_{ord} c'_{0\restriction I}$, les diagrammes engendrés par $\mathfrak{M}$ sur c_0 et c'_0 sont équivalents.*

Une machine est donc robuste si seul l'ordre des signaux importe pour ses configurations initiales (et non leurs positions exactes). Cette définition se conforme à l'idée intuitive de robustesse selon laquelle des perturbations sur l'entrée ne changent pas le calcul, ce qui se traduit ici par une liberté de placement des signaux initiaux, du moment qu'un certain ordre est respecté.

3.3 Complexités sur machines à signaux

Nous avons abordé en début de chapitre la notion de codages par signaux, et nous avons évoqué différents codages possibles suivant la nature des données considérées. En particulier, une valeur réelle x peut être codée par un nombre constant de signaux et par une distance dépendante de x. Inversement, un entier n (ou d'autres valeurs discrètes) peuvent être codées par un ensemble de signaux dont le nombre et le type dépendent de n, mais dont l'espace utilisé est indépendant de cette valeur. Si pour des calculs analogiques, l'espace et le temps nécessaires au calcul semblent être significatifs de la complexité de ce calcul, ce n'est pas le cas pour des calculs de type discrets. Par exemple, un faisceau $[\overrightarrow{bin}(n)]$ codant un entier n sera composé de $\lfloor \log_2(n) \rfloor + 1$ signaux, mais la largeur du faisceau (la distance entre le premier et le dernier signal) peut être choisie aussi petite que l'on veut sans affecter la valeur qu'il code.

Dans le cas de calculs discrets, les mesures naïves en temps et espace continus ne sont donc plus adaptées. Comme dans le cas de modèles classiques, ces mesures devraient être définies uniquement par rapport aux primitives de calculs. Il existe de multiples façons de définir des mesures de complexité dans le cadre de modèles non-conventionnels, comme cela est discuté dans [Blakey 2011a,b]. Des mesures de complexité non-conventionnelles doivent tenir compte soit de ressources non-classiques considérées, soit de la nature du modèle et de ses objets atomiques. Pour les machines à signaux, ces objets étant les signaux et les collisions, une « bonne » définition de complexité devrait donc être basée sur le nombre de signaux et de collisions mis en œuvre lors d'un calcul géométrique.

Nous présentons ici de nouvelles mesures de complexité, particulièrement adaptées au modèle des machines à signaux. Un diagramme espace-temps de machines à signaux peut être vu comme un *graphe dirigé acyclique (DAG)* afin de mieux appréhender les relations entre signaux et collisions entrant en jeu dans ce diagramme, et ce indépendamment de leurs positions spatio-temporelles exactes. La relation fondamentale qui sera alors considérée est la relation de causalité

entre deux objets, c'est-à-dire l'influence éventuelle de l'un sur l'autre. Le DAG muni de cet ordre de causalité peut ensuite être utilisé pour introduire de nouvelles mesures de complexité : la mesure en temps sera donnée par le plus grand nombre de collisions s'enchaînant les unes les autres, et la mesure en espace sera donnée par le plus grand nombre de signaux indépendants entre eux.

3.3.1 Définitions

Nous commençons par donner la définition formelle de ce que nous entendons par la hauteur et la largeur d'un diagramme espace-temps :

Définition 29 (Taille d'un diagramme espace-temps)
Soit $\mathcal{D} : \mathbb{R} \times [0;T] \to \mathbb{V}$ un diagramme espace-temps. La hauteur *du diagramme $\mathcal{D}$ est donnée par $\mathcal{H}aut(\mathcal{D}) = T$. Sa* largeur *est donnée par $\mathcal{L}arg(\mathcal{D}) = sup_{t\in[0;T]}\{|c_t|\}$. La* taille du diagramme $\mathcal{D}$ *est donnée à la fois par sa hauteur et sa largeur.*

Pour un diagramme $\mathcal{D}$ quelconque, les valeurs de $\mathcal{H}aut(\mathcal{D})$ et $\mathcal{L}arg(\mathcal{D})$ peuvent être infinies. On a implicitement considéré ici que l'intervalle de temps $[0;T]$ correspond à la partie « significative » du diagramme : après le temps T, le diagramme n'admet plus de dynamique.

Ces mesures continues sont adaptées pour des calculs analogiques (en prenant soin de relativiser par rapport à une unité de mesure et à la taille de la configuration d'entrée) mais ne sont pas du tout adaptées aux calculs combinatoires. En effet, comme nous l'avons déjà mentionné en Sous-sec. 2.4.1, il est possible d'appliquer des transformations aux diagrammes pour le contracter ou le dilater sans altérer l'historique des collisions, c'est-à-dire sans changer le calcul. La taille (mesurée en valeurs continues) de n'importe quel calcul géométrique peut donc être rendue aussi petite que l'on veut.

Ordre de causalité dans le diagramme espace-temps. Afin de définir des mesures de complexité qui prennent en compte les collisions et les signaux, et non leurs coordonnées exactes (et qui ne tient par conséquent pas compte de la taille du diagramme), nous considérons le diagramme espace-temps comme un *graphe dirigé acyclique (DAG)*. Pour cela, nous commençons par rappeler des définitions concernant les chaînes et les anti-chaînes :

Définition 30 (Chaîne et anti-chaîne)
Soit $(S, \leq_S)$ un ordre partiel. Un ensemble $\mathcal{C} \subseteq S$ est une chaîne *si $\leq_S$ induit sur $\mathcal{C}$ un ordre total : pour tous $a, b \in \mathcal{C}$ on a soit $a \leq_S b$ soit $b \leq_S a$. On dit qu'une chaîne $\mathcal{C}$ est une* chaîne maximale *si pour toute chaîne $\mathcal{C}' \subseteq S$ contenant $\mathcal{C}$, on a $\mathcal{C} = \mathcal{C}'$.*

Un ensemble $\mathcal{A} \subseteq S$ est une anti-chaîne *si les éléments de $\mathcal{A}$ sont tous deux à deux incomparables au sens de $\leq_S$: pour tous $a \neq b \in \mathcal{A}$ on a $a \not\leq_S b$ et $b \not\leq_S a$. On dit qu'une anti-chaîne $\mathcal{A}$ est une* anti-chaîne maximale *si pour toute anti-chaîne $\mathcal{A}' \subseteq S$ contenant $\mathcal{A}$, on a $\mathcal{A} = \mathcal{A}'$.*

Définition 31 (Relation de précédence entre signaux)
*On dit qu'un signal σ*précède immédiatement *un signal σ' s'il existe une collision C telle que $\sigma \in C^+$ et $\sigma' \in C^-$. On note $\sigma \prec \sigma'$.*

On définit la relation de précédence entre signaux *$\prec_*$ par la clôture transitive de $\prec$.*

On a $\sigma \prec_* \sigma$' si et seulement s'il existe une suite de signaux $(\sigma_i)_{1\leq i\leq n}$ et une suite de collisions $(C_i)_{1\leq i\leq n-1}$ telle que :

- $\forall i \in [1, n-2],\ \sigma_{i+1} \in C_i^+ \cap C_{i+1}^-$;
- $\sigma = \sigma_1 \in C_1^-$;
- $\sigma' = \sigma_n \in C_{n-1}^+$.

c'est-à-dire telle que : $\sigma = \sigma_1 \prec \sigma_2 \prec \ldots \sigma_{n-1} \prec \sigma_n = \sigma'$.

On définit également une relation de précédence entre collisions, que l'on appelera *relation de causalité* :

Définition 32 (Relation de causalité entre collisions)
On dit qu'une collision C' est conséquence immédiate *d'une collision C s'il existe un signal σ tel que $\sigma \in C^+$ et $\sigma \in C'^-$. On note $C \to C'$.*

On définit la relation de causalité entre collisions $\twoheadrightarrow$ *(ou simplement relation de causalité) par la clôture transitive de $\to$.*

La relation de causalité peut également se définir à partir de la relation de précédence entre signaux : on a $C \twoheadrightarrow C'$ si et seulement s'il existe des signaux σ, σ' tels que $\sigma \in C^+$, $\sigma' \in C'^-$ et $\sigma \prec_* \sigma'$. Notons que $\sigma = \sigma' \in C^+ \cap C'^-$ signifie que C' est conséquence immédiate de C.

Lemme 7
Les relations $\prec_*$ et $\twoheadrightarrow$ sont des ordres stricts partiels.

Démonstration. Il est clair que la relation $\prec_*$ n'est en général pas totale. Montrons que c'est une relation d'ordre strict. Par définition, $\prec_*$ est transitive en tant que clôture transitive de $\prec$. On montre par l'absurde l'anti-symétrie de $\prec$, qui implique trivialement celle de $\prec_*$. Supposons qu'il existe σ, μ des signaux tels que $\sigma \prec \mu$ et $\mu \prec \sigma$. Alors il existe deux collisions C et C' telles que $\sigma \in C^+$ $\mu \in C^-$ et $\sigma \in C'^-$, $\mu \in C'^+$. Comme $\mu \in C^-$ et $\mu \in C'^+$ (μ est crée par la collision C' et disparaît dans la collision C), on a $t'_C < t_C$ (où t_C désigne la coordonnée temporelle de la collision C). D'autre part, comme $\sigma \in C'^-$ et $\sigma \in C^+$, on a également $t_C < t'_C$, ce qui est absurde. Donc $\mu \prec \sigma$ implique $\sigma \not\prec \mu$. De même, $\prec_*$ est anti-réflexive. : si $\sigma \prec \sigma$ alors il existe une collision C telle que $\sigma \in C^-$ et $\sigma \in C^+$. On en déduit $t_C < t_C$, ce qui est absurde.

On montre de la même façon que $\twoheadrightarrow$ est un ordre partiel strict. □

Lemme 8
Soit $\mathcal{D}$ un diagramme espace-temps d'une machine à signaux. On considère le graphe dirigé $G_\mathcal{D} = (V, E)$ associé à $\mathcal{D}$, où l'ensemble des sommets V est l'ensemble des collisions de $\mathcal{D}$, l'ensemble d'arêtes E est l'ensemble des signaux de $\mathcal{D}$ et l'orientation des arêtes est donnée par la relation $\to$. Alors $G_\mathcal{D}$ est un graphe dirigé acyclique (DAG).

Démonstration. Le Lemme 7 nous permet d'affirmer que le graphe ne contient pas de cycle, par anti-réflexivité de $\twoheadrightarrow$. □

Pour un diagramme $\mathcal{D} : \mathbb{R} \times \mathbb{R}^+ \to \mathcal{M} \cup \mathcal{R} \cup \{\oslash\}$, les ensembles de sommets et d'arrêtes sont donc respectivement donnés par $V = \mathcal{D}^{-1}(\mathcal{R})$ et $E = \mathcal{D}^{-1}(\mathcal{M})$.

Définition de nouvelles mesures de complexité. Nous pouvons maintenant définir des mesures de complexité tenant compte de la structure du diagramme et non de sa taille. L'opération élémentaire du modèle étant la collision, les définitions suivantes sont basées les chaînes et anti-chaînes dans le diagramme, lorsqu'il est muni de la relation de causalité entre collisions.

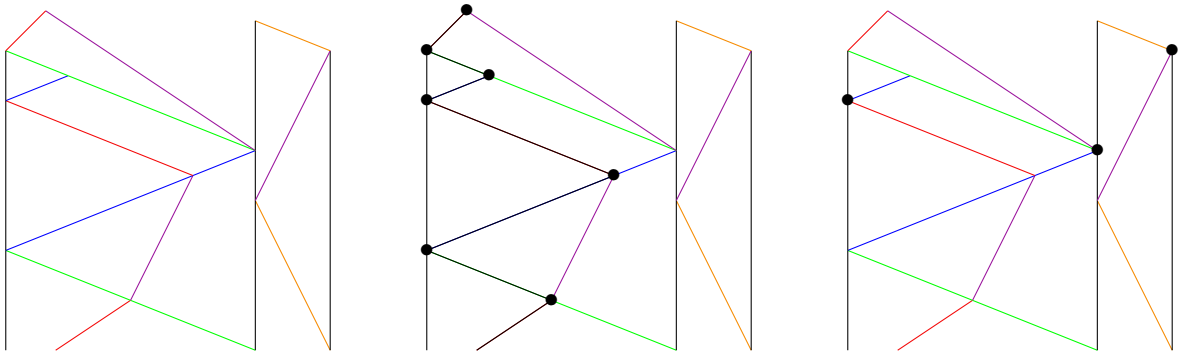
Définition 33 (Profondeur de collisions)
La profondeur de collisions *du diagramme $\mathcal{D}$ est la taille d'une chaîne maximale de $G_\mathcal{D}$ i.e. $pc(\mathcal{D}) = max\{\ card(\mathcal{C}) \mid \mathcal{C} \subseteq V$ est une chaîne de $G_\mathcal{D}\}$.*

Définition 34 (Largeur de collisions)
La largeur de collisions *du diagramme $\mathcal{D}$ est la taille d'une anti-chaîne maximale de $G_\mathcal{D}$ i.e. $lc(\mathcal{D}) = max\{\ card(\mathcal{A}) \mid \mathcal{A} \subseteq V$ est une anti-chaîne de $G_\mathcal{D}\}$.*

Les définitions de chaîne et anti-chaîne appliquées ici donnent : $\mathcal{C} = \{C_i\}_{1 \leq i \leq n}$ est une chaîne si et seulement si pour tous $C, C' \in \mathcal{C}$, on a soit $C \twoheadrightarrow C'$ soit $C' \twoheadrightarrow C$. De même, $\mathcal{A} = \{C_i\}_{1 \leq i \leq n}$ est une anti-chaîne si et seulement si pour tous $C \neq C' \in \mathcal{A}$, on a $C \not\twoheadrightarrow C'$ et $C' \not\twoheadrightarrow C$.

Pour un diagramme $\mathcal{D}$, on notera la profondeur de collisions et la largeur de collisions de $\mathcal{D}$ respectivement par $pc(\mathcal{D})$ et $lc(\mathcal{D})$. La FIGURE 3.10 illustre ces mesures sur un exemple simple de diagramme. Par la suite, nous confondrons un diagramme espace-temps $\mathcal{D}$ avec le graphe correspondant $G_{\mathcal{D}}$.

Remarque 14. Notons qu'on peut également définir des mesures de complexité basées sur la relation de précédence entre signaux, respectivement la *profondeur en signaux* et la *largeur en signaux*.



(a) Un diagramme. (b) Une chaîne maximale. (c) Une anti-chaîne maximale.

FIGURE 3.10 – Mesures de complexité d'un diagramme espace-temps.

Remarque 15. Les notions de temps de collisions et de front de collisions ne sont plus pertinentes dans le cadre d'un diagramme vu comme DAG. En effet, la notion de simultanéité sous-tend celle de front de collision, et les coordonnées spatio-temporelles ne jouent plus aucun rôle dans le DAG : seul l'ordre de causalité importe.

La *complexité globale* d'une résolution géométrique à un problème sera donnée d'une part par la complexité classique des schémas de représentations des instances en machines à signaux et configuration initiales ; et d'autre part par les complexités en collisions des diagrammes résultants.

3.3.2 Propriétés des mesures de complexité en collisions

Nous énonçons ici quelques propriétés immédiates des mesures de complexité en collisions.

LEMME 9
Soit $\mathcal{D}$ un diagramme espace-temps. Alors :

(i) *$pc(\mathcal{D})$ est inférieur ou égal au nombre total de collisions ;*

(ii) *$pc(\mathcal{D})$ est inférieur ou égal au nombre de front de collisions.*

Démonstration. *(i)* Pour toute chaîne $\mathcal{C}$ de $G_{\mathcal{D}} = (V, E)$, on a $\mathcal{C} \subseteq V$.

(ii) Toute collision appartient exactement à un front de collisions et deux collisions enchaînées appartiennent forcément à deux fronts différents (puisque l'une se produit strictement avant l'autre). Le nombre de fronts de collisions est donc supérieur ou égal au plus grand nombre de collisions enchaînées.

□

Lemme 10
Soit $n^{col}_{simult}(\mathcal{D}) = \underset{t\in\mathbb{R}^+}{max}\{card(c_t^{-1}(\mathcal{R})) \mid c_t \textit{est un front de collisions}\}$. *Alors* $n^{col}_{simult}(\mathcal{D}) \leq lc(\mathcal{D})$.

Ce lemme signifie que $lc(\mathcal{D})$ est au moins égal au nombre maximal de collisions se produisant simultanément dans la machine au cours du calcul (ce nombre correspond bien au nombre de collisions d'un front de collisions maximal). Cela provient du fait que des collisions simultanées ne peuvent pas être deux-à-deux conséquences les unes des autres, et formant ainsi une anti-chaîne.

3.3.3 Exemples

On donne ici les complexités en collisions et en temps et espaces continus des algorithmes géométriques présentés en Sec. 3.1 : les complexités sont mesurées par la largeur et la hauteur de la construction dans le cas de la division géométrique par deux et le cas de la soustraction, qui sont de nature analogique. Une addition binaire géométrique, de nature combinatoire, sera mesurée par les mesures de complexité précédemment définies c'est-à-dire par sa profondeur et sa largeur de collisions.

Division par deux. On appelle $\mathcal{D}_x$ le diagramme engendré par $\mathfrak{M}_{mid}$ sur une configuration initiale telle que les signaux verticaux w sont espacés d'une distance réelle x. D'après l'étude faite en Sous-sec. 3.1.1, le diagramme $\mathcal{D}_x$ vérifie $\mathcal{H}aut(\mathcal{D}_x) = x/2$ et $\mathcal{L}arg(\mathcal{D}_x) = x$ et $pc(\mathcal{D}_x) = 2$ et $lc(\mathcal{D}_x) = 1$. Cet algorithme géométrique manipulant des valeurs réelles, nous mesurons son efficacité par la taille du diagramme obtenu, et non par sa largeur et profondeur de collisions. Les complexités en temps et espace continus sont donc toutes deux en $\mathcal{O}(x)$, où x désigne la taille du réel représenté en entrée.

Soustraction. On appelle $\mathcal{D}_{a,b}$ le diagramme engendré par $\mathfrak{M}_{sub}$ sur une configuration initiale telle que les signaux verticaux w sont espacés respectivement par les distances réelles b et a, comme illustré en Sous-sec. 3.1.2. On a alors $\mathcal{H}aut(\mathcal{D}_{a,b}) = a+b+|b-a|$ et $\mathcal{L}arg(\mathcal{D}_{a,b}) = a+b$. Comme pour le calcul du milieu, les complexités en collision sont là aussi constantes : $pc(\mathcal{D}_{a,b}) = 4$ et $lc(\mathcal{D}_{a,b}) = 2$.

Modulo. L'exemple du calcul géométrique du modulo constitue un cas intéressant, car à la fois les mesures en temps et espace continus et les mesures en collisions dépendent de la taille des entrées. On appelle $\mathcal{D}_{a,b}$ le diagramme engendré par $\mathfrak{M}_{mod}$ sur une configuration initiale telle que les signaux verticaux w sont espacés respectivement par les distances réelles b et a, comme illustré en Sous-sec. 3.1.2 (*c.f.* la Fig. 3.3(b)), et dont le résultat est $a \bmod b$. On a alors $\mathcal{H}aut(\mathcal{D}_{a,b}) = \mathcal{O}(a+b)$ et $\mathcal{L}arg(\mathcal{D}_{a,b}) = a+b$ et $pc(\mathcal{D}_{a,b}) = \mathcal{O}(\lfloor a/b \rfloor)$ et $lc(\mathcal{D}_{a,b}) = \mathcal{O}(\lfloor a/b \rfloor)$.

Addition binaire. On appelle $\mathcal{D}_{n,m}$ le diagramme engendré par $\mathfrak{M}_{add}$ sur une configuration initiale dont laquelle sont représentés les entiers n et m par les faisceaux binaires $[\overrightarrow{bin}(n)]$ et $[\overleftarrow{bin}(m)]$, séparés par un signal $+^R$, déclencheur de l'addition. Une telle configuration a été détaillée en Sous-sec. 3.1.3. Soit l la largeur de la configuration initiale. On a alors $\mathcal{H}aut(\mathcal{D}_{n,m}) = l/2$ et $\mathcal{L}arg(\mathcal{D}_{n,m}) = l$. Le nombre total de collisions dans le diagramme est en $\mathcal{O}(\lfloor \log_2(n) \rfloor \times \lfloor \log_2(m) \rfloor)$. Les complexités en collisions sont quant à elles données par $pc(\mathcal{D}_{n,m}) = \mathcal{O}(max(\lfloor \log_2(n) \rfloor, \lfloor \log_2(m) \rfloor))$ et $lc(\mathcal{D}_{n,m}) = \mathcal{O}(\lfloor \log_2(n) \rfloor + \lfloor \log_2(m) \rfloor)$.

Chapitre 4

Accumulations et petites machines

Ce chapitre est consacré aux accumulations et à l'étude de certaines de leurs propriétés. Après un bref rappel du contexte général dans lequel se situent les accumulations (entre autres, leur lien avec le paradoxe de la dichotomie de Zénon) et quelques exemples d'utilisation d'accumulations dans d'autres modèles de calcul, nous rappellerons quelques unes de leurs propriétés. Nous ferons également le lien avec les notions de supports et d'inclusions introduites en Sec. 2.4.2.

Nous présenterons ensuite la contribution principale de ce chapitre, obtenue par l'étude des accumulations sur les « petites » machines à signaux, c'est-à-dire les machines admettant un faible nombre de vitesses distinctes. Nous dégagerons de cette étude une condition nécessaire sur le nombre de vitesses distinctes qu'une machine doit posséder pour pouvoir engendrer des accumulations. Dans le cas général, construire une accumulation nécessite au moins trois vitesses différentes et au moins quatre vitesses dans le cas de machines rationnelles.

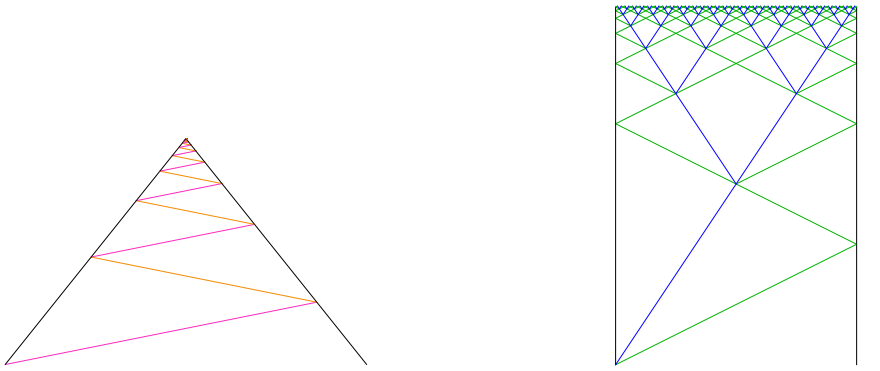
Ce chapitre sera conclu par une application des résultats obtenus à quelques problèmes de décidabilité concernant les machines à signaux.

4.1 Contexte général et nature des accumulations

Cette section regroupe quelques généralités sur les accumulations : leur lien avec le paradoxe de Zénon, leur utilisation dans certains modèles de calcul non conventionnels et quelques unes de leurs propriétés dans le modèle des machines à signaux.

4.1.1 Contexte

Accumulations et paradoxe de Zénon. Zénon d'Élée, l'antique philosophe grec, énonça ses fameux paradoxes pour illustrer la difficulté de cerner et définir les notions de continuité et d'infini. Les accumulations peuvent être vues comme une version d'un des paradoxes de Zénon, celui de la dichotomie. Ce paradoxe est une caractéristique importante des espaces continus (et d'autres espaces comme $\mathbb{Q}$), dans lesquels des parties peuvent être divisées une infinité de fois en parties plus petites. Pourtant, de telles parties divisibles indéfiniment restent de taille finie bien qu'elles soient exprimables en tant que réunion infinie de parties non vides disjointes. Les accumulations sur machines à signaux peuvent être comprises dans le contexte de ce paradoxe : une accumulation est la réalisation d'un nombre infini d'étapes de calcul — des collisions — pendant une durée finie. Par exemple, l'accumulation donnée dans l'introduction et reprise en Fig. 4.1(a) correspond à une infinité d'aller-retours (avec deux collisions à chaque étape) entre les deux signaux qui se rapprochent de plus en plus. La Figure 4.1(b) fournit un autre exemple d'accumulation, qui sera repris par la suite. Cet exemple est important car il illustre à la fois la capacité des machines à signaux à générer des fractales non triviales et à effectuer des opérations parallèles.



(a) Accumulation isolée.

(b) Accumulation non-isolée [Durand-Lose 2003, Fig. B.1(b)].

FIGURE 4.1 – Exemples d'accumulations.

État de l'art. Nous donnons ici quelques exemples de modèles et contextes de calcul dans lesquels les accumulations ont déjà été considérées ou utilisées (la plupart ont déjà été évoqués dans le CHAP. 1).

Par nature, le continu permet l'existence d'accumulations : des quantités continues peuvent en effet être divisées indéfiniment. En tant que modèles dynamiques continus, les systèmes à dérivées constantes par morceaux (PCD systems) permettent les accumulations. Elles y ont d'ailleurs été utilisées pour accélérer les calculs (avec un facteur d'accélération infini), et permettent ainsi d'escalader les hiérarchies arithmétique [Asarin et Maler 1995] et hyper-arithmétique [Bournez 1999]. La FIGURE 4.2(a) illustre l'utilisation d'une dimension spatiale supplémentaire pour implémenter un effet de type Zénon, permettant ainsi d'accélérer un calcul effectué dans les autres dimensions spatiales. Nous remarquons que la structure basique qui est utilisée en FIG. 4.2(a) est la même que l'accumulation « zig-zag » de la FIG. 4.1(a).

Les accumulations sont une caractéristique essentielle du continu, mais il est possible de les considérer dans des contextes discrets. C'est le cas de machines de Turing dont l'exécution est accélérée de manière non-uniforme : les transitions y sont effectuées de plus en plus rapidement et les durées des transitions successives sont données par une suite géométrique, de telle sorte que le temps total de calcul correspond à une série géométrique convergente. Une infinité de transitions, et par conséquent un calcul infini, peut ainsi être effectué en un temps fini. De telles machines se retrouvent sous différentes dénominations dans la littérature : machines de Zeus [Boolos et Jeffrey 1980, pp. 14–15], machines de Turing accélérantes [Copeland 2002], machines de Zénon [Potgieter 2006], *etc*. Notons que l'utilisation de phénomènes de type Zénon pour des calculs discrets avait déjà été évoquée dans les années 1930 par Ralph Blake [Blake 1926], Herman Weyl [Weyl 1927] et Bertrand Russell [Russell 1936].

Les accumulations ont également été considérées dans le cadre des automates cellulaires : dans [Schaller et Svozil 2010], l'espace discret des AC est donné par une grille de type Zénon comme celle de la FIG. 4.2(b). Cette structure permet à une cellule d'être influencée par les états d'une infinité d'autres cellules.

Dans tous ces modèles, ainsi que dans d'autres modèles tels que le modèle « trou noir » dans un contexte relativiste, les accumulations permettent de résoudre des problèmes récursivement énumérables, tels que le problème de l'arrêt, indécidable sur les modèles classiques. La méthode est d'utiliser l'accumulation pour y implémenter une accélération de type Zénon, qui permet d'effectuer une infinité de transitions d'une machine de Turing en un temps fini, et par conséquent de décider de son arrêt en temps fini.

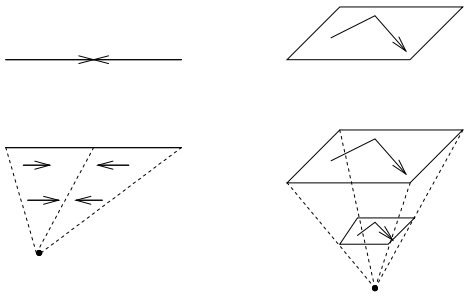


Figure 7: A homogenization of a 1-dimensional system (left) and a 2-dimensional system (right).

(a) Accumulations dans les PCD [Asarin et Maler 1995, Fig. 7].

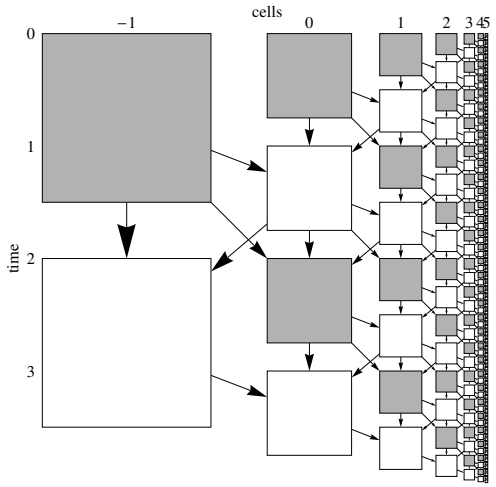


Figure 2: Evolution of a self-similar cellular automaton.

(b) Définition d'un AC basé sur une accumulation [Schaller et Svozil 2010, Fig. 2].

FIGURE 4.2 – Utilisation d'accumulations dans d'autres modèles de calcul.

Nature des accumulations sur machines à signaux. Comme présenté dans la SEC. 2.2, les accumulations dans un diagramme espace-temps peuvent être de deux types : soit statique, soit dynamique. Nous en donnons ici des exemples, et distinguons un type particulier d'accumulation dynamique : les accumulations isolées.

On rappelle que le point (x_0, t_0) est une *accumulation statique* si pour tout voisinage $\mathcal{V}_{(x_0,t_0)}$ de (x_0, t_0), il y a une infinité de signaux ou collisions dans $\mathcal{V}_{(x_0,t_0)} \bigcap (support(c_{t_0}) \times \{t_0\})$. Dans cette définition, on peut remplacer les voisinages de (x_0, t_0) par des voisinages « spatiaux » de x_0 au temps t_0, $\mathcal{V}_{x_0}^{t_0} \subseteq \mathbb{R}$, donnés par $\mathcal{V}_{x_0}^{t_0} = \left\{x \in \mathbb{R} \;\middle|\; (x, t_0) \in \mathcal{V}_{(x_0,t_0)}\right\}$. La condition devient alors : $\mathcal{V}_{x_0}^{t_0} \bigcap support(c_{t_0}) \neq \emptyset$. Une accumulation statique est donc une accumulation purement spatiale, et correspond à une accumulation topologique pour le support de la configuration au temps t_0 dans l'espace $\mathbb{R}$. On rappelle que le point (x_0, t_0) est une *accumulation dynamique* si pour tout voisinage $\mathcal{V}_{(x_0,t_0)}$ de (x_0, t_0), il y a une infinité de signaux ou collisions dans $\mathcal{V}_{(x_0,t_0)} \cap \Gamma^-(x_0, t_0)$, où $\Gamma^-(x_0, t_0)$ désigne le passé causal de (x_0, t_0).

La distinction entre accumulations statique et dynamique (*i.e.* entre accumulations spatiale et temporelle) peut se faire à travers la notion du passé causal. Une accumulation dynamique correspond par définition à une infinité de signaux et collisions dans le passé causal, alors qu'une accumulation statique implique une infinité de valeurs non vides à un instant donné, et donc en dehors du passé causal (et même en dehors du cône de lumière). En effet, deux positions distinctes considérées au même instant donné ne peuvent être dans le passé causal l'une de l'autre.

Parmi les accumulations dynamiques, nous distinguons un type en particulier, les *accumulations isolées* :

DÉFINITION 35 (Accumulation isolée)
Une accumulation au point (x_0, t_0) *est dite* isolée *si suffisamment proche de* (x_0, t_0) *:*

(i) $\exists\ \varepsilon > 0,\ \forall (x', t') \notin \Gamma^-(x, t),\ |x - x'| < \varepsilon \wedge |t - t'| < \varepsilon \Rightarrow \mathcal{D}(x', t') = \oslash,$

(ii) $\forall\ \varepsilon > 0,$ *l'ensemble* $\{(x', t') \in \Gamma^-(x, t) \mid t - \varepsilon < t' < t \wedge \mathcal{D}(x', t') \in \mathcal{M} \cup \mathcal{R}\}$ *est infini.*

La condition *(i)* signifie que la seule valeur possible en dehors du passé causal est $\oslash$, La condition *(ii)* correspond au fait qu'il y a une infinité de signaux et collisions dans le passé causal. Une accumulation isolée est purement dynamique et locale. En effet, une accumulation

isolée ne peut pas être spatiale, la condition *(i)* imposant que la seule valeur en dehors du passé causal soit le vide. Des exemples d'accumulations isolée et non-isolée sont illustrés respectivement par les FIG. 4.1(a) et FIG. 4.1(b), données auparavant.

4.1.2 Propriétés

Nous rappelons ici quelques propriétés des accumulations.

Difficulté de prévoir les accumulations. Il a été démontré que les accumulations sont des phénomènes dynamiques imprévisibles en général. En effet, le problème consistant à décider si une accumulation apparaît dans le diagramme engendré par une machine à signaux rationnelle à partir d'une configuration initiale donnée, a été montré indécidable [Durand-Lose 2003]. En fait, ce problème est même fortement indécidable (un problème fortement indécidable appartient au deuxième niveau de la hiérarchie arithmétique, et est donc un niveau au-dessus du problème de l'arrêt d'une machine de Turing).

Coordonnées des accumulations isolées. Les coordonnées d'accumulations isolées peuvent être caractérisées de manière exacte dans le cas de machines rationnelles [Durand-Lose 2012b] : ce sont exactement les nombres réels calculables (réels *c.e.*) et les différences de réels calculables (réels *d.c.e.*) (voir [Weihrauch 2000] pour les définitions exactes des nombres c.e. et d-c.e.). Plus exactement :

THÉORÈME 11 ([Durand-Lose 2012b])
Les coordonnées des accumulations isolées de machines rationnelles sont des réels d-c.e.
Il existe une machine à signaux rationnelle pouvant générer une accumulation dont la coordonnée temporelle (resp. spatiale) peut être n'importe quel réel c.e. *(resp. n'importe quel réel* d-c.e.*).*

Approche dynamique et accumulations. Le lemme suivant fait le lien entre accumulation dynamique et suite des temps de collisions :

LEMME 12
Il y a une accumulation dynamique au point (x,t) si et seulement s'il existe une suite de collisions $(C_n)_{n\in\mathbb{N}}$ ordonnée par date telle que $\lim_{n\to\infty}(x_n,t_n)=(x,t)$, où (x_n,t_n) sont les coordonnées de la collision C_n.

Démonstration. Dans le cas d'une accumulation dynamique en (x,t), la suite $(C_n)_{n\in\mathbb{N}}$ est donnée par la suite (infinie) des collisions s'accumulant dans $\Gamma^-(x,t)$, le passé causal de (x,t). On a bien $\lim_{n\to\infty}(x_n,t_n)=(x,t)$ (car pour toute collision C_n, $C_n\in\Gamma^-(x,t)$).

Pour l'autre sens, la DÉF. 13 implique directement qu'il y a une accumulation dynamique au point (x,t). □

4.2 Accumulations à deux et quatre vitesses

Nous nous intéressons à l'étude des accumulations sur les « petites » machines à signaux c'est-à-dire sur les machines à signaux à une dimension spatiale et ayant un faible nombre de vitesses distinctes. La problématique ici consiste à déterminer le nombre minimal de vitesses pour pouvoir produire une accumulation. Concrètement, nous distinguons les cas de machines à 2, 3 et 4 vitesses (n rappelle que $\mathfrak{M}$ est machine à n vitesses s'il existe exactement n valeurs distinctes pour les vitesses des méta-signaux de $\mathfrak{M}$). Le cas 1 vitesse n'est pas abordé car étant trivial : en effet, aucune collision n'est possible avec seulement une vitesse, et donc aucune accumulation

a fortiori. Nous allons montrer que les cas de deux et quatre vitesses sont assez immédiats : les accumulations avec deux et quatre vitesses sont respectivement proscrites et faisables. Le cas de trois vitesses est plus délicat à traiter et fera l'objet de la section suivante.

4.2.1 Cas de quatre vitesses

Nous fournissons ici la preuve formelle que l'exemple de la FIG. 4.1(a) (déjà donnée dans l'introduction) est bien une accumulation. Pour générer l'accumulation simple de la FIGURE 4.3, on considère la machine à signaux $\mathfrak{M}_4$ définie par les quatre méta-signaux suivants : left ($\frac{1}{2}$), right ($-\frac{1}{2}$), zig (4), zag (-4) ; et par deux règles de collisions définissant les rebonds de zig (resp. zag) sur right (resp. left) par :

$$\{\mathsf{left}, \mathsf{zag}\} \to \{\mathsf{left}, \mathsf{zig}\} \quad \text{et} \quad \{\mathsf{zig}, \mathsf{right}\} \to \{\mathsf{zag}, \mathsf{right}\} \ .$$

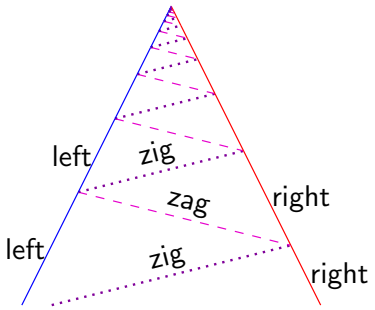


FIGURE 4.3 – Une accumulation simple à quatre vitesses.

La machine $\mathfrak{M}_4$ est une machine à 4 vitesses qui permet de générer des accumulations :

LEMME 13
La machine $\mathfrak{M}_4$ exécutée à partir de la configuration initiale $c_0 = \{\mathsf{left}@{-1}, \mathsf{zig}@{-1}, \mathsf{right}@1\}$ génère une accumulation isolée au point de coordonnées $(0, 2)$.

Démonstration. On considère la suite $(C_n)_{n\in\mathbb{N}}$ des collisions successives dans le diagramme engendré par $\mathfrak{M}_4$ à partir de la configuration c_0 et on note (x_n, t_n) les coordonnées (spatiales et temporelles) correspondantes. On définit également δ_n comme étant le délai entre deux collisions successives C_n et C_{n+1} : $\delta_n = t_{n+1} - t_n$.

Nous allons calculer les coordonnées de chaque collision C_n et montrer que la suite des collisions est une alternance des deux règles de collisions définies ci-dessus. On a pour tout $n \geq 0$:

$$C_n = \begin{cases} \{\mathsf{zig}, \mathsf{right}\} \to \{\mathsf{zag}, \mathsf{right}\} & \text{si } n \text{ est impair } \textit{(rebond à droite)} \\ \{\mathsf{left}, \mathsf{zag}\} \to \{\mathsf{left}, \mathsf{zig}\} & \text{si } n \text{ est pair } \textit{(rebond à gauche)} \end{cases} .$$

On suppose par convention que la première collision C_0 de coordonnées (x_0, t_0) vérifie $t_0 = 0$ et $x_0 = -1$ (c'est-à-dire la position initiale de left et zig) et correspond à un rebond à gauche. On notera que les positions des signaux left et right de la configuration c_0 sont toujours opposées car leurs vitesses sont opposées et ils sont disposés initialement de manière symétrique par rapport à 0. Les règles de collisions assurant qu'ils gardent le même mouvement après chaque collision, leurs positions resteront toujours opposées.
Supposons qu'une configuration au temps t_n soit de la forme $c_{t_n} = \{\mathsf{left}@x_n, \mathsf{zig}@x_n, \mathsf{right}@{-x_n}\}$. Toute configuration provenant d'une collision C_n de coordonnées (x_n, t_n) et de type rebond à gauche vérifie cette forme de configuration. C'est le cas en particulier pour la configuration initiale c_0 avec $x_0 = -1$. Il est clair à partir de la disposition des signaux que la prochaine collision C_{n+1}

qui arrivera sera un rebond à droite *i.e.* $\{\mathsf{zig}, \mathsf{right}\} \to \{\mathsf{zag}, \mathsf{right}\}$ car zig (de vitesse 4) se déplace vers la droite et est situé à gauche de right (de vitesse $-\frac{1}{2}$) qui se déplace vers la gauche. Les signaux left et zig ne peuvent pas rentrer en collision car left est situé à gauche de zig qui se déplace plus rapidement vers la droite que left. La collision entre zig et right se déduit de leurs dynamiques respectives, et ses coordonnées (x,t) vérifient : $\begin{cases} x = 4 \cdot (t - t_n) + x_n & \text{(dynamique de zig)} \\ x = -\frac{1}{2} \cdot (t - t_n) - x_n & \text{(dynamique de right)} \end{cases}$.
La collision entre zig et right arrive au temps $t = t_{n+1}$, et les signaux zig et right occupe la même position spatiale après la durée $\delta_n = t_{n+1} - t_n$ vérifiant $4 \cdot \delta_n + x_n = -\frac{1}{2} \cdot \delta_n - x_n$. On en déduit le délai δ_n et la position x_{n+1} :

$$\delta_n = -\frac{4}{9} \cdot x_n \quad \text{et} \quad x_{n+1} = -\frac{7}{9} \cdot x_n \ . \tag{4.1}$$

Comme les signaux rentrants de la collision C_{n+1} sont remplacés par les signaux sortants, la configuration au temps t_{n+1} est $c_{t_{n+1}} = \{\mathsf{left}@{-x_{n+1}},\ \mathsf{zag}@x_{n+1},\ \mathsf{right}@x_{n+1}\}$.

Pour le cas symétrique, lorsque $c_{t_n} = \{\mathsf{left}@{-x_n},\ \mathsf{zig}@x_n,\ \mathsf{right}@x_n\}$ et C_n est un rebond à droite, la collision suivante C_{n+1} est alors un rebond à gauche et on trouve de la même manière que précédemment sa position x_{n+1} ainsi que la durée δ_n entre C_n et C_{n+1} :

$$\delta_n = \frac{4}{9} \cdot x_n \quad \text{et} \quad x_{n+1} = -\frac{7}{9} \cdot x_n \ . \tag{4.2}$$

Donc à partir de la configuration initiale c_0 et en appliquant successivement les deux calculs précédents aux configurations résultant de chaque collision, on obtient une alternance de rebonds à droite et de rebonds à gauche (en commençant par un rebond à droite). D'après Eq. (4.1) et Eq. (4.2), les suites $(x_n)_{n\in\mathbb{N}}$ et $(\delta_n)_{n\in\mathbb{N}}$ vérifient :

$$\begin{cases} x_0 = -1\ ;\ x_{n+1} = -\frac{7}{9} \cdot x_n \\ \delta_n \ = \ (-1)^{n+1} \cdot \frac{4}{9} \cdot x_n \end{cases} \quad \text{ce qui donne } \forall n \in \mathbb{N} \begin{cases} x_n = (-1)^{n+1} \cdot \left(\frac{7}{9}\right)^n \\ \delta_n = \frac{4}{9} \cdot \left(\frac{7}{9}\right)^n \end{cases} .$$

Finalement, à partir de la relation $t_{n+1} = t_n + \delta_n$ avec $t_0 = 0$, on obtient pour tout $n \in \mathbb{N}$:

$$t_{n+1} \ = \ \sum_{i=0}^{n} \delta_i \ = \ \sum_{i=0}^{n} \frac{4}{9} \times \left(\frac{7}{9}\right)^i \ = \ \frac{4}{9} \times \sum_{i=1}^{n} \left(\frac{7}{9}\right)^i \ .$$

La suite $(t_n)_{n\in\mathbb{N}}$ est infinie, strictement croissante et positive. Elle admet une limite qui est donnée par la limite d'une suite géométrique de raison $\frac{7}{9} \ < \ 1$:

$$\lim_{n\to\infty} t_n = \ \frac{4}{9} \times \sum_{i=0}^{\infty} \left(\frac{7}{9}\right)^i \ = \ \frac{4}{9} \times \frac{1}{1 - \frac{7}{9}} \ = \ 2 \ .$$

La suite $(x_n)_{n\in\mathbb{N}}$ des positions spatiales admet également une limite :

$$\lim_{n\to\infty} x_n \ = \ \lim_{n\to\infty} \left| (-1)^n \cdot \left(\frac{7}{9}\right)^n \right| \ = \ \lim_{n\to\infty} \left(\frac{7}{9}\right)^n \ = \ 0 \ .$$

La machine $\mathfrak{M}_4$ exécutée sur la configuration initiale c_0 produit donc une suite de collisions successives de coordonnées (x_n, t_n) vérifiant $\lim_{n\to\infty}(x_n, t_n) = (0, 2)$. D'après la Déf. 13 et le Lem. 12, on a $c_{\tilde{t}}(0) = c_2(0) = \text{✺}$ c'est-à-dire que le point $(0, 2)$ est un point d'accumulation (isolée). □

On déduit de l'étude de cet exemple que :

Corollaire 14
Une accumulation peut être générée en utilisant seulement quatre vitesses différentes.

Remarque 16. Dès qu'une machine à signaux contient au moins quatre méta-signaux de vitesses distinctes et des règles de collisions permettant à deux de ces méta-signaux de rebondir alternativement sur les deux autres, il devient alors possible de générer des accumulations dans le genre de l'accumulation décrite précédemment (avec une configuration initiale bien choisie). Cela suggère que la capacité d'accumuler n'est pas si rare pour les machines à signaux contenant suffisamment de vitesses différentes.

4.2.2 Cas de deux vitesses

Réduction à une machine normalisée. Soit $\mathfrak{M}^{a,b} = (\mathcal{M}, \mathcal{S}, \mathcal{R})$ une machine à signaux telle que $\mathcal{I}m(\mathcal{S}) = \{a, b\}$ c'est-à-dire que les méta-signaux ne peuvent prendre que les valeurs a et b comme vitesse. On définit la fonction $g : \mathbb{R} \to \mathbb{R}$ par $g(x) = \frac{x}{b-a} - \frac{a}{b-a}$. On a $g(a) = 0$ et $g(b) = 1$ (en fait, g est la fonction f définie dans la Sous-sec. 2.4.1, avec $c = 0$ et $d = 1$). On normalise la machine $\mathfrak{M}^{a,b}$ en une machine $\mathfrak{M}^{0,1} = \mathfrak{M}_g^{a,b} = (\mathcal{M}, \mathcal{S}', \mathcal{R})$ où $\mathcal{S}' = g \circ \mathcal{S}$. Les nouvelles (et seules) valeurs de vitesse sont 0 et 1, les méta-signaux et règles restant inchangés. Comme g est une fonction affine de coefficient strictement positif (car $a < b$) et que $\mathfrak{M}^{0,1} = \mathfrak{M}_g^{a,b}$, on en déduit grâce au Lem. 1 que $\mathfrak{M}^{0,1}$ produit des diagrammes espace-temps équivalents à ceux générés par $\mathfrak{M}^{a,b}$. En particulier, si $\mathfrak{M}^{a,b}$ produit des accumulations, alors $\mathfrak{M}^{0,1}$ produira aussi des accumulations.

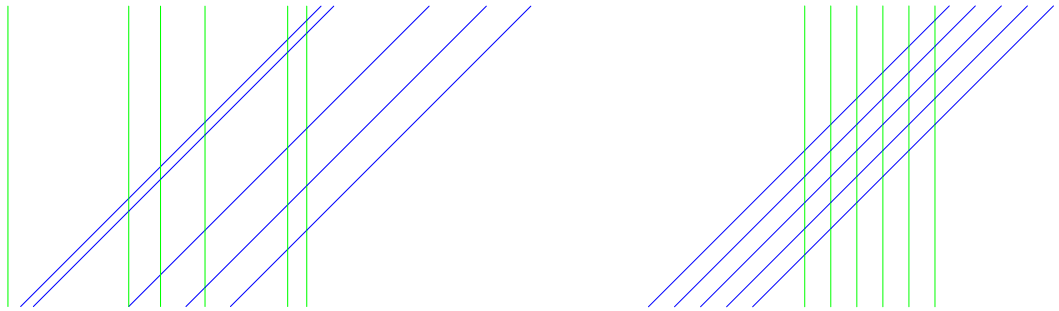
Accumuler est impossible avec uniquement deux vitesses. Nous pouvons maintenant donner la preuve de l'impossibilité de produire une accumulation avec uniquement deux vitesses différentes. Grâce au Lem. 1 et au Cor. 5, il est suffisant de montrer qu'aucune accumulation ne peut survenir dans les diagrammes produits par la machine support ayant uniquement deux méta-signaux : un de vitesse 0 et un de vitesse 1.

On considère maintenant sans perte de généralité une machine à signaux $\mathfrak{M}_2$ ayant uniquement deux méta-signaux : S et $\overrightarrow{\mathsf{R}}$ de vitesses respectives 0 et 1. Comme $\mathfrak{M}_2$ contient uniquement deux méta-signaux, il n'y a qu'une seule règle de collisions à définir et pour que $\mathfrak{M}_2$ soit effectivement une machine support, il s'agit nécessairement de la règle blanche $\{\overrightarrow{\mathsf{R}}, \mathsf{S}\} \to \{\mathsf{S}, \overrightarrow{\mathsf{R}}\}$.

Lemme 15
Soit $\mathcal{D}$ un diagramme espace-temps généré par la machine $\mathfrak{M}_2$ à partir d'une configuration initiale comprenant i signaux $\overrightarrow{\mathsf{R}}$ et j signaux S. Alors le nombre total de collisions se produisant dans $\mathcal{D}$ est inférieur ou égal à $i \times j$.

Démonstration. Commençons par remarquer que toute configuration initiale de $\mathfrak{M}_2$ peut être réarrangée en une configuration initiale qui maximise le nombre de collisions dans le diagramme engendré (voir la Fig. 4.4(a) pour un exemple de configuration quelconque et la Fig. 4.4(b) pour sa version arrangée). En effet, le nombre maximal de collisions est obtenu si chaque signal $\overrightarrow{\mathsf{R}}$ rentre en collision avec chaque signal S. Les signaux S étant stationnaires et les signaux $\overrightarrow{\mathsf{R}}$ se déplaçant vers la droite, cela est possible uniquement si chaque $\overrightarrow{\mathsf{R}}$ est initialement disposé à la gauche de chaque S. Donc pour calculer la borne supérieure exacte du nombre de collisions, on considère une configuration initiale similaire à celle de la Fig. 4.4(b) c'est-à-dire telle que la position de chaque signal $\overrightarrow{\mathsf{R}}$ est strictement inférieure aux positions de tous les signaux S.

Comme l'unique règle de collisions est une règle blanche *i.e.* tous les signaux entrants sont régénérés en sortie, chaque signal $\overrightarrow{\mathsf{R}}$ n'est pas détruit après la collision avec le premier signal S rencontré (celui qui est le plus à gauche) mais continue sa trajectoire et rentre successivement



(a) Diagramme de $\mathfrak{M}_2$ avec configuration initiale quelconque. (b) Maximiser le nombre de collisions.

FIGURE 4.4 – Diagrammes avec deux vitesses.

en collision avec tous les signaux S c'est-à-dire les j signaux S présents dans la configuration initiale. Après la dernière collision (avec le signal S le plus à droite), $\overrightarrow{\mathsf{R}}$ continue de se propager indéfiniment vers la droite et ne provoque plus de collision.

Donc chaque signal $\overrightarrow{\mathsf{R}}$ génère exactement j collisions. Il y a i signaux $\overrightarrow{\mathsf{R}}$ dans la configuration initiale, chacun rentrant en collision avec tous les signaux S, et comme les collisions entre signaux $\overrightarrow{\mathsf{R}}$ sont impossibles (car ils ont la même vitesse), le nombre total de collisions est donc $i \times j$. □

Remarque 17. Étant donnés n signaux dans la configuration initiale, le nombre maximal de collisions $i \times j$ est obtenu en prenant $i = \lfloor \frac{n}{2} \rfloor$ signaux $\overrightarrow{\mathsf{R}}$ et $j = n - i$ signaux S dans la configuration initiale de telle sorte que chaque signal $\overrightarrow{\mathsf{R}}$ est initialement disposé à la gauche de chaque signal S (voir la FIG. 4.4(b)).

On peut maintenant conclure :

PROPOSITION 16
Il est impossible de générer une accumulation avec une machine à signaux à 2 vitesses.

Démonstration. Comme expliqué précédemment, toute machine $\mathfrak{M}$ à 2 vitesses peut être normalisée en une machine $\mathfrak{M}^{0,1}$ à 2 vitesses (0 et 1). Toute machine de la forme $\mathfrak{M}^{0,1}$ admet $\mathfrak{M}_2$ comme machine support. Par le LEM. 15, $\mathfrak{M}_2$ exécutée sur une configuration initiale finie ne peut donner lieu qu'à un nombre fini de collisions. Une condition nécessaire (mais pas suffisante) pour avoir une accumulation étant de contenir une infinité d'objets (signaux/collisions) dans le diagramme, il s'ensuit que la machine $\mathfrak{M}_2$ ne peut pas produire d'accumulation. Le LEMME 1 et le COR. 5 impliquent que ni $\mathfrak{M}^{0,1}$ ni $\mathfrak{M}$ ne peuvent produire d'accumulation (car les diagrammes de $\mathfrak{M}$ sont équivalents à ceux de $\mathfrak{M}^{0,1}$, qui sont inclus dans les diagrammes supports sans accumulation de $\mathfrak{M}_2$). On conclut finalement qu'il est impossible d'accumuler des signaux et des collisions avec uniquement deux vitesses. □

Notons que dans le cas de deux vitesses, la rationalité des vitesses et des positions initiales n'intervient pas car la normalisation de deux vitesses distinctes a et b en vitesses 0 et 1 peut être réalisée pour n'importe quels nombres a et b, rationnels ou non, et les positions initiales n'ont pas besoin d'être rationnelles dans la preuve du LEM. 15). Par contre, l'*hypothèse de finitude de la configuration initiale* (voir DÉF. 3) est quant à elle indispensable.

En effet, il est toujours possible de créer une accumulation avec seulement deux vitesses si on considère des configurations initiales infinies : il suffit de placer des signaux de telle façon qu'ils forment une accumulation statique dans l'espace. Par exemple, le point 0 étant un point d'accumulation topologique de l'ensemble $\left\{ \frac{1}{n} \mid n \in \mathbb{N} \right\}$ dans $\mathbb{R}$, on obtiendra une accumulation

statique en 0 en plaçant un signal à chaque position $\frac{1}{n}$ (remarquons au passage qu'une telle configuration est rationnelle).

Il est ensuite aisé de générer une accumulation dynamique en plaçant un signal stationnaire à chaque position $\frac{1}{n}$ et un signal $\overrightarrow{\mathsf{R}}$ (de vitesse 1) en position -2. Le signal $\overrightarrow{\mathsf{R}}$ rentrera alors en collision avec tous les signaux stationnaires (pour une règle de collisions appropriée) avant le temps $t = 2$, produisant ainsi une accumulation de collisions au point $(0, 2)$, comme illustré en FIG. 4.5.

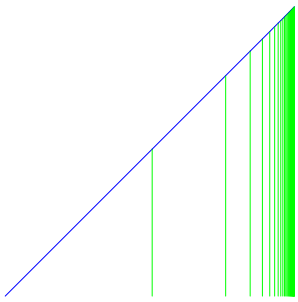


FIGURE 4.5 – Une accumulation à deux vitesses et configuration initiale infinie.

4.3 Accumulations à trois vitesses

Nous avons montré dans la section précédente que s'il est facile de produire une accumulation avec seulement quatre vitesses, cela est en revanche impossible lorsque seulement deux vitesses différentes sont permises. Nous allons maintenant étudier le cas intermédiaire — trois vitesses distinctes — et nous montrons que deux sous-cas doivent être distingués : l'un traitant de machines contenant des valeurs incommensurables (*i.e.* des valeurs de rapport irrationnel), et l'autre traitant de machines purement rationnelles. Le résultat obtenu est une condition nécessaire pour qu'une machine à trois vitesses puisse générer une accumulation :

pour qu'une machine à signaux à 3 vitesses puisse générer une accumulation à partir d'une configuration initiale, il faut qu'elle contienne un rapport irrationnel soit entre des vitesses, soit entre des positions initiales.

Dans un premier temps, nous montrons qu'il est effectivement possible de générer une accumulation avec une machine irrationnelle à 3 vitesses : nous en fournissons deux exemples (un avec des positions initiales de rapport irrationnel, et un avec des vitesses de rapport irrationnel). Nous démontrons dans un second temps qu'il est impossible pour une machine rationnelle à 3 vitesses de produire une accumulation. Nous utilisons pour cela les notions de *machines et diagrammes supports* introduites en SOUS-SEC. 2.4.2.

4.3.1 Cas non rationnel

Pour démontrer qu'il est possible de produire une accumulation avec trois vitesses dans le cas de machines irrationnelles, nous fournissons directement un tel exemple de machine et de configuration initiale. Pour cela, nous implémentons de manière géométrique l'*algorithme d'Euclide*, en utilisant la machine évoquée en SOUS-SEC. 3.1.2 qui permet de calculer le *modulo* de deux nombres, c'est-à-dire le reste dans leur division euclidienne. Le calcul géométrique du *modulo* permet, à partir d'une configuration dans laquelle deux valeurs réelles a et b sont encodées par des distances entre signaux verticaux, d'obtenir une configuration dans laquelle la valeur du reste de la division euclidienne de a par b est encodée de cette même façon. Pour implémenter géométriquement l'algorithme d'Euclide et calculer ainsi le *plus grand diviseur commun* de a et b, l'idée est, comme pour l'algorithme classique, de réitérer le calcul géométrique du *modulo*.

Les notions de division euclidienne, modulo et plus grand diviseur commun, habituellement définies sur les entiers, sont ici généralisées de manière canonique aux réels. Plus précisément, pour $a, b \in \mathbb{R}$, nous entendons par *division euclidienne (ou division entière) de a par b* l'unique $n \in \mathbb{Z}$ (égal à $\lfloor \frac{a}{b} \rfloor$) tel que $a = b \cdot n + r$ avec $r \in \mathbb{R}$ et $0 \leq r < |b|$. Le reste r définit la valeur de a *modulo* b, que l'on notera $a \bmod b$. On dira que le réel b divise le réel a si $a \bmod b = 0$. Le plus grand diviseur commun des réels a et b, noté $\mathrm{pgcd}(a, b)$, est le plus grand réel qui divise à la fois a et b. On ne considérera par la suite que des nombres réels positifs, les division, modulo et pgcd de réels de signes quelconques pouvant se ramener facilement à ceux de leur valeurs absolues.

Implémenter géométriquement l'algorithme d'Euclide avec trois vitesses. On pourrait facilement produire une accumulation en utilisant la machine support $\mathfrak{M}_3^{-1,0,1}$ exécutée depuis une configuration initiale bien choisie, mais nous préférons utiliser ici une autre machine à trois vitesses, appelée $\mathfrak{M}_{pgcd}$, pour produire l'accumulation recherchée. La définition de $\mathfrak{M}_{pgcd}$ permet d'implémenter l'algorithme d'Euclide avec des signaux et de produire ainsi des diagrammes plus clairs et significatifs que la machine support à trois vitesses.

Rappelons d'abord l'algorithme d'Euclide. À partir de deux nombres réels positifs a et b (on peut supposer sans perte de généralité que $b \leq a$, quitte à les échanger), on définit les suites $(a_n)_{n\in\mathbb{N}}$ et $(b_n)_{n\in\mathbb{N}}$ par la récursion suivante :

$$\begin{cases} a_0 = a \\ b_0 = b \end{cases} \quad \text{et} \quad \begin{cases} a_{n+1} = b_n \\ b_{n+1} = a_n \bmod b_n \end{cases} .$$

Cette double récursion permet également de définir la suite $(r_n)_{n\in\mathbb{N}}$ des restes des divisions euclidiennes successives c'est-à-dire que pour tout n, r_n est le reste de la division de a_n par b_n. Lorsque cette suite est nulle à partir d'un certain rang, le dernier reste non nul nous fournit alors $\mathrm{pgcd}(a, b)$, le plus grand diviseur commun de a et b.

Cet algorithme peut être implémenté avec une machine à signaux à 3 vitesses (de valeurs -1, 0 ou 1), et définie par 7 méta-signaux et 8 règles de collisions. La définition de la machine $\mathfrak{M}_{pgcd}$ est donnée par les Fig. 4.6(a) et 4.6(b) ; la Fig. 4.6(c) fournit le diagramme de l'exécution de $\mathfrak{M}_{pgcd}$ sur les entrées $a = 8$ et $b = 3$. Le résultat peut être lu au sommet du diagramme, la distance entre les deux signaux w_0 nous fournissant alors $\mathrm{pgcd}(8, 3)$.

Les méta-signaux stationnaires (w_0, w_a et w_b) sont utilisés pour encoder des valeurs : la valeur a (resp. b) étant codée par la distance entre les signaux w_0 et w_a (resp. entre w_0 et w_b). Dans cette version géométrique, l'étape de récurrence $a_{n+1} = b_n$ est implémentée par la règle $\{\mathsf{w}_b, \overleftarrow{\mathsf{ZAG}}\} \to \{\overleftarrow{\mathsf{ZAG}}, \mathsf{w}_a\}$ et l'étape $b_{n+1} = r_n$ par la règle $\{\overrightarrow{\mathsf{zig}}, \overleftarrow{\mathsf{ZAG}}\} \to \{\overleftarrow{\mathsf{zag}}, \mathsf{w}_b\}$.

Remarque 18. Notons qu'avec trois vitesses $-\nu_1 < 0 < \nu_2$ et qu'à partir d'une configuration initiale comprenant deux signaux verticaux et un autre signal se déplaçant (disons vers la droite) à partir du premier signal vertical en direction du deuxième, et avec des règles définissant des rebonds des signaux non stationnaires sur les stationnaires, on obtient un aller-retour dont le temps dépend à la fois de la distance l séparant les signaux verticaux et des deux vitesses non nulles. Ce temps est égal à la somme des temps mis par chaque signal pour aller d'un signal vertical au second et est donné par $\frac{1}{\nu_1}l + \frac{1}{\nu_2}l$. En notant $\tau = \frac{1}{\nu_1} + \frac{1}{\nu_2}$, le temps d'un aller-retour entre deux murs espacés d'une distance l est τl. Pour le cas de $\mathfrak{M}_{pgcd}$, les signaux non-stationnaires ont des vitesses de valeurs absolues égales à 1 et donc $\tau = 2$.

Pour deux nombres a et b, on note $c^{a,b}$ la configuration définie à partir de quatre signaux (dont seulement un est non-stationnaire) et respectant l'encodage des valeurs de a et b par les distances entre signaux stationnaires c'est-à-dire $c^{a,b} = \left\{\mathsf{w}_0@0,\ \overrightarrow{\mathsf{zig}}@0,\ \mathsf{w}_b@b,\ \mathsf{w}_a@a\right\}$. La distance entre w_0 et w_a (resp. w_b) est égale à a (resp. à b).

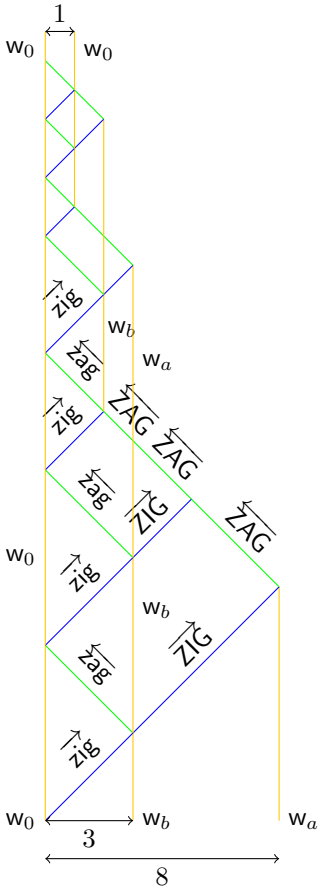
Décrivons brièvement l'évolution de $\mathfrak{M}_{pgcd}$ sur une telle configuration. Après que les signaux $\overrightarrow{\mathsf{ZIG}}$ (initialement $\overrightarrow{\mathsf{zig}}$ devenu $\overrightarrow{\mathsf{ZIG}}$ après la collision avec w_b) et $\overleftarrow{\mathsf{ZAG}}$ ont effectué un aller-retour

Méta-signal	Vitesse
$\overrightarrow{\mathsf{zig}}$, $\overrightarrow{\mathsf{ZIG}}$	-1
w_0, w_a, w_b	0
$\overleftarrow{\mathsf{zag}}$, $\overleftarrow{\mathsf{ZAG}}$	1

(a) Méta-signaux.

$$\{\ \overrightarrow{\mathsf{zig}},\ \mathsf{w}_b\ \} \to \{\ \overleftarrow{\mathsf{zag}},\ \mathsf{w}_b,\ \overrightarrow{\mathsf{ZIG}}\ \}$$
$$\{\ \mathsf{w}_0,\ \overleftarrow{\mathsf{zag}}\ \} \to \{\ \mathsf{w}_0,\ \overrightarrow{\mathsf{zig}}\ \}$$
$$\{\ \mathsf{w}_a,\ \overrightarrow{\mathsf{ZIG}}\ \} \to \{\ \overleftarrow{\mathsf{ZAG}}\ \}$$
$$\{\ \mathsf{w}_b,\ \overleftarrow{\mathsf{ZAG}}\ \} \to \{\ \overleftarrow{\mathsf{ZAG}},\ \mathsf{w}_a\ \}$$
$$\{\ \overrightarrow{\mathsf{zig}},\ \overleftarrow{\mathsf{ZAG}}\ \} \to \{\ \overleftarrow{\mathsf{zag}},\ \mathsf{w}_b\ \}$$
$$\{\ \overrightarrow{\mathsf{ZIG}},\ \overleftarrow{\mathsf{ZAG}}\ \} \to \{\ \overleftarrow{\mathsf{ZAG}}\ \}$$
$$\{\ \overrightarrow{\mathsf{zig}},\ \mathsf{w}_b,\ \overleftarrow{\mathsf{ZAG}}\ \} \to \{\ \overleftarrow{\mathsf{ZAG}},\ \mathsf{w}_0\ \}$$
$$\{\ \mathsf{w}_0,\ \overleftarrow{\mathsf{ZAG}}\ \} \to \{\ \mathsf{w}_0\ \}$$

(b) Règles de collisions.



(c) Exécution de $\mathfrak{M}_{pgcd}$ calculant $\mathrm{pgcd}(8,3) = 1$.

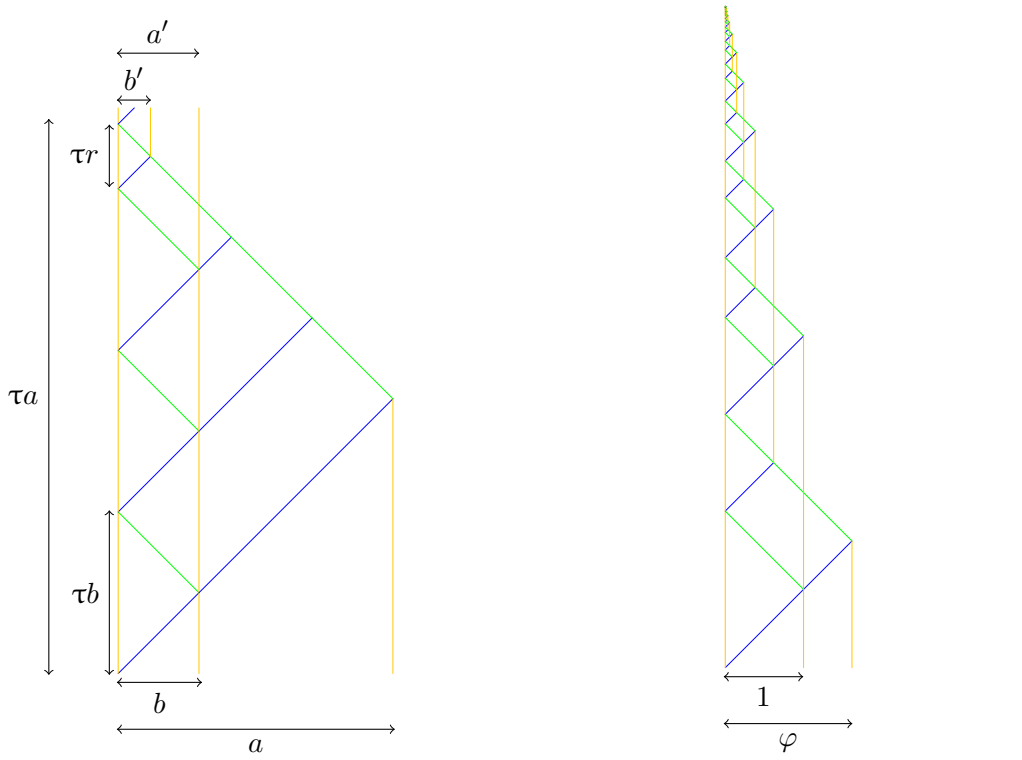
FIGURE 4.6 – La machine $\mathfrak{M}_{pgcd}$: algorithme d'Euclide géométrique.

complet entre w_0 et w_a (donc après une durée τa compte-tenu de la remarque précédente), la configuration vérifie le même schéma (même signaux et même ordre) que la configuration $c^{a,b}$ mais avec les signaux w_a et w_b à de nouvelles positions. En effet, selon les règles de collisions, le signal initial w_a a désormais disparu, et le w_b a été remplacé par w_a lors de la collision avec $\overleftarrow{\mathsf{ZAG}}$. La collision suivante entre $\overrightarrow{\mathsf{zig}}$ (qui rebondissait entre w_0 et w_b, devenant alternativement $\overrightarrow{\mathsf{zig}}$ et $\overleftarrow{\mathsf{zag}}$) et $\overleftarrow{\mathsf{ZAG}}$ génère un nouveau signal w_b. Le mur initial w_0 demeure toujours présent (puisque apparaissant toujours en sortie des collisions l'impliquant).

En appelant a' (resp. b') la distance entre w_0 et le nouveau signal w_a (resp. w_b), conformément à la FIG. 4.7(a), la configuration au temps τa est $c^{a',b'} = \left\{\mathsf{w}_0@0,\ \overrightarrow{\mathsf{zig}}@0,\ \mathsf{w}_b@b',\ \mathsf{w}_a@a'\right\}$. En définissant r comme la distance entre w_0 et le signal nouveau w_b généré, on a :

$$\begin{aligned} & \tau a = k \cdot \tau b + \tau r && \text{où } k \in \mathbb{N} \text{ et } 0 \leq \tau r < \tau b \\ \text{d'où :} \quad & \tau r = \tau a \bmod \tau b && \text{(par définition du modulo)} \\ \text{et donc :} \quad & r \ = a \bmod b \ . \end{aligned}$$

On obtient donc finalement que $a' = b$ et $b' = r = a \bmod b$ et qu'à partir de la configuration $c^{a,b}$, la configuration après une durée τa est $c^{a',b'} = c^{b,a \bmod b}$. Si $r = 0$, cela signifie que a est divisible par b et que la collision entre $\overleftarrow{\mathsf{ZAG}}$ et w_b implique également $\overrightarrow{\mathsf{zig}}$: dans ce cas, l'application de la dernière règle donnée en FIG. 4.6(b) produit deux signaux w_0 alors séparés par la distance b.



(a) Détail d'une étape de calcul de $\mathfrak{M}_{pgcd}$.

(b) Une accumulation avec 3 vitesses.

FIGURE 4.7 – Générer une accumulation à trois vitesses avec $\mathfrak{M}_{pgcd}$.

Les règles de $\mathfrak{M}_{pgcd}$ définissant une itération de cette étape lorsque la valeur r précédemment obtenue est non nulle (c'est-à-dire lorsque la collision entre $\overleftarrow{\mathsf{ZAG}}$ et w_b n'implique pas $\overrightarrow{\mathsf{zig}}$), l'étape de calcul est recommencée mais cette fois à partir de la configuration $c^{b,a \bmod b}$. En commençant avec les valeurs $a = a_0$ et $b = b_0$, nous définissons les suites $(a_n)_{n\in\mathbb{N}}$ et $(b_n)_{n\in\mathbb{N}}$ en identifiant à chaque étape du processus les distances a, a', b et b' respectivement avec les valeurs a_n, a_{n+1}, b_n et b_{n+1}. Les suites ainsi obtenues correspondent alors à celles définies par la double récurrence de l'algorithme d'Euclide. S'il existe $n_0 \in \mathbb{N}$ tel que a_{n_0} est divisible par b_{n_0}, cela signifiera que le reste r associé à cette étape de calcul sera nul et qu'il y aura une triple collision entre les signaux $\overleftarrow{\mathsf{ZAG}}$, w_b et $\overrightarrow{\mathsf{zig}}$. Comme mentionné ci-dessus, les deux dernières règles de collisions de la FIG. 4.6(b) provoqueront alors l'arrêt du processus et ne laisseront dans la machine que deux signaux stationnaires w_0. La distance entre ces deux signaux correspond alors au dernier reste non-nul et fournit donc le résultat recherché, à savoir $\mathrm{pgcd}(a_0, b_0) = \mathrm{pgcd}(a, b)$.

Arrêt de l'algorithme. Afin d'obtenir une accumulation, il faut par définition qu'un nombre infini de collisions se produise en temps fini. Dans le cas de l'algorithme d'Euclide, cela signifie que l'algorithme ne s'arrête jamais. Il suffit pour cela d'utiliser deux réels *incommensurables*, c'est-à-dire tels que leur rapports est irrationnel. En effet, nous savons à partir de [Hardy et Wright 1960, Th. 161, p. 136] que :

$$\textit{l'algorithme d'Euclide termine sur les entrées } a \textit{ et } b \iff \frac{a}{b} \in \mathbb{Q} \ .$$

Pour obtenir ainsi une exécution infinie, il suffit donc de choisir deux valeurs initiales de rapport irrationnel. Nous prenons ici $a = \varphi$ et $b = 1$, où $\varphi = \frac{1+\sqrt{5}}{2}$ est le nombre d'or. La

configuration initiale correspondante est $c_0^{\varphi,1} = \left\{\mathsf{w}_0@0,\ \overrightarrow{\mathsf{zig}}@0,\ \mathsf{w}_b@1,\ \mathsf{w}_a@\varphi\right\}$.

Comme le rapport $\frac{a}{b} = \varphi$ est irrationnel, l'algorithme ne s'arrêtera jamais et produira une suite de restes $(r_n)_{n\in\mathbb{N}}$ qui sera strictement décroissante et strictement positive. De plus, comme φ satisfait $\varphi = 1+\frac{1}{\varphi}$, nous pouvons facilement obtenir le développement de φ en fraction continue :

$$\varphi = 1 + \cfrac{1}{1 + \cfrac{1}{1+\cfrac{1}{\cdots}}}$$

conformément au lien classique entre fraction continue et algorithme d'Euclide [Hardy et Wright 1960]. Ici, on aura pour tout n, $a_n = b_n + r_n$, c'est-à-dire que le quotient de a_n par b_n est égal à 1 pour tout n.

Convergence de la série des a_n et limite de la suite des temps de collisions. L'algorithme exécuté à partir de $c_0^{\varphi,1}$ produit donc bien un diagramme infini, mais pour qu'il y ait effectivement une accumulation, il faut qu'une infinité de collisions soit contenue dans une durée finie. À partir des définitions de $a_{n+1} = b_n$ et $b_{n+1} = r_n$ et de la relation $a_n = b_n \cdot q_n + r_n$ pour tout $n \in \mathbb{N}$, nous pouvons prouver que pour tout n on a $a_{n+2} < \frac{1}{2} \cdot a_n$. En effet:

$$\begin{aligned}
a_n &= b_n \cdot q_n + r_n \\
&= a_{n+1} \cdot q_n + b_{n+1} && \text{car } a_{n+1} = b_n \text{ et } b_{n+1} = r_n\,, \\
&= (b_{n+1} \cdot q_{n+1} + r_{n+1}) \cdot q_n + b_{n+1} \\
&= b_{n+1}(1 + q_{n+1} \cdot q_n) + r_{n+1} \cdot q_n \\
&= a_{n+2}(1 + q_{n+1} \cdot q_n) + r_{n+1} \cdot q_n && \text{car } a_{n+2} = b_{n+1}\,, \\
&\geq 2a_{n+2} + r_{n+1} && \text{car } \forall n,\ q_n \geq 1\ .
\end{aligned}$$

Or, avec les valeurs initiales $a_0 = \varphi$ et $b_0 = 1$, comme $\frac{a_0}{b_0}$ est irrationnel, l'algorithme ne termine pas et pour tout n, on a $r_n > 0$ et $q_n = 1$. La dernière inégalité ci-dessus implique alors que $a_n > 2 \cdot a_{n+2}$ et donc $a_{n+2} < \frac{1}{2} \cdot a_n$. Nous pouvons ensuite montrer par une simple récurrence sur n que $\forall n \geq 1,\ a_{2n} < \frac{1}{2^n} \cdot a_0$ et $\forall n \geq 1,\ a_{2n+1} < \frac{1}{2^n} \cdot a_1$, et il s'ensuit que pour tout $k \in \mathbb{N}$:

$$\begin{aligned}
\sum_{n=0}^{k} a_n = &\sum_{n=0}^{\lfloor k/2 \rfloor} a_{2n} + \sum_{n=0}^{\lfloor k/2 \rfloor} a_{2n+1} \\
< &\sum_{n=0}^{\lfloor k/2 \rfloor} \frac{1}{2^n} a_0 + \sum_{n=0}^{\lfloor k/2 \rfloor} \frac{1}{2^n} a_1 \\
< &\ (a_0 + a_1) \times \sum_{n=0}^{\lfloor k/2 \rfloor} \frac{1}{2^n}
\end{aligned}$$

ce qui donne finalement (et avec $a_1 = b_0$ par définition) :

$$\lim_{k\to\infty} \sum_{n=0}^{k} a_i < 2 \times (a_0 + b_0)\ .$$

Comme tous les a_n sont strictement positifs, la série est strictement croissante et la somme infinie $\sum_{n=0}^{\infty} a_n$ converge vers une limite finie inférieure ou égale à $2 \times (a_0 + b_0)$.

Notons également qu'à chaque itération, au moins une collision se produit (en fait, au moins trois), et donc pour chaque durée a_n, il se produit au moins une collision. Plus exactement, pour $n \in \mathbb{N}$, il existe une collision de coordonnées $(0, a_n)$ (celle entre w_0 et $\overleftarrow{\mathsf{zag}}$). De plus, entre les

temps a_n et a_{n+1}, il y a seulement un nombre fini de collisions. La somme totale $\tilde{t}$ de ces durées est donnée par $\tilde{t} = \sum_{n=0}^{\infty} \tau a_n$ et correspond à la hauteur totale de la construction. Comme démontré ci-dessus, la somme des a_n est bornée par $2 \times (a_0 + b_0)$, et donc $\tilde{t}$ est bornée par $2\tau \times (a_0 + b_0)$. Ici, nous avons $\tau = 2$, $a_0 = \varphi$ et $b_0 = 1$ et donc $\tilde{t} < 4 \times (\varphi + 1)$.

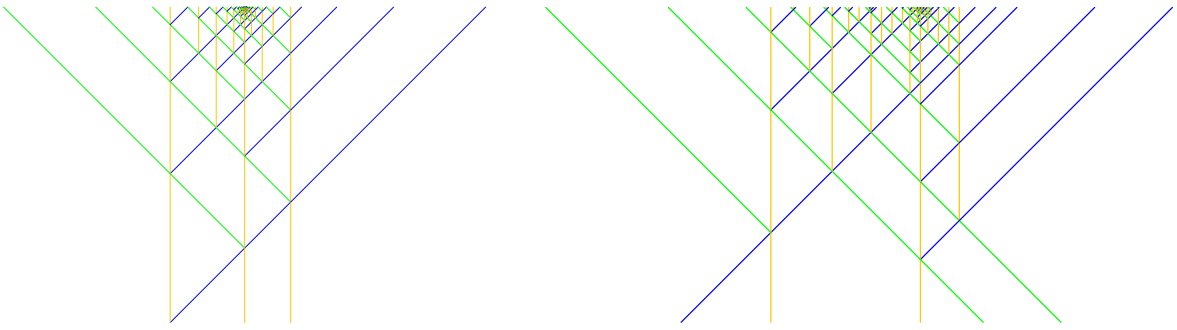
Nous pouvons maintenant conclure qu'avant le temps fini $\tilde{t}$, il se produit un nombre infini de collisions car une collision se produit en $(x = 0, t = a_n)$ avec $a_n < \tilde{t}$ pour tout n. Cela implique par définition de la suite de collisions $(C_n)_{n\in\mathbb{N}}$ (voir la DÉF. 13), que la suite des temps de collision est convergente. Par le LEM. 12, le diagramme de $\mathfrak{M}_{pgcd}$ à partir de $c_0^{\varphi,1}$ contient donc une accumulation aux coordonnées $(0, \tilde{t})$ *i.e.* $c_{\tilde{t}}(0) = ✺$.

De manière générale, en utilisant la propriété que l'algorithme d'Euclide termine sur les entrées a et b si et seulement si le rapport $\frac{a}{b}$ est irrationnel, nous obtenons finalement le lemme suivant :

LEMME 17
La machine $\mathfrak{M}_{pgcd}$ génère une accumulation lorsqu'elle est exécutée à partir de la configuration $c_0^{a,b}$ avec $\frac{a}{b} \notin \mathbb{Q}$.

Remarque 19. En vertu de ce lemme, φ peut donc être remplacé par n'importe quelle autre valeur irrationnelle x (supérieure à 1) : la machine $\mathfrak{M}_{pgcd}$ exécutée sur $c_0^{x,1}$ produira également une accumulation. Dans l'exemple précédent, nous avons choisi la valeur φ à cause de la régularité du diagramme résultant, régularité provenant du fait que les divisions euclidiennes successives obtenues en commençant avec φ et 1 satisfont $q_n = 1$ pour tout n.

Nous pouvons également fournir une accumulation en utilisant $\mathfrak{M}_3 = \widehat{\mathfrak{M}_{pgcd}}$, la machine support de $\mathfrak{M}_{pgcd}$ définie avec seulement trois méta-signaux. À partir de la configuration $\widehat{c_0^{\varphi,1}}$, la configuration support de $c_0^{\varphi,1}$, nous obtenons le diagramme de la FIG. 4.8(a), dans lequel une accumulation se produit en position $x = 1$ et strictement avant le temps $\tilde{t}$ précédemment calculé. La FIGURE 4.8(b) fournit un autre exemple de diagramme support pour $\mathfrak{M}_{pgcd}$ exécutée à partir d'une configuration initiale incluant un rapport irrationnel.



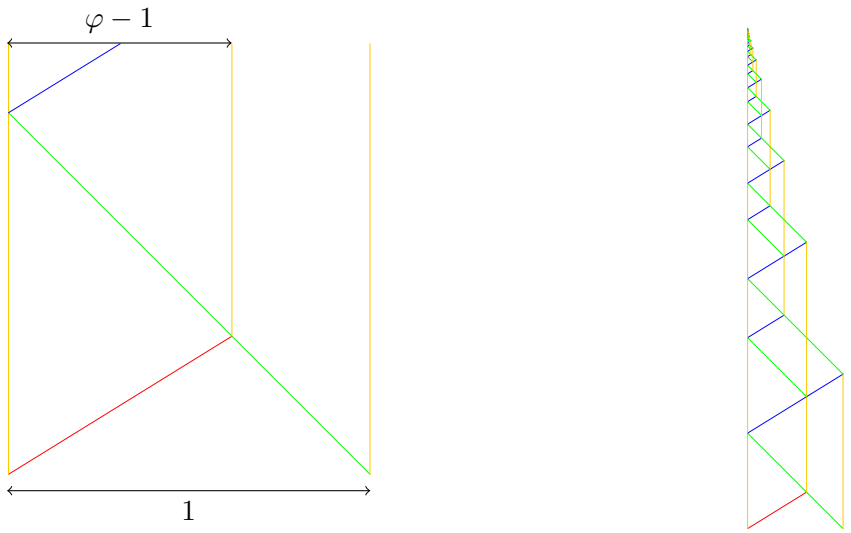
(a) Diagramme support pour la configuration $c_0^{\varphi,1}$. (b) Un autre exemple de diagramme support.

FIGURE 4.8 – Diagrammes supports avec accumulation pour $\mathfrak{M}_{pgcd}$.

Accumulation à partir de vitesses à rapport irrationnel. L'exemple précédant donne un exemple d'accumulation en utilisant une machine ayant toutes ses vitesses rationnelles (-1, 0 et -1 pour $\mathfrak{M}_{pgcd}$) et une configuration initiale à rapport irrationnel. Nous montrons ici qu'il est aisé d'obtenir une accumulation à partir d'une configuration initiale rationnelle et d'une machine dont deux vitesses ont un rapport irrationnel. Il suffit pour cela d'utiliser ces deux vitesses pour

créer un signal à une position irrationnelle de telle sorte à obtenir une configuration de la forme $c_0^{a,b}$ avec $\frac{a}{b} \notin \mathbb{Q}$.

Pour cela, nous utilisons la machine $\mathfrak{M}_{pgcd}$ dans laquelle la vitesse 1 est remplacée par la vitesse $\varphi \in \mathbb{R} \setminus \mathbb{Q}$. Cette machine modifiée implémentera encore l'algorithme d'Euclide. En effet les règles de collisions restent identiques et le changement de la vitesse 1 en φ n'aura comme conséquence que de changer le temps d'un aller-retour entre deux signaux verticaux. Alors qu'on avait pour la version originale de $\mathfrak{M}_{pgcd}$ la valeur $\tau = 2$, nous aurons pour la version modifiée $\tau' = 1 + \frac{1}{\varphi} = \varphi$. Les étapes de l'algorithme resteront alors les mêmes, puisque les restes successivement obtenus sont indépendants de τ. En utilisant le fait que le rapport des vitesses est $\frac{-1}{\varphi} \notin \mathbb{Q}$, nous construisons à partir d'une configuration rationnelle une configuration ayant une rapport irrationnel. Après deux collisions, la configuration irrationnelle ainsi produite peut alors être utilisée pour démarrer une exécution infinie de l'algorithme d'Euclide. En effet, tel qu'illustré par la FIG. 4.9(a), en partant de la configuration rationnelle $c_0 = \left\{\mathsf{w}_0@0,\ \overrightarrow{\mathsf{start}}@0,\ \overleftarrow{\mathsf{zag}}@1,\ \mathsf{w}_a@1\right\}$, un simple calcul montre que la première collision se produit en position $x = \frac{1}{1+\varphi} = \frac{1}{\varphi} = \varphi - 1$, et qu'après la deuxième collision, la configuration au temps 1 est $c_1^{1,\varphi-1} = \left\{\mathsf{w}_0@0,\ \overrightarrow{\mathsf{zig}}@0,\ \mathsf{w}_b@\varphi - 1,\ \mathsf{w}_a@1\right\}$, qui vérifie alors la forme requise pour effectuer l'algorithme d'Euclide. Comme $\frac{1}{\varphi-1} \notin \mathbb{Q}$, l'algorithme ne terminera pas, et la suite $(a_n)_{n\in\mathbb{N}}$ est la même que dans l'exemple précédent, mais décalée d'un rang à cause de l'étape d'initialisation des distances qui fournit comme valeurs d'entrée $a_0 = 1$ et $a_1 = \varphi - 1$ (alors que dans l'exemple précédent, nous avions $a_0 = \varphi$, $a_1 = 1$ et $a_2 = \varphi - 1$). Une accumulation est ainsi générée en position $x = 0$ et temps $\tilde{t}' = 1 + \tau' \sum_{n=1}^{\infty} a_n$. Le diagramme complet est donné par la FIG. 4.9(b).



(a) Créer une position irrationnelle à partir d'une vitesse irrationnelle. (b) Le diagramme complet.

FIGURE 4.9 – Une accumulation à trois vitesses dont une de valeur irrationnelle.

Avec le LEM. 17, nous obtenons finalement la proposition suivante :

PROPOSITION 18
Il existe des accumulations générées par des machines à signaux à 3 vitesses avec des vitesses incommensurables ou des positions initiales incommensurables.

4.3.2 Cas rationnel

Nous allons maintenant montrer que l'existence d'un rapport irrationnel entre deux valeurs est une condition nécessaire pour qu'une machine à 3 vitesses puisse générer une accumulation. Pour cela, nous considérons des machines à 3 vitesses de rapports entre elles rationnels, exécutées sur des configurations initiales dont les rapports des positions sont aussi rationnels. Nous montrons que de telles machines ne peuvent produire que des diagrammes ayant des structures en forme de grilles périodiques et dont la régularité implique l'inexistence d'accumulation.

Normalisation des vitesses. De la même façon que les vitesses ont été normalisées en 0 et 1 pour les machines à 2 vitesses en SOUS-SEC. 4.2.2, nous montrons ici que les vitesses de machines à 3 vitesses distinctes peuvent être normalisées en valeurs -1, 0 et $\nu > 0$.

Soit $\mathfrak{M}^{a,b,c}$ une machine à signaux à 3 vitesses a, b et c telles que $a < b < c$. Nous définissons alors la machine $\mathfrak{M}^{-1,0,\nu}$, qui produira des diagrammes équivalents à ceux $\mathfrak{M}^{a,b,c}$. Considérons la fonction $f : \mathbb{R} \to \mathbb{R}$ définie en SOUS-SEC. 1.3.1, avec $c = -1$ et $d = 0$: $f(x) = \frac{x}{b-a} - \frac{b}{b-a}$. Nous avons alors $f(a) = -1$, $f(b) = 0$ et $f(c) = \frac{c-b}{b-a} > 0$. Notons $\nu = f(c)$.

Comme f est une fonction affine de ratio strictement positif, on déduit par le LEM. 1 que les diagrammes de $\mathfrak{M}^{a,b,c}$ et $\mathfrak{M}^{-1,0,\nu}$ seront équivalents. En particulier, si $\mathfrak{M}^{a,b,c}$ produit un diagramme avec une accumulation, le diagramme correspondant de $\mathfrak{M}^{-1,0,\nu}$ contiendra également une accumulation.

Dans cette sous-section, nous voulons prouver qu'aucune machine à signaux rationnelle à 3 vitesses ne peut produire d'accumulation. En fait, nous prouvons une propriété légèrement plus forte : aucune machine à signaux à 3 vitesses dont les rapports entre les vitesses et les rapports entre les positions initiales sont rationnels, ne produit d'accumulation. Pour cela, pour une machine $\mathfrak{M}^{a,b,c}$, nous ne considérerons que la machine support $\widehat{\mathfrak{M}^{-1,0,\nu}}$ de la machine $\mathfrak{M}^{-1,0,\nu}$, où $\nu = \frac{c-b}{b-a}$. Par définition des machines supports, la machine $\widehat{\mathfrak{M}^{-1,0,\nu}}$ ne contient que 3 méta-signaux : un pour chaque vitesse -1, 0 et ν. Chaque règle de collisions produira en sortie le nombre maximal de méta-signaux, c'est-à-dire les trois méta-signaux de la machine.

Notons que si les rapports entre les valeurs a, b et c sont rationnels (*i.e.* a, b et c sont commensurables), alors ν sera également rationnel. En effet, tous nombres réels x et y non nuls vérifient l'équivalence suivante :

$$\frac{x}{y} \in \mathbb{Q} \Longleftrightarrow \frac{x-y}{y} \in \mathbb{Q} \ .$$

Comme $\nu = \frac{c-b}{b-a} = \frac{c}{b-a} - \frac{b}{b-a}$ et que a, b et c sont tous distincts, on en déduit que

$$\frac{a}{b}, \frac{a}{c} \in \mathbb{Q} \Rightarrow \nu \in \mathbb{Q}$$

(la rationalité de $\frac{b}{c}$ se déduisant des deux autres rapports). Donc si les rapports entre a, b et c sont rationnels, ν sera rationnel (et strictement positif par sa définition). Nous exprimerons parfois ν sous forme de fraction irréductible que nous écrirons $\nu = \frac{p}{q}$, où p et q sont des entiers naturels premiers entre eux. Pour simplifier les notations, on notera $\mathfrak{M}_3^{\nu}$ ou $\mathfrak{M}_3^{p/q}$ pour désigner la machine support $\widehat{\mathfrak{M}^{-1,0,\nu}}$.

Notions de bandes et multi-bandes. Nous construisons une famille de diagrammes telle que tout diagramme généré par une machine à trois vitesses de rapports rationnels est inclus dans au moins un diagramme de cette famille. Comme aucun diagramme de la famille ne contient d'accumulation, nous en déduirons qu'aucune machine à 3 vitesses de rapports rationnels ne peut produire d'accumulation. Cela est effectué en deux étapes. D'abord, nous introduisons un

certain type de configurations initiales ne produisant que des diagrammes réguliers à partir d'un certain temps : nous appelons de tels des diagrammes des *multi-bandes*. Ensuite, nous prouvons que toute configuration initiale rationnelle c_0 de la machine support $\mathfrak{M}_3^\nu$ est incluse dans une des configurations de la forme précédente et que le diagramme généré à partir de c_0 est inclus dans une multi-bande.

Avant de définir les diagrammes que nous appelons *multi-bandes*, nous introduisons les *bandes*, des structures plus simples qui seront utilisées comme éléments de bases pour construire les multi-bandes.

Définition 36 (Bande)
Soient $n \in \mathbb{N}$, $\nu \in \mathbb{Q}$ et $x_0, w \in \mathbb{R}$. Nous appelons $(\nu, n, x_0, w) -$ bande le diagramme généré par la machine $\mathfrak{M}_3^\nu$ à partir de la configuration initiale :

$$c_0 = \left\{ [\overleftarrow{\mathsf{L}}, \overrightarrow{\mathsf{R}}]@x_0,\ \mathsf{S}@x_0 + \frac{i}{n} \cdot w,\ [\overleftarrow{\mathsf{L}}, \overrightarrow{\mathsf{R}}]@x_0 + w \;\middle|\; 0 \leq i \leq n \right\} .$$

La Figure 4.10(b) fournit un exemple de $(\frac{2}{3}, 5, 0, 1) -$ bande (le diagramme a été coupé sur les deux côtés, mais les signaux quittant la partie centrale se propagent indéfiniment).

Le paramètre ν est une vitesse rationnelle (qui correspondra à la vitesse obtenue par la normalisation précédente) ; x_0 est la position de début de la bande *i.e.* la position du signal stationnaire le plus à gauche ; et w est la largeur totale d'une bande (la position dernier signal stationnaire est donc $x_0 + w$). Le paramètre n est le nombre de sous-divisions de l'intervalle $[x_0; x_0 + w]$ en segments de même longueur, égale à $\frac{w}{n}$). Nous montrerons par la suite que lorsque le paramètre est « bien choisi » en fonction de ν, alors il correspond exactement au nombre de telles sous-divisions créées par l'évolution la machine sur une configuration initiale ne contenant pas les signaux stationnaires marquant les sous-divisions, comme illustré par la Fig. 4.10(a). Pour un tel n, les collisions entre $\overrightarrow{\mathsf{R}}$ et $\overleftarrow{\mathsf{L}}$ se produiront exactement aux positions des sous-divisions et seront donc des triples collisions impliquant également le signal S marquant la sous-division. La Figure 4.11(a) illustre la signification géométrique de ces paramètres.

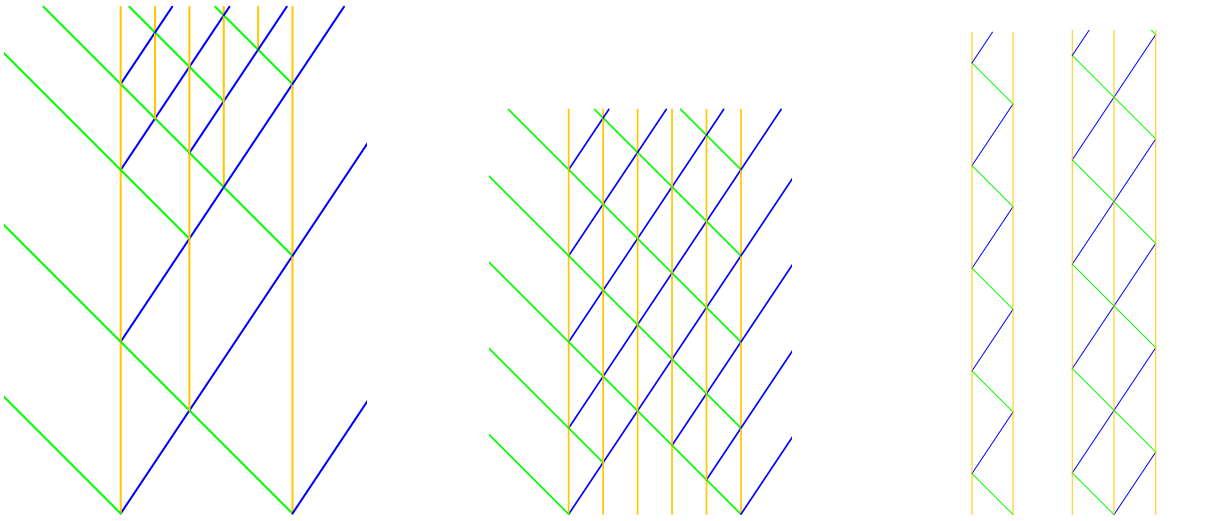
Il y a donc $n - 1$ signaux S entre les deux signaux S les plus extrêmes, et leur position est $x_0 + \frac{i}{n} \cdot w$ pour $1 \leq i \leq n - 1$. En incluant les positions des deux signaux extrêmes, un signal S est placé dans la configuration initiale à chaque position $x_0 + \frac{i}{n}$ pour $0 \leq i \leq n$.

Nous prouvons d'abord grâce à deux lemmes que la structure d'une bande est *régulière* : la partie centrale de la bande se comporte suivant un motif périodique et la partie externe contient uniquement des signaux parallèles qui n'entreront jamais en collision.

Lemme 19
Soit une $(\nu, n, x_0, w) -$ bande. Alors aucune collision ne se produit en dehors de l'intervalle $[x_0; x_0 + w]$.

Démonstration. Tout signal traversant le signal S le plus à gauche (resp. le plus à droite) de la configuration initiale (en position x_0) est nécessairement un signal $\overleftarrow{\mathsf{L}}$ (resp. un signal $\overrightarrow{\mathsf{R}}$). D'après la forme de la configuration initiale, il n'y a pas de signal avant (resp. après) la position x_0 (resp. $x_0 + w$) et comme les signaux partant à gauche (resp. à droite) du premier (resp. dernier) signal S sont tous parallèles, aucune collision ne se produira dans l'espace situé avant la position x_0 (resp. $x_0 + w$). □

Nous montrons maintenant que pour des paramètres judicieusement choisis, une bande correspond à une structure régulière constituée d'une grille verticale et de signaux parallèles s'en échappant sur les côtés.



(a) Une bande « incomplète ».

(b) Une bande.

(c) Zig-zags sur une et deux sous-divisions.

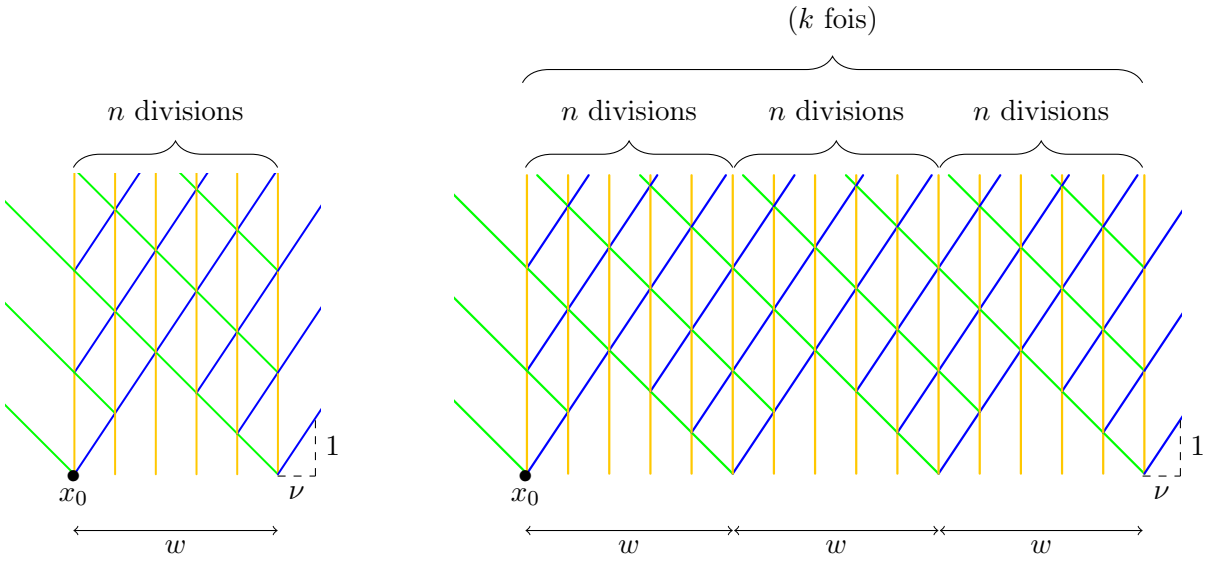
FIGURE 4.10 – Structures de base pour les diagrammes à trois vitesses.

LEMME 20
Toute $(\frac{p}{q}, p+q, x_0, w) -$ *bande devient périodique dans l'intervalle* $[x_0; x_0+w]$ *après le temps* $t_T = \frac{q}{p+q}w$. *Après ce temps, la période est donnée par* $T = \frac{w}{p}$.

Ce lemme signifie que pour $t \geq t_T$ et pour tout $x \in [x_0; x_0+w]$ on a : $c_{t+T}(x) = c_t(x)$. Notons d'abord qu'il est impossible d'avoir une périodicité sur l'espace complet à cause des signaux se propageant en dehors de la partie centrale de la bande. C'est pourquoi nous restreignons l'étude de l'évolution de la bande à l'intervalle défini par les signaux extrémaux, c'est-à-dire l'intervalle $[x_0; x_0+w]$.

Démonstration. Commençons par une remarque. Dans le cas de trois signaux S tels que les deux premiers et les deux derniers sont espacés d'une même distance, les temps d'aller-retour de signaux $\overleftarrow{\mathsf{L}}$ et $\overrightarrow{\mathsf{R}}$ partant en même temps du S central sont les mêmes, comme illustré par la FIG. 4.10(c). En effet, pour deux signaux stationnaires espacés d'une distance l et un signal $\overrightarrow{\mathsf{R}}$ de vitesse ν partant d'un des S, il faudra une durée νl pour que $\overrightarrow{\mathsf{R}}$ atteigne le second S. Après la collision, $\overrightarrow{\mathsf{R}}$ rebondit en un signal $\overleftarrow{\mathsf{L}}$ de vitesse -1 (un autre signal $\overrightarrow{\mathsf{R}}$ est également généré et part de l'autre côté). Le temps mis par $\overleftarrow{\mathsf{L}}$ pour atteindre S est de l. Le temps total de l'aller-retour est donc $\nu l + l = \tau \cdot l$ en posant $\tau = 1+\nu$. L'aller-retour symétrique (d'abord $\overleftarrow{\mathsf{L}}$ puis S) nécessite la même durée $\tau \cdot l$. Ainsi, à partir d'une configuration $\{\mathsf{S}@x, [\overleftarrow{\mathsf{L}}, \mathsf{S}, \overrightarrow{\mathsf{R}}]@x+l, \mathsf{S}@x+2l\}$, après une durée $\tau \cdot l$ (*i.e.* le temps d'un aller-retour), il se produit une triple collision au niveau du signal S central (en position $x+l$), comme cela est illustré par la FIG. 4.10(c). Comme la machine considérée est une machine support, trois signaux figurent en sortie, de telle sorte que le processus de double zig-zag se répète indéfiniment. Cette remarque s'applique également pour plus de trois signaux stationnaires régulièrement espacés. Dans le cas d'une $(\frac{p}{q}, p+q, x_0, w) -$ bande, les paramètres étant $\nu = \frac{p}{q}$ et $n = p+q$ et la distance entre deux signaux S successifs étant donnée par $\frac{w}{n}$, le temps d'un aller-retour sera donc égal à $\tau \cdot \frac{w}{n} = (1+\frac{p}{q}) \cdot \frac{w}{p+q} = \frac{w}{q}$.

Afin d'affirmer que toutes les collisions entre signaux gauche et droit se produisent en des positions de signaux stationnaires, il reste à s'assurer que le signal $\overrightarrow{\mathsf{R}}$ de gauche et $\overleftarrow{\mathsf{L}}$ de droite de la configuration initiale se rencontre bien au niveau d'un signal stationnaire. Cette collision aura bien lieu (car chacun de ces deux signaux peut rencontrer des signaux S mais par définition des règles, il sera également en sortie de collision et continuera donc de traverser la bande). Calculons



(a) Paramètres d'une (ν, n, x_0, w) – bande.

(b) Paramètres d'une (ν, n, x_0, w, k) – multi-bande.

FIGURE 4.11 – Paramètres des bandes et des multi-bandes.

les coordonnées de la collision centrale entre les signaux $\overrightarrow{\mathsf{R}}$ et $\overleftarrow{\mathsf{L}}$ initialement présents en positions respectives x_0 et $x_0 + w$. Les coordonnées (x_C, t_C) de cette collision vérifient $\nu \cdot t_C + x_0 = -t_C + x_0 + w$ et $x_C = \nu \cdot t_C + x_0$, ce qui donne $x_C = \frac{\nu}{\nu+1} w + x_0$ et $t_C = \frac{w}{\nu+1}$. Avec les paramètres $\nu = \frac{p}{q}$, on obtient $x_C = \frac{p}{p+q} w + x_0$ et $t_C = \frac{q}{p+q} w$. Comme $n = p + q$, la position x_C s'écrit $\frac{p}{n} w + x_0$ et est donc bien de la forme $x_0 + \frac{i}{n} w$ avec $0 \leq i \leq n$. Cela correspond donc bien à la position d'un signal S de la configuration initiale, assurant que la collision sera bien une triple collision.

Avec la remarque faite en début de preuve, nous en concluons qu'à partir du temps t_C, toutes les collisions (exceptées celles situées sur les bords) sont des collisions triples et le diagramme devient périodique entre les positions x_0 et $x_0 + w$. Après le temps $t_T = t_C$, la période est trivialement donnée par le temps d'un aller-retour : $T = \frac{w}{q}$. □

Concernant les parties « externes » à la bande, des signaux sont générés avec un espacement régulier et se propagent indéfiniment. Sur la partie gauche, les signaux $\overleftarrow{\mathsf{L}}$ (qui se propagent vers la gauche) sont tous espacés d'une distance $d = \tau \times \frac{w}{p+q} \times 1 = \frac{\tau \cdot w}{p+q}$, qui correspond à la distance parcourue par un signal $\overleftarrow{\mathsf{L}}$ pendant le temps d'un aller-retour d'une sous-division. À droite, les signaux $\overrightarrow{\mathsf{R}}$ qui se propagent à l'infini sont régulièrement espacés d'une distance $d = \frac{\tau \cdot w \cdot p}{q(p+q)}$.

Le paramètre n choisi ici $(n = p + q)$ correspond en fait à $1/\operatorname{pgcd}\left(\frac{\nu}{\nu+1}, 1\right)$, valeur que l'on peut déduire de l'étude des coordonnées de la collision C. Le choix de n'importe quel multiple de $1/\operatorname{pgcd}\left(\frac{\nu}{\nu+1}, 1\right)$ comme valeur de n permet également d'obtenir une bande stable après le temps t_T, la différence étant que les sous-divisions seront plus ou moins fines en fonction du multiple choisi.

À partir des bandes que nous pouvons utiliser comme structures élémentaires, nous définissons maintenant une structure périodique plus générale :

DÉFINITION 37 (Multi-bande)
Soient $n, k \in \mathbb{N}$, $\nu \in \mathbb{Q}$ *et* $x_0, w \in \mathbb{R}$. *On appelle* (ν, n, x_0, w, k) – multi-bande *le diagramme*

généré par $\mathfrak{M}_3^\nu$ *à partir de la configuration :*

$$c_0 = \left\{ [\mathsf{S}, \overleftarrow{\mathsf{L}}, \overrightarrow{\mathsf{R}}]@x_0 + l \cdot w,\ \mathsf{S}@x_0 + \left(\frac{i}{n} + j\right) \cdot w \;\middle|\; 0 \leq l \leq k,\ 0 < i < n \text{ et } 0 \leq j < k \right\} .$$

Comme dans le cas d'une bande, les paramètres d'une multi-bande ont un sens géométrique, qui est illustré par la FIG. 4.11. Intuitivement, une (ν, n, x_0, w, k) – multi-bande correspond à k copies d'une (ν, n, x_0, w) – bande, juxtaposées côte-à-côte de telle sorte que leurs signaux extrémaux se superposent. Comme toutes les copies ont les mêmes paramètres w (leur largeur) et n (leur nombre de sous-divisions), toutes leurs sous-divisions sont de même largeur $\frac{w}{n}$. Une $(\nu, n, x_0, w)-$bande est également une $(\nu, n, x_0, w, 1)-$multi-bande. La FIGURE 4.12 montre une $(\frac{2}{3}, 5, 0, 1, 3)-$multi-bande, qui correspond donc à trois copies de la bande donnée en FIG. 4.10(b).

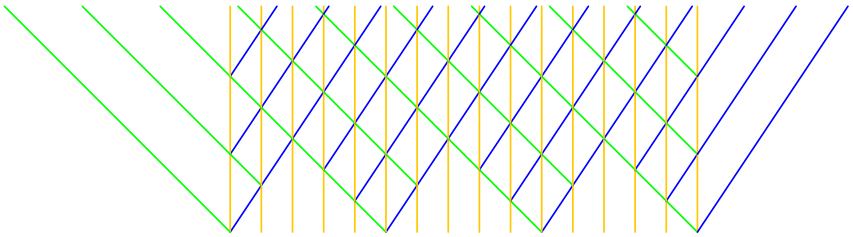


FIGURE 4.12 – Une $(\frac{2}{3}, 5, 0, 1, 3)-$multi-bande : elle est constituée de trois $(\frac{2}{3}, 5, 0, 1)-\mathcal{B}andes$, c'est-à-dire 3 bandes, toutes de largeur 1, ayant chacune 5 sous-divisions, et dont la première bande commence en position 0.

On déduit la périodicité des multi-bandes de celle des bandes :

LEMME 21
Toute $(\frac{p}{q}, p+q, x_0, w, k)$ *– multi-bande devient périodique de période* $T = \frac{w}{p}$*, dans l'intervalle* $[x_0; x_0 + k \cdot w]$ *et après le temps* $t_T = \frac{q}{p+q} \cdot w$*.*

Démonstration. Pour tout $j \in [\![0; k-1]\!]$, la configuration

$$\left\{ \mathsf{S}@x_0 + \left(\frac{i}{n} + j\right) \cdot w, \overleftarrow{\mathsf{L}}@x_0 + j \cdot w, \overrightarrow{\mathsf{R}}@x_0 + j \cdot w \;\middle|\; 0 \leq i \leq n \right\}$$

génère une $(\frac{p}{q}, p+q, x_0 + jw, w)$ – bande (par définition d'une bande). La configuration initiale générant la $(\frac{p}{q}, p+q, x_0, w, k)-$multi-bande est une jonction de telles configurations (jonction suivant les signaux S en position de la forme $x = x_0 + j \cdot w$). Comme ces configurations correspondent toutes à des bandes de mêmes paramètres, tous les aller-retours dans toutes les sous-divisions requièrent la même durée. D'autre part, d'après la preuve du LEM. 20, toutes les collisions entre signaux $\overleftarrow{\mathsf{L}}$ et $\overrightarrow{\mathsf{R}}$ coïncident avec les signaux stationnaires S, y compris ceux faisant la jonction entre deux bandes simples. Comme toutes les bandes sont périodiques de même période à partir du même temps t_C, la multi-bande est elle aussi périodique. Sa période est la période commune à toutes les bandes et est celle donnée par le LEM. 20 : la $(\frac{p}{q}, p+q, x_0, w, k)$ – multi-bande est périodique de période $T = \frac{w}{p}$ à partir du temps $t_T = \frac{q}{p+q} \cdot w$. □

Ce lemme a pour conséquence essentielle le corollaire suivant :

COROLLAIRE 22
Aucune accumulation ne peut apparaître dans une multi-bande.

Démonstration. Par le LEM. 19, il suffit juste de montrer qu'il ne peut y avoir d'accumulation dans l'intervalle $[x_0; x_0 + k \cdot w]$. D'autre part, d'après le LEM. 21, toute multi-bande devient

périodique de période $T = \frac{w}{p}$, à partir du temps t_T et il n'y a qu'un nombre fini de collisions avant le temps t_T. Pour tout $t > t_T$, il n'y a qu'un nombre fini de collisions durant l'intervalle $[t; t+T]$ (durant une période, chaque bande ne contient qu'un aller-retour des signaux $\overleftarrow{\mathsf{L}}$ et $\overrightarrow{\mathsf{R}}$). Comme l'existence d'un intervalle de temps fini contenant une infinité de collisions est une condition nécessaire pour l'existence d'une accumulation, on en déduit qu'il ne peut y avoir d'accumulation dans une multi-bande. □

Tout diagramme de machine rationnelle à 3 vitesses est contenu dans une multi-bande. Nous montrons ici que pour toute configuration initiale finie c_0 de $\mathfrak{M}_3^\nu$, il existe une configuration c_0' qui étend c_0 de telle façon que c_0' génère une multi-bande incluant entièrement le diagramme généré à partir c_0.

Théorème 23
Soit $\nu \in \mathbb{Q}$. Pour tout diagramme espace-temps $\mathcal{D}$ généré par $\mathfrak{M}_3^\nu$ à partir d'une configuration initiale finie et n'ayant que des rapports rationnels entres ses positions, il existe $n, k \in \mathbb{N}$ et $x_0, w \in \mathbb{R}$ tel que $\mathcal{D}$ est inclus dans la (ν, n, x_0, w, k) − multi-bande.

Démonstration. Nous procédons par induction sur le nombre de signaux présents dans la configuration initiale. Nous prouvons d'abord que tout diagramme n'ayant au maximum que deux signaux en configuration initiale est inclus dans une bande, et *a fortiori* dans une multi-bande. Nous prouvons ensuite l'étape d'induction : pour tout signal ajouté sur la droite d'une configuration initiale d'une multi-bande, il est possible de construire une nouvelle multi-bande qui inclura entièrement le diagramme obtenu en ajoutant le nouveau signal à la multi-bande initiale.

Cas initiaux. De manière évidente, aucune configuration initiale ayant 0, 1 ou 2 signaux ne peut donner lieu à une accumulation, et ce, sans tenir compte des vitesses. Avec 0 ou 1 signal, le diagramme ne contiendra aucune collision. Pour 2 signaux initiaux, comme des signaux résultant d'une même collision ne peuvent entrer en collision plus tard, un diagramme initié avec deux signaux ne pourra contenir au maximum qu'une seule collision : celle (éventuelle) entre ces deux signaux.

Cependant, comme nous voulons montrer qu'à partir de n'importe quelle configuration initiale, le diagramme généré est inclus dans une multi-bande, nous devons fournir une multi-bande pour le cas de deux signaux. Pour deux signaux (quelconques) aux positions initiales x_0 et $x_1 > x_0$, une multi-bande suffisante est simplement donnée par la $(\nu, n, x_0, x_1 - x_0)$ − bande, avec $n = 1/\operatorname{pgcd}\left(\frac{\nu}{\nu+1}, 1\right)$, c'est-à-dire de manière équivalente par la $(\nu, n, x_0, x_1 - x_0, 1)$ − multi-bande. La Figure 4.13(a) donne un exemple de configuration initiale à deux signaux $c_0 = \left\{\mathsf{S}@x_0, \overleftarrow{\mathsf{L}}@x_1\right\}$, et la Fig. 4.13(b) donne la multi-bande correspondante, de largeur $w = x_0 - x_1$.

(a) Diagramme avec deux signaux initiaux. (b) La multi-bande correspondante.

Figure 4.13 – Un des cas initiaux.

Le cas de deux signaux (de vitesses distinctes) situés à la même position initiale x_0 et le cas d'un seul signal peuvent être traités en utilisant une bande dégénérée de largeur 0, *i.e.* la bande correspondant à la configuration $\left\{ \overleftarrow{\mathsf{L}}@x_0,\ \mathsf{S}@x_0,\ \overrightarrow{\mathsf{R}}@x_0 \right\}$.

Étape d'induction. Considérons maintenant une $(\nu, n, x_0, w, k)-$ multi-bande notée $\mathcal{B}$. Pour tout signal μ ajouté en position $y > x_0 + kw$ à la configuration c_0 correspondante à $\mathcal{B}$, nous pouvons expliciter les paramètres $\nu, n', x_0' w'$ et k' de la nouvelle multi-bande $\mathcal{B}'$ qui inclura le diagramme obtenu à partir de la configuration $c_0 \cup \{\mu@y\}$. Nous avons supposé ici, sans perte de généralités, que $y > x_0 + kw$: le signal μ est ajouté tout à droite de la configuration initiale. En fait, seuls les paramètres w' et k' changent, car les paramètres ν (la troisième vitesse de la machine), n (qui dépend uniquement de ν) et x_0 (la position du signal le plus à gauche de la configuration initiale) restent identiques. La largeur totale de la multi-bande $\mathcal{B}'$ que nous allons fournir, est donnée par la distance entre les signaux extrémaux c'est-à-dire que $y - x_0$ (la largeur étant également donnée par $k' \cdot w'$). Comme la dernière position d'une multi-bande contient toujours les trois signaux $\overleftarrow{\mathsf{L}}$, S et $\overrightarrow{\mathsf{R}}$, il n'y a pas besoin de distinguer plusieurs sous-cas correspondant au signal ajouté et nous pouvons donner des paramètres qui sont adaptés pour les trois cas à la fois.

Comme la position y du nouveau signal vérifie $y > x_0 + kw$, il existe $d \in \mathbb{R}$ tel que $y = x_0 + kw + d$. Par hypothèse, les positions initiales sont toutes deux-à-deux de rapport rationnel, et donc les valeurs w et d sont commensurables. Posons $w' = \text{pgcd}(w, d)$ et $k' = \frac{kw+d}{w'}$. Comme w et d sont de rapport rationnel, leur pgcd est bien défini et est rationnel. Par ailleurs, $k \in \mathbb{N}$, car comme w' divise w et d, w' divise également $kw + d$. Montrons maintenant que la $(\nu, n, x_0, w', k')-$multi-bande $\mathcal{B}'$ contient la multi-bande initiale $\mathcal{B}$. Nous allons pour cela montrer que chaque $\sigma@x$ dans la configuration initiale c_0 de la multi-bande $\mathcal{B}$ se retrouve également dans la configuration initiale c_0' de la multi-bande $\mathcal{B}'$.

La multi-bande $\mathcal{B}$ est générée à partir de la configuration :

$$c_0 = \left\{ [\mathsf{S}, \overleftarrow{\mathsf{L}}, \overrightarrow{\mathsf{R}}]@x_0 + l \cdot w,\ \mathsf{S}@x_0 + \left(\tfrac{i}{n} + j\right) \cdot w \ \middle|\ 0 \leq l \leq k,\ 0 < i < n \text{ et } 0 \leq j < k \right\}$$

et la $(\nu, n, x_0, w', k')-$ multi-bande $\mathcal{B}'$ à partir de :

$$c_0' = \left\{ [\mathsf{S}, \overleftarrow{\mathsf{L}}, \overrightarrow{\mathsf{R}}]@x_0 + l \cdot w',\ \mathsf{S}@x_0 + \left(\tfrac{i}{n} + j\right) \cdot w' \ \middle|\ 0 \leq l \leq k',\ 0 < i < n \text{ et } 0 \leq j < k' \right\}.$$

Considérons la position x d'un signal $\overleftarrow{\mathsf{L}}$ (ou $\overrightarrow{\mathsf{R}}$) dans c_0 : x s'écrit nécessairement de la forme $x = x_0 + l \cdot w$ avec $0 \leq l \leq k$. On a alors :

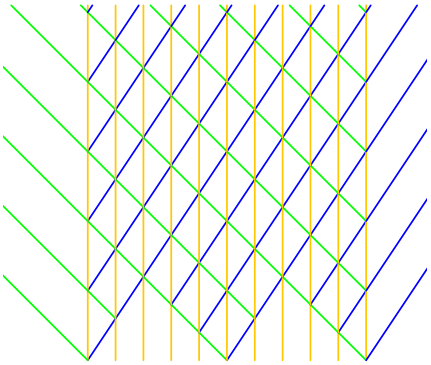
$$\begin{aligned} x &= x_0 + l \cdot w \\ &= x_0 + l \cdot \alpha \cdot w' && \text{où } \alpha \in \mathbb{N} \text{ est défini par } \alpha = \frac{w}{w'} \ (\alpha \in \mathbb{N} \text{ car } w' = \text{pgcd}(w, d)), \\ &= x_0 + l' \cdot w' && \text{en posant } l' = l \cdot \alpha \ (l' \in \mathbb{N} \text{ car } \alpha \in \mathbb{N} \text{ et } l \in \mathbb{N}), \end{aligned}$$

et à partir de $0 \leq l \leq k$, on obtient successivement :

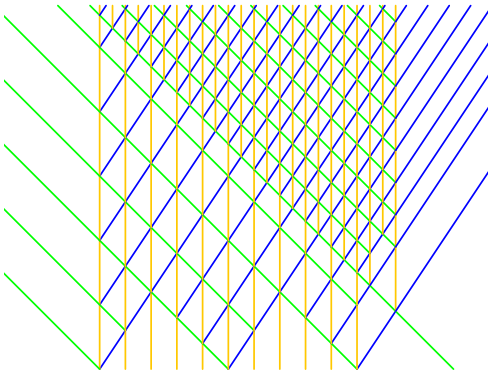
$$\begin{array}{rcccll} 0 & \leq & l \cdot \alpha & \leq & k \cdot \alpha & \\ 0 & \leq & l' & \leq & k \cdot \dfrac{w}{w'} & \\ 0 & \leq & l' & \leq & k \cdot \dfrac{w}{w'} + \dfrac{d}{w'} & \left(\text{car } \dfrac{d}{w'} > 0\right) \\ 0 & \leq & l' & \leq & \dfrac{k \cdot w + d}{w'} & \\ 0 & \leq & l' & \leq & k' & \text{par définition de } k'. \end{array}$$

Donc la position $x = x_0 + l \cdot w$ $(0 \leq l \leq k)$ s'écrit également sous la forme $x = x_0 + l' \cdot w'$ où $0 \leq l' \leq k$: tout signal $\overrightarrow{\mathsf{R}}$ ou $\overleftarrow{\mathsf{L}}$ présent dans c_0 est donc également présent dans c_0'.

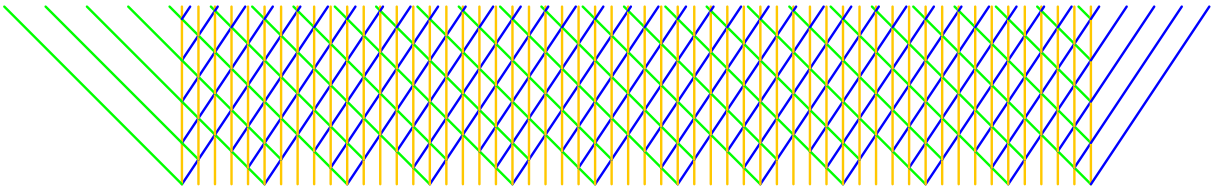
La FIGURE 4.14 détaille cette étape d'induction dans le cas d'un signal $\overleftarrow{\mathsf{L}}$ ajouté à une multi-bande.



(a) La multi-bande initiale avec $w = 1$ et $k = 2$.



(b) Ajout d'un signal L supplémentaire en position $y = \frac{275}{100}$.



(c) La nouvelle multi-bande de paramètres $w' = \frac{1}{4}$ et $k' = 11$.

FIGURE 4.14 – Exemple d'ajout d'un signal L en position $y = \frac{275}{100}$ à une $(\frac{2}{3}, 5, 0, 1, 2)$ – multi-bande. Les nouveaux paramètres obtenus sont $w' = \text{pgcd}(w, y) = \text{pgcd}\left(1, \frac{275}{100}\right) = \frac{1}{4}$ et $k' = \frac{y}{w'} = \frac{275/100}{1/4} = 11$, et la nouvelle multi-bande est donc une $(\frac{2}{3}, 5, 0, \frac{1}{4}, 11)$ – multi-bande.

On procède de la même façon pour montrer que tout signal S présent dans c_0 l'est aussi dans c'_0. On remarque d'abord que les positions de tous les signaux S d'une multi-bande peuvent s'écrire sous la forme $x_0 + r \cdot \frac{w}{n}$ avec $0 \leq r \leq n \cdot k$ (il suffit de regrouper les deux types de positions des signaux S décrites par les indices i et j en un seul type de positions indexées par r). La position x d'un signal S de $\mathcal{B}$ s'écrit donc $x = x_0 + r \cdot \frac{w}{n}$ où $0 \leq r \leq n \cdot k$. En posant $r' = r \cdot \frac{w}{w'}$, on peut alors écrire $x = x_0 + r \cdot \frac{w}{w'} \times \frac{w'}{n} = x_0 + r' \cdot \frac{w'}{n}$. On a $r' \in \mathbb{N}$ car $r \in \mathbb{N}$ et $\frac{w}{w'} \in \mathbb{N}$ (car $w' = \text{pgcd}(w, d)$). De la même façon que dans le cas précédent, on obtient à partir de l'inégalité $0 \leq r \leq n \cdot k$, en la multipliant par $\frac{w}{w'}$ et en la bornant en ajoutant $n \cdot \frac{d}{w'}$ au membre de droite, que $r' \in [\![0; n \cdot k']\!]$. La position x s'écrit donc $x = x_0 + r' \cdot \frac{w'}{n}$ avec $0 \leq r' \leq n \cdot k'$, et correspond donc bien à la position d'un signal S dans la configuration c'_0.

Tous les signaux initiaux de multi-bande $\mathcal{B}$ sont donc également initialement présents dans $\mathcal{B}'$. Il ne reste désormais que le cas du signal μ ajouté en position $y = x_0 + kw + d$. Nous obtenons directement que la position y est une des positions de c'_0 car :

$$y = x_0 + kw + d = x_0 + \frac{kw + d}{w'} \cdot w' = x_0 + k'w' \ .$$

Elle correspond en fait à la dernière position de la configuration c'_0, qui est la position des trois derniers signaux $\overleftarrow{\mathsf{L}}$, S et $\overrightarrow{\mathsf{R}}$.

Nous avons donc montré que c_0 est incluse dans c'_0, et il s'ensuit par la définition des diagrammes supports que le diagramme engendré à partir de c_0 est inclus dans le diagramme

engendré par c'_0. De même, le diagramme généré à partir de $c_0 \cup \{\mu@y\}$ est entièrement inclus dans la multi-bande $\mathcal{B}'$.

Soit un $\mathcal{D}$ un diagramme de la machine $\mathfrak{M}_3^\nu$ généré à partir d'une configuration initiale finie $c_0 = \{\sigma_i@x_i \mid x_i \in \mathbb{R}\}_{0\leq i\leq m}$ avec $x_1 \leq x_2 \leq \ldots \leq x_m$ et telle que pour tout $i, j, l \in [\![0; m]\!]$, la valeur $\frac{x_i - x_j}{x_j - x_l}$ $(x_j \neq x_l)$ est rationnelle. Nous pouvons désormais donner des paramètres $n, k \in \mathbb{N}$ et $x, w \in \mathbb{R}$ tels que le diagramme $\mathcal{D}$ soit inclus dans une (ν, n, x_0, w, k) − multi-bande. Après avoir calculé la bande correspondant à la configuration $\{\sigma_0@x_0, \sigma_1@x_1\}$, nous calculons, grâce à l'étape d'induction, les multi-bandes obtenues en ajoutant successivement les éléments $\sigma_i@x_i$ dans l'ordre. Comme tous les rapports sont rationnels, les pgcd utilisés ci-dessus existent et les paramètres sont tous bien définis. La configuration initiale c_0 étant finie, on obtient après m étapes la (ν, n, x_0, w, k) − multi-bande, dont les paramètres sont ceux obtenus lors du calcul de la dernière multi-bande. On obtient ainsi une multi-bande $\mathcal{B}$ telle que $\mathcal{D} \subseteq \mathcal{B}$. De plus, nous pouvons donné explicitement ces paramètres. Les paramètres ν, n et x_0 sont les mêmes pour toutes les multi-bandes successives (ν est une vitesse fixée par la machine, $n = 1/\operatorname{pgcd}\left(\frac{\nu}{\nu+1}, 1\right)$, et x_0 est donné par la configuration c_0). La largeur w et le nombre k de bandes de la multi-bande globale sont donnés par $w = \operatorname{pgcd}(x_2 - x_1, x_3 - x_2, \ldots, x_m - x_{m-1})$ et $k = \frac{x_m - x_1}{w}$.

□

Nous en déduisons finalement :

Corollaire 24
Aucune machine à signaux à 3 vitesses à rapports rationnels entre vitesses et exécutée à partir d'une configuration à positions de rapports rationnels ne produit d'accumulation.

Démonstration. Par normalisation des vitesses, toute machine $\mathfrak{M}$ à 3 vitesses dont tous les rapports sont rationnels peut être réduite à une machine dont toutes les vitesses sont rationnelles (-1, 0 et ν), et dont la machine support est $\mathfrak{M}_3^\nu$ avec ν rationnel. Par le Cor. 5, si $\mathfrak{M}_3^\nu$ ne génère pas d'accumulation, alors $\mathfrak{M}$ n'en générera pas non plus. D'après le Th. 23, tout diagramme de $\mathfrak{M}_3^\nu$ est inclus dans une multi-bande, et comme aucune multi-bande ne peut contenir d'accumulation par le Cor. 22, il s'ensuit qu'aucune accumulation ne peut être générée par $\mathfrak{M}_3^\nu$, et donc *a fortiori*, par $\mathfrak{M}$. □

4.4 Applications

Nous donnons dans cette courte section quelques conséquences immédiates des résultats obtenus en Sec. 4.2 et Sec. 4.3. que nous formulons en termes de décidabilité de certains problèmes de décision liés aux machines à signaux. Le critère de minimalité du nombre de vitesses nécessaires pour générer une accumulation s'applique ici dans la décidabilité de l'apparition d'un signal pour des machines à signaux dont le nombre de vitesses est donné en paramètre du problème.

Une version générale du problème considéré ici a été démontrée indécidable dans [Durand-Lose 2003]. Elle s'énonce de la façon suivante :

Problème **ApparitionSignal**

entrée : une machine rationnelle $\mathfrak{M}$, un méta-signal μ et une configuration c_0 finie.
sortie : est-ce qu'un signal de type μ apparaît après un nombre fini de collisions dans l'évolution de $\mathfrak{M}$ à partir de c_0 ?

La restriction « un nombre fini de collisions » est importante : elle exclut les accumulations. Le cas général étant indécidable, nous nous intéressons ici à une version paramétrée du problème, dans laquelle les machines données en entrée sont restreintes sur le nombre de vitesses différentes :

Problème n-**VitessesApparitionSignal**

entrée : une machine rationnelle $\mathfrak{M}$ à n vitesses, un méta-signal μ et une configuration c_0 finie.

sortie : est-ce qu'un signal de type μ apparaît après un nombre fini de collisions dans l'évolution de $\mathfrak{M}$ à partir de c_0 ?

En suivant les résultats de la section précédente, nous distinguons trois cas : $n = 2$, $n = 3$ et $n \geq 4$ (le cas $n = 1$ étant sans intérêt, puisqu'une machine à 1 vitesse ne peut pas engendrer de collisions, et dans ce cas, tester si un signal donné apparaît revient à vérifier s'il est présent dans la configuration initiale).

Cas $n = 2$

D'après le Lem. 15, le nombre de collisions d'un diagramme d'une machine à 2 vitesses est fini (et est borné de manière quadratique par le nombre de signaux présents dans la configuration initiale). Il s'ensuit que tout diagramme d'une machine à deux vitesses est fini, et par conséquent le problème 2-**VitessesApparitionSignal** est décidable : il suffit de calculer l'historique des collisions (ce qui prendra un temps fini) et de vérifier si le signal μ apparaît en sortie d'une de ces collisions.

Cas $n = 3$

L'énoncé du problème n-**VitessesApparitionSignal** impose pour les machines données en entrée d'être rationnelles. Le Théorème 23 de la Sous-sec. 4.3.2 implique que tout diagramme d'une machine rationnelle à 3 vitesses est inclus dans un diagramme d'une machine support périodique à partir d'un temps donné (et sans tenir compte d'éventuels signaux se propageant à l'infini vers la droite et la gauche). Contrairement aux machines supports, les machines considérées en instance de n-**VitessesApparitionSignal** peuvent avoir strictement plus de trois méta-signaux, mais leur nombre est toujours fini. Néanmoins, tous les signaux qui apparaissent dans le diagramme sont sur les structures périodiques des multi-bandes. On en déduit donc que tout diagramme d'une machine rationnelle à 3 vitesses exécutée à partir d'une configuration finie est périodique. Il s'ensuit que le problème 3-**VitessesApparitionSignal** est décidable, la méthode étant la même que pour le cas $n = 2$, mais en calculant l'historique des collisions durant une période complète.

Cas $n = 4$

L'indécidabilité du problème 4-**VitessesApparitionSignal** se montre par réduction au problème de l'arrêt d'une machine de Turing. L'existence d'une telle réduction provient de la possibilité de simuler des modèles Turing-universels en utilisant seulement quatre vitesses distinctes. Il est ensuite facile de modifier la simulation de telle sorte que la machine simulée s'arrête si et seulement si un signal μ apparaît.

Nous renvoyons à [Durand-Lose 2011a] pour une simulation d'un *cyclic tag system* avec quatre vitesses (mais plus de méta-signaux). Notons qu'il est également possible de simuler directement une machine de Turing en utilisant seulement quatre vitesses.

Variantes

De nombreuses variantes de ce problème existent : le diagramme d'une machine rationnelle à n vitesses exécutée à partir de c_0 est-il fini ? est-ce qu'un signal μ entre en collision dans l'évolution d'une machine rationnelle à n vitesses à partir de c_0 ? est-ce qu'une accumulation apparaît lors de l'évolution d'une machine rationnelle à n vitesses à partir de c_0 ? *etc.*

Ces problèmes admettent les mêmes valeurs critiques pour le paramètre n : ils sont décidables pour $n \leq 3$ et deviennent indécidables lorsque $n \geq 4$ (cela se montre de la même façon que ci-dessus).

Notons que la décidabilité du problème de l'**ApparitionAccumulation** dans les cas de machines rationnelles à 2 et 3 vitesses a été montrée de manière directe dans la Sous-sec. 4.2, puisqu'aucune accumulation ne peut survenir dans ces cas. L'indécidabilité du cas $n = 4$ provient de la possibilité de simuler une machine de Turing avec 4 vitesses de telle sorte que l'exécution de la machine est infinie si et seulement s'il se produit une accumulation.

Cas de machines non-rationnelles

L'hypothèse de rationalité des machines dans les énoncés des problèmes considérés ci-dessus est indispensable. En effet, les modèles classiques de calcul comme les machines de Turing, ne permettent pas de manipuler de manière exacte des nombres irrationnels, c'est pour cela que les énoncés de problème de décision du type n-**VitessesApparitionSignal**, sont restreints aux machines rationnelles pour ne pas sortir du cadre de la calculabilité classique. Il semble cependant possible d'étudier la décidabilité des problèmes précédents dans le cas de machines non-rationnelles dans des modèles de calcul analogiques tels que l'analyse récursive ou le modèle BSS.

Néanmoins, il est naturel de se poser la question de savoir s'il est possible de simuler des machines de Turing (ou tout autre modèle équivalent) avec des machines non-rationnelles à n vitesses. Il est clair qu'avec des machines à 1 ou 2 vitesses, une telle simulation est impossible (indépendamment de la rationalité des vitesses ou des positions). Lorsque $n \geq 4$, cela est possible, car déjà possible avec des machines rationnelles à 4 vitesses. Le cas limite de 3 vitesses a été résolu récemment dans [Durand-Lose 2013]. À partir des résultats de ce chapitre, il y est démontré qu'il est possible de simuler des machines de Turing par des machines à signaux irrationnelles à 3 vitesses, et que par conséquent, *les machines à signaux irrationnelles à* 3*-vitesses sont Turing-universelles*. Notons que comme dans le cas de machines rationnelles, le fait qu'il est impossible d'être Turing-universel avec uniquement 3 vitesses se déduit de manière directe du Th. 23 et de la périodicité des multi-bandes : en effet, de telles machines rationnelles ne peuvent pas simuler de machine de Turing universelle puisqu'elles ne peuvent simuler que des calculs périodiques à partir d'une certaine étape.

Chapitre 5

Fractales et signaux

Après avoir étudié dans le chapitre précédent les phénomènes d'accumulations de manière générale, nous allons maintenant nous en intéresser à un type particulier : les *fractales*. Les fractales sont en effet des accumulations avec une structure régulière, obtenues en itérant des algorithmes géométriques simples. Nous avons vu dans le Chap. 4 qu'il est possible de générer des accumulations avec peu de vitesses (quatre vitesses distinctes, trois dans le cas de machines non rationnelles). Nous allons montrer ici qu'avec un faible nombre de vitesses supplémentaires, il est possible de générer simplement certaines structures fractales avec des signaux. Cette étude se place dans la continuité du chapitre précédent, et fournit déjà un premier exemple du parallèlisme massif du modèle des machines à signaux.

Nous commencerons par quelques rappels généraux sur les fractales — méthodes de constructions, fractales classiques, notion de dimension fractale — et nous donnerons également un état de l'art très succinct sur l'étude de fractale dans des modèles de calcul non-classiques.

Nous présenterons ensuite quelques machines à signaux permettant de construire des fractales simples. Nous nous servirons de ces exemples pour démontrer que pour n'importe quelle dimension d strictement entre 0 et 1, il existe une machine à signaux à 1 dimension permettant de générer une fractale de dimension d.

Enfin, nous montrerons qu'il est également possible d'approximer ces fractales de deux façons essentiellement différentes : de manière *spécifique* et de manière *générique*. Ces deux notions d'approximations, appliquées ici à l'exemple de l'*arbre binaire fractale*, fournissent un moyen de découper l'espace de manière automatique et parallèle, et seront utilisées dans les chapitres suivants pour implémenter géométriquement des algorithmes efficaces de résolution de problèmes de satisfaisabilité.

5.1 Le monde des fractales

Cette section regroupe une très brève présentation des fractales en général, et de quelques fractales classiques, ainsi qu'un état de l'art succinct sur l'étude et l'utilisation des fractales dans certains modèles de calcul (surtout dans des modèles non-conventionnels).

5.1.1 Définitions des fractales

Les fractales sont aujourd'hui étudiées et utilisées dans de nombreux domaines : physique, économie, géologie, biologie... Mais elles ont d'abord été étudiées d'un point de vue strictement mathématique, afin de comprendre certaines notions telles la continuité ou l'infini. Informellement, les fractales correspondent à des « objets qui présentent les mêmes (ir)régularités à toutes les échelles ».

Notions de dimension fractale et ensembles fractales. Les ensembles fractales ont été introduits de manière formelle par Benoît Mandelbrot [Mandelbrot 1967, 1982]. À la base de la formalisation d'ensemble fractale se trouve la notion de *dimension fractale*. Celle-ci permet de formuler de manière mathématique le concept de « répétition (ir)régulière de motifs à différentes échelles ». Il existe en fait plusieurs définitions de dimension fractale : dimension de Hausdorff, dimension de Minkowski, *etc*. De même, il existe plusieurs définitions non équivalentes des fractales, dont certaines sont basées sur ces différentes notions de dimensions fractales. La définition de fractale qui est couramment admise est celle formulée par Benoît Mandelbrot et est basée sur la notion de dimension de Hausdorff (nous renvoyons à [Edgar 2008] pour les définitions des différents concepts de dimensions et de fractales, qui sortent du cadre de ce manuscrit).

Nous allons voir un autre cadre, légèrement moins général, permettant de formaliser la plupart des fractales : les *systèmes de fonctions itérées*. C'est ce cadre qui sera utilisé dans ce manuscrit pour évoquer les fractales.

Systèmes de Fonctions Itérées et auto-similarité. Une manière simple de définir et construire une large gamme de fractales est donnée par les *systèmes de fonctions itérées* ou *iterated function systems (IFS)*. De tels systèmes ont été introduits dans [Barnsley et Demko 1985]. Ils sont définis à partir d'un nombre fini d'applications contractantes $(h_i)_{1\leq i\leq n}$ définies de E dans E et de rapports respectifs $(r_i)_{1\leq i\leq n}$ *i.e.* $h_i : E \to E$ de rapport $r_i \in]0;1[$. Le résultat qui fait le lien entre ces systèmes et les fractales est l'existence d'un unique ensemble compact K tel que $K = \bigcup_{1\leq i\leq n} h_i(K)$ (intuitivement, un ensemble est *compact* si c'est un espace fermé et borné). On sait de plus qu'il existe un unique nombre $s \in \mathbb{R}$ qui vérifie $\sum_{i=1}^{n} r_i^s = 1$. Ce nombre s définit une autre notion de dimension fractale appelée *dimension d'auto-similarité de* K. On parlera alors de *fractale* si $s \notin \mathbb{N}$. Les fractales dans ce sens le sont aussi au sens de Mandelbrot. Cependant il existe des fractales au sens de Mandelbrot qui ne le sont pas dans le sens des IFS.

Dans ce manuscrit, nous considérons que les fractales sont définies comme étant des compacts attracteurs d'IFS et tels que leur dimension d'auto-similarité n'est pas entière. Cela permet de capturer de manière simple toutes les fractales définies à partir des construction géométriques simples.

Méthodes pour générer des fractales. Les fractales sont en général construites à partir d'une définition récursive qui itère un processus simple en partant d'un ensemble simple. Il existe différents moyens pour réaliser cela.

Tout d'abord, une fractale peut être définie par *évidement* : en partant d'un ensemble compact on retire successivement des sous-parties de cet ensemble selon le même schéma à chaque étape. Par exemple, l'ensemble de Cantor, le triangle de Sierpiński ou encore l'éponge de Menger sont construits de cette manière.

On peut également procéder par *remplacement de motif* : il s'agit de remplacer un compact par un motif particulier qui permettra ensuite de l'appliquer aux parties de l'ensemble obtenu. Par exemple, la courbe et le flocon de von Koch, les courbes de Peano, Hilbert, Lebesgue, et de manière générale, toutes les fractales dérivant de L-systèmes, sont obtenus par remplacement de motifs.

Enfin, les fractales peuvent être définies à travers le cadre des systèmes dynamiques : elles y sont alors définies comme étant l'attracteur d'un système dynamique, ou bien à partir d'une propriété du système dynamique. Les IFS ou encore les ensembles de Mandelbrot et de Julia rentrent dans cette méthode de génération de fractales.

Nous renvoyons le lecteur à l'ouvrage [Edgar 2008] pour plus de détails sur les constructions et les propriétés des ensembles fractales évoqués ici.

5.1.2 Quelques fractales classiques

Nous rappelons brièvement ici quelques fractales célèbres et classiques, dont nous implémenterons des constructions par signaux par la suite.

Ensembles de Cantor. L'un des premiers ensembles fractales qui a été formellement étudié est le *triadique de Cantor* $\mathcal{C}$, défini par Georg Cantor [Cantor 1884]. Le triadique est l'ensemble compact attracteur de l'IFS donné par : $\begin{cases} h_1(x) & = \frac{1}{3}x \\ h_2(x) & = \frac{1}{3}x + \frac{2}{3} \end{cases}$, où $h_1, h_2 : [0,1] \to [0,1]$. L'ensemble $\mathcal{C}$ vérifie $\mathcal{C} = h_1[\mathcal{C}] \bigcup h_2[\mathcal{C}]$. Sa dimension fractale (d'auto-similarité) est donnée par la solution de $\sum_{i=1}^{n} \left(\frac{1}{3}\right)^s = 1$ et vaut $dim_{frac}(\mathcal{C}) = \frac{ln(2)}{ln(3)} \approx 0,6309$.

De manière géométrique, l'IFS considéré ci-dessus correspond à partir du segment $[0,1]$, retirer le tiers central, puis recommencer successivement sur les segments restant. La FIGURE 5.1 illustre les premières itérations de la construction du triadique de Cantor.

FIGURE 5.1 – Les premières itérations de la construction du triadique de Cantor.

Courbe de von Koch. La courbe de Helge von Koch [von Koch 1904], construite en tant qu'exemple de courbe continue nulle part dérivable, fournit un exemple de fractale définie par remplacement de motifs. Cet ensemble fractale est défini à partir d'un segment et de l'étape d'itération consistant à remplacer le tiers central du segment par deux segments en suivant un triangle équilatéral. Cette étape est alors réitérée sur chaque portion de la ligne brisée ainsi obtenue, comme cela est illustrée par la FIG. 5.2.

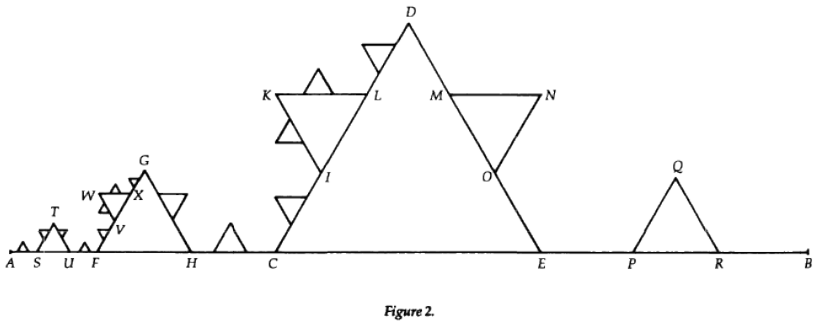


FIGURE 5.2 – Courbe originale de von Koch [Edgar 1993, Fig. 2, p. 29].

Triangle de Sierpiński. Le triangle (ou fanion) de Sierpiński a été introduit Waclaw Sierpiński dans [Sierpiński 1915]. Une méthode géométrique classique pour construire le triangle de Sierpiński consiste à partir d'un triangle équilatéral, à le partitionner en quatre triangles équilatéraux et à lui retirer le triangle équilatéral central, laissant ainsi trois triangles équilatéraux sur lesquels est de nouveau appliquée cette étape. La FIGURE 5.3 montre le résultat obtenu après 7 itérations. Plus formellement, le triangle de Sierpinski se définit comme l'ensemble compact

$\mathcal{S} \subset \mathbb{R}^2$ attracteur de l'IFS suivant : $\begin{cases} h_1(x,y) &= \frac{1}{2}(x,y) \\ h_2(x,y) &= \frac{1}{2}(x,y) + (\frac{1}{2},0) \\ h_3(x,y) &= \frac{1}{2}(x,y) + (0,\frac{1}{4}) \end{cases}$.

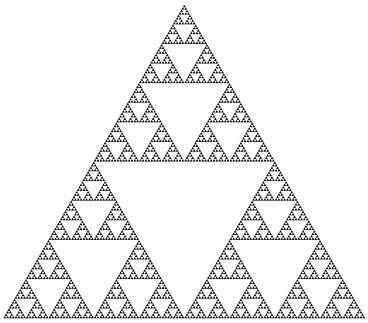


FIGURE 5.3 – Les premières itérations de la construction du triangle de Sierpiński.

Autres fractales. Il existe de nombreuses autres fractales classiques : l'ensemble de Mandelbrot, la courbe de Peano, la courbe de Hilbert, l'éponge de Menger, les dragons, les H-fractales, les arbres... Ces derniers sont particulièrement intéressants parce qu'ils sont très simples à générer et facile à comprendre. Ils sont basés sur la répétition simple d'un motif lui aussi très simple : on ajoute deux segments de même longueur à la même extrémité d'un segment initial. Cette opération est ensuite répétée sur les deux nouveaux segments et en gardant les mêmes paramètres (rapport de longueur entre les trois segments, et l'angle formé). Cette méthode génère toute une famille de fractales dont les structures varient suivant les paramètres choisis.

Nous renvoyons une fois de plus à l'ouvrage [Edgar 2008] pour l'étude formelle de nombreux exemples de fractales classiques.

5.1.3 Fractales et modèles de calcul non-conventionnels

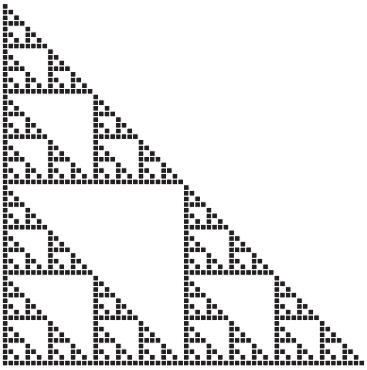
Les fractales, beaucoup étudiées formellement en mathématiques et dans des cadres classiques, l'ont également été sur des modèles de calcul non-conventionnels : nous en donnons ici quelques brefs exemples.

Modèles géométriques. Par leur nature géométrique, les fractales ont bien évidemment été étudiées de manière naturelle sur des modèles géométriques de calcul, tant pour utiliser certaines de leur propriétés que simplement pour étudier la possibilité de les générer et de les construire.

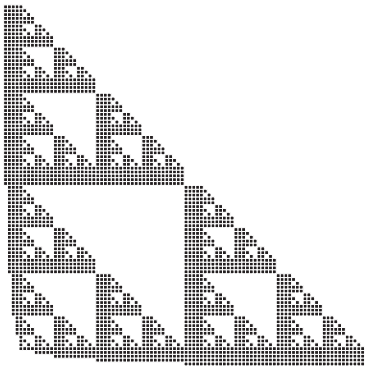
Un des exemples les plus simples est l'apparition naturelle de fractales dans les automates cellulaires élémentaires : par exemple, l'ACE de règle 90 produit une version discrète du triangle de Sierpinski, comme celle de la FIG. 5.4(a). De telles structures présentes naturellement dans certains ACE ont ensuite été étudiées pour de plus larges classes d'AC : citons par exemple l'études des comportements d'AC linéaires et leurs liens avec les fractales [Čulik II et Dube 1989], ainsi que l'étude de leur propriétés d'auto-similarité [Haeseler *et al.* 1995] ou la dimension fractale des diagrammes espace-temps engendrés [Willson 1984]. L'utilisation de diverses structures fractales a également fournie des outils pour la synchronisation d'AC : comme déjà mentionné au CHAP. 1, le problème du fusilier se résout à partir de constructions géométriques continues et toutes les solutions efficaces ont été obtenus à partir d'un motif fractale (voir [Yunès 2007] pour des solutions au problème du fusilier à partir d'arbres infinis et [Yunès 2008] une solution d'une cas du problème du fusilier à partir du triangle de Sierpiński discret).

Citons également quelques études de fractales en auto-assemblage : il a été montré qu'il est possible de générer certaines structures fractales avec des jeu de tuiles auto-assemblantes. C'est

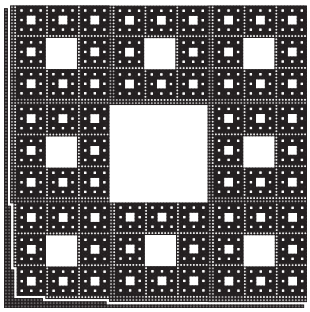
le cas par exemple du triangle de Sierpiński « fibré » [Lathrop *et al.* 2009], ainsi qu'une variante du tapis de Sierpiński [Patitz et Summers 2010], comme sont respectivement illustrés par les Fig. 5.4(b) et 5.4(c). Les pavages ont également donnés lieu à de nombreuses constructions autour des fractales, ainsi qu'à l'étude de certaines propriétés telles que l'auto-similarité dans les pavages [Durand 1996].



(a) Triangle de Sierpiński discret [Lathrop *et al.* 2009, Fig. 1].



(b) Version auto-assemblante [Lathrop *et al.* 2009, Fig. 2].



(c) Un tapis de Sierpiński auto-assemblant [Patitz et Summers 2010, Fig. 14].

Figure 5.4 – Fractales discrètes et systèmes auto-assemblants.

Autres modèles. D'autres modèles de calcul, non géométriques, ont également fourni un cadre pour l'étude de certains problèmes sur les fractales. C'est le cas par exemple du statut du problème de l'appartenance à l'ensemble de Mandelbrot c'est-à-dire le problème consistant à savoir si un point du plan complexe est ou non dans l'ensemble de Mandelbrot. Ce problème n'est pas formalisable dans les modèles de calcul classiques car il est défini sur des valeurs réelles. Il est cependant formalisable dans le modèle de BSS, qui permet de manipuler de manière exacte des réels. Il a alors été montré dans [Blum *et al.* 1998] que ce problème est indécidable (mais récursivement énumérable) au sens de BSS.

L'analyse récursive, en tant que modèle de calcul sur les réels, fournit également un formalisme permettant d'étudier la calculabilité des structures fractales. Par exemple, la calculabilité et la complexité de courbes fermées fractales et de leur intérieur ont ainsi pu être traitées dans le cadre de l'analyse récursive [Ko 2003].

5.2 Construction de fractales par machines à signaux

Nous illustrons dans cette section la capacité des machines à signaux à engendrer naturellement des structures fractales. Nous donnons principalement des exemples de fractales à partir de machines à une dimension spatiale, Il faut cependant distinguer plusieurs façons de définir les fractales ainsi générés : les ensembles fractales peuvent soit être définis en tant qu'ensemble support d'une configuration à un instant donné, soit en tant que support du diagramme espace-temps tout entier. Dans le premier cas, il s'agit donc de générer des fractales sur machines à signaux de manière purement spatiale, tandis que dans le deuxième cas, les fractales devront être perçus à la fois dans l'espace et dans le temps.

5.2.1 Exemples de fractales sur machines à signaux à une dimension spatiale

Nous commençons par donner des machines permettant de construire quelques fractales classiques : un arbre binaire infini, le triadique et une fractale de type von Koch.

L'arbre binaire fractale. En itérant le calcul géométrique du milieu, il est possible de construire facilement un *arbre binaire fractale*. Il suffit pour cela de modifier la machine $\mathfrak{M}_{mid}$ de la SOUS-SEC. 3.1.1 pour recommencer indéfiniment et de manière symétrique le processus de marquage du milieu d'un segment par un signal vertical w. On utilise pour cela la machine définie par les FIG. 5.5(a) et 5.5(b). Exécutée à partir de la configuration initiale $c_0^{tree} = \{\ [\mathsf{w}, \overrightarrow{\mathsf{a}}, \overrightarrow{\overrightarrow{\mathsf{a}}}]@0,\ \mathsf{w}@1\}$, elle engendre le diagramme donné dans la FIG. 5.5(c).

L'arbre ainsi obtenu est une représentation de l'*ensemble de Cantor abstrait*, formellement défini par $\mathcal{C} = \{0,1\}^{\mathbb{N}}$ (un chemin dans l'arbre correspond à un élément $x \in \mathcal{C}$). Notons que cet arbre a largement été utilisé pour résoudre le problème du fusilier dans les AC, comme nous l'avons déjà mentionné en SOUS-SEC. 1.2.2.

Méta-signal	Vitesse
$\overrightarrow{\overrightarrow{\mathsf{a}}}$	3
$\overrightarrow{\mathsf{a}}$	1
w	0
$\overleftarrow{\mathsf{a}}$	−1
$\overleftarrow{\overleftarrow{\mathsf{a}}}$	−3

(a) Méta-signaux.

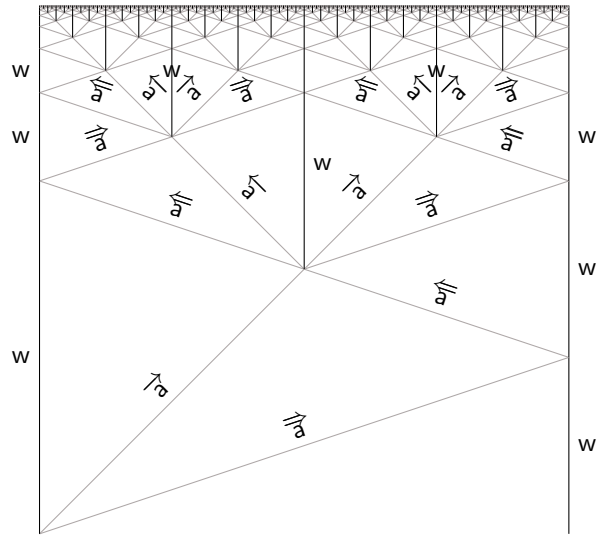
$\{\ \mathsf{w}, \overleftarrow{\overleftarrow{\mathsf{a}}}\ \} \to \{\ \mathsf{w}, \overrightarrow{\overrightarrow{\mathsf{a}}}\ \}$

$\{\ \overrightarrow{\overrightarrow{\mathsf{a}}}, \mathsf{w}\ \} \to \{\ \overleftarrow{\overleftarrow{\mathsf{a}}}, \mathsf{w}\ \}$

$\{\ \overrightarrow{\mathsf{a}}, \overleftarrow{\mathsf{a}}\ \} \to \{\ \overleftarrow{\mathsf{a}}, \overleftarrow{\mathsf{a}}, \mathsf{w}, \overrightarrow{\mathsf{a}}, \overrightarrow{\overrightarrow{\mathsf{a}}}\ \}$

$\{\ \overrightarrow{\overrightarrow{\mathsf{a}}}, \overleftarrow{\mathsf{a}}\ \} \to \{\ \overleftarrow{\overleftarrow{\mathsf{a}}}, \overleftarrow{\mathsf{a}}, \mathsf{w}, \overrightarrow{\mathsf{a}}, \overrightarrow{\overrightarrow{\mathsf{a}}}\ \}$

(b) Règles de collisions.



(c) Diagramme espace-temps.

FIGURE 5.5 – La machine $\mathfrak{M}_{tree}$: construction d'un arbre binaire fractale.

Décrivons brièvement le fonctionnement de la machine. Les deux premières règles de la FIG. 5.5(b) correspondent aux rebonds de $\overrightarrow{\overrightarrow{\mathsf{a}}}$ et $\overleftarrow{\overleftarrow{\mathsf{a}}}$ sur les murs w. Les deux dernières règles définissent l'étape d'induction pour initier le niveau suivant : les signaux initiaux $\overrightarrow{\mathsf{a}}$ et $\overrightarrow{\overrightarrow{\mathsf{a}}}$ sont dupliqués à gauche et à droite et le signal stationnaire w est crée exactement au milieu du niveau précédent définissent par les deux murs w les plus proches. Il est également possible d'obtenir un arbre binaire infini sans générer tous les signaux verticaux : c'est le cas de l'arbre déjà présenté en FIG. 4.1(b) (voir également [Durand-Lose 2003, p. 148, Fig. B.1(b)]).

Bien que le processus de division par des segments se poursuive indéfiniment, la construction reste de hauteur bornée : si la largeur est w alors la hauteur est également w (pour des vitesses 1 et 3). En effet, on sait d'après le calcul du milieu donné en exemple dans la SOUS-SEC. 3.1.1 que chaque étage a une hauteur égale à la moitié de la hauteur de l'étage précédent (car le processus de calcul du milieu à partir d'une largeur x nécessite une durée $x/2$). Donc en partant d'une largeur totale w (*i.e.* la distance initiale entre les deux signaux w est de w), la hauteur de l'arbre complet est donnée par :

$$\lim_{n\to\infty} \sum_{i=1}^{n} \frac{w}{2^i} = w \cdot \sum_{i=1}^{\infty} \frac{1}{2^i} = w \ .$$

Notons qu'il y a au temps $t = w$ une infinité d'accumulations non-isolées : pour tout point (x, t) tel que $x \in [0; w]$ et $t = w$, il existe une infinité de points $(x', t') \in \mathcal{V}_{(x,t)}$ (le voisinage de (x, t)) tel que $c(x', t') = ✺$.

Le triadique de Cantor. En changeant les vitesses de la machine $\mathfrak{M}_{mid}$, on peut facilement construire une machine calculer le tiers d'une segment donné au lieu de calculer son milieu. Pour générer le triadique de Cantor, le principe est ensuite de réitérer ce calcul du tiers, ainsi que le calcul de deux tiers d'une distance, de la même façon que le calcul du milieu était itérer pour générer l'arbre binaire infini. Une version du triadique du Cantor a déjà été fournie dans [Durand-Lose 2003, p. 150, Fig. B.3] : nous en présentons ici une autre, dont nous détaillons la machine $\mathfrak{M}_{Cantor}$.

Les méta-signaux et règles de collisions de la machine $\mathfrak{M}_{Cantor}$ sont respectivement donnés en FIG. 5.6(a) et 5.6(b). La FIGURE 5.6(c) fournit le diagramme produit par la machine $\mathfrak{M}_{Cantor}$ à partir de la configuration initiale $c_0^{triadic} = \left\{ [\mathsf{w}, \overrightarrow{\mathsf{a}}, \overrightarrow{\mathsf{b}}, \overrightarrow{\mathsf{a}}, \overrightarrow{\mathsf{b}}]@0, \mathsf{w}@1 \right\}$.

Méta-signal	Vitesse
$\overrightarrow{\overrightarrow{\mathsf{a}}}$	5
$\overrightarrow{\overrightarrow{\mathsf{b}}}$	4
$\overrightarrow{\mathsf{b}}$	2
$\overrightarrow{\mathsf{a}}$	1
w	0
$\overleftarrow{\mathsf{a}}$	−1
$\overleftarrow{\mathsf{b}}$	−2
$\overleftarrow{\overleftarrow{\mathsf{b}}}$	−4
$\overleftarrow{\overleftarrow{\mathsf{a}}}$	−5

(a) Méta-signaux.

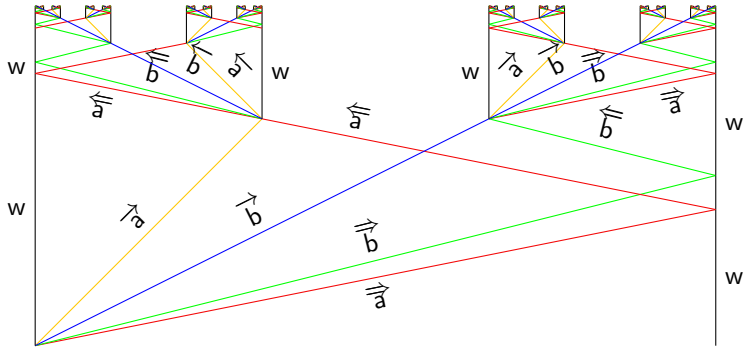
$\{ \mathsf{w}, \overleftarrow{\overleftarrow{\mathsf{a}}} \} \to \{ \mathsf{w}, \overrightarrow{\overrightarrow{\mathsf{a}}} \}$ $\quad$ $\{ \overrightarrow{\overrightarrow{\mathsf{a}}}, \overleftarrow{\mathsf{a}} \} \to \{ \mathsf{w}, \overrightarrow{\mathsf{a}}, \overrightarrow{\mathsf{b}}, \overrightarrow{\overrightarrow{\mathsf{a}}}, \overrightarrow{\overrightarrow{\mathsf{b}}} \}$

$\{ \overrightarrow{\overrightarrow{\mathsf{a}}}, \mathsf{w} \} \to \{ \overleftarrow{\overleftarrow{\mathsf{a}}}, \mathsf{w} \}$ $\quad$ $\{ \overrightarrow{\mathsf{a}}, \overleftarrow{\overleftarrow{\mathsf{a}}} \} \to \{ \overleftarrow{\overleftarrow{\mathsf{b}}}, \overleftarrow{\overleftarrow{\mathsf{a}}}, \overleftarrow{\mathsf{b}}, \overleftarrow{\mathsf{a}}, \mathsf{w} \}$

$\{ \mathsf{w}, \overleftarrow{\overleftarrow{\mathsf{b}}} \} \to \{ \mathsf{w}, \overrightarrow{\overrightarrow{\mathsf{b}}} \}$ $\quad$ $\{ \overrightarrow{\overrightarrow{\mathsf{b}}}, \overleftarrow{\mathsf{b}} \} \to \{ \mathsf{w}, \overrightarrow{\mathsf{a}}, \overrightarrow{\mathsf{b}}, \overrightarrow{\overrightarrow{\mathsf{a}}}, \overrightarrow{\overrightarrow{\mathsf{b}}} \}$

$\{ \overrightarrow{\overrightarrow{\mathsf{b}}}, \mathsf{w} \} \to \{ \overleftarrow{\overleftarrow{\mathsf{b}}}, \mathsf{w} \}$ $\quad$ $\{ \overrightarrow{\mathsf{b}}, \overleftarrow{\overleftarrow{\mathsf{b}}} \} \to \{ \overleftarrow{\overleftarrow{\mathsf{b}}}, \overleftarrow{\overleftarrow{\mathsf{a}}}, \overleftarrow{\mathsf{b}}, \overleftarrow{\mathsf{a}}, \mathsf{w} \}$

(b) Règles de collisions.



(c) Diagramme espace-temps.

FIGURE 5.6 – La machine $\mathfrak{M}_{Cantor}$: construction du triadique de Cantor.

Comme pour l'arbre binaire, la hauteur de la construction reste bornée : pour une largeur initiale de w, la hauteur est $\frac{1}{2} \cdot w$ (pour les vitesses choisies de 1 à 5). En effet, chaque étage a une hauteur égale au tiers de celle de l'étage précédent car marquer le tiers d'une distance x s'effectue en une durée $x/3$. De plus, avec les vitesses choisies ici, le premier et le second tiers sont tous deux marqués de manière synchrone. Partant d'une largeur initiale de w, on obtient ainsi comme hauteur totale :

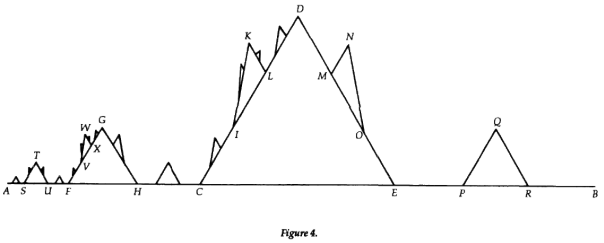
$$\lim_{n\to\infty} \sum_{i=1}^{n} \frac{w}{3^i} = w \cdot \sum_{i=1}^{\infty} \frac{1}{3^i} = \frac{w}{2} \ .$$

L'ensemble obtenu correspond bien à l'ensemble triadique de Cantor. En effet, l'IFS permettant de générer le triadique de Cantor est ici implémenté par la machine $\mathfrak{M}_{Cantor}$. On constate avec des calculs simples qu'en partant de la configuration c correspondant au premier étage (on ne considère par la configuration initiale qui initie le processus et qui n'est pas symétrique) on obtient après une durée finie (égale à un neuvième de la largeur de c) une configuration c' telle que

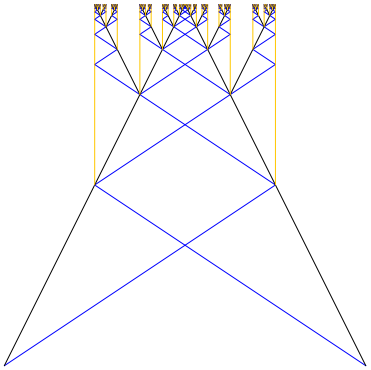
$c' = h_1(c) \cup h_2(c)$, avec h_1, h_2 respectivement définies par $h_1(x) = \frac{x}{3}$ et $h_2(x) = \frac{x}{3} + \frac{2}{3}$ pour tout $x \in \mathbb{R}$. Cette étape, ainsi que les étapes successives, correspondent bien à l'IFS de définition donné en SOUS-SEC. 5.1.2, qui est ainsi implémenté par la machine $\mathfrak{M}_{Cantor}$. La structure résultante correspond donc bien au triadique de Cantor.

Remarque 20. Ce n'est pas exactement le triadique de Cantor qui est construit ici : seules les extrémités des segments sont générées, alors que dans la version classique du triadique, il existe des points qui ne correspondent à aucune extrémité d'un segment retiré (c'est le cas par exemple du réel 1/4).

Une courbe de von Koch. Les angles présents dans la courbe de von Koch, déjà illustrée en SOUS-SEC. 5.1.2, n'autorise pas une construction directe par machine à signaux : celle-ci nécessiterait en effet des signaux de vitesse infinie du fait des segments horizontaux de la courbe. Néanmoins, une variante de la courbe de von Koch, donnée dans le papier original dont est tirée la FIG. 5.7(a), peut être approximée par signaux, en générant un diagramme comme celui de la FIG. 5.7(b). Le principe est de réitérer l'accumulation de type Zénon sur les deux bords qui encadrent les zig-zags. Une accumulation contenue entre deux signaux se rapprochant, joue alors le rôle d'un triangle dans la courbe classique de von Koch.



(a) Courbe de von Koch modifiée (version originale, [Edgar 1993, Fig. 4, p. 41]).



(b) Approximation par signaux.

FIGURE 5.7 – Construction d'une courbe de von Koch.

Le diagramme de la FIG. 5.7(b) s'obtient en générant le diagramme de la FIG. 5.8(e) et le diagramme symétrique (obtenu en prenant toutes les versions symétriques des signaux et les règles de collisions associées), qu'il suffit de « coller » suivant le signal w central (celui-ci n'est d'ailleurs plus nécessaire aux rebonds, puisque un signal arrivant de la partie symétrique joue le rôle du signal issu d'un rebond). La structure principale est celle de l'accumulation simple déjà vue dans le chapitre précédent. Deux types de règles complètent celles du « zig-zag simple », dont les rebonds sont définis dans le TAB. 5.8(b). Celles du TAB. 5.8(d) permettent de réitérer de manière récursive la construction d'une accumulation sur le bord gauche $\overrightarrow{\mathsf{w}_l}$. Les deux bords délimitant cette nouvelle structure sont générés successivement, après l'accomplissement d'un zig-zag complet dans la structure principale. Ainsi, à chaque rebond, en plus du bord principal $\overrightarrow{\mathsf{w}_l}$ et du signal $\overrightarrow{\mathsf{zig}}$ issu du rebond, trois nouveaux signaux sont générés : w, $\overleftarrow{\mathsf{zag}_0}$ et $\overleftarrow{\mathsf{w}_r}$. Le premier signal constitue le bord pour la structure suivante, tandis que les deux autres signaux complètent la structure initiée au rebond précédent. Les règles du TAB. 5.8(c) gèrent le cas du tout premier rebond, où seul un signal w doit être généré.

Bien que ces diagrammes respectent les grands lignes de la construction de la courbe de von Koch et son principe d'itération de la structure triangulaire, on note cependant plusieurs différences. Contrairement à la courbe originale, les structures ne sont ici itérées que sur un

(a) Méta-signaux et vitesses.

$\overrightarrow{\overrightarrow{\mathsf{zig}}}$, $\overrightarrow{\overrightarrow{\mathsf{zig}_0}}$	$3/2$
$\overrightarrow{\mathsf{w}_l}$	$1/2$
w	0
$\overleftarrow{\mathsf{w}_r}$	$-1/2$
$\overleftarrow{\overleftarrow{\mathsf{zag}}}$, $\overleftarrow{\overleftarrow{\mathsf{zag}_0}}$	$-3/2$

(b) Règles de rebonds.

$$\{ \mathsf{w}, \overleftarrow{\overleftarrow{\mathsf{zag}}} \} \to \{ \mathsf{w}, \overrightarrow{\overrightarrow{\mathsf{zig}}} \}$$
$$\{ \mathsf{w}, \overleftarrow{\overleftarrow{\mathsf{zag}_0}} \} \to \{ \mathsf{w}, \overrightarrow{\overrightarrow{\mathsf{zig}_0}} \}$$
$$\{ \overrightarrow{\overrightarrow{\mathsf{zig}}}, \mathsf{w} \} \to \{ \mathsf{w}, \overleftarrow{\overleftarrow{\mathsf{zag}}} \}$$
$$\{ \overrightarrow{\overrightarrow{\mathsf{zig}_0}}, \mathsf{w} \} \to \{ \mathsf{w}, \overleftarrow{\overleftarrow{\mathsf{zag}_0}} \}$$

(c) Règles d'initiation.

$$\{ \overrightarrow{\mathsf{w}_l}, \overleftarrow{\overleftarrow{\mathsf{zag}_0}} \} \to \{ \mathsf{w}, \overrightarrow{\mathsf{w}_l}, \overrightarrow{\overrightarrow{\mathsf{zig}}} \}$$
$$\{ \overrightarrow{\overrightarrow{\mathsf{zig}_0}}, \overleftarrow{\mathsf{w}_r} \} \to \{ \overleftarrow{\overleftarrow{\mathsf{zag}}}, \overleftarrow{\mathsf{w}_r}, \mathsf{w} \}$$

(d) Règles d'itération.

$$\{ \overrightarrow{\mathsf{w}_l}, \overleftarrow{\overleftarrow{\mathsf{zag}}} \} \to \{ \overleftarrow{\mathsf{w}_r}, \overleftarrow{\overleftarrow{\mathsf{zag}_0}}, \mathsf{w}, \overrightarrow{\mathsf{w}_l}, \overrightarrow{\overrightarrow{\mathsf{zig}}} \}$$
$$\{ \overrightarrow{\overrightarrow{\mathsf{zig}}}, \overleftarrow{\mathsf{w}_r} \} \to \{ \overleftarrow{\overleftarrow{\mathsf{zag}}}, \overleftarrow{\mathsf{w}_r}, \mathsf{w}, \overrightarrow{\mathsf{w}_l}, \overrightarrow{\overrightarrow{\mathsf{zig}_0}} \}$$

(e) Diagramme espace-temps.

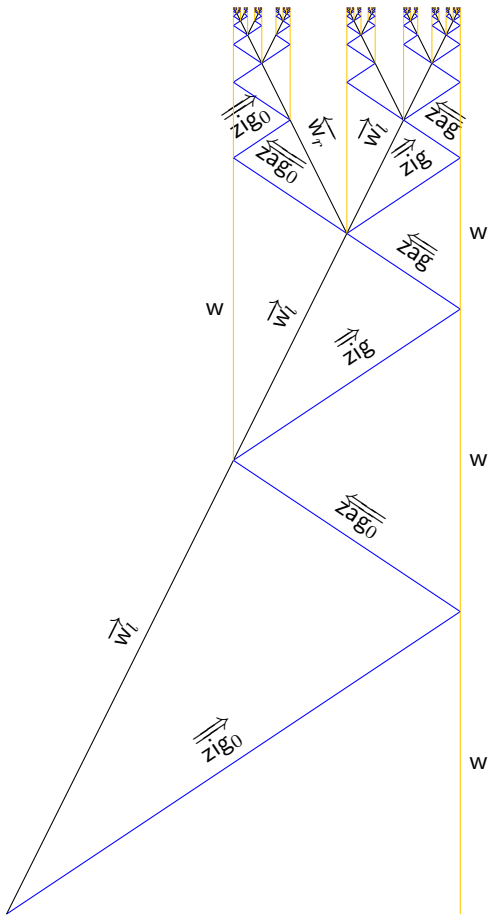


FIGURE 5.8 – La machine $\mathfrak{M}_{Koch}$: construction d'une courbe de von Koch.

seul côté de la structure triangulaire sous-jacente. De même, alors que les itérations se font normalement de part et d'autre d'un triangle, elles ne se font ici qu'en direction du haut, en suivant l'axe temporel.

Comme pour les fractales précédentes, la hauteur du diagramme ainsi généré est bornée : si la largeur est l, alors la hauteur est également l (pour des signaux $\overrightarrow{\mathsf{w}_l}$ et $\overleftarrow{\mathsf{w}_r}$ de vitesses absolues $1/2$). Notons également que le diagramme de la FIG. 5.7(b) contient une infinité d'accumulations spatiales et d'accumulations non-isolées. Il s'agit en fait d'une *accumulation d'ordre* ω, c'est-à-dire que les objets s'accumulant ne sont plus simplement des signaux ou des collisions, mais ce sont également des valeurs d'accumulations (accumulations d'ordre 2), d'accumulations d'accumulations (accumulations d'ordre 3)...

5.2.2 Génération de fractales de n'importe quelle dimension entre 0 et 1

Ensembles de Cantor généralisés. L'ensemble triadique de Cantor admet de multiples variantes : il est possible soit de retirer un intervalle central de longueur plus grande ou plus petite que un tiers, soit de retirer plusieurs intervalles de même longueur ou de longueurs différentes...

Nous nous intéressons ici à la version généralisée qui consiste à retirer un intervalle central de longueur $\gamma < 1$, à chaque itération et en appliquant le même principe que pour le triadique de Cantor.

À la $n-ième$ itération, 2^n intervalles de longueur γ^n sont donc retirés. Pour $\gamma = \frac{1}{3}$, on retrouve l'ensemble triadique classique. Les ensembles de Cantor généralisés bénéficient des mêmes

propriétés topologiques que le triadique classique (voir [Dindoš 2001] pour une étude succincte des ensembles de Cantor généralisés).

On note $\mathcal{C}_\gamma$ l'ensemble de Cantor généralisé de paramètre γ, construit en retirant des intervalles centraux de longueur $\gamma < 1$. La dimension d'un Cantor généralisé de paramètre γ est donnée par $dim_{frac}(\mathcal{C}_\gamma) = -\frac{\log 2}{\log(\frac{1-\gamma}{2})}$. En effet, la dimension d'auto-similarité se calcule en considérant le fait que $\mathcal{C}_\gamma$ est égale à deux copies de lui-même, réduites par une homothétie de rapport $\frac{1-\gamma}{2}$.

Construction par signaux. Pour chaque $\gamma \in]0;1[$, il existe une machine à signaux $\mathfrak{M}^\gamma_{Cantor}$, définie par le TAB. 5.1, qui permet de construire l'ensemble $\mathcal{C}_\gamma$. Il suffit pour cela de modifier les vitesses des méta-signaux de la machines $\mathfrak{M}_{Cantor}$, afin de créer des signaux aux positions $\frac{1-\gamma}{2}$ et $\frac{1+\gamma}{2}$ (au lieu de $\frac{1}{3}$ et $\frac{2}{3}$ pour le triadique). Pour cela, nous assignons les vitesses 1, $\frac{3+\gamma}{1-\gamma}$, $\frac{1+\gamma}{1-\gamma}$ et $\frac{3-\gamma}{1-\gamma}$ respectivement aux signaux $\mathsf{a}^{\overrightarrow{\gamma}}$, $\mathsf{a}^{\overrightarrow{\overrightarrow{\gamma}}}$, $\mathsf{b}^{\overrightarrow{\gamma}}$ et $\mathsf{b}^{\overrightarrow{\overrightarrow{\gamma}}}$. Les versions gauches de ces méta-signaux sont définies de la même façon mais avec des vitesses opposées aux versions droites. Chaque méta-signal noté μ^γ joue le même rôle que le méta-signal μ utilisé pour construire le triadique de Cantor, l'exposant γ indiquant que sa vitesse dépend du paramètre γ. Notamment, les règles de collisions sont les mêmes que celles du triadique, au détail près de l'exposant γ.

En considérant la même configuration initiale que celle utilisée pour construire le triadique, *i.e.* $c_0 = \left\{[\mathsf{w}, \mathsf{a}^{\overrightarrow{\gamma}}, \mathsf{b}^{\overrightarrow{\gamma}}, \mathsf{a}^{\overrightarrow{\overrightarrow{\gamma}}}, \mathsf{b}^{\overrightarrow{\overrightarrow{\gamma}}}]@0,\ \mathsf{w}@1\right\}$, un simple calcul montre que les signaux $\mathsf{a}^{\overrightarrow{\gamma}}$ et $\mathsf{a}^{\overleftarrow{\overleftarrow{\gamma}}}$ entrent en collision au temps $t = \frac{1-\gamma}{2}$ et en position $x = \frac{1-\gamma}{2}$. De même, les signaux $\mathsf{b}^{\overrightarrow{\gamma}}$ et $\mathsf{b}^{\overleftarrow{\overleftarrow{\gamma}}}$ entrent en collision en position $x' = \frac{1+\gamma}{2}$, et également au temps $t = \frac{1-\gamma}{2}$. Ces positions correspondent bien à celles du Cantor généralisé défini ci-dessus. De plus, ces deux positions sont marquées simultanément, ce qui garantit qu'à chaque itération, le processus reste synchronisé. Les règles de collision de la machine $\mathfrak{M}^\gamma_{Cantor}$ étant les mêmes que celles de $\mathfrak{M}_{Cantor}$, le processus est réitéré de manière symétrique sur les deux intervalles restant. La FIGURE 5.9 montre deux exemples de tels ensembles : avec des paramètres $\gamma = \frac{1}{10}$ (FIG. 5.9(a)) et $\gamma = \frac{1}{2}$ (FIG. 5.9(b)).

(a) Méta-signaux et vitesses.

$\mathsf{a}^{\overrightarrow{\overrightarrow{\gamma}}}$	$(3+\gamma)/(1-\gamma)$
$\mathsf{b}^{\overrightarrow{\overrightarrow{\gamma}}}$	$(3-\gamma)/(1-\gamma)$
$\mathsf{b}^{\overrightarrow{\gamma}}$	$(1+\gamma)/(1-\gamma)$
$\mathsf{a}^{\overrightarrow{\gamma}}$	1
w	0
$\mathsf{a}^{\overleftarrow{\gamma}}$	-1
$\mathsf{b}^{\overleftarrow{\gamma}}$	$-(1+\gamma)/(1-\gamma)$
$\mathsf{b}^{\overleftarrow{\overleftarrow{\gamma}}}$	$-(3-\gamma)/(1-\gamma)$
$\mathsf{a}^{\overleftarrow{\overleftarrow{\gamma}}}$	$-(3+\gamma)/(1-\gamma)$

(b) Règles de rebonds.

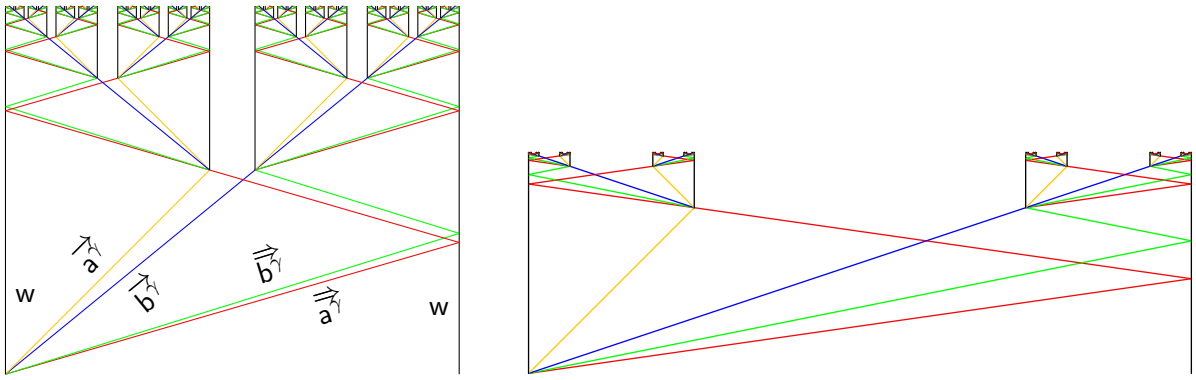
$$\{\mathsf{w}, \mathsf{a}^{\overleftarrow{\overleftarrow{\gamma}}}\} \rightarrow \{\mathsf{w}, \mathsf{a}^{\overrightarrow{\overrightarrow{\gamma}}}\}$$
$$\{\mathsf{a}^{\overrightarrow{\overrightarrow{\gamma}}}, \mathsf{w}\} \rightarrow \{\mathsf{a}^{\overleftarrow{\overleftarrow{\gamma}}}, \mathsf{w}\}$$
$$\{\mathsf{w}, \mathsf{b}^{\overleftarrow{\overleftarrow{\gamma}}}\} \rightarrow \{\mathsf{w}, \mathsf{b}^{\overrightarrow{\overrightarrow{\gamma}}}\}$$
$$\{\mathsf{b}^{\overrightarrow{\overrightarrow{\gamma}}}, \mathsf{w}\} \rightarrow \{\mathsf{b}^{\overleftarrow{\overleftarrow{\gamma}}}, \mathsf{w}\}$$

(c) Règles d'itérations.

$$\{\mathsf{a}^{\overrightarrow{\overrightarrow{\gamma}}}, \mathsf{a}^{\overleftarrow{\gamma}}\} \rightarrow \{\mathsf{w}, \mathsf{a}^{\overrightarrow{\gamma}}, \mathsf{b}^{\overrightarrow{\gamma}}, \mathsf{a}^{\overrightarrow{\overrightarrow{\gamma}}}, \mathsf{b}^{\overrightarrow{\overrightarrow{\gamma}}}\}$$
$$\{\mathsf{a}^{\overrightarrow{\gamma}}, \mathsf{a}^{\overleftarrow{\overleftarrow{\gamma}}}\} \rightarrow \{\mathsf{b}^{\overleftarrow{\overleftarrow{\gamma}}}, \mathsf{a}^{\overleftarrow{\overleftarrow{\gamma}}}, \mathsf{b}^{\overleftarrow{\gamma}}, \mathsf{a}^{\overleftarrow{\gamma}}, \mathsf{w}\}$$
$$\{\mathsf{b}^{\overrightarrow{\overrightarrow{\gamma}}}, \mathsf{b}^{\overleftarrow{\gamma}}\} \rightarrow \{\mathsf{w}, \mathsf{a}^{\overrightarrow{\gamma}}, \mathsf{b}^{\overrightarrow{\gamma}}, \mathsf{a}^{\overrightarrow{\overrightarrow{\gamma}}}, \mathsf{b}^{\overrightarrow{\overrightarrow{\gamma}}}\}$$
$$\{\mathsf{b}^{\overrightarrow{\gamma}}, \mathsf{b}^{\overleftarrow{\overleftarrow{\gamma}}}\} \rightarrow \{\mathsf{b}^{\overleftarrow{\overleftarrow{\gamma}}}, \mathsf{a}^{\overleftarrow{\overleftarrow{\gamma}}}, \mathsf{b}^{\overleftarrow{\gamma}}, \mathsf{a}^{\overleftarrow{\gamma}}, \mathsf{w}\}$$

TABLEAU 5.1 – La machine $\mathfrak{M}^\gamma_{Cantor}$: construction d'ensembles de Cantor généralisés.

Notons qu'en partant de la configuration initiale définie ci-dessus et de taille 1, le temps limite $\tilde{t}$ existe et est fini car $0 < \gamma < 1$. Il correspond à la hauteur totale de la construction et est donné par $\tilde{t} = \sum_{i=1}^{\infty} \left(\frac{1-\gamma}{2}\right)^i = \frac{1-\gamma}{1+\gamma}$. Comme pour le triadique, les diagrammes ainsi générés



(a) Avec $\gamma = \frac{1}{10}$. (b) Avec $\gamma = \frac{1}{2}$.

FIGURE 5.9 – Exemples d'ensembles de Cantor généralisés.

contiennent donc des accumulations.

Les constructions par signaux des ensembles de Cantor généralisés nous permettent de démontrer qu'il est possible de générer par signaux des ensembles de réels de dimension fractale quelconque strictement comprise entre 0 et 1 :

THÉORÈME 25
Pour tout $d \in]0,1[$, il existe une machine à signaux $\mathfrak{M}^d$ et une configuration initiale c_0 tel que l'évolution de $\mathfrak{M}^d$ à partir de c_0 produit une configuration c^d telle que la dimension fractale de $support(c^d)$ est égale à d.

Démonstration. Il suffit d'utiliser la machine $\mathfrak{M}^d = \mathfrak{M}^{\gamma}_{Cantor}$, avec $\gamma = 1 - 2^{1-\frac{1}{d}}$ et de l'exécuter sur la configuration initiale c_0 précédente. Lorsque $d \in]0;1[$, on a bien $\gamma \in]0;1[$. La configuration recherchée c^d est alors donnée par la configuration limite $c^d = c_{\tilde{t}}$. En effet, cette configuration vérifie bien : $dim_{frac}(support(c_{\tilde{t}})) = -\frac{\log 2}{\log\left(\frac{1-\left(1-2^{1-1/d}\right)}{2}\right)} = -\frac{\log 2}{\log 2^{-1/d}} = d.$ □

Remarque 21. Le cas $d = 1$ correspond en fait à l'arbre binaire de la SOUS-SEC. 5.2.1. Dans ce cas, on a $\gamma = 0$.

5.3 Approximations de fractales

Nous proposons ici deux méthodes pour approximer des fractales, que nous illustrons sur l'exemple de l'arbre binaire infini. Ces méthodes suivent les notions introduites en SOUS-SEC. 3.2.3 concernant les méthodes de résolution de problèmes par famille de machines ou par machine unique. La première méthode est une *méthode spécifique* : la machine utilisée dépend de l'approximation souhaitée. Ainsi, pour chaque nombre d'itérations, une machine à signaux différente est utilisée. La deuxième méthode nécessite quant à elle une unique machine qui permet d'obtenir n'importe quelle approximation de l'arbre infini : le nombre d'itération voulu est seulement codé dans la configuration initiale, et non pas dans la machine elle-même. Une telle méthode sera appelée *générique*.

Notons que nous utilisons implicitement comme notion d'approximation la construction d'une configuration dans laquelle des positions appartenant à la fractale sont marquées d'un signal vertical. Cette configuration est obtenue après l'exécution complète de la machine à partir de la configuration initiale correspondante. Il est également possible d'approximer l'arbre binaire infini en utilisant la machine $\mathfrak{M}_{tree}$ que l'on exécute pendant une certaine durée prédéfinie en

fonction du niveau d'approximation souhaité. Mais dans ce cas, la configuration finale obtenue sera encore munie d'une dynamique (puisque la machine, qui produit normalement la fractale complète, est ici interrompue).

5.3.1 Approximation de fractales par une famille de machines

L'idée d'approximation spécifique d'une fractale $\mathcal{F}$ est de construire une famille de machines à signaux $\left(\mathfrak{M}_{\mathcal{F}}^{spe(n)}\right)_{n\in\mathbb{N}}$ telle que pour tout $n \in \mathbb{N}$ la machine $\mathfrak{M}_{\mathcal{F}}^{spe(n)}$ fournit une approximation des n premières itérations d'une fractale. De manière générale, la définition de la machine $\mathfrak{M}_{\mathcal{F}}^{spe(n)}$ (ses méta-signaux et ses règles) dépendent donc de n.

On illustre ce principe sur l'exemple de l'arbre binaire. L'approximations spécifiques des ensembles de Cantor généralisés (incluant donc le triadique de Cantor) suivent la même idée de l'utilisation d'un ensemble de signaux compteurs avec des règles d'incrémentation.

Approximation de l'arbre binaire infini par une famille de machines. Dans le cas de l'arbre binaire, l'idée est de construire de manière directe l'arbre donné en Sous-sec. 5.2.1 en s'arrêtant au niveau n voulu. Pour cela, on utilise un ensemble de signaux compteurs $\overrightarrow{\mathsf{a}_1}$, $\overrightarrow{\mathsf{a}_2}$, …, $\overrightarrow{\mathsf{a}_n}$, et on utilise des règles d'incrémentation correspondantes, c'est-à-dire telles que les collisions impliquant un signal $\overrightarrow{\mathsf{a}_i}$ en entrée produisent un signal $\overrightarrow{\mathsf{a}_{i+1}}$ en sortie. Mis à part la gestion du compteur, ces règles sont similaires à celles utilisées en Sous-sec. 5.2.1 pour construire l'arbre infini. Une dernière forme de règle permet de gérer l'arrêt de la construction : lorsque le compteur atteint le nombre n de niveaux voulus, seuls des signaux verticaux sont générés et le processus itératif est arrêté. L'Algorithme 5.1 génère ces méta-signaux et règles à partir d'un entier n et produit la machine $\mathfrak{M}_{tree}^{spe(n)}$ qui permet de construire les n premiers niveaux de l'arbre binaire fractale.

À titre d'exemple, pour $n = 3$, l'Algo. 5.1 produit la machine $\mathfrak{M}_{tree}^{spe(3)}$, dont les méta-signaux et règles sont donnés par le Tab. 5.2. Le diagramme généré par $\mathfrak{M}_{tree}^{spe(3)}$ à partir de la configuration initiale $c_0 = \left\{\mathsf{w}@0,\ \overrightarrow{\mathsf{a}_1}@0,\ \overrightarrow{\mathsf{a}}@0,\ \mathsf{w}@1\right\}$ est donné par la Fig. 5.10 : celui-ci correspond à une approximation spécifique jusqu'au niveau 3 de l'arbre binaire infini.

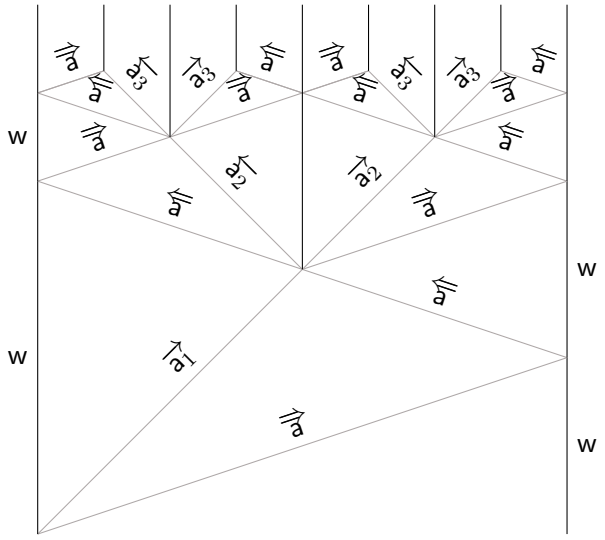


Figure 5.10 – Approximation spécifique de l'arbre fractale par la machine $\mathfrak{M}_{tree}^{spe(3)}$.

L'Algorithme 5.1 de compilation de la machine pour l'approximation des n premiers niveaux est clairement linéaire en n (car les deux boucles « **Pour** » sont bornées par n et sont

Entrée : un entier n, niveau de l'approximation.
Sortie : la machine $\mathfrak{M}_{tree}^{spe(n)}$ et c_0 la configuration initiale associée.

// méta-signaux
$\mathsf{w} \longleftarrow \mathfrak{M}_{tree}^{spe(n)}.ajouter\_méta\_signal\_de\_vitesse(0)$
$\overrightarrow{\mathsf{a}} \longleftarrow \mathfrak{M}_{tree}^{spe(n)}.ajouter\_méta\_signal\_de\_vitesse(3)$
$\overleftarrow{\mathsf{a}} \longleftarrow \mathfrak{M}_{tree}^{spe(n)}.ajouter\_méta\_signal\_de\_vitesse(-3)$
Pour i **de** 1 **à** n **Faire**
  $\overrightarrow{\mathsf{a}_i} \longleftarrow \mathfrak{M}_{tree}^{spe(n)}.ajouter\_méta\_signal\_de\_vitesse(1)$
  $\overleftarrow{\mathsf{a}_i} \longleftarrow \mathfrak{M}_{tree}^{spe(n)}.ajouter\_méta\_signal\_de\_vitesse(-1)$

// règles de collisions
$\mathfrak{M}_{tree}^{spe(n)}.ajouter\_règle\left(\{\overrightarrow{\mathsf{a}}, \mathsf{w}\} \to \{\overleftarrow{\mathsf{a}}, \mathsf{w}\}\right)$ // règles de rebonds sur les murs
$\mathfrak{M}_{tree}^{spe(n)}.ajouter\_règle\left(\{\mathsf{w}, \overleftarrow{\mathsf{a}}\} \to \{\mathsf{w}, \overrightarrow{\mathsf{a}}\}\right)$
Pour i **de** 1 **à** $n-1$ **Faire**
  $\mathfrak{M}_{tree}^{spe(n)}.ajouter\_règle\left(\{\overrightarrow{\mathsf{a}_i}, \overleftarrow{\mathsf{a}}\} \to \{\overleftarrow{\mathsf{a}}, \overleftarrow{\mathsf{a}_{i+1}}, \mathsf{w}, \overrightarrow{\mathsf{a}_{i+1}}, \overrightarrow{\mathsf{a}}\}\right)$
  $\mathfrak{M}_{tree}^{spe(n)}.ajouter\_règle\left(\{\overrightarrow{\mathsf{a}}, \overleftarrow{\mathsf{a}_i}\} \to \{\overleftarrow{\mathsf{a}}, \overleftarrow{\mathsf{a}_{i+1}}, \mathsf{w}, \overrightarrow{\mathsf{a}_{i+1}}, \overrightarrow{\mathsf{a}}\}\right)$
$\mathfrak{M}_{tree}^{spe(n)}.ajouter\_règle\left(\{\overrightarrow{\mathsf{a}_n}, \overleftarrow{\mathsf{a}}\} \to \{\mathsf{w}\}\right)$ // règles d'arrêt
$\mathfrak{M}_{tree}^{spe(n)}.ajouter\_règle\left(\{\overrightarrow{\mathsf{a}}, \overleftarrow{\mathsf{a}_n}\} \to \{\mathsf{w}\}\right)$
$\mathfrak{M}_{tree}^{spe(n)}.créer\_configuration(c_0)$
$c_0 \longleftarrow \{\mathsf{w}@0,\ \overrightarrow{\mathsf{a}_1}@0,\ \overrightarrow{\mathsf{a}}@0,\ \mathsf{w}@1\}$ // création configuration initiale
Retourner $\mathfrak{M}_{tree}^{spe(n)}$, c_0

ALGORITHME 5.1 – Générer la machine $\mathfrak{M}_{tree}^{spe(n)}$.

non imbriquées). Les complexités en collisions se déduisent facilement des diagrammes et de la définition des règles (définies de telles sorte à s'enchaîner pendant n étapes), d'où une profondeur de collisions qui est linéaire en n. La largeur de collisions est quant à elle exponentielle en n.

5.3.2 Approximation de fractales par une unique machine générique

Contrairement à l'approximation spécifique, une approximation générique d'une fractale $\mathcal{F}$ ne requiert qu'une unique machine à signaux $\mathfrak{M}_{\mathcal{F}}^{gen}$ permettant d'approximer *n'importe quel* niveau d'itération de la fractale. Les méta-signaux, et donc *a fortiori* les règles, ne dépendent pas de n : la machine $\mathfrak{M}_{\mathcal{F}}^{gen}$ est donc complètement indépendante de n. Elle doit cependant permettre de représenter n'importe quelle valeur de n : le seul moyen est donc de représenter n dans une configuration initiale adéquate $c_0^{gen(n)}$.

Là aussi, ce principe est uniquement illustré sur l'exemple de l'arbre binaire, les approximations génériques des ensembles de Cantor généralisés (incluant donc le triadique de Cantor) suivant quant à eux le même principe d'utilisation d'un ensemble fixe de signaux qui, muni d'un ensemble fixe de règles, permet de représenter le niveau n d'approximation souhaitée à l'aide d'un codage unaire de l'entier n.

Approximation générique de l'arbre binaire infini. Alors que pour l'approximation spécifique, n était représenter par n méta-signaux, l'idée ici est d'utiliser un nombre fixe de signaux qui nous permette de représenter en unaire n'importe quelle valeur de n : un entier sera donc représenté par un faisceau de n signaux, en se basant sur le codage unaire proposé dans la SOUS-SEC. 3.2.1.

Cette représentation unaire de n est ensuite insérée dans la structure de l'arbre fractale complet, dont les règles ont déjà été données en SOUS-SEC. 5.2.1. Mais nous voulons maintenant

(a) Méta-signaux et vitesses.

$\overrightarrow{\overrightarrow{\mathsf{a}}}$	3
$\overrightarrow{\mathsf{a}_1}$, $\overrightarrow{\mathsf{a}_2}$, $\overrightarrow{\mathsf{a}_3}$	1
w	0
$\overleftarrow{\mathsf{a}_1}$, $\overleftarrow{\mathsf{a}_2}$, $\overleftarrow{\mathsf{a}_3}$	-1
$\overleftarrow{\overleftarrow{\mathsf{a}}}$	-3

(b) Règles de collisions.

$\{ \mathsf{w}, \overleftarrow{\overleftarrow{\mathsf{a}}} \} \to \{ \mathsf{w}, \overrightarrow{\overrightarrow{\mathsf{a}}} \}$
$\{ \overrightarrow{\overrightarrow{\mathsf{a}}}, \mathsf{w} \} \to \{ \overleftarrow{\overleftarrow{\mathsf{a}}}, \mathsf{w} \}$
$\{ \overrightarrow{\mathsf{a}_1}, \overleftarrow{\overleftarrow{\mathsf{a}}} \} \to \{ \overleftarrow{\overleftarrow{\mathsf{a}}}, \overleftarrow{\mathsf{a}_2}, \mathsf{w}, \overrightarrow{\mathsf{a}_2}, \overrightarrow{\overrightarrow{\mathsf{a}}} \}$
$\{ \overrightarrow{\overrightarrow{\mathsf{a}}}, \overleftarrow{\mathsf{a}_1} \} \to \{ \overleftarrow{\overleftarrow{\mathsf{a}}}, \overleftarrow{\mathsf{a}_2}, \mathsf{w}, \overrightarrow{\mathsf{a}_2}, \overrightarrow{\overrightarrow{\mathsf{a}}} \}$
$\{ \overrightarrow{\mathsf{a}_2}, \overleftarrow{\mathsf{a}} \} \to \{ \overleftarrow{\overleftarrow{\mathsf{a}}}, \overleftarrow{\mathsf{a}_3}, \mathsf{w}, \overrightarrow{\mathsf{a}_3}, \overrightarrow{\overrightarrow{\mathsf{a}}} \}$
$\{ \overrightarrow{\overrightarrow{\mathsf{a}}}, \overleftarrow{\mathsf{a}_2} \} \to \{ \overleftarrow{\overleftarrow{\mathsf{a}}}, \overleftarrow{\mathsf{a}_3}, \mathsf{w}, \overrightarrow{\mathsf{a}_3}, \overrightarrow{\overrightarrow{\mathsf{a}}} \}$
$\{ \overrightarrow{\mathsf{a}_3}, \overleftarrow{\overleftarrow{\mathsf{a}}} \} \to \{ \mathsf{w} \}$
$\{ \overrightarrow{\overrightarrow{\mathsf{a}}}, \overleftarrow{\mathsf{a}_3} \} \to \{ \mathsf{w} \}$

TABLEAU 5.2 – La machine $\mathfrak{M}_{tree}^{spe(3)}$: méta-signaux et règles.

arrêter la construction récursive de l'arbre à un certain niveau donné. Nous utilisons pour cela quatre nouveaux type de méta-signaux : z et ζ qui servent de compteur ; $\mathsf{w}_\circ$, une version « désactivée » de w qui permet de décrémenter le compteur ; et enfin $\mathsf{a}_\circ$, la version de a qui permet définir la règle d'arrêt de la fractale.

Un bloc `[until(`n`)]` est alors inséré dans la configuration initiale pour couper la fractale exactement après les n premiers niveaux. Le bloc est constitué de la manière suivante : `[until(`n`)]` $= (\overrightarrow{\mathsf{z}}\,\overrightarrow{\zeta}^{n-1})$. En notant `[tree]` $= (\mathsf{w}\ \overrightarrow{\mathsf{a}}\ \overrightarrow{\overrightarrow{\mathsf{a}}}\ \mathsf{w})$ la configuration correspondant à la fractale complète, on obtient comme configuration initiale pour l'approximation générique du niveau n, la configuration $c_0^{gen(n)} =$ `[until(`n`)][tree]` $= (\overrightarrow{\mathsf{z}}\,\overrightarrow{\zeta}^{n-1}\,\mathsf{w}\ \overrightarrow{\mathsf{a}}\ \overrightarrow{\overrightarrow{\mathsf{a}}}\ \mathsf{w})$.

Détaillons le fonctionnement du compteur générique, dont les règles sont données par le TAB. 5.3. Pour coder un compteur de manière générique, on utilise la pile constituée de $n-1$ signaux $\overrightarrow{\zeta}$ et d'un signal $\overrightarrow{\mathsf{z}}$. Ce faisceau va ensuite se propager dans l'arbre, en étant à la fois « décrémenté » et dupliqué à chaque étage. Les signaux $\overrightarrow{\zeta}$ servent à la fois de compteur et de protection pour le signal $\overrightarrow{\mathsf{z}}$. En effet, le faisceau $\overrightarrow{\zeta}^{n-1}$ constitue un ensemble d'inhibiteurs du signal $\overrightarrow{\mathsf{z}}$: en transformant le signal w en une version désactivée $\mathsf{w}_\circ$, ils empêchent une collision directe entre $\overrightarrow{\mathsf{z}}$ et w. Ce dernier $\mathsf{w}_\circ$ redevient w après le passage de $\overrightarrow{\mathsf{z}}$, conformément aux règles du TAB. 5.3(b) et au diagramme de la FIG. 5.11(b). Un inhibiteur $\overrightarrow{\zeta}$ est consommé à chaque niveau et après $n-1$ niveaux, $\overrightarrow{\mathsf{z}}$ qui n'est alors plus protégé, peut interagir directement avec w. Les règles du TAB. 5.3(c) permettent alors de dupliquer le signal $\overrightarrow{\mathsf{z}}$ en des versions rapides $\overrightarrow{\overrightarrow{\mathsf{z}}}$ et $\overleftarrow{\overleftarrow{\mathsf{z}}}$, qui vont rattraper les signaux $\overrightarrow{\mathsf{a}}$ et $\overleftarrow{\mathsf{a}}$ de constructions de l'arbre pour les transformer respectivement en $\overrightarrow{\mathsf{a}_\circ}$ et $\overleftarrow{\mathsf{a}_\circ}$. Ces derniers ont alors pour rôle est d'annihiler les signaux $\overrightarrow{\mathsf{a}}$ et $\overleftarrow{\mathsf{a}}$ de construction de l'arbre fractal : la règle $\{\overrightarrow{\overrightarrow{\mathsf{a}}}, \overleftarrow{\mathsf{a}_\circ}\} \to \emptyset$ et sa version gauche permettent alors d'arrêter la génération de l'arbre infini. Cette dernière étape est illustrée par la FIG. 5.11(b).

Les conditions de validité à vérifier sont que le faisceau défini par `[until(`n`)]` n'empiète pas sur plusieurs branches de l'arbre en même temps, et que les signaux rapides $\overrightarrow{\overrightarrow{\mathsf{z}}}$ et $\overleftarrow{\overleftarrow{\mathsf{z}}}$ rattrapent bien les signaux $\overrightarrow{\mathsf{a}}$ et $\overleftarrow{\mathsf{a}}$. En fait, la validité de la deuxième condition implique celle de la première, et un simple calcul montre qu'il faut que la largeur du faisceau $(\overrightarrow{\mathsf{z}}\,\overrightarrow{\zeta}^{n-1}\,\overrightarrow{\mathsf{a}})$ soit inférieure à un tiers de la largeur du dernier niveau, c'est-à-dire inférieure à $\frac{1}{3}\times\frac{1}{2^n}$. Ainsi, pour un construire l'approximation entre deux murs placés en 0 et 1, il suffit que le faisceau soit entièrement inclus dans $\left]-\frac{1}{3\times 2^n}; 0\right[$. Pour approximer les n premiers niveaux de l'arbre, la configuration initiale $c_0^{gen(n)} = \left\{\overrightarrow{\mathsf{z}}@\frac{-n}{3\times 2^n\times(n+1)},\ \overrightarrow{\zeta}@\frac{-(n-1)}{3\times 2^n\times(n+1)}, \ldots, \overrightarrow{\zeta}@\frac{-1}{3\times 2^n\times(n+1)},\ [\mathsf{w}, \overrightarrow{\mathsf{a}}, \overrightarrow{\overrightarrow{\mathsf{a}}}]@0,\ \mathsf{w}@1\right\}$ convient donc.

(a) Méta-signaux et vitesses.

$\overrightarrow{\overrightarrow{z}}$	3
$\overrightarrow{\zeta}$, $\overrightarrow{z}$, $\overrightarrow{a_\circ}$	1
w, $w_\circ$	0
$\overleftarrow{\zeta}$, $\overleftarrow{z}$, $\overleftarrow{a_\circ}$	−1
$\overleftarrow{\overleftarrow{z}}$	−3

(b) Décrémenter le compteur.

$\{ \overrightarrow{\zeta}, w \} \to \{ w_\circ \}$

$\{ \overrightarrow{\zeta}, w_\circ \} \to \{ \overleftarrow{\zeta}, w_\circ, \overrightarrow{\zeta} \}$

$\{ \overrightarrow{z}, w_\circ \} \to \{ \overleftarrow{z}, w, \overrightarrow{z} \}$

(c) Étape d'arrêt.

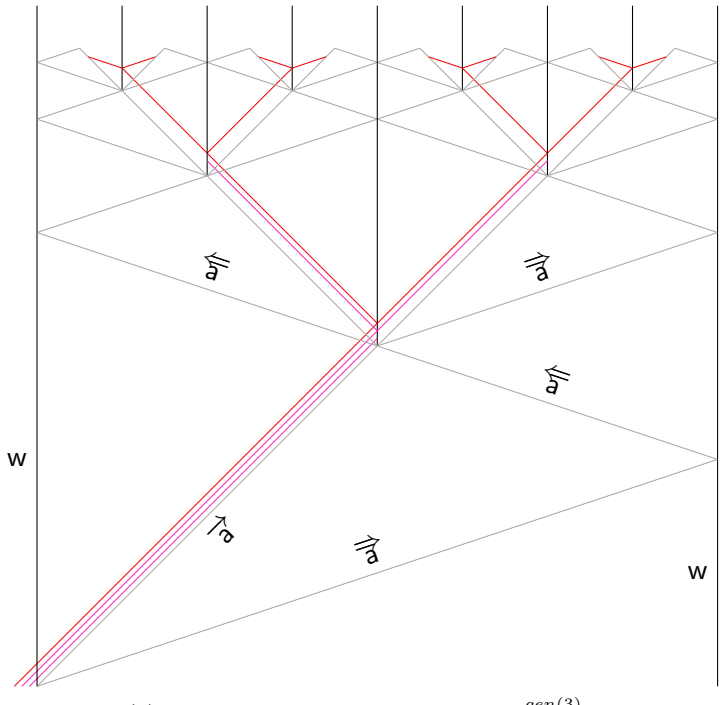
$\{ \overrightarrow{z}, w \} \to \{ \overrightarrow{\overrightarrow{z}}, w, \overleftarrow{\overleftarrow{z}} \}$

$\{ \overrightarrow{\overrightarrow{z}}, \overrightarrow{a} \} \to \{ \overrightarrow{a_\circ} \}$

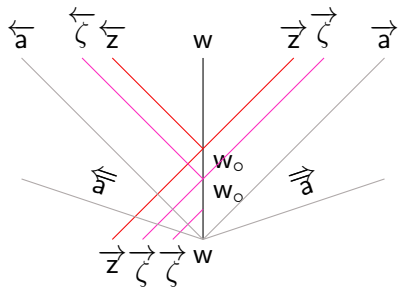
$\{ \overrightarrow{\overrightarrow{a}}, \overleftarrow{a_\circ} \} \to \emptyset$

TABLEAU 5.3 – Couper l'arbre fractale (règles version droite).

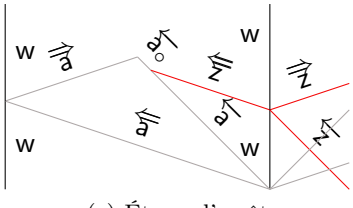
La SOUS-SECTION 7.2.2 présentera une condition du même type pour assurer la validité d'une utilisation calculatoire de l'arbre et de son approximation. Pour $n = 3$, on obtient le diagramme de la FIG. 5.11, à partir de la configuration initiale ($\overrightarrow{z}$ $\overrightarrow{\zeta}$ $\overrightarrow{\zeta}$ w $\overrightarrow{a}$ $\overrightarrow{\overrightarrow{a}}$ w), les positions initiales exactes étant données par $c_0^{gen(3)} = \left\{ \overrightarrow{z}@\frac{-3}{100},\ \overrightarrow{\zeta}@\frac{-2}{100},\ \overrightarrow{\zeta}@\frac{-1}{100},\ w@0,\ \overrightarrow{a}@0,\ \overrightarrow{\overrightarrow{a}}@0,\ w@1 \right\}$.



(a) Diagramme complet à partir de $c_0^{gen(3)}$.



(b) Décrémentation.



(c) Étape d'arrêt.

FIGURE 5.11 – Approximation générique de l'arbre fractale par la machine $\mathfrak{M}_{tree}^{gen}$.

Nous verrons au CHAP. 8 une technique — appelée *la lentille* — qui garantit un rétrécissement automatique du faisceau à chaque niveau. Dans ce cas, la validité de l'approximation de l'arbre est garantie en plaçant le faisceau dans un intervalle initial fixé, et indépendamment du niveau n d'approximation souhaité.

Chapitre 6

Modules géométriques

Les preuves de corrections des algorithmes géométriques simples présentés dans le Chap. 3, nécessitaient des preuves peu complexes mais longues et fastidieuses pour démontrer des faits apparemment basiques. Une preuve de type combinatoire, comme la preuve faite en Sous-sec. 3.1.3 pour la correction de l'additionneur binaire, permet de simplifier la chose. Mais sur des constructions plus complexes, la longueur d'une preuve combinatoire explosera, nécessitant l'étude de nombreux sous-cas. C'est pourquoi il est nécessaire pour prouver la correction de constructions géométriques complexes, de développer une approche de type *modulaire*. Cela consiste à ramener l'étude d'un diagramme espace-temps à l'étude — plus simple — de chacune de ses parties, de telle sorte que les propriétés globales du diagramme se déduisent de celles de chacune de ses parties.

Une approche similaire a déjà été proposée dans le cadre des automates cellulaires par Gaétan Richard [Richard 2008]. Dans ces travaux, les diagrammes d'AC définis à partir de systèmes de particules et collisions discrètes sont décomposés en composantes géométriques (les « faces »). Leur agencement est défini par des schémas de ligatures de faces, qui sont ensuite décrits à l'aide de formules logiques dans l'arithmétique de Presburger, la validité du diagramme espace-temps se déduisant ainsi de celle de la formule correspondante. Citons également Florent Becker qui a introduit dans [Becker 2009] un langage de programmation géométrique basé sur des systèmes de signaux pour l'auto-assemblage.

Ici, l'idée est de se placer à un niveau « macro ». Nous ne considérons pas directement le niveau atomique des signaux/collisions, mais nous regardons des portions d'espace-temps comme des boîtes ayant une certaine fonctionnalité : nous appellerons de telles structures des *modules*. Un module sera défini par une machine à signaux munie d'ensembles de configurations (les entrées/sorties du module), à partir desquels une interprétation du diagramme espace-temps est définie. Nous définissons ensuite différentes manières d'agencer les modules entre eux, obtenant ainsi différents types de composition des fonctionnalités.

Avant de définir les notions de modules géométriques, nous commencerons par définir des opérateurs sur les machines à signaux, tels que l'union ou la jointure de machines, qui permettent de construire de nouvelles machines à partir de machines déjà existantes. Ces opérateurs serviront ensuite de bases pour définir les modules géométriques. Ceux-ci seront présentés dans la seconde section. Nous définirons d'abord les modules de manière générale (les *H-modules*), puis ensuite des types particuliers de modules seront présentés (les *T-modules* et les *R-modules*). Les différentes opérations de composition de ces modules (compositions séquentielles, compositions parallèles) seront ensuite étudiées, en détaillant particulièrement les compositions parallèles à travers divers exemples qui serviront à prouver certaines constructions dans les chapitres suivants. Enfin, le lien entre les complexités en collisions et les différentes compositions de modules sera abordé : des bornes sur les complexités en collisions d'une composition de modules seront fournies en fonction

de celles des modules utilisés pour la composition.

6.1 Opérations sur les machines

Nous définissons ici quelques opérations de bases sur les machines, dont le but est de pouvoir construire facilement de nouvelles machines à partir de machines déjà existantes. Nous définissons d'abord un concept de *sous-machines à signaux*, et nous introduisons ensuite des opérateurs (l'*union de machines* et les *schémas d'extension de machines*) permettant d'étendre des machines de telle sorte que les machines auxquelles sont appliqués ces opérateurs soient des sous-machines des nouvelles machines étendues.

6.1.1 Sous-machines et unions de machines

Sous-machines à signaux. Nous présentons ici une notion de *sous-machines à signaux*. Celle-ci justifie les opérateurs d'extension de machines qui seront définis par la suite. La notion de sous-machines doit se comprendre dans un sens intuitif : une machine à signaux $\mathfrak{M}_2$ est une sous-machine à signaux de la machine $\mathfrak{M}_1$ si ses méta-signaux sont tous des méta-signaux de la machine $\mathfrak{M}_1$, et que leurs vitesses concordent. De plus, les règles définies à partir des méta-signaux de $\mathfrak{M}_2$ sont exactement les mêmes que les règles de $\mathfrak{M}_1$ impliquant des méta-signaux de $\mathfrak{M}_2$ en entrée. Formellement :

Définition 38 (Sous-machine)
Soit $\mathfrak{M}_1 = (\mathcal{M}_1, \mathcal{S}_1, \mathcal{R}_1)$ *une machine. On dit que la machine* $\mathfrak{M}_2 = (\mathcal{M}_2, \mathcal{S}_2, \mathcal{R}_2)$ *est une* sous-machine *de* $\mathfrak{M}_1$ *si :*

(i) $\mathcal{M}_2 \subseteq \mathcal{M}_1$;

(ii) $\mathcal{S}_2 = \mathcal{S}_{1\restriction\mathcal{M}_2}$;

(iii) $\mathcal{R}_2 = \mathcal{R}_{1\restriction\mathcal{M}_2}$.

L'hypothèse que $\mathfrak{M}_2$ est une machine à signaux est nécessaire, pour garantir que les règles de collisions de $\mathfrak{M}_2$ sont bien déterministes (ce que la condition *(iii)* ne garantit pas à elle toute seule).

Union simple de machines. On définit ici l'*union simple de deux machines* de manière intuitive. Naïvement, l'union simple de deux machines est la machine dont les ensemble de méta-signaux et règles sont donnés les unions respectives des ensembles de méta-signaux et règles. Il faut cependant faire attention aux objets communs aux deux machines : un méta-signal présent dans les deux machines doit avoir la même vitesse dans ces deux machines et aucune machine ne doit avoir de règle de collisions contredisant une règle de l'autre machine. Si une de ces conditions n'est pas respectée, alors l'union simple ne définira pas une machine à signaux.

Définition 39 (Union simple de machines)
Soient $\mathfrak{M}_1 = (\mathcal{M}_1, \mathcal{S}_1, \mathcal{R}_1)$ *et* $\mathfrak{M}_2 = (\mathcal{M}_2, \mathcal{S}_2, \mathcal{R}_2)$ *deux machines à signaux telles que tout signal* $\mu \in \mathcal{M}_1 \cap \mathcal{M}_2$ *vérifie* $\mathcal{S}_1(\mu) = \mathcal{S}_2(\mu)$ *et pour toutes règles* $\rho_1 \in \mathcal{R}_1, \rho_2 \in \mathcal{R}_2$ *telles que* $\rho_1^- = \rho_2^-$ *on a* $\rho_1^+ = \rho_2^+$. *On définit alors* $\mathfrak{M}_1 \cup \mathfrak{M}_2 = (\mathcal{M}, \mathcal{S}, \mathcal{R})$ *la* machine union simple des machines $\mathfrak{M}_1$ et $\mathfrak{M}_2$, *par :*

(i) $\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2$;

(ii) $\mathcal{S} : \mathcal{M} \to \mathbb{R}$ *telle que* $\mathcal{S}(\mu) = \begin{cases} \mathcal{S}_1(\mu) & \text{si } \mu \in \mathcal{M}_1 \\ \mathcal{S}_2(\mu) & \text{sinon} \end{cases}$;

(iii) $\mathcal{R}^{sign} = \mathcal{R}_1^{sign} \cup \mathcal{R}_2^{sign}$ *est l'ensemble des règles non-blanches ;*
et $\mathcal{R} = \mathcal{R}^{sign} \cup \{\rho^- \to \rho^- \mid \rho^-$ *n'est un ensemble de signaux entrants dans* $\mathcal{R}^{sign}\}$.

Les conditions de la définition sont importantes : elles signifient que tous les signaux à la fois présents dans les deux machines d'origine ont bien le même comportement. En particulier, cela signifie qu'il ne doit pas y avoir de règles de collisions définies de manière contradictoire c'est-à-dire il ne doit pas exister de règles ayant les mêmes signaux entrants (communs aux deux machines) mais ayant des signaux sortants différents, à l'exception des règles de collisions blanches. Ainsi, une règle pourra être blanche pour une machine, mais non-blanche pour l'autre. Dans ce cas, c'est la règle non-blanche qui est retenue pour la définition de la machine union. C'est le sens de la condition *(iii)*, où la notation $\mathcal{R}^{sign}$ désigne l'ensemble des règles significatives (*i.e.* non-blanches), toutes les autres règles n'apparaissant pas dans $\mathcal{R}^{sign}$ étant considérées comme blanches.

Lorsque l'union simple des machines $\mathfrak{M}_1$ et $\mathfrak{M}_2$ est bien définie, alors les machines $\mathfrak{M}_1$ et $\mathfrak{M}_2$ sont des sous-machines de la machine union $\mathfrak{M}_1 \cup \mathfrak{M}_2$. Notons que dans l'union de deux machines, aucune nouvelle règle n'est formée, que ce soit à partir de règles déjà existantes ou à partir du nouvel ensemble de méta-signaux. Nous allons maintenant introduire un opérateur qui permet justement d'enrichir une machine de nouvelles règles.

6.1.2 Schémas d'extension de machines

Il est possible de modifier des machines en appliquant des règles génériques sur des sous-ensembles de signaux pour produire de nouvelles règles. De telles règles sont appelées des *schémas d'extension de machines*. Le principe est d'ajouter de nouvelles règles (et si nécessaire, de nouveaux méta-signaux) à la machine selon un schéma qui dépend d'un sous-ensemble de méta-signaux donnés. Le schéma peut également utiliser certains signaux et règles comme paramètres, indépendants de la machine à laquelle est appliqué le schéma. Un *schéma d'extension appliqué la machine* $\mathfrak{M}$ ayant un sous-ensemble de méta-signaux $\mathcal{M}' \subseteq \mathcal{M}$, revêt la forme suivante :

1. *ajouter les éventuels paramètres du schéma, donnés par des méta-signaux* $\sigma_1, \dots \sigma_m$ *et des règles définies sur ces méta-signaux uniquement ;*
2. *pour tout méta-signal* $\mu \in \mathcal{M}' \subseteq \mathcal{M}$, *pour* $i \in [\![1; m]\!]$ *:*
   (i) ajouter de nouveaux méta-signaux $\mu_1^i, \dots, \mu_n^i$ *;*
   (ii) ajouter la règle $\{\mu, \sigma_i\} \to \{\mu_1^i, \dots, \mu_n^i\} \cup \{\sigma_j\}_{j \in I^{\mu,i}}$, *où* $I^{\mu,i} \subseteq [\![1; m]\!]$.

Tous les méta-signaux $\sigma_1, \dots \sigma_m$, ainsi que les règles associées ne dépendent que du schéma d'extension considéré. Les méta-signaux $\mu_1^i, \dots, \mu_n^i$ dépendent quant à eux du signal μ et du signal paramètre σ_i, dont on veut justement définir la collision. La condition 2.(ii) signifie que la sortie de la collision entre μ et σ_i ne dépend que de ces deux signaux, et que les signaux de sortie sont parmi des signaux paramètres et des signaux ajoutés en 2.(i).

On peut définir de la même façon des schémas de règles pouvant avoir plus de deux signaux entrants. On peut également définir des schémas de règles impliquant des méta-signaux provenant de deux ou plusieurs sous-ensembles de méta-signaux distincts, avec une fonction permettant d'associer au sein d'une nouvelle règle de collisions des méta-signaux provenant de ces différents ensembles. Certaines schémas d'extension peuvent s'appliquer à n'importe quelle machine à signaux vérifiant certaines propriétés : par exemple, les machines n'ayant que des méta-signaux de vitesses positives, les machines admettant des règles de rebond définies que pour un seul côté...

Un schéma d'extension correspond à une méthode pour générer de nouvelles règles selon un schéma prédéfini : par conséquent, à tout schéma d'extension correspond un algorithme, qui prend en entrée une machine à signaux vérifiant certaines conditions et qui retourne la machine enrichie des nouveaux objets (signaux et règles) correspondant au schéma d'extension. Nous expliciterons donc souvent un schéma d'extension par l'algorithme associé.

Notons que des machines auxquelles ont été appliquées des schémas d'extension constituent des sous-machines des nouvelles machines obtenus grâce au schémas d'extension. Nous allons illustrer ce concept de schéma d'extension de machines par deux exemples, qui seront grandement utiles pour la suite.

Dupliquer un faisceau. Nous avons vu au CHAP. 3 que les faisceaux permettaient de coder simplement et de manière concise de l'information (*e.g.* des valeurs entières ou réelles). Nous donnons ici un exemple de schéma d'extension qui permet de définir à partir d'une *machine-faisceau* $\mathfrak{M}$ une nouvelle machine $\mathfrak{M}^{split}$ qui étend $\mathfrak{M}$ et qui permet de dupliquer un faisceau d'information et de le propager dans deux directions opposées de l'espace. On rappelle qu'une *machine-faisceau de vitesse* ν est une machine telle que tous ses méta-signaux sont de vitesse ν ; une telle machine ne contient donc pas de règle de collisions. Le schéma d'extension présenté ici peut s'appliquer à toute machine-faisceau de vitesse ν non nulle et est défini par :

1. *ajouter le méta-signal paramètre* w *de vitesse* 0 *;*
2. *pour tout méta-signal* $\mu \in \mathcal{M}$ *:*
   (i) ajouter le méta-signal μ^{sym} *de vitesse* $-\nu$ *;*
   (ii) ajouter la règle de collisions $\{\mu, \mathsf{w}\} \to \{\mu^{sym}, \mathsf{w}, \mu\}$.

De manière plus générale, ce schéma peut en fait s'appliquer à des machines quelconques si on restreint l'application du schéma à un sous-ensemble de méta-signaux qui ont tous la même vitesse. L'application effective de ce schéma d'extension est donnée par l'ALGO. 6.1.

Entrée : $\mathfrak{M} = (\mathcal{M}, \mathcal{S}, \mathcal{R})$ une machine-faisceau de vitesse ν.
Condition : $\nu \neq 0$.
Sortie : $\mathfrak{M}^{split}$, la machine $\mathfrak{M}$ pouvant dupliquer des faisceaux.

```
   // ajout de méta-signaux
1  w ⟵ M^split.ajouter_méta_signal_de_vitesse(0)        // signal duplicateur
2  Pour chaque μ ∈ M Faire                              // version symétrique des signaux
3  |  μ_sym ⟵ M^split.ajouter_méta_signal_de_vitesse(−ν)
   // ajout de règles
4  Pour chaque μ ∈ M Faire
5  |  M^split.ajouter_règle({w, μ} → {μ_sym, w, μ})     // duplication du signal μ
6  Retourner M^split
```

ALGORITHME 6.1 – Générer une machine dupliquant des faisceaux.

Ici, le signal central du schéma d'extension est le signal w : c'est en effet lui qui va dupliquer tout signal le rencontrant, et qui va produire ce même signal ainsi que sa version symétrique. La FIGURE 6.1 illustre le principe de la duplication : ici, le diagramme est généré par la machine obtenue après application de l'ALGO. 6.1 à une machine-faisceau dont l'ensemble des méta-signaux est $\mathcal{M} = \{\mu_i\}_{1 \leq i \leq 5}$ (tous les méta-signaux étant de vitesse 1). Il est également possible d'ajouter des signaux et règles pour « nettoyer » les signaux utilisés pour la duplication. Cela s'effectue à l'aide d'un méta-signal paramètre clean, également de vitesse ν et avec la règle paramètre $\{\mathsf{clean}, \mathsf{w}\} \to \emptyset$.

Concentrer un faisceau. De même qu'il est utile de pouvoir « copier » des données en dupliquant un faisceau, il est intéressant de pouvoir rendre un faisceau aussi fin que l'on souhaite. Sur le même principe que pour le schéma d'extension de duplication d'un faisceau, il est possible de construire à partir d'une machine-faisceau une nouvelle machine pouvant resserrer le faisceau. Nous appelons *lentille* l'ensemble des nouveaux méta-signaux et nouvelles règles ajoutés à la

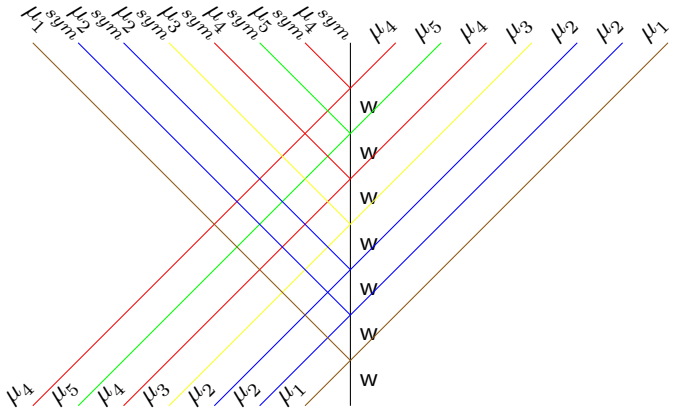


FIGURE 6.1 – Duplication d'un faisceau.

machine : ces objets servent en effet à concentrer le faisceau de la machine initiale, à la manière d'une lentille optique. Comme pour l'exemple de la duplication, le schéma d'extension de la lentille peut s'appliquer à toute machine-faisceau de vitesse non nulle. Ici, deux schémas de règles sont utilisés, le premier pour accélérer un faisceau et le deuxième pour le ralentir :

1. *ajouter le méta-signal paramètre* accel *de vitesse* 3ν *;*
2. *pour tout méta-signal* $\mu \in \mathcal{M}$ *:*
   (i) ajouter le méta-signal μ^{fast} *de vitesse* 3ν *;*
   (ii) ajouter la règle de collisions $\{\mu, \mathsf{accel}\} \to \{\mathsf{accel}, \mu^{fast}\}$ *;*

et

1. *ajouter le méta-signal paramètre* deccel *de vitesse* 0 *;*
2. *pour tout méta-signal* $\mu^{fast} \in \mathcal{M}'$ *(où* $\mathcal{M}'$ *est l'ensemble des méta-signaux de vitesse* 3ν*) :*
   (i) ajouter le méta-signal μ *de vitesse* ν *;*
   (ii) ajouter la règle de collisions $\{\mu^{fast}, \mathsf{deccel}\} \to \{\mathsf{deccel}, \mu\}$.

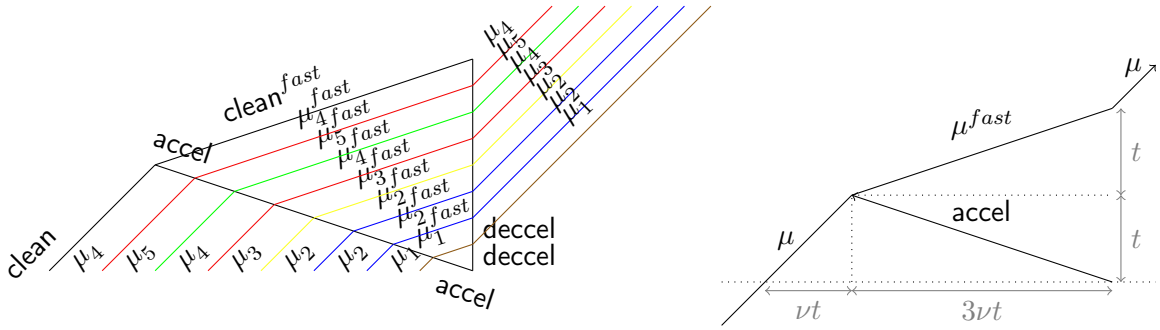
La première permet d'accélérer les signaux, tandis que la deuxième réutilise les méta-signaux générés par la première règle et génère les règles de collisions pour ralentir les signaux rapides. Le schéma de la lentille s'obtient en appliquant successivement ces deux schémas à une machine. L'ALGORITHME 6.2 permet d'appliquer automatiquement le schéma à une machine-faisceau de vitesse non nulle. La nouvelle machine peut engendrer des diagrammes comme ceux de la FIG. 6.2(a) : il s'agit d'un diagramme engendré par la machine obtenu après application de l'ALGO. 6.2 à la machine-faisceau $\mathfrak{M}$ de l'exemple précédent, et la lentille est ici appliqué au faisceau déjà utilisé dans la FIG. 6.1. Comme pour le schéma de duplication, il est également possible d'ajouter des règles permettant de nettoyer les signaux de construction de la lentille, grâce aux signaux paramètres clean et clean_{fast} (respectivement de vitesse ν et 3ν) et grâce aux règles $\{\mathsf{accel}, \mathsf{clean}\} \to \{\mathsf{clean}_{fast}\}$ et $\{\mathsf{deccel}, \mathsf{clean}_{fast}\} \to \emptyset$. La version de lentille utilisée dans le diagramme de la FIG. 6.2(a) est un exemple de version « nettoyante » qui supprime les signaux paramètres propres à la lentille (accel et deccel). Nous utiliserons par la suite une version de la lentille qui conserve ces signaux pour une éventuelle utilisation ultérieure.

L'exemple de lentille donné ici correspond à un rétrécissement par 2 de la largeur du faisceau, mais il est possible de concevoir des lentilles pour n'importe quel facteur de rétrécissement d'un faisceau, en changeant les vitesses du signal accel. La FIGURE 6.2(b) illustre le fait que le facteur de la lentille est effectivement 2 dans le cas présent (pour un faisceau de vitesse ν et un signal accel de vitesse 3ν) : sur la figure, il est clair que le faisceau entrant est de largeur $4\nu t$, et est de hauteur $2t$ en sortie de lentille, *i.e.* de largeur $2\nu t$ (car le faisceau sortant est de nouveau de vitesse ν).

Entrée : $\mathfrak{M} = (\mathcal{M}, \mathcal{S}, \mathcal{R})$ une machine-faisceau de vitesse ν.
Condition : $\nu \neq 0$.
Sortie : $\mathfrak{M}^{lens}$, la machine $\mathfrak{M}$ pouvant concentrer des faisceaux.

```
// ajout de méta-signaux
```
1 accel $\longleftarrow \mathfrak{M}^{lens}.ajouter\_méta\_signal\_de\_vitesse(-3\nu)$ `// signal d'accélération`
2 deccel $\longleftarrow \mathfrak{M}^{lens}.ajouter\_méta\_signal\_de\_vitesse(0)$ `// signal de décélération`
3 **Pour chaque** $\mu \in \mathcal{M}$ **Faire** `// version rapide des signaux`
4 $\quad \mu_{fast} \longleftarrow \mathfrak{M}^{lens}.ajouter\_méta\_signal\_de\_vitesse(3\nu)$
```
// ajout de règles
```
5 **Pour chaque** $\mu \in \mathcal{M}$ **Faire**
6 $\quad \mathfrak{M}^{lens}.ajouter\_règle\,(\{\text{accel}, \mu\} \rightarrow \{\text{accel}, \mu_{fast}\})$ `// accélération`
7 $\quad \mathfrak{M}^{lens}.ajouter\_règle\,(\{\text{deccel}, \mu_{fast}\} \rightarrow \{\text{deccel}, \mu\})$ `// décélération`
8 **Retourner** $\mathfrak{M}^{lens}$

ALGORITHME 6.2 – Générer une machine concentrant des faisceaux (avec la valeur 2 comme paramètre de réduction du faisceau).



(a) Diagramme d'une lentille.

(b) Principe de la lentille.

FIGURE 6.2 – Concentration d'un faisceau.

Autres exemples de schémas d'extension. Il existe de nombreux autres exemples de schémas d'extension, que nous ne détaillons pas ici. Ainsi, de nombreuses constructions présentées dans [Durand-Lose 2003], telles que les translations ou contractions automatiques, peuvent se formuler sous forme de schémas d'extension et en constituent autant d'autres exemples. D'autres applications des schémas d'extension méritent également d'être mentionnées : un cas intéressant est celui de schémas de « symétrisation » de machines à signaux. Ainsi, la machine de l'exemple précédent peut être étendue en une nouvelle machine avec une lentille permettant d'accélérer et de rétrécir des faisceaux dans les deux directions. En mélangeant les schémas de duplication et de la lentille, il est aisé de concevoir un nouveau schéma d'extension qui transforme des machines-faisceaux de telle sorte à pouvoir à la fois dupliquer un faisceau tout en le concentrant (il suffit de changer le schéma de règle de décélération dans la lentille pour que tout signal décéléré soit également dupliqué).

Comme nous l'avons déjà mentionné, les schémas d'extension peuvent avoir des schémas de règles beaucoup plus complexes, et un même schéma pourra contenir plusieurs formes de règles en fonction de différents cas à traiter. Ainsi, le schéma de duplication peut être décrit en fonction de plusieurs sous-ensembles de méta-signaux de telle sorte à soit dupliquer un signal, soit le supprimer, ou encore soit le dupliquer en le remplaçant par un autre type de signal et son symétrique, en fonction de l'ensemble auquel appartient le signal. Nous verrons des applications des schémas de duplication, de lentille ainsi que des variantes (lentille avec duplication, duplication avec suppressions de certains signaux...) dans les chapitres suivants.

6.2 Modules géométriques : définitions et exemples

À partir de machines simples calculant des fonctions « de bases », on construit d'autres machines calculant des fonctions plus complexes. On utilise pour cela une notion de *module*. Grossièrement, un module est constitué d'une machine à signaux, d'entrées/sorties et d'une structure géométrique polygonale. Les entrées/sorties seront données par des ensembles de configurations. La structure géométrique associée à un module est une zone définie par un polygone (c'est donc une surface plane) qui contient toute l'évolution de la machine. Elle fournit un contrôle spatio-temporel sur la machine. La sortie d'un module est donnée par la configuration obtenue après l'évolution de la machine pendant une durée définie, en fonction de l'entrée.

Un module sera ensuite muni d'une *interprétation* : des fonctions de représentation seront associées aux ensembles d'entrées et de sorties, de telle sorte que le module sera interprété comme calculant une fonction. L'interprétation d'un module n'est pas directement incluse dans la définition d'un module, car il nous apparaît nécessaire de différencier le niveau dynamique (le module) du niveau sémantique (son interprétation).

6.2.1 Notion de H-Module

Afin de ne pas alourdir les notations, les fonctions d_1, d_2, s_1, s_2, l_1 et l_2 apparaissent ici comme des paramètres mais tous dépendent de la configuration d'entrée c^{in}. Sauf quand le contexte l'impose, on parlera de paramètres au lieu de fonctions et l'argument c^{in} sera omis : par exemple, le paramètre d_1 désignera la valeur $d_1(c^{in})$. De même, on allège les notations de $x_{min}^{c^{in}}$ et $x_{max}^{c^{in}}$ que l'on écrira juste x_{min} et x_{max}, qui désignent les éléments extrémaux de $support(c^{in})$ et qui sont donc dépendants de c^{in}. On rappelle que la taille d'une configuration, $|c^{in}|$, est donnée par $|c^{in}| = x_{max}^{c^{in}} - x_{min}^{c^{in}}$. Donnons maintenant la définition formelle d'un H-module, dont les paramètres géométriques sont illustrés par la Fig. 6.3 :

Définition 40 (H-module)
Un H-module A *est un* 4*-uplet* $A=(\mathfrak{A}, \mathcal{C}onfig^{in}, \mathcal{C}onfig^{out}, \mathcal{Z})$ *tel que :*

(i) $\mathfrak{A}$ *est une machine à signaux, appelée* machine sous-jacente *du module A ;*

(ii) $\mathcal{C}onfig^{in}$ *et* $\mathcal{C}onfig^{out}$ *sont des ensembles de configurations de la machine* $\mathfrak{A}$*, appelés respectivement* ensembles d'entrées et sorties *du module A ;*

(iii) $\mathcal{Z}$ *est un « hexagone » défini par le* 6*-uplet* $\mathcal{Z} = (d_1, d_2, s_1, s_2, l_1, l_2)$ *de fonctions-paramètres* $d_1, d_2 : \mathcal{C}onfig^{in} \to \mathbb{R}^+$ *telles que* $d_1 + d_2 > 0$ *;* $s_1, s_2 : \mathcal{C}onfig^{in} \to \mathbb{R}$ *et* $l_1, l_2 : \mathcal{C}onfig^{in} \to \mathbb{R}^+$ *vérifiant pour toute* $c^{in} \in \mathcal{C}onfig^{in}$ *et pour tout* $t \in [0; d_1]$ *:*

$$c_t\Big(\mathbb{R} \setminus \Big[x_{min} + \frac{s_1}{d_1} \times t \ ; \ x_{max} - \frac{x_{max} - x_{min} - s_1 - l_1}{d_1} \times t\Big]\Big) = \{\oslash\}$$

et pour tout $t \in [d_1; d_1 + d_2]$ *:*

$$c_t\Big(\mathbb{R} \setminus \Big[x_{min} + s_1 + \frac{s_2}{d_2} \times (t - d_1) \ ; \ x_{min} + s_1 + l_1 - \frac{l_1 - l_2 - s_2}{d_2} \times (t - d_1)\Big]\Big) = \{\oslash\} \ ;$$

(iv) pour toute $c^{in} \in \mathcal{C}onfig^{in}$ *:*

$$c_{d_1+d_2}\Big(x_{min} + s_1 + s_2\Big) \neq \oslash \quad et \quad c_{d_1+d_2}\Big(x_{min} + s_1 + s_2 + l_2\Big) \neq \oslash \ ;$$

(v) $c_{d_1+d_2} \in \mathcal{C}onfig^{out}$.

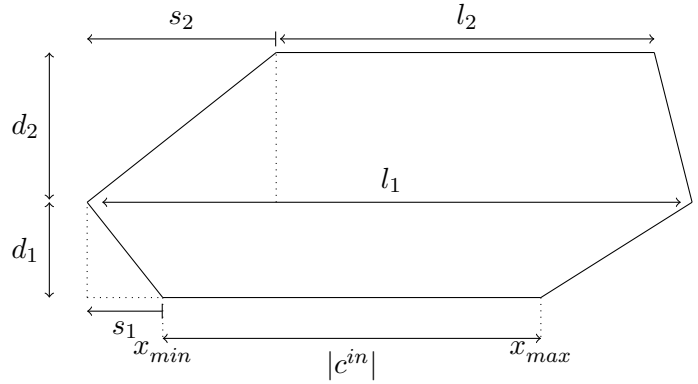


FIGURE 6.3 – Aire géométrique et paramètres d'un H-module.

La condition *(ii)* de la DÉF. 40 indique les ensembles de configuration de la machine $\mathfrak{A}$ qui seront considérées comme les entrées et sorties du H-module A.

La condition *(iii)* définit l'*aire géométrique de travail* $\mathcal{Z}$ du H-module sous la forme d'une sous-partie hexagonale de l'espace-temps telle que l'évolution entière du module à partir d'une configuration d'entrée quelconque reste confinée dans cet hexagone : en dehors de l'hexagone, aucun objet (signal ou collision) n'est présent. La base inférieure de cet hexagone correspond à la configuration d'entrée et la base supérieure correspond à la configuration de sortie. Ces deux bases sont donc parallèles puisqu'elles correspondent toutes deux à des configurations.

Les paramètres définissant cet hexagone sont illustrés dans FIG. 6.3. L'aire hexagonale de travail peut être vue comme deux trapèzes collés, chacun ayant respectivement comme paramètres d_1, s_1 et l_1 pour le trapèze du bas, et d_2, s_2 et l_2 pour celui du haut.

La condition « $d_1 + d_2 > 0$ » implique qu'au moins l'un de ces deux trapèzes est de hauteur non nulle et donc que l'hexagone ne peut pas être dégénéré, c'est-à-dire qu'on ne peut pas avoir un hexagone « plat » (les conditions de *(iii)* ne sont évidemment valables que pour $d_1 \neq 0$ et $d_2 \neq 0$, si l'une des deux fonctions est nulle, la condition correspondante disparaît). Les valeurs de la fonction $d_1 + d_2$ correspondent à la durée totale de fonctionnement du module, et c'est pour cette raison que l'on impose qu'au moins l'une des deux ne soit pas nulle, sinon il n'y aurait pas de dynamique du tout. Si l'une de ces deux fonctions est nulle, l'aire devient alors un trapèze, ce qui fera l'objet de la DÉF. 41.

La condition *(iv)* implique que la longueur de la base du haut est égale à la longueur de la configuration de sortie, cette même condition pour la base du bas découlant directement des définitions de x_{min} et x_{max}.

La dernière condition *(v)* impose à la configuration finale (celle qui correspond à la base supérieure de l'hexagone) d'être une configuration de sortie du module. Cette condition et la condition *(iv)* excluent donc la configuration vide (définie par $c(x) = \oslash$ pour tout $x \in \mathbb{R}$) comme configuration de sortie.

Remarque 22. On remarquera que les fonctions s_1 et s_2 sont toutes deux à valeurs dans $\mathbb{R}$: la définition permet donc d'autres formes que l'hexagone de la FIG. 6.3. Notamment, les hexagones ne sont pas nécessairement convexes : ils peuvent être, par exemple, en forme de sablier si l'on inverse le trapèze du bas avec celui du haut dans la FIG. 6.3, ou bien encore en forme de vase avec deux trapèzes devenant de plus en plus larges. La seule contrainte sur la forme géométrique de l'hexagone est que la base du bas et celle du haut soient toutes deux parallèles à l'axe spatial du diagramme espace-temps, car elles doivent toutes deux correspondre à des configurations.

La configuration vide ne doit appartenir ni à $\mathcal{C}onfig^{in}$, ni à $\mathcal{C}onfig^{out}$. Par contre, c^{in} et c^{out} peuvent être réduites à un point, bien que dans le cas où c^{in} est réduite à un point, il ne peut pas y avoir de dynamique en termes de collisions.

Notation. On rappelle que $\mathfrak{A}_d(c_0)$ désigne la configuration obtenue après une durée d par l'évolution de la machine $\mathfrak{A}$ exécutée sur la configuration c_0. On rappelle également que c_t désigne la configuration au temps t (la machine correspondante étant fixée), et que donc, à partir d'une configuration c_{t_0} à un temps t_0, la configuration c_t au temps $t > t_0$ est donnée par $c_t = \mathfrak{A}_{t-t_0}(c_{t_0})$. Généralement, $t_0 = 0$ et donc $c_t = \mathfrak{A}_t(c_0)$. Lorsqu'une machine est la machine sous-jacente d'une structure de module, on notera $\mathrm{A}(c^{in})$ la configuration obtenue à partir de la configuration c^{in} par l'évolution du module A ayant comme machine sous-jacente la machine $\mathfrak{A}$ *i.e.* $\mathrm{A}(c^{in}) = \mathfrak{A}_{d_1+d_2}(c^{in})$ (où d_1 et d_2 sont ceux de la définition du module A).

Soit c^{in} une configuration d'entrée pour le module A. En notant $c^{out} = \mathrm{A}(c^{in}) = \mathfrak{A}_{d_1+d_2}(c^{in})$, les conditions *(iv)* et *(v)* de la définition deviennent respectivement $|c^{out}| = l_2(c^{in})$ et $c^{out} \in \mathcal{C}onfig^{out}$ (la condition *(iii)* garantit la concordance des valeurs extrémales de c^{out} avec les extrémités de la base du haut de l'hexagone).

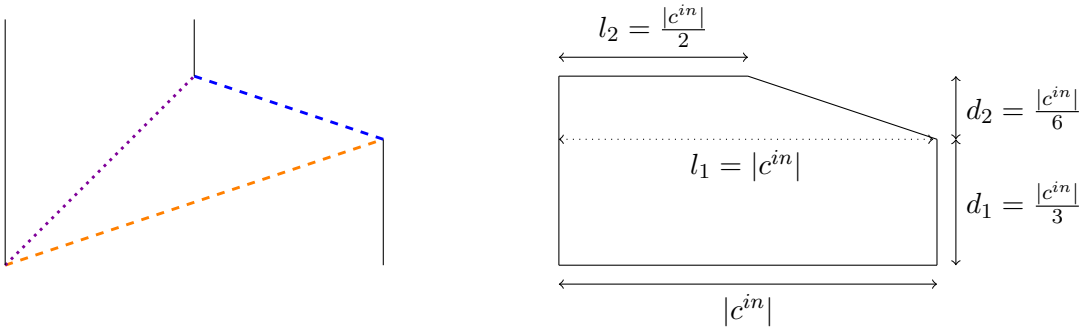
Exemple 4 : H-module de division par deux.
On formalise ici sous forme de H-module l'algorithme géométrique de division d'une grandeur réelle par deux donné en Sous-sec. 3.1.1. On reprend la machine $\mathfrak{M}_{div}$ comme machine sous-jacente, que l'on munie des ensembles d'entrées et sorties suivants :

- $\mathcal{C}onfig^{in} = \left\{ c = \{[\mathsf{w}, \overrightarrow{\mathsf{hi}}, \overrightarrow{\mathsf{lo}}]@0,\ \mathsf{w}@a\} \;\middle|\; a > 0 \right\} /_{\sim_{trans}}$;
- $\mathcal{C}onfig^{out} = \left\{ c = \{\mathsf{w}@0,\ \mathsf{w}@x\} \;\middle|\; x > 0 \right\} /_{\sim_{trans}}$.

Alors le quadruplet $(\mathfrak{M}_{div}, \mathcal{C}onfig^{in}, \mathcal{C}onfig^{out}, (d_1; d_2; s_1; s_2; l_1; l_2))$ avec :

$$\begin{cases} d_1 : c^{in} \mapsto \frac{|c^{in}|}{3} \\ d_2 : c^{in} \mapsto \frac{|c^{in}|}{6} \\ s_1 : c^{in} \mapsto 0 \\ s_2 : c^{in} \mapsto 0 \\ l_1 : c^{in} \mapsto |c^{in}| \\ l_2 : c^{in} \mapsto \frac{|c^{in}|}{2} \end{cases}$$

est un H-module. Un exemple de diagramme espace-temps de la machine $\mathfrak{M}_{div}$ est rappelé par Fig. 6.4(a) et les paramètres géométriques de l'aire de travail sont donnés dans Fig. 6.4(b).



(a) Diagramme.

(b) Polygone et paramètres associés (avec $s_1 = s_2 = 0$).

Figure 6.4 – Aire géométrique et paramètres du module de division par deux.

Les valeurs des fonctions-paramètres sont déduites des valeurs (positions et dates des collisions) calculées en Sous-sec. 3.1.1. La preuve que ces paramètres satisfont les conditions *(iii)*

et *(v)* de la définition d'un module découle directement de la preuve faite en SOUS-SEC. 3.1.1, qui fournit l'historique des collisions, et donc la configuration de la machine à chaque instant.

Il est également possible de définir un module correspondant au cas de diagrammes initiés par un signal $\overrightarrow{\mathsf{div}}$ venant de la gauche, comme évoqué en SOUS-SEC. 3.1.1.

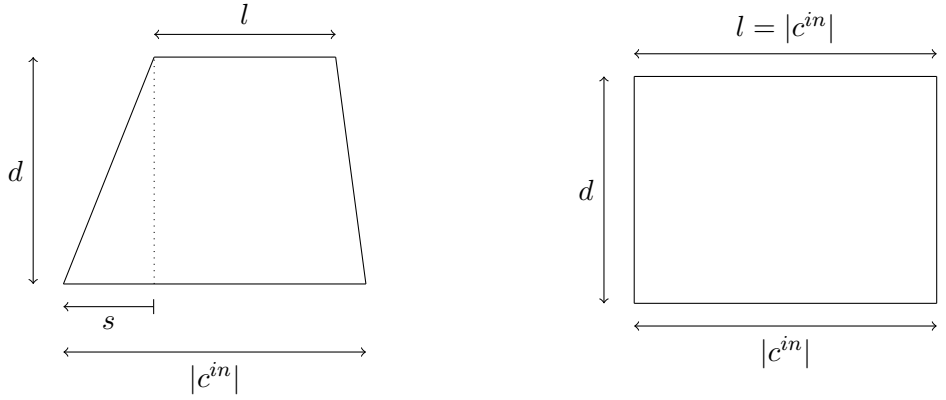
6.2.2 Notion de T-Module

Beaucoup de constructions et de modules restent confinés à des parties de l'espace-temps plus simples qu'une aire hexagonale comme celle utilisée dans le définition d'un H-module. En effet, une aire trapézoïdale ou rectangulaire est suffisante pour beaucoup de constructions et est plus facile à manipuler qu'un hexagone, notamment parce qu'elle sera toujours convexe (alors que l'aire d'un H-module ne l'est pas forcément). Nous définissons ici de tels modules ne nécessitant qu'un trapèze ou qu'un rectangle comme aire de travail, et ces définitions correspondent à la définition d'un H-module pour lequel certains des paramètres géométriques sont dégénérés. Nous appelons ces modules des T-modules (pour un « module-trapèze ») et R-modules (pour un « module-rectangle »).

Un T-module est un H-module dont un seul des deux trapèzes est réellement utilisé. Comme les paramètres des deux trapèzes sont définis de la même façon, on peut considérer indifféremment l'un des deux trapèzes comme étant celui qui est significatif et l'autre comme étant un trapèze dégénéré (de hauteur nulle). Dans la définition suivante, nous définissons l'aire du travail d'un T-module comme étant le trapèze inférieur de l'aire de travail d'un H-module:

DÉFINITION 41 (T-module)
Un T-module A *est un* H*-module* $(\mathfrak{A}, \mathcal{C}onfig^{in}, \mathcal{C}onfig^{out}, (d, 0, s, 0, l, l))$.

On note alors A $= (\mathfrak{A}, \mathcal{C}onfig^{in}, \mathcal{C}onfig^{out}, (d, s, l))$. Dans cette définition, les conditions $d_2 = 0$, $s_2 = 0$ et $l_2 = l_1$ doivent être comprises au sens de l'égalité de fonctions : la fonction d_2 est identiquement nulle (on a pour toute $c^{in} \in \mathcal{C}onfig^{in}$, $d_2(c^{in}) = 0$) et pour le cas de l_1 et l_2, on a pour toute $c^{in} \in \mathcal{C}onfig^{in}$, $l_2(c^{in}) = l_1(c^{in})$. L'aire de travail d'un T-module, ainsi que les paramètres associés, est illustrée par la FIG. 6.5(a).



(a) Aire de travail d'un T-module.

(b) Aire de travail d'un R-module.

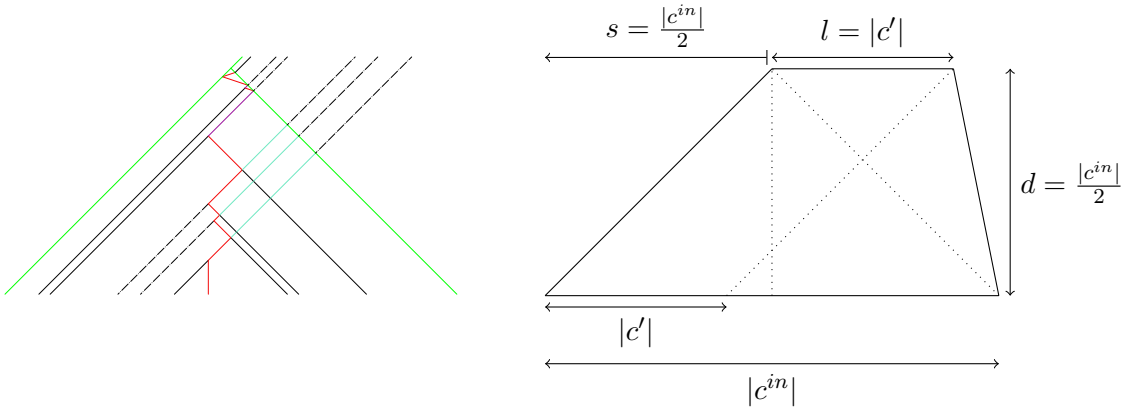
FIGURE 6.5 – Aires géométriques et paramètres d'un T-module et d'un R-module.

Pour un T-module, la condition *(iii)* de DÉF. 40 devient pour toute $c^{in} \in \mathcal{C}onfig^{in}$ et pour tout $t \in [0; d]$:

$$c_t\Big(\mathbb{R} \setminus \left[x_{min} + \frac{s}{d} \times t \ ; \ x_{max} - \frac{x_{max} - x_{min} - s - l}{d} \times t\right]\Big) = \{\oslash\} \tag{6.1}$$

Notons que la base du haut du trapèze d'un T-module peut être plus large que celle du bas : c'est le cas s'il existe c^{in} telle que $l(c^{in}) > |c^{in}|$. Le module utilisera alors de plus en plus d'espace pendant son évolution, et on parlera dans ce cas de *module expansif*. Sinon, le module sera dit *non-expansif* : on aura alors $l(c^{in}) \leq |c^{in}|$ pour toute c^{in}. Un module sera dit *contractant* si cette dernière inégalité est stricte.

Exemple 5 : module d'addition binaire.
On donne ici les paramètres d'un module d'addition binaire, dont la machine, les configurations d'entrée et sortie sont celles déjà données dans la Sous-sec. 3.1.3. Ce module fournit un exemple de module où les paramètres ne dépendent pas uniquement de la taille de la configuration d'entrée c^{in}. Dans le cas présent, le paramètre l correspondant à la taille de la configuration de sortie, est donné par la taille d'une sous-configuration c' de c^{in}. Comme c' est une sous-configuration stricte de c^{in} et que $l = |c'|$, on en déduit que ce module est contractant. L'étude de cette machine réalisée en Sous-sec. 3.1.3 nous fournit directement les expressions des paramètres en fonction de la configuration d'entrée, qui figurent sur la Fig. 6.6.



(a) Diagramme. (b) Trapèze et paramètres associés.

Figure 6.6 – Aire géométrique et paramètres du module d'addition binaire.

6.2.3 Notion de R-Module

De même qu'un T-module est un H-module avec certains paramètres dégénérés, un R-module est défini comme étant un T-module ayant également certains paramètres dégénérés, de telle sorte que son aire de travail sera réduite à un simple rectangle, précédemment illustré par la Fig. 6.5(b) :

Définition 42 (R-module)
Un R-module A *est un* T*-module* $(\mathfrak{A}, \mathcal{C}onfig^{in}, \mathcal{C}onfig^{out}, (d, 0, l))$ *avec* $l = c^{in} \mapsto |c^{in}|$.

On note alors simplement $\mathrm{A} = (\mathfrak{A}, \mathcal{C}onfig^{in}, \mathcal{C}onfig^{out}, d)$. La condition $l = |\cdot|$ signifie que pour tout c^{in} on a $l(c^{in}) = |c^{in}|$: la longueur de la base supérieure du trapèze est directement donnée par la taille de la configuration d'entrée et est donc la même que celle de la base inférieure (en particulier, un R-module est non-expansif). Avec la condition $s = 0$, on en déduit qu'il s'agit d'un rectangle de hauteur d et de largeur $|c^{in}|$.

Dans le cas d'un R-module, étant donné que $s = 0$ et que $l(c^{in}) = |c^{in}| = x_{max} - x_{min}$, la condition *(6.1)* qui exprime que tous les signaux et collisions restent confinés à l'intérieur de

l'aire géométrique s'écrit alors :

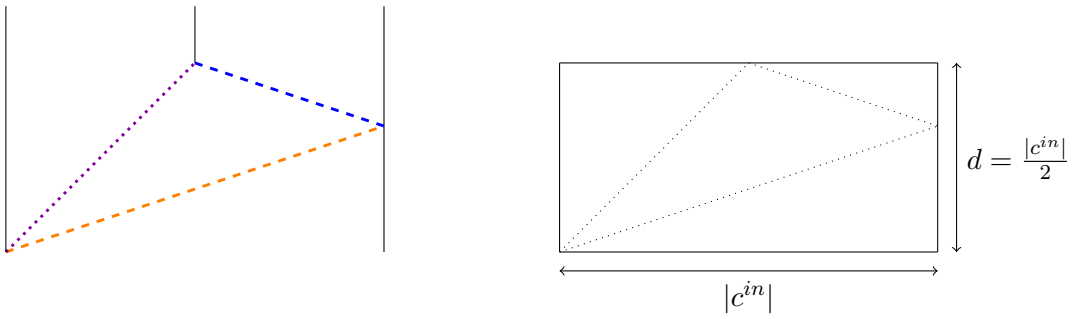
$$\forall t \in [0;d],\quad \forall c^{in} \in \mathcal{C}onfig^{in},\quad c_t\Big(\mathbb{R} \setminus [x_{min}\ ;\ x_{max}]\Big) = \{\oslash\}\ . \tag{6.2}$$

On retrouve bien le fait qu'à tout instant t, l'espace utilisé est le segment $[x_{min};x_{max}]$, et que la portion d'espace-temps correspondant à l'aire de travail est donnée par le rectangle $[x_{min};x_{max}] \times [0;d] \subseteq \mathbb{R} \times \mathbb{R}^+$.

Exemple 6 : module de calcul du milieu.
Nous reprenons ici l'exemple du calcul du milieu de la section Sec. 3.1. En considérant comme machine sous-jacente la machine $\mathfrak{M}_{mid}$ et en prenant comme ensembles d'entrées et sorties :

- $\mathcal{C}onfig^{in} = \Big\{c = \{[\mathsf{w}, \overrightarrow{\mathsf{hi}}, \overrightarrow{\mathsf{lo}}]@a,\ \mathsf{w}@b\} \ \Big|\ b > a\Big\}$ et
- $\mathcal{C}onfig^{out} = \big\{c = \{\mathsf{w}@a,\ \mathsf{w}@\frac{b-a}{2},\ \mathsf{w}@b\} \ \big|\ b > a\big\}$,

on obtient un R-module correspondant au calcul du milieu. On notera plusieurs différences avec l'Ex. 4 : d'abord les machines sous-jacentes, bien que presque identiques, diffèrent d'une règle de collisions. Ensuite, les aires géométriques sont différentes : il s'agit d'un pentagone dans le cas de l'Ex. 4, alors qu'ici c'est un rectangle, comme illustré par la Fig. 6.7. Et surtout, les ensembles de configurations d'entrée et de sortie ne sont pas les mêmes : dans le cas de la division par deux, les configurations requises sont celles qui sont conformes au schéma d'entrée défini à translations près, alors que celles requises pour le calcul du milieu sont les configurations conformes au même schéma mais en conservant les positions exactes. Ainsi, la valeur significative d'une configuration d'entrée pour la division par 2 est la distance séparant les deux signaux w, indépendamment de leurs positions et une configuration d'entrée pour le calcul du milieu nécessite les positions exactes de ces deux signaux.



(a) Diagramme. (b) Rectangle associé.

Figure 6.7 – Aire géométrique et paramètres du module de calcul du milieu.

Un autre cas particulier de module, que nous ne détaillons pas ici, est le cas des *modules triangulaires* : ce sont des T-modules dont toutes les configurations de sortie sont réduites à un point. Dans ce cas, l'aire géométrique correspondant est un triangle dont les caractéristiques sont données par les deux paramètres d et s.

6.2.4 Interprétations de modules

Nous avons défini un module comme une machine à signaux munie de deux ensembles de configurations : un pour les configurations que nous considérerons comme les entrées et un autre pour celles que nous considérerons comme les sorties du module.

Nous allons formaliser ici les notions de codage/décodage de configurations déjà discutées en Sous-sec. 3.2.4, afin de définir la notion de *module calculant une fonction*.

Définition 43 (Interprétation d'un module)
Une interprétation $\mathcal{I}$ *est un triplet* $\mathcal{I} = (f, R^{in}, R^{out})$ *tel que :*

(i) f *est une fonction définie de* $\mathcal{D}om(f)$ *dans* $\mathcal{V}al(f)$ *;*

(ii) R^{in} *est une fonction surjective de décodage définie d'un ensemble* E *dans* $\mathcal{D}om(f)$ *;*

(iii) R^{out} *est une fonction surjective de décodage définie d'un ensemble* F *dans* $\mathcal{V}al(f)$.

On dit que $\mathcal{I} = (f, R^{in}, R^{out})$ *est une* interprétation du module $\mathrm{A} = (\mathfrak{A}, \mathcal{C}onfig^{in}_{\mathrm{A}}, \mathcal{C}onfig^{out}_{\mathrm{A}}, \mathcal{Z}_{\mathrm{A}})$ *si :*

(i) $R^{in} : E \to \mathcal{D}om(f)$ *est définie sur* $E = \mathcal{C}onfig^{in}_{A}$ *;*

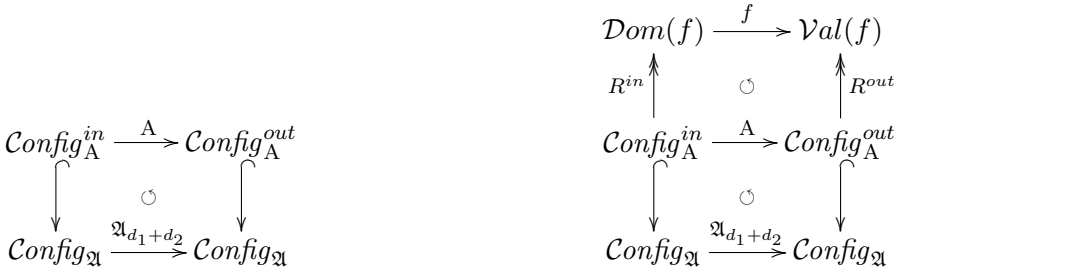
(ii) $R^{out} : F \to \mathcal{V}al(f)$ *est définie sur* $F = \mathcal{C}onfig^{out}_{A}$.

On note $\mathrm{A}^{\mathcal{I}}$ le module A muni de l'interprétation $\mathcal{I}$ et on parlera de *module interprété*. Dans ce cas, on appelle *domaine de* $A^{\mathcal{I}}$, noté $\mathcal{D}om(\mathrm{A})$ (resp. *l'ensemble des valeurs de* $A^{\mathcal{I}}$, noté $\mathcal{V}al(\mathrm{A})$) l'ensemble $\mathcal{D}om(f)$ (resp. l'ensemble $\mathcal{V}al(f)$).

Définition 44 (Fonction calculée par un module)
On dit que la fonction $f : E \to F$ *est* calculée par le module $\mathrm{A} = (\mathfrak{A}, \mathcal{C}onfig^{in}_{\mathrm{A}}, \mathcal{C}onfig^{out}_{\mathrm{A}}, \mathcal{Z}_{\mathrm{A}})$ *s'il existe une interprétation* $\mathcal{I} = (f, R^{in}, R^{out})$ *tel que le diagramme associé au module* $A^{\mathcal{I}}$ *est* commutatif, *c'est-à-dire si :*

$$\forall c^{in} \in \mathcal{C}onfig^{in}_{A} \quad R^{out}(A(c^{in})) = f(R^{in}(c^{in})) \ .$$

La Figure 6.8 présente les diagrammes respectivement associés à un module A et à ce même module interprété par $\mathcal{I} = (f, R^{in}, R^{out})$.



(a) Diagramme du module. (b) Diagramme du module interprété par $\mathcal{I} = (f, R^{in}, R^{out})$.

Figure 6.8 – Diagrammes commutatifs associés au module A.

Remarque 23. L'interprétation d'un module dépend non seulement des fonctions de décodage des entrées et des sorties, mais également des ensembles de configurations choisis comme ensembles d'entrées et de sortie. Par exemple, en prenant pour chaque ensemble de sorties $\mathcal{C}onfig^{out} = \{\mathsf{w}@x,\ \mathsf{w}@y,\ \mathsf{w}@z\}$, le module ayant comme machine sous-jacente la machine $\mathfrak{M}_{mid}$ peut être interprété afin de :

- calculer le milieu des segments $[a; b]$ lorsque l'ensemble des entrées est $\mathcal{C}onfig^{in} = \left\{ c = \{[\mathsf{w}, \overrightarrow{\mathsf{hi}}, \overrightarrow{\mathsf{lo}}]@a,\ \mathsf{w}@b\} \;\middle|\; b > a \right\}$;
- calculer les valeurs $x/2$ lorsque l'ensemble des entrées est $\mathcal{C}onfig^{in} = \left\{ c = \{[\mathsf{w}, \overrightarrow{\mathsf{hi}}, \overrightarrow{\mathsf{lo}}]@0,\ \mathsf{w}@x\} \;\middle|\; x > 0 \right\} /_{\sim_{trans}}$;
- générer un nouveau signal vertical entre deux signaux verticaux de positions 0 et x lorsque l'ensemble des entrées est $\mathcal{C}onfig^{in} = \left\{ c = \{[\mathsf{w}, \overrightarrow{\mathsf{hi}}, \overrightarrow{\mathsf{lo}}]@0,\ \mathsf{w}@x\} \;\middle|\; x > 0 \right\} /_{\sim_{ord}}$.

Exemple 7 : calcul de l'addition binaire.
On reprend l'exemple de module d'addition binaire de l'Ex. 5. La machine sous-jacente de ce

module a été conçue pour calculer l'addition de deux entiers codés en binaire. Formalisons le fait que ce module calcule bien l'addition. Pour cela, on le munit de l'interprétation suivante :

- $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ est l'addition entière usuelle : $f(n,m) = n + m$;
- $R^{in} : \mathcal{C}onfig^{in} \to \mathbb{N} \times \mathbb{N}$ telle que $\forall\ (n,m) \in \mathbb{N} \times \mathbb{N}$, $R^{in}\left(\left\{\ [\overrightarrow{bin}(n)],\ +^R,\ [\overleftarrow{bin}(m)]\ \middle|\ n,m \in \mathbb{N}\ \right\}\right) = (n,m)$;
- $R^{out} : \mathcal{C}onfig^{out} \to \mathbb{N}$ telle que $\forall\ n \in \mathbb{N}$, $R^{out}\left(\left\{\ [\overrightarrow{bin}(n)]\ \right\}\right) = n$.

On utilise la preuve faite en SOUS-SEC. 3.1.3 que pour une configuration d'entrée $c^{n,m}$ codant les entiers n et m, l'évolution de la machine produit finalement un faisceau de sortie codant en binaire l'entier $n+m$. Ce résultat peut maintenant être reformulé par $\mathrm{A}(c^{n,m}) = c^{n+m}$. On en déduit que :

$$\begin{aligned} \forall c^{n,m} \in \mathcal{C}onfig^{in} \quad R^{out}(\mathrm{A}(c^{n,m})) &= n+m \\ &= f(n+m) \\ &= f(R^{in}(c^{n,m}))\ . \end{aligned}$$

D'après la DÉF. 44, le module calcule donc bien la fonction $f(n,m) = n+m$.

6.3 Compositions de modules

Nous définissons dans cette section des opérateurs de construction de modules à partir de modules donnés. L'objectif est de concevoir des modules plus complexes en agrégeant, suivant des schémas particuliers, des modules plus simples dont nous connaissons et maîtrisons le comportement.

Nous nous contentons de définir ces opérateurs pour certaines classes de modules, essentiellement les T-modules non-expansifs et les R-modules. Ces classes sont en effet largement suffisantes pour concevoir la plupart des algorithmes géométriques, notamment ceux qui sont présentés et étudiés dans ce manuscrit.

6.3.1 Composition parallèle de modules

Nous définissons dans un premier temps un opérateur de composition de modules qui est essentiel à la programmation sur machines à signaux : il s'agit de la composition parallèle de modules. En effet, de par l'origine (les AC), le parallélisme est une des caractéristiques importantes des machines à signaux en tant que modèle de calcul.

Nous proposons ici une modélisation d'exécution parallèle de deux modules synchrones (c'est-à-dire avec la même durée de fonctionnement). Cet opérateur de *composition parallèle synchrone* nous permettra ensuite d'implémenter certains calculs distribués par le biais des signaux. La FIGURE 6.9 montre le principe de cette composition, formalisé par le définition suivante :

DÉFINITION/PROPOSITION 45 (Composition parallèle synchrone de modules)
Soient $A=(\mathfrak{A}, \mathcal{C}onfig_A^{in}, \mathcal{C}onfig_A^{out}, \mathcal{Z}_A)$ *et* $B=(\mathfrak{B}, \mathcal{C}onfig_B^{in}, \mathcal{C}onfig_B^{out}, \mathcal{Z}_B)$ *des* T*-modules non expansifs tels que l'union simple des machines* $\mathfrak{A}$ *et* $\mathfrak{B}$ *est définie. On suppose que les fonctions-paramètres de* $\mathcal{Z}_A$ *et* $\mathcal{Z}_B$ *vérifient* $\frac{|c|-s_A(c)-l_A(c)}{d_A(c)} \leq \frac{s_b(c')}{d_B(c')}$ *pour toutes configurations* $c \in \mathcal{C}onfig_A^{in}$ *et* $c' \in \mathcal{C}onfig_B^{in}$*. On définit le 4-uplet* $M=(\mathfrak{M}, \mathcal{C}onfig_M^{in}, \mathcal{C}onfig_M^{out}, \mathcal{Z}_M)$ *par :*

(i) $\mathfrak{M} = \mathfrak{A} \cup \mathfrak{B}$ *est la machine à signaux définie par l'union simple des machines* $\mathfrak{A}$ *et* $\mathfrak{B}$ *;*

(ii) $\mathcal{C}onfig_M^{in} = \left\{c \frown^{\varepsilon} c' \ \middle|\ c \in \mathcal{C}onfig_A^{in},\ c' \in \mathcal{C}onfig_B^{in} \text{ telles que } d_A(c) = d_B(c')\right\}$ *;*

(iii) $\mathcal{C}onfig_M^{out} = \mathcal{C}onfig_A^{out} \frown^{\varepsilon} \mathcal{C}onfig_B^{out}$ *;*

(iv) $\mathcal{Z}_M$ *est défini par le triplet* (d,s,l) *tel que :*

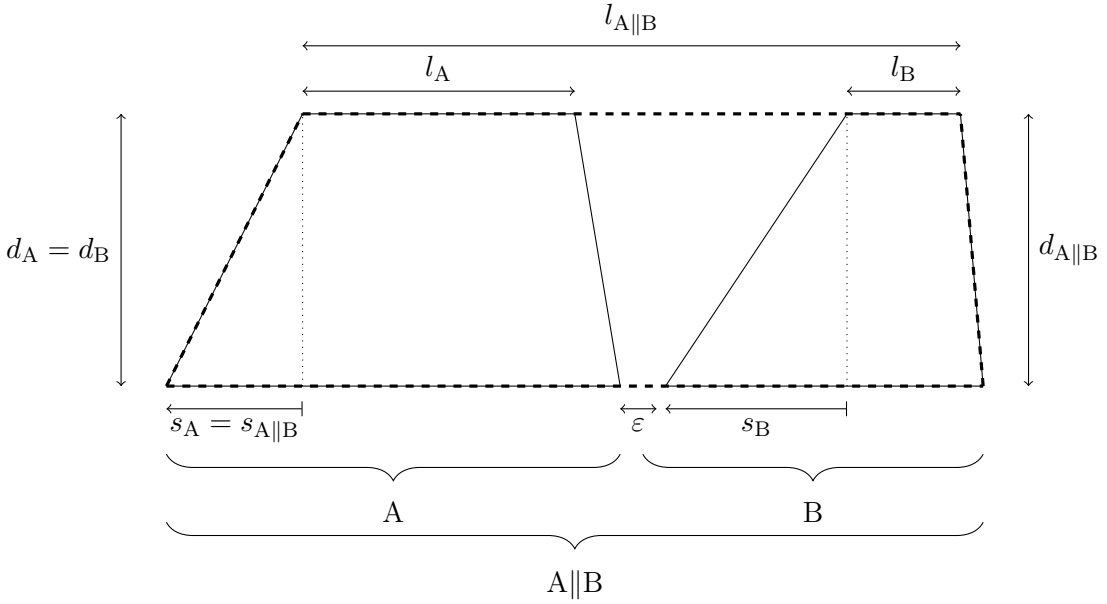


FIGURE 6.9 – Composition parallèle synchrone des modules A et B.

- $d : \mathcal{C}onfig_M^{in} \to \mathbb{R}^{+*}$ *est donné par* $d(c^{in}) = d_A(c)$ *;*
- $s : \mathcal{C}onfig_M^{in} \to \mathbb{R}$ *est donné par* $s(c^{in}) = s_A(c)$ *;*
- $l : \mathcal{C}onfig_M^{in} \to \mathbb{R}^{+}$ *est donné par* $l(c^{in}) = |c| - s_A + \varepsilon + s_B + l_B(c)$,
  où $c^{in} = c^\frown{}^\varepsilon c' \in \mathcal{C}onfig_A^{in} \frown^\varepsilon \mathcal{C}onfig_B^{in}$ *est la concaténation des configurations* c *et* c' *à distance* ε*.*

Alors le 4-uplet $(\mathfrak{M}, \mathcal{C}onfig_M^{in}, \mathcal{C}onfig_M^{out}, \mathcal{Z}_M)$ *est un* T*-module, appelé* composition parallèle synchrone*. On note ce module par* $A\|B$*.*

La FIGURE 6.9 illustre les paramètres géométriques obtenus par ce type de composition. La condition $\frac{|c|-s_A(c)-l_A(c)}{d_A(c)} \leq \frac{s_b(c')}{d_B(c')}$ impose que la pente de droite du module de gauche est inférieure à la pente de gauche du module de droite, ce qui garantit que les aires géométriques ne se superposent pas.

Démonstration. On montre que le 4-uplet ainsi défini est bien un T-module. $\mathfrak{M}$ est l'union simple des machines $\mathfrak{A}$ et $\mathfrak{B}$ et est par définition une machine à signaux (voir DÉF. 39). Les ensembles $\mathcal{C}onfig_{\mathrm{M}}^{in}$ et $\mathcal{C}onfig_{\mathrm{M}}^{out}$ sont bien des configurations pour la machine $\mathfrak{M}$, toujours par définition de la machine $\mathfrak{M}$ qui contient tous les méta-signaux et règles de collisions des machines $\mathfrak{A}$ et $\mathfrak{B}$.

Montrons que le triplet (d, s, l) vérifie la condition *(6.1)*. Pour tout $t \in [0; d]$ et pour toute $c^{in} \in \mathcal{C}onfig_{\mathrm{M}}^{in}$, que l'on écrit sous la forme $c^{in} = c^\frown{}^\varepsilon c'$, on a :

$$
\begin{aligned}
& c_t\Big(\mathbb{R} \setminus \Big[x_{min} + \tfrac{s}{d} \times t \ ; \ x_{max} - \tfrac{x_{max} - x_{min} - s - l}{d} \times t\Big]\Big) \\
&= c_t\Big(\mathbb{R} \setminus \Big[x_{min} + \tfrac{s_A}{d_A} \times t \ ; \ x_{max} - \tfrac{|c^{in}| - s_A(c) - (|c| - s_A(c) + \varepsilon + s_B(c) + l_B(c))}{d} \times t\Big]\Big) \\
&= c_t\Big(\mathbb{R} \setminus \Big[x^c_{min} + \tfrac{s_A}{d_A} \times t \ ; \ x^{c'}_{max} - \tfrac{|c'| - l_B(c') - s_B}{d} \times t\Big]\Big) \text{ car } c^{in} = c^\frown{}^\varepsilon c' \text{ et } |c^{in}| = |c| + \varepsilon + |c'| \\
&= \{\oslash\} \ .
\end{aligned}
$$

□

Cette définition donne un cadre formel à l'idée « d'exécuter deux calculs géométriques côte à côte »: ici, les exécutions des deux modules sont indépendantes, car leur deux aires de travail le sont. On obtient donc des configurations de sortie qui sont des configurations que l'on peut obtenir

en exécutant les modules dans deux diagrammes différents puis en prenant la concaténation des deux configurations de sortie obtenues. La définition DÉF. 45 permet de considérer cette démarche dans un seul et unique module, qui regroupe également les deux machines en une seule. Une configuration c_t à un instant $t \in [0; d]$ est donnée par une concaténation d'une configuration $\mathrm{A}_t(c)$ et d'une configuration $\mathrm{B}_t(c')$, pour les configurations c et c' correspondantes.

Remarque 24. Un cas particulier de composition synchrone est celui de durées d_A et d_B ne dépendant que de la taille des configurations d'entrées, avec de plus une dépendance linéaire : il existe $\alpha_1, \alpha_2 \in \mathbb{R}^{+*}$ tels que $d_\mathrm{A}(c^{in}) = \alpha_1 \times |c^{in}|$ et $d_\mathrm{B}(c^{in}) = \alpha_2 \times |c^{in}|$. Dans ce cas, l'ensemble de configurations d'entrées pour le module composé est donné par $\mathcal{C}onfig^{in}_{\mathrm{A}\|\mathrm{B}} = \left\{ c^\frown c' \;\middle|\; c \in \mathcal{C}onfig^{in}_\mathrm{A}, c' \in \mathcal{C}onfig^{in}_\mathrm{B} \text{ et } |c'| = \frac{\alpha_1}{\alpha_2} \times |c| \right\}$.

Interprétation d'une composition parallèle. Nous montrons ici que lorsque les deux modules initiaux sont munis d'une interprétation, leur composition parallèle est également munie d'une interprétation qui correspond au produit des deux interprétations initiales :

PROPOSITION 26 (Interprétation d'une composition parallèle synchrone)
Si les A et B sont respectivement munis des interprétations $\mathcal{I} = (f, R^{in}_A, R^{out}_A)$ *et* $\mathcal{J} = (g, R^{in}_B, R^{out}_B)$ *telles que A calcule f et B calcule g et soit* $(\mathfrak{M}, \mathcal{C}onfig^{in}_M, \mathcal{C}onfig^{out}_M, \mathcal{Z}_M) = A\|B$.

Soit $\mathcal{K}$ *l'interprétation* (h, R^{in}_M, R^{out}_M) *telle que :*

(i) h *est une fonction définie de* $\mathcal{D}om(f) \times \mathcal{D}om(g)$ *dans* $\mathcal{V}al(f) \times \mathcal{V}al(g)$ *par* $h(x, y) = (f(x), g(y))$ *;*

(ii) $R^{in}_M : \mathcal{C}onfig^{in}_M \to \mathcal{D}om(f) \times \mathcal{D}om(g)$ *est définie telle que pour tout* $c^{in} \in \mathcal{C}onfig^{in}_M$*, avec* c *et* c' *les configurations telles que* $c^{in} = c^\frown{}^\varepsilon c'$*, on a* $R^{in}_M(c^{in}) = (R^{in}_A(c), R^{in}_B(c'))$ *;*

(iii) $R^{out}_M : \mathcal{C}onfig^{out}_M \to \mathcal{V}al(f) \times \mathcal{V}al(g)$ *est définie telle que pour tout* $c^{out} \in \mathcal{C}onfig^{out}_M$*, avec* c *et* c' *les configurations telles que* $c^{out} = c^\frown{}^\varepsilon c'$*, on a* $R^{out}_M(c^{out}) = (R^{out}_A(c), R^{out}_B(c'))$.

Alors $\mathcal{K}$ *est une interprétation du module* $A\|B$ *telle que* $A\|B$ *calcule la fonction* h.

En d'autres termes : si les diagrammes des modules interprétés $\mathrm{A}^\mathcal{I}$ et $\mathrm{B}^\mathcal{J}$ sont commutatifs, alors le diagramme du module interprété $(\mathrm{A}\|\mathrm{B})^\mathcal{K}$ sera aussi commutatif (*c.f.* FIG. 6.10, dans lequel les fonctions $proj_i$ désignent les fonctions classiques de projections de la $i-ème$ composante). La condition *(iii)* suppose que l'on sait récupérer à partir de c^{out} les configurations c et c' appartenant respectivement aux ensembles de configurations de sortie des modules A et B. Pour cela, on utilise les paramètres (connus) l_A et l_B des modules A et B : les configurations c et c' se déduisent immédiatement de c^{out} que l'on restreint d'une part avec l_A pour obtenir c, et d'autre part avec l_B pour obtenir c'. On obtient ainsi de manière unique les configurations c et c' telles que $c^{out} = c^\frown{}^\varepsilon c'$. Notons que sans les paramètres l_A et l_B, l'unicité de cette écriture n'est pas garantie.

Démonstration. Montrons d'abord que le triplet $(h, R^{in}_\mathrm{M}, R^{out}_\mathrm{M})$ ainsi défini est une interprétation pour le module A‖B. Les fonctions de décodage sont bien des fonctions surjectives. En effet, la surjectivité de R^{in}_M (l'argument pour celle de R^{out}_M est exactement le même) provient directement de la surjectivité des fonctions R^{in}_A et R^{in}_B : pour tout $(x, y) \in \mathcal{D}om(f) \times \mathcal{D}om(g)$, il existe $c \in \mathcal{C}onfig^{in}_\mathrm{A}$ et $c' \in \mathcal{C}onfig^{in}_\mathrm{B}$ telles que $R^{in}_\mathrm{A}(c) = x$ et $R^{in}_\mathrm{B}(c) = y$. La concaténation $c \frown^\varepsilon c'$ est alors dans $\mathcal{C}onfig^{in}_\mathrm{M}$ et on a bien $R^{in}_\mathrm{M}(c^\frown{}^\varepsilon c') = (x, y)$.

Montrons maintenant que le module A‖B calcule bien la fonction h. Soit $c_0 = c^\frown{}^\varepsilon c' \in$

$\mathcal{C}onfig^{in}_{\mathrm{M}}$. On a :

$$\begin{aligned} h(R^{in}_{\mathrm{M}}(c_0)) &= h(R^{in}_{\mathrm{M}}(c \frown^{\varepsilon} c')) \\ &= h(R^{in}_{\mathrm{A}}(c), R^{in}_{\mathrm{B}}(c')) \\ &= (f(R^{in}_{\mathrm{A}}(c)), g(R^{in}_{\mathrm{B}}(c'))) \\ &= (R^{out}_{\mathrm{A}}(\mathrm{A}_d(c)), R^{out}_{\mathrm{B}}(\mathrm{B}_d(c')) \text{ par commutativité des diagrammes de A et B} \\ &= R^{out}_{\mathrm{M}}((\mathrm{A}\|\mathrm{B})_d(c_0)) \text{ car } (\mathrm{A}\|\mathrm{B})_d(c_0) = \mathrm{A}_d(c) \frown^{\varepsilon} \mathrm{B}_d(c') \text{ (car les aires sont disjointes)} \end{aligned}$$

donc le diagramme de $(\mathrm{A}\|\mathrm{B})^{\mathcal{K}}$ est commutatif et A$\|$B calcule bien la fonction h. □

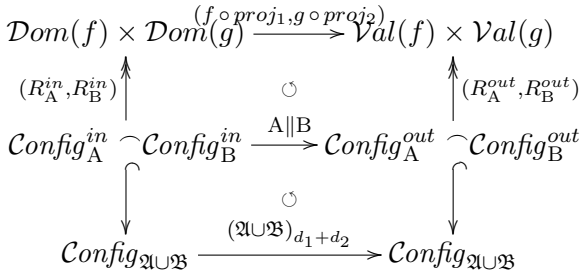


FIGURE 6.10 – Diagramme commutatif associé à la composition parallèle des modules A et B.

Exemple 8 : évaluation de deux formules booléennes.
On suppose que l'on dispose d'un module permettant d'évaluer une formule booléenne, représentée sous forme de faisceau. Un tel module sera détaillé dans les chapitres suivants. Ce module produit à partir d'un faisceau $[\overleftarrow{for}(\phi)]$ codant une formule booléenne ϕ et d'un signal stationnaire eval utilisé pour l'évaluation, un signal codant une valeur booléenne (il s'agit donc d'un module triangulaire). À l'aide d'une composition parallèle de deux tels modules, il est aisé de concevoir un module évaluant en parallèle deux formules booléennes distinctes : la FIG. 6.11 illustre une telle construction.

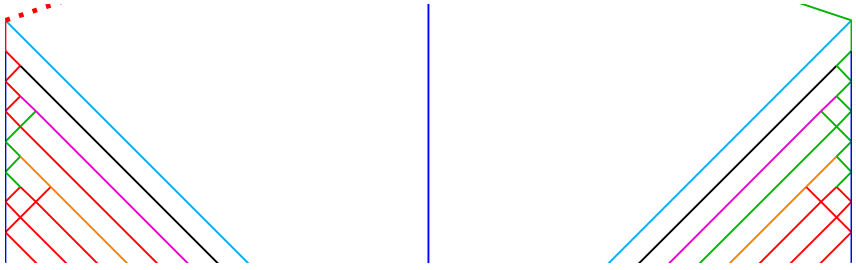


FIGURE 6.11 – Module d'évaluation de deux formules booléennes.

Dans cet exemple, les deux modules sont séparés par un signal stationnaire, qui ne joue pour le moment aucun rôle mais qui sera utilisé par la suite. Ce signal peut soit être inclus dans un des deux modules (celui de gauche par exemple, qui devient alors un R-module contrairement à celui de droite qui reste un module triangulaire), ou bien la construction complète peut être vue comme la composition parallèle de trois modules, le troisième module étant uniquement constitué par le seul signal stationnaire central.

Cas particulier : parallélisation d'un module. On s'intéresse ici à un cas particulier de composition parallèle synchrone : celle où les deux modules sont un seul et même module R-module A. Nous pouvons facilement obtenir un module pour effectuer n exécutions parallèles pour $n \geq 2$. Nous appelons ce cas particulier une *parallélisation du module A*, illustré par la FIG. 6.12.

DÉFINITION 46 (Parallélisation d'un module)
Soit $A=(\mathfrak{A}, \mathcal{C}onfig_A^{in}, \mathcal{C}onfig_A^{out}, \mathcal{Z}_A)$ *un T-module tel que* $\forall c \in \mathcal{C}onfig_A^{in}$, $|c|-s_A(c)-l_A(c) \leq s_A(c)$ *et tel que* $\forall c, c' \in \mathcal{C}onfig_A^{in}$ $|c| = |c'| \Rightarrow d(c) = d(c')$. *Soit* $n \in \mathbb{N}^*$. *On définit la* n-parallélisation de A, *notée* $\|^n A$, *par :* $\|^n A = A\|(A\|\dots(A\|A))$, *où le module A apparaît n fois.*

Les conditions énoncées ci-dessus correspondent à celles de la DÉF. 45 pour le cas particulier A=B. Notons que la relation de composition parallèle est associative : on a pour tous modules A, B et C : A$\|$(B$\|$C) = (A$\|$B)$\|$C. Il s'ensuit que l'itération d'un module A peut simplement se définir par : $\|^n$A =A$\|$A$\|\dots$A$\|$A.

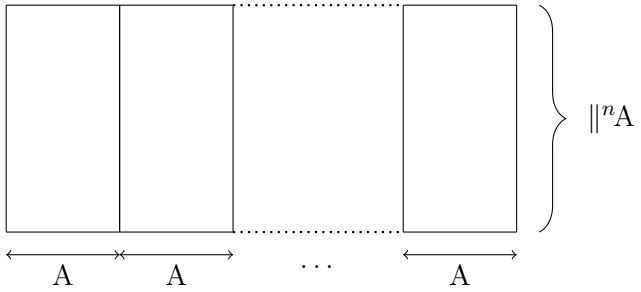


FIGURE 6.12 – Parallélisation d'un R-module A.

Exemple 9 : additions de couples d'entiers.
À partir du module d'addition binaire, on peut facilement concevoir un module effectuant en parallèle l'addition de deux couples d'entiers : il suffit d'effectuer la parallélisation du module d'addition binaire, comme illustré par la FIG. 6.13, dans laquelle deux mêmes modules d'addition binaire sont exécutés en parallèle. Les configurations d'entrée (resp. de sortie) sont la concaténation de deux configurations d'entrée (resp. de sortie) pour le module simple d'addition : quatre entiers sont donc représentés en entrée et deux en sortie. Le résultat est le faisceau binaire obtenu par la concaténation des deux faisceaux résultant des deux additions binaires respectives des deux premiers et des deux derniers entiers de l'entrée.

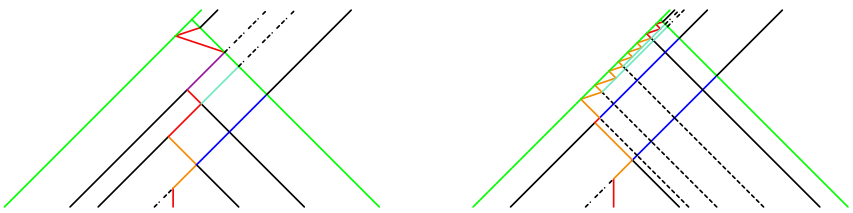


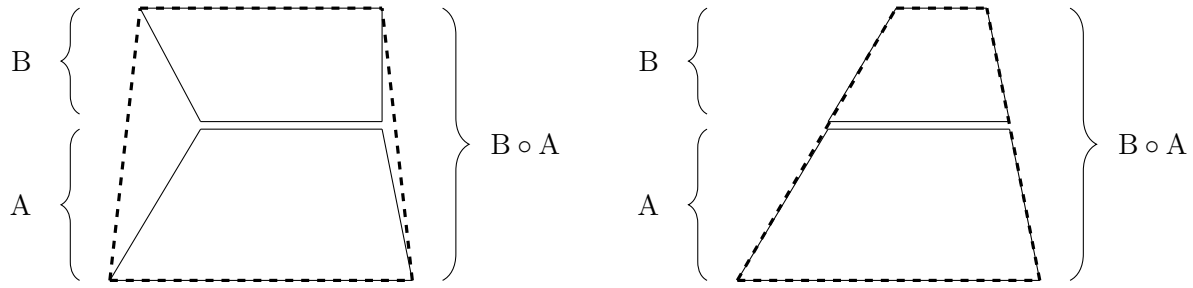
FIGURE 6.13 – Parallélisation de l'additionneur binaire envoyant le faisceau de sortie vers la droite. Exemple d'exécution sur l'entrée $((6,3),(2,17))$, produisant le couple $(9,19)$ en sortie.

6.3.2 Composition séquentielle de modules

Après avoir précédemment défini une manière parallèle de composer les modules, nous définissons une *composition séquentielle*. Une telle composition correspond à une exécution séquentielle

au sens intuitif de deux modules : les deux modules sont exécutés l'un après l'autre, de telle sorte que la sortie du premier module est l'entrée du second. Comme pour la composition parallèle, nous nous restreignons ici aux T-modules.

Les conditions de la définition suivante garantissent que les aires géométriques des deux modules seront du même type que celles de la FIG. 6.14(a) : le module B ne doit pas « se resserrer » plus rapidement que le module A. On peut facilement voir que ces conditions garantissent que l'aire du module résultant est bien un trapèze. Sans ces conditions, le module résultant posséderait une aire hexagonale dans le cas général. La FIGURE 6.14(b) illustre une composition séquentielle dans le cas particulier où les pentes des aires des deux modules correspondent.



(a) Cas de la DÉF. 47. (b) Cas particulier.

FIGURE 6.14 – Composition séquentielle des modules A et B.

DÉFINITION/PROPOSITION 47 (Composition séquentielle de modules)
Soient $A=(\mathfrak{A}, \mathcal{C}onfig_A^{in}, \mathcal{C}onfig_A^{out}, \mathcal{Z}_A)$ *et* $B=(\mathfrak{B}, \mathcal{C}onfig_B^{in}, \mathcal{C}onfig_B^{out}, \mathcal{Z}_B)$ *deux* T*-modules tels que l'union simple des machines* $\mathfrak{A}$ *et* $\mathfrak{B}$ *soit définie. On suppose que :*

- $\mathcal{C}onfig_A^{out} \subseteq \mathcal{C}onfig_B^{in}$;
- $\frac{s_B}{d_B} \leq \frac{s_A}{d_A}$ *et*
- $\frac{|c|-s_A(c)-l_A(c)}{d_A(c)} \leq \frac{|c|-s_B(c)-l_B(c)}{d_B(c)}$.

On définit le 4*-uplet* $(\mathfrak{M}, \mathcal{C}onfig_M^{in}, \mathcal{C}onfig_M^{out}, \mathcal{Z}_M)$ *par :*

(i) $\mathfrak{M} = \mathfrak{A} \cup \mathfrak{B}$ *est la machine à signaux définie par l'union simple des machines* $\mathfrak{A}$ *et* $\mathfrak{B}$;

(ii) $\mathcal{C}onfig_M^{in} \subseteq \mathcal{C}onfig_M$ *est l'ensemble* $\mathcal{C}onfig_A^{in}$;

(iii) $\mathcal{C}onfig_M^{out} \subseteq \mathcal{C}onfig_M$ *est l'ensemble* $\mathcal{C}onfig_B^{out}$;

(iv) $\mathcal{Z}_M$ *est défini par le triplet* (d, s, l) *tel que :*
- $d : \mathcal{C}onfig_M^{in} \to \mathbb{R}^{+*}$ *est donné par* $d(c^{in}) = d_A(c^{in}) + d_B(A_{d_A}(c^{in}))$;
- $s : \mathcal{C}onfig_M^{in} \to \mathbb{R}$ *est donné par* $s(c^{in}) = s_A(c^{in}) + s_B(A_{d_A}(c^{in}))$;
- $l : \mathcal{C}onfig_M^{in} \to \mathbb{R}^{+}$ *est donné par* $l(c^{in}) = l_B(A_{d_A}(c^{in}))$.

Alors le 4*-uplet* $(\mathfrak{M}, \mathcal{C}onfig_M^{in}, \mathcal{C}onfig_M^{out}, \mathcal{Z}_M)$ *est un* T*-module, appelé* composition séquentielle des modules A et B *. On le note alors* $B \circ A$.

Interprétation d'une composition séquentielle. Lorsque les modules composés séquentiellement sont munis d'interprétations, l'interprétation de la composition est fournie par la composition (classique) des fonctions correspondant aux interprétations des modules initiaux :

PROPOSITION 27 (Interprétation d'une composition séquentielle)
Si les A et B sont respectivement munis des interprétations $\mathcal{I}=(f, R_A^{in}, R_A^{out})$ *et* $\mathcal{J}=(g, R_B^{in}, R_B^{out})$ *telles que A calcule* f *et B calcule* g *et soit* $(\mathfrak{M}, \mathcal{C}onfig_M^{in}, \mathcal{C}onfig_M^{out}, \mathcal{Z}_M) = B \circ A$ *défini précédemment. On suppose que* $R_A^{out} = R_B^{in} \restriction_{\mathcal{C}onfig_A^{out}}$. *Soit* $\mathcal{K}$ *l'interprétation* (h, R_M^{in}, R_M^{out}) *telle que :*

(i) k *est une fonction définie de* $\mathcal{D}om(f)$ *dans* $\mathcal{V}al(g)$ *par* $h(x) = g(f(x))$ *i.e.* $h = g \circ f$;

(ii) $R_M^{in} : \mathcal{C}onfig_M^{in} \to \mathcal{D}om(f)$ *est définie par* $R_M^{in} = R_A^{in}(c)$ *pour tout* $c \in \mathcal{C}onfig_M^{in}$ *;*
(iii) $R_M^{out} : \mathcal{C}onfig_M^{out} \to \mathcal{V}al(g)$ *est définie par* $R_M^{out} = R_B^{out}(c)$ *pour tout* $c \in \mathcal{C}onfig_M^{out}$.
Alors $\mathcal{K}$ *est une interprétation du module* $B \circ A$ *telle que* $B \circ A$ *calcule la fonction* h.

En d'autres termes : si les diagrammes des modules interprétés $\mathrm{A}^{\mathcal{I}}$ et $\mathrm{B}^{\mathcal{J}}$ sont commutatifs et si la fonction de décodage des entrées de B coïncide avec la fonction de décodage des sorties de A, alors le diagramme du module interprété $(\mathrm{B} \circ \mathrm{A})^{\mathcal{K}}$ sera aussi commutatif, comme illustré par la FIG. 6.15.

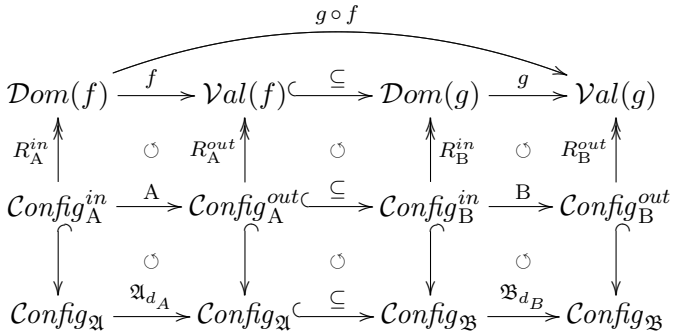


FIGURE 6.15 – Diagramme commutatif de la composition séquentielle des modules A et B.

Démonstration. Montrons d'abord que le triplet $(h, R_{\mathrm{M}}^{in}, R_{\mathrm{M}}^{out})$ ainsi défini est une interprétation pour le module B∘A, c'est-à-dire que les fonctions de décodage sont bien des fonctions surjectives. Comme $R_{\mathrm{M}}^{in} = R_{\mathrm{A}}^{in}$ et que R_{A}^{in} est une fonction surjective de décodage des entrées de A, R_{M}^{in} est également surjective. De même, la fonction R_{M}^{out} étant égale à R_{B}^{out}, on déduit immédiatement sa surjectivité de celle de la fonction de décodage des sorties de B. Notons que la condition $R_{\mathrm{A}}^{out} = R_{\mathrm{B}}^{in} \restriction_{\mathcal{C}onfig_{\mathrm{A}}^{out}}$ suppose implicitement que $\mathcal{C}onfig_{\mathrm{A}}^{out} \subseteq \mathcal{C}onfig_{\mathrm{B}}^{in}$, condition vérifiée par la définition du module B ∘ A.

Montrons maintenant que le module B ∘ A calcule effectivement la fonction h. Soit $c_0 \in \mathcal{C}onfig_{\mathrm{M}}^{in}$. On a :

$$\begin{aligned}
h(R_{\mathrm{M}}^{in}(c_0)) &= h(R_{\mathrm{A}}^{in}(c_0)) \\
&= g(f(R_{\mathrm{A}}^{in}(c_0))) \text{ car } h = g \circ f \\
&= g(R_{\mathrm{A}}^{out}(\mathrm{A}_{d_A}(c_0)) \text{ car A calcule } f \\
&= g(R_{\mathrm{B}}^{in}(\mathrm{A}_{d_A}(c_0))) \\
&= R_{\mathrm{B}}^{out}(\mathrm{B}_{d_B}(\mathrm{A}_{d_A}(c_0)))) \text{ car B calcule } g \text{ et } \mathrm{A}_{d_A}(c_0) \in \mathcal{C}onfig_{\mathrm{B}}^{in} \\
&= R_{\mathrm{M}}^{out}(\mathrm{M}_{d_A+d_B}(c_0)) \text{ par définition de la machine } \mathfrak{M} \\
&= R_{\mathrm{M}}^{out}(\mathrm{M}_{d_M}(c_0)) \ .
\end{aligned}$$

Donc le diagramme de $(\mathrm{B} \circ \mathrm{A})^{\mathcal{K}}$ est commutatif et B ∘ A calcule bien la fonction h. □

Remarque 25. La condition $\mathcal{C}onfig_{\mathrm{A}}^{out} \subseteq \mathcal{C}onfig_{\mathrm{B}}^{in}$ peut également être remplacée par la condition un peu plus générale de l'existence d'une injection $i : \mathcal{C}onfig_{\mathrm{A}}^{out} \to \mathcal{C}onfig_{\mathrm{B}}^{in}$. Cela suffit en effet pour démontrer la commutativité du diagramme composé.

Exemple 10 : disjonction des résultats de l'évaluation de deux formules booléennes. On reprend dans cet exemple le module de l'EX. 8 effectuant les évaluations de deux formules

booléennes, qui sera donc interprété par la fonction :

$$\begin{aligned}\mathcal{E}val_2 : \{\phi \mid \phi \text{ est une formule booléenne}\}^2 &\to \mathbb{B}^2\\ (\phi_1, \phi_2) &\mapsto (val(\phi_1), val(\phi_2))\end{aligned}$$

où $\mathbb{B} = \{\texttt{vrai}, \texttt{faux}\}$. On compose séquentiellement ce module avec un module effectuant la disjonction logique de deux booléens. On l'interprète par la fonction :

$$\begin{aligned}\mathcal{D}isj : \mathbb{B}^2 &\to \mathbb{B}\\ (b_1, b_2) &\mapsto b_1 \vee b_2\end{aligned}$$

La fonction f obtenue sera la composée de ces deux fonctions et on aura :

$$\begin{aligned}f : \{\phi \mid \phi \text{ est une formule booléenne}\}^2 &\to \mathbb{B}\\ (\phi_1, \phi_2) &\mapsto val(\phi_1) \vee val(\phi_2)\end{aligned}$$

Pour le module d'évaluation envoyant le résultat vers la droite, l'ensemble des configurations d'entrée est $\mathcal{C}onfig^{in} = \left\{c = \{\mathsf{eval}, [\overleftarrow{for}(\phi)]\} \;\middle|\; \phi \text{ est une formule booléenne}\right\}$ et celui des sorties est $\mathcal{C}onfig^{out} = \left\{c = \{\overrightarrow{\mathsf{bool}}\} \;\middle|\; \mathsf{bool} \in \mathbb{B}\right\}$, où eval est un méta-signal stationnaire permettant l'évaluation (en jouant un rôle de mur activateur), $[\overleftarrow{for}(\phi)]$ est un faisceau de signaux de vitesse -1 représentant une formule booléenne (cette représentation sera détaillée au CHAP. 7) et $\overrightarrow{\mathsf{bool}}$ correspond à un méta-signal de type booléen (soit un signal $\overrightarrow{\mathsf{vrai}}$, soit un signal $\overrightarrow{\mathsf{faux}}$). La FIGURE 6.16 illustre le module résultant de cette composition séquentielle (le premier module étant déjà lui-même une composition parallèle de modules). Dans cet exemple, les formules codées à gauche et à droite sont respectivement fausse et vraie, et le résultat de la disjonction des deux est représenté tout en haut du module par un signal de type `vrai`.

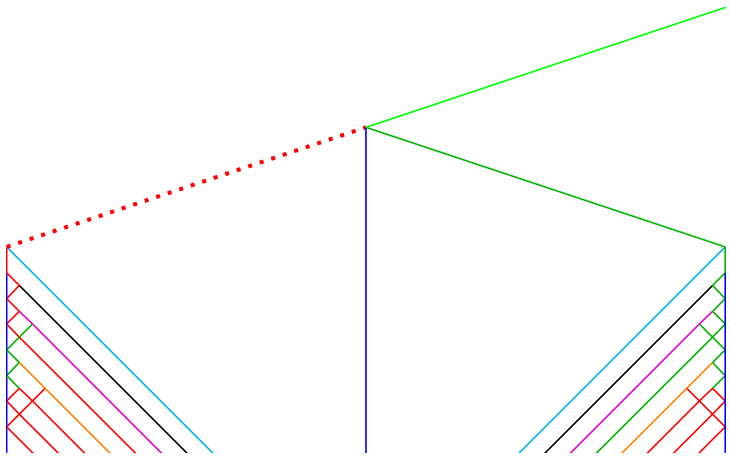


FIGURE 6.16 – Composition séquentielle d'un module évaluant en parallèle deux formules booléennes avec le module de calcul de la disjonction booléenne.

Exemple 11 : addition de quatre entiers.
De même, il est possible de concevoir un module effectuant l'addition binaire de quatre entiers. Pour cela, on reprend le module de l'additionneur binaire. On le compose d'abord de manière parallèle, comme dans l'EX. 9, de telle sorte que le module de gauche envoie le résultat de l'addition vers la gauche. On obtient alors une configuration conforme aux entrées du module simple d'addition binaire. On peut donc ensuite composer séquentiellement avec une addition binaire simple. Le résultat de ces compositions est donné en FIG. 6.17, le faisceau résultant correspondant à l'addition des quatre entiers donnés en entrées.

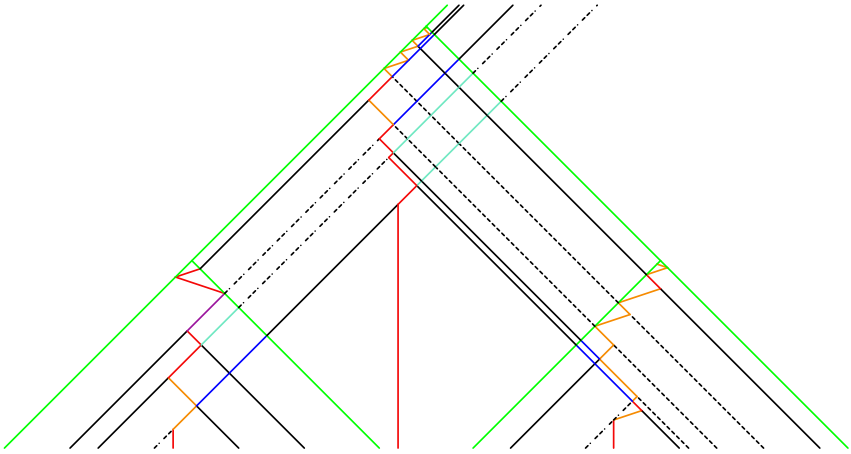
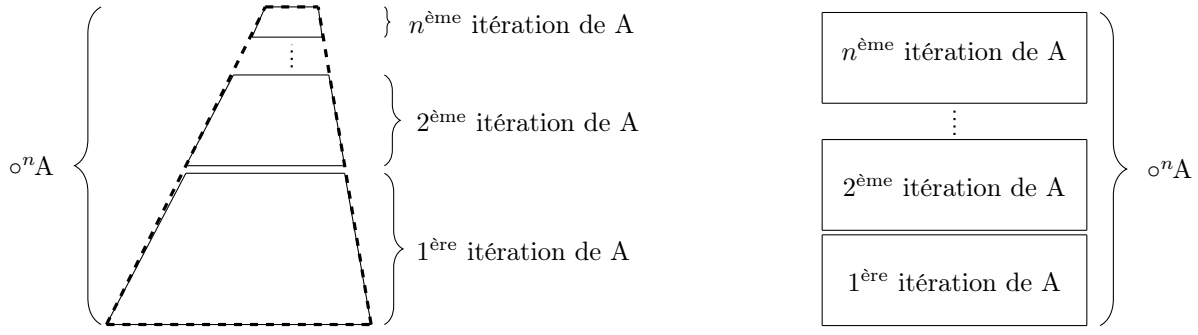


FIGURE 6.17 – Composition parallèle de deux additionneurs binaires, puis composition séquentielle avec un autre additionneur binaire. On obtient à partir de l'entrée $((6,3),(2,17))$ d'abord le couple $(9,19)$, puis l'entier 28.

Cas particulier : itération d'un module. De même que nous avons défini la parallélisation d'un module à partir de la composition parallèle, il est possible de définir l'*itération* d'un module à partir de la composition séquentielle :

DÉFINITION 48 (Itération d'un module)
Soit $A=(\mathfrak{A}, \mathcal{C}onfig_A^{in}, \mathcal{C}onfig_A^{out}, \mathcal{Z}_A)$ *un* T*-module non-expansif tel que* $\mathcal{C}onfig_A^{in} \subseteq \mathcal{C}onfig_A^{out}$*. Soit* $n \in \mathbb{N}^*$*. On définit la* n-itération de A*, notée* $\circ^n A$*, par :* $\circ^n A = A \circ (A \circ \ldots (A \circ A))$*, où le module* A *apparaît* n *fois.*

La DÉFINITION 47 implique que $\circ^n$A est bien un module. Si A est muni d'une interprétation $\mathcal{I} = (f, R_{\mathrm{A}}^{in}, R_{\mathrm{A}}^{out})$ telle que A calcule f, alors par la PROP. 27, $\circ^n$A calculera la fonction $f^{(n)} = f \circ f \circ \ldots \circ f$. La FIGURE 6.18 illustre le principe de l'itération d'un module, dans le cas de T-module et dans le cas de R-module.



(a) Cas d'un T-module. (b) Cas d'un R-module.

FIGURE 6.18 – Itération d'un module.

Exemple 12 : algorithme d'Euclide.
On peut définir l'algorithme géométrique d'Euclide comme une itération du module calculant le modulo. La machine sous-jacente est la machine $\mathfrak{M}_{pgcd}$ (*c.f.* CHAP. 4). D'après la SOUS-SEC. 4.3.1, il est clair que l'algorithme géométrique d'Euclide correspond à une itération séquentielle d'un module calculant le modulo de deux valeurs (la configuration obtenue après une itération du modulo est de la même forme que la configuration d'entrée, il est donc possible de réitérer).

6.3.3 Extensions de modules

À partir du concept de schéma d'extension introduit en Sec. 6.1, on définit une notion similaire en termes de modules géométriques : les *extensions de modules*. De manière générale et informelle, une extension de module est un module qui s'applique à d'autres modules, et qui permet de leur ajouter de nouvelles fonctionnalités.

Afin d'illustrer cette notion d'extension de module, nous reprenons les deux exemples de schémas d'extension — duplication et concentration de faisceaux — introduites en Sec. 6.1.

Module de duplication de faisceau. Le schéma d'extension de duplication de la Sec. 6.1 permet de définir un T-module correspondant. Le rôle de ce module est de pouvoir générer des configurations contenant des duplications de faisceaux à partir d'un simple faisceau. Ce module pourra être appliqué à tout module générant des faisceaux comme configurations de sortie : il s'agira donc bien d'une extension de module.

Pour appliquer une telle extension à un module, on applique déjà le schéma de duplication à un ensemble de méta-signaux qui constitue un faisceau. Il en résulte une nouvelle machine qui étend la machine originale et qui permet maintenant de dupliquer le faisceau considéré. Les configurations d'entrée considérées pour le module résultant sont les configurations faisceaux, comprenant en plus un signal duplicateur. À partir des ces configurations d'entrée, on obtient des configurations de sortie qui contiennent deux copies du faisceau (la première copie du faisceau contient les signaux dans l'ordre inverse du faisceau entrant). Ces configurations sont des configurations à supports symétriques par rapport à la position du signal duplicateur. De plus, pour chaque position occupé par un signal, la position symétrique est occupée par la version symétrique du signal.

Les paramètres de l'aire géométrique correspondant à cette extension de module (voir la Fig. 6.19) se déduisent du schéma donné en Sec. 6.1 et sont donnés par :

$$\begin{cases} d : c^{in} \mapsto \frac{|c^{in}|}{\nu} \\ s : c^{in} \mapsto 0 \\ l : c^{in} \mapsto 2|c^{in}| \end{cases} .$$

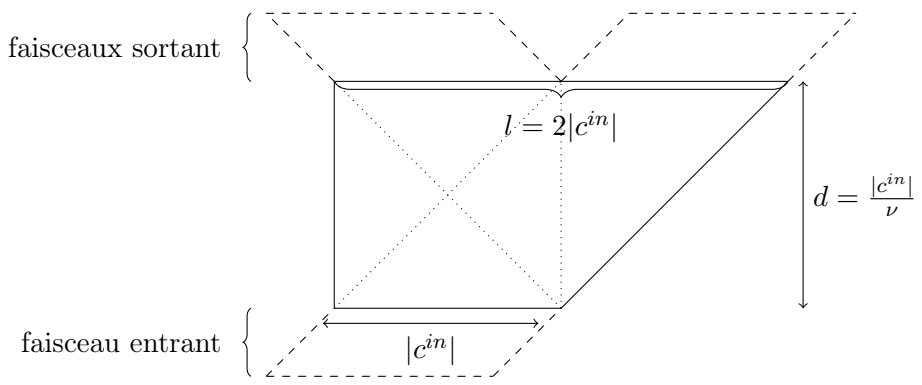


Figure 6.19 – Aire géométrique et paramètres du module de duplication.

Module de concentration de faisceau. De même que nous avons défini une extension de module de duplication à partir du schéma de duplication, nous définissons une extension de module en lentille à partir du schéma d'extension correspondant. Il s'agit d'un H-module, dont

les paramètres sont illustrés par la FIG. 6.20 et définis par :

$$\begin{cases} d_1 : c^{in} \mapsto \frac{|c^{in}|}{4\nu} \\ d_2 : c^{in} \mapsto \frac{|c^{in}|}{4\nu} \\ s_1 : c^{in} \mapsto \frac{|c^{in}|}{4} \\ s_2 : c^{in} \mapsto \frac{3|c^{in}|}{4} \\ l_1 : c^{in} \mapsto |c^{in}| \\ l_2 : c^{in} \mapsto \frac{|c^{in}|}{2} \end{cases}.$$

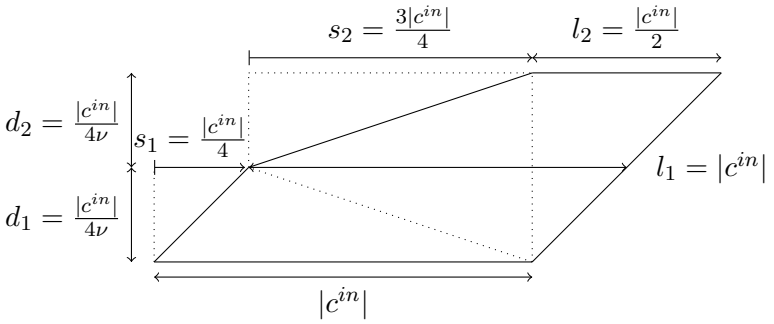


FIGURE 6.20 – Aire géométrique et paramètres du module de concentration.

6.4 Modules et complexités de collisions

Nous montrons ici dans le cadre des machines à signaux des résultats analogues à des résultats classiques : la complexité en espace d'une exécution parallèle correspond à la somme des mesures en espace de chacune des exécutions et la complexité en temps d'une composition séquentielle d'exécutions est donnée par la somme des mesures en temps de chaque exécution. Nous nous intéressons ici aux mesures de complexités définies en SEC. 3.3 et inhérentes aux machines à signaux : la *profondeur de collisions* et la *largeur de collisions*, correspondant respectivement à la profondeur et la largeur d'un diagramme espace-temps vu comme un *DAG* (*graphe dirigé acyclique*). Elles fournissent les mesures respectives de complexité en temps et de complexité en espace.

La profondeur et la largeur de collisions n'ont été définies que pour des diagrammes espace-temps, mais on s'autorisera ici à parler de la profondeur et la largeur de collisions d'un module : il s'agira de celles d'un diagramme espace-temps engendré par ce module.

Complexités d'une composition parallèle. Intuitivement, la composition parallèle synchrone de modules correspond à ces deux modules juxtaposés côte à côte dans l'espace, et évoluant de manière indépendante. La largeur de collisions — la mesure de complexité en espace sur machines à signaux — d'une composition parallèle de deux modules sera donc bornée par la somme des largeurs de collisions de chacun de ses modules. En fait, nous montrons que dans le cas d'une composition parallèle synchrone telle que définie précédemment, la largeur de collisions de la composition sera *exactement* la somme des largeurs de collisions des modules utilisés. La profondeur de collision — la mesure en temps sur machines à signaux — est quant à elle bornée par le maximum des profondeurs de collisions.

On remarquera que la taille des configurations d'entrée, ainsi que l'aspect synchrone de la composition parallèle, n'intervient pas.

Lemme 28 (Complexités en collisions d'une composition parallèle de modules)
Soient A et B deux modules tels que $A\|B$ soit défini. Alors :

(i) $pc(A\|B) = max\,\{pc(A), pc(B)\}$, *et*

(ii) $lc(A\|B) = lc(A) + lc(B)$.

Démonstration. Dans une composition parallèle synchrone, les deux sous-diagrammes sont disjoints et n'interfèrent pas. Les deux DAG correspondant sont donc déconnectés. Or, la plus grande chaîne d'un DAG est la plus grande chaîne de ses composantes connexes. D'où $pc(\mathrm{A}\|\mathrm{B}) = max\,\{pc(\mathrm{A}), pc(\mathrm{B})\}$.

De même, tout élément d'un composante connexe n'est pas comparable (au sens de l'ordre défini sur le graphe) avec tout élément n'appartenant pas à la même composante connexe, en particulier tout élément d'une anti-chaîne maximale d'une composante est incomparable avec tout élément d'une anti-chaîne maximale d'une autre composante connexe. L'union de deux anti-chaînes maximales provenant de deux composantes différentes est donc une anti-chaîne maximale pour l'union de ces deux composantes. D'où $lc(\mathrm{A}\|\mathrm{B}) = lc(\mathrm{A}) + lc(\mathrm{B})$. □

On en déduit que la profondeur de collision de la parallélisation d'un module A est le maximum des profondeurs de collisions des exécutions parallèles de A et que sa largeur de collision est la somme des largeurs de collisions des exécutions. C'est-à-dire : $pc(\|^n\mathrm{A}) = max_{1\leq i\leq n}\{pc(\mathrm{A}^{(i)})\}$, et $lc(\|^n\mathrm{A}) = \sum_{i=0}^{i=n} lc(\mathrm{A}^{(i)})$ où $\mathrm{A}^{(i)}$ désigne la $i^{\text{ème}}$ exécution parallèle du module A.

Complexités d'une composition séquentielle. La complexité d'une composition séquentielle constitue en quelque sorte la situation duale de la complexité d'une composition parallèle : la complexité en temps correspond intuitivement à la somme des complexités des deux exécutions séquentielles (elle est en fait bornée par cette somme).

Lemme 29 (Complexités en collisions d'une composition séquentielle de modules)
Soient A et B deux modules et soit $B \circ A$ le module obtenu par leur composition séquentielle. Alors :

(i) $pc(B \circ A) \leq pc(A) + pc(B)$, *et*

(ii) $lc(B \circ A) \leq lc(A) + lc(B)$.

Démonstration. Une composition séquentielle correspond à mettre « bout-à-bout » deux DAG c'est-à-dire qu'on met au moins un élément maximal du premier DAG en relation avec un élément minimal du deuxième. S'ils appartiennent tous les deux à une chaîne maximale de leur graphe respectif, alors on obtiendra une chaîne maximale du graphe union. Donc $pc(\mathrm{B} \circ \mathrm{A}) \leq pc(\mathrm{A}) + pc(\mathrm{B})$.

De même, on a clairement $lc(\mathrm{B} \circ \mathrm{A}) \leq lc(\mathrm{A}) + lc(\mathrm{B})$ car toute anti-chaîne de $\mathrm{B} \circ \mathrm{A}$ est constituée par l'union d'une anti-chaîne de A et d'une anti-chaîne de B. □

Il s'ensuit que la profondeur de collision de l'itération d'un module A est la somme des profondeurs de collisions des exécutions itérées de A et que sa largeur de collision est le maximum des largeurs de collisions des exécutions : $pc(\circ^n\mathrm{A}) = \sum_{i=0}^{i=n} pc(\mathrm{A}^{(i)})$ et $lc(\circ^n\mathrm{A}) = max_{1\leq i\leq n}\{lc(\mathrm{A}^{(i)})\}$, où $\mathrm{A}^{(i)}$ désigne la $i^{\text{ème}}$ itération du module A.

Complexités d'une extension de module. De manière générale, la complexité d'une extension de module dépend du schéma de règle utilisé. Nous reprenons ici les deux exemples traités précédemment : celui de l'extension en module duplicateur de faisceau et celui de la lentille. On désignera par $card(c_{\mathrm{A}}^{out})$ le nombre de signaux dans la configuration de sortie du module A (qui sera donc aussi le nombre de signaux dans la configuration d'entrée du duplicateur et de la lentille).

Lemme 30 (Complexités en collisions d'une extension de duplication)
Soient A un module produisant des faisceaux et A^{split} le module obtenu par application du schéma de duplication à A. Alors :

(i) $pc(A^{split}) = pc(A) + card(c_A^{out})$, *et*

(ii) $lc(A^{split}) = max\left\{lc(A), 2 \cdot card(c_A^{out})\right\}$.

En effet, pour un faisceau entrant comprenant n signaux, le faisceau qui est réfléchi (celui qui repart en direction opposée de celui qui rentre) croise intégralement le faisceau entrant. Il y aura donc en tout n^2 collisions dans l'intersection entre les deux faisceaux (la version rentrante et la version qui a rebondi) mais un seul signal du faisceau enchaînera n collisions. En sommant avec la profondeur de collision du module A, et avec $n = card(c_{\mathrm{A}}^{out})$, on obtient le résultat *(i)*. Le résultat *(ii)* est quant à lui immédiat.

On obtient de manière similaire les complexités en collisions de la lentille, en fonction des complexités du module A auquel elle est appliquée :

Lemme 31 (Complexités en collisions d'une extension de concentration)
Soient A un module produisant des faisceaux et A^{lens} le module obtenu par application du schéma de lentille à A. Alors :

(i) $pc(A^{lens}) = max\left\{pc(A) + 2, card(c_A^{out})\right\}$, *et*

(ii) $lc(A^{lens}) = lc(A)$.

Chapitre 7

Applications : résolutions spécifiques de SAT et Q-SAT

Nous allons montrer dans les trois prochains chapitres comment l'utilisation de l'arbre binaire fractale et de ses approximations vues au Chap. 5, combinées à l'approche modulaire du Chap. 6, permet de résoudre géométriquement et efficacement des problèmes algorithmiquement difficiles. Les problèmes ciblés ici sont **SAT** et **Q-SAT**, les problèmes classiques de satisfaisabilité d'une formule du calcul propositionnel. Les constructions de ce chapitre ont donné lieu aux publications [Duchier *et al.* 2010a, 2011], et correspondent à des *solutions spécifiques*, c'est-à-dire que la machine à signaux utilisée pour résoudre le problème sur une instance donnée dépend de cette instance. Une solution *générique*, c'est-à-dire utilisant une unique machine pour toutes les instances du problème, sera présentée dans le chapitre suivant.

Nous commencerons par donner quelques rappels sur les problèmes de satisfaisabilité, ainsi que des références sur la résolution de problèmes de satisfaisabilité sur modèles de calcul non-classiques. Nous expliquerons dans la section suivante comment l'arbre binaire fractale peut être interprété comme un arbre binaire de décision, utilisé pour la résolution des problèmes de satisfaisabilité booléenne. Nous donnerons également des conditions de validité pour son utilisation. Nous détaillerons dans la Sec. 7.3 les résolutions géométriques des problèmes **SAT** et **Q-SAT**, ainsi que les différentes étapes. Enfin, nous conclurons en discutant des complexités des ces constructions.

7.1 Contexte

7.1.1 Rappels sur les problèmes de satisfaisabilité

Nous rappelons ici les définitions et propriétés importantes des problèmes de satisfaisabilité, afin de souligner leur importance dans l'étude de l'efficacité d'un modèle de calcul. En effet, les problèmes de satisfaisabilité de formules du calcul propositionnel sont des représentants naturels des classes de complexité de problèmes que l'on considère habituellement comme difficiles, c'est-à-dire pour lesquels aucun algorithme en temps polynomial n'est connu. Nous renvoyons à [Papadimitriou 1994] pour plus de détails.

Le problème SAT. Le problème **SAT** est un problème classique consistant à déterminer la satisfaisabilité d'une formule du calcul propositionnel (non quantifié). Formellement :

Problème **SAT**

entrée : $\varphi(x_1, \ldots, x_n)$ une formule du calcul propositionnel.
sortie : φ est-elle satisfaisable *i.e.* existe-t-il une assignation de $(x_1, \ldots, x_n)$ telle que φ est vraie?

Une *assignation* (ou *valuation*) désigne ici une affectation des variables $x_1, \ldots, x_n$ en valeurs booléennes $\{\texttt{vrai}, \texttt{faux}\}$. Une formule du calcul propositionnel étant une formule logique utilisant les connecteurs booléens usuels ($\wedge$, $\vee$, et $\neg$) et des variables $x_1, \ldots, x_n$ à valeurs dans $\{\texttt{vrai}, \texttt{faux}\}$, nous parlons également de *formule booléenne*. Ce problème est **NP**-complet [Cook 1971 ; Levin 1973], où **NP** désigne la classe des problèmes décidables en temps polynomial par une machine de Turing non déterministe. C'est l'un des premiers problèmes à avoir démontré être **NP**-complet. En tant que tel, c'est aussi l'un des plus naturels à considérer pour étudier l'efficacité algorithmique d'un modèle de calcul. Il existe différents énoncés de ce problème, tous **NP**-complets, qui sont principalement des restrictions du problème général à un type d'entrée particulier : **3-SAT**, **CNF-SAT** ... Nous considérerons par la suite la version générale, énoncée ci-dessus, où n'importe quelle formule du calcul propositionnel ayant un nombre fini de variables constitue une instance d'entrée du problème. Nous paramétrerons la taille de l'instance d'entrée par le nombre de variables utilisées et par le nombre de symboles figurant dans la formule.

L'algorithme classique de résolution de **SAT** est un algorithme de recherche exhaustive, de type « brute-force ». Il s'agit de tester toutes les assignations possibles jusqu'à en trouver une satisfaisant la formule. Si une telle assignation est trouvée, alors le problème admet un réponse positive pour la formule d'entrée ; sinon la formule n'est pas satisfaisable. Nous allons montrer dans ce chapitre comment mener cette recherche de façon parallèle à l'aide de machines à signaux : toutes les assignations possibles seront testées en parallèle, chacune dans une zone d'espace bien définie.

Le problème Q-SAT. Le problème **Q-SAT** est une extension du problème **SAT** dans laquelle la formule donnée en instance d'entrée est *quantifiée* : toutes les variables sont liées par des quantificateurs (universels ou existentiels). Une telle formule est une formule close et admet donc une valeur de vérité lorsque le domaine des variables est l'ensemble $\{\texttt{vrai}, \texttt{faux}\}$. Le problème **Q-SAT** s'énonce de la façon suivante :

Problème **Q-SAT**
entrée : φ une formule quantifiée du calcul propositionnel.
sortie : φ est-elle vraie ?

Le problème **Q-SAT** est **PSPACE**-complet [Stockmeyer et Meyer 1973], où **PSPACE** désigne la classe de complexité des problèmes algorithmiques pouvant être décidés par des machines de Turing à espace polynomial.

L'algorithme classique « brute-force » de résolution de **Q-SAT** est donné par l'algorithme récursif $\mathbb{A}$ suivant :

$$\begin{cases} \mathbb{A}(\exists x\ \psi) & = & \mathbb{A}(\psi[x \leftarrow \texttt{faux}]) \ \vee \ \mathbb{A}(\psi[x \leftarrow \texttt{vrai}]) \\ \mathbb{A}(\forall x\ \psi) & = & \mathbb{A}(\psi[x \leftarrow \texttt{faux}]) \ \wedge \ \mathbb{A}(\psi[x \leftarrow \texttt{vrai}]) \\ \mathbb{A}(F) & = & \texttt{eval}(F) \text{ si } F \text{ est une formule booléenne sans variable} \end{cases}$$

où $\texttt{eval}(F)$ désigne la valeur booléenne de l'expression booléenne F. L'algorithme $\mathbb{A}$ résoud le problème **Q-SAT** en temps exponentiel et espace polynomial. C'est cet algorithme qui sera implémenté par signaux pour résoudre géométriquement le problème **Q-SAT**, en utilisant le fort parallélisme des machines à signaux.

Notons que le problème **SAT** est un cas particulier du problème **Q-SAT** : en effet, une formule instance de **SAT** correspond à une formule quantifiée instance de **Q-SAT** dans laquelle tous les quantificateurs sont existentiels.

7.1.2 Résolutions non-conventionnelles de problèmes difficiles

Nous avons proposé un rapide tour d'horizon des modèles de calcul, classiques et non-conventionnels, au Chap. 1. Après avoir été étudiés pour leur puissance de calcul, la plupart de ces modèles ont été étudiés d'un point de vue de l'efficacité algorithmique. Pour de telles études, les problèmes **NP**-complets tels que **SAT** et les problèmes **PSPACE**-complets tels que **Q-SAT** restent des choix privilégiés, et leur résolution permet généralement de dégager de nombreuses propriétés du modèle.

Nous proposons ici un bref état de l'art sur la résolution de problèmes algorithmiquement difficiles sur de tels modèles non-conventionnels. Une grande partie des modèles évoqués ci-dessous bénéficient d'un fort parallélisme qui est utilisé pour résoudre efficacement (*i.e.* en temps « propre au modèle » polynomial) les problèmes considérés.

L'un des premiers et plus célèbres modèles sortant du cadre classique et permettant de résoudre en temps polynomial des problèmes difficiles est certainement le modèle des machines quantiques : Peter Shor a en effet montré [Shor 1994] que ces machines peuvent résoudre le problème de factorisation en temps polynomial (notons cependant que ce problème est **FNP**-dur, **FNP** étant l'équivalent de **NP** pour les problèmes de fonctions). De nombreuses autres variantes de modèles quantiques existent, et ont été utilisées sous diverses hypothèses physiques pour résoudre d'autres problèmes : citons entre autres [Aaronson et Watrous 2009] qui proposent une étude des liens entre la classe **PSPACE** et des classes quantiques, sous l'hypothèse de l'existence de certains objets relativistes, les courbes de temps fermées ou *CTC*.

Ces derniers objets issus des théories de la relativité ont d'ailleurs été utilisés de manière directe pour résoudre **Q-SAT** par un algorithme polynomial utilisant les CTC [Brun 2003].

Les modèles optiques permettent également de résoudre des problèmes **NP**-complets en temps polynomial : il a été montré respectivement dans [Goliaei et Jalili 2011, 2012] que le calcul optique par sélection de longueur d'ondes permet de résoudre **SAT** et **3-Coloriage**.

Du côté des modèles géométriques de calcul, des problèmes tels que **3-SAT** peuvent être résolus efficacement à la fois avec des jeux de tuiles d'auto-assemblantes [Brun 2008, 2012] et avec des automates cellulaires hyperboliques [Margenstern et Morita 2001]. Dans les deux cas, le temps requis est polynomial en la taille de l'entrée, et les ressources utilisées (nombre d'assemblages en parallèle et nombre de cellules) sont en nombre exponentiel. La machine utilisée est quant à elle de taille fixe, et est utilisée pour résoudre toutes les instances du problème considéré.

Le calcul basé sur des processus inspirés du vivant fournit lui aussi des solutions efficaces à des problèmes difficiles. Leonard Adleman a ainsi montré [Adleman 1994] que le calcul par ADN permet de résoudre le problème du chemin hamiltonien. Il a ensuite été montré que le calcul par ADN et plus généralement le calcul moléculaire permet de résoudre **Q-SAT** [Zerjatke et Sturm 2011], ainsi que d'autres problèmes [Beigel et Fu 1998]. Ces calculs sont tous effectués en temps polynomial et nécessitent un nombre exponentiel de molécules. Le calcul par membrane a également fourni un cadre algorithmique pour l'étude de problèmes **NP**-complets [Păun 2001] et de problèmes **PSPACE**-complets comme **Q-SAT** [Sosík 2003 ; Alhazov et Pérez-Jiménez 2007].

De manière plus anecdotique, citons également la résolution de **SAT** en temps polynomial par des algorithmes naturels de type calcul bilatéral [Arulanandham 2005].

Les modèles évoqués ci-dessus permettent tous de résoudre des problèmes algorithmiquement difficiles en temps polynomial, avec des mesures de complexité en temps et en espace propres à chaque modèle. Presque toutes ces solutions nécessitent cependant une autre ressource qui est de mesure exponentielle (espace utilisé, nombre d'objets impliqués...), ce qui n'est guère surprenant puisque les constructions proposées utilisent généralement le fort parallélisme des modèles considérés, et utilisent de ce fait un espace (ou son équivalent suivant les modèles) exponentiel pour traiter de manière parallèle des recherches de type « brute-fore ».

Plus généralement, le statut des problèmes algorithmiquement difficiles et le lien entre les

classes **P** et **NP** sont liés aux hypothèses physiques sous-jacentes au modèle considéré. Nous renvoyons à [Abrams et Lloyd 1998] pour le lien entre quantique et complexité de calculs, à [Aaronson 2005] pour les relations entre classes de complexité relativement à différentes théories physiques, et plus spécifiquement à [Stannett 2013] pour une étude sur les liens entre certains objets de la théorie de la relativité et la relation entre **P** et **NP**.

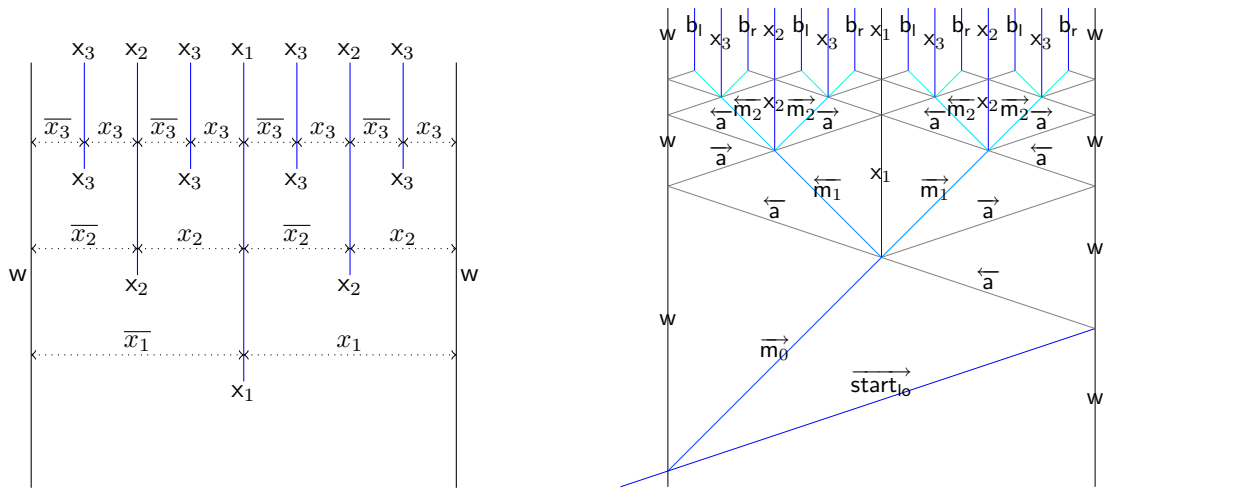
7.2 Calculer dans l'arbre binaire fractale

Nous expliquons ici comment l'arbre binaire fractale et son approximation spécifique présentés au CHAP. 5, peuvent être interprétés comme *arbre binaire de décision*, permettant ainsi d'aborder des problèmes de décisions sur les booléens. Nous donnons des conditions de validité sur une telle utilisation de l'arbre.

7.2.1 L'arbre fractale en tant qu'arbre binaire de décision

Afin de déterminer par « brute-force » si une formule propositionnelle sur n variables est satisfaisable ou pas, 2^n cas différents doivent être considérés. Ces cas peuvent être énumérés de manière récursive en utilisant un *arbre binaire de décision*. Nous expliquons ici comment construire en parallèle l'arbre complet permettant de traiter tous ces cas simultanément par une partition de l'espace, à partir de l'approximation de l'arbre binaire fractale proposée en SEC. 5.3.

Nous avons vu que l'arbre binaire fractale permettait de découper l'espace de manière uniforme, et que l'utilisation des n premiers étages de l'arbre générait 2^n parties de même taille, chaque partie étant délimitée par des signaux stationnaires. Pour utiliser cette construction en tant qu'arbre de décision, l'idée est de représenter le point de décision pour une variable x_i un signal stationnaire qui séparera l'espace en deux parties. L'espace de gauche sera alors interprété comme x_i = `false`, et celui de droite comme x_i = `true`. De manière similaire, les espaces de gauche et de droite seront à leur tour divisés par des signaux stationnaires x_{i+1}, correspondant aux sous-cas possibles d'assignation de la variable x_{n+1}. Le processus de division se poursuit de manière récursive pour toutes les variables (voir la FIG. 7.1(a)).



(a) Identification des cas.

(b) Division de l'espace.

FIGURE 7.1 – L'arbre binaire de décision.

Comme déjà mentionné au CHAP. 5, l'arbre est construit en temps borné, indépendamment de la taille de la formule. On notera qu'au dernier niveau sont disposés des signaux stationnaires d'un certain type ($\mathsf{b_r}$ et $\mathsf{b_l}$) : ces signaux seront utilisés pour évaluer la formule.

7.2.2 Conditions de validité

L'idée principale sur laquelle reposent toutes les constructions suivantes est donc d'utiliser l'arbre pour y insérer un calcul, de telle sorte que ce calcul se propage dans l'arbre en respectant l'interprétation de l'espace proposé en SOUS-SEC. 7.2.1. L'information, dans notre cas la formule booléenne dont on veut déterminer la satisfaisabilité, sera codée sous forme de faisceau et c'est ce faisceau qui se propagera dans l'arbre, comme montré par le schéma de la FIG. 7.2(a). Le faisceau se propage en suivant l'arbre, de manière parallèle aux signaux correspondant aux « branches » de l'arbre.

Nous proposons deux méthodes pour assurer une telle condition. La première méthode, utilisée dans [Duchier *et al.* 2010a], nécessite un seul méta-signal supplémentaire, ayant une vitesse rationnelle dépendant de la taille de la formule. La deuxième méthode correspond à un calibrage géométrique automatique et utilise des signaux compteurs (voir [Duchier *et al.* 2011]).

Conditions pour une distribution correcte du calcul. La propagation du faisceau à travers l'arbre est illustrée en FIG. 7.2(a). Afin d'assurer que cette propagation est correcte et que les signaux codant les variables seront assignés en valeurs booléennes `true` et `false` exactement une fois et que les évaluations des différents sous-cas se font de manière strictement indépendante, la largeur du faisceau doit être ajustée en fonction de la hauteur du dernier niveau. Nous expliquons ici la condition nécessaire à la bonne propagation du faisceau à travers l'arbre de décision. Nous nous assurons également que le faisceau généré en début de construction est suffisamment étroit pour se propager correctement dans tous les niveaux de l'arbre.

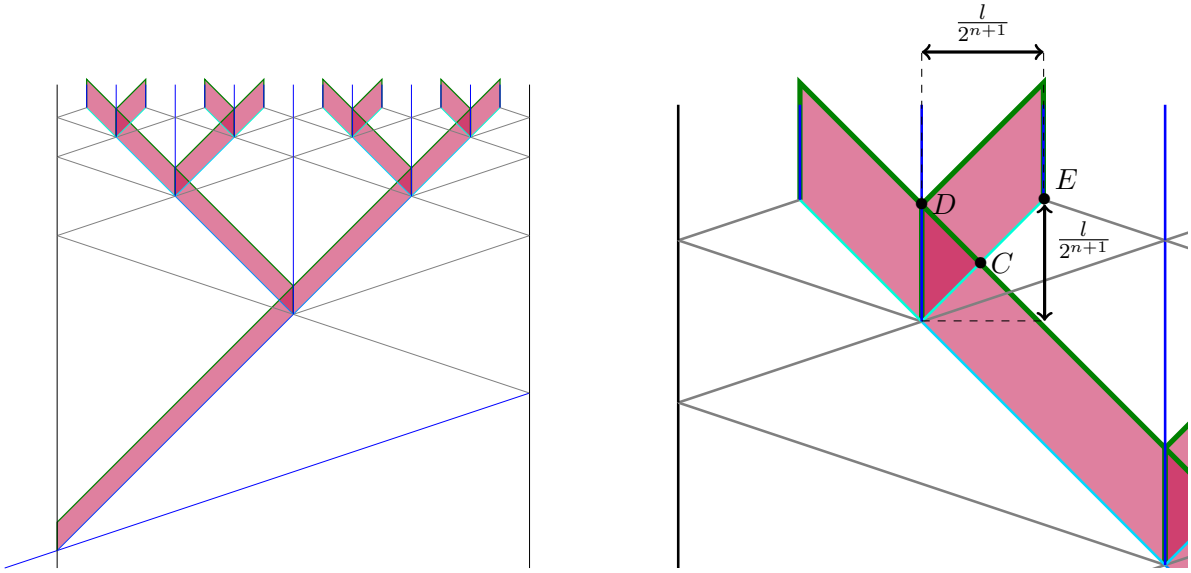
Soit l la largeur de l'espace de travail *i.e.* celle de l'arbre et soit n le nombre de variables. Tout en haut de la construction, l'espace est divisé en 2^{n+1} parties égales. Chaque partie a donc une largeur de $\frac{l}{2^{n+1}}$. Les signaux du faisceaux ayant pour vitesse 1 ou -1, le délai entre le dernier niveau de décision et le niveau d'évaluation est également de $\frac{l}{2^{n+1}}$.

Afin d'assurer que les signaux correspondant aux variables seront assignés en valeurs booléennes une fois et seulement une et que l'évaluation de la formule dans un cas n'interférera pas avec les autres cas, la largeur du faisceau ne doit pas excéder la taille des zones d'évaluation *i.e.* la largeur du faisceau doit être inférieure à $\frac{l}{2^{n+1}}$.

Cette condition sur la propagation au dernier niveau est illustrée par la FIG. 7.2(b) : le point D, marquant la fin de la dernière duplication du faisceau, doit être en dessous du point E marquant le début de l'évaluation de la formule. Cela assure également que le point de croisement C est situé strictement dessous E et donc que les processus de duplication et d'évaluation sont complètement indépendants dans le temps : l'évaluation commence strictement après la fin du dernier niveau.

L'étape d'initiation du faisceau suit deux étapes : d'abord, la distance entre les signaux parallèles constituant le faisceau est calculée de telle sorte à assurer la bonne propagation dans tout l'arbre ; ensuite, le faisceau est généré en respectant cet espacement entre signaux. Nous présentons ci-dessous deux méthodes différentes pour effectuer le calcul de la distance (la deuxième étape consistant à générer ensuite le faisceau étant essentiellement la même dans les deux méthodes).

Méthode 1 : utilisation d'un signal spécial. La première méthode proposée utilise le méta-signal $\overrightarrow{\phi_{\mathsf{R}}}$, pour générer le faisceau. La vitesse « spéciale » permet d'obtenir un espacement correct entre les signaux du faisceau. Le principe, illustré par la FIG. 7.3, est le suivant. Un signal $\overrightarrow{\phi_{\mathsf{R}}}$ sera émis depuis l'origine, en même temps que deux signaux ($\overrightarrow{\mathsf{m}_0}$ et $\overrightarrow{\mathsf{a}}$), respectivement de vitesse 1 et 3. Le rebond de $\overrightarrow{\mathsf{a}}$ produit, en plus du signal $\overleftarrow{\mathsf{a}}$ utilisé pour construire l'arbre, un autre signal $\overleftarrow{\phi_{\mathsf{L}}}$ de vitesse -6. Celui-ci entrera alors en collision successivement avec $\overrightarrow{\mathsf{m}_0}$ et $\overrightarrow{\phi_{\mathsf{R}}}$, produisant ainsi deux signaux verticaux ϕ_{W} espacés d'une distance qui dépendra des vitesses de ces trois signaux



(a) A travers l'arbre. (b) Au dernier niveau.

FIGURE 7.2 – Conditions géométriques pour une propagation correcte.

impliqués. Comme les vitesses de $\overrightarrow{\mathsf{m}_0}$ et $\overleftarrow{\phi_{\mathsf{L}}}$ ont été fixées, l'espacement entre les deux signaux dépend uniquement de la vitesse du signal $\overrightarrow{\phi_{\mathsf{R}}}$. Explicitons la valeur de cette vitesse et montrons qu'elle convient effectivement pour assurer la condition de validité précédemment définie.

Soit k le nombre total de signaux dans le faisceau. Ces signaux sont répartis avec un espacement constant de taille δ, comme montré dans la FIG. 7.3. Comme leur vitesse est 1 (resp. -1), la largeur et la hauteur du faisceau est $(k-1)\delta$. En posant $\delta_0 = \frac{l}{k.2^{n+1}}$, on assure que le faisceau se sépare correctement à un niveau avant que le processus de l'étape suivante ne commence.

Cette distance entre signaux doit être posée correctement dès le début de la construction. Pour cela, nous utilisons un processus de génération du faisceau, qui utilise des rebonds entre deux signaux verticaux séparés par la moitié de la distance δ_0 voulue. Sur la FIG. 7.3, il y a un zig-zag vertical composé de signaux de vitesse 1 et -1, de telle sorte que le délai — le δ recherché — est le double de la distance entre les deux signaux verticaux.

La collision entre $\overrightarrow{\mathsf{m}_0}$ et $\overleftarrow{\phi_{\mathsf{L}}}$ (de vitesse -6 et émis en $(l, \frac{l}{3})$) se produit aux coordonnées $(\frac{3l}{7}, \frac{3l}{7})$. Si $\overleftarrow{\phi_{\mathsf{L}}}$ traverse ce point translaté par $(-\frac{\delta_0}{2}, 0)$, alors le δ final fonctionnera également car il sera plus petit. Afin d'assurer que $\overleftarrow{\phi_{\mathsf{L}}}$ passe par ce point, et sachant qu'il est émis depuis l'origine, sa vitesse doit être :

$$\frac{\frac{3l}{7} - \frac{\delta_0}{2}}{\frac{3l}{7}} = \frac{6l - 7\frac{l}{k.2^{n+1}}}{6l} = \frac{6k.2^{n+1} - 7}{6k.2^{n+1}} \ ,$$

ce qui donne :

$$1 - \frac{7}{3k \times 2^{n+2}} \ .$$

Ce nombre rationnel peut être calculé en temps polynomial en fonction du nombre de variables et du nombre de signaux composant le faisceau, plus exactement en temps au plus quadratique.

Méthode 2 : utilisation de signaux compteurs. La deuxième méthode utilise le calcul géométrique du milieu, qui est réitéré jusqu'à obtenir une distance suffisamment petite et qui correspondra à l'espacement entre les signaux du faisceau.

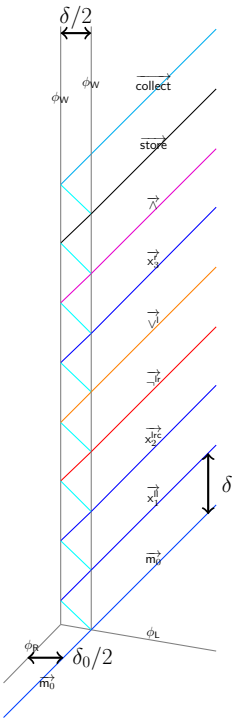


FIGURE 7.3 – Générer un faisceau (ici, le faisceau d'une formule) avec un délai correct.

On appelle w la largeur du faisceau et k le nombre de signaux le composant. L'espacement régulier entre les signaux du faisceau doit donc être égal à $\frac{w}{k}$. Mais au lieu de calculer géométriquement cette distance, nous allons utiliser l'algorithme géométrique du calcul du milieu afin d'effectuer suffisamment de division par 2 pour obtenir un espacement assez petit. La distance entre signaux du faisceau sera donc de la forme $\frac{l}{2^i}$ où i est le nombre de divisions successives par 2 et on aura donc finalement $w = (k-1) \times \frac{l}{2^i}$ (on considère $k-1$ espacements dans le faisceau puisque celui-ci est composé de k signaux régulièrement espacés). Pour cela, considérons :

$$\tau \;=\; min\; \left\{ i \in \mathbb{N} \;\middle|\; \frac{(k-1) \times 2^{n+1}}{l} \leq 2^i \right\} \quad .$$

Un tel entier τ existe toujours et est le plus petit nombre de divisions par 2 assurant une largeur de faisceau suffisamment petite. À partir d'un espace initial de taille l, on obtiendra une distance $2^{-\tau}$, disposée à gauche de l'espace de travail utilisé pour la construction de l'arbre comme cela est illustré par la FIG. 7.4. On note $\delta = \frac{1}{2^\tau}$.

Cette distance ainsi calculée est suffisante pour garantir la bonne propagation du faisceau et l'indépendance des différentes évaluations au sommet de l'arbre. En effet, le faisceau contient k signaux parallèles régulièrement espacés par une distance $2^{-\tau}$, donc sa largeur totale est $w = \frac{k-1}{2^\tau}$. Par définition de τ, on a :

$$\frac{(k-1) \times 2^{n+1}}{l} \leq 2^\tau \quad i.e. \quad \frac{k-1}{2^\tau} \leq \frac{l}{2^{n+1}}$$

et il s'ensuit que :

$$w \leq \frac{l}{2^{(n+1)}} \quad .$$

La condition de validité de propagation est donc bien vérifiée. Précisons que le nombre τ peut être calculé en temps polynomial, plus précisément en temps au plus quadratique en k et n.

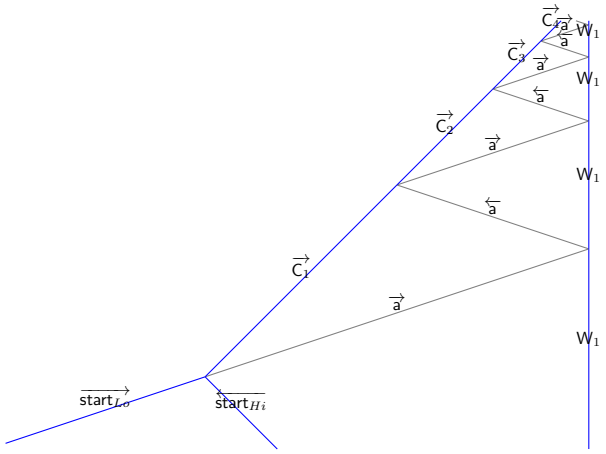
Cette construction prend place dans la partie gauche de la construction entière, la partie droite servant dans un second temps à la construction de l'arbre et à l'évaluation proprement

Meta-signal	Vitesse
$\overrightarrow{\text{start}_{Lo}}$, $\overrightarrow{\text{a}}$, $\overrightarrow{\text{w}}$	3
$\overrightarrow{\text{c}_1}$, …, $\overrightarrow{\text{c}_\tau}$	1
W_1, W_2	0
$\overleftarrow{\text{start}_{Hi}}$	−1
$\overleftarrow{\text{a}}$	−3

(a) Meta-signaux.

$$\{ \overrightarrow{\text{start}_{Lo}}, \overleftarrow{\text{start}_{Hi}} \} \to \{ \overrightarrow{\text{c}_1}, \overrightarrow{\text{a}} \}$$
$$\{ \overrightarrow{\text{a}}, \text{W}_1 \} \to \{ \overleftarrow{\text{a}}, \text{W}_1 \}$$
$$\{ \overrightarrow{\text{c}_i}, \overleftarrow{\text{a}} \} \to \{ \overrightarrow{\text{c}_{i+1}}, \overrightarrow{\text{a}} \}$$
$$\{ \overrightarrow{\text{c}_\tau}, \overleftarrow{\text{a}} \} \to \{ \text{W}_2, \overrightarrow{\text{w}} \}$$

(b) Règles de collisions (avec $1 \leq i \leq \tau - 1$).



(c) Diagramme.

FIGURE 7.4 – Calcul géométrique du délai.

dite de la formule. Après cette étape de calcul géométrique du délai, le faisceau est généré de la même manière que celle de la première méthode : les signaux compris entre les deux murs espacés par la distance δ rebondissent en zig-zaguant et génèrent successivement avec un délai $\frac{2}{3}\delta$ les signaux parallèles du faisceau (le délai est de $\frac{2}{3}\delta$ à cause des allers-retours à vitesse 3 entre les murs). On obtient finalement des signaux parallèles régulièrement espacés et se déplaçant à vitesse 1, formant ainsi le faisceau voulu. Le processus d'initiation est montré dans sa globalité par la FIG. 7.5.

7.3 Résolution de SAT par une famille de machines

Nous montrons ici comment le problème **SAT** peut être résolu par signaux en utilisant l'arbre de la façon décrite à la section précédente c'est-à-dire en y propageant un faisceau. Celui-ci représentera une formule booléenne dont on veut savoir si elle est satisfaisable. Les sous-sections suivantes détaillent successivement comment coder et propager une formule en un faisceau de signaux, comment évaluer une formule booléenne sans variable et enfin comment collecter les résultats intermédiaires pour récupérer la réponse finale.

7.3.1 Codage et propagation des formules

Nous expliquons ici comment représenter une formule propositionnelle par un ensemble de signaux. Nous illustrerons ce mécanisme avec l'exemple de la formule suivante :

$$\varphi = (x_1 \vee \neg x_2) \wedge x_3 \ .$$

L'idée de base est qu'une formule propositionnelle peut être vue comme un arbre dont les nœuds sont étiquetés par les symboles de la formule (connecteurs logiques et variables).

Après assignations des variables pour tous les cas possibles, l'évaluation de la formule pour chacun de ces cas suit un processus *bottom-up* partant des feuilles de l'arbre pour remonter jusqu'à la racine. Pour modéliser ce processus, nous allons représenter chaque nœud de l'arbre par un signal. Afin de distinguer les différentes occurrences d'un même symbole dans l'arbre et

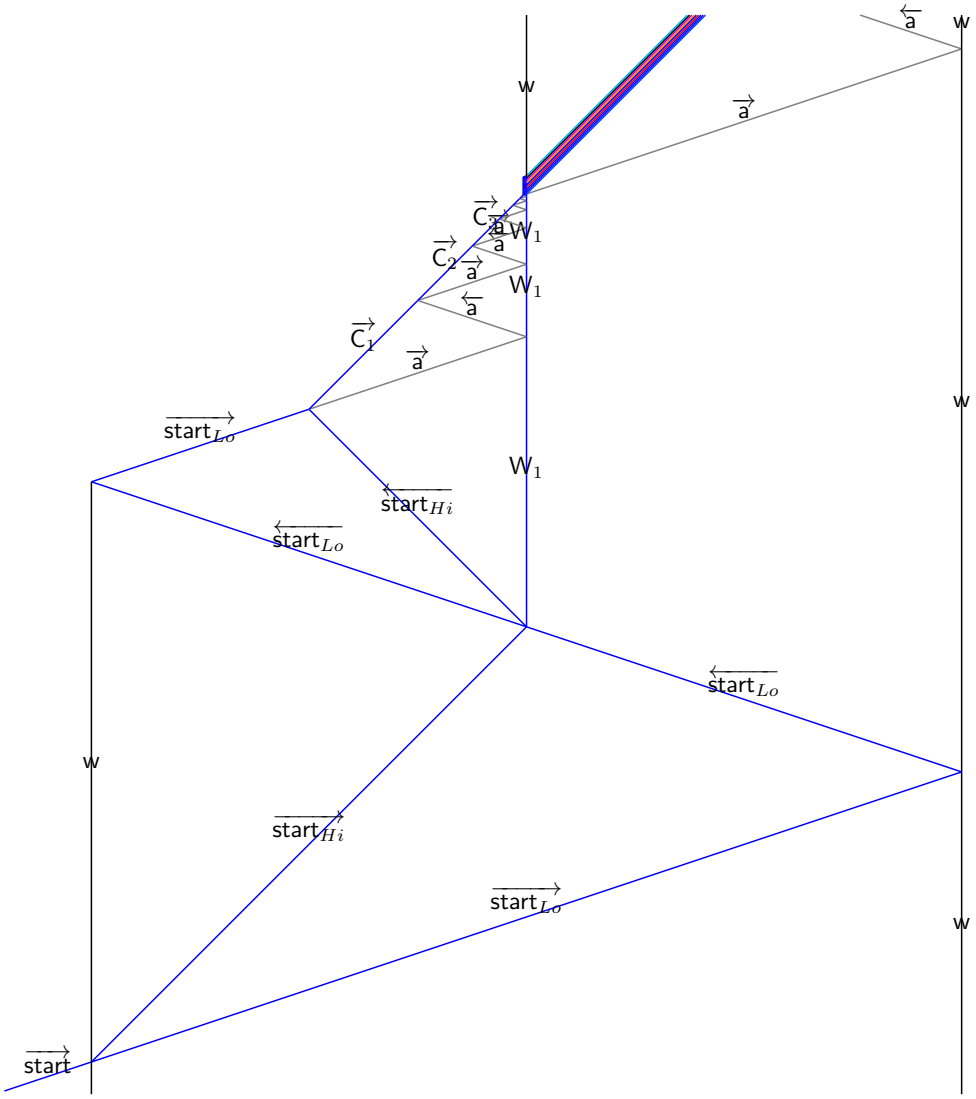


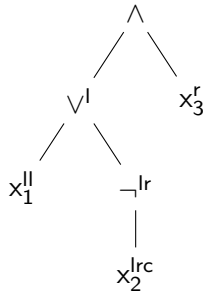
FIGURE 7.5 – L'étape complète d'initiation.

de permettre à un nœud d'interagir uniquement avec son nœud-père, les nœuds sont en plus décorés d'un chemin qui identifie de manière unique leur position dans l'arbre depuis la racine.

Les nœuds ainsi décorés de leur chemin fournissent les noms des méta-signaux correspondants. Ainsi une formule de taille s nécessitera la définition de $2s$ méta-signaux : s méta-signaux de vitesse 1 et s de vitesse -1.

Ces signaux sont ensuite disposés de manière à former un faisceau. Comme ils sont tous de même vitesse, le faisceau est uniquement constitué de signaux parallèles. Les signaux sont placés de telle sorte que ceux qui codent les sous-formules les plus internes sont placés d'abord. Un signal codant un nœud sera donc situé au-dessus de tous les signaux codant ses nœuds fils. Le respect de cet ordre est essentiel pour que le processus d'évaluation par percolation se déroule correctement.

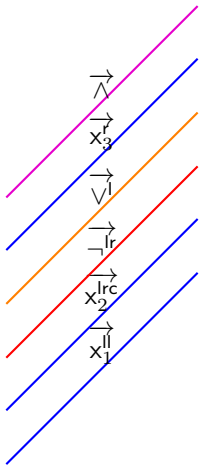
Nous expliquons ici comment une formule propositionnelle peut être vue comme un arbre étiqueté et comment nous utilisons cette forme pour générer les méta-signaux et règles de collisions correspondant à la formule considérée. Les signaux et règles ne dépendant pas de la formule sont donnés par le TAB. 7.1 et sont ceux utilisés pour préparer le faisceau suivant la méthode 1 vue précédemment ; ceux qui dépendent de la formule sont générés par l'ALGO. 7.1.



(a) Arbre étiquetté.

Méta-signal	Vitesse
$\overleftarrow{\wedge}$, $\overleftarrow{\vee^{\mathsf{l}}}$, $\overleftarrow{\neg^{\mathsf{lr}}}$, $\overleftarrow{x_1^{\mathsf{ll}}}$, $\overleftarrow{x_2^{\mathsf{lrc}}}$, $\overleftarrow{x_3^{\mathsf{r}}}$	-1
$\overrightarrow{\wedge}$, $\overrightarrow{\vee^{\mathsf{l}}}$, $\overrightarrow{\neg^{\mathsf{lr}}}$, $\overrightarrow{x_1^{\mathsf{ll}}}$, $\overrightarrow{x_2^{\mathsf{lrc}}}$, $\overrightarrow{x_3^{\mathsf{r}}}$	1

(b) Méta-signaux générés.



(c) Codage de la formule en faisceau.

FIGURE 7.6 – Compilation de la formule $\varphi = (x_1 \vee \neg x_2) \wedge x_3$ en faisceau.

Méta-signaux	Vitesse
$\overrightarrow{\mathsf{start}_{\mathsf{lo}}}$	3
$\overrightarrow{\mathsf{m}_0}$	1
ϕ_{W}, w	0
$\overleftarrow{\mathsf{m}_0}$	-1
$\overleftarrow{\mathsf{a}}$	-3
$\overleftarrow{\phi_{\mathsf{L}}}$	-6

Règles de collisions
$\{\ \overrightarrow{\mathsf{start}_{\mathsf{lo}}},\ \mathsf{w}\ \} \to \{\ \overleftarrow{\phi_{\mathsf{L}}},\ \overleftarrow{\mathsf{a}},\ \mathsf{w}\ \}$
$\{\ \overrightarrow{\mathsf{m}_0},\ \overleftarrow{\phi_{\mathsf{L}}}\ \} \to \{\ \overleftarrow{\phi_{\mathsf{L}}},\ \overleftarrow{\mathsf{m}_0},\ \phi_{\mathsf{W}},\ \overrightarrow{\mathsf{m}_0}\ \}$
$\{\ \phi_{\mathsf{W}},\ \overleftarrow{\mathsf{m}_0}\ \} \to \{\ \phi_{\mathsf{W}},\ \overrightarrow{\mathsf{m}_0}\ \}$
$\{\ \phi_{\mathsf{W}},\ \overleftarrow{\mathsf{a}}\ \} \to \{\ \overleftarrow{\mathsf{a}}\ \}$

TABLEAU 7.1 – Préparer le faisceau codant une formule.

Les arbres étiquetés. Nous décrivons ici comment identifier une formule à un arbre étiqueté dont les nœuds sont identifiés par leur chemin jusqu'à la racine. Un chemin π est un mot (fini) sur l'alphabet $\mathbb{N}$, et les nœuds seront étiquetés par des connecteurs logiques ou des variables propositionnelles. On suppose que Σ est la signature de symboles de fonction f, $g\ldots$, chacun étant muni d'une arité $\mathsf{ar} \geq 0$. On note $\mathbb{N}^*$ l'ensemble $\mathbb{N} \setminus \{0\}$.

Un *domaine d'arbre* D est un sous-ensemble fini de $\mathbb{N}^*$ qui est clos par préfixe et coupe à droite :

$$\forall \pi, \pi' \in \mathbb{N}_0^* \qquad \pi\pi' \in D \ \Rightarrow\ \pi \in D$$
$$\forall \pi \in \mathbb{N}_0^*,\ \forall i, j \in \mathbb{N}_0 \qquad i < j \wedge \pi j \in D \ \Rightarrow\ \pi i \in D$$

Un *arbre étiqueté* $\tau = (D_\tau, L_\tau)$ est la donnée d'un domaine d'arbre D_τ et d'une fonction d'étiquetage $L_\tau : D_\tau \to \Sigma$ qui associe un symbole à chaque nœud en respectant les arités :

$$\mathsf{ar}(L_\tau(\pi)) = n \ \Rightarrow\ \pi n \in D_\tau \ \wedge\ \pi(n+1) \notin D_\tau \qquad \forall \pi \in D_\tau, \forall n > 0$$
$$\mathsf{ar}(L_\tau(\pi)) = 0 \ \Rightarrow\ \pi 1 \notin D_\tau \qquad \forall \pi \in D_\tau$$

Les formules propositionnelles en tant qu'arbres étiquetés. On prend comme signature Σ celle qui est formée à partir de variables propositionnelles (arité 0), le connecteur $\neg$ (arité 1), et les connecteurs $\wedge$ et $\vee$ (arité 2).

Étant donnée une formule $\varphi = (D_\varphi, L_\varphi)$, l'idée est de définir un méta-signal pour chacun de ses nœuds. Considérons un nœud $\pi \in D_\varphi$ avec $L_\varphi(\pi) = \ell$: les méta-signaux associés seront notés $\overrightarrow{\ell^\pi}$ et $\overleftarrow{\ell^\pi}$ (respectivement pour les versions droite et gauche). On note $\mathcal{N}_\varphi$ l'ensemble des nœuds étiquetés de φ *i.e.* $\mathcal{N}_\varphi = \{L_\varphi(\pi)^\pi\}_{\pi \in D_\varphi} = \{\ell^\pi\}_{\pi \in D_\varphi}$.

Les méta-signaux codant la formule ont tous la même vitesse, qui est celle du faisceau : vitesse 1 (respectivement -1) pour le faisceau se déplaçant vers la droite (respectivement vers la gauche).

On note $\prec$ l'ordre lexicographique sur D_φ et $\lceil D_\varphi \rceil$ l'élément maximal au sens de $\prec$ dans D_φ. On note $\lfloor \pi \rfloor$ le $\prec$-prédécesseur de π dans D_φ. Intuitivement, si $\pi \prec \pi'$, alors $\overrightarrow{L_\varphi(\pi)^\pi}$ sera émis après $\overrightarrow{L_\varphi(\pi')^{\pi'}}$.

Entrée : $\mathfrak{M}$ une machine à signaux et une formule φ (de taille t et à n variables).
Sortie : $\mathfrak{M}$ étendue pour générer le faisceau codant la formule φ.

; // méta-signal de vitesse ν et règle pour le calibrage du faisceau
$\overrightarrow{\phi_{\mathsf{R}}} \longleftarrow \mathfrak{M}.ajouter\_méta\_signal\_de\_vitesse(\nu)$ // $\nu = 1 - \frac{7}{3(t+3)\times 2^{n+2}}$
$\mathfrak{M}.ajouter\_règle\left(\left\{\overrightarrow{\phi_{\mathsf{R}}}, \overleftarrow{\phi_{\mathsf{L}}}\right\} \to \{\phi_{\mathsf{W}}\}\right)$
// méta-signaux codant la formule
Pour $\ell^\pi \in \mathcal{N}_\varphi$ **Faire**
  $\overrightarrow{\ell^\pi} \longleftarrow \mathfrak{M}.ajouter\_méta\_signal\_de\_vitesse(1)$
  $\overleftarrow{\ell^\pi} \longleftarrow \mathfrak{M}.ajouter\_méta\_signal\_de\_vitesse(-1)$
// règles de génération du faisceau
$\mathfrak{M}.ajouter\_règle\left(\{\overrightarrow{\mathsf{m}_0}, \phi_{\mathsf{W}}\} \to \left\{\overleftarrow{\ell^{\lceil D_\varphi \rceil}}, \overrightarrow{\mathsf{m}_0}, \phi_{\mathsf{W}}\right\}\right)$
Pour $\ell^\pi \in \mathcal{N}_\varphi$ **Faire**
  Si $\pi = \varepsilon$ **Alors**
    $\mathfrak{M}.ajouter\_règle\left(\left\{\phi_{\mathsf{W}}, \overleftarrow{\ell^\varepsilon}\right\} \to \left\{\phi_{\mathsf{W}}, \overrightarrow{\ell^\varepsilon}\right\}\right)$
    $\mathfrak{M}.ajouter\_règle\left(\left\{\overrightarrow{\ell^\varepsilon}, \phi_{\mathsf{W}}\right\} \to \left\{\overleftarrow{\mathsf{store}}, \phi_{\mathsf{W}}, \overrightarrow{\ell^\varepsilon}\right\}\right)$
  Sinon
    $\mathfrak{M}.ajouter\_règle\left(\left\{\phi_{\mathsf{W}}, \overleftarrow{\ell^\pi}\right\} \to \left\{\phi_{\mathsf{W}}, \overrightarrow{\ell^\pi}\right\}\right)$
    $\mathfrak{M}.ajouter\_règle\left(\left\{\overrightarrow{\ell^\pi}, \phi_{\mathsf{W}}\right\} \to \left\{\overleftarrow{\ell^{\lfloor \pi \rfloor}}, \phi_{\mathsf{W}}, \overrightarrow{\ell^\pi}\right\}\right)$
$\mathfrak{M}.ajouter\_règle\left(\left\{\phi_{\mathsf{W}}, \overleftarrow{\mathsf{store}}\right\} \to \left\{\overrightarrow{\mathsf{store}}, \phi_{\mathsf{W}}\right\}\right)$
$\mathfrak{M}.ajouter\_règle\left(\left\{\overrightarrow{\mathsf{store}}, \phi_{\mathsf{W}}\right\} \to \left\{\overleftarrow{\mathsf{collect}}, \phi_{\mathsf{W}}, \overrightarrow{\mathsf{store}}, \right\}\right)$
$\mathfrak{M}.ajouter\_règle\left(\left\{\phi_{\mathsf{W}}, \overleftarrow{\mathsf{collect}}\right\} \to \left\{\phi_{\mathsf{W}}, \overrightarrow{\mathsf{collect}}, \right\}\right)$

ALGORITHME 7.1 – Générer les règles spécifiques de création du faisceau codant une formule.

Exemple 13 : $\varphi = (x_1 \vee \neg x_2) \wedge x_3$**.**
On reprend la formule propositionnelle $\varphi = (x_1 \vee \neg x_2) \wedge x_3$ utilisée pour générer les diagrammes. La formule φ possède six nœuds donnés par $\mathcal{N}_\varphi = \{\mathsf{x}_1^{11}, \mathsf{x}_2^{121}, \neg^{12}, \vee^1, \mathsf{x}_3^2, \wedge^\varepsilon\}$. Pour rendre les notations plus simples, on notera plutôt l, r et c pour désigner respectivement la gauche, la droite et le centre dans le chemin d'un nœud. On désignera par ε le mot vide, qui correspond au chemin du premier nœud de l'arbre étiqueté. L'exemple précédent devient alors $\mathcal{N}_\varphi = \{\mathsf{x}_1^{\mathsf{ll}}, \mathsf{x}_2^{\mathsf{lrc}}, \neg^{\mathsf{lr}}, \vee^{\mathsf{l}}, \mathsf{x}_3^{\mathsf{r}}, \wedge\}$.

Propager le faisceau. Nous expliquons ici comment propager le faisceau codant la formule à travers l'arbre binaire de décision. Pour chaque point de décision, le faisceau est dupliqué : le faisceau principal passe au travers du point de décision et il est également réfléchi. On obtient ainsi un exemplaire du faisceau qui est envoyé vers la gauche et un autre exemplaire qui est envoyé vers la droite. Ainsi, le faisceau sera propagé dans chaque branche de l'arbre et atteindra chaque point de décision au moins une fois. Si le faisceau est suffisamment étroit, cette garantie devient « le faisceau atteint chaque point de décision exactement une fois ». Nous avons précédemment montré dans la SOUS-SEC. 7.2.2 comment obtenir cette condition sur la largeur du faisceau en émettant depuis l'origine le signal $\overleftarrow{\phi_{\mathsf{L}}}$, de vitesse $\nu = 1 - 7/(3(t+3)\cdot 2^{n+2})$, où t est le nombre de symbole de la formule et n est le nombre de variables. Pour propager la formule jusqu'aux différents cas à évaluer au sommet de la construction, le faisceau se propage sur la base de la construction de l'arbre, suivant les couloirs de la FIG. 7.2(a) donnée en SOUS-SEC. 7.2.2. Par construction de la vitesse ν, la précédente condition sur la largeur du faisceau nous assure que chaque dédoublement du faisceau à un niveau se termine avant que le dédoublement du niveau suivant ne commence.

Quand le faisceau rencontre un point de décision (un signal stationnaire), le faisceau se duplique et produit deux nouvelles branches, chacune étant une copie du faisceau arrivant. À part le changement de sens (vitesse opposée), tous les signaux restent identiques dans les deux branches, sauf les signaux de variables qui correspondent au niveau de décision produisant la duplication : au $i^{\text{ème}}$ niveau, les signaux x_i deviennent `false` dans la branche gauche et `true` dans la branche droite, les étiquettes respectives étant conservées. La FIGURE 7.7 zoome sur l'intersection du faisceau avec le premier point de décision : les signaux incidents $\overrightarrow{\mathsf{x}_1^{\mathsf{ll}}}$ deviennent $\overleftarrow{\mathsf{f}^{\mathsf{ll}}}$ partant à gauche et $\overrightarrow{\mathsf{t}^{\mathsf{ll}}}$ partant à droite. La règle de collisions qui permet l'assignation des variables est $\{\overrightarrow{\mathsf{x}_i^{\pi}}, \mathsf{x}_i\} \to \{\overleftarrow{\mathsf{f}^{\pi}}, \mathsf{x}_i, \overrightarrow{\mathsf{t}^{\pi}}\}$, où π désigne le chemin de la variable x_i dans l'arbre de la formule.

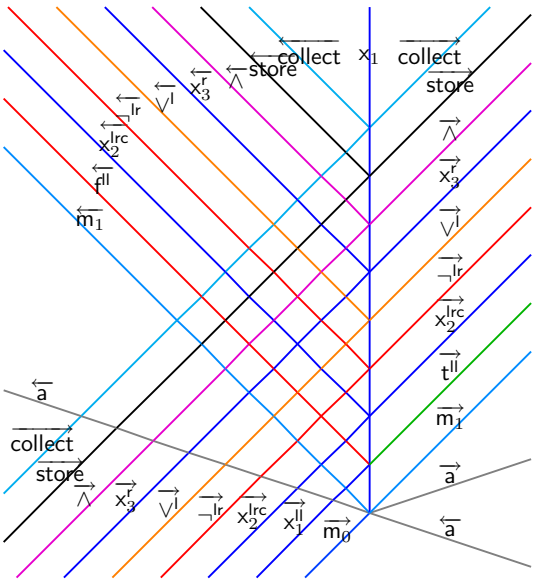


FIGURE 7.7 – Duplication du faisceau au premier niveau.

Ainsi, niveau après niveau, toutes les occurrences de variables dans la formule sont remplacées par des signaux booléens étiquetés par leur chemin dans l'arbre de la formule. Comme chaque point de décision est rencontré exactement une fois pour chaque variable dans chaque branche de l'arbre, après n niveaux, tous les signaux correspondant aux variables ont disparu et ont tous été remplacés par des signaux booléens.

L'algorithme qui permet de générer les règles de propagation est donné par l'Algo. 7.2. Tandis que tous les autres algorithmes de compilation pour cette construction sont en temps linéaire en la taille de la formule, l'Algo. 7.2 fonctionne en temps quadratique (à cause des deux boucles « **Pour** » imbriquées et toutes deux bornées par t).

Entrée : $\mathfrak{M}$ une machine à signaux.
Sortie : $\mathfrak{M}$ étendue pour propager le faisceau de la formule.

1 $\mathsf{R} \longleftarrow \mathfrak{M}.ajouter\_méta\_signal\_de\_vitesse(0)$
2 $\mathsf{L} \longleftarrow \mathfrak{M}.ajouter\_méta\_signal\_de\_vitesse(0)$
`// méta-signaux booléens pour assigner les variables`
3 **Pour chaque** $\ell^\pi \in \mathcal{N}_\phi$ **Faire**
4 **Si** $\ell = x_i$ **Alors**
5 $\overrightarrow{\mathsf{t}^\pi} \longleftarrow \mathfrak{M}.ajouter\_méta\_signal\_de\_vitesse(1)$
6 $\overrightarrow{\mathsf{f}^\pi} \longleftarrow \mathfrak{M}.ajouter\_méta\_signal\_de\_vitesse(1)$
7 $\overleftarrow{\mathsf{t}^\pi} \longleftarrow \mathfrak{M}.ajouter\_méta\_signal\_de\_vitesse(-1)$
8 $\overleftarrow{\mathsf{f}^\pi} \longleftarrow \mathfrak{M}.ajouter\_méta\_signal\_de\_vitesse(-1)$
`// règles aux points de décision`
9 **Pour** i **de** 1 **à** n **Faire**
10 **Pour chaque** $\ell^\pi \in \mathcal{N}_\phi$ **Faire**
11 **Si** $\ell = x_i$ **Alors**
 `// les variables` x_i `sont affectées en signaux booléens au niveau` i
12 $\mathfrak{M}.ajouter\_règle\left(\left\{\overrightarrow{\ell^\pi}, \mathsf{x}_i\right\} \to \left\{\overleftarrow{\mathsf{f}^\pi}, \mathsf{x}_i, \overrightarrow{\mathsf{t}^\pi}\right\}\right)$
13 $\mathfrak{M}.ajouter\_règle\left(\left\{\overleftarrow{\ell^\pi}, \mathsf{x}_i\right\} \to \left\{\overleftarrow{\mathsf{f}^\pi}, \mathsf{x}_i, \overrightarrow{\mathsf{t}^\pi}\right\}\right)$
14 **Sinon**
 `// les autres signaux sont dupliqués`
15 $\mathfrak{M}.ajouter\_règle\left(\left\{\overrightarrow{\ell^\pi}, \mathsf{x}_i\right\} \to \left\{\overleftarrow{\ell^\pi}, \mathsf{x}_i, \overrightarrow{\ell^\pi}\right\}\right)$
16 $\mathfrak{M}.ajouter\_règle\left(\left\{\overleftarrow{\ell^\pi}, \mathsf{x}_i\right\} \to \left\{\overleftarrow{\ell^\pi}, \mathsf{x}_i, \overrightarrow{\ell^\pi}\right\}\right)$
`// duplication des signaux ne dépendant pas de la formule`
17 $\mathfrak{M}.ajouter\_règle\left(\left\{\overrightarrow{\mathsf{store}}, \mathsf{x}_i\right\} \to \left\{\overleftarrow{\mathsf{store}}, \mathsf{x}_i, \overrightarrow{\mathsf{store}}\right\}\right)$
18 $\mathfrak{M}.ajouter\_règle\left(\left\{\mathsf{x}_i, \overleftarrow{\mathsf{store}}\right\} \to \left\{\overleftarrow{\mathsf{store}}, \mathsf{x}_i, \overrightarrow{\mathsf{store}}\right\}\right)$
19 $\mathfrak{M}.ajouter\_règle\left(\left\{\overrightarrow{\mathsf{collect}}, \mathsf{x}_i\right\} \to \left\{\overleftarrow{\mathsf{collect}}, \mathsf{L}, \overrightarrow{\mathsf{collect}}\right\}\right)$
20 $\mathfrak{M}.ajouter\_règle\left(\left\{\mathsf{x}_i, \overleftarrow{\mathsf{collect}}\right\} \to \left\{\overleftarrow{\mathsf{collect}}, \mathsf{R}, \overrightarrow{\mathsf{collect}}\right\}\right)$

Algorithme 7.2 – Générer les règles spécifiques de propagation du faisceau.

7.3.2 Évaluation des formules

Nous avons généré au sommet de l'arbre des signaux stationnaires $\mathsf{b_l}$ et $\mathsf{b_r}$ pour marquer la division de l'espace : leur rôle est d'initier le processus d'évaluation. Le signal $\mathsf{b_l}$ est utilisé pour démarrer l'évaluation sur un branche du faisceau allant à gauche, alors que $\mathsf{b_r}$ correspond à l'évaluation d'une branche à droite. La Figure 7.8 zoome sur un cas d'évaluation pour l'exemple de la formule φ.

Le processus d'évaluation commence une fois que toute les occurrences de variables ont été remplacées par les valeurs booléennes en fonction du chemin de la branche dans l'arbre. Le faisceau arrive ainsi sur les signaux $\mathsf{b_l}$ et $\mathsf{b_r}$ après la séparation du dernier niveau et l'assignation des dernières variables restantes, représentées par les signaux x_n. Les règles pour l'évaluation sont

définies de telle sorte à respecter les opérations booléennes classiques. L'évaluation de la formule suit un processus de reconstruction qui est l'inverse de celui impliqué dans la génération de la formule : les signaux se collisionnent en respectant l'ordre lexicographique défini sur l'ensemble de nœuds $\mathcal{N}_\phi$. Les étiquettes de chemin sont ici essentielles pour assurer que la bonne sous-formule interagit avec les bonnes occurrences des connecteurs parents.

Les deux cas étant symétriques, nous expliquons seulement le cas de $\mathsf{b_r}$. Les signaux de la formule valuée arrivent par la gauche et sont traités comme sur la FIG. 7.8. L'idée principale est que tout signal de la formule (variable ou nœud connecteur) est évaluée avant d'atteindre le signal $\mathsf{b_r}$. L'invariant est donc que tous les signaux qui atteignent $\mathsf{b_r}$ ont des valeurs booléennes. Cette valeur booléenne est alors réfléchie sur $\mathsf{b_r}$ pour ensuite interagir avec le reste du faisceau arrivant, de telle façon qu'elle rencontre son connecteur parent avant que celui-ci n'atteigne $\mathsf{b_r}$. Les valeurs réfléchies sont notées en majuscules et conservent les étiquettes de leur chemin : par exemple, $\overrightarrow{\mathsf{t^{ll}}}$ devient après réflexion le signal $\overleftarrow{\mathsf{T^{ll}}}$. Un signal booléen noté en minuscules doit donc attendre d'atteindre $\mathsf{b_r}$ pour être réfléchi et et devenir la valeur notée en majuscules, qui peut alors interagir avec les prochains signaux rencontrés. L'ordre de disposition des signaux dans le faisceau et les étiquettes assurent que les signaux réfléchis sont pris en argument par les connecteurs parents correspondants. L'ALGO. 7.3 permet de générer les règles d'évaluation correspondant aux signaux étiquettés de la formule.

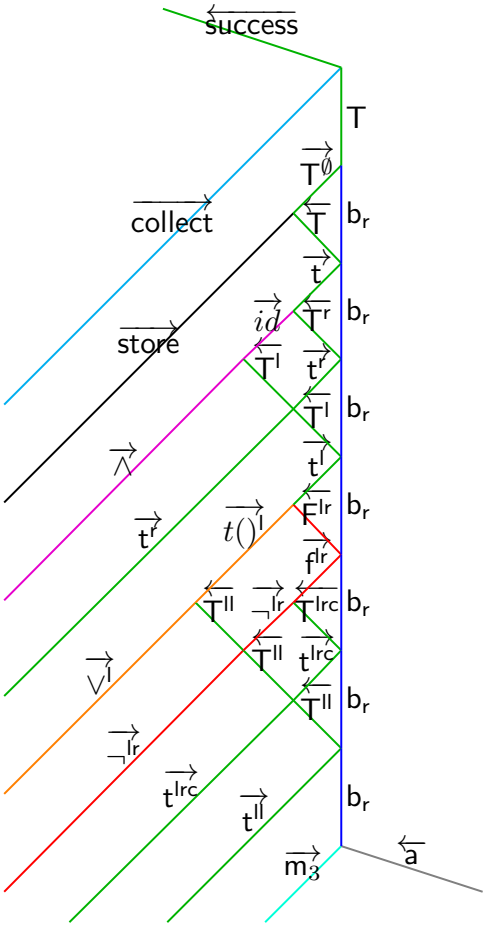


FIGURE 7.8 – Évaluation de la formule $\varphi(x_1, x_2, x_3)$ lorsque $x_1 = x_2 = x_3 = vrai$.

Un connecteur est évalué par les collisions avec les signaux booléens (majuscules) de ses arguments. Par exemple, le signal de disjonction collisionne avec son premier argument. Suivant la valeur de ce premier argument, il devient soit le signal constant $\overrightarrow{t()^{\mathsf{l}}}$ attendant un argument

Entrée : $\mathfrak{M}$ une machine à signaux.
Sortie : $\mathfrak{M}$ étendue pour évaluer une formule booléenne.

`// méta-signaux de connecteurs et booléens et activation`
Pour $\ell^\pi \in \mathcal{N}_\phi$ **Faire**
  Pour $\mathsf{b} \in \{\mathsf{t}, \mathsf{f}, \mathsf{T}, \mathsf{F}\}$ **Faire**
    $\overrightarrow{\mathsf{b}^\pi} \longleftarrow \mathfrak{M}.ajouter\_méta\_signal\_de\_vitesse(1)$
  $\mathfrak{M}.ajouter\_règle\left(\left\{\overrightarrow{\mathsf{t}^\pi}, \mathsf{b_r}\right\} \to \left\{\overleftarrow{\mathsf{T}^\pi}, \mathsf{b_r}\right\}\right)$
  $\mathfrak{M}.ajouter\_règle\left(\left\{\overrightarrow{\mathsf{f}^\pi}, \mathsf{b_r}\right\} \to \left\{\overleftarrow{\mathsf{F}^\pi}, \mathsf{b_r}\right\}\right)$
  Si *($\ell = \vee$* **ou** *$\ell = \wedge$)* **Alors**
    $\overrightarrow{id()^\pi} \longleftarrow \mathfrak{M}.ajouter\_méta\_signal\_de\_vitesse(1)$
    Si $\ell = \vee$ **Alors**
      $\overrightarrow{t()^\pi} \longleftarrow \mathfrak{M}.ajouter\_méta\_signal\_de\_vitesse(1)$
    Sinon
      $\overrightarrow{f()^\pi} \longleftarrow \mathfrak{M}.ajouter\_méta\_signal\_de\_vitesse(1)$

`// implémenter les opérateurs booléens`
Pour $\ell^\pi \in \mathcal{N}_\phi$ **Faire**
  Si $\ell = \neg$ **Alors**
    $\mathfrak{M}.ajouter\_règle\left(\left\{\overrightarrow{\ell^\pi}, \overleftarrow{\mathsf{T}^{\pi c}}\right\} \to \left\{\overrightarrow{\mathsf{f}^\pi}\right\}\right)$
    $\mathfrak{M}.ajouter\_règle\left(\left\{\overrightarrow{\ell^\pi}, \overleftarrow{\mathsf{F}^{\pi c}}\right\} \to \left\{\overrightarrow{\mathsf{t}^\pi}\right\}\right)$
  Si *($\ell = \vee$* **ou** *$\ell = \wedge$)* **Alors**
    $\mathfrak{M}.ajouter\_règle\left(\left\{\overrightarrow{id()^\pi}, \overleftarrow{\mathsf{T}^{\pi r}}\right\} \to \left\{\overrightarrow{\mathsf{T}^\pi}\right\}\right)$
    $\mathfrak{M}.ajouter\_règle\left(\left\{\overrightarrow{id()^\pi}, \overleftarrow{\mathsf{F}^{\pi r}}\right\} \to \left\{\overrightarrow{\mathsf{F}^\pi}\right\}\right)$
    Si $\ell = \vee$ **Alors**
      $\mathfrak{M}.ajouter\_règle\left(\left\{\overrightarrow{\ell^\pi}, \overleftarrow{\mathsf{T}^{\pi l}}\right\} \to \left\{\overrightarrow{t()^\pi}\right\}\right)$
      $\mathfrak{M}.ajouter\_règle\left(\left\{\overrightarrow{\ell^\pi}, \overleftarrow{\mathsf{F}^{\pi l}}\right\} \to \left\{\overrightarrow{id()^\pi}\right\}\right)$
    Sinon
      $\mathfrak{M}.ajouter\_règle\left(\left\{\overrightarrow{\ell^\pi}, \overleftarrow{\mathsf{T}^{\pi l}}\right\} \to \left\{\overrightarrow{id()^\pi}\right\}\right)$
      $\mathfrak{M}.ajouter\_règle\left(\left\{\overrightarrow{\ell^\pi}, \overleftarrow{\mathsf{F}^{\pi l}}\right\} \to \left\{\overrightarrow{f()^\pi}\right\}\right)$

`// projeter le résultat de l'évaluation sur le mur`
Pour $\mathsf{b} \in \{\mathsf{T}, \mathsf{F}\}$ **Faire**
  $\mathfrak{M}.ajouter\_règle\left(\left\{\overrightarrow{\mathsf{store}}, \overleftarrow{\mathsf{b}}\right\} \to \left\{\overrightarrow{\mathsf{b}^\emptyset}\right\}\right)$
  $\mathfrak{M}.ajouter\_règle\left(\left\{\overrightarrow{\mathsf{b}^\emptyset}, \mathsf{b_r}\right\} \to \{\mathsf{b}\}\right)$

ALGORITHME 7.3 – Générer les règles spécifiques d'évaluation d'une formule.

(lorsque le premier argument était `true`) soit le signal identité $\overrightarrow{id^{\mathsf{l}}}$ attendant un argument (lorsque le premier argument était `false`). Ces deux cas correspondent bien aux règles classiques d'évaluation d'une disjonction : une disjonction ayant déjà un argument vrai sera vraie indépendamment du second argument alors qu'une disjonction ayant reçu un argument faux prendra la valeur de son deuxième argument. Ces deux cas possibles d'une disjonction partiellement évaluée et en attente du deuxième argument sont respectivement codés par les signaux $\overrightarrow{t()^{\mathsf{l}}}$ et $\overrightarrow{id^{\mathsf{l}}}$. Les règles d'évaluation de la disjonction sont données dans le TAB. 7.2, qui correspond aux règles générées

par l'ALGO. 7.3 (ainsi que les règles symétriques) pour le cas de la disjonction $\vee^l$.

$$
\begin{array}{lll}
\{ \overrightarrow{\vee^{\mathsf{l}}}, \overleftarrow{\mathsf{T}^{\mathsf{ll}}} \} \to \{ \overrightarrow{t()^{\mathsf{l}}} \} & \{ \overrightarrow{t()^{\mathsf{l}}}, \overleftarrow{\mathsf{T}^{\mathsf{lr}}} \} \to \{ \overrightarrow{\mathsf{t}^{\mathsf{l}}} \} & \{ \overrightarrow{id^{\mathsf{l}}}, \overleftarrow{\mathsf{T}^{\mathsf{lr}}} \} \to \{ \overrightarrow{\mathsf{t}^{\mathsf{l}}} \} \\
\{ \overrightarrow{\vee^{\mathsf{l}}}, \overleftarrow{\mathsf{F}^{\mathsf{ll}}} \} \to \{ \overrightarrow{id^{\mathsf{l}}} \} & \{ \overrightarrow{t()^{\mathsf{l}}}, \overleftarrow{\mathsf{F}^{\mathsf{lr}}} \} \to \{ \overrightarrow{\mathsf{t}^{\mathsf{l}}} \} & \{ \overrightarrow{id^{\mathsf{l}}}, \overleftarrow{\mathsf{F}^{\mathsf{lr}}} \} \to \{ \overrightarrow{\mathsf{f}^{\mathsf{l}}} \} \\
\{ \overleftarrow{\vee^{\mathsf{l}}}, \overrightarrow{\mathsf{T}^{\mathsf{ll}}} \} \to \{ \overleftarrow{t()^{\mathsf{l}}} \} & \{ \overleftarrow{t()^{\mathsf{l}}}, \overrightarrow{\mathsf{T}^{\mathsf{lr}}} \} \to \{ \overleftarrow{\mathsf{t}^{\mathsf{l}}} \} & \{ \overleftarrow{id^{\mathsf{l}}}, \overrightarrow{\mathsf{T}^{\mathsf{lr}}} \} \to \{ \overleftarrow{\mathsf{t}^{\mathsf{l}}} \} \\
\{ \overleftarrow{\vee^{\mathsf{l}}}, \overrightarrow{\mathsf{F}^{\mathsf{ll}}} \} \to \{ \overleftarrow{id^{\mathsf{l}}} \} & \{ \overleftarrow{t()^{\mathsf{l}}}, \overrightarrow{\mathsf{F}^{\mathsf{lr}}} \} \to \{ \overleftarrow{\mathsf{t}^{\mathsf{l}}} \} & \{ \overleftarrow{id^{\mathsf{l}}}, \overrightarrow{\mathsf{F}^{\mathsf{lr}}} \} \to \{ \overleftarrow{\mathsf{f}^{\mathsf{l}}} \}
\end{array}
$$

TABLEAU 7.2 – Évaluer la disjonction $\vee^l$.

À la fin de l'évaluation, le signal $\overrightarrow{\mathsf{store}}$ projette la valeur de vérité de la formule initiale sur $\mathsf{b_r}$, qui permet de stocker la valeur jusqu'à ce que $\overrightarrow{\mathsf{collect}}$ déclenche la collecte des résultats.

7.3.3 Collecte et résultat final

Les résultats partiels provenant de l'étape d'évaluation doivent maintenant être regroupés afin d'obtenir la réponse finale, à savoir dans le cas de **SAT** s'il existe une évaluation ayant produit un résultat *vrai*. Le cas de **Q-SAT** doit en plus prendre en compte le type de quantificateurs correspondant aux variables.

Récupérer les résultats partiels (cas de SAT). Après les phases de propagation et d'évaluation des différents cas, les résultats partiels sont enregistrés sous forme de signaux stationnaires, remplaçant les signaux $\mathsf{b_l}$ et $\mathsf{b_r}$ utilisés lors de l'évaluation.

Nous devons maintenant effectuer la disjonction de tous ces résultats afin d'obtenir la solution finale au problème **SAT**. Cette *phase de collecte* est déclenchée par les signaux $\overleftarrow{\mathsf{collect}}$ et $\overrightarrow{\mathsf{collect}}$ et procède par disjonction des signaux booléens deux par deux, en respectant une structure inverse de l'arbre initial. La partie du diagramme correspondant à la phase de collecte est donnée par la FIG. 7.9. L'ensemble des règles nécessaires est donné par le TAB. 7.3.

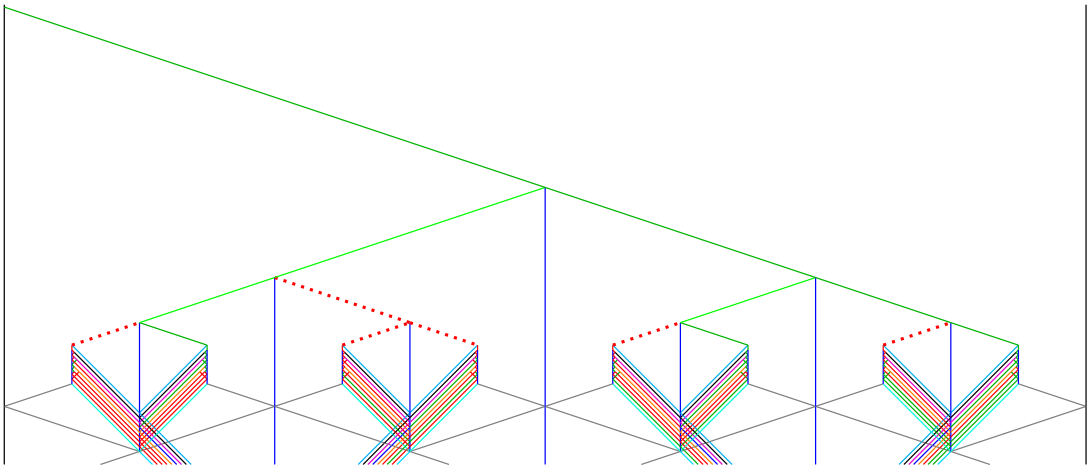


FIGURE 7.9 – Collecte des résultats.

$$
\begin{array}{ll}
\{ B, \overrightarrow{\mathsf{collect}} \} \to \{ \overleftarrow{B} \} & \{ \overrightarrow{\mathsf{collect}}, \mathsf{x}_i \} \to \{ \overleftarrow{\mathsf{collect}}, \mathsf{L}, \overrightarrow{\mathsf{collect}} \} \\
\{ B, \overleftarrow{\mathsf{collect}} \} \to \{ \overrightarrow{B} \} & \{ \mathsf{x}_i, \overleftarrow{\mathsf{collect}} \} \to \{ \overleftarrow{\mathsf{collect}}, \mathsf{R}, \overrightarrow{\mathsf{collect}} \} \\
\{ \overrightarrow{B_1}, \mathsf{R}, \overleftarrow{B_2} \} \to \{ \overrightarrow{B_3} \} & \text{pour } B, B_1, B_2, B_3 \in \{\mathsf{T}, \mathsf{F}\} \\
\{ \overrightarrow{B_1}, \mathsf{L}, \overleftarrow{B_2} \} \to \{ \overleftarrow{B_3} \} & \text{et } B_3 = B_1 \vee B_2
\end{array}
$$

TABLEAU 7.3 – Collecter les résultats (sans quantificateur).

Récupérer les résultats partiels en respectant les quantificateurs (cas de Q-SAT). Après les phases de propagation et d'évaluation, le résultat de l'évaluation de la formule non quantifiée pour chaque cas possible d'assignation est stocké sous forme de signal stationnaire qui remplace les signaux $\mathsf{b_l}$ et $\mathsf{b_r}$. Nous devons maintenant regrouper ces 2^n résultats (un pour chaque assignation possible) afin d'obtenir la valeur de vérité de la formule quantifiée.

Nous avions construit l'arbre de telle façon que chaque niveau corresponde à une variable : le niveau 1 pour la variable x_1, le niveau 2 pour x_2... Pour agréger les résultats, nous utilisons un processus qui « défait » cette structure d'arbre en joignant les résultats deux par deux : nous obtenons ainsi une structure d'arbre binaire inversé ou chaque niveau correspond à la jonction des résultats partiels pour les cas de la variable correspondant à ce niveau, et où la jonction des résultats deux par deux respecte le quantificateur liant cette variable.

Pour la formule ϕ précédemment considérée, les règles de jonction des résultats pour le premier niveau de la collecte sont données dans le TAB. 7.4. Ce niveau correspond à la variable x_3, et comme x_3 est liée dans ϕ par un quantificateur universel, chaque jonction de ce niveau effectuera une conjonction des deux résultats booléens arrivant.

$$
\begin{array}{ll}
\{\ \overrightarrow{\mathsf{T}},\ \mathrm{R}^3_\forall,\ \overleftarrow{\mathsf{T}}\ \} \to \{\ \overrightarrow{\mathsf{T}}\ \} & \{\ \overrightarrow{\mathsf{T}},\ \mathrm{L}^3_\forall,\ \overleftarrow{\mathsf{T}}\ \} \to \{\ \overleftarrow{\mathsf{T}}\ \} \\
\{\ \overrightarrow{\mathsf{T}},\ \mathrm{R}^3_\forall,\ \overleftarrow{\mathsf{F}}\ \} \to \{\ \overrightarrow{\mathsf{F}}\ \} & \{\ \overrightarrow{\mathsf{T}},\ \mathrm{L}^3_\forall,\ \overleftarrow{\mathsf{F}}\ \} \to \{\ \overleftarrow{\mathsf{F}}\ \} \\
\{\ \overrightarrow{\mathsf{F}},\ \mathrm{R}^3_\forall,\ \overleftarrow{\mathsf{T}}\ \} \to \{\ \overrightarrow{\mathsf{F}}\ \} & \{\ \overrightarrow{\mathsf{F}},\ \mathrm{L}^3_\forall,\ \overleftarrow{\mathsf{T}}\ \} \to \{\ \overleftarrow{\mathsf{F}}\ \} \\
\{\ \overrightarrow{\mathsf{F}},\ \mathrm{R}^3_\forall,\ \overleftarrow{\mathsf{F}}\ \} \to \{\ \overrightarrow{\mathsf{F}}\ \} & \{\ \overrightarrow{\mathsf{F}},\ \mathrm{L}^3_\forall,\ \overleftarrow{\mathsf{F}}\ \} \to \{\ \overleftarrow{\mathsf{F}}\ \} \\
\{\ \mathsf{x}_3,\ \overleftarrow{\mathsf{collect}}\ \} \to \{\ \overleftarrow{\mathsf{collect}},\ \mathrm{R}^3_\forall,\ \overrightarrow{\mathsf{collect}}\ \} & \{\ \overrightarrow{\mathsf{collect}},\ \mathsf{x}_3\ \} \to \{\ \overleftarrow{\mathsf{collect}},\ \mathrm{L}^3_\forall,\ \overrightarrow{\mathsf{collect}}\ \}
\end{array}
$$

TABLEAU 7.4 – Collecter les résultats : exemple d'un quantificateur universel sur x_3.

À chaque duplication du faisceau, c'est-à-dire à chaque fois qu'il rencontre un signal stationnaire de décision x_i, le signal $\overrightarrow{\mathsf{collect}}$ (respectivement $\overleftarrow{\mathsf{collect}}$) qui se situe tout en haut du faisceau, change le signal stationnaire x_i en $\mathrm{L}^i_{\mathrm{Q}_i}$ (respectivement $\mathrm{R}^i_{\mathrm{Q}_i}$). Les signaux $\mathrm{L}^i_{\mathrm{Q}_i}$ et $\mathrm{R}^i_{\mathrm{Q}_i}$ encodent chacun deux informations : Q_i désigne le quantificateur liant x_i dans ϕ ($\exists$ ou bien $\forall$), et L (respectivement R) indique la direction dans laquelle envoyer le résultat combiné des cas $x_i = \texttt{false}$ (arrivant par la gauche) et $x_i = \texttt{true}$ (arrivant par la droite). La collecte est réalisée de manière symétrique à l'arbre binaire de décision et ainsi, chaque niveau de collecte correspond à tous les sous-cas possibles pour une variable donnée : le niveau 1 de la collecte (le premier niveau juste après l'évaluation) correspond à la variable x_n, le niveau 2 à la variable x_{n-1}, et ainsi de suite jusqu'au niveau n, le dernier niveau, qui correspond à la jonction des deux sous-cas assignés dès le début par l'arbre de décision pour la variable x_1. C'est ce dernier niveau qui produit la réponse finale. L'opération booléenne effectuant la combinaison de deux résultats à niveau donné dépend du type de quantificateur liant la variable correspondant à ce niveau : $\vee$ pour $\exists$ et $\wedge$ pour $\forall$.

L'étape complète de collecte avec quantificateur correspond au diagramme de la FIG. 7.10, et les règles requises sont résumées par le TAB. 7.5.

Le TABLEAU 7.5 fournit les règles de collisions nécessaires à la collecte des résultats de toutes les évaluations. Comme cette collecte dépend à chaque niveau des quantificateurs liant les variables dans la formule **Q-SAT**, ces règles dépendent elles aussi de la formule. Les signaux $\mathrm{R}^i_{\mathrm{Q}_i}$ et $\mathrm{L}^i_{\mathrm{Q}_i}$ sont ceux émis durant la phase de propagation, où Q_i désigne le quantificateur de x_i (voir l'ALGO. 7.2). Le TABLEAU 7.4 donne les règles explicites pour la collecte au 3ème niveau sur l'exemple courant. Comme la variable x_3 est liée par un quantificateur universel, les jonctions au niveau 3 devront effectuer des conjonctions booléennes.

Diagrammes globaux. Tous les processus précédemment décrits forment les diagrammes espace-temps globaux présentés en FIG. 7.11 et 7.12. Le diagramme de la FIG. 7.11 correspond à une solution pour une instance de **SAT** et utilise la première méthode de calibrage du faisceau.

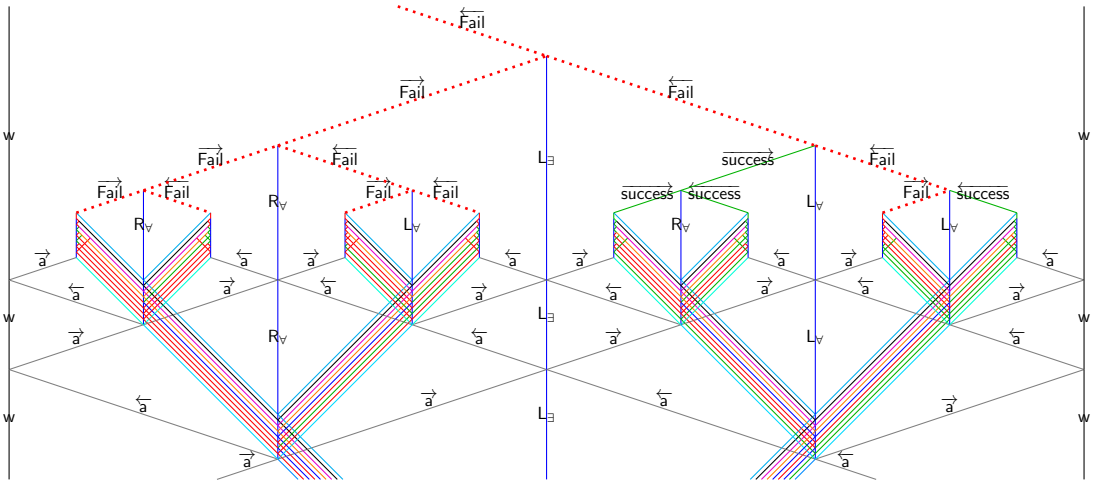


FIGURE 7.10 – Collecte des résultats avec prise en compte de quantificateurs.

$$
\begin{array}{rl}
\{\ \overrightarrow{\mathsf{collect}},\ B\ \} \to \{\ \overleftarrow{B}\ \} & \{\ \overrightarrow{\mathsf{collect}},\ \mathsf{x}_i\ \} \to \{\ \overleftarrow{\mathsf{collect}},\ \mathrm{L}^i_{\mathrm{Q}_i},\ \overrightarrow{\mathsf{collect}}\ \} \\
\{\ B,\ \overleftarrow{\mathsf{collect}}\ \} \to \{\ \overrightarrow{B}\ \$ & \{\ \mathsf{x}_i,\ \overleftarrow{\mathsf{collect}}\ \} \to \{\ \overleftarrow{\mathsf{collect}},\ \mathrm{R}^i_{\mathrm{Q}_i},\ \overrightarrow{\mathsf{collect}}\ \} \\
\{\ \overrightarrow{B_1},\ \mathrm{R_Q},\ \overleftarrow{B_2}\ \} \to \{\ \overrightarrow{B_3}\ \} & \text{avec } B, B_1, B_2, B_3 \in \{\mathsf{T}, \mathsf{F}\},\ \mathrm{Q} \in \{\exists, \forall\} \text{ et} \\
\{\ \overrightarrow{B_1},\ \mathrm{L_Q},\ \overleftarrow{B_2}\ \} \to \{\ \overleftarrow{B_3}\ \} & B_3 = B_1 \vee B_2 \text{ si } \mathrm{Q} = \exists,\ B_3 = B_1 \wedge B_2 \text{ si } \mathrm{Q} = \forall
\end{array}
$$

TABLEAU 7.5 – Collecter les résultats en respectant les quantificateurs.

La deuxième méthode de calibrage produit un diagramme comme celui de la FIG. 7.12 (lequel correspond également à une solution pour une instance de **Q-SAT**). Les résultats sont donnés par les signaux quittant la construction tout en haut à gauche : la formule **SAT** du diagramme en FIG. 7.11 est satisfaisable (le signal vert est un signal `true`), tandis que la formule **Q-SAT** du diagramme en FIG. 7.12 est fausse (le signal en pointillés rouge est un signal `false`).

7.4 Complexités de la construction

Complexités des algorithmes de compilation. L'ALGORITHME 7.1, qui génère les signaux et règles nécessaires pour créer le faisceau codant la formule, est linéaire en la taille de la formule. Les règles nécessaires à l'évaluation étant en nombre linéaire (16 règles pour chaque symbole de connecteur binaire, et 4 pour chaque symbole de négation), elles sont également générées en temps linéaire par l'ALGO. 7.3. L'ALGORITHME 7.2 étant quant à lui quadratique, le temps total pour générer les méta-signaux et les règles de la machine à signaux qui représente une formule de taille t, est donc en $\mathcal{O}(t^2)$. Rappelons que le calcul de la valeur spéciale ν s'effectue également en temps quadratique.

La taille de la machine générée est composée de la façon suivante. Le nombre de méta-signaux est linéaire en n pour l'arbre et en t pour la partie correspondant à la formule. Le nombre de règles de collisions non blanches est proportionnel à $n \cdot t$ (car chaque méta-signal de symbole de la formule se duplique à chaque niveau), ce qui donne un nombre de règles en $\mathcal{O}(t^2)$ (car $n \leq t$).

Nous insistons sur le fait qu'il n'y ait que 7 vitesses différentes : -6, -3, -1, 0, 1, 3 et la vitesse spéciale ν utilisée pour le calibrage du faisceau.

Complexités en collisions. Tout en haut de l'arbre, lors de l'évaluation, 2^n processus indépendants de taille t se déroulent en parallèle. Il existe donc une anti-chaîne de longueur inférieure à $t \cdot 2^n$. En constatant que les autres anti-chaînes sont forcément de longueur inférieure, on en déduit que la largeur de collision est exponentielle.

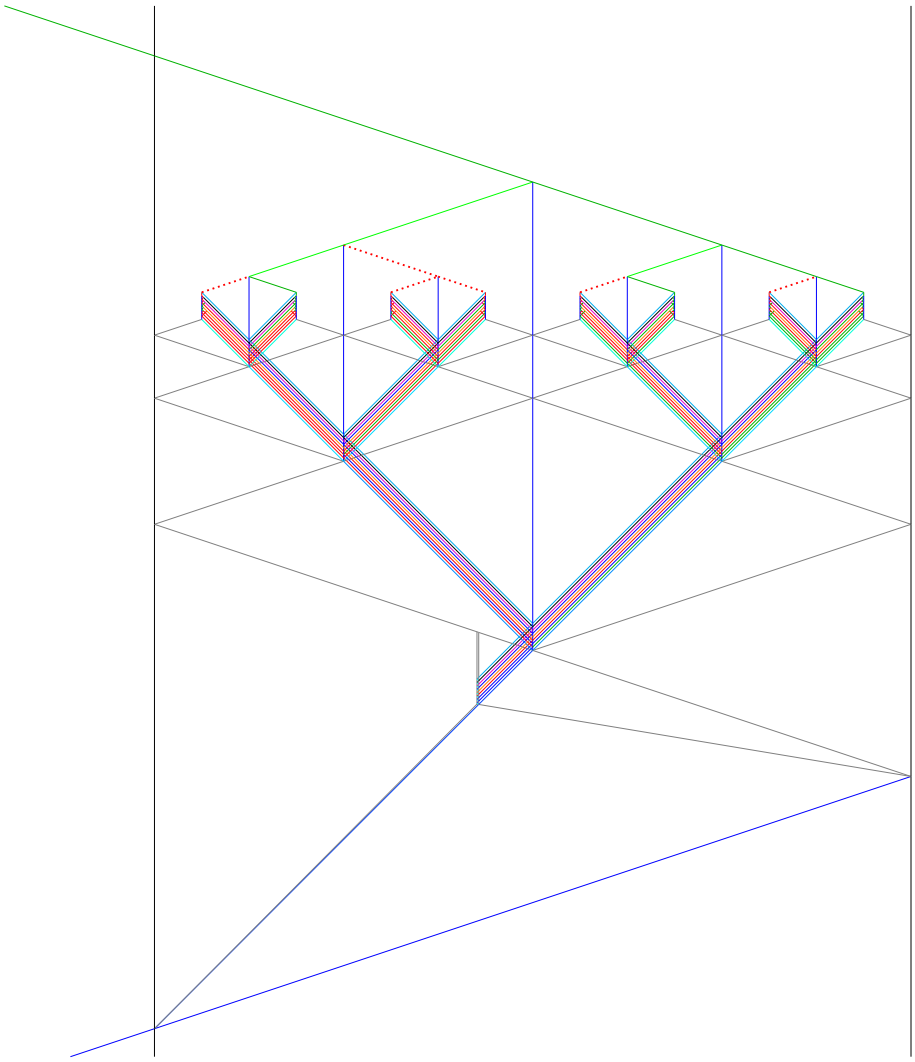


FIGURE 7.11 – Diagramme global de résolution spécifique du problème **SAT** pour l'instance $\varphi = (x_1 \vee \neg x_2) \wedge x_3$ (avec la première méthode de calibrage du faisceau).

Concernant la profondeur de collision : la mise en place de l'arbre, le calibrage du faisceau, la propagation, l'évaluation et la collecte contribuent tous en un nombre de collisions qui est au plus linéaire quand on suit n'importe quelle chaîne de collisions. Cependant, à chaque duplication du faisceau, les intersections entre les deux faisceaux sortants (le faisceau incident et le faisceau réfléchi) ajoutent un nombre de collisions qui est linéaire en t. Comme il y a n niveaux dans l'arbre, et donc n duplications quand on suit un chemin dans le faisceau, on obtient un nombre de collisions enchaînées en $\mathcal{O}(n \cdot t)$. Finalement, la profondeur de collision est donc en $\mathcal{O}(t^2)$.

Complexités globales. La complexité globale est donnée à la fois par la complexité de la partie classique *i.e.* celle des algorithmes de compilation de la machine à partir de la formule et par la complexité de la partie intrinsèque au modèle *i.e.* celle du diagramme espace-temps correspondant à l'instance considérée.

Nous obtenons donc une construction globalement en temps quadratique en la taille de la formule, et en espace exponentiel (à cause de la largeur de collision du diagramme qui est exponentielle).

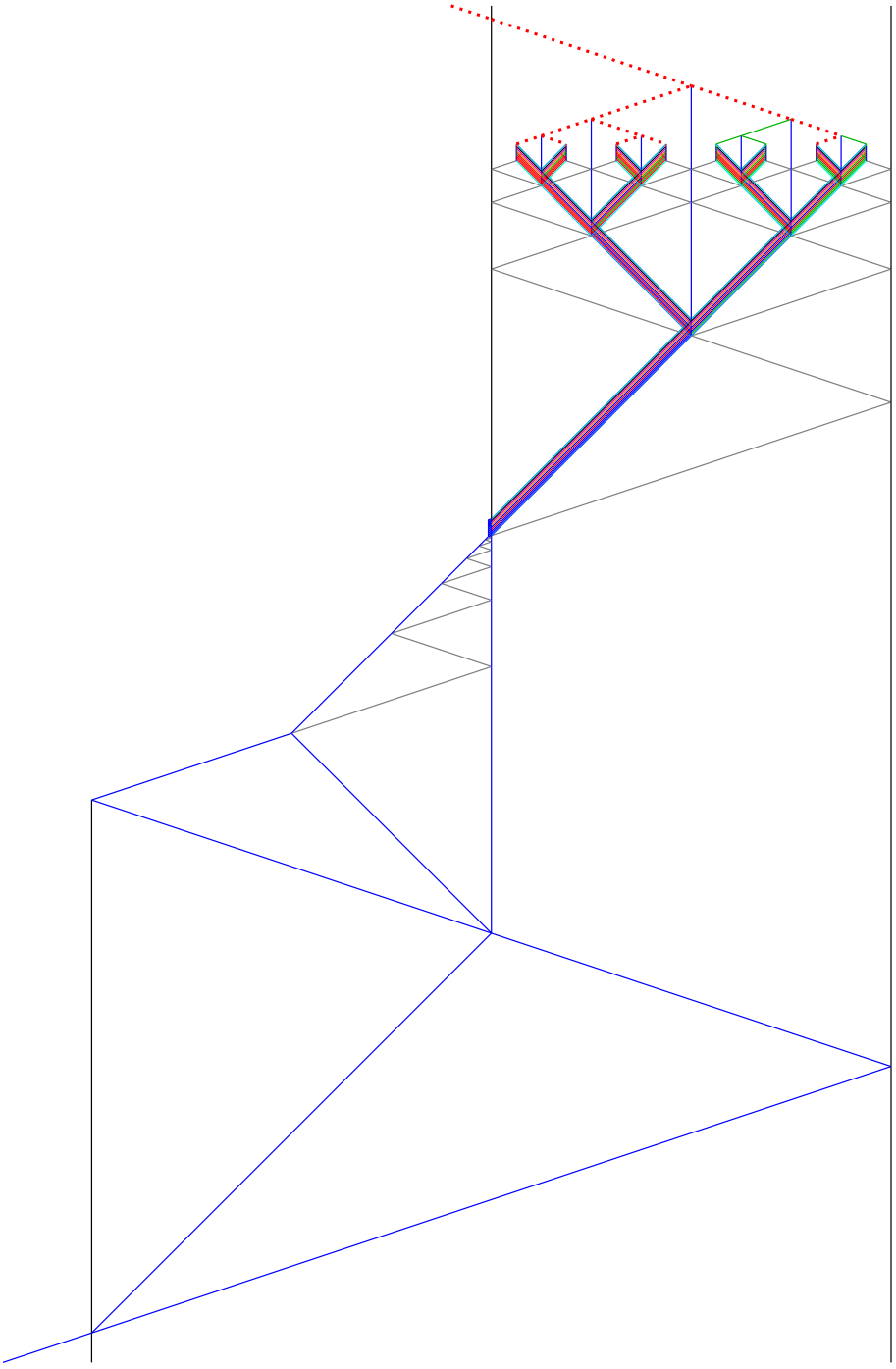


FIGURE 7.12 – Diagramme global de résolution spécifique du problème **Q-SAT** pour l'instance $\varphi = \exists x_1 \forall x_2 \forall x_3 \;\; x_1 \wedge (\neg x_2 \vee x_3)$ (avec la deuxième méthode de calibrage du faisceau).

Chapitre 8

Résolution générique de Q-SAT

Le chapitre précédent contenait une résolution spécifique des problèmes **SAT** et **Q-SAT** : à chaque instance de ces problèmes étaient associés une machine à signaux et un diagramme espace-temps qui fournissait la réponse au problème pour l'instance considérée. Nous présentons ici la construction de [Duchier *et al.* 2012] qui correspond à une résolution *générique* du problème **Q-SAT** : l'unique machine qui sera détaillée dans ce chapitre permet de résoudre **Q-SAT** pour *n'importe quelle instance du problème*. Au lieu d'être codée dans la machine (*i.e.* dans les méta-signaux et les règles de collisions) comme au chapitre précédent, l'instance sera représentée uniquement dans la configuration initiale. La résolution générique suit le même principe que la résolution spécifique : l'arbre binaire est utilisé pour y propager la formule jusqu'au dernier niveau où elle y sera évaluée pour toutes les assignations des variables possibles ; les différents résultats partiels sont ensuite récupérés et agrégés en fonction des quantificateurs pour obtenir la réponse finale. Toutes ces étapes sont effectuées par une unique machine, ayant un nombre fixe et fini de méta-signaux et de règles. La correction de la construction découle de l'approche modulaire développée au Chap. 6 et des modules qui y ont été donnés en exemples.

Comme pour le chapitre précédent, nous allons détailler les différentes étapes de la construction, la plupart correspondant à des modules. Nous commencerons par présenter l'arbre générique de décision et comment la lentille permet de se passer de l'étape de calibrage du faisceau. Nous expliquerons ensuite comment représenter et évaluer une formule booléenne de manière générique c'est-à-dire en utilisant un ensemble fixe et fini de méta-signaux et de règles. Nous donnerons enfin la décomposition modulaire de la construction. De cette dernière se déduira une preuve de correction, ainsi que la profondeur et la largeur de collisions de la construction, respectivement cubique et exponentielle en la taille de la formule d'entrée.

8.1 Se propager dans l'arbre générique

Dans cette section, nous expliquons comment utiliser l'arbre générique en tant qu'arbre binaire de décision, et comment assurer une bonne propagation du faisceau grâce au module de lentille, qui permet de rétrécir le faisceau au fur et à mesure qu'il progresse dans les différents niveaux de l'arbre.

8.1.1 L'arbre générique de décision

Comme pour la construction du Chap. 7, seuls les n premiers niveaux de l'arbre binaire sont requis pour évaluer une formule définie sur n variables en suivant l'interprétation de l'arbre binaire en tant qu'arbre de décision. Pour effectuer cette construction de manière générique, nous reprenons l'approximation générique de l'arbre binaire donnée en Sous-sec. 5.3.2. Alors que la

construction présentée en SOUS-SEC. 5.3.2 requérait un ajustement fin de la largeur du faisceau en fonction du niveau n d'approximation souhaité, nous en présentons ici une variante dont la validité est plus simple à garantir : il suffira en effet que le faisceau soit initialement placé dans un intervalle de départ fixe et indépendant du nombre de niveaux. Nous rajoutons pour cela une *lentille-duplicatrice*, dont les idées principales (duplication et concentration de faisceaux) ont déjà été présentées en SOUS-SEC. 6.1.2. La FIGURE 8.1 fournit le diagramme de la construction des quatre premiers niveaux de l'arbre, à l'aide d'une lentille. Ce diagramme est semblable à celui de FIG. 5.11(a), excepté les passages du faisceau à chaque à niveau, qui, en plus de le dupliquer, le concentre en divisant sa largeur d'un facteur deux.

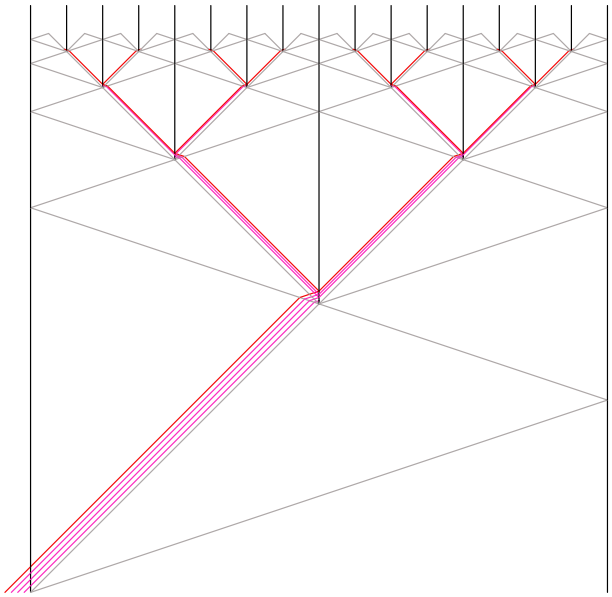


FIGURE 8.1 – Approximation générique de l'arbre fractale par la machine $\mathfrak{M}^{gen}_{tree}$.

Le TABLEAU 8.1 fournit les méta-signaux et règles nécessaires à cette l'approximation avec utilisation de la lentille. Le fonctionnement de la machine suit le même principe que celui de la machine $\mathfrak{M}^{gen}_{tree}$ donnée au CHAP. 5. Il y a cependant deux différences : les règles sont modifiées pour prendre en compte la lentille, et l'étape d'arrêt est légèrement différente. En effet, la collision entre les signaux $\overrightarrow{\overrightarrow{\mathsf{z}}}$, $\overrightarrow{\mathsf{a}}$ et $\overleftarrow{\mathsf{a}}$ qui permet de transformer ces derniers en versions d'arrêt $\overrightarrow{\mathsf{a}_\circ}$ et $\overleftarrow{\mathsf{a}_\circ}$ s'effectue ici de manière non symétrique.

(a) Méta-signaux et vitesses.

$\overrightarrow{\overrightarrow{\zeta}}$, $\overrightarrow{\overrightarrow{\mathsf{z}}}$	3
$\overrightarrow{\zeta}$, $\overrightarrow{\mathsf{z}}$, $\overrightarrow{\mathsf{a}_\circ}$	1
w, $\mathsf{w}_\circ$	0
$\overleftarrow{\zeta}$, $\overleftarrow{\mathsf{z}}$, $\overleftarrow{\mathsf{a}_\circ}$	-1
$\overleftarrow{\overleftarrow{\zeta}}$, $\overleftarrow{\overleftarrow{\mathsf{z}}}$	-3

(b) Décrémenter le compteur.

$$\{\ \overrightarrow{\overrightarrow{\zeta}}, \mathsf{w}\ \} \to \{\ \mathsf{w}_\circ\ \}$$
$$\{\ \overrightarrow{\overrightarrow{\zeta}}, \mathsf{w}_\circ\ \} \to \{\ \overleftarrow{\zeta}, \mathsf{w}_\circ, \overrightarrow{\zeta}\ \}$$
$$\{\ \overrightarrow{\overrightarrow{\mathsf{z}}}, \mathsf{w}_\circ\ \} \to \{\ \overleftarrow{\mathsf{z}}, \mathsf{w}, \overrightarrow{\mathsf{z}}\ \}$$
$$\{\ \overrightarrow{\overrightarrow{\mathsf{z}}}, \mathsf{w}\ \} \to \{\ \mathsf{w}, \overrightarrow{\overrightarrow{\mathsf{z}}}\ \}$$

(c) Étape d'arrêt.

$$\{\ \overrightarrow{\overrightarrow{\zeta}}, \overleftarrow{\mathsf{a}}\ \} \to \{\ \overleftarrow{\mathsf{a}_\circ}, \overrightarrow{\overrightarrow{\zeta}}\ \}$$
$$\{\ \overrightarrow{\overrightarrow{\mathsf{z}}}, \overleftarrow{\mathsf{a}}\ \} \to \{\ \overleftarrow{\mathsf{a}_\circ}, \overrightarrow{\overrightarrow{\mathsf{z}}}\ \}$$
$$\{\ \overrightarrow{\overrightarrow{\mathsf{z}}}, \overleftarrow{\mathsf{a}_\circ}\ \} \to \{\ \overleftarrow{\mathsf{a}}, \overrightarrow{\overrightarrow{\mathsf{z}}}\ \}$$
$$\{\ \overrightarrow{\overrightarrow{\mathsf{z}}}, \overrightarrow{\mathsf{a}}\ \} \to \{\ \overrightarrow{\mathsf{a}_\circ}\ \}$$
$$\{\ \overrightarrow{\overrightarrow{\mathsf{a}}}, \overleftarrow{\mathsf{a}_\circ}\ \} \to \emptyset$$

TABLEAU 8.1 – Couper l'arbre fractale avec la lentille.

La seule condition de validité à vérifier est que le faisceau défini par `[until(`n`)]` n'empiète pas sur plusieurs branches de l'arbre en même temps, ce qui sera garanti si la collision entre $\overrightarrow{\overrightarrow{\mathsf{z}}}$ et $\overrightarrow{\mathsf{a}}$ se produit avant que $\overrightarrow{\mathsf{a}}$ ne rencontre $\overleftarrow{\overleftarrow{\mathsf{a}}}$. Alors que sans la lentille, cette condition était

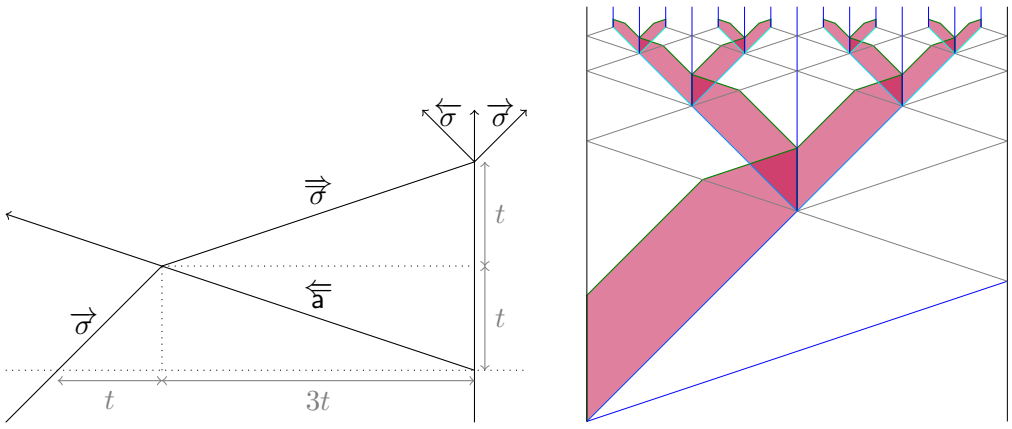
assuré lorsque le faisceau était contenu dans l'intervalle $\left]-\frac{1}{3\times 2^n};0\right]$ (où n est le nombre de niveau souhaité), il suffit ici de placer le faisceau initial dans l'intervalle $]-\frac{1}{3};0[$ (pour les murs délimiteurs en positions 0 et 1). En effet, la largeur du faisceau initial sera donc au plus $\frac{1}{3}$ et comme la lentille divise la largeur du faisceau par deux à chaque niveau, on obtiendra bien au dernier niveau un faisceau de largeur inférieure à $\frac{1}{3\times 2^n}$. La séquence (*i.e.* le type et l'ordre) des signaux initiaux reste quant à elle la même que pour la machine sans lentille, à savoir $(\overrightarrow{\mathsf{z}}\,\overrightarrow{\zeta}^{n-1}\,\mathsf{w}\,\overrightarrow{\mathsf{a}}\,\overrightarrow{\overrightarrow{\mathsf{a}}}\,\mathsf{w})$ pour une approximation de n niveaux.

Pour une formule quantifiée définie sur n variables, chaque variable requiert un niveau pour être assignée en valeur booléenne, et après le niveau n il ne reste plus qu'une formule propositionnelle close qui doit être évaluée. Comme dans le chapitre précédent, cette évaluation nécessite un niveau supplémentaire : $n+1$ niveaux doivent donc être construits. Le premier module inséré est donc `[until(`$n+1$`)]` dont la fonction est de créer exactement les $n+1$ niveaux requis. On ajoute ensuite le module `[decide(`n`)]` afin de configurer chaque niveau en tant que point de décision pour chaque variable. Ce module a juste pour rôle de transformer les signaux verticaux de chaque niveau de l'arbre générique en signaux verticaux servant à assigner les variables, faisant ainsi de l'arbre binaire un arbre binaire de décision. Ce bloc est constitué de manière simple par `[decide(`n`)]` $= \overrightarrow{\alpha}^n$ (un signal $\overrightarrow{\alpha}$ par niveau) et par les règles suivantes : $\{\overrightarrow{\overrightarrow{\alpha}},\mathsf{a}\} \to \{\mathsf{x}\}$ (transformation des signaux stationnaires en signaux assignant les variables) et $\{\overrightarrow{\overrightarrow{\alpha}},\mathsf{x}\} \to \{\overleftarrow{\alpha},\mathsf{x},\overrightarrow{\alpha}\}$ (duplication des signaux $\overrightarrow{\alpha}$ restants pour les prochains niveaux).

8.1.2 Schéma de lentille-duplicatrice avec fonctionnalités.

Afin de garantir que le faisceau se propage correctement dans l'arbre de la même façon qu'en Sous-sec. 7.2.2, nous réutilisons la lentille-duplicatrice de la sous-section précédente, que nous allons appliquer à tous les signaux du faisceau (et non plus uniquement aux signaux permettant de construire l'arbre). La lentille précédemment introduite va être enrichie de nouvelles règles en fonction des différents sous-faisceaux présents dans le faisceau principal.

La Figure 8.2(a) présente un zoom sur l'étape de concentration et illustre le fonctionnement de la lentille, ainsi que le paramètre de rétrécissement (qui vaut ici deux, à cause du choix des vitesses 1 et 3, respectivement pour les signaux du faisceau et pour leurs versions rapides). La Figure 8.2(b) montre l'effet qu'a la lentille sur la propagation d'un faisceau à travers l'arbre dans sa totalité.



(a) Rétrécissement par deux.

(b) Effet sur la propagation.

Figure 8.2 – La lentille-duplicatrice appliquée à la propagation dans l'arbre.

La lentille-duplicatrice est appliquée à la machine générique (qui sera décrite dans le reste

chapitre) par le schéma d'extension suivant :

sauf indications contraires, tout signal $\overrightarrow{\sigma}$ est accéléré par $\overleftarrow{\mathsf{a}}$ et décéléré et dupliqué par tout signal stationnaire s.

Il y a néanmoins quelques cas spéciaux à gérer au moment de la décélération et de la duplication, certains signaux particuliers devant être stoppés à ce moment-là alors que d'autres doivent être changés en de nouveaux signaux comme c'est le cas par exemple des signaux de variables qui doivent devenir des signaux `true` et `false`. Mais dans tous les cas, $\overleftarrow{\mathsf{a}}$ accélère tous les signaux qu'il rencontre. La FIGURE 8.3 zoome sur un niveau particulier — le premier — et illustre l'effet de la lentille sur le faisceau (ainsi que l'assignation de la variable x_1 codée ici par le plus haut signal $\overrightarrow{\mathsf{x}}$, expliqué en SOUS-SEC. 8.2.1). Nous détaillons ci-dessous tous les cas correspondant à la définition de la lentille.

- *Cas général* : pour tout signal stationnaire s (plus exactement, s est parmi w, $\mathsf{a}_\circ$, x, $\mathsf{x}_\circ$, $\exists_L$, $\exists_R$, $\forall_L$ et $\forall_R$) et pour tout signal $\overrightarrow{\mathsf{i}}$ distinct de $\overrightarrow{\zeta}$ et $\overrightarrow{\mathsf{z}}$, on a :

$$\{\ \overrightarrow{\mathsf{i}},\ \overleftarrow{\mathsf{a}}\ \} \to \{\ \overleftarrow{\mathsf{a}},\ \overrightarrow{\mathsf{i}}\ \} \qquad\qquad \{\ \overrightarrow{\mathsf{i}},\ \mathsf{s}\ \} \to \{\ \overleftarrow{\mathsf{i}},\ \mathsf{s},\ \overrightarrow{\mathsf{i}}\ \}$$

- *Cas de $\overrightarrow{\zeta}$ et $\overrightarrow{\mathsf{z}}$* : les règles d'application de la lentille aux signaux $\overrightarrow{\zeta}$ et $\overrightarrow{\mathsf{z}}$ ont déjà été données précédemment (*c.f.* SOUS-SEC. 8.1.1). Dans ce cas de signaux compteurs, le signal stationnaire est w, qui supprime le premier $\overrightarrow{\zeta}$, puis devient $\mathsf{a}_\circ$. Ce signal agit de la manière spécifiée par le schéma de la lentille : il ralentit et dédouble les prochains signaux $\overrightarrow{\zeta}$ et $\overrightarrow{\mathsf{z}}$, puis redevient de nouveau w.
- *Cas de $\overrightarrow{\alpha}$* : de même, les règles correspondant à ce cas ont été précédemment fournies. Ces règles permettent de transformer les signaux stationnaires w en les spécifiant en signaux stationnaires x, dont le rôle sera d'assigner les signaux de variables en signaux booléens, suivant le même principe qu'au chapitre précédent.
- *Cas des quantificateurs* : le premier signal quantificateur — $\overrightarrow{\forall}$ ou $\overrightarrow{\exists}$ — rentrant en collision avec le signal stationnaire x est stoppé et transforme x en un signal stationnaire correspondant — $\forall_D$ ou $\exists_D$, avec $D \in \{R, L\}$. Les règles correspondantes sont :

$$\{\ \mathsf{x},\ \overleftarrow{\exists}\ \} \to \{\ \exists_R\ \} \qquad \{\ \overrightarrow{\exists},\ \mathsf{x}\ \} \to \{\ \exists_L\ \} \qquad \{\ \mathsf{x},\ \overleftarrow{\forall}\ \} \to \{\ \forall_R\ \} \qquad \{\ \overrightarrow{\forall},\ \mathsf{x}\ \} \to \{\ \forall_L\ \}$$

  Les quantificateurs suivants sont décélérés et dupliqués par le nouveau signal stationnaire $\forall_D$ ou $\exists_D$ (dans les règles suivantes, on a $D \in \{R, L\}$) :

$$\{\ \overrightarrow{\exists},\ \exists_D\ \} \to \{\ \overleftarrow{\exists},\ \exists_D,\ \overrightarrow{\exists}\ \} \qquad \{\ \overrightarrow{\exists},\ \forall_D\ \} \to \{\ \overleftarrow{\exists},\ \forall_D,\ \overrightarrow{\exists}\ \}$$
$$\{\ \overrightarrow{\forall},\ \exists_D\ \} \to \{\ \overleftarrow{\forall},\ \exists_D,\ \overrightarrow{\forall}\ \} \qquad \{\ \overrightarrow{\forall},\ \forall_D\ \} \to \{\ \overleftarrow{\forall},\ \forall_D,\ \overrightarrow{\forall}\ \}$$

8.2 Résolution de Q-SAT par une unique machine

Nous présentons ici les différentes étapes de la résolution générique de **Q-SAT** : la représentation des formules, leurs évaluations et la collecte des résultats partiels pour obtenir le résultat final.

8.2.1 Représentation des variables

Nous voulons que la variable x_i soit assignée en valeurs booléennes au niveau i. Cela est réalisé en utilisant une pile de $i-1$ signaux inhibiteurs et en appliquant les règles du TAB. 8.2.

Pour une variable x_i, l'idée est de protéger le signal $\overrightarrow{\mathsf{x}}$ d'être assigné en $\overleftarrow{\mathsf{f}}$ et $\overrightarrow{\mathsf{t}}$ avant qu'il n'atteigne le $i^{\text{ème}}$ niveau. Cela peut être réalisé en codant la variable x_i par une pile de

$i-1$ signaux $\overrightarrow{\overrightarrow{\beta}}$: à chaque niveau, le premier signal $\overrightarrow{\overrightarrow{\beta}}$ change le signal stationnaire x en un signal $\mathsf{x}_\circ$ (la version « non-assignante » de x). Les $\overrightarrow{\overrightarrow{\beta}}$ suivants et le $\overrightarrow{\overrightarrow{\mathsf{x}}}$ de la pile sont simplement dupliqués, $\overrightarrow{\overrightarrow{\mathsf{x}}}$ rechangeant $\mathsf{x}_\circ$ en x. Ainsi, après $i-1$ niveaux, tous les $\overrightarrow{\beta}$ ont disparus. Finalement, le signal $\overrightarrow{\overrightarrow{\mathsf{x}}}$, n'étant plus protégé par des signaux $\overrightarrow{\beta}$, rentre directement en collision avec x (au lieu $\overrightarrow{\beta}$ comme dans les niveaux précédents). Cette collision produit enfin les signaux booléens, conformément aux assignations possibles de x_i : $\overleftarrow{\mathsf{f}}$ est envoyé vers la gauche et $\overrightarrow{\mathsf{t}}$ vers la droite.

La variable x_i sera initialement codée par $\texttt{[var}(x_i)\texttt{]} = \overrightarrow{\mathsf{x}}\overrightarrow{\beta}^{i-1}$. Les deux cas (avec et sans signaux protecteurs $\overrightarrow{\beta}$) sont illustrés dans la FIG. 8.3.

$$\{\overrightarrow{\overrightarrow{\beta}},\ \mathsf{x}\} \to \{\ \mathsf{x}_\circ\ \} \qquad \{\overrightarrow{\overrightarrow{\beta}},\ \mathsf{x}_\circ\} \to \{\ \overleftarrow{\beta},\ \mathsf{x}_\circ,\ \overrightarrow{\beta}\ \}$$
$$\{\overrightarrow{\overrightarrow{\mathsf{x}}},\ \mathsf{x}\} \to \{\ \overleftarrow{\mathsf{f}},\ \mathsf{x},\ \overrightarrow{\mathsf{t}}\ \} \qquad \{\overrightarrow{\overrightarrow{\mathsf{x}}},\ \mathsf{x}_\circ\} \to \{\ \overleftarrow{\mathsf{x}},\ \mathsf{x},\ \overrightarrow{\mathsf{x}}\ \}$$

TABLEAU 8.2 – Assigner de manière générique les variables en booléens.

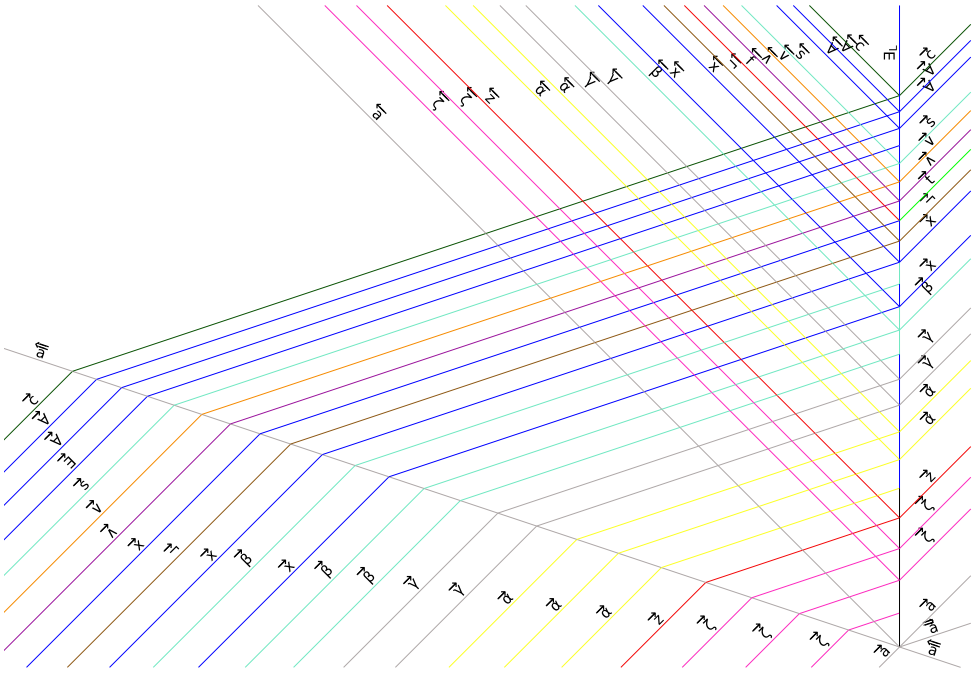


FIGURE 8.3 – Duplication du faisceau au premier niveau et assignation de x_1 en booléens.

8.2.2 Représentation des formules

Pour résoudre **Q-SAT** par une machine à signaux générique, une formule booléenne quantifiée donnée en entrée doit être représentée par une configuration initiale de cette machine à signaux. L'intuition est de compiler cette formule en un faisceau de signaux sous forme polonaise inversée. Chaque type de nœud (connecteurs et variables) de la formule sera représenté par un ou plusieurs méta-signaux (un type de méta-signal pour chaque connecteur, deux type de méta-signaux pour coder les variables selon la SOUS-SEC. 8.2.1). À cela s'ajoutent des méta-signaux qui seront utilisés pour l'évaluation de la formule : les méta-signaux représentant des fonctions booléennes d'évaluation partielle (les fonctions $t()$, $f()$ et $id()$ du chapitre précédent) ; des méta-signaux d'inhibition utilisés pour effectuer l'évaluation de manière générique ; et des versions désactivées des méta-signaux de connecteurs. Le TABLEAU 8.3 résume les différents types et rôles des méta-signaux utilisés pour représenter et évaluer une formule de manière générique.

Type	Rôle
$\wedge$, $\vee$, $\neg$	connecteurs
$\wedge_\circ$, $\vee_\circ$, $\neg_\circ$	connecteurs désactivés
$\wedge_+$	fonction booléenne identité (évaluation partielle de $\wedge$)
f_+	fonction booléenne unaire fausse (évaluation partielle de $\wedge$)
$\vee_+$	fonction booléenne identité (évaluation partielle de $\vee$)
t_+	fonction booléenne unaire vraie (évaluation partielle de $\vee$)
γ, γ_+	inhibiteurs

TABLEAU 8.3 – Méta-signaux permettant de représenter génériquement une formule booléenne.

Au niveau $n+1$, toutes les variables ont été assignées, et sont devenues des signaux $\overrightarrow{\mathsf{t}}$ ou $\overrightarrow{\mathsf{f}}$. La formule propositionnelle close — sans variables — ainsi obtenue peut être vue comme un arbre syntaxique et sa valeur de vérité peut être calculée par une approche *bottom-up* : un signal résultant de l'évaluation d'une sous-expression est envoyé de façon à interagir avec son connecteur parent.

Une formule est représentée par un faisceau de signaux, chaque sous-formule étant elle-même codée par un sous-faisceau de signaux contigus. Lorsqu'une sous-formule arrive au niveau $n+1$, elle initie son évaluation en percutant le signal stationnaire a. Une fois calculée, la valeur de vérité de cette sous-formule est réfléchie de manière à interagir avec le prochain signal arrivant et correspondant à son connecteur parent.

Dans le cas des connecteurs binaires, son premier argument — un signal booléen — arrive avant, est réfléchi contre le signal vertical a et devient un signal booléen activé prêt à interagir avec un signal connecteur arrivant en sens opposé. Pour atteindre son signal connecteur parent, il doit d'abord éventuellement traverser une partie du faisceau codant d'autres sous-expressions et ne doit par interagir avec ces signaux, mais les traverser afin d'atteindre le signal correspondant à son connecteur parent. Pour cette raison, à chaque sous-expression est associé un faisceau $\overrightarrow{\gamma}^k$ d'inhibiteurs qui empêche la valeur de vérité résultante d'interagir avec les k premiers connecteurs rencontrés et permet ainsi au résultat de l'évaluation d'une sous-expression d'interagir avec son connecteur parent même s'il n'est pas disposé immédiatement au-dessus de lui.

Nous notons $C[\![\psi]\!]$ le résultat de la compilation de ψ en un bloc de signaux parallèles, qui sera intégré à la configuration initiale, et on note $\| \psi \|$ le nombre d'occurrences de connecteurs dans la formule ψ.

Le schéma de compilation suivant produit un faisceau de signaux dont la taille — le nombre de signaux — est au plus quadratique en la taille t de la formule d'entrée. Clairement, pour chaque nœud (*i.e.* symbole) de la formule, les signaux inhibiteurs $\overrightarrow{\gamma}$ produits sont en plus en nombre linéaire en t (car il y a moins d'un signal $\overrightarrow{\gamma}$ par symbole de connecteur de la formule). La configuration $C[\![\psi]\!]$ est donc composé d'au plus $\mathcal{O}(t{\cdot}t) = \mathcal{O}(t^2)$ signaux. La compilation est réalisée par induction sur la structure de la formule par le schéma de la FIG. 8.4. La FIGURE 8.5 montre la configuration résultante du schéma de compilation pour l'exemple courant $\varphi = \exists x_1 \forall x_2 \forall x_3\ (x_1 \wedge \neg x_2) \vee x_3$.

8.2.3 Évaluation des formules

Les règles de collisions correspondant à l'étape d'évaluation suivent les règles des connecteurs booléens classiques. Néanmoins, certains signaux spécifiques sont requis — les signaux inhibiteurs $\overrightarrow{\gamma}$ — afin de garantir qu'un connecteur interagit avec la valeur de vérité des sous-formules correspondant bien à ses arguments directs, et non pas avec la valeur booléenne d'une de ses sous-

$$
\begin{array}{lcl}
C[\![\psi]\!] & = & C[\![\psi]\!]^0 \\
C[\![\psi_1 \wedge \psi_2]\!]^k & = & \overrightarrow{\wedge}\ \overrightarrow{\gamma}^k\ C[\![\psi_1]\!]^0\ C[\![\psi_2]\!]^{\|\psi_1\|} \\
C[\![\psi_1 \vee \psi_2]\!]^k & = & \overrightarrow{\vee}\ \overrightarrow{\gamma}^k\ C[\![\psi_1]\!]^0\ C[\![\psi_2]\!]^{\|\psi_1\|} \\
C[\![\neg\psi]\!]^k & = & \overrightarrow{\neg}\ \overrightarrow{\gamma}^k\ C[\![\psi]\!]^0 \\
C[\![x_i]\!]^k & = & \texttt{[var(}x_i\texttt{)]}\ \overrightarrow{\gamma}^k
\end{array}
$$

FIGURE 8.4 – Schéma de compilation générique d'une formule booléenne.

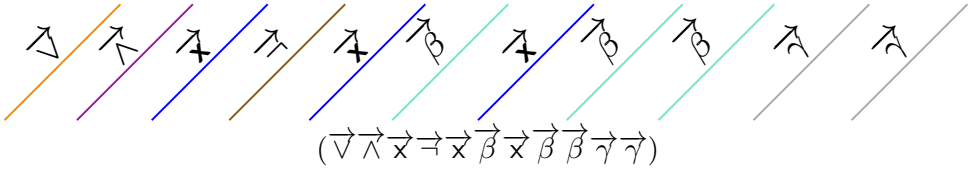


FIGURE 8.5 – Configuration représentant la formule $\varphi = \exists x_1 \forall x_2 \forall x_3\ (x_1 \wedge \neg x_2) \vee x_3$.

formules indirectes. Les signaux $\overrightarrow{\gamma}$ permettent donc à un signal booléen résultat de l'évaluation d'une sous-formule de traverser le reste du faisceau jusqu'au connecteur parent direct de cette sous-formule. Le principe général est illustré par la FIGURE 8.6, qui montre le calcul de $\psi(\mathtt{t},\mathtt{t},\mathtt{t})$ où ψ est la formule des exemples précédents : ceci correspond à l'évaluation de la formule ψ dans le cas $x_1 = x_2 = x_3 = \mathtt{t}$.

Comme pour l'évaluation spécifique du CHAP. 7, lorsque le résultat de l'évaluation d'une sous-formule rencontre le signal stationnaire a au niveau $n+1$, les signaux booléens sont transformés en version *activée* (notées en majuscules) : $\overrightarrow{\mathtt{t}}$ devient $\overleftarrow{\mathsf{T}}$ et $\overrightarrow{\mathtt{f}}$ devient $\overleftarrow{\mathsf{F}}$. Une version activée peut alors interagir avec le premier signal de connecteur rencontré afin de calculer la valeur de vérité de la formule en suivant les règles données dans le TAB. 8.4. Celles-ci suivent les règles booléennes classiques, et utilisent comme au CHAP. 7 des signaux représentant des fonctions booléennes unaires : ce sont les signaux $\wedge_+$, $\mathtt{f}_+$ $\mathtt{t}_+$ (et $\vee_+$ qui est, comme $\wedge_+$, une fonction identité), qui jouent les rôles respectifs des fonctions $id()$, $f()$ et $t()$ de la SOUS-SEC. 7.3.2. Elles correspondent aux connecteurs binaires ayant déjà reçu leur premier argument, dont la valeur conditionne déjà le résultat. Par exemple, une disjonction ayant reçu un premier argument *vrai* sera évaluée comme *vrai*, indépendamment de la valeur du second argument : c'est le rôle du signal $\mathtt{t}_+$. D'autre part, comme nous l'avons déjà expliqué en SOUS-SEC. 8.2.2, des signaux inhibiteurs sont nécessaires pour procéder à une évaluation générique. Ceux-ci permettent de désactivé certains connecteurs afin qu'ils n'interagissent pas avec le prochain signal booléen les rencontrant.

La correction de l'évaluation est assurée par l'ordre des signaux générés par le schéma de compilation de la figure FIG. 8.4 et se montre par induction. Le cas de base d'une formule réduite à une variable est immédiat. Prenons le cas d'une conjonction : soit une formule $\psi = \psi_1 \wedge \psi_2$. La compilation produit dans ce cas la configuration $C[\![\psi_1 \wedge \psi_2]\!]^k = \overrightarrow{\wedge}\ \overrightarrow{\gamma}^k\ C[\![\psi_1]\!]^0\ C[\![\psi_2]\!]^{\|\psi_1\|}$, où $\|\ \psi_1\ \|$ désigne le nombre d'occurrence de symboles de connecteur dans la formule ψ_1. Par hypothèse d'induction, les formules ψ_1 et ψ_2 ont été évaluées correctement et les valeurs de vérité de ces évaluations se présentent sous forme de signaux booléens activés $\overleftarrow{B_1}$ et $\overleftarrow{B_2}$ se propageant depuis le signal stationnaire a vers le faisceau de signaux $\overrightarrow{\wedge}\ \overrightarrow{\gamma}^k$. Le résultat $\overleftarrow{B_2}$ de l'évaluation de ψ_2 est accompagné d'un faisceau de $\|\ \psi_1\ \|$ signaux inhibiteurs $\overleftarrow{\gamma}$, qui se trouvent sous le signal booléen d'après les schémas de la FIG. 8.4 : ces derniers traversent donc le faisceau de la formule ψ_1 avant le signal $\overleftarrow{B_2}$. Chaque signal inhibiteur $\overleftarrow{\gamma}$ est désactive alors exactement un signal de connecteur de ψ_1, qui l'annihile. Le signal $\overleftarrow{\mathtt{b}}$ traverse ainsi le faisceau de la formule ψ_1 sans interagir avec les connecteurs, et atteint ensuite le signal $\overrightarrow{\wedge}$ (après avoir traversé le faisceau $\overrightarrow{\gamma}^k$ avec lequel les signaux booléens n'interagissent pas). Le résultat de l'évaluation de ψ_1 atteint quant à lui

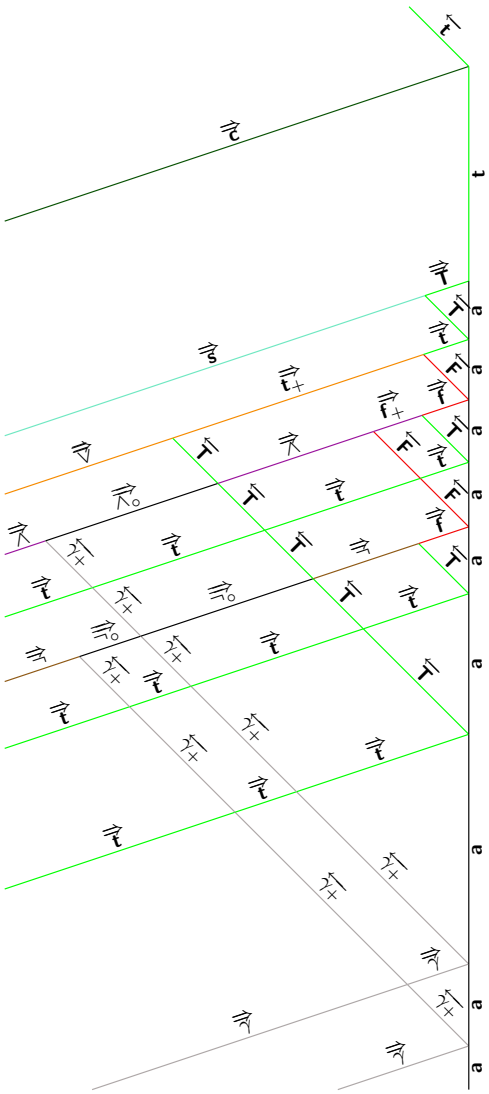


FIGURE 8.6 – Évaluation de la formule $\exists x_1 \forall x_2 \forall x_3\ (x_1 \wedge \neg x_2) \vee x_3$ dans le cas $x_1 = x_2 = x_3 = \mathtt{t}$.

directement le connecteur $\overrightarrow{\wedge}$: celui-ci reçoit alors bien les deux signaux $\overleftarrow{B_1}$ et $\overleftarrow{B_2}$, résultats de l'évaluation des sous-formules ψ_1 et ψ_2. L'évaluation de $B_1 \wedge B_2$ est alors effectuée en suivant les règles du TAB. 8.4(b). Le résultat est ensuite activé en rebondissant sur `a`, et se propage ensuite précédé du faisceau d'inhibiteurs $\overleftarrow{\gamma}^k$, à la rencontre des faisceaux de sous-formules non encore évaluées.

Sauvegarde et mise en place des résultats Afin de rendre les résultats des évaluations facilement exploitables lors de l'étape suivante — la phase de collecte — le résultat de l'évaluation est stocké sous forme d'un signal booléen stationnaire à la sortie du niveau $n+1$: il remplace le signal `a`, qui devient `t` ou `f`. Cette opération est effectuée en rajoutant le bloc `[store]` dans la configuration initiale, où `[store]` $= \overrightarrow{\mathtt{s}}$. Les règles de collisions correspondantes sont données par le TAB. 8.5 : elles permettent de projeter une version activée (en majuscule) des booléens sur le signal stationnaire. Le fait que le booléen projeté soit une version activée (et non une version non-activée, notée en minuscule, comme c'était le cas lors de l'évaluation) signifie qu'il constitue le résultat final de l'évaluation : celui remplace alors le signal stationnaire.

(a) Rebonds et activations.

$$\{\ \overrightarrow{\mathsf{t}},\ \mathsf{a}\ \} \to \{\ \overleftarrow{\mathsf{T}},\ \mathsf{a}\ \}$$
$$\{\ \overrightarrow{\mathsf{f}},\ \mathsf{a}\ \} \to \{\ \overleftarrow{\mathsf{F}},\ \mathsf{a}\ \}$$
$$\{\ \overrightarrow{\gamma},\ \mathsf{a}\ \} \to \{\ \overleftarrow{\gamma_+},\ \mathsf{a}\ \}$$

(b) Conjonction.

$$\{\ \overrightarrow{\wedge},\ \overleftarrow{\mathsf{T}}\ \} \to \{\ \overrightarrow{\wedge_+}\ \}$$
$$\{\ \overrightarrow{\mathsf{f}_+},\ \overleftarrow{\mathsf{T}}\ \} \to \{\ \overrightarrow{\mathsf{f}}\ \}$$
$$\{\ \overrightarrow{\wedge_+},\ \overleftarrow{\mathsf{T}}\ \} \to \{\ \overrightarrow{\mathsf{t}}\ \}$$
$$\{\ \overrightarrow{\wedge},\ \overleftarrow{\mathsf{F}}\ \} \to \{\ \overrightarrow{\mathsf{f}_+}\ \}$$
$$\{\ \overrightarrow{\mathsf{f}_+},\ \overleftarrow{\mathsf{F}}\ \} \to \{\ \overrightarrow{\mathsf{f}}\ \}$$
$$\{\ \overrightarrow{\wedge_+},\ \overleftarrow{\mathsf{F}}\ \} \to \{\ \overrightarrow{\mathsf{f}}\ \}$$

(c) Disjonction.

$$\{\ \overrightarrow{\vee},\ \overleftarrow{\mathsf{T}}\ \} \to \{\ \overrightarrow{\mathsf{t}_+}\ \}$$
$$\{\ \overrightarrow{\mathsf{t}_+},\ \overleftarrow{\mathsf{T}}\ \} \to \{\ \overrightarrow{\mathsf{t}}\ \}$$
$$\{\ \overrightarrow{\vee_+},\ \overleftarrow{\mathsf{T}}\ \} \to \{\ \overrightarrow{\mathsf{t}}\ \}$$
$$\{\ \overrightarrow{\vee},\ \overleftarrow{\mathsf{F}}\ \} \to \{\ \overrightarrow{\vee_+}\ \}$$
$$\{\ \overrightarrow{\mathsf{t}_+},\ \overleftarrow{\mathsf{F}}\ \} \to \{\ \overrightarrow{\mathsf{t}}\ \}$$
$$\{\ \overrightarrow{\vee_+},\ \overleftarrow{\mathsf{F}}\ \} \to \{\ \overrightarrow{\mathsf{f}}\ \}$$

(d) Négation.

$$\{\ \overrightarrow{\neg},\ \overleftarrow{\mathsf{T}}\ \} \to \{\ \overrightarrow{\mathsf{f}}\ \}$$
$$\{\ \overrightarrow{\neg},\ \overleftarrow{\mathsf{F}}\ \} \to \{\ \overrightarrow{\mathsf{t}}\ \}$$

(e) Inhibition.

$$\{\ \overrightarrow{\wedge},\ \overleftarrow{\gamma_+}\ \} \to \{\ \overrightarrow{\wedge_\circ}\ \}$$
$$\{\ \overrightarrow{\vee},\ \overleftarrow{\gamma_+}\ \} \to \{\ \overrightarrow{\vee_\circ}\ \}$$
$$\{\ \overrightarrow{\neg},\ \overleftarrow{\gamma_+}\ \} \to \{\ \overrightarrow{\neg_\circ}\ \}$$

(f) Désinhibition.

$$\{\ \overrightarrow{\wedge_\circ},\ \overleftarrow{\mathsf{T}}\ \} \to \{\ \overleftarrow{\mathsf{T}},\ \overrightarrow{\wedge}\ \}$$
$$\{\ \overrightarrow{\vee_\circ},\ \overleftarrow{\mathsf{T}}\ \} \to \{\ \overleftarrow{\mathsf{T}},\ \overrightarrow{\vee}\ \}$$
$$\{\ \overrightarrow{\neg_\circ},\ \overleftarrow{\mathsf{T}}\ \} \to \{\ \overleftarrow{\mathsf{T}},\ \overrightarrow{\neg}\ \}$$
$$\{\ \overrightarrow{\wedge_\circ},\ \overleftarrow{\mathsf{F}}\ \} \to \{\ \overleftarrow{\mathsf{F}},\ \overrightarrow{\wedge}\ \}$$
$$\{\ \overrightarrow{\vee_\circ},\ \overleftarrow{\mathsf{F}}\ \} \to \{\ \overleftarrow{\mathsf{F}},\ \overrightarrow{\vee}\ \}$$
$$\{\ \overrightarrow{\neg_\circ},\ \overleftarrow{\mathsf{F}}\ \} \to \{\ \overleftarrow{\mathsf{F}},\ \overrightarrow{\neg}\ \}$$

TABLEAU 8.4 – Évaluer de manière générique une formule.

$$\{\ \overrightarrow{\mathsf{s}},\ \overleftarrow{\mathsf{T}}\ \} \to \{\ \overrightarrow{\mathsf{T}}\ \} \qquad \{\ \overrightarrow{\mathsf{s}},\ \overleftarrow{\mathsf{F}}\ \} \to \{\ \overrightarrow{\mathsf{F}}\ \} \qquad \{\ \overrightarrow{\mathsf{T}},\ \mathsf{a}\ \} \to \{\ \mathsf{t}\ \} \qquad \{\ \overrightarrow{\mathsf{F}},\ \mathsf{a}\ \} \to \{\ \mathsf{f}\ \}$$

TABLEAU 8.5 – Sauvegarder le résultat.

8.2.4 Collecte et résultat final

Comme expliqué précédemment, les résultats de l'évaluation de chaque cas doivent être regroupés en respectant les quantificateurs. Chaque variable correspondant à un niveau, les résultats seront regroupés deux par deux et pour une variable liée par un quantificateur existentiel (respectivement universel), les résultats des sous-cas correspondant à cette variable seront combinés en utilisant un $\vee$ (respectivement un $\wedge$).

L'étape de collecte est préparée à chaque niveau de l'arbre binaire de décision en transformant à la fin du passage et du dédoublement du faisceau les signaux stationnaires en version quantificateurs. Ces signaux stationnaires codant les quantificateurs doivent également coder la direction dans laquelle envoyer la combinaison logique ($\vee$ ou $\wedge$) des résultats de deux sous-cas. Ainsi, le signal $\exists_L$ représente un $\vee$ — un quantificateur existentiel — dont le résultat sera envoyé vers la gauche (mémorisé par l'indice L). Les règles pour mettre en place les signaux stationnaires des quantificateurs sont données par le TAB. 8.6, et sont les mêmes que celles définies pour appliquer la lentille au cas des quantificateurs. Le bloc de la configuration initiale correspondant à cette étape est donné par $\texttt{[quantif}(Q_1x_1 \cdots Q_nx_n)\texttt{]} = \overrightarrow{Q_n} \ldots \overrightarrow{Q_1}$.

$$\{\ \mathsf{x},\ \overleftarrow{\exists}\} \to \{\ \exists_R\ \} \qquad \{\ \overrightarrow{\exists},\ \mathsf{x}\} \to \{\ \exists_L\ \} \qquad \{\ \mathsf{x},\ \overleftarrow{\forall}\} \to \{\ \forall_R\ \} \qquad \{\ \overrightarrow{\forall},\ \mathsf{x}\} \to \{\ \forall_L\ \}$$

TABLEAU 8.6 – Mettre en place les quantificateurs pour la collecte.

La collecte des résultats est déclenchée par le signal $\overrightarrow{\mathsf{c}}$ et est ensuite réalisée en suivant les règles données dans le TAB. 8.7. Les résultats booléens arrivant de gauche et droite sont alors combinés deux par deux, résultant en un signal booléen correspondant à une disjonction ou une conjonction des deux signaux booléens entrants. Ceci est possible grâce à un signal stationnaire

indiquant le type d'opération à effectuer (une conjonction pour $\forall$ et une disjonction pour $\exists$) et la direction du signal résultant (L pour gauche et R pour droite).

Le processus complet de collecte est illustré en Fig. 8.7 et son déclenchement est codé par le bloc suivant de la configuration initial : `[collect]` $= \overrightarrow{\mathtt{c}}$.

(a) Initier la collecte.	(b) Réaliser une disjonction.	(c) Réaliser une conjonction.
$\{\ \overrightarrow{\overrightarrow{\mathtt{c}}},\ \mathtt{t}\ \} \to \{\ \overleftarrow{\mathtt{t}}\ \}$	$\{\ \overrightarrow{\mathtt{t}},\ \exists_L,\ \overleftarrow{\mathtt{t}}\ \} \to \{\ \overleftarrow{\mathtt{t}}\ \}$	$\{\ \overrightarrow{\mathtt{t}},\ \forall_L,\ \overleftarrow{\mathtt{t}}\ \} \to \{\ \overleftarrow{\mathtt{t}}\ \}$
$\{\ \overrightarrow{\overrightarrow{\mathtt{c}}},\ \mathtt{f}\ \} \to \{\ \overleftarrow{\mathtt{f}}\ \}$	$\{\ \overrightarrow{\mathtt{t}},\ \exists_L,\ \overleftarrow{\mathtt{f}}\ \} \to \{\ \overleftarrow{\mathtt{t}}\ \}$	$\{\ \overrightarrow{\mathtt{t}},\ \forall_L,\ \overleftarrow{\mathtt{f}}\ \} \to \{\ \overleftarrow{\mathtt{f}}\ \}$
	$\{\ \overrightarrow{\mathtt{f}},\ \exists_L,\ \overleftarrow{\mathtt{t}}\ \} \to \{\ \overleftarrow{\mathtt{t}}\ \}$	$\{\ \overrightarrow{\mathtt{f}},\ \forall_L,\ \overleftarrow{\mathtt{t}}\ \} \to \{\ \overleftarrow{\mathtt{f}}\ \}$
	$\{\ \overrightarrow{\mathtt{f}},\ \exists_L,\ \overleftarrow{\mathtt{f}}\ \} \to \{\ \overleftarrow{\mathtt{f}}\ \}$	$\{\ \overrightarrow{\mathtt{f}},\ \forall_L,\ \overleftarrow{\mathtt{f}}\ \} \to \{\ \overleftarrow{\mathtt{f}}\ \}$
	$\{\ \overrightarrow{\mathtt{t}},\ \exists_R,\ \overleftarrow{\mathtt{t}}\ \} \to \{\ \overrightarrow{\mathtt{t}}\ \}$	$\{\ \overrightarrow{\mathtt{t}},\ \forall_R,\ \overrightarrow{\mathtt{t}}\ \} \to \{\ \overrightarrow{\mathtt{t}}\ \}$
	$\{\ \overrightarrow{\mathtt{t}},\ \exists_R,\ \overleftarrow{\mathtt{f}}\ \} \to \{\ \overrightarrow{\mathtt{t}}\ \}$	$\{\ \overrightarrow{\mathtt{t}},\ \forall_R,\ \overrightarrow{\mathtt{f}}\ \} \to \{\ \overrightarrow{\mathtt{f}}\ \}$
	$\{\ \overrightarrow{\mathtt{f}},\ \exists_R,\ \overleftarrow{\mathtt{t}}\ \} \to \{\ \overrightarrow{\mathtt{t}}\ \}$	$\{\ \overrightarrow{\mathtt{f}},\ \forall_R,\ \overrightarrow{\mathtt{t}}\ \} \to \{\ \overrightarrow{\mathtt{f}}\ \}$
	$\{\ \overrightarrow{\mathtt{f}},\ \exists_R,\ \overleftarrow{\mathtt{f}}\ \} \to \{\ \overrightarrow{\mathtt{f}}\ \}$	$\{\ \overrightarrow{\mathtt{f}},\ \forall_R,\ \overrightarrow{\mathtt{f}}\ \} \to \{\ \overrightarrow{\mathtt{f}}\ \}$

Tableau 8.7 – Collecter de manière générique les résultats en respectant les quantificateurs.

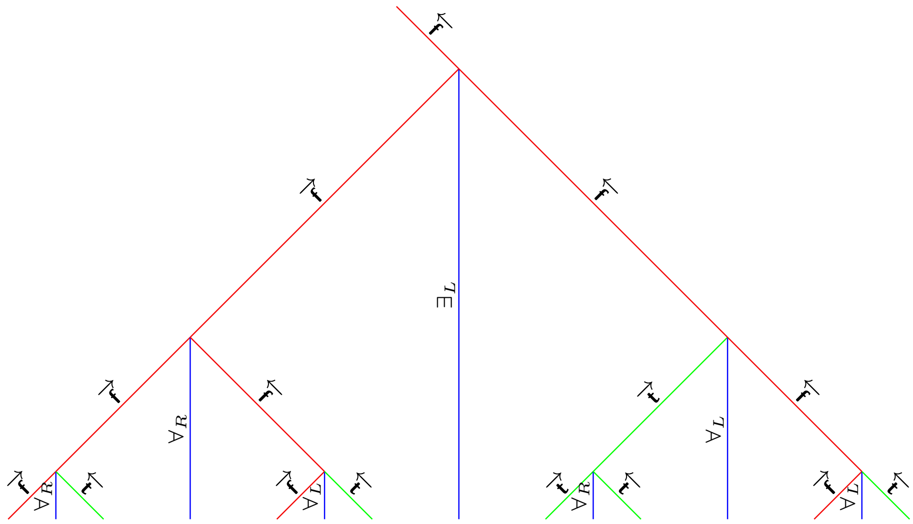


Figure 8.7 – Collecte des résultats.

Diagramme global. Toutes les étapes précédentes mises ensemble constituent le diagramme de la Fig. 8.8 : il représente une exécution de la machine sur la configuration initiale correspondant à la formule $\exists x_1 \forall x_2 \forall x_3\ (x_1 \wedge \neg x_2) \vee x_3$. Cette formule est une instance négative au problème **Q-SAT**, et la réponse finale obtenue se lit sur le diagramme par le signal de type booléen tout en haut de la construction et qui se dirige à gauche : ici, le signal est donc de type `faux`.

La Figure 8.9 propose un zoom sur la configuration initiale du diagramme. Celle-ci s'obtient en concaténant tous les blocs détaillés au cours des sections précédentes. Ainsi,la configuration obtenue s'écrit sous forme de la séquence $(\overrightarrow{\mathtt{c}}\overrightarrow{Q_n}\ldots\overrightarrow{Q_1}\,\overrightarrow{\mathtt{s}}\,C[\psi]\overrightarrow{\alpha}^n\overrightarrow{\mathtt{z}}\,\overrightarrow{\zeta}^n\,\mathtt{w}\ \overrightarrow{\mathtt{a}}\ \overrightarrow{\overrightarrow{\mathtt{a}}}\ \mathtt{w})$ *i.e.* sous forme de blocs : (`[collect][quantif(`$Q_1x_1\cdots Q_nx_n$`)][store]`$C[\psi]$`[decide(`n`)][until(`$n{+}1$`)][tree]`).

Concernant la position des signaux : comme nous avons vu en Sous-sec. 8.1.1, pour garantir la bonne propagation du faisceau, la partie du faisceau générant l'arbre devait être incluse dans l'intervalle $]-\frac{1}{3};0]$. La validité globale sera donc assurée en insérant la configuration ci-dessus

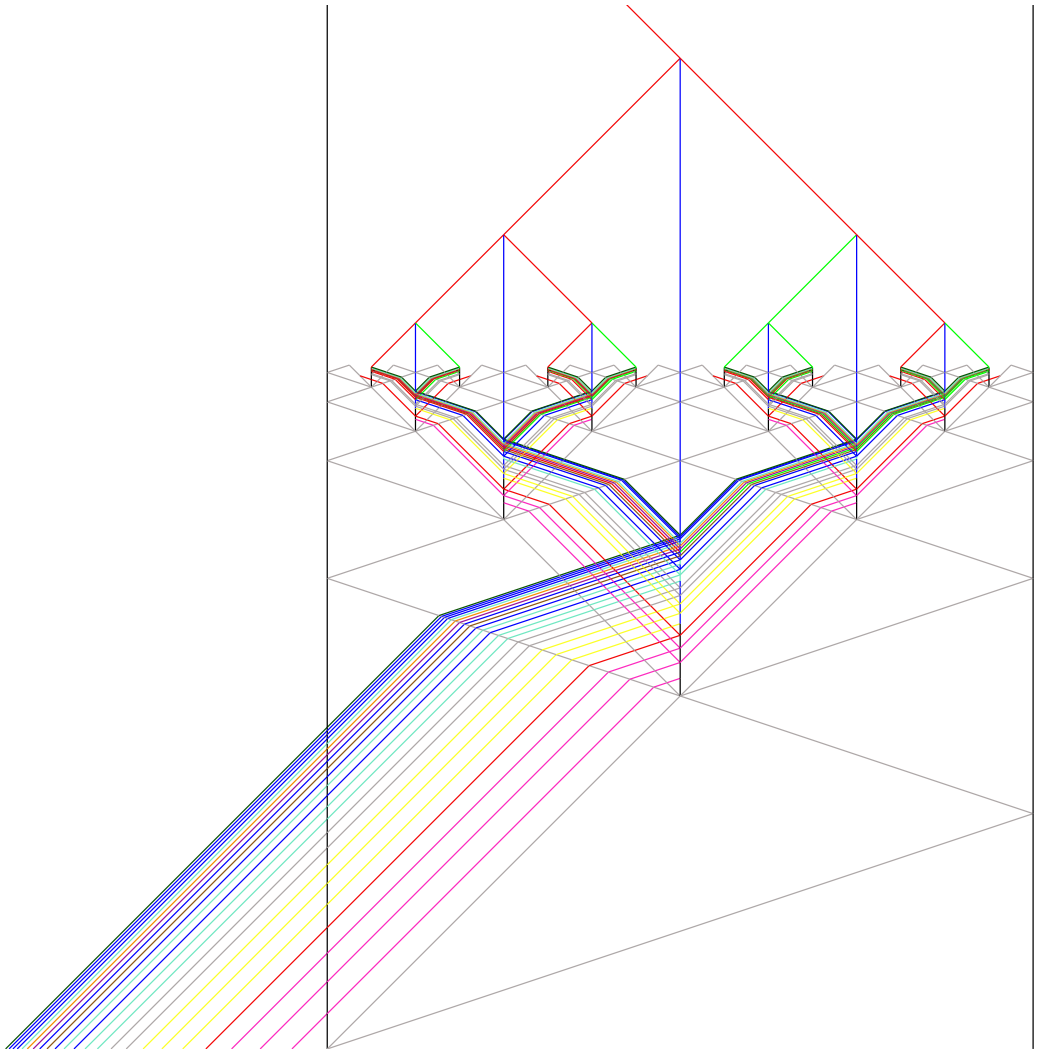


FIGURE 8.8 – Diagramme global de résolution générique de la formule $\exists x_1 \forall x_2 \forall x_3\ (x_1 \wedge \neg x_2) \vee x_3$.

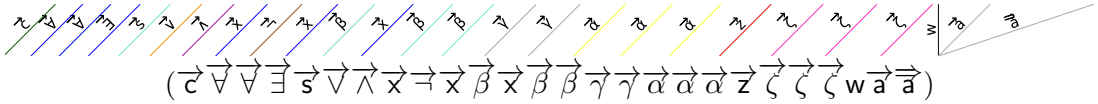


FIGURE 8.9 – Configuration initiale correspondant à la formule $\varphi = \exists x_1 \forall x_2 \forall x_3\ (x_1 \wedge \neg x_2) \vee x_3$ (le signal w présent tout à droite n'est pas représenté ici).

(excepté le dernier signal w qui sera en position 1) dans l'intervalle $]-\frac{1}{3};0]$. À partir de la taille s de la configuration initiale générée, on en déduit qu'en mettant tous les signaux dans l'ordre depuis la gauche aux positions $\frac{-1}{3} + \frac{i}{3s}$ $(1 \leq i \leq s)$. On sait également que s est borné par $t^2 + 3t + 3$, où t désigne la taille de la formule (le terme quadratique provient de la configuration $C[\psi]$, les autres blocs n'ayant qu'un apport linéaire).

Il est également possible de ne pas disposer les signaux du faisceau à intervalles réguliers, du moment que le faisceau est entièrement contenu dans l'intervalle de départ $]-\frac{1}{3};0]$ (en fait, un intervalle plus large convient encore, du moment que le bloc `[until(n)]` reste entre $-\frac{1}{3}$ et 0). C'est ce qui a été fait pour générer un diagramme plus lisible et avoir un faisceau plus espacé en configuration initiale : les signaux deviennent de plus en plus rapprochés vers la gauche du faisceau (chaque signal est à une distance du signal qui le précède à droite, égale à un dixième de la distance totale restant à disposition pour être contenue dans l'intervalle $]\frac{1}{2};0[$).

8.3 Correction et complexités de la construction

Comme nous l'avons déjà mentionné, la construction se décompose en de nombreux modules, qui ont tous déjà été mentionnés au CHAP. 6. Le diagramme précédent peut ainsi s'écrire comme une séquence de compositions parallèles et séquentielles de modules. On en déduit que le calcul correspondant au diagramme est bien celui attendu, en nous basant largement sur les notions et l'étude des exemples faite dans le CHAP. 6.

8.3.1 Preuve de correction

La machine générique résolvant **Q-SAT** est composée de tous les méta-signaux et règles donnés dans les tables de ce chapitre. Elle s'obtient également par union des machines sous-jacentes aux modules correspondant aux différentes étapes : la machine d'approximation générique de l'arbre avec lentille-duplicatrice, la machine de représentation-évaluation de la formule et la machine de jonction de booléens. Les principaux modules sont les modules d'initiation, de propagation, de duplication-concentration du faisceau, d'évaluation et de jonction des résultats. Tous ces modules, mis à part le module d'initiation, sont des R-modules (le module d'initiation étant défini comme un T-module). Ces modules ont été choisis et définis de telle sorte que leur composition parallèle et séquentielle soit naturelle. Chaque étage de la construction est constituée par une composition parallèle de plusieurs modules du même type (c'est en fait même une parallélisation de modules). Les étages sont ensuite empilés suivant des compositions séquentielles. La FIGURE 8.10 est une décomposition en modules du diagramme espace-temps précédent [1] , que nous allons maintenant commenter.

Le module d'initiation consiste juste en la propagation du faisceau de la configuration initiale jusqu'à ce qu'il se retrouve entièrement entre les deux murs w, qui délimitent l'espace qui sera utilisé par l'arbre. Les configurations entrantes et sortantes de ce module sont quasiment les mêmes (à translations près) : seul le signal rapide de construction de l'arbre peut éventuellement être devenu sa version symétrique après le rebond contre le mur de droite. Cela dépend en fait de la largeur initiale du faisceau et donc du temps mis par le faisceau initial pour pénétrer entièrement l'espace entre les deux murs.

Les modules de propagation impliquent deux processus : le faisceau codant la formule se propage suivant l'arbre, et dans un même temps les signaux de construction de l'arbre — plus rapides que le faisceau — se propagent vers les murs puis rebondissent pour revenir vers le faisceau. Ces modules produisent des configurations de sortie contenant le même faisceau qu'en entrée. La fin de fonctionnement de ce module est marquée par la collision entre les signaux de construction de l'arbre (le signal rapide et le signal lent de la base du faisceau). Cette collision produit les signaux et la configuration requise pour l'application de la lentille-duplicatrice.

La version de la lentille-duplicatrice utilisée ici est une version non-nettoyante : les signaux d'accélération et de décélération (également signal de duplication/assignation), continuent de se propager après l'application de la lentille. Cette étape produit ainsi deux configurations symétriques, de même structure que les configurations entrantes : un faisceau et le signal rapide de construction de l'arbre se propageant entre deux signaux w. Le faisceau est cependant légèrement modifié : les piles de signaux compteurs se décrémentent et les signaux de variables correspondant aux niveaux sont assignés en valeurs booléennes. La configuration de sortie du module de concentration permet d'appliquer un module de propagation de manière parallèle et symétrique sur les deux configurations ainsi obtenues.

Après une application alternée de modules de propagation et duplication-concentration et après que les signaux compteurs utilisés pour couper l'arbre fractale au niveau voulu (n niveaux

1. Pour plus de clarté, les proportions du diagramme de la FIG. 8.10 ne sont pas respectées sur le schéma de décomposition (notamment, les modules de propagation ont dans le schéma une durée beaucoup plus grande qu'ils ne l'ont dans le diagramme).

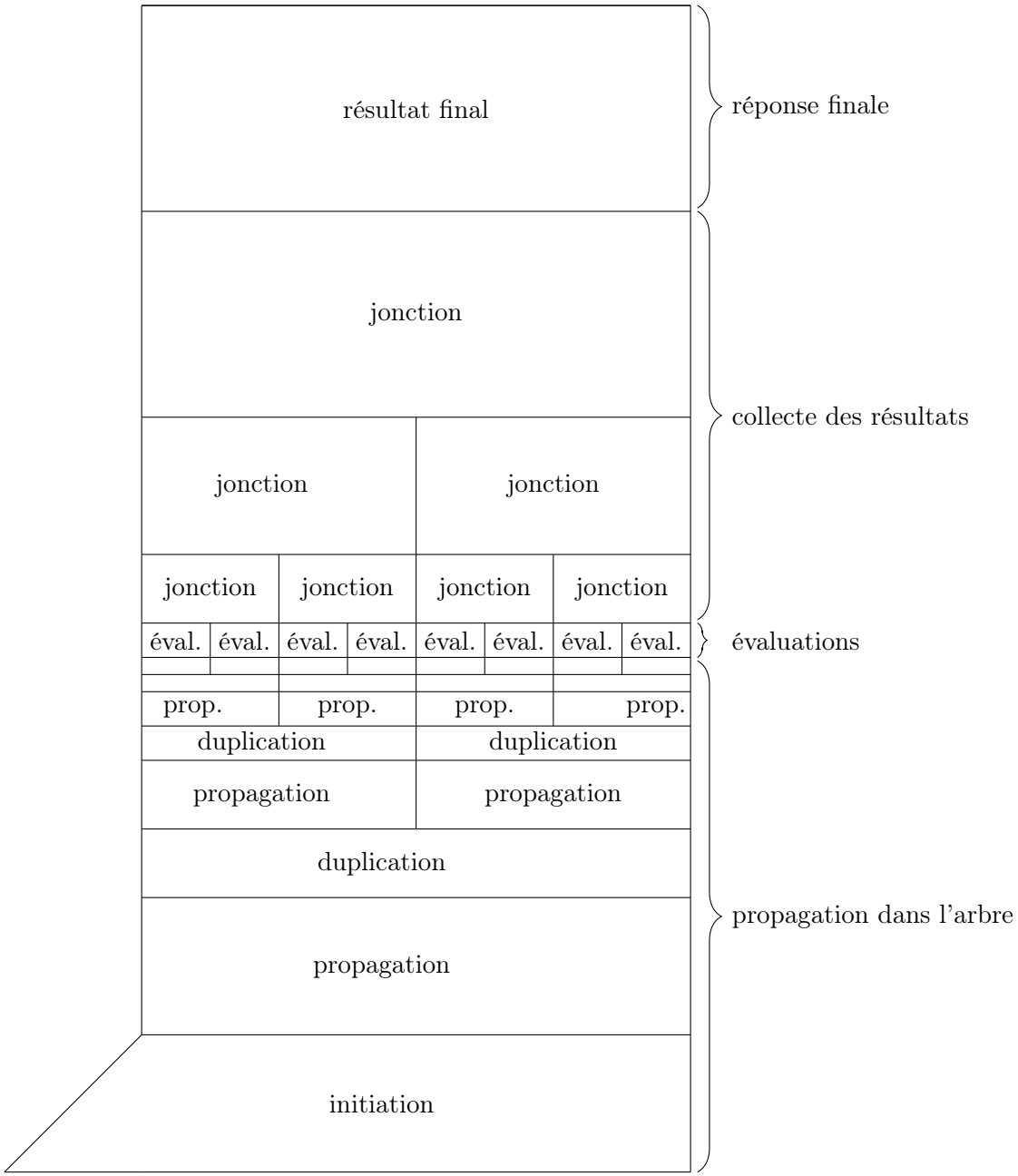


FIGURE 8.10 – Décomposition en modules du diagramme de la FIG. 8.8.

pour n variables) ont tous été épuisés, la construction de l'arbre stoppe, et le processus de lentille-duplication s'arrête. Le processus d'évaluation des formules booléennes s'opère alors. Constatons d'abord qu'aucun des faisceaux présents au dernier niveau de l'arbre ne contient de signaux de variables : seuls persistent dans les signaux de connecteurs logiques, les signaux booléens, les signaux inhibiteurs et les signaux déclencheurs de la collecte. En effet, les variables sont représentées par des piles contenant un signal de type `x` et au plus $n-1$ signaux β, et comme un signal β disparaît à chaque niveau et que le signal `x` est assigné en signal booléen quand il ne reste plus que lui dans la pile, il s'ensuit que ces signaux ont tous disparus après n niveaux, ne laissant à leur place que des signaux de type `t` ou `f`.

Les modules d'évaluations sont composés parallèlement comme vu dans l'EX. 8. Chacun de ces modules produit un seul signal booléen correspondant au résultat de l'évaluation pour une assignation des variables. La configuration-jonction de tous les signaux booléens fournit une

configuration permettant d'appliquer des modules de jonction de booléens deux-à-deux comme vu en Ex. 10. En composant ainsi successivement de tels modules de jonctions de manière parallèle, puis séquentielle, on obtient une configuration finale ne contenant qu'un seul signal booléen entre les deux signaux murs : ce signal représente le résultat final.

Soient *distrib* et *collect* les modules respectivement définis par :

$$collect = answer \circ (jonc \circ (\|^2 jonc \ldots ((\|^{2^{n-2}} jonc) \circ (\|^{2^{n-1}} jonc)) \ldots)) \text{ et}$$

$$distrib = (\|^{2^n}(prop \circ dup)) \circ (\ldots \circ (\|^2(prop \circ dup) \circ (prop \circ dup)))$$

où $\circ$ et $\|^k$ désignent respectivement la composition séquentielle de deux modules et la parallélisation de k fois le même module. Le diagramme se décompose alors de la façon suivante :

$$collect \circ (\|^{2^n} eval \circ (distrib \circ (prop \circ init))) \ .$$

Il est fastidieux mais aisé de vérifier que chaque configuration de sortie d'un module correspond bien à une configuration d'entrée valide pour le module suivant avec lequel il est composé séquentiellement (le cas des compositions parallèles étant immédiat). En respectant les interprétations de modules ainsi que les interprétations des compositions, on obtient ainsi que la fonction calculée par le module global résultant est la fonction $\mathcal{E}val : \{\varphi \text{ formule booléenne quantifiée}\} \mapsto \{\mathsf{vrai}, \mathsf{faux}\}$, qui fournit la valeur de vérité d'une formule booléenne quantifiée.

8.3.2 Analyse des complexités

Soit t la taille de la formule et n le nombre de variables. La largeur de collisions — complexité en espace — est alors exponentielle. En effet, au moment de l'évaluation, 2^n calculs indépendants sont exécutés en parallèle, chacun impliquant au plus t^2 signaux. Le nombre de signaux existant simultanément est donc en $\mathcal{O}(t^2 \cdot 2^n)$.

Concernant la profondeur de collisions — complexité en temps — il faut tout d'abord remarquer que le processus de compilation génère une configuration initiale contenant au plus $\mathcal{O}(t^2)$ signaux. En effet, pour chaque sous-formule, un nombre de signaux au plus linéaire en t est ajouté, et t étant la taille de la formule, on obtient que le module codant la formule contient au plus $\mathcal{O}(t^2)$ signaux. Les autres modules contenant au plus un nombre linéaire de signaux, la taille de la configuration initiale est donc bien en $\mathcal{O}(t^2)$.

La première contribution au nombre de collisions suivant un chemin ascendant provient du croisement entre les faisceaux dupliqué et incident au moment de la duplication. Un signal croisera donc l'intégralité du faisceau réfléchi, donnant lieu à $\mathcal{O}(t^2)$ collisions blanches. Comme il y a n niveaux, le nombre total de collisions jusqu'à l'étape d'évaluation sera de $\mathcal{O}(n \cdot t^2)$, c'est-à-dire cubique en la taille de la formule. Les étapes suivantes (évaluation et collecte) n'apporte qu'un nombre au plus quadratique de collisions. On obtient finalement une profondeur de collisions qui est cubique en la taille de la formule.

Ces faits découlent directement des résultats de Sous-sec. 6.4 : le Lem. 28 affirme en effet que la profondeur (resp. largeur) de collision d'une composition parallèle est donnée par le maximum (resp. par la somme) des profondeurs des modules composés ; le Lem. 29 énonce que dans le cas d'une composition séquentielle, la somme des profondeurs (resp. largeurs) borne la profondeur (resp. la largeur) du module global.

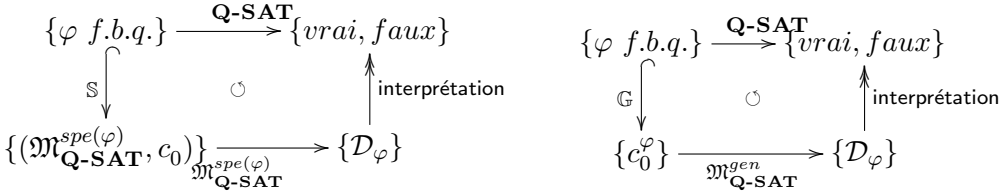
Ici, la largeur de collision est donnée par la composition parallèle des 2^n modules d'évaluations, chacun ayant une largeur de collision en $\mathcal{O}(t^2)$: on retrouve bien le résultat $\mathcal{O}(t^2 \cdot 2^n)$.

En appliquant le lemme à la profondeur de collisions des compositions séquentielles successives des modules de propagation et duplication, on retrouve bien une profondeur de collision en $\mathcal{O}(t^3)$. En effet, le faisceau ayant un nombre de signaux quadratique en la taille de la formule, et la duplication des faisceaux ajoutant à la profondeur de collision un nombre linéaire en la taille du faisceau, on obtient en sommant sur les n niveaux une profondeur de collision globale en $\mathcal{O}(n \cdot t^2)$ *i.e.* en $\mathcal{O}(t^3)$.

8.3.3 Différences entre les versions spécifique et générique

La machine générique présentée ici résout uniquement le problème **Q-SAT**, mais elle permet aussi de résoudre facilement le problème **SAT** de manière générique. En effet, une instance du problème **SAT** est une formule propositionnelle sans quantificateur dont on veut décider si elle est satisfaisable ou non. Une telle instance peut être vue comme une instance particulière du problème **Q-SAT**, dont tous les quantificateurs sont existentiels.

Diagrammes commutatifs de résolution. Pour conclure la présentation des résolutions géométriques du problème **Q-SAT**, nous résumons les deux approches — spécifique et générique — par les diagrammes commutatifs correspondant. On note $\mathbb{S}$ et $\mathbb{G}$ les algorithmes de compilation d'une formule φ respectivement une machine spécifique $\mathfrak{M}^{spe(\varphi)}_{\textbf{Q-SAT}}$ (avec la configuration associée c_0, qui ne dépend pas de φ, comme vu au Chap. 7) et une configuration générique c_0^{φ}. C'est-à-dire que l'on a $\mathbb{S}(\varphi) = (\mathfrak{M}^{spe(\varphi)}_{\textbf{Q-SAT}}, Config[0])$ et $\mathbb{G}(\varphi) = c_0^{\varphi}$. L'algorithme $\mathbb{S}$ s'obtient par une composition séquentielle des algorithmes présentés au Chap. 7, et l'algorithme $\mathbb{G}$ se déduit surtout du schéma de compilation de la Fig. 8.4 (les autres blocs contribuant à la configuration étant simples). D'après les chapitres précédents, on a bien pour toute formule booléenne quantifiée (*f.b.q.*) φ $\textbf{Q-SAT}(\varphi) = \mathsf{interprétation}(\mathfrak{M}^{spe(\varphi)}_{\textbf{Q-SAT}}(c))$ et $\textbf{Q-SAT}(\varphi) = \mathsf{interprétation}(\mathfrak{M}^{gen}_{\textbf{Q-SAT}}(c_0^{\varphi}))$. Les diagrammes commutatifs donnés par la Fig. 8.11, illustrent le fait que les méthodes présentées ici constituent bien des résolution géométriques au sens des Déf. 24 et 25. Les fonctions de représentation des entrées étant fournies par les algorithmes $\mathbb{S}$ et $\mathbb{G}$, celles-ci sont calculables. Elles sont de plus injectives. De même, les fonctions d'interprétation sont calculables (ils suffit de calculer la valeur du signal booléen dans la configuration finale, qui est rationnelle) et surjectives.



(a) Résolution spécifique. (b) Résolution générique.

Figure 8.11 – Les deux résolution du problème **Q-SAT** sur machines à signaux.

Tailles des machines et complexités. Alors que pour la résolution spécifique, la taille de la machine dépendait de manière quadratique de la taille de la formule, la machine générique a quant à elle une taille fixe par définition. La machine présentée dans ce chapitre a été implémentée par exactement 121 méta-signaux et 196 règles de collisions (non blanches). Cette implémentation n'est pas optimale : certains méta-signaux peuvent être utilisés pour plusieurs rôles différents, mais cela se ferait au détriment d'une séparation nette entre les différents modules.

Les configurations initiale différent également très nettement : alors que dans le cas spécifique, la configuration est fixe (5 signaux initiaux pour la résolution spécifique avec calibrage du faisceau par la première méthode), celle de la machine générique dépend de la formule et sa taille est au plus quadratique en la taille de l'instance d'entrée.

La profondeur de collision prend un degré polynomial (de quadratique à cubique) lorsqu'on passe de la version non-générique à la version générique. La configuration initiale générique est plus complexe que dans le cas spécifique, mais elle est robuste et sa compilation reste simple.

Concernant ce problème sur machines à signaux, le prix de la généricité peut donc être évalué à un degré polynomial.

Chapitre 9

Variantes autour du problème SAT

La machine à signaux générique présentée au Chap. 8 résolvant le problème **Q-SAT** permet également de résoudre facilement le problème **SAT** de manière générique : une instance du problème **SAT** peut être vue comme une instance particulière du problème **Q-SAT** dont tous les quantificateurs sont existentiels. Nous présentons ici comment cette machine peut être modifiée afin de résoudre, toujours de manière générique, d'autres types de problèmes de satisfaisabilité booléenne : #**SAT**, **All-SAT** et **Max-SAT**. Ces problèmes sont respectivement des problèmes de comptage, énumération et optimisation. Les constructions présentées dans ce chapitre illustrent la capacité des machines à signaux à résoudre efficacement des problèmes autres que les problèmes de décisions, souvent les seuls considérés sur la plupart des modèles non-conventionnels de calculs.

Comme les constructions présentées dans ce chapitre sont des variantes de la construction du Chap. 7, elles sont toutes basées sur l'arbre binaire fractale, utilisé pour décider les variables et opérer les évaluations d'une formule booléenne pour toutes les assignations de variables possibles. Grâce à l'approche modulaire, nous pouvons facilement modifier la machine résolvant **Q-SAT** pour obtenir les machines résolvant les problèmes considérés ici. L'intuition est d'ajouter après l'étape d'évaluation de nouveaux modules, correspondant aux diverses fonctions requises pour résoudre les problèmes considérés. La facilité avec laquelle ces nouvelles machines sont obtenues à partir de celle donnée au Chap. 8 illustre la puissance de l'approche modulaire.

Nous commencerons par traiter le cas de #**SAT**, connu pour être un problème difficile de comptage. Nous proposerons ensuite des solutions géométriques aux problèmes **One-SAT** et **All-SAT**, qui sont des problèmes respectivement de recherche de certificat et d'énumération. Enfin, nous finirons en discutant du problème d'optimisation **Max-SAT** sur machines à signaux.

9.1 Le problème #SAT

Le problème #**SAT** est défini comme le problème de comptage du nombre de solutions du problème **SAT** c'est-à-dire qu'étant donnée une formule propositionnelle φ, on souhaite calculer le nombre d'assignations de variables satisfaisant φ :

Problème #**SAT**
| *entrée* : une formule propositionnelle $\varphi(x_1, \ldots, x_n)$.
| *sortie* : le nombre d'assignations de $(x_1, \ldots, x_n)$ satisfaisant φ.

On rappelle que ce problème est complet pour la classe de complexité #**P**, définie comme la classe des problèmes **NP** dont les solutions peuvent être comptées en temps polynomial par une machine de Turing déterministe. On renvoie à [Papadimitriou 1994, Th. 18.1, p. 442] pour une preuve de #**P**-complétude et pour plus de détails et résultats sur la classe #**P**.

Pour résoudre #**SAT** sur machine à signaux, le principe est de réutiliser la machine générique permettant de résoudre **SAT**, que l'on modifie en ajoutant un module qui compte le nombre de signaux t (*i.e.* les booléens *vrai*) au moment de la collecte des résultats. Comme chaque évaluation qui produit un signal t correspond à une assignation des variables qui satisfait la formule, cette somme est la valeur recherchée.

L'additionneur binaire modifié. Le décompte des solutions est réalisé grâce à une variante du module additionneur binaire formalisé dans l'Ex. 5. Nous préférons utiliser ici une variante de ce module dans laquelle l'addition de deux bits s'effectue en une seule collision, comme illustré par la Fig. 9.1 qui donne le diagramme du calcul binaire de 3 et 5 par cette version de l'additionneur. Nous définissons pour cela des règles de triples collisions, données par le Tab. 9.1. Les règles sont données seulement pour la version envoyant le résultat à droite. La direction du résultat est indiquée par l'exposant R dans le nom des méta-signaux concernés, et les règles pour l'exposant L (résultat envoyé à gauche) sont similaires mais avec un signal sortant de vitesse -1. Les principales différences avec l'additionneur donné au Chap. 3 est l'utilisation de signaux verticaux pour reporter la retenue, et l'utilisation d'un signal $\mathsf{e_{nd}}$ pour décaler le signal $\overrightarrow{\mathsf{end}}$ afin d'ajouter un signal supplémentaire au faisceau dans le cas de la création d'un nouveau bit, comme pour l'exemple de la Fig. 9.1.

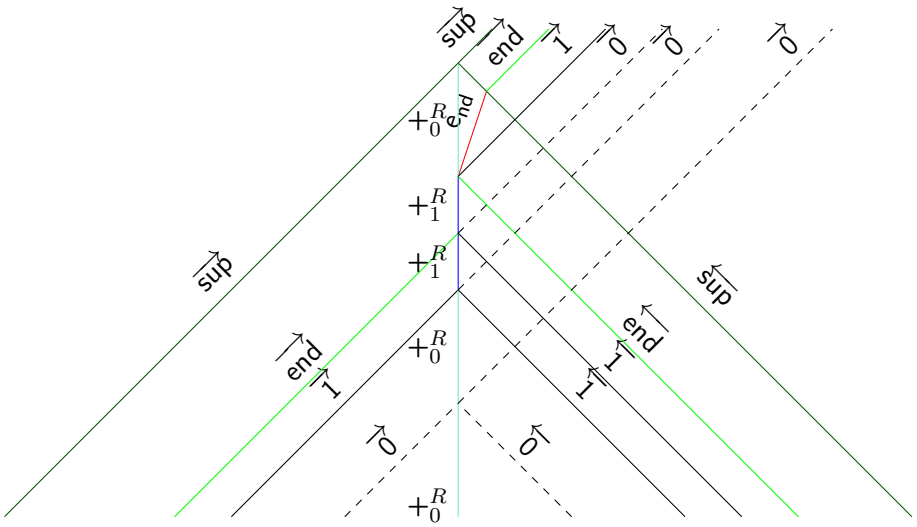


Figure 9.1 – Calcul de $2 + 6$ avec l'additionneur binaire modifié.

L'additionneur binaire ainsi obtenu n'est pas robuste car les deux faisceaux doivent être en partie symétrique au niveau des positions, de telle sorte que les collisions triples se réalisent et avec deux signaux correspondant au même rang de bits. Néanmoins, cet additionneur est adapté à la construction considérée ici, dans laquelle la propagation assure la symétrie de positions des faisceaux dupliqués. Il permet notamment d'assurer une meilleure synchronisation de toutes les additions effectuées en parallèles, les triples collisions de chaque niveau de la collecte étant simultanées.

Dénombrement des assignations. L'étape de collecte des résultats nécessite la mise en place des signaux verticaux $+_0^R$ et $+_0^L$ déclencheurs des additions à chaque étage de la collecte. Ces signaux sont créés à chaque niveau de l'arbre, au moment de la duplication du faisceau, grâce aux règles $\{\overrightarrow{+}, \mathsf{x}\} \to \{+_0^L\}$ et $\{\mathsf{x}, \overleftarrow{+}\} \to \{+_0^R\}$ (où le méta-signal x est le même que celui utilisé au Chap. 8). Une pile de n signaux $\overrightarrow{+}$ (un par niveau) doit alors être ajoutée au faisceau initial, contribuant ainsi à la configuration initiale par la sous-configuration $(\overrightarrow{+}^n)$.

(a) Méta-signaux et vitesses.

$\overrightarrow{0}$, $\overrightarrow{1}$, $\overrightarrow{\mathsf{end}}$, $\overrightarrow{\mathsf{sup}}$	1
$\overleftarrow{0}$, $\overleftarrow{1}$, $\overleftarrow{\mathsf{end}}$, $\overleftarrow{\mathsf{sup}}$	-1
$+_0^R$, $+_1^R$	0
$\mathsf{e_{nd}}$	1/3

(b) Addition simultanée de bits.

$\{\ \overrightarrow{0},+_0^R,\overleftarrow{0}\ \} \to \{\ +_0^R,\overrightarrow{0}\ \}$
$\{\ \overrightarrow{0},+_1^R,\overleftarrow{0}\ \} \to \{\ +_0^R,\overrightarrow{1}\ \}$
$\{\ \overrightarrow{1},+_0^R,\overleftarrow{0}\ \} \to \{\ +_0^R,\overrightarrow{1}\ \}$
$\{\ \overrightarrow{1},+_1^R,\overleftarrow{0}\ \} \to \{\ +_1^R,\overrightarrow{0}\ \}$
$\{\ \overrightarrow{0},+_0^R,\overleftarrow{1}\ \} \to \{\ +_0^R,\overrightarrow{1}\ \}$
$\{\ \overrightarrow{0},+_1^R,\overleftarrow{1}\ \} \to \{\ +_1^R,\overrightarrow{0}\ \}$
$\{\ \overrightarrow{1},+_0^R,\overleftarrow{1}\ \} \to \{\ +_1^R,\overrightarrow{0}\ \}$
$\{\ \overrightarrow{1},+_1^R,\overleftarrow{1}\ \} \to \{\ +_1^R,\overrightarrow{1}\ \}$

(c) Fin d'un faisceau binaire.

$\{\ \overrightarrow{0},+_0^R,\overleftarrow{\mathsf{end}}\ \} \to \{\ +_0^R,\overrightarrow{0}\ \}$
$\{\ \overrightarrow{0},+_1^R,\overleftarrow{\mathsf{end}}\ \} \to \{\ +_0^R,\overrightarrow{1}\ \}$
$\{\ \overrightarrow{1},+_0^R,\overleftarrow{\mathsf{end}}\ \} \to \{\ +_0^R,\overrightarrow{1}\ \}$
$\{\ \overrightarrow{1},+_1^R,\overleftarrow{\mathsf{end}}\ \} \to \{\ +_1^R,\overrightarrow{0}\ \}$

(d) Gestion des bits supplémentaires seuls.

$\{\ \overrightarrow{0},+_0^R\ \} \to \{\ +_0^R,\overrightarrow{0}\ \}$
$\{\ \overrightarrow{0},+_1^R\ \} \to \{\ +_0^R,\overrightarrow{1}\ \}$
$\{\ \overrightarrow{1},+_0^R\ \} \to \{\ +_0^R,\overrightarrow{1}\ \}$
$\{\ \overrightarrow{1},+_1^R\ \} \to \{\ +_1^R,\overrightarrow{0}\ \}$

(e) Gestion du dernier bit.

$\{\ \overrightarrow{\mathsf{end}},+_0^R,\overleftarrow{\mathsf{end}}\ \} \to \{\ +_0^R,\overrightarrow{\mathsf{end}}\ \}$
$\{\ \overrightarrow{\mathsf{end}},+_1^R,\overleftarrow{\mathsf{end}}\ \} \to \{\ +_0^R,\overrightarrow{1},\mathsf{e_{nd}}\ \}$
$\{\ \overrightarrow{\mathsf{end}},+_0^R\ \} \to \{\ +_0^R,\overrightarrow{\mathsf{end}}\ \}$
$\{\ \overrightarrow{\mathsf{end}},+_1^R\ \} \to \{\ +_0^R,\overrightarrow{1},\mathsf{e_{nd}}\ \}$

(f) Fin de fusion des faisceaux.

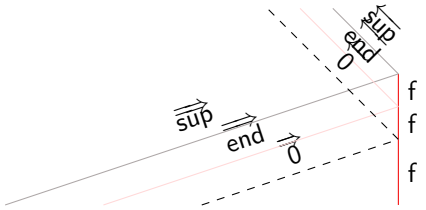
$\{\ \mathsf{e_{nd}},\overleftarrow{\mathsf{sup}}\ \} \to \{\ \overleftarrow{\mathsf{sup}},\overrightarrow{\mathsf{end}}\ \}$
$\{\ \overrightarrow{\mathsf{sup}},+_0^R,\overleftarrow{\mathsf{sup}}\ \} \to \{\ \overrightarrow{\mathsf{sup}}\ \}$

Tableau 9.1 – La machine $\mathfrak{M}_{add}^{modif}$: méta-signaux et règles.

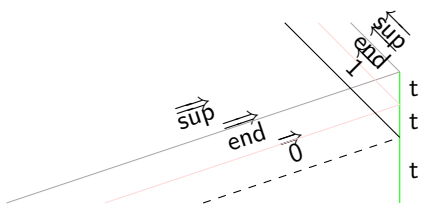
L'étape de collecte est déclenchée par les signaux stationnaires $+_0^R$ et $+_0^L$, mais les faisceaux binaires doivent auparavant être initialisés. Juste après l'évaluation des formules, les seules interprétations possibles des faisceaux binaires sont 0 et 1 (selon si l'assignation satisfait ou non la formule). La mise en place de ces faisceaux binaires se fait grâce à l'ajout de la sous-configuration ($\overrightarrow{\mathsf{sup}}\ \overrightarrow{\mathsf{end}}\ \overrightarrow{0}$) à la fin du faisceau initial, et aux règles du Tab. 9.2, qui permettent d'incrémenter le faisceau binaire (de valeur initiale 0) de 1 si la valeur obtenue dans le cas d'une évaluation est t. La Figure 9.2 illustre les deux cas possibles. L'étape de collecte est ensuite réalisée grâce aux additions binaires (par la machine $\mathfrak{M}_{add}^{modif}$) en parallèle, puis niveau par niveau, de tous les faisceaux ainsi produits.

$\{\ \overrightarrow{\overrightarrow{\mathsf{end}}},\ \mathsf{t}\ \} \to \{\ \overleftarrow{\mathsf{end}}\ \}$ $\quad$ $\{\ \overrightarrow{\overrightarrow{\mathsf{sup}}},\ \mathsf{t}\ \} \to \{\ \overleftarrow{\mathsf{sup}}\ \}$ $\quad$ $\{\ \overrightarrow{\overrightarrow{0}},\ \mathsf{t}\ \} \to \{\ \overleftarrow{1}\ \}$
$\{\ \overrightarrow{\overrightarrow{\mathsf{end}}},\ \mathsf{f}\ \} \to \{\ \overleftarrow{\mathsf{end}}\ \}$ $\quad$ $\{\ \overrightarrow{\overrightarrow{\mathsf{sup}}},\ \mathsf{f}\ \} \to \{\ \overleftarrow{\mathsf{sup}}\ \}$ $\quad$ $\{\ \overrightarrow{\overrightarrow{0}},\ \mathsf{f}\ \} \to \{\ \overleftarrow{0}\ \}$

Tableau 9.2 – Déclencher le dénombrement des solutions.



(a) Cas d'une assignation ne satisfaisant pas la formule.



(b) Cas d'une assignation satisfaisant la formule.

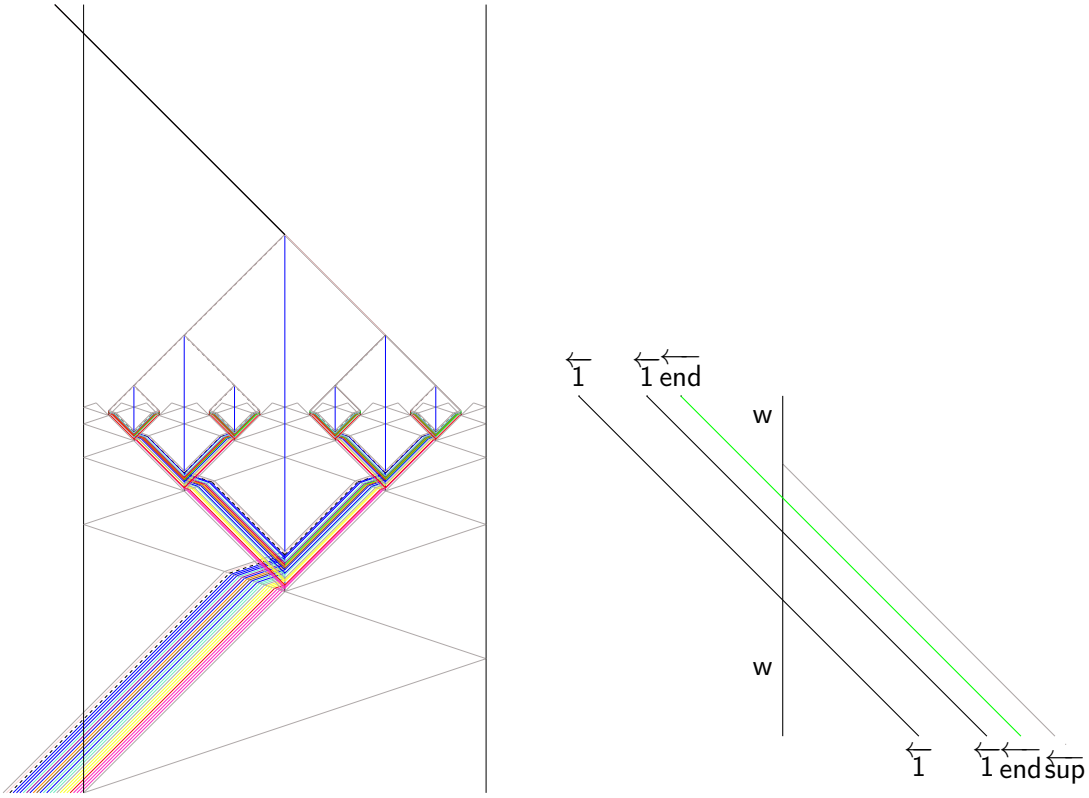
Figure 9.2 – Initier le décompte des assignations satisfaisant la formule.

Nous ne détaillons pas les autres versions symétriques et les versions rapides de tous les

méta-signaux supplémentaires utilisés ici (qui sont de type $+$, $+_0^R$, $+_0^L$, $\mathsf{0}$, $\mathsf{1}$, end et sup), ainsi que les règles de collisions nécessaires à leur propagation dans l'arbre (notamment au niveau de la lentille) : ces dernières respectent en effet le schéma de règles de la lentille-duplicatrice donné en SOUS-SEC. 8.1.2.

Diagramme global. Les diagrammes globaux de résolution géométrique de #**SAT** sont similaires aux diagrammes de résolution de **Q-SAT**, la seule différence étant que l'étape de collecte des résultats nécessite des faisceaux et non plus de simples signaux seuls. Un exemple de tel diagramme est donné par la FIG. 9.3(a), qui fournit le diagramme global d'une résolution de #**SAT** pour l'instance $\varphi = x_1 \wedge (\neg x_2 \vee x_3)$. La FIGURE 9.3(b) présente un zoom sur le résultat obtenu, constitué par le faisceau qui traverse le mur tout en haut à gauche. Après la suppression du signal $\overleftarrow{\mathsf{sup}}$, on obtient comme sortie un faisceau binaire qui représente la valeur 3, correspondant bien au nombre d'assignations de (x_1, x_2, x_3) en booléens de telle sorte que $\varphi(x_1, x_2, x_3)$ soit satisfaite.

Notons que par rapport à la machine résolvant **Q-SAT**, seuls dix méta-signaux supplémentaires sont nécessaires pour résoudre #**SAT** (soit 131 méta-signaux au total). Il faut en revanche définir une cinquantaine de nouvelles règles de collisions non blanches (soit 245 règles en tout), qui proviennent surtout de l'étape de collecte, plus complexe pour #**SAT** que pour **Q-SAT** (notamment à cause de l'additionneur binaire).



(a) Diagramme global. (b) Le faisceau résultat.

FIGURE 9.3 – Résolution de #**SAT** pour l'instance $\varphi = x_1 \wedge (\neg x_2 \vee x_3)$. Le faisceau sortant en haut à gauche représente le nombre 3 en binaire, qui correspond au nombre d'assignations satisfaisant φ.

9.2 Les problèmes One-SAT et All-SAT

Après le problème de comptage vu en Sec. 9.1, nous allons maintenant étudier des problèmes d'énumération de solutions à des problèmes de satisfaisabilité. Parmi les variantes qui consistent à énumérer des solutions du problème **SAT**, nous en distinguons deux, très proches : la première qui consiste à ne rendre *qu'une* seule assignation satisfaisant la formule, et qui fait alors office de *certificat* ; la seconde consiste à trouver et à énumérer *toutes* les solutions. Nous nommons ces deux variantes respectivement **One-SAT** et **All-SAT**. Comme dans la section précédente, nous allons maintenant détailler les étapes et donner les règles nécessaires à la résolution de ces problèmes. Ces étapes et nouvelles règles s'ajoutent à celles de la machine du Chap. 8, et comme pour la résolution de #**SAT**, elles remplacent essentiellement l'étape de collecte de **Q-SAT**. Certaines règles ne seront pas données, car elles se déduisent soit des schémas donnés au chapitre précedent (comme c'est le cas des règles et des méta-signaux rapides utilisés par la lentille-duplicatrice), soit par symétrie.

9.2.1 Le problème One-SAT

On considère ici **One-SAT**, une variante plus forte du problème **SAT** consistant à retourner un certificat pour les instances positives. Le problème **One-SAT** consiste donc à retourner une assignation (quand elle existe) satisfaisant la formule propositionnelle donnée en instance d'entrée de **SAT**. Un algorithme de résolution de **One-SAT** permet également de résoudre **SAT** puisque si une assignation est retournée, alors la formule est satisfaisable (et l'assignation en est un certificat), sinon elle ne l'est pas. Formellement, le problème s'énonce :

Problème **One-SAT**
entrée : une formule propositionnelle $\varphi(x_1, \ldots, x_n)$.
sortie : une assignation de $(x_1, \ldots, x_n)$ satisfaisant φ, s'il en existe.

Trouver et sauvegarder une assignation solution. L'intuition ici est de rajouter un sous-faisceau permettant de sauvegarder les assignations jusqu'au niveau final. Pour cela, on rajoute à la fin du faisceau initial une pile de n signaux de variables, encadrés par deux signaux $\overrightarrow{\mathsf{v}_e}$ et $\overrightarrow{\mathsf{v}}$ qui marqueront respectivement le début et la fin de l'assignation. La sous-configuration correspondante est donc donnée par $(\overrightarrow{\mathsf{v}_e}\ \overrightarrow{\mathsf{x}}^n\overrightarrow{\mathsf{v}})$. Par une légère modification de module de duplication (précisément, on ajoute deux nouveaux méta-signaux permettant d'affecter en signal booléen le premier de ces signaux de type x qui arrive sur le signal vertical de duplication), le $i^{\text{ème}}$ signal x est décidé au niveau i. Au dernier niveau, et après l'évaluation, il ne subsiste pour chaque cas possible d'assignation qu'un faisceau qui représente exactement l'assignation des variables correspondant à l'évaluation de ce cas. Si l'évaluation est positive, alors le faisceau-assignation correspond à une assignation satisfaisant la formule. Il est alors réfléchi pour ensuite se propager dans la phase de collecte des résultats. Des exemples de cas de réflexion et d'annihilation d'assignations sont respectivement donnés par les Fig. 9.4(a) et Fig. 9.4(b). Les règles correspondantes sont données dans le Tab. 9.3 (pour le cas de rebonds à droite), où les signaux v_+ et $\mathsf{v}_\circ$ signifient que l'assignation est respectivement une assignation qui satisfait et ne satisfait pas la formule.

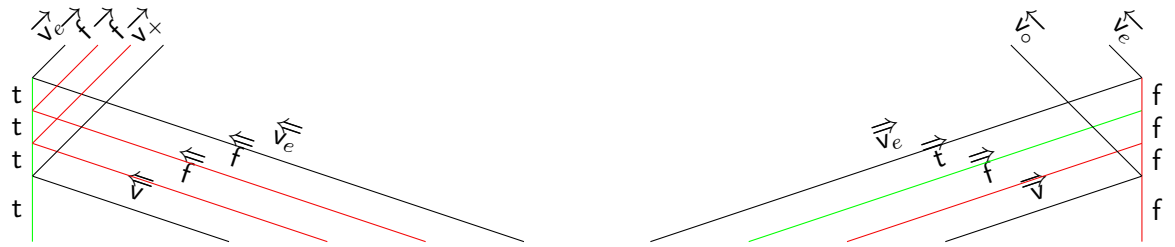
(a) Renvoyer une assignation.

$\{\ \overrightarrow{\mathsf{v}}, \mathsf{t}\ \} \to \{\ \overleftarrow{\mathsf{v}_+}, \mathsf{t}\ \}$ $\quad$ $\{\ \overrightarrow{\mathsf{v}_e}, \mathsf{t}\ \} \to \{\ \overleftarrow{\mathsf{v}_e}\ \}$
$\{\ \overrightarrow{\mathsf{t}}, \mathsf{t}\ \} \to \{\ \overleftarrow{\mathsf{t}}, \mathsf{t}\ \}$ $\quad$ $\{\ \overrightarrow{\mathsf{f}}, \mathsf{t}\ \} \to \{\ \overleftarrow{\mathsf{f}}, \mathsf{t}\ \}$

(b) Annihiler une assignation.

$\{\ \overrightarrow{\mathsf{v}}, \mathsf{f}\ \} \to \{\ \overleftarrow{\mathsf{v}_\circ}, \mathsf{f}\ \}$ $\quad$ $\{\ \overrightarrow{\mathsf{v}_e}, \mathsf{f}\ \} \to \{\ \overleftarrow{\mathsf{v}_e}\ \}$
$\{\ \overrightarrow{\mathsf{t}}, \mathsf{f}\ \} \to \{\ \mathsf{f}\ \}$ $\quad$ $\{\ \overrightarrow{\mathsf{f}}, \mathsf{f}\ \} \to \{\ \mathsf{f}\ \}$

Tableau 9.3 – Sauvegarder les assignations solutions.



(a) Cas d'une assignation gagnante.

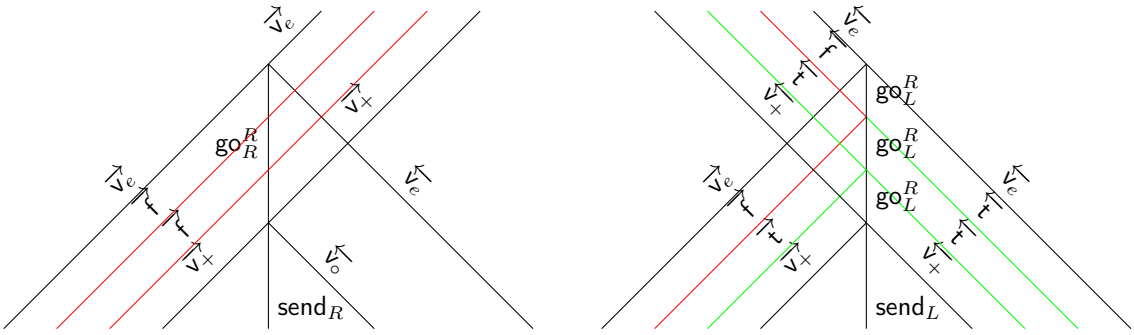
(b) Cas d'une assignation non-satisfaisante.

FIGURE 9.4 – Sélection des assignations satisfaisant la formule.

Toutes les assignations satisfaisant la formule se propagent ainsi en remontant. Plusieurs cas se produisent aux jonctions : soit aucun des sous-cas correspondant à la jonction n'a produit d'assignation ; soit une seule assignation arrive à la jonction, auquel cas elle est automatiquement propagée au niveau suivant ; soit deux assignations arrivent à la jonction. Dans ce dernier cas, une seule doit être choisie par définition du problème **ONE-SAT** : nous avons fixé les règles arbitrairement de telle sorte à propager l'assignation qui arrive par la gauche. Deux de ces trois cas sont illustrés par la FIG. 9.5 ; les règles sont données par le TAB. 9.4 pour le cas d'un résultat à envoyer vers la droite en sortie de jonction. Dans ces diagrammes, la notation go_Y^X signifie « envoyer dans la direction Y l'assignation arrivant de la direction X ». Les signaux verticaux send_R et send_L, mis en place à chaque duplication, indiquent la direction dans laquelle envoyer le faisceau sortant de la jonction.

$$\{ \overrightarrow{\mathsf{v}_+}, \mathsf{send}_R, \overleftarrow{\mathsf{v}_+} \} \to \{ \mathsf{go}_R^R, \overrightarrow{\mathsf{v}_+} \} \qquad \{ \overrightarrow{\mathsf{v}_+}, \mathsf{send}_R, \overleftarrow{\mathsf{v}_\circ} \} \to \{ \mathsf{go}_R^R, \overrightarrow{\mathsf{v}_+} \}$$
$$\{ \overrightarrow{\mathsf{v}_\circ}, \mathsf{send}_R, \overleftarrow{\mathsf{v}_+} \} \to \{ \mathsf{go}_R^L, \overrightarrow{\mathsf{v}_+} \} \qquad \{ \overrightarrow{\mathsf{v}_\circ}, \mathsf{send}_R, \overleftarrow{\mathsf{v}_\circ} \} \to \{ \mathsf{send}_R, \overrightarrow{\mathsf{v}_\circ} \}$$
$$\{ \overrightarrow{B_1}, \mathsf{go}_R^L, \overleftarrow{B_2} \} \to \{ \mathsf{go}_R^L, \overrightarrow{B_1} \} \qquad \{ \overrightarrow{B_1}, \mathsf{go}_R^R, \overleftarrow{B_2} \} \to \{ \mathsf{go}_R^R, \overrightarrow{B_2} \}$$
$$\{ \overrightarrow{B_1}, \mathsf{go}_R^L \} \to \{ \mathsf{go}_R^L, \overrightarrow{B_1} \} \qquad \{ \mathsf{go}_R^R, \overleftarrow{B_1} \} \to \{ \mathsf{go}_R^R, \overrightarrow{B_1} \}$$
$$\{ \overrightarrow{\mathsf{v}_e}, \mathsf{go}_R^L, \overleftarrow{\mathsf{v}_e} \} \to \{ \overrightarrow{\mathsf{v}_e} \} \qquad \{ \overrightarrow{\mathsf{v}_e}, \mathsf{go}_R^R, \overleftarrow{\mathsf{v}_e} \} \to \{ \overrightarrow{\mathsf{v}_e} \}$$
$$\{ \overrightarrow{\mathsf{v}_e}, \mathsf{send}_R, \overleftarrow{\mathsf{v}_e} \} \to \{ \overrightarrow{\mathsf{v}_e} \}$$

TABLEAU 9.4 – Règles pour les jonctions des assignations (données pour le faisceau résultant envoyé vers la droite). Les notations B_1 et B_2 désignent des booléens *i.e.* $B_1, B_2 \in \{\mathsf{t}, \mathsf{f}\}$.



(a) Cas de jonction avec une seule assignation.

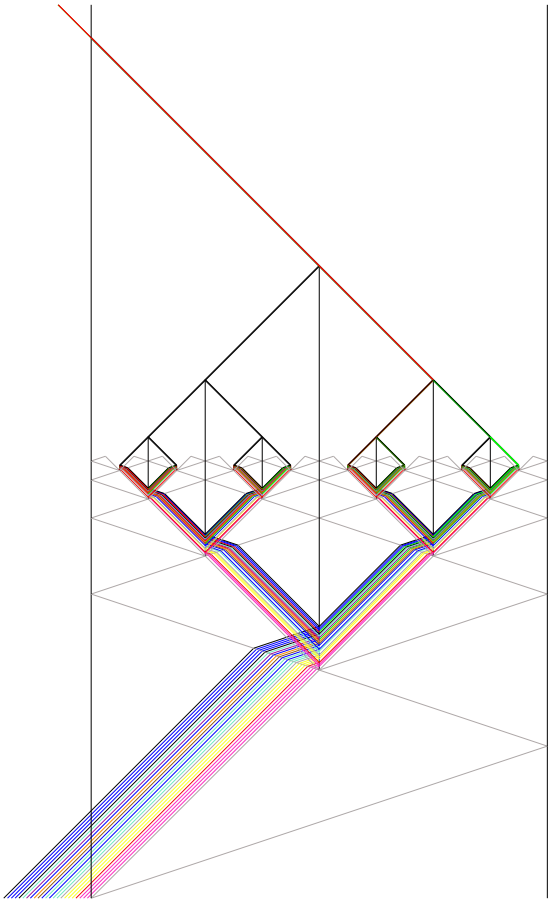
(b) Cas de jonction avec deux assignations.

FIGURE 9.5 – Collecte des assignations solutions.

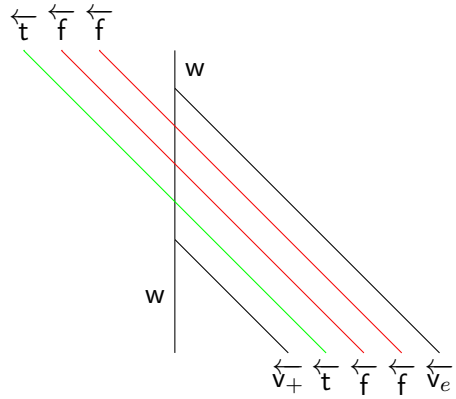
Diagramme global. Comme les règles ont été définies pour qu'en cas de jonction de deux assignations, ce soit celle de gauche qui est transmise au niveau suivant, il s'ensuit que si la

formule donnée en entrée est satisfaisable, le certificat rendu par la machine sera le plus petit certificat dans l'ordre lexicographique, en considérant que $faux$ précède $vrai$.

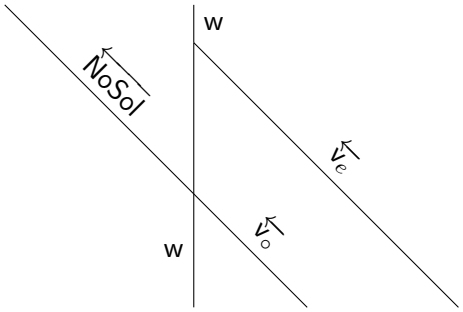
La FIGURE 9.6(a) fournit le diagramme global correspondant à la formule $\varphi = x_1 \wedge (\neg x_2 \vee x_3)$. Dans cet exemple, il existe plusieurs assignations satisfaisant la formule, mais c'est celle qui le plus à gauche — $(x_1, x_2, x_3) = (vrai, faux, faux)$ — qui est finalement retournée. Cette solution est renvoyée sous forme de faisceau se propageant vers la gauche, tel qu'illustré par la FIG. 9.6(b). Elle peut ainsi être lue temporellement, en position 0 : les valeurs booléennes des variables sont données par les types des signaux qui arrivent successivement au niveau du mur de gauche de la construction. Lorsque la formule n'est pas satisfaisable, seul un signal $\overleftarrow{\text{NoSol}}$ est retourné (voir la FIG. 9.6(c)). Avec un peu plus de 130 méta-signaux et 220 règles non blanches, la résolution de **ONE-SAT** nécessite une machine de taille équivalente à la machine résolvant #**SAT**.



(a) Diagramme global.



(b) Le faisceau résultat.



(c) Cas sans solution.

FIGURE 9.6 – Résolution de **ONE-SAT** pour l'instance $\varphi = x_1 \wedge (\neg x_2 \vee x_3)$. L'assignation $(x_1, x_2, x_3) = (vrai, faux, faux)$ est une solution (la première dans l'ordre lexicographique), donnée sous forme de faisceau sortant en haut à gauche de la construction.

9.2.2 Le problème ALL-SAT

On peut également considérer **ALL-SAT**, une version d'énumération du problème **SAT**. Le problème **ALL-SAT** est une variante « plus forte » du problème **ONE-SAT**, puisqu'il consiste à retourner *toutes* les assignations satisfaisant une instance du problème **SAT** :

Problème **ALL-SAT**

entrée : une formule propositionnelle $\varphi(x_1, \ldots, x_n)$.
sortie : toutes les assignations de $(x_1, \ldots, x_n)$ satisfaisant φ.

Retourner toutes les assignations de variables satisfaisant une formule ψ peut être effectué de manière simple sur machine à signaux à partir de la machine résolvant **One-SAT** modifiée afin de stocker les assignations de variables sous forme de faisceaux de signaux stationnaires. Ici, le principe est même plus simple que pour **One-SAT** puisque les étapes de jonctions de faisceaux-assignation ne sont plus nécessaires.

Sauvegarder toutes les assignations solutions. Les signaux et règles supplémentaires pour pouvoir sauvegarder et propager les assignations dans l'arbre sont les mêmes que pour le problème **One-SAT**, ainsi que la contribution à la configuration initiale. La différence se situe au niveau du traitement des assignations après l'évaluation. Alors que pour **One-SAT**, les assignations satisfaisantes étaient réfléchies, elles sont ici « verticalisées » (*i.e.* immobilisées) par les signaux v_+, les assignations qui ne satisfont pas la formule étant quant à elles annihilées par les signaux $\mathsf{v}_\circ$. Les règles correspondantes à ces deux cas sont données par le Tab. 9.5, pour des faisceaux arrivant de gauche (les règles pour le cas de droite s'en déduisant facilement par symétrie). Les deux cas — annihilation et sauvegarde d'une assignation — sont illustrés par la Fig. 9.7.

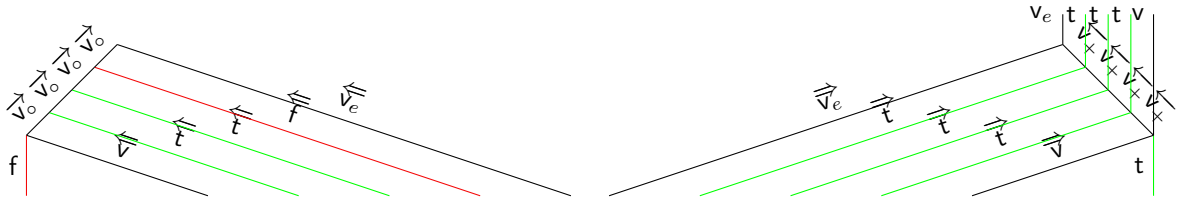
(a) Annihiler une assignation.

$\{ \overrightarrow{\mathsf{v}}, \mathsf{f} \} \to \{ \overleftarrow{\mathsf{v}_\circ} \}$ $\quad \{ \overrightarrow{\mathsf{v}_e}, \overleftarrow{\mathsf{v}_\circ} \} \to \emptyset$
$\{ \mathsf{t}, \overleftarrow{\mathsf{v}_\circ} \} \to \{ \overleftarrow{\mathsf{v}_\circ} \}$ $\quad \{ \overrightarrow{\mathsf{f}}, \overleftarrow{\mathsf{v}_\circ} \} \to \{ \overleftarrow{\mathsf{v}_\circ} \}$

(b) Verticaliser une assignation.

$\{ \overrightarrow{\mathsf{v}}, \mathsf{t} \} \to \{ \overleftarrow{\mathsf{v}_+}, \mathsf{v} \}$ $\quad \{ \overrightarrow{\mathsf{v}_e}, \overleftarrow{\mathsf{v}_+} \} \to \{ \mathsf{v}_e \}$
$\{ \overrightarrow{\mathsf{t}}, \overleftarrow{\mathsf{v}_+} \} \to \{ \overleftarrow{\mathsf{v}_+}, \mathsf{t} \}$ $\quad \{ \overrightarrow{\mathsf{f}}, \overleftarrow{\mathsf{v}_+} \} \to \{ \overleftarrow{\mathsf{v}_+}, \mathsf{f} \}$

Tableau 9.5 – Verticaliser les assignations solutions.



(a) Annihiler une assignation perdante.

(b) Verticaliser une assignation gagnante.

Figure 9.7 – Énumération des résultats sous forme de faisceaux stationnaires.

Diagramme global. Comme les assignations sont stockées sous forme de faisceaux stationnaires, le diagramme global de résolution du problème **All-SAT** est légèrement différent des diagrammes correspondant aux résolutions des autres variantes de **SAT** : la phase de collecte est inexistante, puisqu'ici tous les résultats sont sauvegardés directement après les évaluations. Les assignations résultantes se présentent sous forme de faisceaux dont les deux signaux extrêmes (v et v_e) indiquent l'ordre de lecture des variables c'est-à-dire qu'à partir de v vers v_e, les signaux verticaux booléens correspondent aux valeurs successives des variables x_1, x_2, …, x_n. La Figure 9.8 donne le diagramme de résolution de **All-SAT** pour l'instance $\varphi = x_1 \wedge (\neg x_2 \vee x_3)$, et la Fig. 9.7(b) donnée ci-dessus, fournit un zoom sur un des faisceaux sortants. Celui-ci correspond à l'assignation $(vrai, vrai, vrai)$, représentée par le faisceau stationnaire (v_e t t t v).

Contrairement à la résolution de **One-SAT**, les assignations satisfaisantes sont sauvegardées sous forme de faisceaux stationnaires. De ce fait, il ne reste à la fin que des signaux verticaux,

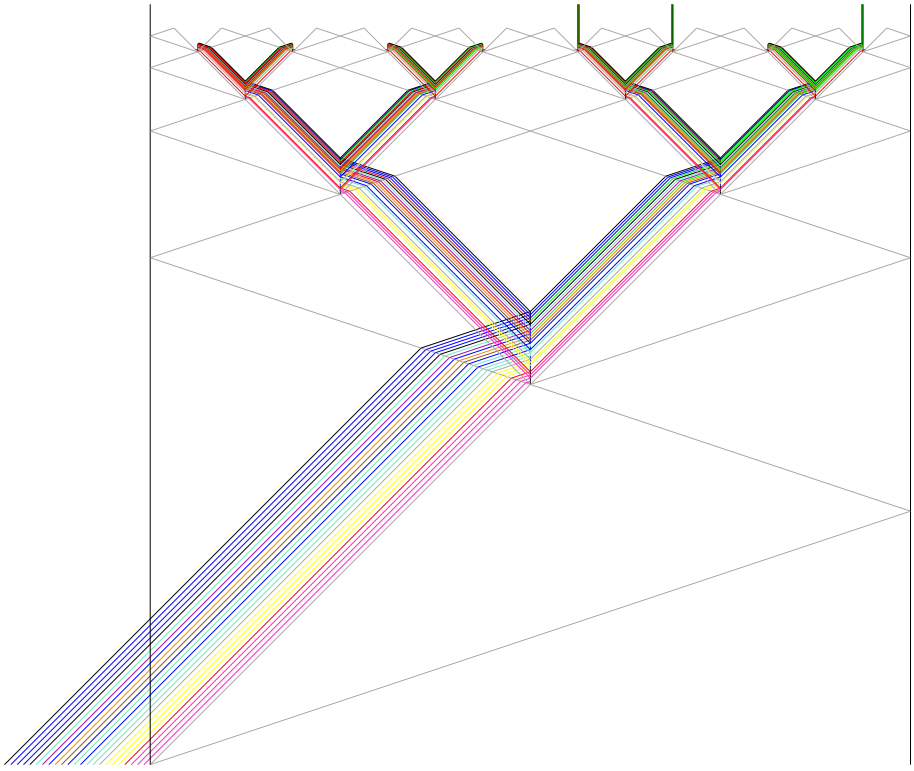


FIGURE 9.8 – Résolution de **ALL-SAT** pour l'instance $\varphi = x_1 \wedge (\neg x_2 \vee x_3)$. Les trois faisceaux stationnaires résultants correspondent aux trois seules assignations qui satisfont la formule c'est-à-dire aux assignations $(vrai, faux, faux)$, $(vrai, faux, vrai)$ et $(vrai, vrai, vrai)$.

et plus aucune dynamique n'est possible. Les solutions sont stockées ici de manière statique, et sont lues spatialement.

Notons également que, parmi les machines présentées dans ce manuscrit pour résoudre des problèmes de satisfaisabilité, cette machine est la plus petite en taille : 127 méta-signaux et 186 règles non blanches. Contrairement aux autres machines, il n'y a pas ici d'étape de collecte, faisant ainsi l'économie de nombreuses règles de collisions.

9.3 Le problème MAX-SAT

Nous donnons ici l'intuition et les grandes lignes d'une machine générique permettant de résoudre le problème **MAX-SAT**. Il s'agit d'un problème d'optimisation qui consiste à trouver le maximum de *clauses* parmi k clauses données en entrée pouvant être satisfaites par une même assignation de variables :

Problème **MAX-SAT**

> *entrée* : k clauses $C_1(x_1, \ldots, x_n)$, $\ldots$, $C_k(x_1, \ldots, x_n)$.
> *sortie* : le nombre maximum de clauses qu'une assignation de $(x_1, \ldots, x_n)$ satisfait.

Pour résoudre **MAX-SAT**, il faut donc compter le nombre de clauses satisfaites plutôt que le nombre d'assignations comme c'est le cas pour résoudre **#SAT**. Le problème **MAX-SAT** peut être étendu afin de retourner en plus l'assignation satisfaisant le plus grand nombre de clauses possibles parmi les k clauses données en entrée.

Ramené à un problème de décision, ce problème est **NP**-dur et le problème tel qu'énoncé ci-dessus est complet pour la classe **APX**, la classe des problèmes d'optimisation qui admettent des algorithmes d'approximation en temps polynomial et à facteur d'approximation constant

(voir [Papadimitriou 1994, pp. 301–302] pour plus de détails sur **Max-SAT**).

Principe d'une résolution géométrique de Max-SAT. Nous décrivons ici informellement une machine à signaux pour résoudre une version plus générale de **Max-SAT** : nous ne restreignons pas les instances de **Max-SAT** à un ensemble de clauses mais considérons un ensemble quelconque (mais fini) de formules propositionnelles. La machine est basée sur la machine précédente résolvant **Q-SAT** de manière générique à laquelle nous ajoutons le module d'addition binaire déjà utilisé pour résoudre #**SAT**. Il suffit ensuite de rajouter un module supplémentaire permettant de calculer le maximum de deux nombres codés en binaire.

Chaque formule parmi les k formules données en entrée est compilée par le même schéma que celui utilisé pour **Q-SAT**, résultant en un faisceau de signaux parallèles codant la formule. Les k faisceaux correspondant aux k formules sont alors placés bout à bout constituant ainsi la partie de la configuration initiale codant l'ensemble des k formules, une formule étant représentée par un *sous-faisceau*.

Chaque formule est ensuite évaluée exactement de la même façon que pour **Q-SAT** et pour chaque assignation possible, les évaluations sont réalisées de manière indépendante pour chaque sous-faisceau, c'est-à-dire pour chaque formule.

Dans chaque cas d'assignation possible, le résultat de l'évaluation d'une formule est ajouté grâce à l'additionneur binaire à un faisceau codant le nombre de formules déjà satisfaites. Ce compteur binaire est d'abord initialisé par la première valeur *vrai* reçue, selon le même principe que celui de la machine pour #**SAT**. Il est ensuite incrémenté de un par chaque nouvelle *vrai*, grâce à l'additionneur binaire.

À la fin de l'étape globale d'évaluation, on obtient ainsi pour chaque cas possible un faisceau codant en binaire le nombre de formules satisfaites dans le cas de l'assignation considérée. L'étape de collecte s'opère alors sur le même principe que les variantes précédentes mais prend en compte en plus la comparaison des nombres obtenus précédemment : à chaque jonction des résultats provenant de deux cas possibles, les nombres de formules satisfaites dans chaque cas sont comparés grâce au module calculant le maximum en binaire, et le plus grand nombre est transmis au niveau supérieur pour les comparaisons suivantes. Enfin, au sommet de la construction, après que toutes les comparaisons ont été effectuées, le nombre maximal de formules pouvant être satisfaites par une même assignation est donné par un faisceau binaire.

L'assignation qui satisfait le plus grand nombre de formules peut également être retournée en combinant la construction résolvant **Max-SAT** avec le module utilisé en Sec. 9.2 permettant de sauvegarder et de renvoyer une assignation qui satisfait la formule.

Calculer le maximum de deux nombres binaires. Nous décrivons ici brièvement le fonctionnement d'un module de calcul du maximum de deux nombres binaires c'est-à-dire d'un comparateur binaire. Comme pour l'additionneur binaire, les deux entiers à comparer sont représentés par des faisceaux binaires de directions opposées et suivant le codage proposé en Sous-sec. 3.2.1 (pour un faisceau se propageant vers la droite, les signaux les plus à droite désignent les bits de poids les plus faibles). Chaque faisceau contient deux copies de l'entier qu'il représente : la première pour la comparaison proprement dite, la deuxième étant transmise en sortie de module dans le cas échéant. Ainsi, pour un entier n d'écriture binaire $b_k \dots b_1 b_0$, le faisceau de gauche utilisé pour représenté n dans le module de comparateur binaire est donné par $(\overrightarrow{\mathsf{sup}}\ \overrightarrow{\mathsf{end}}\ \overrightarrow{\mathsf{b}_k} \dots \overrightarrow{\mathsf{b}_0}\ \overrightarrow{\mathsf{comp}}\ \overrightarrow{\mathsf{end}}\ \overrightarrow{\mathsf{b}_k} \dots \overrightarrow{\mathsf{b}_0}) = (\overrightarrow{\mathsf{sup}}\ [\overrightarrow{bin}(n)]\ \overrightarrow{\mathsf{comp}}\ [\overrightarrow{bin}(n)])$. La configuration d'entrée pour comparer deux nombres n et m est donc donnée par : $(\overrightarrow{\mathsf{sup}}\ [\overrightarrow{bin}(n)]\ \overrightarrow{\mathsf{comp}}\ [\overrightarrow{bin}(n)]\ =_t\ [\overleftarrow{bin}(m)]\ \overleftarrow{\mathsf{comp}}\ [\overleftarrow{bin}(m)]\ \overleftarrow{\mathsf{sup}})$, où $=_t$ est le signal qui déclenche la comparaison lors de la première triple collision entre les deux bits de poids faible.

Les règles de collisions implémentent le fait que le maximum de deux nombres binaires est celui soit qui possède le plus grand nombre de bits, soit qui possède le plus grand bit dont

la position est telle que pour les bits de positions supérieures sont égaux. Les bits sont donc succesivement comparés deux-à-deux. Les signaux verticaux $=_t$, $\gtrsim_t$ et $\lesssim_t$ (l'indice t signifiant « temporaire ») permettent de sauvegarder le côté du nombre qui est le plus grand à ce moment de la comparaison. Si les bits comparés sont égaux, alors le maximum temporaire ne change pas. Sinon, le maximum devient le nombre dont le bit était le plus grand. Après comparaison de tous les bits, le résultat final est donné par la valeur du dernier signal vertical de comparaison temporaire : le résultat de la comparaison est ainsi sauvegardé par un signal vertical $=$, $\gtrsim$ ou $\lesssim$, signifiant respectivement que les nombres sont égaux, que le nombre de gauche est le maximum et que celui de droite est le maximum. Suivant les cas, le signal vertical laisse passer le faisceau correspondant au maximum et annihile l'autre faisceau (si les deux nombres sont égaux, on laisse passé le faisceau par gauche par convention).

La FIGURE 9.9 illustre une telle comparaison géométriques de deux nombres binaires avec l'exemple du calcul de $max(3, 5)$.

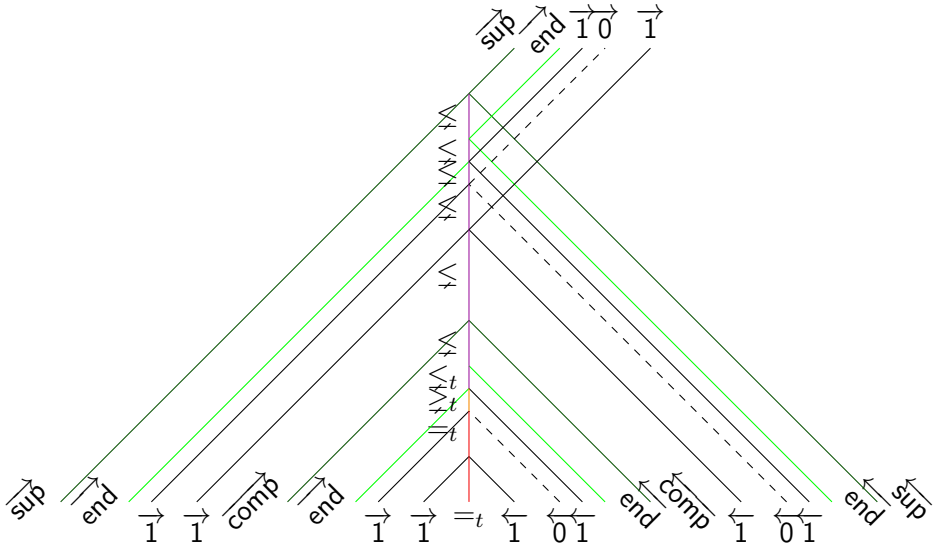


FIGURE 9.9 – Calcul de $max(3, 5)$ avec un comparateur géométrique.

Conclusion

Nous concluons ce manuscrit en synthétisant les contributions apportées et en rappelant leur contenu ; puis en évoquant leurs futurs prolongements directs, ainsi que d'autres pistes de recherches qui nous semblent pertinentes.

Bilan

Nous avons proposé dans ce manuscrit une étude de certaines propriétés des machines à signaux, portant essentiellement sur l'utilisation de l'espace à des fins d'algorithmique géométrique.

Cette investigation s'est faite dans un premier temps au travers des phénomènes inhérents à l'espace continu — les *accumulations* — et à un type particulier de tels phénomènes — les structures *fractales*. Dans un second temps, les notions nécessaires pour pouvoir parler d'*algorithmes géométriques* ont été définies, ainsi qu'une approche modulaire pour concevoir de tels algorithmes. Enfin, à partir des constructions obtenues sur les fractales et des méthodes dégagées par l'approche modulaire, ont été proposées diverses résolutions géométriques efficaces de problèmes difficiles à résoudre sur des modèles classiques.

Nous pouvons donc résumer l'apport de ce manuscrit à trois contributions principales, portant sur les structures d'accumulations et de fractales ; sur les algorithmes géométriques ; et sur l'utilisation massivement parallèle de l'espace pour résoudre efficacement certains problèmes difficiles.

Accumulations et fractales. Les accumulations sur machines à signaux avaient déjà été considérées auparavant. Néanmoins, aucun critère n'avait été proposé pour caractériser les accumulations ou caractériser les machines pouvant les produire. Nous avons palié en partie à cela, en montrant que certains paramètres des machines à signaux — leur rationalité, leur nombre de vitesses distinctes — permet de formuler une condition nécessaire pour qu'une machine à signaux puisse générer des accumulations. Cette condition est le résultat principal du Chap. 4 et se résume par l'énoncé suivant : *une accumulation nécessite l'utilisation d'au moins* 3 *vitesses distinctes dans le cas général et* 4 *dans le cas de machines rationnelles.* En particulier, cette condition définit une frontière au sein des *petites machines à signaux* c'est-à-dire les machines ayant un faible nombre de vitesses distinctes. Outre le nombre de vitesses, la rationalité s'avère être un critère important. Cette étude a également permis de dégager plusieurs notions telles que l'inclusion de diagrammes, les supports de machines et de diagrammes ou encore les multi-bandes.

En tant que type particulièrement simple et régulier d'accumulations, les *fractales* apparaissent facilement dans le monde des machines à signaux. C'est donc de manière naturelle que nous nous sommes intéressés à leur construction sur des machines à signaux : le Chap. 5 en fournit de nombreux exemples, ainsi que des moyens d'approximer certaines structures fractales. L'étude des accumulations avec des petites machines s'est également faite de manière implicite au

Chap. 5 : en effet, toutes constructions de fractales proposées utilisent peu de vitesses (*e.g.* 4 pour l'arbre binaire simple, 9 pour les ensembles de Cantor). Du Théorème 25 découle même que *pour tout réel* $d \in]0;1[$*, il est possible de générer une fractale de dimension* d *avec une machine à signaux n'utilisant que* 9 *vitesses.*

Algorithmique géométrique. La formalisation d'*algorithmes et de programmation géométrique* s'est faite à la fois dans le Chap. 3 et le Chap. 6. Des exemples d'algorithmes géométriques ont d'abord été proposés dans le Chap. 3, et ont permis d'introduire les notions de codages, de représentations, d'entrées/sorties, de résolutions de problèmes, de complexités de calculs... sur machines à signaux. Ces notions ont ensuite fait l'objet de définitions formelles, fournissant un cadre pour une étude théorique des algorithmes géométriques.

C'est dans ce cadre qu'a été développée dans le Chap. 6 une *méthode de programmation géométrique modulaire.* Les algorithmes géométriques y sont pensés comme des « boîtes géométriques » ayant certaines fonctionnalités. Suivant l'aire géométrique « minimale » nécessaire au calcul (hexagone, trapèze ou rectangle), différentes notions de modules ont été décrites (H-module, T-module et R-module). À partir de ces modules et de leurs interprétations en termes de fonctions calculées, nous avons reformulé de manière modulaire et géométrique les notions classiques d'exécutions séquentielle et parallèle d'algorithmes. Ces compositions parallèles et séquentielles de modules géométriques permettent de construire à partir de modules donnés, de nouveaux modules plus complexes, et définissent ainsi une puissante méthode de programmation géométrique par modules. Néanmoins, ces opérations de compositions n'ont pas été définies de la manière la plus générale possible (nous avons défini des compositions uniquement pour les T-modules).

Utilisation massivement parallèle de l'espace et algorithmes efficaces. Comme suggéré par l'étude des fractales faite au Chap. 5, les machines à signaux permettent de découper l'espace de manière systématique et régulière. Combiné à l'approche modulaire développée au Chap. 6, ce découpage automatique de l'espace par des fractales peut être utilisé par des processus s'exécutant en parallèle. Ceci est illustré par les Chap. 7, 8 et 9. Dans ces chapitres sont proposées des machines à signaux permettant de résoudre les variantes classiques des problèmes de satisfaisabilité booléenne, dont les plus connues sont **SAT** et **Q-SAT**, respectivement **NP**-complet et **PSPACE**-complet. Ces solutions sont de deux types : résolution d'un problème par une famille de machines ou *résolution spécifique* (voir Chap. 7) et résolution par une unique machine à signaux ou *résolution générique* (voir Chap. 8).

Grâce à la programmation modulaire, adaptée aux constructions parallèles, et grâce à l'approximation de l'arbre binaire, des machines à signaux résolvant les problèmes de satisfaisabilité ont été construites. Pour cela, la structure de l'arbre binaire a été utilisée pour définir une sorte de grille de calcul, dans laquelle ont été insérés en parallèle les calculs nécessaires à la résolution des problèmes considérés. Cette construction se comprend au travers des différents modules géométriques utilisés, dont les compositions permettent d'assurer, en vertu des résultats du Chap. 6, la correction de la construction globale ainsi que sa complexité en collisions. Les constructions des Chap. 7 et 8 démontrent qu' *il existe des machines à signaux qui résolvent respectivement les problèmes* **SAT** *et* **Q-SAT** *avec des profondeurs de collisions polynomiales.*

Ces constructions sont une parfaite illustration du pouvoir de calcul fortement parallèle des machines à signaux. En effet, l'efficacité de ce parallélisme est montré à travers les résolutions géométriques efficaces des problèmes de décision réputés difficiles que sont **SAT** et **Q-SAT**. De plus, ces solutions ont permis de facilement dériver au Chap. 9 des solutions pour les problèmes #**SAT**, **All-SAT** et **Max-SAT**. Cela montre qu'en plus des problèmes de décision,

les machines à signaux peuvent également résoudre de manière parallèle et efficace des problèmes respectivement de comptage, d'énumération et d'optimisation.

Perspectives

Évoquons maintenant les perspectives entrouvertes (directement ou non) durant la thèse. Nous avons regroupé ces pistes de recherche principalement en trois catégories qui correspondent aux thématiques des contributions du manuscrit : accumulations, algorithmique géométrique et complexités. Nous finirons en donnant quelques idées sur certaines variantes et extensions possibles du modèle.

Fractales, accumulations et minimalité. Il existe de nombreuses directions de recherche sur l'études des fractales et des accumulations. Nous en dégageons ici trois qui sont en continuité directe avec les résultats du manuscrit.

La première consiste à continuer l'étude des fractales engendrables par machines à signaux, et à caractériser de manière exacte de telles fractales. Il semble *a priori* que les fractales de dimension inférieure à 1 et définies par IFS peuvent être générées par des machines à signaux à une dimension.

Dans un second temps, nous souhaiterions compléter et généraliser l'étude menée au Chap. 4. Celle-ci a mis en évidence l'existence d'un *effet de seuil* sur le nombre de vitesses requises pour pouvoir construire une accumulation. De tels seuils (sur les vitesses ou sur le nombre de méta-signaux) existent-ils aussi pour d'autres propriétés des accumulations, comme le fait d'être isolée ou non ? Plus particulièrement, nous nous posons la question d'obtenir une version très générale du critère obtenu en Chap. 4 : est-il possible de construire des accumulations avec $n+1$ vitesses distinctes qu'il n'est pas possible de construire avec n vitesses ?

Il faut pour cela formaliser une notion d'« accumulations distinctes »: cela constitue une troisième direction de recherche. En effet, les accumulations sont bien définies ainsi que leurs principaux types (dynamique, statique, isolé ou non-isolé), mais de nombreuses propriétés nous sont encore inconnues. Afin de mener l'étude de ces propriétés, des concepts sont encore à définir et les différents types d'accumulations peuvent certainement être affinés et enrichis. Par exemple, il semblerait que l'accumulation simple de type Zénon à 4 vitesses consistant en un simple zig-zag (voir la Fig. 4.1(a)) soit une sorte d'« accumulation basique » dont la structure se retrouve dans les autres accumulations dynamiques, comme le triadique de Cantor qui peut être perçu comme une superposition d'accumulations simples. Cette idée d'accumulation élémentaire est-elle formalisable ?

Algorithmique géométrique. Nous avons proposé au Chap. 6 une approche modulaire et défini plusieurs types de compositions de modules. L'approche définie ici admet plusieurs continuations naturelles, dont nous en donnons ci-dessous deux qui sont immédiates et une troisième plus générale.

La première concerne une définition des modules qui soit la plus générale possible. En effet, bien que suffisamment puissantes pour construire les algorithmes géométriques présentés, les différentes compositions définies sur les modules ne concernent que des types particuliers de modules (les T-modules ou les R-modules) et ne s'appliquent pas aux modules de manière générale (*i.e.* aux H-modules). Des opérations de compositions parallèles et séquentielles définies pour n'importe quels modules pourraient permettre d'exprimer *n'importe quel* algorithme géométrique en termes de modules. La définition de telles compositions n'apparaît cependant pas simple, étant donnée la multitude de paramètres d'un H-module à prendre en compte.

Le second prolongement consiste à définir le même type de cadre que celui proposé au Chap. 6, mais avec une approche *top-down* c'est-à-dire une approche permettant de *décomposer automatiquement* un diagramme en modules. En effet, la méthode de programmation modulaire définie ici est de type *bottom-up* : de nouvelles machines et modules sont construits à partir de composants de base. L'approche duale serait donc de déduire les modules constituant un diagramme. Pour cela, des notions de sous-machines et sous-modules sont nécessaires.

Une troisième piste consiste à préciser le lien entre les notions de calculs géométriques et la notion de diagrammes équivalents donnée dans le Chap. 2, qui formalise l'intuition de « même calcul ». De cette définition peuvent être dérivées des notions formelles de simulations entre machines à signaux, reflétant de manière plus profonde les différents liens possibles entre diagrammes espace-temps. L'un de nos objectifs serait de définir des notions de « sous-calculs géométriques », ainsi que diverses notions de simulations entre machines à signaux. L'approche modulaire semble pouvoir fournir des outils adaptés pour construire des simulations entre machines.

Complexités. Les chapitres traitant des solutions géométriques des problèmes de satisfaisabilité qui caractérisent les grandes classes de complexité classique, soulèvent plusieurs questions sur le lien entre complexité de problèmes et machines à signaux.

Nous souhaiterions d'abord résoudre de manière directe d'autres problèmes **NP**-complets (sans utiliser de réduction à **SAT**). Par exemple, trouver une approche par signaux qui permet de représenter et étudier de manière simple des graphes permettrait de résoudre de nombreux problèmes de graphes qui sont **NP**-complets. Cela passerait par une formulation intuitive du concept de graphe sur machines à signaux, de la même façon que l'arbre binaire utilisé de manière canonique en tant qu'arbre de décision pour des problèmes booléens. Nous souhaiterions également éclaircir le statut, encore inconnu, d'autres problèmes plus difficiles comme des problèmes **EXPTIME**-complets ou **EXPSPACE**-complets.

Après avoir constaté que les constructions des Chap. 7, 8 et 9 sont toutes de profondeur de collisions polynomiales, indépendamment de la difficulté classique du problème considéré, il est naturel de se poser la question de savoir s'il existe des problèmes non résolubles par des machines à signaux rationnelles avec une profondeur de collisions polynomiale. Formulé autrement : cela a-t-il un sens de définir l'équivalent des classes de complexité classiques pour les machines à signaux ? Nous le pensons, et nous souhaitons à l'avenir définir formellement et étudier les propriétés et relations de classes de complexité intrinsèques aux machines à signaux, comme par exemple la classe **POLY-DEPTH**, définie comme la classe des problèmes résolubles par machines à signaux en profondeur de collisions polynomiale.

Enfin, nous souhaiterions caractériser de manière exacte l'efficacité des machines à signaux rationnelles à profondeur de collisions polynomiale. Nous avons en effet montré ici de manière informelle dans le Chap. 8 que **PSPACE** $\subseteq$ **POLY-DEPTH**. L'inclusion réciproque est-elle vraie ? Cette question est équivalente à celle de savoir si les machines à signaux vérifient l'*hypothèse du calcul parallèle* (la *Parallel Computation Thesis* ou *PCT*) définie dans [van Emde Boas 1990]. Un modèle de calcul parallèle vérifie la PCT si la classe des problèmes décidables en temps parallèle polynomial (pour la mesure de complexité en temps définie sur le modèle) est égale à la classe des problèmes décidables en espace polynomial classique. Cette thèse peut être formulée de manière intuitive par « le temps parallèle est-il équivalent à l'espace séquentiel ? »

Autour du modèle. Le modèle des machines à signaux présenté dans le manuscrit peut être étendu de multiples façons, et de nombreuses variantes du modèle semblent pouvoir être définies. Nous évoquons ici une généralisation possible, ainsi qu'une perspective de rapprochement entre les machines à signaux et les automates cellulaires.

La généralisation la plus naturelle du modèle est celle des machines à signaux en dimension n quelconque. S'il paraît simple de définir un tel modèle, ses capacités en termes de calculabilité et de complexité ne s'en déduisent pas immédiatement. Par exemple, une seconde dimension spatiale permet-elle un gain sur la complexité en collisions pour la résolution **Q-SAT** ? Dans ce cas, nous conjecturons que **Q-SAT** peut être résolu par une machine à signaux générique à deux dimensions spatiales en profondeur quadratique (l'intuition est de réaliser la construction du CHAP. 8 dans un plan et de passer dans un autre plan pour éviter les croisements de faisceaux lors des duplications) De manière plus générale, nous souhaiterions établir une distinction entre l'efficacité de machines de différentes dimensions, et établir si possible une caractérisation exacte de cette efficacité en fonction de la dimension, à la manière de ce qui a été fait dans [Bournez 1999] pour les systèmes PCD.

Comme nous l'avons mentionné en SOUS-SEC. 1.3.2, nous mettons à part toutes les questions directes concernant la construction de machines à signaux concrètes. De même, nous éludons les principales problématiques soulevées par ces questions, telles que la formulation d'un modèle physique des machines à signaux (prise en compte des phénomènes quantiques, machines avec vitesses non constantes, signaux avec épaisseur...). Néanmoins, nous pouvons chercher à nous rapprocher, d'un point de vue théorique, d'un modèle physiquement crédible. C'est dans ce contexte que nous pouvons situer la problématique importante de la *discrétisation* des diagrammes espace-temps continus. Nous ne parlons pas ici d'un découpage automatique de l'espace comme celui qui a été réalisé grâce aux structures fractales, mais d'une méthode qui permettrait de passer automatiquement du modèle des machines à signaux au modèle dont elles tirent leurs origines, à savoir les automates cellulaires. Une piste, non aboutie, a déjà été initiée dans [Levorato et Senot 2010]. Cette question de la discrétisation nous apparaît importante car, en plus de rapprocher les machines à signaux d'un modèle de calcul physiquement crédible, une méthode de discrétisation fournirait surtout un nouveau point de vue pour l'étude théorique à la fois des machines à signaux et des automates cellulaires, ainsi qu'un nouvel outil pour leur conception.

Comme nous l'avons vu, tant à travers les contributions que les perspectives proposées, les machines à signaux constituent un sujet très riche, offrant des connexions avec de nombreux domaines de l'informatique fondamentale. Bien que récentes, plusieurs de leurs propriétés importantes étaient déjà connues. Nous en avons démontré ici de nouvelles, élargissant ainsi les connaissances actuelles sur le modèle des machines à signaux. Les directions de recherche correspondant aux contributions ont été choisies parmi les nombreuses directions existantes pour leur importance au niveau de la compréhension du modèle. Leur étude nous a permis en plus de dégager des outils originaux pour l'étude du calcul géométrique par signaux, et d'explorer de nouvelles pistes de recherche, dont nous proposons également des prolongements.

Références bibliographiques

Aaronson, S. (2005). NP-complete problems and physical reality (guest column). *ACM SIGACT Newsletter*, 36(1):30–52.

Aaronson, S. et Watrous, J. (2009). Closed timelike curves make quantum and classical computing equivalent. *Proceedings of the Royal Society A*, 465(2102):631–647.

Abrams, D. S. et Lloyd, S. (1998). Computational complexity and physical law. *Dans* Williams, C. P., éditeur : *1st NASA International Conference on Quantum Computing and Quantum Communications (QCQC '98)*, numéro 1509 de LNCS, pages 167–173. Springer.

ACRI (2012). Sirakoulis, G.C. et Bandini, S., éditeurs : *Proceedings of the 10th International Conference on Cellular Automata for Research and Industry*, numéro 7495 de LNCS. Springer.

Adamatzky, A., éditeur (2002). *Collision-Based Computing*. Springer.

Adamatzky, A. et Durand-Lose, J. (2012). Collision-based computing. *Dans* [Handbook of Nat. Comp. 2012], chapitre 58, pages 1949–1978.

Adleman, L. M. (1994). Molecular computation of solutions to combinatorial problems. *Science*, 266(5187):1021–1024.

Alhazov, A. et Pérez-Jiménez, M. J. (2007). Uniform solution of Q-SAT using polarizationless active membranes. *Dans* Durand-Lose, J. et Margenstern, M., éditeurs : *5th International Conference on Machines, Computations and Universality (MCU '07)*, numéro 4664 de LNCS, pages 122–133. Springer.

Amos, M. (1997). *DNA computation*. Thèse de doctorat, University of Warwick.

Arulanandham, J. (2005). *Natural algorithms*. Thèse de doctorat, University of Auckland.

Asarin, E. et Maler, O. (1995). Achilles and the Tortoise climbing up the arithmetical hierarchy. *Dans* Thiagarajan, P., éditeur : *15th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS '95)*, numéro 1026 de LNCS, pages 471–483. Springer.

Barnsley, M. F. et Demko, S. (1985). Iterated Function Systems and the global construction of fractals. *Proceedings of the Royal Society A*, 399:243–275.

Becker, F. (2008). *Géométrie pour l'auto-assemblage*. Thèse de doctorat, École Normale Supérieure de Lyon. En français.

Becker, F. (2009). Pictures worth a thousand tiles, a geometrical programming language for self-assembly. *Theoret. Comput. Sci.*, 410(16):1495–1515.

Beigel, R. et Fu, B. (1998). Solving intractable problems with DNA computing. *Dans 13th International IEEE Conference on Computational Complexity (CCC '98)*, pages 154–169. IEEE Computer Society.

Berger, R. (1966). The undecidability of the domino problem. *Memoirs of the American Mathematical Society*, 66.

Blake, R. M. (1926). The paradox of temporal process. *Journal of Philosophy*, 23:645–654.

Blakey, E. (2011a). Computational complexity in non-Turing models of computation—the what, the why and the how. *Electronic Notes in Theoret. Comput. Sci.*, 270(1):17–28.

Blakey, E. (2011b). Unconventional complexity measures for unconventional computers. *Natural Computing*, 10(4):1245–1259.

Blum, L., Cucker, F., Shub, M. et Smale, S. (1998). *Complexity and Real Computation*. Springer New-York.

Blum, L., Shub, M. et Smale, S. (1989). On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bulletin of the American Mathematical Society*, 21(1):1–46.

Boccara, N., Nasser, J. et Roger, M. (1991). Particle-like structures and interactions in spatio-temporal patterns generated by one-dimensional deterministic cellular automaton rules. *Physical Review A*, 44(2):866–875.

Boolos, G. et Jeffrey, R. C. (1980). *Computability and logic*. Cambridge University Press.

Bournez, O. (1997). Some bounds on the computational power of piecewise constant derivative systems. *Dans* Degano, P., Gorrieri, R. et Marchetti-Spaccamela, A., éditeurs : *24th International Colloquium on Automata, Langages and Programming (ICALP '97)*, numéro 1256 de LNCS, pages 143–153. Springer.

Bournez, O. (1999). Achilles and the Tortoise climbing up the hyper-arithmetical hierarchy. *Theoret. Comput. Sci.*, 210(1):21–71.

Bournez, O. et Campagnolo, M. L. (2009). A survey on continuous time computations. *ArXiv preprints*, 0907.3117. Available at `http://arxiv.org/abs/0907.3117`.

Brun, T. A. (2003). Computers with closed timelike curves can solve hard problems efficiently. *Foundations of Physics Letters*, 16:245–253.

Brun, Y. (2008). Solving NP-complete problems in the tile assembly model. *Theoret. Comput. Sci.*, 395(1):31–46.

Brun, Y. (2012). Efficient 3-SAT algorithms in the tile assembly model. *Natural Computing*, 11(2):209–229.

Cantor, G. (1884). De la puissance des ensembles parfaits de points (On cardinality of perfect points sets). *Acta Mathematica*, 4:381–392. *Traduit et réédité dans* [Edgar 1993], pages 11–23.

Church, A. (1933). A set of postulates for the foundation of logic. *Annals of Mathematics*, 34:839–864. *Réédité dans* [Davis 2004].

Cook, M. (2004). Universality in elementary cellular automata. *Complex Systems*, 15(1):1–40.

Cook, S. A. (1971). The complexity of theorem proving procedures. *Dans* Harrison, M. A., Banerji, R. B. et Ullman, J. D., éditeurs : *3rd ACM Symposium on Theory of Computing (STOC '71)*, pages 151–158. ACM.

Copeland, J. B. (2002). Accelerating Turing machines. *Minds and Machines*, 12:281–301.

Čulik II, K. et Dube, S. (1989). Fractal and recurrent behavior of cellular automata. *Complex Systems*, 3:253–267.

Čulik II, K., Hurd, L. P. et Yu, S. (1990). Computation theoretic aspects of cellular automata. *Physica D*, 45:357–378.

Das, R., Crutchfield, J. P. et Mitchell, M. (1994). A genetic algorithm discovers particle-based computation in cellular automata. *Dans* Davidor, Y., Schwefel, H.-P. et Männer, R., éditeurs : *3rd International Conference on Parallel Problem Solving from Nature (PPSN III)*, numéro 866 de LNCS, pages 344–353. Springer.

Das, R., Crutchfield, J. P., Mitchell, M. et Hanson, J. E. (1995). Evolving globally synchronized cellular automata. *Dans* Eshelman, L. J., éditeur : *6th International Conference on Genetic Algorithms (ICGA '95)*, pages 336–343. Morgan Kaufmann Publishing.

Davis, M. (2004). *The Undecidable: Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions*. Dover Publication.

Delorme, M. et Mazoyer, J. (1999). *Cellular Automata: a Parallel Model*. Series on Mathematics and its Applications. Kluwer Academic Publishers.

Delorme, M. et Mazoyer, J. (2002). Signals on cellular automata. *Dans* [Adamatzky 2002], chapitre 9, pages 231–275.

Deutsch, D. (1985). Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 400(1818):97–117.

Dindoš, M. (2001). Generalized Cantor sets and sets of sums of convergent alternating series. *Journal of Applied Analysis*, 7(1):131–150.

Duchier, D., Durand-Lose, J. et Senot, M. (2010a). Fractal parallelism: solving SAT in bounded space and time. *Dans* Cheong, O., Chwa, K.-Y. et Park, K., éditeurs : *21st International Symposium on Algorithms and Computation (ISAAC '10)*, numéro 6506 de LNCS, pages 279–290. Springer.

Duchier, D., Durand-Lose, J. et Senot, M. (2010b). Massively parallel automata in Euclidean space-time. *Dans 4th IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW '10), Spatial Computing Workshop (SCW '10)*, pages 104–109. IEEE Computer Society.

Duchier, D., Durand-Lose, J. et Senot, M. (2011). Solving Q-SAT in bounded space and time by geometrical computation. *Dans* Ganchev, H., Löwe, B., Normann, D., Soskov, I. et Soskova, M., éditeurs : *Models of Computation in Context, 7th International Conference on Computability in Europe (CiE '11) (abstract and handout booklet)*, pages 76–86. St. Kliment Ohridski University Press, Sofia University.

Duchier, D., Durand-Lose, J. et Senot, M. (2012). Computing in the fractal cloud: modular generic solvers for SAT and Q-SAT variants. *Dans* Agrawal, M., Cooper, S. B. et Li, A., éditeurs : *9th International Conference on Theory and Applications of Models of Computation (TAMC '12)*, numéro 7287 de LNCS, pages 435–447. Springer.

Durand, B. (1996). Self-similarity viewed as a local property via tile sets. *Dans* Penczek, W. et Szalas, A., éditeurs : *21st international symposium on Mathematical Foundations of Computer Science (MFCS '96)*, numéro 1113 de LNCS, pages 312–323. Springer.

Durand-Lose, J. (1997). Intrinsic universality of a one-dimensional reversible cellular automaton. *Dans 14th Annual Symposium on Theoretical Aspects of Computer Science (STACS '97)*, numéro 1200 de LNCS, pages 439–450. Springer.

Durand-Lose, J. (2002). Computing inside the billiard ball model. *Dans* [Adamatzky 2002], pages 135–160.

Durand-Lose, J. (2003). *Calculer géométriquement sur le plan — machines à signaux.* Habilitation à Diriger des Recherches, École Doctorale STIC, Université de Nice-Sophia Antipolis. En français.

Durand-Lose, J. (2004). Abstract geometrical computation for black hole computation (extended abstract). *Dans* Margenstern, M., éditeur : *4th International Conference on Machines, Computations and Universality (MCU '04)*, numéro 3354 de LNCS, pages 176–187. Springer.

Durand-Lose, J. (2006a). Abstract geometrical computation 1: embedding black hole computations with rational numbers. *Fundamenta Informaticae*, 74(4):491–510.

Durand-Lose, J. (2006b). Reversible conservative rational abstract geometrical computation is Turing-universal. *Dans* Beckmann, A. et Tucker, J. V., éditeurs : *Logical Approaches to Computational Barriers, 2nd International Conference on Computability in Europe (CiE '06)*, numéro 3988 de LNCS, pages 163–172. Springer.

Durand-Lose, J. (2007). Abstract geometrical computation and the linear Blum, Shub and Smale model. *Dans* Cooper, S. B., Löwe, B. et Sorbi, A., éditeurs : *Computation and Logic in the Real World, 3rd International Conference on Computability in Europe (CiE '07)*, numéro 4497 de LNCS, pages 238–247. Springer.

Durand-Lose, J. (2008). Abstract geometrical computation with accumulations: beyond the Blum, Shub and Smale model. *Dans* Beckmann, A., Dimitracopoulos, C. et Löwe, B., éditeurs : *Logic and Theory of Algorithms, 4th International Conference on Computability in Europe (CiE '08)*, pages 107–116. University of Athens.

Durand-Lose, J. (2011a). Abstract geometrical computation 4: small Turing universal signal machines. *Theoret. Comput. Sci.*, 412(1-2):57–67.

Durand-Lose, J. (2011b). Abstract geometrical computation 5: embedding computable analysis. *Natural Computing*, 10(4):1261–1273. Special Issue on 8th International Conference on Unconventional Computation (UC '09).

Durand-Lose, J. (2012a). Abstract geometrical computation 6: a reversible, conservative and rational based model for black hole computation. *International Journal of Unconventional Computing*, 8(1):33–46.

Durand-Lose, J. (2012b). Abstract geometrical computation 7: geometrical accumulations and computably enumerable real numbers. *Natural Computing*, 11(4):609–622. Special Issue on 10th International Conference on Unconventional Computation (UC '11).

Durand-Lose, J. (2013). Irrationality is needed to compute with signal machines with only three speeds. *Dans* Bonizzoni, P., Brattka, V. et Löwe, B., éditeurs : *The Nature of Computation, 9th International Conference on Computability in Europe (CiE '13)*, LNCS. Springer. To appear.

Edgar, G. A. (1993). *Classic on Fractals*. Addison-Wesley.

Edgar, G. A. (2008). *Measure, Topology, and Fractal Geometry*. Springer, New-York.

Etesi, G. et Németi, I. (2002). Non-Turing computations via Malament–Hogarth space-time. *International Journal of Theoretical Physics*, 41(2):341–370.

Feynman, R. P. (1986). Quantum mechanical computers. *Foundations of Physics (Historical Archives)*, 16(6):507–531.

Fischer, P. C. (1965). Generation of primes by a one-dimensional real-time iterative array. *Journal of the Association for Computing Machinery*, 12(3):388–394.

Fredkin, E. et Toffoli, T. (1982). Conservative logic. *International Journal of Theoretical Physics*, 21(3):219–253.

Gandy, R. (1980). Church's thesis and principles for mechanisms. *Dans* Barwise, J., Keisler, J. H. et Kunen, K., éditeurs : *The Kleene Symposium*, volume 101 de *Studies in Logic and the Foundations of Mathematics*, pages 123–148. North-Holland Publishing Company, Amsterdam.

Gödel, K. (1931). On formally undecidable propositions of Principia Mathematica and related systems I. *Dans* Feferman, S., éditeur : *Kurt Gödel Collected works (1986)*, volume 1, pages 144–195. Oxford University Press. *Également réédité dans* [Davis 2004].

Goliaei, S. et Foroughmand-Araabi, M. H. (2012). Lower bounds on the complexity of the wavelength-based machine. *Dans* [UCNC 2012], pages 94–105.

Goliaei, S. et Jalili, S. (2011). Optical graph 3-colorability. *Dans* Dolev, S. et Oltean, M., éditeurs : *International Workshop on Optical Super Computing (OSC '10)*, numéro 6748 de LNCS, pages 16–22. Springer.

Goliaei, S. et Jalili, S. (2012). An optical solution to the 3-SAT problem using wavelength based selectors. *The Journal of Supercomputing*, 62(2):663–672.

Goto, E. (1966). Puzzles on automata. *Dans* Kitagawa, T., éditeur : *The Road to Information Science*, pages 67–92. Kyoristu Shuppan Publishing.

Gramss, T., Bornholdt, S., Gross, M., Mitchell, M. et Pellizzari, T. (1998). *Non-Standard Computation*. Wiley-VCH.

Haeseler, F., Peitgen, H.-O. et Skordev, G. (1995). Global analysis of self-similarity features of cellular automata: selected examples. *Physica D*, 86:64–80.

Hagiya, M. (2005). Discrete state transition systems on continuous space-time: a theoretical model for amorphous computing. *Dans* Calude, C. S., Dinneen, M. J., Paŭn, G., Pérez-Jiménez, M. et Rozenberg, G., éditeurs : *4th International Conference on Unconventional Computation (UC '05)*, numéro 3699 de LNCS, pages 117–129. Springer.

Hamkins, J. D. et Lewis, A. (2000). Infinite time Turing machines. *Journal of Symbolic Logic*, 65(2):567–604.

Handbook of Nat. Comp. (2012). Rozenberg, G. et Bäck, T. et Kok, J.N., éditeurs : *Handbook of Natural Computing (4 volumes)*. Springer-Verlag.

Hardy, G. H. et Wright, E. M. (1960). *An Introduction to the Theory of Numbers (4th edition)*. Oxford University Press.

Hedlund, G. A. (1969). Endomorphisms and automorphisms of the shift dynamical systems. *Mathematical Systems Theory*, 3(4):320–375.

Hogarth, M. (1994). Non-Turing computers and non-Turing computability. *Dans* Hull, D., Forbes, M. et Burian, R., éditeurs : *Proceedings of the Biennial Meeting of the Philosophy of Science Association (PSA '94)*, volume 1, pages 126–138. JSTOR.

Huckenbeck, U. (1989). Euclidian geometry in terms of automata theory. *Theoret. Comput. Sci.*, 68(1):71–87.

Huckenbeck, U. (1991). A result about the power of geometric oracle machines. *Theoret. Comput. Sci.*, 88(2):231–251.

Ito, T., Inokuchi, S. et Mizoguchi, Y. (2008). An abstract collision system. *Dans* Adamatzky, A., Alonso-Sanz, R., Lawniczak, A. T., Martínez, G. J., Morita, K. et Worsch, T., éditeurs : *International Workshop on Theory and Applications of Cellular Automata (AUTOMATA '08)*, pages 339–355. Luniver Press, UK. Available at `http://uncomp.uwe.ac.uk/free-books/automata2008reducedsize.pdf`.

Jacopini, G. et Sontacchi, G. (1990). Reversible parallel computation: an evolving space-model. *Theoret. Comput. Sci.*, 73(1):1–46.

Kleene, S. C. (1936). General recursive functions on natural numbers. *Mathematische Annalen*, 112:727–742. *Réédité dans* [Davis 2004].

Ko, K.-I. (2003). Computational complexity of fractals. *Dans* Downey, R., Decheng, D., Tung, S. P., Qiu, Y. H. et Yasugi, M., éditeurs : *Proceedings of the 8th Asian Logic Conference (ALC '02)*, pages 252–269. World Scientific Singapore.

Lafitte, G. (2001). How powerful are infinite time machines? *Dans* Freivalds, R., éditeur : *13th International Symposium on Fundamentals of Computation Theory (FCT '01)*, numéro 2138 de LNCS, pages 252–263. Springer.

Lathrop, J., Lutz, J. H. et Summers, S. M. (2009). Strict self-assembly of discrete Sierpinski triangles. *Theoret. Comput. Sci.*, 410(4-5):384–405.

Lathrop, J. I., Lutz, J. H., Patitz, M. J. et Summers, S. M. (2011). Computability and complexity in self-assembly. *Theory of Computing Systems*, 48(3):617–647.

Levin, L. (1973). Universal'nye perebornye zadachi (Universal search problems). *Problems of Information Transmission*, 9(3):265–266.

Levorato, V. et Senot, M. (2010). Discrete signal machines via pretopology—one step from signal machines to cellular automata. *Dans* Bordhin, H., Freund, R., Hinze, T., Holzer, M., Kutrib, M. et Otto, F., éditeurs : *2nd International Workshop on Non-Classical Models of Automata and Applications (NCMA '10)*, numéro 263 de OCG Books, pages 127–140. Austrian Computer Society.

Lindgren, K. et Nordahl, M. G. (1990). Universal computation in simple one-dimensional cellular automata. *Complex Systems*, 4:299–318.

Mandelbrot, B. B. (1967). How long is the coast of Britain? Statistical self-similarity and fractional dimension. *Science*, 156:636–638.

Mandelbrot, B. B. (1982). *Fractal Geometry of Nature*. Freeman, Wilfried H., New-York.

Margenstern, M. et Morita, K. (2001). NP-problems are tractable in the space of cellular automata in the hyperbolic plane. *Theoret. Comput. Sci.*, 259(1–2):99–128.

Mazoyer, J. (1987). A six-state minimal time solution to the Firing Squad Synchronization Problem. *Theoret. Comput. Sci.*, 50(2):183–238.

Mazoyer, J. (1996). Computations on one-dimensional cellular automata. *Annals of Mathematics and Artificial Intelligence*, 16:285–309.

Mazoyer, J. et Terrier, V. (1999). Signals in one-dimensional cellular automata. *Theoret. Comput. Sci.*, 217(1):53–80.

Mazoyer, J. et Yunès, J.-B. (2012). Computations on cellular automata. *Dans* [Handbook of Nat. Comp. 2012], chapitre 5, pages 159–188.

Minsky, M. (1967). *Computation: Finite and Infinite Machines (1st edition)*. Prentice-Hall International, New Jersey.

Moore, C. (1996). Recursion theory on the reals and continuous-time computation. *Theoret. Comput. Sci.*, 162(1):23–44.

Mycka, J., Coelho, F. et Costa, J. F. (2006). The Euclid Abstract Machine: trisection of the angle and the Halting problem. *Dans* Calude, C. S., Dinneen, M. J., Păun, G., Rozenberg, G. et Stepney, S., éditeurs : *5th International Conference on Unconventional Computation (UC '06)*, numéro 4135 de LNCS, pages 195–206. Springer.

Naughton, T. J. et Woods, D. (2005). An optical model of computation. *Theoret. Comput. Sci.*, 334(1-3):227–258.

Németi, I. et Dávid, G. (2006). Relativistic computers and the Turing barrier. *Applied Mathematics and Computation*, 178(1):118–142.

Ollinger, N. et Richard, G. (2009). Automata on the plane vs particles and collisions. *Theoret. Comput. Sci.*, 410(27-29):2767–2773.

Ollinger, N. et Richard, G. (2011). Four states are enough! *Theoret. Comput. Sci.*, 412(1-2):22–32.

Papadimitriou, C. (1994). *Computational Complexity*. Addison-Wesley.

Patitz, M. J. et Summers, S. M. (2010). Self-assembly of discrete self-similar fractals. *Natural Computing*, 9(1):135–172.

Păun, G. (2000). Computing with membranes. *Journal of Computer and System Sciences*, 61(1):108–143.

Păun, G. (2001). P-systems with active membranes: attacking NP-Complete problems. *Journal of Automata, Languages and Combinatorics*, 6(1):75–90.

Păun, G. et Rozenberg, G. (2002). A guide to membrane computing. *Theoret. Comput. Sci.*, 287(1):73–100.

Păun, G., Rozenberg, G. et Salomaa, A. (1998). *DNA Computing, New Computing Paradigms*, volume 287 de *EATCS Series*. Springer-Verlag.

Post, E. (1943). Formal reductions of the combinatorial decision problem. *American Journal of Mathematics*, 65(2):197–215. *Réédité dans* [Davis 2004].

Potgieter, P. H. (2006). Zeno machines and hypercomputation. *Theoret. Comput. Sci.*, 358(1): 23–33.

Reif, J. H., Tygar, J. D. et Yoshida, A. (1990). The computability and complexity of optical beam tracing. *Dans 31st Annual Symposium on Foundations of Computer Science (FOCS '90)*, pages 106–114. IEEE Computer Society.

Richard, G. (2008). *Systèmes de particules et collisions discrètes dans les automtates cellulaires*. Thèse de doctorat, École Normale Supérieure de Lyon. En français.

Rothemund, P. W. K. et Winfree, E. (2000). The program-size complexity of self-assembled squares (extended abstract). *Dans* Yao, F. F. et Luks, E. M., éditeurs : *32nd Annual ACM Symposium on Theory of Computing (STOC '00))*, pages 459–468. ACM.

Russell, B. A. (1936). The limits of empiricism. *Proceedings of the Aristotelian Society*, 36:131–150.

Schaller, M. et Svozil, K. (2010). Zeno squeezing of cellular automata. *International Journal of Unconventional Computing*, 6(5):399–416.

Shannon, C. E. (1941). Mathematical theory of the differential analyzer. *Journal of Mathematics and Physics of MIT*, 20:337–354.

Shor, P. W. (1994). Algorithms for quantum computation: discrete logarithms and factoring. *Dans 35th Annual Symposium on Foundations of Computer Science (FOCS '94)*, pages 124–134. IEEE Computer Society.

Siegelmann, H. T. et Sontag, E. D. (1995). On the computational power of neural nets. *Journal of Computer and Systems Science*, 50(1):132–150.

Sierpiński, W. (1915). Sur une courbe dont tout point est un point de ramification. *Comptes Rendus de l'Académie des Sciences de Paris*, 160:302–305.

Sosík, P. (2003). Solving a PSPACE-Complete problem by P-systems with active membranes. *Dans* Cavaliere, M., Martín-Vide, C. et Păun, G., éditeurs : *Brainstorming Week on Membrane Computing*, pages 305–312. Tarragona: Universidad Rovira i Virgili.

Stannett, M. (2006). The case for hypercomputation. *Applied Mathematics and Computation*, 178(1):8–24.

Stannett, M. (2013). Computation and spacetime structure. *International Journal of Unconventional Computing*, 9(1-2):173–184.

Stevens, W. M. (2011). Computing with planar toppling domino arrangements. *Dans* Calude, C. S., Kari, J., Petre, I. et Rozenberg, G., éditeurs : *10th International Conference on Unconventional Computation (UC '11)*, numéro 6714 de LNCS, pages 224–233. Springer.

Stockmeyer, L. J. et Meyer, A. R. (1973). Word problems requiring exponential time: preliminary report. *Dans* Aho, A. V., Borodin, A., Constable, R. L., Floyd, R. W., Harrison, M. A., Karp, R. M. et Strong, H. R., éditeurs : *5th ACM Symposium on Theory of Computing (STOC '73)*, pages 1–9. ACM.

Takeuti, I. (2005). Transition systems over continuous time-space. *Electronic Notes in Theoret. Comput. Sci.*, 120:173–186.

Turing, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42(2):230–265. *Réédité dans* [Davis 2004].

Turing, A. M. (1939). Systems of logic based on ordinals. *Proceedings of London Mathematical Society*, 45(2):161–228.

UCNC (2012). Durand-Lose, J. et Jonoska, N., éditeurs : *Proceedings of the 11th International Conference on Unconventional Computation and Natural Computation*, numéro 7445 de LNCS. Springer.

van Emde Boas, P. (1990). Machine Models and Simulation. *Dans* van Leeuwen, J., éditeur : *Handbook of Theoretical Computer Science, Algorithms and Complexity*, volume A, chapitre I, pages 1–66. Elsevier North-Holland and MIT Press.

Vollmer, H. (1999). *Introduction to circuit complexity—a uniform approach.* Texts in Theoretical Computer Science. Springer.

von Koch, H. (1904). On a continuous curve without tangents, constructible from elementary geometry. *Dans* [Edgar 1993], pages 25–45.

von Neumann, J. (1966). Theory of self-reproducing automata. *Dans* Burks, A. W., éditeur : *Theory of Self-Reproducing Automata.* University of Illinois Press.

Wang, H. (1960). Proving theorems by pattern recognition I. *Communications of the ACM*, 3(4):220–234.

Weihrauch, K. (2000). *Computable Analysis: an introduction.* Springer-Verlag.

Weyl, H. H. (1927). *Philosophie der Mathematik und Naturwissenschaft (Philosophy of Mathematics and Natural Science).* R. Oldenbourg, Munich. *Traduit et réédité* par Princeton University Press, 1949.

Willson, S. J. (1984). Growth rates and fractal dimensions in cellular automata. *Physica D*, 10:69–74.

Winfree, E. (1998). *Algorithmic self-assembly of DNA.* Thèse de doctorat, California Institute of Technology.

Wolfram, S. (1984). Computation theory of cellular automata. *Communications in Mathematical Physics*, 96(1):15–57.

Wolfram, S. (2002). *A New Kind of Science.* Wolfram Media.

Yunès, J.-B. (1994). Seven-state solutions to the Firing Squad Synchronization Problem. *Theoret. Comput. Sci.*, 127(2):313–332.

Yunès, J.-B. (2007). Simple new algorithms which solve the Firing Squad Synchronization Problem: a 7-states $4n$-steps solution. *Dans* Durand-Lose, J. et Margenstern, M., éditeurs : *5th International Conference on Machines, Computations and Universality (MCU '07)*, numéro 4664 de LNCS, pages 316–324. Springer.

Yunès, J.-B. (2008). Goto's construction and Pascal's triangle: new insights into cellular automata synchronization. *Dans* Durand, B., éditeur : *1st Symposium on Cellular Automata « Journées Automates Cellulaires » (JAC '08)*, pages 195–203. MCCME Publishing House, Moscow.

Yunès, J.-B. (2012). Grids and universal computations on one-dimensional cellular automata. *Natural Computing*, 11(2):303–309.

Zerjatke, T. et Sturm, M. (2011). Solving a PSPACE-complete problem by gene assembly. *Journal of Logic and Computation*, pages 1–12. Available at `http://logcom.oxfordjournals.org/content/early/2011/12/20/logcom.exr052`.

Ziegler, M. (2009). Physically-relativized Church-Turing hypotheses: physical foundations of computing and complexity theory of computational physics. *Applied Mathematics and Computation*, 215(4):1431–1447.

Maxime SENOT

Modèle géométrique de calcul : fractales et barrières de complexité

Résumé. Les modèles géométriques de calcul permettent d'effectuer des calculs à l'aide de primitives géométriques. Parmi eux, le modèle des *machines à signaux* se distingue par sa simplicité, ainsi que par sa puissance à réaliser efficacement de nombreux calculs. Nous nous proposons ici d'illustrer et de démontrer cette aptitude, en particulier dans le cas de processus massivement parallèles.
Nous montrons d'abord à travers l'étude de *fractales* que les machines à signaux sont capables d'une utilisation massive et parallèle de l'espace.
Une méthode de *programmation géométrique modulaire* est ensuite proposée pour construire des machines à partir de composants géométriques de base — les *modules* — munis de certaines fonctionnalités. Cette méthode est particulièrement adaptée pour la conception de calculs géométriques parallèles.
Enfin, l'application de cette méthode et l'utilisation de certaines des structures fractales résultent en une résolution géométrique de problèmes difficiles comme les problèmes de satisfaisabilité booléenne **SAT** et **Q-SAT**. Ceux-ci, ainsi que plusieurs de leurs variantes, sont résolus par machines à signaux avec une complexité en temps intrinsèque au modèle, appelée *profondeur de collisions*, qui est polynomiale, illustrant ainsi l'efficacité et le pouvoir de calcul parallèle des machines à signaux.

Mots clés. Modèle de calcul, calcul non-conventionnel, calcul géométrique, machines à signaux, fractales, complexité de problèmes.

Geometrical model of computation: fractals and complexity gaps

Abstract. Geometrical models of computation allow to compute by using geometrical elementary operations. Among them, the *signal machines* model distinguishes itself by its simplicity, along with its power to realize efficiently various computations. We propose here an illustration and a study of this ability, especially in the case of massively parallel processes.
We show first, throught a study of *fractals*, that signal machines are able to make a massive and parallel use of space.
Then, a framework of *geometrical modular programmation* is proposed for designing machines from basic geometrical components —called *modules*— supplied with given functionnalities. This method fits particulary with the conception of geometrical parallel computations.
Finally, the joint use of this method and of fractal structures provides a geometrical resolution of difficult problems such as the boolean satisfiability problems **SAT** and **Q-SAT**. These ones, as well as several variants, are solved by signal machines with a model-specific time complexity, called *collisions depth*, which is polynomial, illustrating thus the efficiency and the parallel computational abilities of signal machines.

Keywords. Model of computation, unconventional computing, geometrical computation, signal machines, fractals, computational complexity.

Laboratoire d'Informatique Fondamentale d'Orléans
Bâtiment 3IA, rue Léonard de Vinci, B.P. 6759
45067 ORLEANS cedex 2, FRANCE