Calculer géométriquement sur le plan - machines à signaux -

Jérôme Durand-Lose

HABILITATION
DIRIGER DES RECHERCHES

PRÉSENTÉE À

L'UNIVERSITÉ de NICE

ÉCOLE DOCTORALE STIC

Par **Jérôme Durand-Lose**

SPÉCIALITÉ : INFORMATIQUE

Calculer géométriquement sur le plan
— machines à signaux —

Soutenue le : 13 décembre 2003

Après avis des rapporteurs :

Jean-Paul Delahaye	Professeur, U. Sciences et Technologies de Lille
Étienne Grandjean	Professeur, U. Caen
Maurice Nivat	Professeur, U. Paris VII

Devant la commission d'examen composée de :

Michel Cosnard	Professeur, U. Nice
Jean-Paul Delahaye	Professeur, U. Sciences et Technologies de Lille
Jean-Marc Fédou ...	Professeur, U. Nice
Étienne Grandjean	Professeur, U. Caen
Jacques Mazoyer ...	Professeur, IUFM Lyon
Géraud Sénizergues	Professeur, U. Bordeaux I

– 2003 –

Resumé

Ce mémoire se place dans l'étude des *modèles du calcul continus*. Nous y montrons que la géométrie plane permet de calculer. Nous définissons un calcul géométrique et utilisons la continuité de l'espace et du temps pour stocker de l'information au point de provoquer des accumulations.

Dans le monde des automates cellulaires, on parle souvent de *particules* ou de *signaux* (qui forment des lignes discrètes sur les diagrammes espace-temps) tant, pour analyser une dynamique que, pour concevoir des automates cellulaires particuliers. Le point de départ de nos travaux est d'envisager des versions continues de ces signaux. Nous définissons un modèle de calcul continu, les *machines à signaux*, qui engendre des figures géométriques suivant des règles strictes. Ce modèle peut se comprendre comme une extension continue des automates cellulaires. Le mémoire commence par une présentation des automates cellulaires et des particules. Nous faisons ensuite une classification des différents modèles de calcul existants et mettons en valeur leurs aspects discrets et continus. À notre connaissance, notre modèle est le seul à temps et espace continus mais à valeurs et mises à jour discrètes.

Dans la première partie du mémoire, nous présentons ce modèle, les machines à signaux, et montrons comment y mener tout calcul au sens de Turing (par la simulation de tout automate à deux compteurs). Nous montrons comment modifier une machine de manière à réaliser des transformations géométriques (translations, homothéties) sur les diagrammes engendrés. Nous construisons également les itérations automatiques de ces constructions de manière à contracter le calcul à une bande (espace borné) puis, à un triangle (temps également borné).

Dans la seconde partie du mémoire, nous cherchons à caractériser les points d'accumulation. Nous reformulons de manière topologique les diagrammes espace-temps : pour chaque position, la valeur doit correspondre au voisinage sur un ouvert suffisamment petit. Muni de cet outil, nous regardons les plus simples accumulations possibles (les singularités isolées) et proposons un critère pour y prolonger le calcul ; mais le déterminisme peut être perdu dans le cône d'influence. Enfin, en construisant pour tout automate à deux compteurs une machine à signaux et une configuration initiale simulant l'automate pour toutes les valeurs possibles, nous montrons que le problème de la prévision de l'apparition d'une accumulation est Σ_2^0-complet.

Le mémoire se conclut par la présentation de nombreuses perspectives de recherches.

Mots-clefs

Automates cellulaires, géométrie, modèle du calcul continu, machine à signaux et signaux.

Abstract

This memoir for accreditation to supervise research deals with continuous models of computation. It is proven that plane geometry provides computation power.

In the world of cellular automata, one often hears of particles or signals (that forms discrete lines on space-time diagrams), both for analyzing some dynamics and for conceiving particular cellular automata. The starting point of this work is to consider continuous counterparts of signals. We define a continuous model, *signal machines*, that generates geometrical figures complying strict rules. This model can be understood as a continuous extension of cellular automata. The memoir begins with a brief introduction to cellular automata and signals, and a classification of models of computation that stresses on their continuous and discrete facets. To our knowledge, our model is the only one with continuous space and time but discrete values and updating rules.

In the first part of the memoir, we present the model, signal machines, and show that it has Turing computing capability (through 2-counter automata simulation). We show how to modify a machine in order to make geometrical transformations (translations and homothecies) on the generated space-time diagrams. We also construct iterated modifications that constrain calculus to ribbons (bounded space) and then to triangles (time is also bounded).

In the second part, we try to characterize accumulation points. Space-time diagrams are reformulated topologically: each position must correspond to its immediate open neighborhood. Equipped with this tool, we investigate the simplest possible accumulations (isolated ones) and provide a criterion for extending the computation in such a case (although determinism can be lost inside the influence cone). Finally, we construct, for any 2-counter automaton, a machine and an initial configuration that simulate the automaton for each possible initial value. Using this construction, we prove that predicting the outbreak of an accumulation is Σ_2^0-complete.

The memoir ends with the presentation of research prospects.

Key-words

Cellular automata, geometry, continuous models of computation, signal machines, signals.

Remerciements

Tout d'abord, je tiens à remercier vivement Michel Cosnard qui m'a fait l'honneur de présider le jury d'une main de maître.

Je tiens à remercier tout aussi chaudement Jean-Paul Delahaye, Étienne Grandjean et Maurice Nivat pour leurs minutieux et constructifs rapports ainsi que pour les questions profondes et pertinentes qu'ils m'ont adressées.

Je remercie tout autant Jean-Marc Fédou, Jacques Mazoyer et Géraud Sénizergues pour la richesse des dialogues que nous avons eus.

Plus globalement, je remercie toutes les personnes présentes en ce samedi matin de soutenance (et qui n'ont pas cédé à l'appel du lit ou des courses de Noël).

Cette HDR ne s'est pas écrite en une nuit ; c'est le résultat de plusieurs années de travail tant scientifique que pédagogique et administratif (et secrétarial, technique, manutentionnaire...). Je tiens donc aussi à remercier tous les membres de

- l'équipe MC2, et par delà tout le LIP, pour son accueil chaleureux et son émulation scientifique,
- l'équipe RÉCIF, et par delà tout l'I3S, pour sa très bonne et sereine ambiance,
- le Département d'Informatique de l'UFR Sciences où malgré son volume le travail s'effectue dans la bonne humeur et la complicité.

Par delà encore, je remercie tout ceux, amis et famille, disparus, présents et à venir, sans lesquels la vie serait sans grand intérêt.

Table des matières

Table des figures

Chapitre 1

Introduction

Ce mémoire se place dans l'étude des *modèles du calcul continus*. Nous y montrons que la géométrie plane permet de calculer. Nous définissons un calcul géométrique et utilisons la continuité de l'espace et du temps pour stocker de l'information au point de provoquer des accumulations.

Dans le monde des automates cellulaires, on parle souvent de *particules* ou de *signaux* (qui forment des lignes discrètes sur les diagrammes espace-temps) tant, pour analyser une dynamique que, pour concevoir des automates cellulaires particuliers. Le point de départ de nos travaux est d'envisager des versions continues de ces signaux. Nous définissons un modèle de calcul continu, les *machines à signaux*, qui engendre des figures géométriques suivant des règles strictes. Ce modèle peut se comprendre comme une extension continue des automates cellulaires. .

Dans cette introduction, nous présentons les automates cellulaires (un modèle du calcul massivement parallèle, profondément discret) et donnons quelques exemples de particules et de signaux discrets à l'origine de notre modèle. Après un rapide état de l'art sur ce que nous connaissons de plus proche de notre approche, nous présentons ce modèle, nos résultats et l'organisation de ce mémoire.

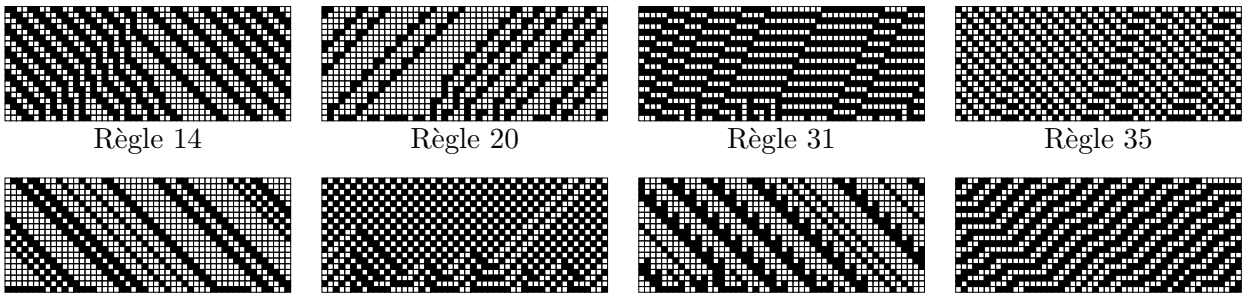
1.1 Automates cellulaires

Les automates cellulaires sont discrets par nature : l'espace est discrétisé en cellules (typiquement sur $\mathbb{Z}^d$), chaque cellule ne peut être que dans un nombre fini d'états et ne perçoit qu'une partie finie des cellules qui l'entourent. La mise à jour est faite de manière itérative, chaque cellule changeant d'état en fonction de ce qu'elle perçoit autour d'elle. Toutes les cellules sont mises à jour simultanément et en suivant les mêmes règles. Le processus est synchrone, uniforme et local.

Les automates cellulaires sont très étudiés depuis les années 1950 [Ula52]. En plus des études théoriques sur automates cellulaires eux-mêmes, on distingue deux façons de les aborder : pour simuler (un système existant) ou pour concevoir.

Dans le premier cas, le but est de simuler sur ordinateur des systèmes réels (qu'ils soient physiques, biologiques, économiques...). Lors de telles simulations, on observe assez souvent

des *signaux* (ou des *particules*) dans les diagrammes espace-temps. Ces signaux peuvent aussi bien être des artefacts de la simulation que correspondre à des phénomènes réels : de leur étude, on déduit des propriétés du système étudié. Par exemple, si l'on prend les diagrammes espace-temps (les itérations successives sont mises les unes sur les autres) de différents automates cellulaires élémentaires (deux états et chaque cellule ne perçoit les états que des deux cellules les plus proches), on peut observer des « signaux » se déplaçant comme sur les différents exemples de la Fig. 1.1 ou de la Figure 1.2 [BNR91, HSC01, JSS02, Siw01] (sous la dénomination de particule ou soliton).



(Les temps vont de bas en haut)

FIG. 1.1 – Différents diagrammes espace-temps d'automates cellulaires élémentaires.

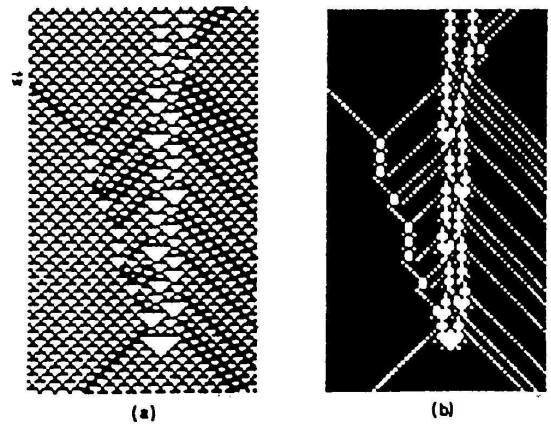


FIG. 7. Rule 54. (a) Annihilation of the radiating particle. (b) The same as (a) with the mapping defined in Fig. 6.

(a) [BNR91, Fig. 7]

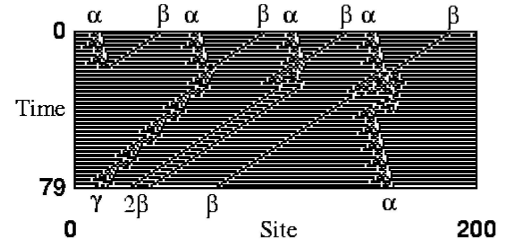


FIG. 7. The four different (out of 14 possible) interaction products for the $\alpha + \beta$ interaction.

(b) [HSC01, Fig. 7]

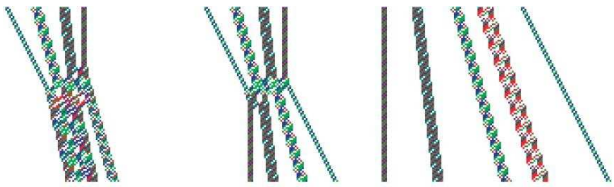


Figure 5. Two collisions of filtrons, and five free filtrons supported by the FPS model; ST diagram applies $q = 1$.

(c) [Siw01, Fig. 5]

(Les temps vont de haut en bas)

FIG. 1.2 – Exemples de particules.

Dans le second cas, on conçoit une machine (ou un circuit) massivement parallèle avant de la réaliser, ou un automate cellulaire particulier ayant certaines propriétés. Le but n'est plus

d'étudier un système mais de concevoir un système menant à bien une tâche prédéterminée. Il est souvent nécessaire de transporter de l'information d'un endroit à un autre. Pour cela, on met généralement en place des signaux véhiculant de l'information. Il arrive même que tout soit défini avec des signaux, ceux-ci définissant la structure comme les données [DL97, Fis65, LN90, Maz96a]. Éventuellement ce sont les signaux par eux-mêmes qui sont étudiés [DM02, MT99]. Les figures 1.3 à 1.6 montrent différentes illustrations de ces cas.

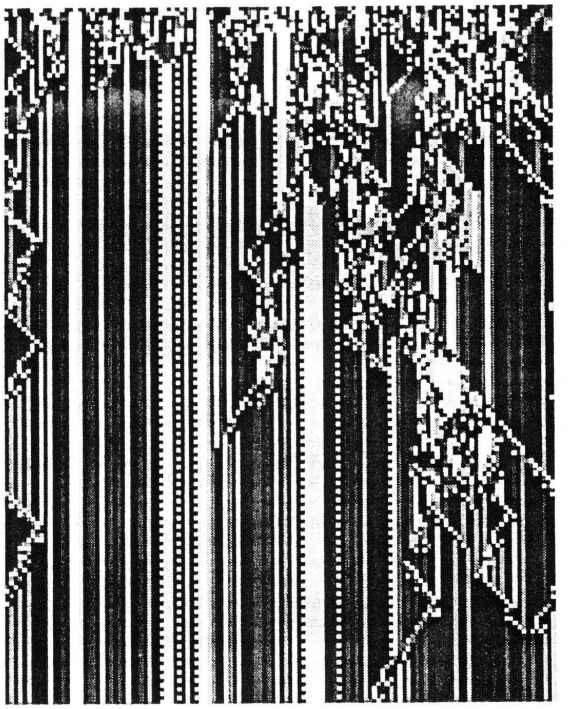


Figure 3: A simulation of the $k = 7$, $r = 1$ universal CA of table 3 for an uncorrelated initial state (with a density of blanks equal to 0.76). Symbols y, 0, 1, A, B, $\sqcup$, and T are represented by

(a) [LN90, Fig. 3]

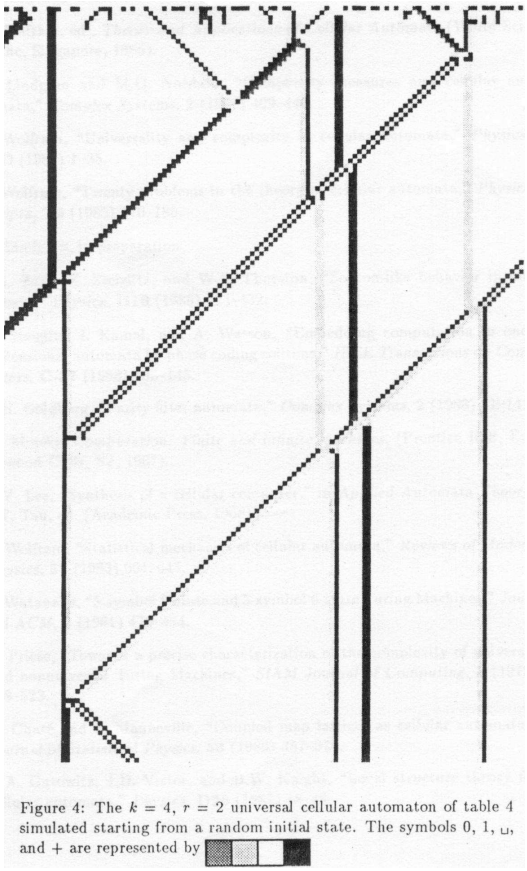


Figure 4: The $k = 4$, $r = 2$ universal cellular automaton of table 4 simulated starting from a random initial state. The symbols 0, 1, $\sqcup$, and + are represented by

(b) [LN90, Fig. 4]

(Les temps vont de haut en bas)

FIG. 1.3 – Utilisation de signaux pour construire une machine universelle [LN90].

De plus, différents algorithmes ont pour base des dessins *à la règle et au compas* qui sont ensuite « discrétisés » pour être exprimés sous forme d'automates cellulaires, de manière à ce que les diagrammes espace-temps de ceux-ci correspondent aux dessins. En particulier, il existe de nombreux algorithmes pour résoudre la *synchronisation d'une ligne de fusiliers (Firing Squad Synchronisation Problem)* [Maz96b]. Ce problème est celui de la synchronisation globale d'automates régulièrement disposés et « cadencés » à la même fréquence. Reformulé dans le contexte des automates cellulaires, cela correspond à avoir deux cellules distinguées (à une extrémité, un général qui démarre la synchronisation et, à l'autre, un délimiteur),

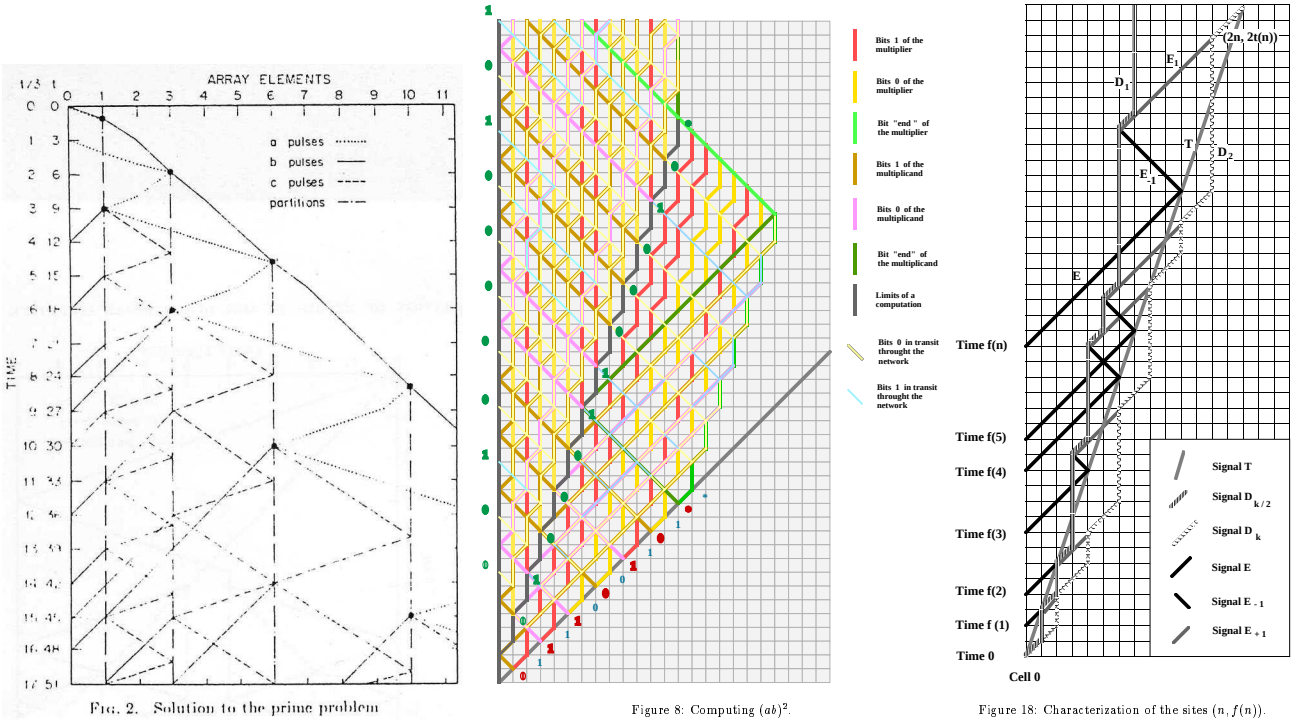


FIG. 2. Solution to the prime problem

Figure 8: Computing $(ab)^2$.

Figure 18: Characterization of the sites $(n, f(n))$.

(a) [Fis65, Fig. 2] (b) [Maz96a, Fig. 8] (c) [MT99, Fig. 18]

(Les temps vont de haut en bas à gauche et de bas en haut pour les deux autres)

FIG. 1.4 – Algorithmes géométriques.

toutes les autres cellules sont dans un état quiescent (elles y restent tant qu'aucune information ne leur parvient). Le but est que toutes les cellules passent en même temps dans un état qui n'aura pas été atteint précédemment. La plupart des algorithmes sont fondés sur des stratégies de type *diviser pour régner* (ce qui garantit un passage à l'échelle) : on coupe la configuration en parts « égales » et on recommence jusqu'à ne plus pouvoir couper, il y a alors synchronisation.

Un des plus anciens automates cellulaires conçu dans ce but est celui de GOTO [Got66] (*c.f.* Fig. 1.5). On voit bien, d'un côté, une approche purement géométrique et de l'autre sa discrétisation en automate cellulaire.

Une autre approche est celle de VARSHAVSKY, MARAKHOVSKY et PESCHANSKY. On va chercher la moitié, le quart, le huitième... (tant que l'on n'a pas encore atteint la taille des cellules) depuis le général. Du milieu et de l'autre extrémité, on va aussi chercher au quart, huitième... On recommence à chaque échelle comme on le voit sur la Fig. 1.6.

Nous ne rentrons pas plus avant dans ces algorithmes, ce qui est important pour nous est qu'ils ont des origines géométriques comme en témoignent les figures extraites des articles originels, avant une discrétisation de l'espace et du temps[1]. Notre souhait est de développer

[1]Il y a parfois utilisation de la discrétisation, comme sur la Fig. 1.6 pour engendrer une famille infinie de signaux de pente 2^{-n}.

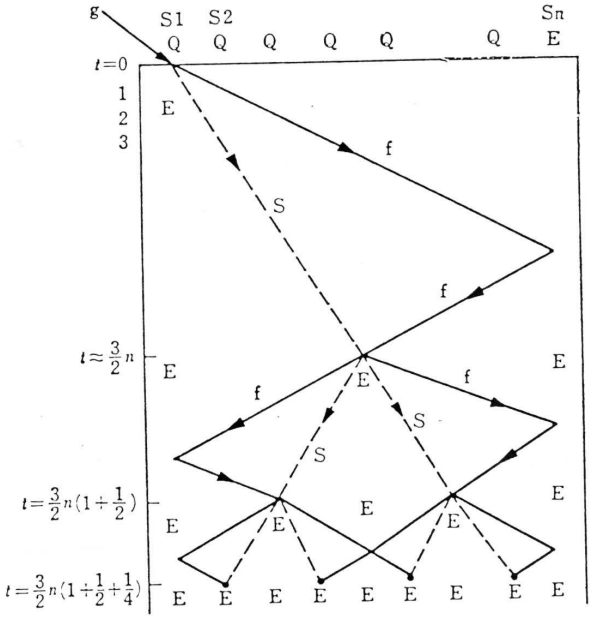
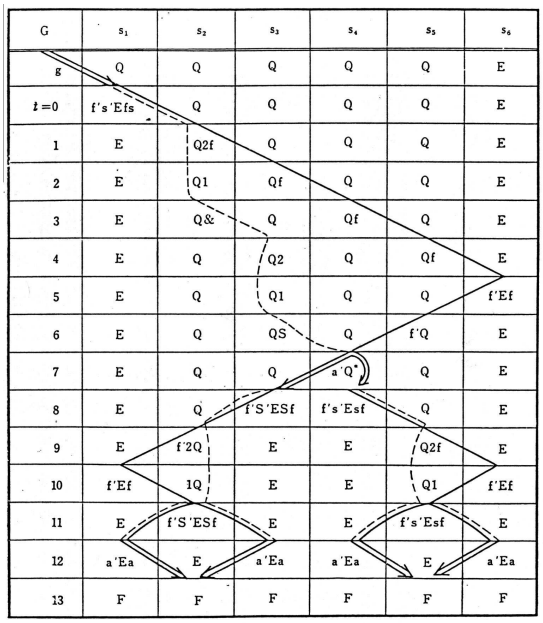


図 3·5 一斉射撃の問題（連続近似）



G	s_1	s_2	s_3	s_4	s_5	s_6
g	Q	Q	Q	Q	Q	E
t=0	f's'Efs	Q	Q	Q	Q	E
1	E	Q2f	Q	Q	Q	E
2	E	Q1	Qf	Q	Q	E
3	E	Q&	Q	Qf	Q	E
4	E	Q	Q2	Q	Qf	E
5	E	Q	Q1	Q	Q	f'Ef
6	E	Q	QS	Q	f'Q	E
7	E	Q	Q	a'Q'	Q	E
8	E	Q	f'S'ESf	f's'Esf	Q	E
9	E	f'2Q	E	E	Q2f	E
10	f'Ef	1Q	E	E	Q1	f'Ef
11	E	f'S'ESf	E	E	f's'Esf	E
12	a'Ea	E	a'Ea	a'Ea	E	a'Ea
13	F	F	F	F	F	F

図 3·6 一斉射撃解（n=6）

(a) [Got66, Fig. 3] (b) [Got66, Fig. 6]

(Les temps vont de haut en bas)

FIG. 1.5 – Algorithmes géométriques pour la synchronisation de [Got66].

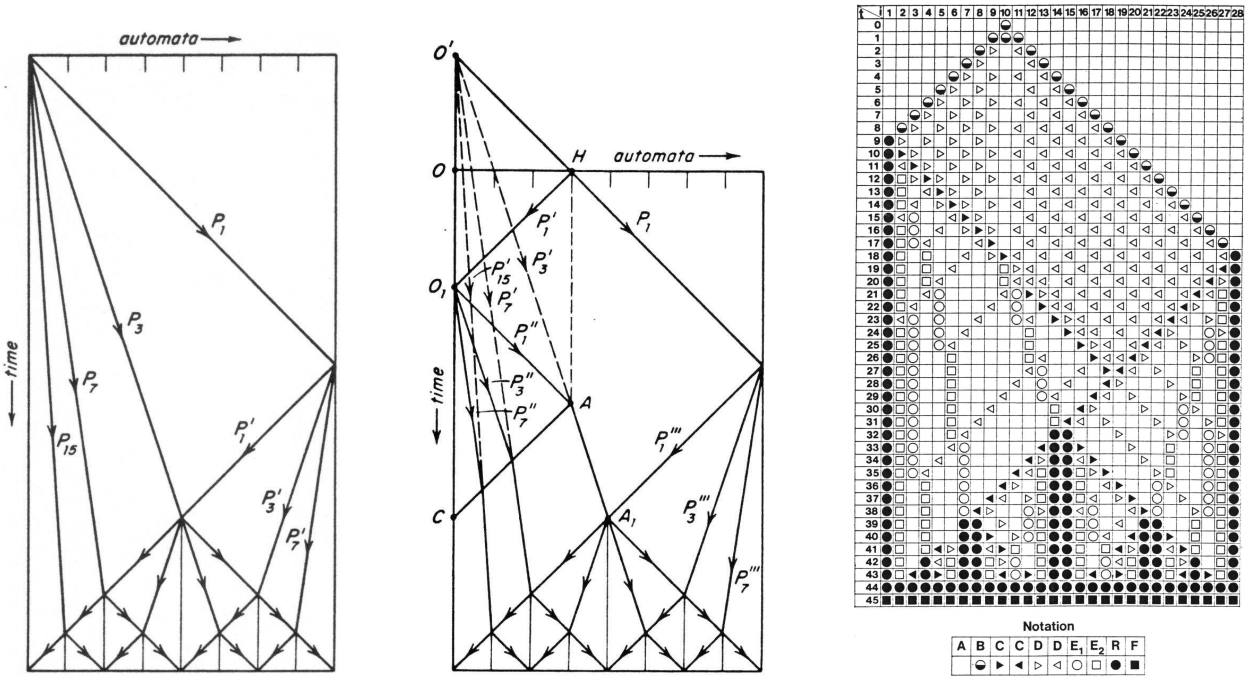
un modèle de calcul correspondant à ces dessins.

1.2 Signaux continus

Notre démarche est de prendre ces signaux sur automates cellulaires comme éléments de base et continuer à travailler sans chercher à les discrétiser. C'est une forme d'extension continue des automates cellulaires ou plutôt, la suppression de la discrétisation après une conception continue.

L'espace comme le temps ne sont alors plus discrets mais continus. Une configuration pour nous est un espace affine où sont localisés des signaux. Les signaux sont des instances de méta-signaux. Ces derniers caractérisent complètement les signaux : information véhiculée et vitesse de déplacement. Les signaux se déplacent de manière rectiligne uniforme, de leurs collisions naissent et disparaissent des signaux. Les règles de collisions sont entièrement définies au niveau des méta-signaux. Les diagrammes espace-temps sont des figures géométriques représentant les traces des signaux dans un espace d'une dimension supérieure (pour le temps).

Une autre façon d'envisager le modèle est de considérer des particules en mouvement rectiligne uniforme, dont les collisions sont instantanées et sans dimension.



(a) [VMP70, Fig. 1] (b) [VMP70, Fig. 2] (c) [VMP70, Fig. 3]

(Les temps vont de haut en bas)

FIG. 1.6 – Algorithmes géométriques pour la synchronisation de [VMP70].

1.3 État de l'art

À notre connaissance, il n'existe pas de travaux de recherche sur cette approche. Le plus proche que nous connaissons est celui des les *automates de Mondrian* de JACOPINI et SONTACCHI [JS90]. Ce modèle considère des polyèdres connexes (avec éventuellement des arêtes / faces internes) de volume fini sans point d'accumulation. Chaque élément du polyèdre (arêtes et faces de toutes dimensions) est coloré. Pour chaque couleur, il y a une boule de référence imposant les couleurs dans le voisinage. Pour être valide, le polyèdre doit, en chaque point, être localement identique à la boule de référence de la couleur du point.

Les auteurs définissent un critère de réversibilité pour les boules qui ne laisse pas de choix pour les autres faces incidentes. Ils montrent que, même en l'imposant, il est possible de définir, pour toute machine de Turing, des boules de référence telles que pour toute entrée de la machine, correspondant à une partie du polyèdre, on ne puisse compléter le polyèdre qu'en réalisant le calcul correspondant. Ce modèle est donc universel pour le calcul au sens de Turing.

Notre modèle est issu de traces de signaux et produit des diagrammes espace-temps qui ne sont pas sans rapport avec ceux de [JS90]. Nous n'utilisons une approche topologique que dans la seconde partie du mémoire et comparons alors les deux modèles (Sous-sect. 8.1.1 du Chap. 8).

1.4 Modèle et résultats

Nous définissons une *machine à signaux* comme un ensemble fini de méta-signaux et de règles correspondant aux collisions de deux ou plus de ces méta-signaux. Chaque signal est caractérisé par le méta-signal dont il est une instance (cela définit sa vitesse). Une configuration est la position de tous les signaux dans un espace affine ($\mathbb{R}$ dans tout le mémoire). Si deux ou plusieurs signaux se rencontrent, ils disparaissent et sont remplacés par de nouveaux signaux suivant la règle correspondante. Par exemple, la rencontre des méta-signaux $(\mathsf{bleu}, 2)$ et $(\mathsf{rouge}, -1)$ donne $(\mathsf{violet}, \frac{3}{2})$.

Nous nommons *diagramme espace-temps* la trace des signaux au cours d'un calcul. Les signaux allant en ligne droite et les mêmes rencontres provoquant les même collisions, on obtient un dessin géométrique régulier. La Figure 1.7 montre un exemple de diagramme avec des instances des trois méta-signaux du paragraphe précédent.

La Figure 1.8 montre des diagrammes beaucoup plus compliqués.

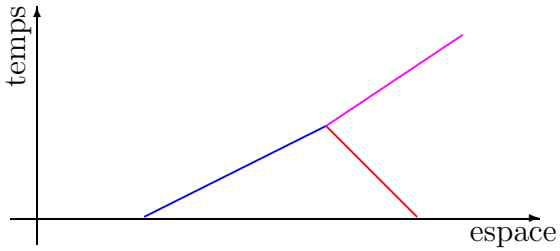


FIG. 1.7 – Exemple de diagramme espace-temps.

Ce modèle est capable de simuler tout automate à deux compteurs pour n'importe quelle entrée. Il est donc universel au sens de Turing.

Les diagrammes étant des dessins, nous avons cherché des opérateurs ayant une signification géométrique. Nous avons montré que l'on pouvait faire, au fil du calcul, des translations et des homothéties avec ou sans gel du calcul. De plus, nous avons pu mettre en place des signaux de manière à réitérer ces opérations, ce qui permet : de restreindre un calcul à une bande d'espace borné et, de contraindre une telle bande à une partie finie de l'espace-temps. Ceci fonctionne uniquement parce que l'espace comme le temps sont continus ; on retrouve l'idempotence de l'espace affine réel et de toute partie bornée.

Cette contraction de n'importe quel diagramme espace-temps à un triangle provoque l'apparition d'un point d'accumulation. Les machines ne sont pas prévues pour gérer les accumulations ; leur cadre n'est pas le plus approprié pour les étudier.

Pour gérer les accumulations, nous considérons les diagrammes espace-temps en tant que tels et les redéfinissons de manière topologique. Nous montrons qu'ils correspondent exactement à ceux définis par les traces des signaux. Équipé de cette caractérisation topologique, nous proposons une typologie pour les singularités isolées ainsi qu'un critère de validité pour les diagrammes en contenant. Le déterminisme peut être perdu en sortie d'accumulation (problème de *l'œuf et la poule*). Nous montrons que la prévision de l'apparition d'un point d'accumulation est Σ_2^0 dans le hiérarchie arithmétique (*i.e.* fortement indécidable).

Il existe de nombreuses directions de recherches pour prolonger ce travail, nous essayons de les présenter en fin de mémoire.

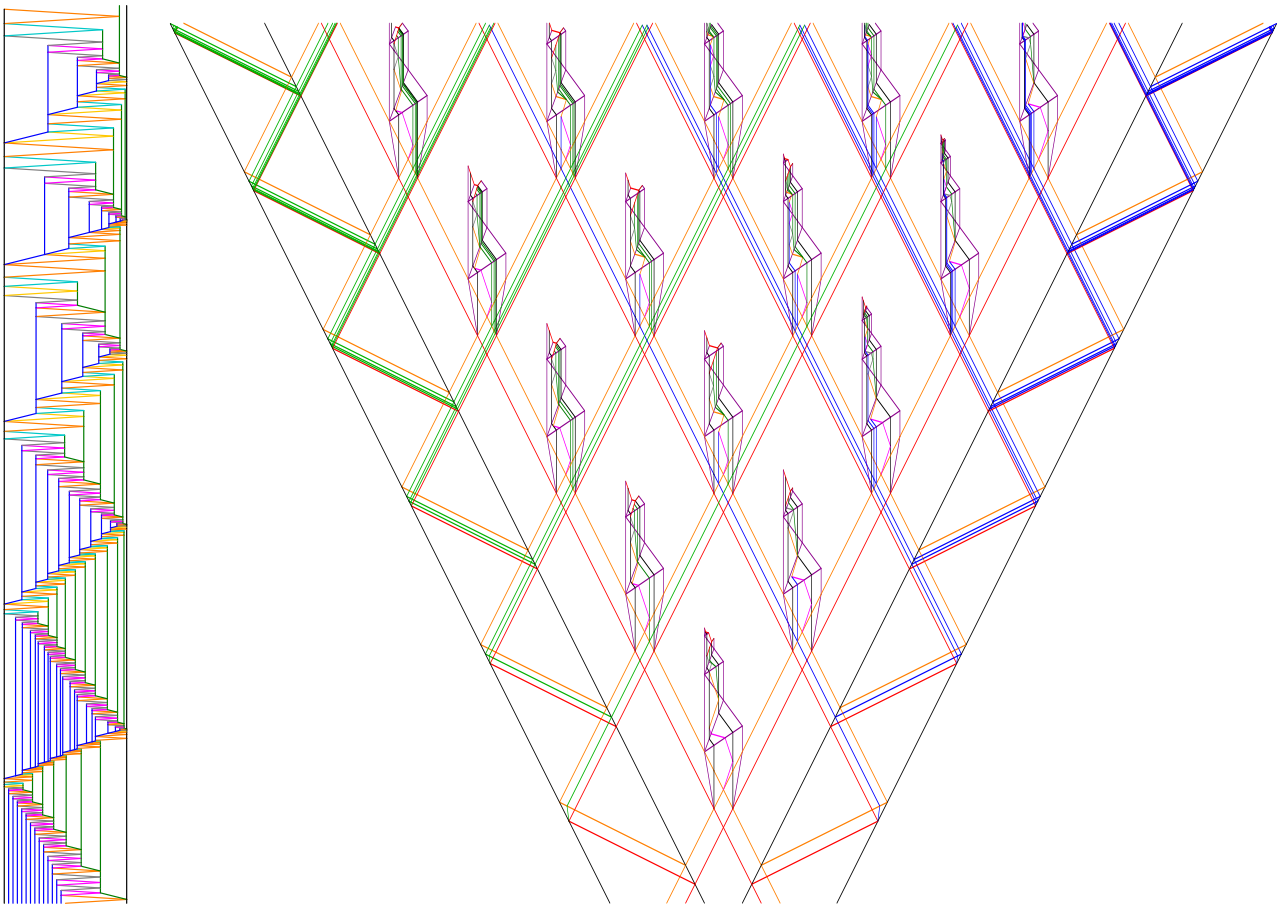


FIG. 1.8 – Diagrammes plus complexes.

1.5 Plan du mémoire

Le mémoire comporte deux parties. La première se concentre sur l'approche machine à signaux, capacité à calculer, modifications géométriques sans points d'accumulation. La seconde porte sur la reformulation topologique, la gestion des singularités et la difficulté de prévision d'une accumulation.

Dans un chapitre préliminaire (numéro 2), nous tentons de définir les concepts de calcul, de continu et de discret. Nous y faisons une classification des modèles existants ce qui nous permet de situer notre modèle. Les machines à signaux sont des modèles à dynamique locale et, à notre connaissance, elles forment le seul modèle à temps et support continus mais à valeurs et mises à jour discrètes.

La Partie I est composée comme suit :

dans le Chapitre 3, sont regroupées toutes les définitions nécessaires : méta-signaux, règles de collision, diagramme espace-temps...

dans le Chapitre 4, nous montrons comment simuler tout automate à deux compteurs par une machine à signaux, et prouvons ainsi les capacités à calculer au sens de Turing du modèle ;

dans le Chapitre 5, nous donnons une définition de simulation entre diagrammes espace-temps et montrons quelques simulations simples par des modifications statiques, *i.e.* de la machine ou de la configuration initiale ;

dans le Chapitre 6, nous montrons comment geler un calcul et le mettre en translation. En utilisant le théorème de Thalès, il est possible de contracter un calcul en temps comme en espace. En itérant à l'infini cette contraction, n'importe quel calcul peut être restreint à un espace borné ;

dans le Chapitre 7, nous montrons comment réaliser une homothétie sur une partie du diagramme espace-temps sans pour autant geler le calcul. Nous faisons une construction en abîme qui contracte tout calcul spatialement borné à un triangle borné. Tout calcul peut donc être restreint à un triangle ; nous retrouvons l'idempotence du plan et du triangle, mais nous créons volontairement un point d'accumulation.

Les machines à signaux ne gèrent pas les points d'accumulation. La Partie II propose une approche pour cela. Elle est composée comme suit :

dans le Chapitre 8, nous proposons une définition topologique des diagrammes espace-temps et prouvons qu'elle engendre les mêmes diagrammes que précédemment. Des singularités sont introduites pour définir les points où le diagramme ne peut être défini normalement ;

dans le Chapitre 9, nous établissons une typologie des singularités isolées et proposons un critère de validité pour prolonger les diagrammes espace-temps. Nous montrons avec des exemples que l'on peut perdre le déterminisme ;

dans le Chapitre 10, nous montrons que l'on ne peut prédire l'apparition de points d'accumulation. Ceci est fait en montrant, dans un premier temps, que, pour certaines constructions, leurs apparitions correspondent exactement à l'absence d'arrêt de machines à deux compteurs. Dans un second temps, nous faisons une construction qui a un point d'accumulation si, et seulement si, une machine à deux compteurs ne s'arrête pas pour au moins une entrée. La prédiction est donc Σ_2^0-difficile. Nous montrons que ce problème est dans Σ_2^0 et est donc Σ_2^0-complet.

Le dernier Chapitre (numéro 11) conclut le mémoire et propose diverses perspectives de recherche.

L'Annexe A présente le logiciel écrit pour réaliser toutes les figures. Toutes les constructions du mémoire y sont implantées.

L'Annexe B montre des cas où les singularités ne sont pas isolées et présente l'état de notre réflexion sur ce sujet. Elle ne fait pas partie du corps du mémoire car nous ne la considérons pas suffisamment aboutie.

Chapitre 2

Modèles de calcul et continuité

Avant de définir notre modèle, il nous a semblé utile de faire un tour d'horizon des modèles de calcul existant pour le situer.

Nous commençons par nous interroger sur le pourquoi et le comment du calcul. À partir de là, nous proposons une typologie des modèles de calcul de manière à discerner le continu[1] du discret. Nous illustrons ceci à l'aide de différents modèles de calcul.

2.1 Calculer

Nous considérons le calcul comme étant le moyen de parvenir, pas à pas, à un résultat à partir de données ou, de manière plus abstraite, comme le moyen d'extraire méthodiquement des informations d'autres informations.

Pourquoi calcule-t-on ? Pour compter (gestion, commerce), pour évaluer (prédiction, construction), pour apprécier (décision), pour mécaniser le raisonnement et traiter l'information (traduction, communication, et toutes les tâches de notre « société de l'information »). Dans ce tour d'horizon chronologique, se dessinent les différents sujets du calcul : nombres entiers puis décimaux (réels ?), les valeurs qualitatives et l'information au sens large (que nous ne tentons pas de définir).

Comment calcule-t-on ? En utilisant l'arithmétique, des abaques, des « recettes de cuisine », des procédés laborieux mais sûrs (algorithmes), des machines prévues à cet effet (bouliers, règles à calcul, ordinateurs). Au fur et à mesure des progrès technologiques, les demandes en calcul ont augmenté : données plus importantes, besoins de plus de rapidité, de précision, de robustesse... Conjointement, les éléments matériels pour calculer se sont aussi développés (roues crénelées, commutateurs électriques, lampes à vide, transistors). Alors que le calcul se mécanisait, philosophiquement, s'est posée la question de le définir.

Datons la définition formelle du calcul à 1936 avec les fonctions récursives de Gödel [Göd31], le λ-calcul de Church [Chu36b, Chu36a], les machines de Turing [Tur36] et les travaux de Post [Pos21, Pos36] et Kleene [Kle36b, Kle36a][2]. La première approche définit des ensembles de fonctions de manière inductive, algébrique, sans se soucier de

[1] Nous n'utilisons pas le terme « analogique », car nous le considérons trop restrictif en tant qu'antonyme de « digital ».

[2] Nous renvoyons à [Soa99] pour un historique de la notion de calculabilité.

leurs évaluations pratiques. Nous nommons cette approches *algébrique*. La seconde approche définit une classe de fonctions, mais fournit en plus des réductions (leur ordre étant laissé à l'utilisateur ou à un algorithme externe). En revanche, la troisième part de l'« activité humaine » : connaissance finie (états en nombre fini), quantité d'information non bornée (le ruban) qui n'est perçu que partiellement (case) mais est totalement accessible (par le mouvement de la tête). Une machine est un procédé *effectif* de calcul. Elle fait, de plus, référence implicitement au temps ; ce qui n'existe pas dans la première approche. Un *calcul* est une suite de configurations menant des données au résultat. Nous nommons cette approche *dynamique*.

Les modèles algébriques n'ont pas de dynamique : ils définissent des fonctions mais non leurs évaluations ; *e.g.* l'ordre d'évaluation des paramètres ou l'utilisation de la programmation dynamique ne font pas partie des fonctions récursives mais relève d'implantations.

Pour le λ-calcul, nous pouvons dire qu'il y a une dynamique dans le sens relation de *réécriture*. Le modèle prévoit des réécritures possibles et suppose des propriétés de confluence (théorèmes de Church-Rosser). En plus du λ-calcul, on peut classer parmi les modèles par réécriture les algorithmes de Markov, les machines de Kolmogorov et les systèmes de Post. À chaque fois, l'idée est de remplacer une partie de l'objet sur lequel porte le calcul (arbre syntaxique, mot ou graphe) en fonction de la partie remplacée et de règles. En λ-calcul, l'objet est un arbre, la réduction peut modifier un nombre non borné de feuilles. Dans les autre cas, le remplacement est local.

Si l'on considère une machine comme un système dynamique, un calcul est une *orbite*. Il est à noter que, d'une part, les notions de non déterminisme, de probabilisme et de parallélisme correspondent à cette approche et que, d'autre part, les définitions par système dynamique posent le problème de la définition des données et résultats : l'entrée (résultat) peut ne pas être présente dans la configuration initiale (finale) mais être fournie (engendrée) au cours du temps[3], ce qui pose un problème d'intégration entre la temporisation de l'entrée et le temps propre au système.

Parmi les systèmes dynamiques, nous voulons singulariser les systèmes par *propagation* finie du calcul. Ce sont plus que déterministes : le nombre d'opérations est fini et connu à l'avance ; de plus toute affectation est définitive, et l'ordre d'évaluation des opérations possibles ne change pas le résultat. Parmi ces modèles, se trouvent les circuits booléens, les réseaux de neurones non ré-entrants et le *straight line programming*.

2.2 Discret et continu

Ces deux notions sont relativement difficiles à cerner[4]. On considère sans contestation $\mathbb{N}$ et tout ensemble fini comme discrets et, $\mathbb{R}$ comme continu. Voici deux exemples limites. Le premier est l'ensemble $\mathbb{Q}$ muni de la topologie engendrée par l'ordre ; par l'intermédiaire des coupures de Dedekin, on retrouve toute la richesse de la topologie de $\mathbb{R}$. Le second est plus ambiguë : un ordinal dénombrable (ensemble dénombrable avec des points d'accumulation)

[3] C'est déjà le cas avec un système d'exploitation ou temps-réel ; car on ignore les entrées futures.

[4] Nous renvoyons à divers articles de [SS90]. Ce colloque a réuni de mathématiciens, physiciens et philosophes pour une réflexion sur le concept du continu. Dans divers communications, le continu est relié ou opposé au fini ou au discret.

peut être considéré comme continu avant chaque ordinal limite et, discret ailleurs, et donc, être *hybride* (à la fois l'un et l'autre selon l'élément). La distinction ne se fait donc pas sur le cardinal. Elle est topologique[5] ; mais il faudrait que la topologie soit toujours explicite. Notre point de vue est qu'il faut une connaissance intentionnelle (ou dynamique) pour pouvoir se prononcer. Regardons nos différentes classes de modèles.

Pour les modèles algébriques, par réécriture et par propagation, la distinction se fait par les arguments et valeurs possibles ainsi que par les fonctions présentes. Pour les modèles dynamiques, il faut regarder ce qui définit le système : l'échelle des temps, les ensembles de configurations et la règle de mise à jour. Si l'on approche le système par les orbites ou *diagrammes espace-temps* engendrés, il faut alors considérer les média ou *supports* pour le temps et l'espace, les valeurs locales des configurations et le critère de validité / correction des diagrammes (*e.g.* ce critère correspond-il à une formule logique du premier ou deuxième ordre ?).

Par exemple, pour une machine de Turing, le temps (itérations sur $\mathbb{N}$), l'espace (le ruban, $\mathbb{Z}$), les valeurs locales (états et symboles) et la fonction de transition (table finie) sont discrets. Inversement, un modèle basé sur des équations différentielles, sur des fonctions de $\mathbb{R}$ dans $\mathbb{R}$ serait, sans contestation, continu.

De l'observation de différents modèles de calcul dynamiques, on peut en extraire deux classes : ceux à *dynamique locale* et ceux à *dynamique globale*. Nous appelons locale toute dynamique dont la mise à jour de tout site du support ne dépend que des sites voisins (selon une topologie, éventuellement implicite, du support). Elle est globale si un site peut être influencé par un site arbitrairement éloigné. Cette classification n'est pas stricte : d'un côté, elle peut dépendre de la nature des configurations et, de l'autre, il existe des cas intermédiaires.

Notre classification des modèles est subjective et basée sur une compréhension elle aussi subjective de la nature de ces modèles.

Tous les modèles, sauf indication, sont capables de simuler toutes les machines de Turing et sont donc universels pour le calcul. Par ailleurs, certains modèles ayant des caractéristiques continues ont des capacités de calcul super-Turing. Une synthèse de cette classification, où sont inclus les automates cellulaires et les machines à signaux, est donnée dans les figures 2.1 (classification) et 2.2 (aspects continus / discrets).

2.2.1 Modèles algébriques

Nous ne revenons pas sur les fonctions récursives qui sont discrètes.

Fonctions analytiques récursives [Moo96] Cris Moore a défini un pendant analytique de la théorie de la récursion classique. Les fonctions de base sont de $\mathbb{R}^n$ dans $\mathbb{R}$ et les opérateurs sont adaptés (*e.g.* la récursion primitive devient l'intégration). Ce modèle est

[5] Un ensemble est continu en x si x appartient à la fermeture du complémentaire de $\{x\}$, discret sinon. Un ensemble est continu (resp. discret) s'il est continu (resp. discret) en chacun de ses éléments. Un ensemble ni continu ni discret est hybride (faute de meilleur terme, nous prenons celui des systèmes hybrides). Cela correspond à nos exemples et aux topologies discrètes (tous les ensembles sont ouverts) et s'applique aux sous-ensembles grâce aux topologies traces.

excessivement puissant : en plus des polynômes et des fonctions trigonométriques, on trouve aussi la fonction caractéristique de $\mathbb{Q}$. Des amendements sur les fonctions de base et les opérateurs ont été proposés pour remédier à cet excès et des liens ont été établis avec l'analyse récursive [CM01, Hai03]. Les valeurs manipulées sont continues.

Nous pouvons aussi placer ici les Machines de Matiyasevich, dont la seule opération est de dire si une équation diophantienne a ou non une solution ; ces équations pouvant coder les ensembles non récursifs.

2.2.2 Modèles dynamiques par réécriture

Nous ne revenons pas sur le λ-calcul qui est discret. Tous les modèles présentés ici sont discrets.

Algorithmes de Markov [MY78, Section 1.8] C'est un système de réécriture du type grammaires contextuelles. Les règles ont un ordre de priorité et certaines sont marquées comme terminales. Le modèle est déterministe.

Machines de Kolmogorov [US93] Ce modèle se veut le plus général possible pour englober tout ce qui a une dynamique locale. À l'inverse des systèmes de réécriture classiques, au lieu de considérer des structures de listes (*i.e.* les mots), elles travaillent sur des graphes (et non plus seulement des chaînes).

Systèmes de Post [Pos46] C'est un système de réécriture du type grammaires contextuelles. Toutes les règles sont à deux sens (*i.e.* de la forme $m \leftrightarrow m'$). Le système est totalement non déterministe (et le *problème du mot* est indécidable).

2.2.3 Modèles dynamiques par propagation

La puissance de calcul est très compliquée à définir pour les modèles suivants. En effet, chaque instance ne définit qu'une fonction élémentaire de n variables dans p, comme un ensemble d'opérations de base avec un ordre (pas forcément complet) d'évaluation. La question de la puissance se pose si l'on considère une famille de telles fonctions. Mais il faut encore définir des familles ; sans contraintes, tout est calculable. Les contraintes sont souvent définies à partir d'un autre modèle de calcul[6], ce qui ne rend pas la classification simple. Ceci est normal puisque ces modèles ont pour origine le calcul sur des ensembles finis et non sur $\mathbb{N}$ et, pour certains, la conception de circuits imprimés.

Tous ces modèles sont à temps discret, le nombre d'itérations étant déterminé à l'avance. Le support est discret (nombre fini de variables).

[6]Par exemple, les circuits engendrés en temps polynomial en fonction de 1^n caractérisent P.

Circuits booléens [Vol99, vL90, Chap. 14] Un circuit est un graphe orienté fini sans circuit (au sens de la théorie des graphes) dont les sommets internes sont étiquetés par des portes logiques. Les racines sont des entrées binaires à fournir et les feuilles, les sorties.

Nous ne présentons pas les versions probabilistes et quantiques [For03], qui n'apportent rien à notre classification.

Réseaux de neurones non-ré-entrants Ce sont aussi des graphes orientés sans circuit, mais les portes logiques sont remplacées par des fonctions seuils (ou des sigmoïdes, les valeurs étant alors réelles) en fonction de la valeur d'une forme linéaire sur les entrées.

Straight line programming La machine est une suite d'instructions de la forme numéro de ligne := opération({numéro de ligne}*). Les numéros opérandes devant être inférieurs au numéro courant.

Selon les cas, les valeurs et les opérations peuvent être discrètes ou continues.

2.2.4 Modèles à dynamique locale

Nous ne revenons pas sur les machine de Turing, foncièrement discrètes. Les modèles de cette classe sont majoritairement discrets.

Analyse récursive [Wei00] C'est le domaine des machines de Turing de type 2 : l'entrée et la sortie sont sur deux rubans distincts. L'entrée est infinie et on ne peut ni y écrire ni revenir en arrière. La sortie est produite caractère à caractère, on ne peut ni y lire ni y récrire ; elle est elle aussi de taille infinie. Les mots infinis inscrits sur ces rubans représentent des réels. Pour obtenir un résultat exact, il faut un nombre infini d'itérations de la machine ; pour une « approximation » un temps fini suffit.

Le temps est discret ; les configurations sont à support discret ($\mathbb{N}$) à valeurs discrètes (alphabet fini), manipulées discrètement (*i.e.* lettre à lettre). Malgré cela, nous serions tenté de dire qu'il est continu car les valeurs que l'on veut manipuler sont réelles et les fonctions correspondantes de $\mathbb{R}$ dans $\mathbb{R}$ sont continues pour la topologie usuelle de $\mathbb{R}$.

Automates temporisés [AD94] Dans le domaine de la vérification de systèmes temps-réel, on ne peut se contenter d'une simple relation de successeur ou de dates entières. Pour cela des automates finis temporisés manipulant des mots temporisés (*time-event*, une date est ajoutée à chaque entrée de l'automate) ou des *signaux* (fonctions constantes par morceaux) ont été introduits et intensivement étudiés [Rab98, Rab03, Tra01]. L'échelle des temps est continue, mais les changements d'états sont discrets. Les cas d'accumulation de ces dates (paradoxe de Zénon) sont généralement exclus, bien qu'une approche pour les traiter ait été proposée [BP00].

Ce modèle n'est pas Turing-universel, la mémoire des automates étant finie. Ils correspondent plus à la caractérisation d'un comportement qu'à un calcul.

On peut se demander, ici, si la donnée est continue (mot temporisé $(\Sigma \times \mathbb{R}_+)^*$ et les itérations discrètes ou, si les données sont discrètes mais le temps continu (selon la manière dont on considère la temporisation des mots).

Machine de Turing quantiques [Gru99, Hir01] Ce sont des machine de Turing, mais avec des opérateurs atomiques différents. Les valeurs des amplitudes sont complexes, les opérateurs sont des opérateurs linéaires unitaires. Les valeurs et la mise à jour sont donc continues.

Le modèle n'est pas tout à fait local car des cases arbitrairement éloignées et en nombre quelconque peuvent être corrélées[7]. Il se crée ainsi des liens de voisinage.

2.2.5 Modèles à dynamique globale

Automates à compteurs [Min67] Nous ne présentons pas ici ce modèle, car cela est fait dans la Sect. 4.1. Nous le considérons comme global car il peut, à tout moment, modifier n'importe quel compteur et donc agir à distance[8].

Continuous-space optical model [NW01] Dans ce modèle, les données sont des images carrées à deux couleurs (*i.e.* des fonctions caractéristiques de sous-ensembles de $[0,1]^2$), les opérateurs disponibles sont des redimensionnements et des filtres de décompositions en séries de Fourrier.

Nous l'avons inclus dans notre liste pour montrer la variété des modèles proposés. La dynamique est par itération, le support est discret ($\mathbb{Z}^2$) mais les valeurs sont continues. Pour chaque opération seulement deux parties finies de $\mathbb{Z}^2$ sont concernées ; mais comme la taille des redimensionnements n'est pas bornée et qu'il n'y a pas non plus de distance entre la donnée et le résultat, il faut le considérer comme global.

Équations aux dérivées partielles [Orp94, ŠO01] Ce sont des modèles où les configurations ont des valeurs continues, le temps est lui aussi continu et la dynamique est une équation différentielle. La trajectoire est le calcul ; le résultat la valeur limite. C'est un modèle continu bien que le nombre de variables soit fini.

Modèle de Blum, Shub et Smale [BSS89, BCSS98] Ce modèle peut être vu comme un programme comportant des évaluations de polynômes et des branchements. Il peut calculer une fonction de $\mathbb{A}^n$ dans $\mathbb{A}$ polynomiale par morceaux ($\mathbb{A}$ étant un anneau). Ces morceaux sont délimités par des polynômes et peuvent être en nombre infini. Ce modèle a une capacité de calcul super-Turing dès que l'on accepte des valeurs réelles quelconques car le modèle peut alors s'en servir comme oracles.

Le temps est discret ; les configurations ont un support discret (nombre de variables) mais des valeurs dans un ensemble discret ou continu (selon l'anneau, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$ ou $\mathbb{C}$).

Ordinateurs analogiques [Sha41, Sha42, PE74] Avant le règne du tout digital, devant la faible capacité des ordinateurs disponibles et les types de problèmes à résoudre, on a conçu des ordinateurs analogiques, plus électriques qu'électroniques. Une théorie a été développée

[7] Typiquement, la *téléportation quantique* est instantanée quelle que soit la distance entre les éléments de la paire ERP (on peut se poser la question de savoir s'il n'y a pas altération de l'espace).

[8] Nous ne parlons pas du modèle RAM [AHU74, CR73] qui se classent comme les automates à compteurs.

sur ce modèle qui a été utilisé de la seconde guerre mondiale jusqu'à la fin des années soixante. Les seuls problèmes sont la précision et le bruit.

Système hybrides [Bra95] Ce sont des modèles à temps et configurations continus mais en partie discrets : l'évolution se faisant parfois par paliers discrets (*e.g.* dirigée par un automate fini, configurations dans $\mathbb{R}^n \times \mathbb{N}$).

Par exemple, le modèle à dérivée constante par morceaux (ces morceaux étant polygonaux et en nombre fini) a été étudié et des liens avec la hiérarchie analytique ont été établis [AM98, Bou99a, Bou99b]. Le système est à temps continu, mais les seuls instants importants sont ceux où l'on change de morceau. Ceux-ci sont discrets, mais ils peuvent s'accumuler (problème d'Achille et de la tortue) ce qu'utilise le modèle.

(Analog) recurrent neural networks (RNN et ARNN) [SS95] Ce sont des réseaux de neurones où le graphe d'interconnexion peut avoir des cycles. Les conditions de fin du calcul sont variées.

L'aspect local ou global d'un réseau de neurones n'est pas simple : si l'on considère les liaisons synaptiques comme réseau d'interconnexions, tout est local ; mais le modèle permet aussi que la valeur d'un neurone dépende de toutes les autres, ce qui est global.

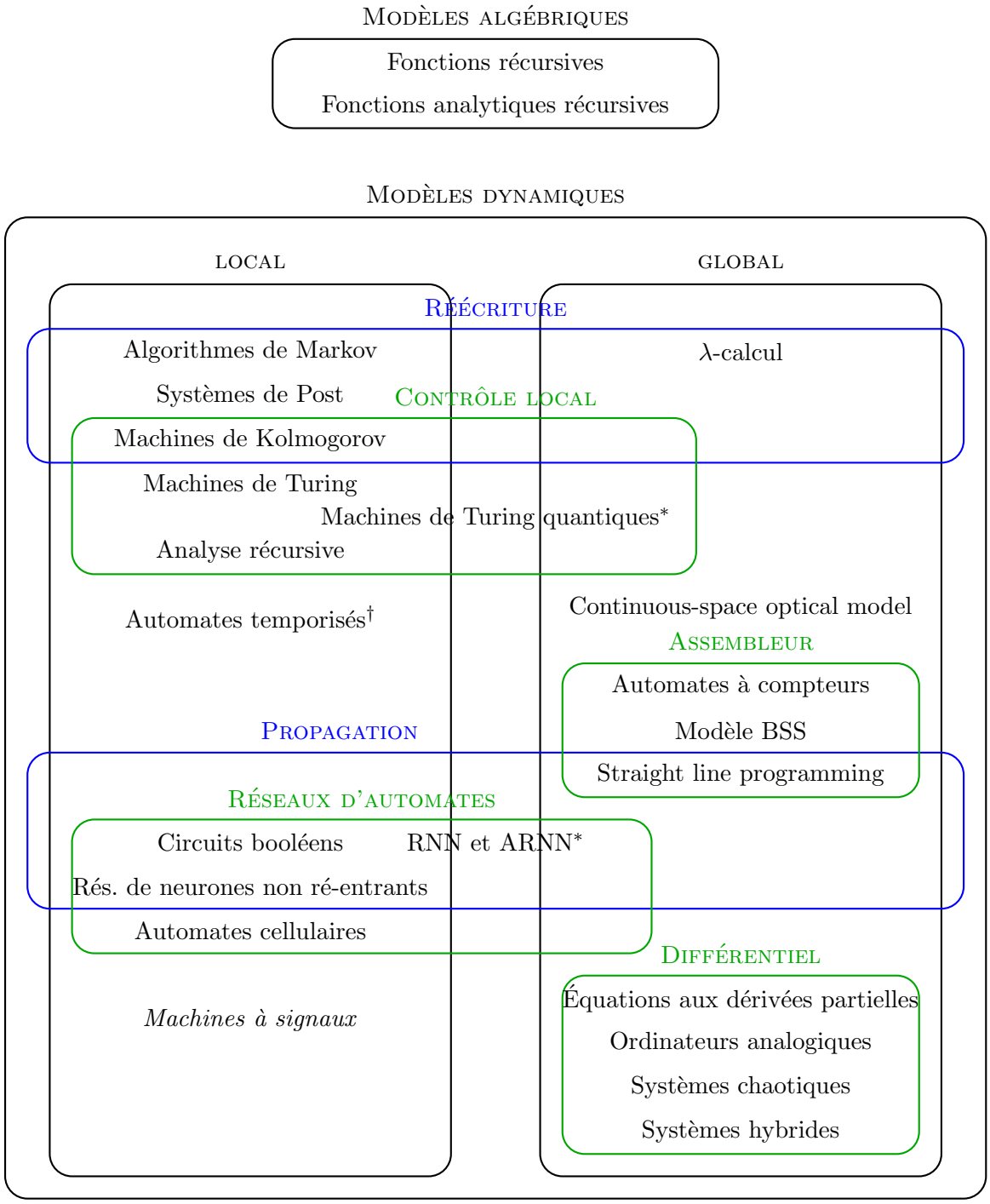
Citons aussi la bio-informatique (*e.g.* ADN [PRS98] que l'on pourrait considérer comme continu en tant que grand fini), et les systèmes chaotiques [Moo91, Sie96] (foncièrement continus).

Cette classification peut être poursuivie, par exemple on peut définir les classes de modèles suivantes :

- *Assembleur*, *i.e.* suite d'instructions sur des variables : automates à compteurs, modèle BSS et straight line programming ;
- *Contrôle local* : ils déterminent localement, à l'endroit de la précédente modification, l'action à faire : machines de Kolmogorov, machines de Turing (dont quantiques) et analyse récursive ;
- *Différentiels* : ils utilisent des équations différentielles et ont donc besoin de temps continu : équations aux dérivées partielles, ordinateurs analogiques, systèmes chaotiques et hybrides ;
- *Réseaux d'automates* : circuits, tous les réseaux de neurones et les automates cellulaires.

Les machines à signaux sont donc des modèles à dynamique locale. C'est à notre connaissance, le seul modèle à temps et support continus mais à valeurs et mises à jour discrètes. C'est aussi le seul de nature géométrique.

Terminons ce chapitre en mentionnant que le continu pourrait ne pas exister et nous ferions alors fausse route. En effet la thèse d'une réalité totalement discrète (et même finie) existe [Ela01, Zei01].



* classification ambiguë, voir la description du modèle.
† ce n'est pas un modèle de calcul.

Fig. 2.1 – Classification des modèles.

Modèle	Temps	Support	Valeurs	Mise à jour
λ-calcul	discret	discret	discret	discret
Algorithmes de Markov	discret	discret	discret	discret
Machines de Kolmogorov	discret	discret	discret	discret
Systèmes de Post	discret	discret	discret	discret
Automates cellulaires	discret	discret	discret	discret
Analyse récursive	discret	discret	discret[1]	discret
Automates temporisés	cont./disc.[2]	continu	discret	discret
Machines de Turing	discret	discret	discret	discret
Machines à signaux	*continu*	*continu*	*discret*	*discret*
Circuits booléens	discret	discret	discret	discret
Réseaux de neurones non ré-entrants	discret	discret	discret	discret
ARNN	discret	discret	continu	continu
RNN	discret	discret	discret	discret
Machines de Turing quantiques	discret	discret	continu	continu
Automates à compteurs	discret	discret	discret	discret
Modèle BSS	discret	discret	cont./disc.[3]	discret
Straight line programming	discret	discret	cont./disc.[3]	discret
Continuous optical model	discret	discret	continu	continu
Équations aux dérivées partielles	continu	continu	continu	continu
Ordinateurs analogiques	continu	discret	continu	continu
Systèmes chaotiques	continu	continu	continu	continu
Systèmes hybrides	cont./disc.[4]	discret	cont./disc.[4]	cont./disc.[4]

1 : même si globalement c'est un réel qui est codé.
2 : évènements discrets mais sur un support continu.
3 : selon les éléments / l'anneau considéré.
4 : selon le modèle, continu ou discret peut disparaître d'une des cases.

FIG. 2.2 – Composantes des modèles dynamiques.

Première partie

Comportement totalement discret

Chapitre 3

Définitions

Dans ce chapitre, après un exemple informel, nous définissons ce qui compose le monde des machines à signaux : espace-temps, méta-signaux, signaux, règles, collisions et diagrammes espace-temps. Nous n'abordons la possibilité d'avoir des accumulations de collisions qu'à partir du Chap. 9, d'ici là nous considérons qu'il n'y a jamais de phénomènes « non discrets ».

3.1 Exemple minimaliste

3.1.1 Énoncé

« *Considérons la droite réelle. Au temps* $t = 0$*, il y a un point (ou signal) bleu en position* $x = 2$ *et un signal rouge en position* $x = 8$*. Les signaux bleus se déplacent toujours à la vitesse* $(\frac{dx}{dt})$ 2 *et les signaux rouges à la vitesse* -1*.*

Quand un signal bleu et un signal rouge se rencontrent, ils disparaissent et sont remplacés par un signal violet. Les signaux violets se déplacent à la vitesse $\frac{3}{2}$*.*

Décrire l'évolution du système. »

3.1.2 Dynamique

Clairement, les signaux bleu et rouge vont se rencontrer à $x = 6$ et $t = 2$. Après il ne reste qu'un signal violet.

La Fig. 3.1 montre les configurations : initiale ($t = 0$), à mi-temps de la première collision ($t = 1$), durant la collision ($t = 2$) et après la collision ($t = 3$).

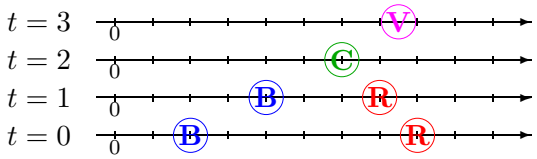


FIG. 3.1 – Configurations à quatre instants donnés.

3.1.3 Représentation

Les positions et les dates pouvant prendre n'importe quelle valeur dans $\mathbb{R}$, le phénomène se représente plus clairement en faisant une figure à deux dimensions : une horizontale pour l'espace et, une verticale pour le temps. Les signaux se déplaçant à vitesse constante, les différentes positions des signaux forment des lignes sur ce diagramme espace-temps comme on le voit sur la Fig. 3.2.

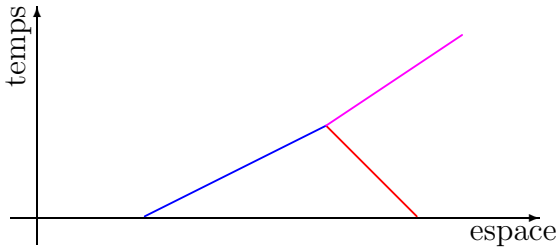


FIG. 3.2 – Diagramme espace-temps correspondant à l'exemple.

Ce sont ces types de phénomènes et figures qui nous intéressent dans ce mémoire. Définissons-les formellement.

3.2 Espace-temps

Les signaux sont des « informations ponctuelles » se déplaçant dans un *espace* au cours du *temps*. Avant de définir plus avant ce que sont des signaux, définissons ce couple espace-temps. L'*espace* est un espace affine de dimension 1 : $\mathbb{R}$. Le temps est lui aussi continu ; il est réel : $\mathbb{R}^+$ (il a une origine, 0). Une *position* est toujours comprise dans l'espace-temps, *i.e.* une position dans l'espace à un instant donné.

Définition 1 L'espace-temps est $\mathbb{R} \times \mathbb{R}_{0\leq t}$ (les temps commencent à 0), noté $\mathbb{R}^2_{0\leq t}$, les éléments de cet ensemble sont appelés des *positions*. Une position est notée $p = (x, t)$. Elle est *antérieure / postérieure (strictement)* à une position (x', t') si, et seulement si, $t \leq t'$ / $t' \leq t$ ($t < t'$ / $t' < t$).

Ce mémoire ne traite que du cas unidimensionnel. Néanmoins les définitions et certains résultats s'étendent naturellement aux dimensions supérieures.

3.3 Signal

Chaque signal est une instance d'un méta-signal donné. Celui-ci détermine l'information transportée et le mouvement. Il détermine aussi ce qui se passe quand des signaux se rencontrent.

Définition 2 Un *méta-signal* se définit par une paire information transportée et vitesse. Il est noté $\mu = (\iota, \nu)$.

L'exemple définit trois méta-signaux : $(\mathsf{bleu}, 2)$, $(\mathsf{rouge}, -1)$ et $(\mathsf{violet}, \frac{3}{2})$.

L'information est une valeur, lettre, symbole dans un ensemble fini quelconque. Il n'y a qu'un nombre fini de méta-signaux pour une machine donnée. Quand cela ne prête pas à confusion, nous utilisons l'information véhiculée pour désigner le méta-signal.

La vitesse est une valeur réelle, elle représente un déplacement par unité de temps. On peut aussi la considérer comme une vitesse dans l'espace-temps en ajoutant une coordonnée valant 1 et la voir comme un élément d'un plan projectif. Les signaux ne peuvent se déplacer sans que le temps ne s'écoule. Le temps ne peut qu'augmenter.

Deux méta-signaux sont dits *parallèles* s'ils ont la même vitesse (tout méta-signal est parallèle à lui-même).

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

Définition 3 Un *signal* se définit par son méta-signal et sa position. Il est noté $\sigma = (\mu, p)$.

Dans l'exemple, il y a deux signaux au départ et, à la fin, il n'y en a qu'un.

Pour simplifier, nous parlons de la vitesse d'un signal pour la vitesse de son méta-signal. Deux signaux sont dits *parallèles* s'ils ont la même vitesse (*i.e.* si leurs méta-signaux sont parallèles).

Les signaux se déplacent de manière rectiligne uniforme suivant leurs vitesses. Tant qu'un signal $(\mu, p) = ((\iota, \nu), (x, t))$ n'a pas rencontré d'autres signaux, il correspond toujours au même méta-signal et sa position à l'instant t' (postérieur à t) est donnée par :

$$p_{t'} = (x + \nu(t' - t), t') \ .$$

3.4 Collision

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

Les signaux étant des instances de méta-signaux à des positions données, le résultat d'une rencontre entre signaux ne dépend que de leurs méta-signaux. Une règle de collision se définit donc par un ensemble d'au moins deux méta-signaux se rencontrant et un ensemble de méta-signaux les remplaçant. Les collisions ne sont que des applications des règles.

Définition 4 Une *règle (de collision)* se définit par un ensemble d'au moins deux méta-signaux entrants et par l'ensemble des méta-signaux sortants, aucun de ces ensembles ne contenant deux signaux distincts parallèles. Elle est notée $\rho = \{\mu_i^-\}_i \to \{\mu_j^+\}_j$. La règle est dite *blanche* si $\{\mu_i^-\}_i = \{\mu_j^+\}_j$, comme si les signaux se croisaient sans interagir.

Pour définir graphiquement des règles, nous les traçons dans des cercles, les signaux entrants en bas et les sortants en haut, comme sur la Fig. 3.3 qui est l'unique règle de l'exemple : $\{\,\mathsf{bleu}, \mathsf{rouge}\,\} \to \{\,\mathsf{violet}\,\}$.

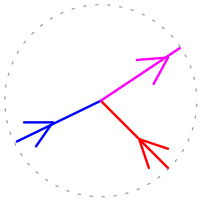
$\{\,\mathsf{bleu},\,\mathsf{rouge}\,\} \to \{\,\mathsf{violet}\,\}$ 

FIG. 3.3 – La règle de l'exemple.

Il ne peut y avoir qu'une règle définie pour le même ensemble de méta-signaux, le système est déterministe[1].

Il ne peut y avoir deux signaux parallèles dans les signaux entrants, car pour être présents à la collision, ils devraient être au même endroit précédemment. De même pour les signaux sortants, si deux signaux sont de même vitesse, ils seraient instantanément en collision ce qui pourrait provoquer des collisions en chaîne sans que le temps avance[2].

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

Définition 5 Si des signaux correspondant aux méta-signaux $\{\mu_i^-\}_i$ se rencontrent à la position p, et s'il existe une règle $\{\mu_i^-\}_i \to \{\mu_j^+\}_j$, alors les signaux disparaissent et sont remplacés par des instances des méta-signaux $\{\mu_j^+\}_j$ à la même position p. Nous parlons de *mort* pour les signaux entrants, et de *naissance* pour les signaux sortants. Une collision est notée $\pi = (p, \rho)$.

Il n'y a qu'une collision dans l'exemple. Elle a lieu à la position $(6, 2)$.

Si aucune règle n'est définie pour une collision, la machine s'arrête[3].

Une collision peut augmenter ou réduire le nombre de signaux, tant qu'il n'y a qu'un nombre fini de collision, un nombre fini de signaux n'engendre qu'un nombre fini de signaux.

3.5 Machine à signaux

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

Définition 6 Une *machine à signaux* se définit par des ensembles finis de méta-signaux et de règles de collisions. Elle est notée $\mathcal{M} = (\{\mu_i\}_i, \{\rho_j\}_j)$.

La machine de l'exemple se définit par trois méta-signaux et une règle :

$$\left(\left\{ \ (\mathsf{bleu}, 2),\ (\mathsf{rouge}, -1),\ (\mathsf{violet}, \tfrac{3}{2}) \ \right\},\ \left\{ \ \{\,\mathsf{bleu}, \mathsf{rouge}\,\} \to \{\,\mathsf{violet}\,\} \ \right\} \right) .$$

[1] On pourrait bien entendu considérer un système non déterministe où il y aurait plusieurs règles pour un même ensemble de signaux entrants. Ce cas n'est pas considéré dans ce mémoire.

[2] Si de tels superpositions sont souhaitées, cela peut être fait en ajoutant de nouveaux méta-signaux correspondant à cette superposition. Les nouveaux méta-signaux et leurs règles étant alors définis et inclus dans la machine.

[3] Dans l'implantation réalisée, ce comportement est paramétrable et, on peut choisir que les signaux disparaissent ou se croisent sans interférer (la collision est blanche). D'autres comportements peuvent être définis et ajoutés, *c.f.* Ann. A.

Il est à noter que pour n méta-signaux, il y a $2^n - n - 1$ règles possibles (pas de collisions avec zéro ou un signal).

3.5.1 Configuration (de la machine à un instant donné)

Elle est définie par l'ensemble des signaux présents et des collisions se produisant à un instant donné, associés à leurs coordonnées spatiales.

Définition 7 La configuration d'une machine à un instant donné est l'ensemble des signaux et collisions présents à cet instant. Elle est notée c_t (ou c).

La Fig. 3.1 donne la configuration de la machine à quatre instants donnés.
L'entrée de la machine est définie par l'ensemble des signaux présents à $t = 0$.

Définition 8 L'*entrée*, ou la *configuration initiale* pour une machine à signaux est la liste des signaux et collisions présents à l'instant initial (*i.e.* $t = 0$), associés à leurs coordonnées spatiales. Elle est notée c_0.

La configuration initiale de l'exemple est $\{(\mathsf{bleu}, 2), (\mathsf{rouge}, 8)\}$.

3.5.2 Machine rationnelle

Définition 9 Une machine est *rationnelle* si toutes les vitesses sont rationnelles et si toutes les positions des signaux sur la configuration initiale doivent être rationnelles.

La machine de l'exemple est rationnelle.
Pour toute machine rationnelle, les opérations en jeu garantissent que l'on reste avec des valeurs rationnelles. Une récurrence immédiate montre qu'au bout d'un nombre fini de collisions, celles-ci ont toutes eu lieu à des positions (à coordonnées) rationnelles.
Dès qu'il faut coder les machines ou les simuler sur ordinateur, nous nous restreignons aux machines rationnelles. Cette restriction permet de simuler exactement car tous les rationnels sont représentables ainsi que leurs opérations.

3.5.3 Fin et résultat du calcul

Pour une machine à signaux, la fin du calcul correspond, au choix, à une collision non définie, à l'apparition d'un signal donné, à ne plus avoir de collisions, à un temps ou nombre de collisions prédéfini... La dernière configuration atteinte est dite *finale* et correspond au résultat du calcul.
Le calcul effectué correspond à toutes configurations parcourues. Il peut être infini.

Définition 10 Le *calcul* de la machine sur une entrée est l'historique de tous les signaux et collisions. Il est noté Γ.

3.5.4 Diagramme espace-temps

Un calcul peut être vu comme l'ensemble des signaux auxquels on ajoute leurs positions de naissance et de mort (le mouvement étant rectiligne toutes les positions intermédiaires se calculent simplement) ; les collisions sont alors définies implicitement.

Graphiquement, un calcul est très simple à représenter, les traces des signaux étant des segments de droite, comme sur la Fig. 3.2. Nous nommons cette représentation un *diagramme espace-temps*. Les axes du temps et de l'espace ne sont plus représentés. Par convention l'espace est la dimension horizontale et le temps va de bas en haut.

Sur ces diagrammes, on peut lire la configuration initiale, les pentes (donc les vitesses) des signaux et les définitions des règles de collisions entrant en jeu dans le calcul. Par exemple, la Fig. 3.3 suffit pour définir la règle et celle-ci n'est qu'un « extrait » du diagramme espace-temps de la Fig. 3.2.

Définition 11 Un *diagramme espace-temps* $\mathbb{D}$ est une fonction de $\mathbb{R}^2_{0\leq t}$ dans $\{\oslash\} \cup \{\mu_i\}_i \cup \{\rho_j\}_j$ correspondant à la trace d'un calcul. La valeur $\oslash$ signifie qu'il n'y a rien à la position donnée, *i.e.* ni signal ni collision.

Dans la seconde partie, nous considérons les diagrammes espace-temps de manière topologique pour traiter les points d'accumulation.

Cônes espace-temps

Ils permettent de percevoir la « localité du calcul ». Si l'on prend un point de l'espace-temps, l'information qui s'y trouve ne peut influencer qu'une partie restreinte de l'espace-temps ; il faut qu'un signal puisse atteindre l'une depuis l'autre.

Les méta-signaux étant en nombre fini, les vitesses sont bornées. Nous notons $\nu^{\mathcal{M}}_{min}$, $\nu^{\mathcal{M}}_{max}$ et $\nu^{\mathcal{M}}_{abs}$ les plus petite et plus grande vitesses présentes et, la plus grande vitesse absolue.

Nous appelons *cône d'influences émises* issus de p et notons $\mathcal{C}_+(p)$, l'ensemble des positions de l'espace-temps potentiellement influençables depuis cette position. Nous appelons *cône d'influences reçues* issus de p et notons $\mathcal{C}_-(p)$, l'ensemble des positions de l'espace-temps potentiellement influençant cette position. Le *cône espace-temps* issu d'une position est l'union de ces deux cônes.

Définition 12 Pour une position (x,t), les cônes espace-temps issus de (x,t), $\mathcal{C}(x,t)$, d'influences reçues, $\mathcal{C}_-(x,t)$, et, d'influences émises, $\mathcal{C}_+(x,t)$, sont définis par

$$\begin{aligned}
\mathcal{C}(x,t) &= \left\{ \ (x',t') \ \middle| \ 0 \leq \left(\nu^{\mathcal{M}}_{max}(t-t') - x + x'\right).\left(x - x' - \nu^{\mathcal{M}}_{min}(t-t')\right) \ \right\} \ , \\
\mathcal{C}_-(x,t) &= \mathcal{C}(x,t) \ \cap \ \mathbb{R} \times [0,t] \ , \\
\mathcal{C}_+(x,t) &= \mathcal{C}(x,t) \ \cap \ \mathbb{R} \times [t,\infty[\ .
\end{aligned}$$

La Fig. 3.4 représente les cônes correspondant à l'exemple.

Le calcul fait qu'il y a ou non effectivement influence ; par contre, il est sûr que ce qui se passe en dehors du cône est totalement indépendant, même si des causes ou des conséquences communes peuvent exister dans les intersections des cônes.

Les influences émises / reçues des uns et des autres se correspondent :

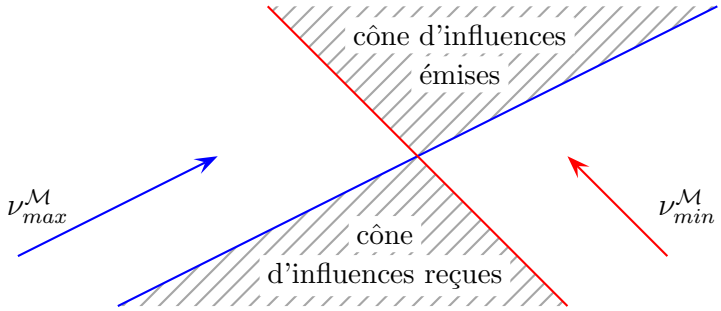


FIG. 3.4 – Cônes d'influences.

Propriété 13 *Pour tout couple de positions p et q*

$$\mathcal{C}_-(p) \subseteq \mathcal{C}_-(q) \quad \Leftrightarrow \quad p \in \mathcal{C}_-(q) \quad \Leftrightarrow \quad q \in \mathcal{C}_+(p) \quad \Leftrightarrow \quad \mathcal{C}_+(q) \subseteq \mathcal{C}_+(p) \ .$$

Chapitre 4

Universalité au sens du calcul

Nous commençons par quelques rappels informels sur le calcul au sens de Turing, avant de donner un exemple puis la construction de la simulation de n'importe quel automate à deux compteurs par une machine à signaux. Ce chapitre se termine par quelques résultats d'indécidabilité découlant de l'universalité au sens du calcul.

4.1 Rappels

Nous rappelons les notions de fonctions calculables, de simulation, de réduction, d'indécidabilité et d'universalité. Cette présentation est très informelle, *e.g.*, nous ne parlons pas de codage et nous ne nous soucions pas de savoir si le calcul porte sur des entiers ou des mots. Nous renvoyons pour des présentations plus extensives et formelles à des manuels classiques [DSW94, Deh93, Gru97, HU79, LP88, Sip97, Wol91] ou des états de l'art plus avancés comme [Gri99, vL90, chapitres 1 à 3]. Seules les notions utilisées sont formalisées.

4.1.1 Fonctions calculables

Avant même l'apparition de l'« ordinateur moderne », les hommes (en particulier les mathématiciens et les logiciens) se sont interrogés sur l'automatisation des calculs : ce que cela pouvait être et les propriétés et les limites de l'automatisation.

Les *fonctions calculables* capturent cette notion. Elles peuvent se définir à partir des fonctions récursives, du λ-calcul ou des machines de Turing. La grâce est que toutes les définitions proposées ont produit le même ensemble de fonctions ce qui justifie la thèse de Church-Turing (il n'y a que cet ensemble et il capture cette notion). Cette thèse permet d'utiliser indifféremment fonction, machine, programme, numéro ou index pour désigner une « méthode automatique effective de calculer ».

Le fait d'expliciter le passage de n'importe quelle fonction définie dans un formalisme (*e.g.* une classe de machines) dans un autre s'appelle une *simulation*. Il faut définir comment sont transcrites les machines et leurs entrées.

Définition 14 Une fonction partielle $g : B \to B$ (Turing-)*simule* une fonction partielle $f : A \to A$ si, et seulement si, il existe deux fonctions calculables $\mathsf{code}: A \to B$ et $\mathsf{décode}: B \to A$

telles que :
- décode∘code soit l'identité sur A,
- pour tout a de A, si $f(a)$ est définie alors $g(\mathsf{code}(a))$ l'est, et $f(a) = \mathsf{décode}(g(\mathsf{code}(a)))$, sinon (noté $f(a) = \perp$) $g(\mathsf{code}(a))$ n'est pas non plus défini.

Cette définition correspond au diagramme de la Fig. 4.1. L'identité est ajoutée car code (resp. décode) n'est pas forcément surjective (resp. injective). Les fonctions f et g étant partielles, un nouvel élément, $\perp$, est ajouté pour signifier « indéfini » et, la fonction décode est étendue par $\mathsf{décode}(\perp) = \perp$.

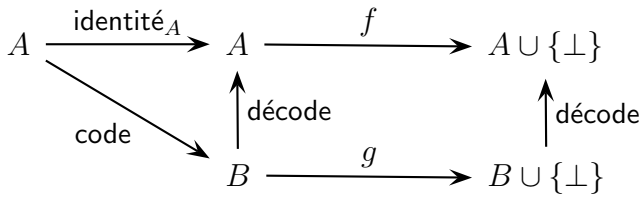


FIG. 4.1 – Diagramme de « g simule f ».

4.1.2 Problème et réduction

Nous appelons *problème* toute paire données à fournir et question sur ces données. Nous notons toujours les problèmes comme suit :

Nom du problème
Instance
Données à fournir
Question
Question posée

Un problème est dit *décidable* (ou *calculable* si la réponse n'est pas « oui » ou « non ») s'il existe une fonction calculable qui, prenant les données en entrée, retourne toujours la bonne réponse à la question. Il est dit *indécidable* (ou non calculable) dans le cas contraire. Dans ce mémoire, les seuls problèmes rencontrés ont pour réponse « oui » ou « non », ce sont des problèmes de décision.

Il peut être utile de transformer un problème en un autre pour tirer parti d'une connaissance sur l'un des problèmes. Les transformations entre machines sont des simulations, entre problèmes, on parle de réduction.

Définition 15 Un problème P est *Turing-réductible* à un problème Q si, et seulement si, il existe deux fonctions calculables, f des données de P dans les données de Q et, g des réponses de Q dans les réponses de P telles que : quelle que soit l'entrée x de P, si la réponse de P est y_P et la réponse Q à $f(x)$ est y_Q alors $y_P = g(y_Q)$.

Les problèmes peuvent être perçus comme des fonctions implicites de leurs instances dans leurs réponses possibles, la réduction correspond alors au diagramme de la Fig. 4.2.

Si Q est calculable, alors il en est de même pour P puisque f et g sont calculables. Par contre, si P n'est pas calculable, alors ce n'est pas non plus le cas pour Q.

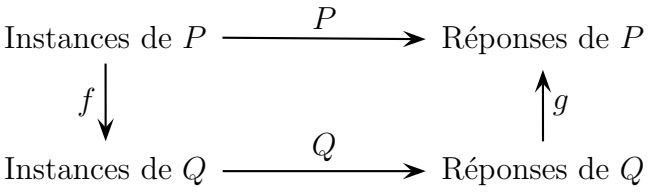


FIG. 4.2 – Diagramme de la Turing-réduction de P à Q.

Pour les problèmes de décision, la réponse étant « oui » ou « non » (g est souvent implicitement l'identité), on peut confondre un problème avec l'ensemble des instances pour lesquelles la réponse est « oui ». La réduction peut alors s'écrire :

$$\forall x \in \text{ Instances de } P, \qquad x \in P \iff f(x) \in Q \ .$$

4.1.3 Indécidabilité

Les fonctions ne sont pas toutes calculables. Elles ne sont pas toutes aussi « impossibles » à calculer ; une infinité de classes d'indécidabilité existe. Dans la plus simple de ces classes, se trouvent des prédictions sur des comportements des fonctions calculables. Ce sont des questions du type : « étant donné le code d'une machine et une entrée, le calcul s'arrêtera-t-il ? », « étant donné le code d'une machine et une entrée, le calcul passera-t-il par un état donné ? » ... Ces problèmes de décision se nomment *récursivement énumérables* ou Σ_1^0 dans la hiérarchie arithmétique, cette classe de problèmes contient tous les problèmes « plus simples », *e.g.* décidables.

Définition 16 Un ensemble d'entiers P est *récursivement énumérable*, ou dans Σ_1^0, s'il existe une machine M à deux entrées, x et n, telles que : M s'arrête toujours et, pour tout x, x appartient à P si, et seulement si, il existe un entier n tel que M réponde « oui » pour (x, n) :

$$\forall x, \qquad x \in P \iff \exists n \in \mathbb{N},\ M(x,n) \text{ répond « oui »} \ .$$

Pour la première question citée, la machine simule n itérations de la machine dont on veut savoir si elle s'arrête ; si l'arrêt est atteint, elle répond « oui », sinon elle répond « non ». La machine ainsi engendrée s'arrête donc toujours. Pour la seconde question, la machine simule de même n itérations et répond « oui » si l'on est passé par l'état donné. On peut aussi définir Σ_1^0 en termes logiques, quasi-similaires :

Définition 17 Un ensemble d'entiers P est dans Σ_1^0, s'il existe un prédicat récursif (*i.e.* décidable) total $\mathcal{P}$ à deux variables libres tel que x appartient à P si, et seulement si, il existe un entier n tel que $\mathcal{P}(x, n)$ soit vrai :

$$\forall x \in \mathbb{N}, \qquad x \in P \iff \exists n \in \mathbb{N},\ \mathcal{P}(x,n) \text{ vrai} \ .$$

Remarque 18 Les Σ_n^0 et les Π_n^0 forment la hiérarchie arithmétique. Pour $n = 0$, on obtient l'ensemble des ensembles récursifs : $\Sigma_0^0 = \Pi_0^0 = \mathcal{R}$. Pour $n = 1$, on obtient les ensembles des ensembles récursivement énumérables et co-récursivement énumérables : $\Sigma_1^0 = \mathcal{RE}$ et

$\Pi_1^0 = \text{co}-\mathcal{RE}$. Nous présentons plus en détail cette hiérarchie au Chap. 10 où nous atteignons un niveau plus élevé de cette hiérarchie.

Certains problèmes « capturent » la complexité d'une classe, on les nomme complets pour la classe. Ils se définissent de la manière suivante :

Définition 19 Soit $\mathcal{K}$ une classe de problèmes, un problème est *$\mathcal{K}$-(Turing-)difficile* si, et seulement si, tout problème de $\mathcal{K}$ peut y être (Turing-)réduit.

Soit $\mathcal{K}$ une classe de problèmes, un problème est *$\mathcal{K}$-(Turing-)complet* si, et seulement si, il appartient à $\mathcal{K}$ et est $\mathcal{K}$-(Turing-)difficile.

Un ensemble est donc Σ_1^0-complet s'il est dans Σ_1^0 et tout ensemble Σ_1^0 peut y être réduit. Les deux problèmes énoncés plus haut sont tous les deux Σ_1^0-complets. La réduction se faisant en transformant la machine qui répond « oui » ou « non », par une machine qui essaie toutes les valeurs de n et ne s'arrête que lorsqu'elle obtient la réponse « oui ».

4.1.4 Universalité au sens du calcul

Parmi les fonctions calculables, il en existe de particulières, dites (Turing-)*universelles*, qui peuvent simuler n'importe quelle fonction calculable (avec les mêmes « étapes ») pourvu que la fonction à simuler lui soit indiquée en entrée.

4.1.5 Automates à deux compteurs

Les automates à deux compteurs[1] étudiés par Minsky [Min67] sont certainement un des modèles représentant toutes les fonctions calculables les plus simples à décrire, à mettre en œuvre et à simuler. Ils sont théoriquement très intéressants mais, pratiquement inutilisables tant ils sont laborieux et tant les données qu'ils manipulent peuvent croître.

Un automates à deux compteurs est formé d'un code (suite d'instructions avec branchements conditionnels) facilement transposable en automate fini et de deux compteurs contenant des valeurs entières positives ou nulles.

Définition 20 Un *automate à deux compteurs* se compose de deux compteurs (`A` et `B`) à valeurs positives ou nulles et d'une suite d'instructions. Les seules instructions possibles sont :

- « `A++` » : augmenter la valeur de `A` de 1,
- « `B++` » : augmenter la valeur de `B` de 1,
- « `A--` » : si `A` ne vaut pas zéro, alors diminuer sa valeur de 1,
- « `B--` » : si `B` ne vaut pas zéro, alors diminuer sa valeur de 1,
- « `A != 0` étiq » : si `A` ne vaut pas zéro, alors aller à l'instruction étiquetée étiq,
- « `B != 0` étiq » : si `B` ne vaut pas zéro, alors aller à l'instruction étiquetée étiq.

Nous notons $\mathcal{A}$ un automate à deux compteurs, et a et b les valeurs des compteurs `A` et `B`. L'automate étant défini, il faut encore préciser son fonctionnement.

[1] Aussi connus comme « machines à deux registres ».

Définition 21 L'*exécution* d'un automate à deux compteurs est la suivante : l'entrée est la valeur des compteurs au départ ; on part de la première instruction. Sauf branchement, on passe à chaque fois à l'instruction suivante. Le calcul se termine quand on passe après la dernière instruction, le résultat du calcul étant la valeur des compteurs.

La configuration d'un automate à deux compteurs se définit par (l, a, b) où l est le numéro de la prochaine ligne d'instruction à exécuter, a et b sont respectivement les valeurs des compteurs `A` et `B`.

4.2 Simulation

Dans cette section, nous établissons une simulation effective des automates à deux compteurs par des machines à signaux.

4.2.1 Principe

La valeur des compteurs n'intervient pas, seule leur non-nullité est discriminante. Les compteurs sont donc codés en unaire. Ceci permet d'ajouter ou de retrancher un ainsi que de tester la nullité de manière très simple.

Trois sortes de méta-signaux sont utilisées pour la simulation : pour borner la configuration (noir), pour coder les compteurs (bleu et vert) et, pour les instructions. Les configurations contiennent, de gauche à droite : borne, a signaux bleus, instruction, b signaux verts, borne. L'instruction fait des aller-retours entre les deux compteurs. Les autres signaux sont parallèles de vitesses nulles.

4.2.2 Exemple

Partons du code d'automate à deux compteurs de la partie gauche de la Fig. 4.3. Décrivons brièvement ce que fait ce code : selon la parité de a, il le remplace par $\frac{a}{2}$ ou $\frac{3a+1}{2}$.

Sur le bas de chaque diagramme, on peut lire la condition initiale : le nombre de traits bleu est a (1 puis 3 puis 13), le trait oblique kaki correspond à la première instruction (ici « `beg : A++` »), le nombre de traits verts est b (0 dans les trois cas). Les traits codant les valeurs des compteurs sont verticaux, le zigzag médian correspond au déroulement des instructions. De temps à autre, il y a aussi des traits obliques bleus et verts pour augmenter les compteurs.

On observe sur les diagrammes espace-temps de la Fig. 4.3 les valeurs de `A`, ce sont les traits bleu sur la gauche. Sur le premier, on observe 1 puis 2 puis 1 puis 2... Sur le deuxième, on lit 3 puis 5 puis 8 puis 4 puis 2 puis 1... Sur le troisième, 13, 20, 10, 5, 8, 4... Nous laissons en « exercice » la question de savoir si quelle que soit la valeur de `A` au départ, la « frise » atteint toujours le même motif périodique.

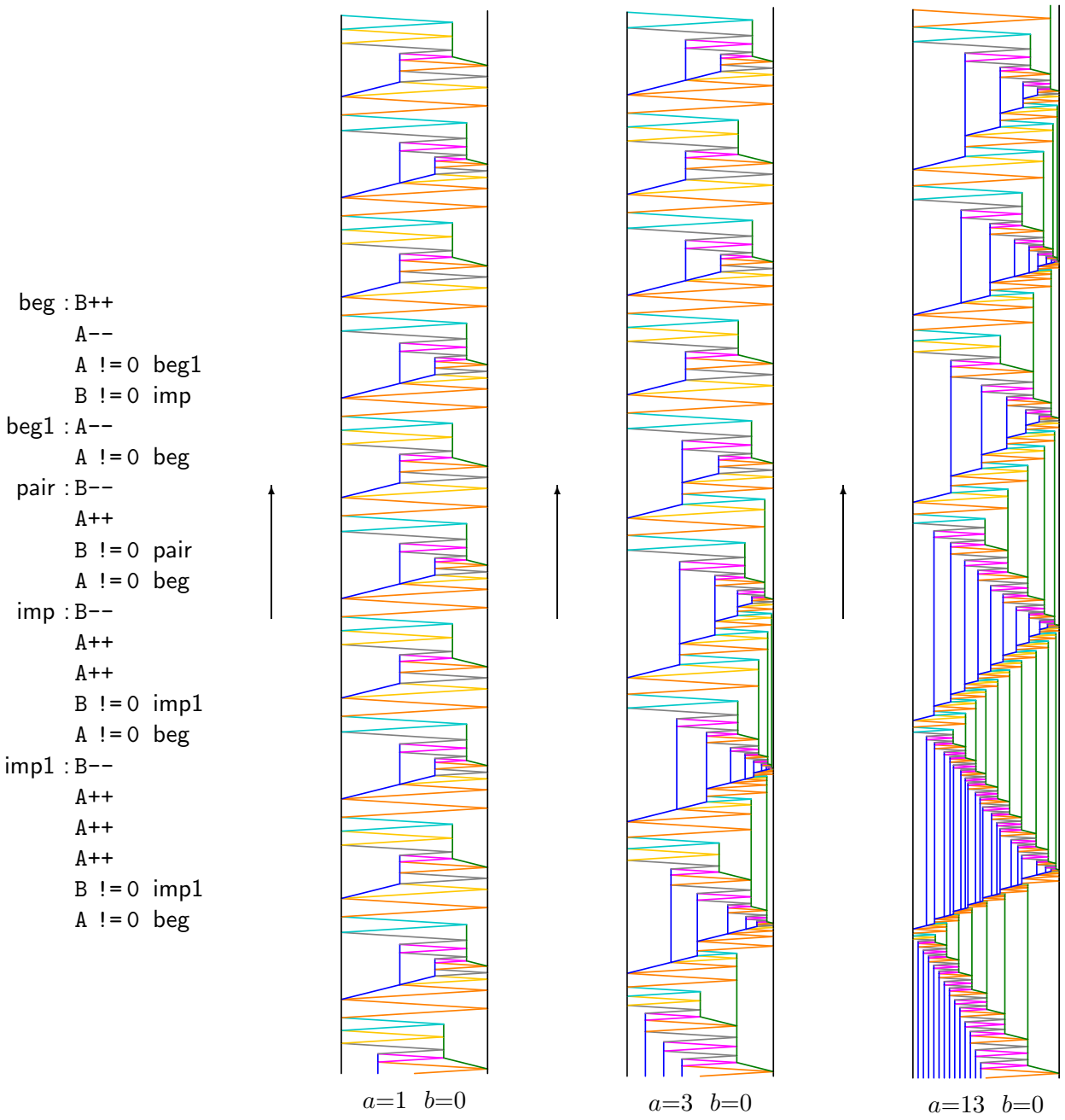


FIG. 4.3 – Automate à deux compteurs et trois simulations de calculs.

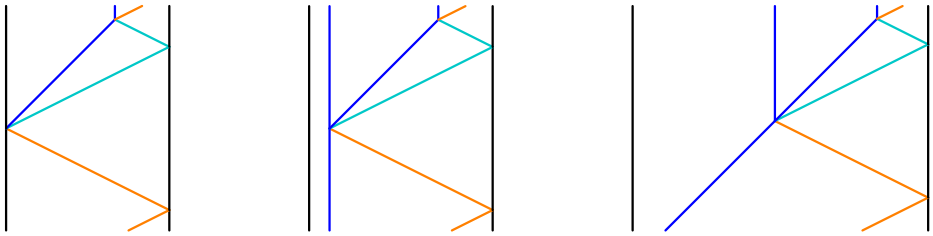
4.2.3 Construction

La réduction d'un automate à deux compteurs se fait simplement par la définition des méta-signaux, des collisions et des configurations de base. Nous commençons par les méta-signaux pour les bords, et les compteurs puis la configuration initiale. Ensuite nous décrivons les méta-signaux et les collisions pour les instructions. Enfin nous prouvons la correction de notre simulation.

Structure de base

Le méta-signal pour les bords est de vitesse nulle et d'information bord, il est représenté en noir. Les signaux bord marquent les bords gauche et droit de la partie codante de la configuration. Si un signal représentant une instruction l'atteint, cela signifie que `A` ou `B`, selon le côté, vaut zéro. Ceci permet, en plus d'empêcher les signaux codant les instructions de sortir, de faire tous les tests de branchement correspondant à la non nullité des compteurs.

Pour coder `A`, on utilise un autre méta-signal, de vitesse nulle et d'information a. Pour diminuer `A` de 1, une collision en fait disparaître un, par contre pour ajouter 1, il faut en créer un autre, mais ailleurs. Ceci est réalisé grâce à un autre signal de faible vitesse que remplace un signal a de vitesse nulle, au retour d'un signal d'instruction. Cette construction est donnée par la définition des méta-signaux et règles sur la Fig. 4.6. La Fig. 4.4 montre comment se passe l'ajout de 1 à `A`.



(a) $a = 0$ (b) $a \neq 0$ (c) précédé de « `A++` »

FIG. 4.4 – Augmenter `A` de un, les trois cas.

Les configurations contiennent, de gauche à droite : bord, a signaux a (le dernier pouvant être un aMv), instruction, b signaux b (le premier pouvant être un bMv), bord. La configuration initiale est décrite dans la Fig. 4.5.

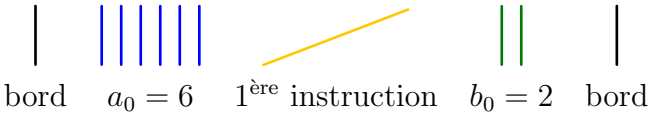


FIG. 4.5 – Configuration initiale pour la simulation d'un automate à deux compteurs.

Les instructions

Pour chaque ligne d'instruction, des méta-signaux et des règles en fonction du numéro de ligne (pour l'identifier) et de l'instruction sont engendrés. Tous ces méta-signaux sont de vitesse 2 ou -2. Les six instructions sont traitées les unes après les autres. À chaque fois, un tableau définit les méta-signaux et les règles, et une représentation de chaque règle est donnée.

« A++ » (et « B++ ») Ajouter 1 à A se fait en ajoutant un signal bleu à gauche du plus à droite des signaux bleus. L'instruction fait un aller-retour sans rien modifier à droite, mais au retour, elle crée un signal bleu se déplaçant qui sera mis en place au prochain aller-retour. Un méta-signal est donc ajouté : aMv (resp. bMv pour ajouter 1 à B) de vitesse 1 (resp. -1). Tous les signaux représentant des instructions ont des vitesses plus élevées, ici 2, pour les précéder.

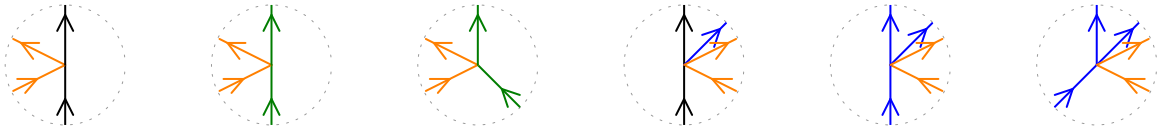
Quand un signal portant une instruction va à droite (aller), il y a trois collisions possibles : avec le bord bord (B est nul), avec un b fixe (B est strictement positif), avec un bMv venant à sa rencontre (B est strictement positif).

Quand un signal portant une instruction va à gauche (retour), il y a trois collisions possibles : avec le bord bord (A est nul), avec un a fixe (A est strictement positif), avec un aMv venant à sa rencontre (A est strictement positif).

L'aller-retour se passe simplement, sans rien changer si ce n'est fixer d'éventuels aMv et bMv mouvants. Par contre, en revenant un aMv est engendré ; s'il y en avait déjà un, il est fixé.

On fait de même pour B ++, par symétrie. Les règles et les signaux correspondant sont définis dans la Fig. 4.6, ils engendrent les diagrammes espace-temps de la Fig. 4.4.

Ligne		Méta-signal		Règles
Numéro	Instruction	Information	Vitesse	
n	A ++	aller_n	2	{ aller_n, bord } → { retour_n, bord }
				{ aller_n, b } → { retour_n, b }
				{ aller_n, bMv } → { retour_n, b }
		retour_n	-2	{ bord, retour_n } → { bord, aMv, aller_n+1 }
				{ a, retour_n } → { a, aMv, aller_n+1 }
				{ aMv, retour_n } → { a, aMv, aller_n+1 }



Ligne		Méta-signal		Règles
Numéro	Instruction	Information	Vitesse	
n	B ++	aller_n	2	{ aller_n, bord } → { retour_n, bord, bMv }
				{ aller_n, b } → { retour_n, b, bMv }
				{ aller_n, bMv } → { retour_n, b, bMv }
		retour_n	-2	{ bord, retour_n } → { bord, aller_n+1 }
				{ a, retour_n } → { a, aller_n+1 }
				{ aMv, retour_n } → { a, aller_n+1 }

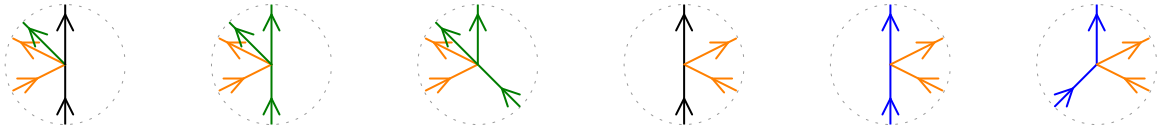
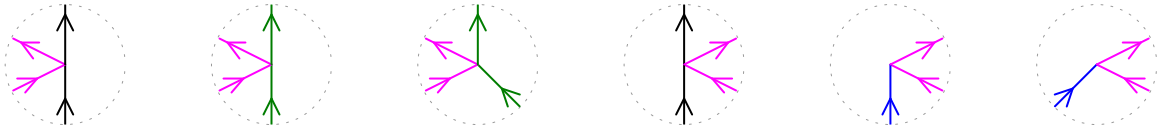


FIG. 4.6 – Méta-signaux et règles pour « A++ » et « B ++ ».

« A-- » (et « B-- ») Diminuer A de 1 se fait en enlevant le signal bleu le plus à gauche s'il y en a un. L'instruction fait un aller-retour sans rien modifier à droite, mais par contre, à gauche la mise à jour se fait en enlevant le signal rencontré s'il s'agit de a ou aMv. On procède de même, symétriquement, pour B--. Les méta-signaux et collisions correspondant sont définis dans la Fig. 4.7.

Ligne		Méta-signal		Règles
Numéro	Instruction	Information	Vitesse	
n	A --	aller_n	2	{ aller_n, bord } → { retour_n, bord }
				{ aller_n, b } → { retour_n, b }
				{ aller_n, bMv } → { retour_n, b }
		retour_n	-2	{ bord, retour_n } → { bord, aller_n+1 }
				{ a, retour_n } → { aller_n+1 }
				{ aMv, retour_n } → { aller_n+1 }



Ligne		Méta-signal		Règles
Numéro	Instruction	Information	Vitesse	
n	B --	aller_n	2	{ aller_n, bord } → { retour_n, bord }
				{ aller_n, b } → { retour_n }
				{ aller_n, bMv } → { retour_n }
		retour_n	-2	{ bord, retour_n } → { bord, aller_n+1 }
				{ a, retour_n } → { a, aller_n+1 }
				{ aMv, retour_n } → { a, aller_n+1 }

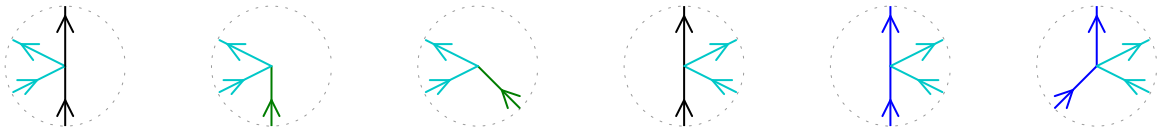


FIG. 4.7 – Méta-signaux et règles pour « A -- » et « B -- ».

« A !=0 m » L'instruction fait un aller-retour sans rien modifier à droite ni à gauche (si ce n'est fixer d'éventuels aMv ou bMv). Au retour, si l'on rencontre bord, c'est que A est nul et on passe à l'instruction n+1 (*i.e.* aller_n+1), sinon (rencontre de a ou aMv) A n'est pas nul et on passe à l'instruction m (*i.e.* aller_m). Les règles et les signaux correspondants sont définis dans la Fig. 4.8

« B !=0 m » Faire le test et le branchement par rapport à B, est à peine plus compliqué : c'est en faisant demi-tour que l'on sait si B est nul ou non, mais c'est à la fin du retour que l'on doit passer à l'instruction n+1 ou à l'instruction m. Deux méta-signaux, retVrai_n et retFaux_n, selon la nullité de B, sont définis pour le retour, le branchement s'effectue à la fin de celui-ci. Les règles et les signaux correspondants sont définis dans la Fig. 4.9.

Ligne		Méta-signal		Règles
Numéro	Instruction	Information	Vitesse	
n	A != 0 m	aller_n	2	{ aller_n, bord } → { retour_n, bord }
				{ aller_n, b } → { retour_n, b }
				{ aller_n, bMv } → { retour_n, b }
		retour_n	-2	{ bord, retour_n } → { bord, aller_n+1 }
				{ a, retour_n } → { a, aller_m }
				{ aMv, retour_n } → { a, aller_m }

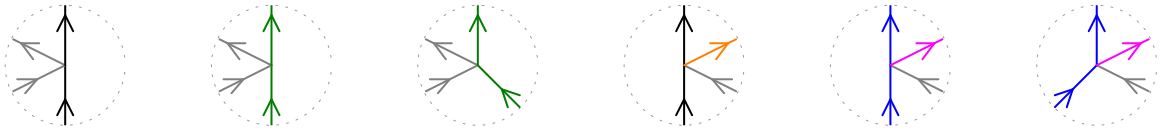


FIG. 4.8 – Méta-signaux et règles pour « A != 0 m ».

Ligne		Méta-signal		Règles
Numéro	Instruction	Information	Vitesse	
n	B != 0 m	aller_n	2	{ aller_n, bord } → { retFaux_n, bord }
				{ aller_n, b } → { retVrai_n, b }
				{ aller_n, bMv } → { retVrai_n, b }
		retFaux_n	-2	{ bord, retFaux_n } → { bord, aller_n+1 }
				{ a, retFaux_n } → { a, aller_n+1 }
				{ aMv, retFaux_n } → { a, aller_n+1 }
		retVrai_n	-2	{ bord, retVrai_n } → { bord, aller_m }
				{ a, retVrai_n } → { a, aller_m }
				{ aMv, retVrai_n } → { a, aller_m }

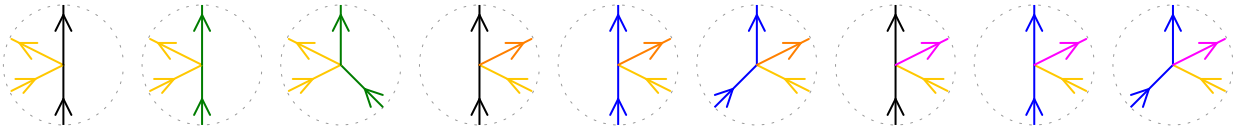


FIG. 4.9 – Méta-signaux et règles pour « B != 0 m ».

La fin du calcul d'un automate à deux compteurs correspond au passage après la dernière instruction. Pour la machine à signaux, cela correspond, au choix, à une collision non définie, à l'apparition d'un signal donné ou à ne plus avoir de collisions.

Pour cela, nous ajoutons deux méta-signaux, stopRi et stopLe, qui n'apparaissent jamais auparavant. Ils fixent les aMv et bMv et disparaissent. Ceci correspond à la fois à la fin du calcul indiqué par l'apparition d'un signal et à ne plus avoir de collisions. La Fig. 4.10 définit les règles concernant ces méta-signaux.

Cette construction tire parti du fait que l'espace est continu, il y a toujours de la place pour ajouter des a et des b au milieu de la configuration. Cette construction est impossible avec un automate cellulaire car la discrétisation de l'espace en cellules ne permet pas de mettre une quantité d'information non bornée entre deux cellules.

Avec notre construction, pour l lignes, il y a entre $8 + 2l$ et $8 + 3l$ méta-signaux et trois fois plus de règles. Si les valeurs initiales des compteurs sont a_0 et b_0, la configuration initiale

Méta-signal Information	Vitesse	Règles
stopRi	2	{ stopRi, bord } → { stopLe, bord }
		{ stopRi, b } → { stopLe, b }
		{ stopRi, bMv } → { stopLe, b }
stopLe	-2	{ bord, stopLe } → { bord }
		{ a, stopLe } → { a }
		{ aMv, stopLe } → { a }

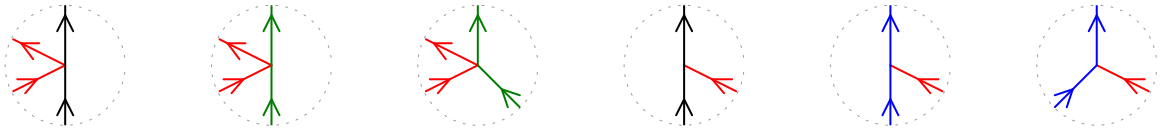


FIG. 4.10 – Règles pour stopRi et stopLe.

possède $3 + a_0 + b_0$ signaux. Durant le calcul, il y a au maximum $3 + a_M + b_M$ signaux (a_M et b_M étant les valeurs maximales prises par les compteurs).

Cette construction a été programmée ; elle est présentée dans l'Ann. A.

Correction de la simulation

Soit $\mathcal{A}$ un automate à deux compteurs et (l_0, a_0, b_0) (avec $l_0 = 1$) une configuration initiale pour $\mathcal{A}$. Soit $\mathcal{M}$ la machine à signaux construite à partir de $\mathcal{A}$ et la configuration initiale telle que définie par la Fig. 4.5.

Notons que la place exacte des signaux n'est pas importante pour notre simulation, ce qui est important est la liste des signaux de gauche à droite, ainsi que le considère le prédicat suivant.

Prédicat 22 ($\mathcal{P}_{red}(k)$) Si la configuration de $\mathcal{A}$ au bout de n itérations est (l_k, a_k, b_k) alors, entre la $2k^{\text{e}}$ et la $2k+1^{\text{e}}$ collisions $\mathcal{M}$ est composée, de gauche à droite des signaux :

- bord,
- (a_k fois a) ou ((a_k−1) fois a puis aMv),
- aller_l_k
- (b_k fois b) ou (bMv puis (b_k−1) fois b),
- bord.

Lemme 23 *$\mathcal{P}_{red}(n)$ est vrai pour tout n.*

Ce résultat se prouve très simplement par induction. Nous ne traitons que le plus compliqué des 6 cas, « `B != 0 m` », les autres ne soulevant pas de réels problèmes.

►► Preuve.

$\mathcal{P}_{red}(0)$ est vrai comme on le voit sur la Fig. 4.5.

Si $\mathcal{P}_{red}(k)$ est vérifié et la $l_k{}^{\text{e}}$ ligne est « `B != 0 m` ». Posons n=l_k pour alléger les notations. La prochaine collision ne peut avoir lieu qu'autour de aller_n puisque tous les autres signaux

sont parallèles ; les seules vitesses non nulles sont aller_n ou celles de ses deux plus proches voisins. Le prédicat $\mathcal{P}_{red}(k)$ étant vérifié, il ne peut y avoir qu'une des neuf possibilités de la Fig. 4.11.

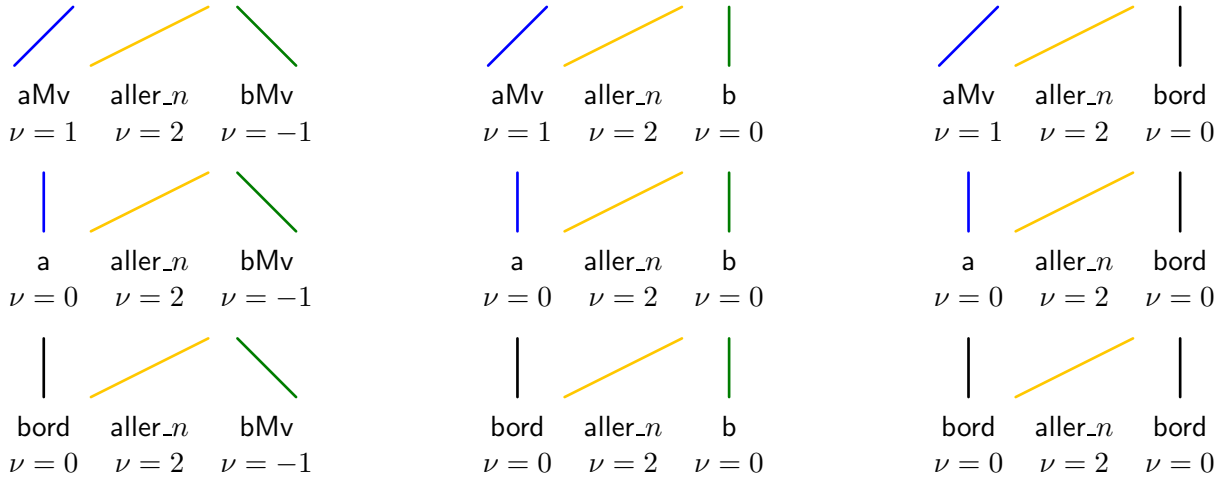


FIG. 4.11 – Neuf cas de signaux autour de aller_n.

Des vitesses, on déduit que la prochaine collision ne peut être qu'entre aller_n et le signal immédiatement à sa droite et qu'aucun autre signal ne rentre dans cette collision. Il y a trois possibilités pour le signal de droite : bord, b et bMv.

Si le signal à droite est un bord (première colonne de la Fig. 4.11, ce qui signifie que B est nul, le signal bord reste en place et c'est le signal retFaux_n qui repart (première règle de la Fig. 4.9).

La collision suivante ne peut être qu'avec le signal directement à gauche : bord, a ou aMv (4^e, 5^e et 6^e règles de la Fig. 4.9). Dans le premier cas, le bord reste en place, dans les deux autres, a est fixé ; le nombre de signaux a et aMv est inchangé. Dans les trois cas, il repart vers la droite ; le signal aller_m et les valeurs des compteurs n'ont pas changé.

En deux collisions, la machine est passée d'un codage de (n B != 0 $m, a_k, 0$) à ($m \ldots, a_k, 0$), ce qui correspond à la transition de $\mathcal{A}$; $\mathcal{P}_{red}(k+1)$ est vérifié.

Les autres possibilités concernent la rencontre avec b ou bMv ; les deux cas où B n'est pas nul. Dans ces deux cas (2^e et 3^e règles de la Fig. 4.9), il ressort retVrai_n qui repart à gauche, et b, fixe. Au retour, il y a rencontre avec bord, a ou aMv. Il se passe la même chose que précédemment, mais c'est aller_n+1 qui repart à droite.

En deux collisions, la machine est passée d'un codage de (n B != 0 m, a_k, b_k) avec $0 < b_k$ à (n+1 $\ldots, a_k, b_k$), ce qui correspond à la transition de $\mathcal{A}$; $\mathcal{P}_{red}(k+1)$ est vérifié.

Dans tous les cas, $\mathcal{P}_{red}(k+1)$ est vérifié.

On le démontre de même pour les autres instructions. Par induction, $\mathcal{P}_{red}(k)$ est donc vrai pour tout k.

◁◀◀

La Fig. 4.12 donne les deux collisions dans les neuf cas. Les trois cas de la colonne de droite correspondent au cas où B est nul.

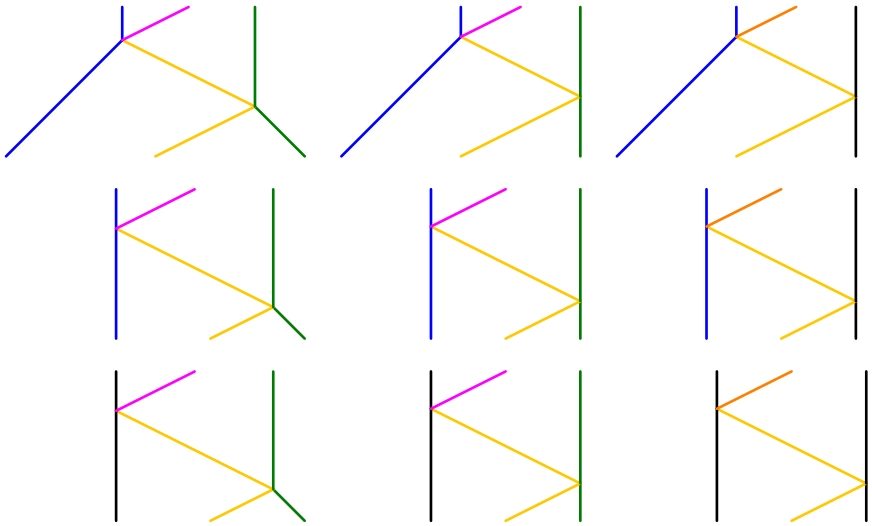


FIG. 4.12 – Neuf cas pour la simulation de « `B != 0 m` ».

Remarque 24 On aurait pu mettre les deux compteurs du même côté. Les signaux des instructions se contentent alors de croiser sans les modifier les signaux du compteur sur lequel ne porte pas l'instruction. On peut superposer de la sorte un nombre quelconque de compteurs.

4.3 Quelques résultats d'indécidabilité

Le résultat le plus classique de la décidabilité est qu'il est indécidable de savoir si un calcul se termine, quel que soit le modèle dans lequel ce calcul est effectué. Ceci est bien entendu vrai pour les automates à deux compteurs. Définissons ce problème.

Arrêt d'un automate à deux compteurs
Instance
$\mathcal{A}$: automate à deux compteurs,
(a_0, b_0) : valeurs initiales pour les deux compteurs
Question
$\mathcal{A}$ démarré avec (a_0, b_0) s'arrête-t-il ?

Ce problème est Σ_1^0-complet.

4.3.1 Nombre de collisions fini

Le problème est le suivant :

Nombre fini de collisions
Instance
$\mathcal{M}$: machine à signaux rationnelle,
c_0 : configuration initiale où il n'y a qu'un nombre fini de signaux (ceux-ci étant à des positions rationnelles).
Question
L'évolution de $\mathcal{M}$ sur c_0 n'occasionne-t-elle qu'un nombre fini de collisions ?

Théorème 25 *Le problème* NOMBRE FINI DE COLLISIONS *est* Σ_1^0*-complet.*

►► Preuve.
Il faut montrer qu'il est dans Σ_1^0 et, que l'on peut y réduire n'importe quel problème Σ_1^0 ou un problème Σ_1^0-complet.

Toutes les données étant rationnelles et en nombre fini, on peut faire un programme qui simule $\mathcal{M}$ jusqu'à ce qu'il n'y ait plus de collisions et s'arrête alors. Il se réduit donc au problème de l'arrêt et est donc dans Σ_1^0.

Le problème ARRÊT D'UN AUTOMATE À DEUX COMPTEURS se réduit simplement à NOMBRE FINI DE COLLISIONS, on utilise la construction de la Sous-sect. 4.2.3, la machine $\mathcal{M}$ et son entrée sont bien à vitesses rationnelles. Dans la configuration initiale, il n'y a qu'un nombre fini de signaux (ceux-ci sont à des positions rationnelles). L'exécution de $\mathcal{M}$ n'a un nombre fini de collisions si, et seulement si, l'automate à deux compteurs s'arrête. S'il ne s'arrête pas, les aller-retours non plus.

◁◄◄

4.3.2 Apparition d'un méta-signal

Le problème est le suivant :

Apparition d'un méta-signal
Instance
$\mathcal{M}$: machine à signaux rationnelle,
c_0 : configuration initiale où il n'y a qu'un nombre fini de signaux (ceux-ci étant à des positions rationnelles),
μ : méta-signal.
Question
Dans l'évolution de $\mathcal{M}$ sur c_0, le signal μ apparaîtra-t-il au bout d'un nombre fini de collisions ?

Le nombre fini de collisions est important car il peut y avoir des effets d'accumulation, mais ceci n'est considéré que dans la Partie II et, est explicitement exclu dans l'énoncé.

Théorème 26 *Le problème* APPARITION D'UN MÉTA-SIGNAL *est* Σ_1^0*-complet.*

►► **Preuve.**
Montrons que ce problème est dans Σ_1^0 et, que l'on peut y réduire un problème Σ_1^0-complet.

Toutes les données étant rationnelles et en nombre fini, on peut faire un programme qui simule $\mathcal{M}$ jusqu'à ce que le méta-signal μ apparaisse et s'arrête alors. Il se réduit donc au problème de l'arrêt et est donc dans Σ_1^0.

Le problème Arrêt d'un automate à deux compteurs se réduit simplement à Apparition d'un méta-signal, on utilise la construction de la Sous-sect. 4.2.3, la machine $\mathcal{M}$ est rationnelle et la configuration initiale ne contient qu'un nombre fini de signaux. Le méta-signal dont on guette l'apparition est stopRi. Ce méta-signal apparaît si, et seulement si, l'automate à deux compteurs s'arrête.

◁◄◄

4.3.3 Signal entrant en collision

Le problème est le suivant :

Signal entrant en collision
Instance
$\mathcal{M}$: machine à signaux rationnelle,
c_0 : configuration initiale où il n'y a qu'un nombre fini de signaux (ceux-ci étant à des positions rationnelles),
σ : signal de la configuration initiale.
Question
Au cours de l'évolution de $\mathcal{M}$ sur c_0, y aura-t-il une collision avec σ au bout d'un nombre fini de collisions ?

Théorème 27 *Le problème* Signal entrant en collision *est* Σ_1^0*-complet.*

►► **Preuve.**
Toutes les données étant rationnelles et en nombre fini, on peut faire un programme qui simule $\mathcal{M}$ jusqu'à ce qu'il y ait une collision avec σ et s'arrête alors. Il se réduit donc au problème de l'arrêt et est donc dans Σ_1^0.

Le problème Arrêt d'un automate à deux compteurs se réduit simplement à Signal entrant en collision, on utilise la construction de la 4.2.3, la machine $\mathcal{M}$ et son entrée sont bien à vitesses rationnelles et à positions rationnelles. Cette fois on ajoute un méta-signal de vitesse nulle et un signal σ correspondant à droite de la configuration initiale. Sans plus de modification, rien ne se passe et ce signal reste sur le côté. Ajoutons les méta-signaux nécessaires et modifions la règle pour stopLe de manière à ce qu'elle fasse apparaître un nouveau signal partant sur la droite et continuant son chemin quel que soit le signal rencontré jusqu'à atteindre σ (les résultats de ces collisions sont sans importance). Il y a collision si, et seulement si, la machine à deux compteurs s'arrête.

◁◄◄

Chapitre 5

Équivalence et premières constructions

Dans ce chapitre, nous montrons qu'il est possible de changer des paramètres d'une machine à signaux sans modifier les calculs effectués. Nous commençons par définir une équivalence, entre calculs, et une réduction (comme emboîtement entre calculs, à partir d'ordres partiels sur les collisions). Nous montrons ensuite comment on peut modifier de manière statique l'évolution sans modifier le calcul.

Toutes les déformations de ce chapitre, ainsi que celles des deux chapitres suivants, correspondent directement aux modifications que l'on peut faire sur les diagrammes espace-temps. À chaque fois, nous illustrons les transformations sur un digramme espace-temps périodique. Nous utilisons un vocabulaire géométrique pour désigner les transformations et indiquer à quoi elles correspondent sur les diagrammes espace-temps.

5.1 Similarité et emboîtement

L'évolution d'une machine à signaux est complètement définie par l'ensemble des collisions et des signaux présents. Ces signaux sont définis à partir des signaux initiaux et des signaux sortant des collisions. Tous les signaux de la configuration initiale et toutes les collisions déterminent donc le calcul. On peut même se passer des signaux initiaux, car chacun d'eux, soit se retrouve en entrée d'une collision, soit n'intervient dans aucune collision et donc pas dans le calcul (sa présence ou son absence n'a finalement rien changé au calcul).

Les collisions se produisent à des instants donnés, elles sont basiquement pré-ordonnées par leurs dates. Mais cet ordre représente mal les liens de causalité entre les collisions, *e.g.*, la simultanéité ne résulte que du hasard ou d'une construction délibérée ; de même, une collision peut avoir lieu avant une autre mais sans avoir de lien avec elle. Nous définissons donc la causalité par :

Définition 28 (Causalité entre collisions) Il y a un lien de *causalité direct* entre deux collisions π_1 et π_2 lorsqu'un signal issu de π_1 arrive dans π_2. Nous appelons *causalité* la fermeture transitive de la causalité directe. Elle est notée $\pi_1 \prec \pi_2$.

La causalité est un ordre partiel : elle est en effet transitive (par définition) et antisymétrique (car la causalité directe implique des dates distinctes et successives, ce que conserve la fermeture transitive). C'est un sous-ordre de l'ordre suivant les dates.

Définition 29 (Similitude des calculs) Deux calculs Γ_1 et Γ_2 sont dit *similaires* s'il existe une bijection entre les règles entrant en jeu dans les deux calculs identifiant les deux ordres de causalité. Par abus de langage, nous écrivons parfois qu'il s'agit du *même calcul*. Cela est noté $\Gamma_1 \approx \Gamma_2$.

La similitude se conçoit donc au niveau de l'enchaînement des collisions. Les signaux ne participant à aucune collision ne sont donc pas concernés. Savoir si un signal joue un rôle ou non dans un calcul est indécidable (Th. 27 du Chap. 4).

Notons au passage qu'il s'agit bien d'une relation d'équivalence, la composition de bijections étant une bijection.

Plutôt que d'exiger d'avoir le même ordre, on peut se contenter d'un sous-ordre (toujours à un renommage des règles près). Ceci correspond à « extraire » un calcul d'un autre en écartant autant de collisions et de causalités directes que nécessaire. Nous définissons alors la relation de simulation par :

Définition 30 (Emboîtement simple) Un calcul est *(simplement) emboîté* dans un autre s'il existe une surjection des règles utilisées par le second dans celles utilisées dans le premier telle que, par l'intermédiaire de cette surjection, l'ordre de causalité du premier soit un sous-ordre de l'ordre de causalité du second. Si le calcul Γ_1 est emboîté dans Γ_2, alors Γ_2 *englobe* Γ_1.

Attention, ce sous-ordre n'est pas forcément induit. Dans les constructions présentées dans les deux chapitres suivants, nous ajoutons des méta-signaux et des règles aux machines à signaux pour obtenir différentes modifications du diagramme sans perturber le calcul. Cette structure supplémentaire crée de nombreux liens de causalité directe qui ne sont dûs qu'à la structure et ne concernent pas le calcul originel. Par exemple, si un signal de la structure croise beaucoup de signaux du calcul originel en ne provoquant que des collisions blanches (*e.g.* pour changer de côté), il modifie le diagramme mais n'altère pas la nature du calcul originel. Il y a des collisions et des liens de causalité en plus, mais ils ne signifient rien pour le calcul originel.

Il est aussi possible de poser toutes ces définitions en ne considérant pas les collisions blanches. Les signaux se croisent alors simplement sans créer de liens de causalité directe, ce qui provoque moins de causalités. Nous n'avons pas fait ce choix car une collision blanche pourrait n'être qu'un cas particulier dans un ensemble de règles nécessaires au calcul.

Nous aurions pu prendre une approche différente consistant à projeter le calcul englobant en fonction des méta-signaux et des règles de manière à le réduire (disparition de collisions et de liens de causalité directe) et considérer ensuite un sous-ordre induit. Cette définition plus restrictive ne nous a pas semblé pertinente dans le cadre de ce mémoire. En effet, elle complique notre présentation sans rien apporter à notre propos. Par ailleurs il semble que toutes les constructions proposées soient aussi des emboîtements en ce sens.

Deux calculs similaires s'emboîtent l'un dans l'autre. Deux calculs finis s'emboîtant l'un dans l'autre sont similaires (les ordres sont égaux et les deux injections permettent de

construire une bijection). Par contre, deux calculs s'emboîtant l'un dans l'autre ne sont pas nécessairement similaires comme le montre l'exemple de la Fig. 5.1 où les calculs s'étendent à l'infini, de plus en plus larges au fur et à mesure que le temps passe. Les diagrammes espace-temps sont en haut ; les ordres correspondants sont représentés en dessous (les couleurs correspondent aux règles). L'ordre de droite est visiblement un sous-ordre de celui de gauche. L'emboîtement réciproque est indiqué par différents cercles à droite. Dans cet exemple, les couleurs codent les règles de collisions, elles sont identiques de chaque côté.

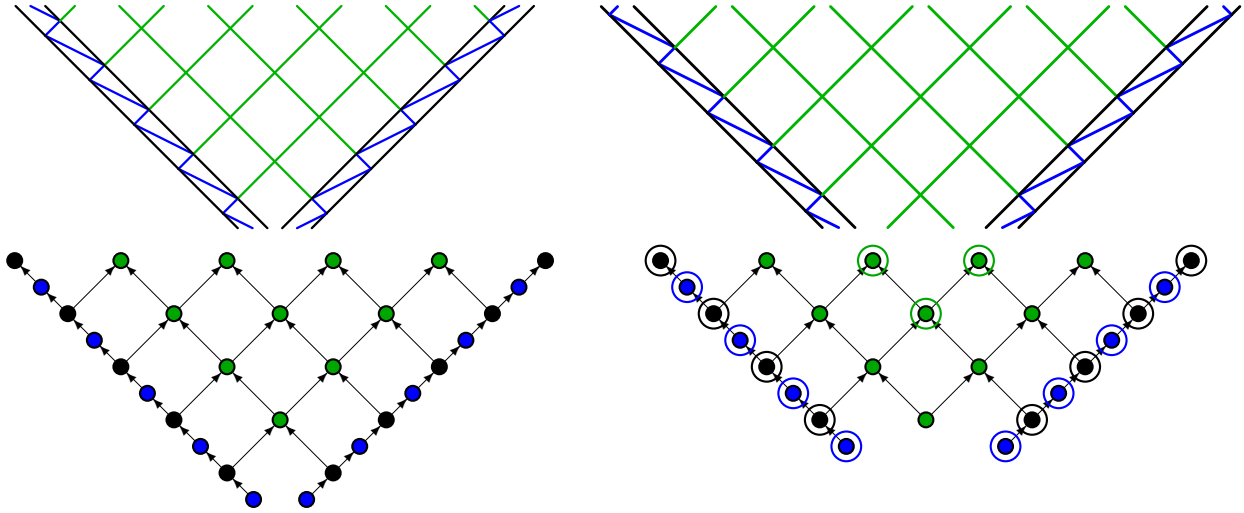


FIG. 5.1 – Deux calculs infinis non similaires mais s'emboîtant l'un dans l'autre.

5.2 Modifications statiques ou totales

Dans cette section, nous rassemblons des modifications que l'on peut apporter à la configuration initiale et / ou à la machine à signaux sans changer le calcul. Les résultats sont plus de l'ordre du fait, de la remarque, mais ils sont utiles : ils permettent de mieux saisir les notions abordées et, par la suite d'avoir des cas généraux très simples.

5.2.1 Positions initiales des signaux

Modifier de manière arbitraire les positions des signaux dans la configuration initiale change généralement complètement le calcul, *e.g.* ce ne sont plus les mêmes rencontres ; il y a néanmoins une certaine stabilité d'échelle.

Théorème 31 (Invariance d'échelle) *Quelle que soit la composition de translations et d'homothéties (de rapport strictement positif) que l'on fait subir à toutes les positions initiales, on obtient le même calcul.*

►► **Preuve.**

Une translation de la configuration initiale provoque une translation identique de tous les signaux et collisions.

Une homothétie de rapport h strictement positif (que l'on peut centrer sur le point de coordonnée 0 à une translation près), déplace une coordonnée du calcul (collision ou position d'un signal) de (x, t) à $(h.x, h.t)$. On retrouve les mêmes vitesses et les mêmes règles.

◁◄◄

Corollaire 32 *Pour une machine rationnelle et une configuration initiale composée d'un nombre fini de signaux, il est possible de faire le même calcul avec la configuration initiale dans* $[0, 1]$ *ou uniquement composée de positions entières positives.*

►► **Preuve.**

Toutes les positions spatiales deviennent positives par retranchement de la plus petite d'entre elles.

Pour les rendre entières, il suffit de les multiplier par le plus petit commun multiple de leurs dénominateurs. Pour les rendre inférieures ou égales à 1 il suffit de les diviser par la plus grande.

◁◄◄

Ces résultats restent vrais en toute dimension. Ils correspondent à l'isentropie de l'espace et à la robustesse par rapport aux changements d'échelle. Ils sont rendus possibles tant localement, puisque le support est continu, que par l'aspect globalement non borné du temps et de l'espace.

La Fig. 5.2 présente le calcul qui va nous servir pour illustrer toutes les transformations, à gauche, et à droite, le même, mais avec une homothétie de rapport $\frac{2}{3}$ des positions des signaux dans la configuration initiale.

Nous avons défini à zéro la valeur du temps pour la configuration initiale, mais nous aurions pu prendre n'importe quelle valeur. De même, nous pouvons partir de la position des signaux à un temps donné t_0 d'un calcul pour faire une nouvelle configuration initiale. Le calcul obtenu est bien évidemment emboîté dans le calcul originel et identique au calcul originel privé des dates de zéro à t_0.

5.2.2 Modifications des vitesses

Modifier de manière arbitraire des vitesses de méta-signaux change généralement complètement le calcul, *e.g.* ce ne sont plus les mêmes rencontres ; il y a néanmoins une certaine stabilité.

Théorème 33 (Invariance par changement affine positif des vitesses) *Quelle que soit la composition d'ajouts de constantes et de multiplications (par un coefficient stricte-ment positif) que l'on fait subir à toutes les vitesses des méta-signaux, on obtient le même calcul.*

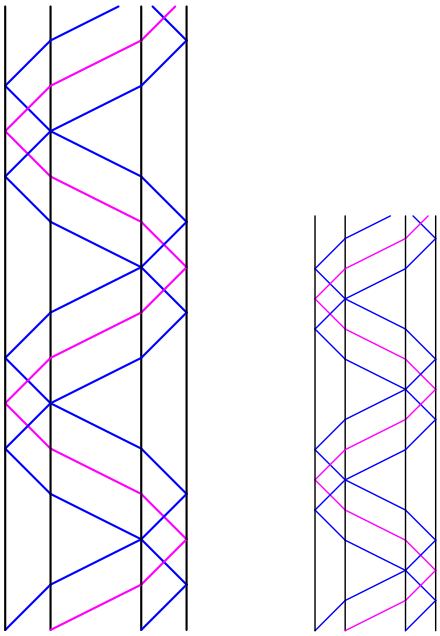


FIG. 5.2 – Bande avant et après une homothétie de la configuration initiale de $\frac{2}{3}$.

►► Preuve.

Un ajout de ν_0 à toutes les vitesses provoque une déviation progressive des positions. Une collision ou un signal à la position (x,t) se retrouve à la $(x+\nu_0.t,t)$ et rien d'autre n'est engendré. Les dates ne sont pas modifiées.

Une multiplication des vitesses par h, n'a qu'une incidence sur la date des positions : (x,t) à $(x,\frac{t}{h})$. On retrouve les mêmes positions spatiales.

◁◀◀

La nullité d'une vitesse ne signifie rien ; par ajout d'un même terme à toutes les vitesses on peut rendre n'importe quelle vitesse nulle. La notion importante est celle de parallélisme.

Si l'on regarde les vitesses dans l'espace-temps, sur le plan projectif, les modifications faites n'impliquent rien car la coordonnée temporelle reste toujours 1.

La Fig. 5.3 montre les diagrammes espace-temps engendrés par différentes modifications affines des vitesses.

Corollaire 34 *Toute machine à signaux rationnelle peut être remplacée par une autre machine réalisant exactement les mêmes calculs dont toutes les vitesses sont positives et entières ou dont toutes les vitesses sont de module inférieur ou égal à* 1.

►► Preuve.

Toutes les vitesses deviennent positives en leur retranchant la plus petite d'entre elles.

Il suffit de multiplier toutes les vitesses pas le plus petit commun multiple des dénominateurs des vitesses. Il suffit de diviser par le plus grand module pour n'avoir que des vitesses de module inférieur ou égal à 1.

◁◀◀

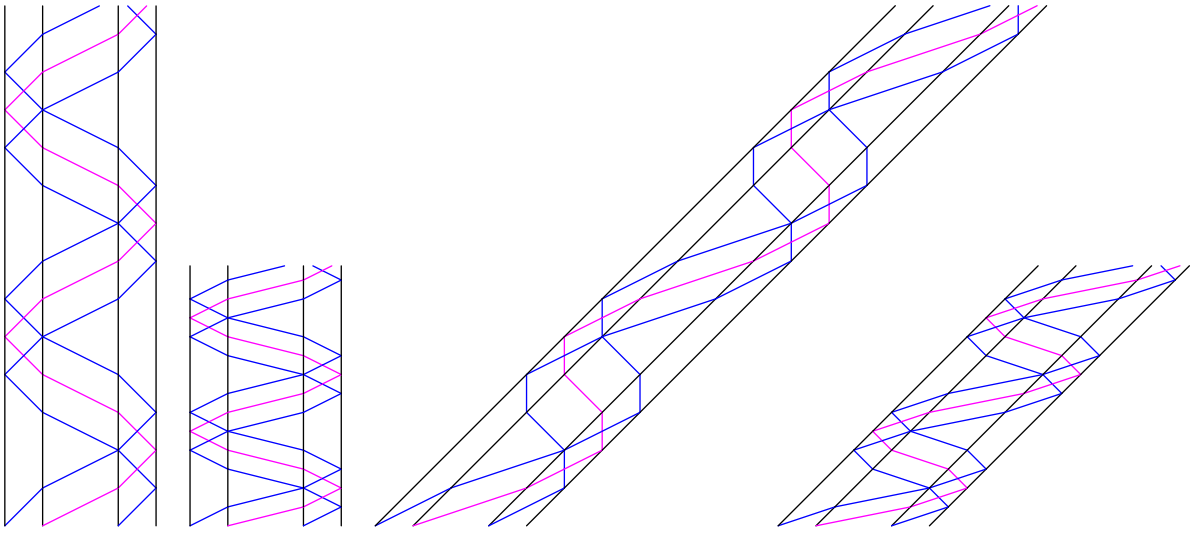


FIG. 5.3 – Modifications des vitesses par différentes fonctions affines.

Toutes les modifications présentées sont élémentaires et ne modifient en rien le calcul. Il est intéressant de noter qu'elles ont des interprétations géométriques directes sur les diagrammes espace-temps. Si l'on considère les diagrammes espace-temps comme des figures (éventuellement infinies) composées de segments de droites sur un demi-plan (ou demi-espace en dimension supérieure), elles correspondent à différentes déformations affines de cet espace.

Il est aussi possible de multiplier par -1 les positions spatiales initiales et les vitesses. Le même calcul est obtenu, toutes les positions spatiales du calcul étant elles aussi multipliées par -1, *i.e.* le symétrique du diagramme espace-temps par rapport à l'axe du temps. En dimension 2 et plus, il est aussi possible de coupler une modification des vitesses et des configurations initiales, *e.g.* des rotations.

Dans ce chapitre, nous avons défini des notions de similarité et d'emboîtement. Nous avons vu qu'il est possible d'accélérer le calcul (rapprochement des points, multiplication des vitesses) ou de le faire se déporter d'un côté ou de l'autre. La modification est alors pour tout le calcul et est déterminée à l'avance sans avoir à connaître le calcul.

Cela nous permet aussi d'affirmer que les largeur, durée et surface ne peuvent servir de mesure de complexité en l'absence de contraintes sur les vitesses et les positions initiales.

Dans les chapitres suivants, nous proposons des méthodes qui permettent de faire cela pour une portion de l'espace-temps, le calcul engendrant automatiquement ces modifications.

Chapitre 6

Translations

Dans ce chapitre et le suivant, nous nous intéressons aux déformations partielles et dynamiques du diagramme espace-temps ; ce que l'on pourrait aussi considérer comme une déformation de l'espace-temps tel qu'il est « perçu » par le calcul originel. Par *partielle* nous entendons que ce n'est qu'une partie, finie ou non, du diagramme espace-temps qui est affectée ; par *dynamique* que cela se fait au cours du calcul.

Pour nos constructions, différentes modifications sont faites aux machines (ajouts de méta-signaux et de règles qui contrôlent les déformations) et des signaux doivent être ajoutés aux configurations initiales. Nous parlons de machine / calcul originel pour désigner les machines / calculs avant construction. Pour chaque construction, nous présentons l'idée et un exemple puis, décrivons les méta-signaux et règles ajoutées et, prouvons sa correction. La construction la plus complexe du chapitre est la contraction de calculs pouvant occuper un espace non borné à une bande dont les positions spatiales sont bornées.

Pour toutes les constructions données dans ce chapitre et le suivant, il n'y a pas d'effets de bord (aucun méta-signal n'est enlevé, aucune règle n'est modifiée). Tous les théorèmes d'emboîtement supposent les conditions sur les paramètres vérifiées et les signaux nécessaires ajoutés à la configuration initiale.

6.1 Translations simples

Il est possible de « geler » un calcul, de le translater et de le « dégeler ». La Fig. 6.1 décrit ce qu'est une *translation* ou *re-localisation* du calcul. Un signal vient d'un côté, il met en mouvement tout ce qu'il rencontre (signal ou collision) avec une même vitesse donnée et sort de l'autre coté. Les signaux sont donc parallèles entre eux et, aucune collision ne peut avoir lieu. Plus tard, un signal correspondant au même méta-signal passe et rechange les signaux en ce qu'ils étaient. Le calcul a été gelé durant la translation. Nous appelons *bascule* le (méta-)signal qui gèle et dégèle le calcul et le notons toggle.

La Fig. 6.2 montre quelques exemples de translations sur notre bande témoin. Il y a un signal en plus sur la droite pour stopper les deux toggle.

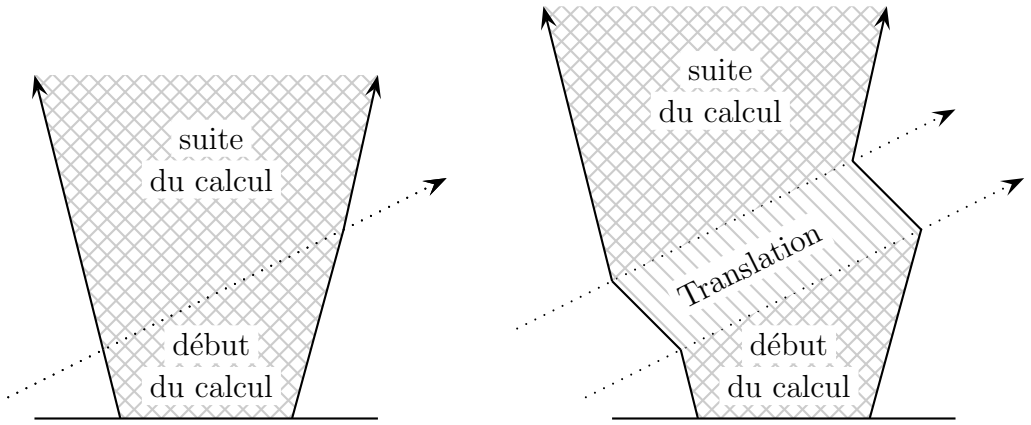


FIG. 6.1 – Principe de translation.

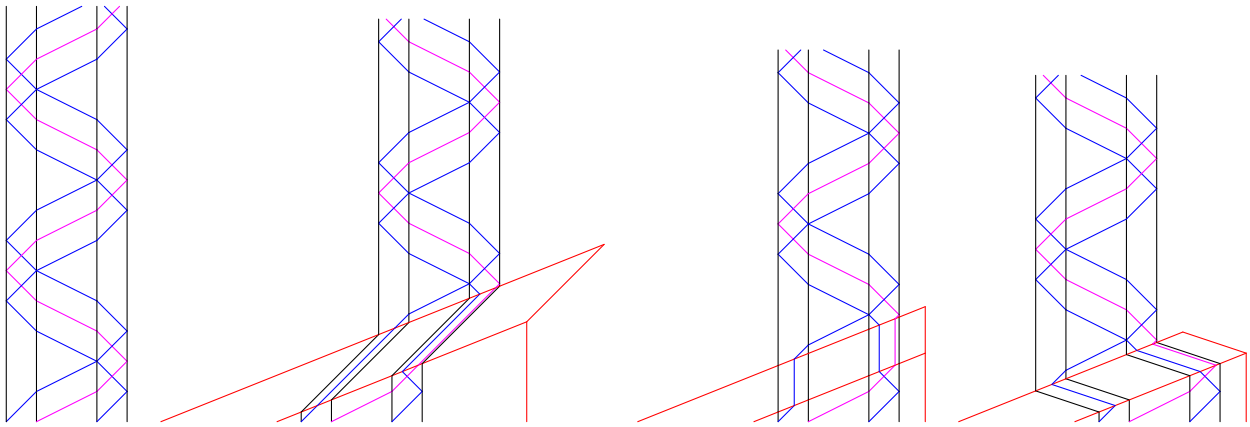


FIG. 6.2 – Exemples de translations.

6.1.1 Construction

Une transition est définie par deux paramètres : β, la vitesse de toggle et, θ, la vitesse des signaux en translation. Les trois parties du calcul, avant, durant et après la translation sont isolées. La modification d'une machine ne peut se faire que si la condition suivante est vérifiée :

$$\big(\ (\nu^{\mathcal{M}}_{max} < \beta\ ) \wedge (\ \theta < \beta\ )\ \big)\ \vee\ \big(\ (\ \beta < \nu^{\mathcal{M}}_{min}) \wedge (\ \beta < \theta\ )\ \big) \tag{6.1}$$

Cette condition est importante car elle signifie que toggle rattrape tous les signaux, que les signaux en translation sont bien entre les deux toggle et qu'une fois dégelé, aucun signal ne peut rattraper toggle.

La machine est modifiée de la façon suivante (l'algorithme est défini sur la Fig. 6.3) :
Tout d'abord le méta-signal toggle est créé, sa vitesse est β.

Pour chaque méta-signal originel μ, un nouveau méta-signal μ' de pente θ est ajouté ainsi que deux règles assurant la bascule entre μ et μ' au passage de toggle. Cela correspond aux lignes 6 à 9 de l'algorithme de la Fig. 6.3.

Entrée :
1: $\mathcal{M}$: machine à signaux
2: β : rationnel { *vitesse de la bascule* }
3: θ : rationnel { *vitesse de translation* }
Antécédent :
4: (($\nu^{\mathcal{M}}_{max} < \beta$) $\wedge$ ($\theta < \beta$)) $\vee$ (($\beta < \nu^{\mathcal{M}}_{min}$) $\wedge$ ($\beta < \theta$))
Faire :
{ *pour la structure* }
5: toggle $\leftarrow$ $\mathcal{M}$.ajouter_nouveau_méta-signal_pente(β)
{ *pour les méta-signaux* }
6: **pour tout** μ méta-signal initial de $\mathcal{M}$ **faire**
7: $\mu' \leftarrow \mathcal{M}$.ajouter_nouveau_méta-signal_pente(θ)
8: $\mathcal{M}$.ajouter_règle({ toggle, μ } $\rightarrow$ { toggle, μ' })
9: $\mathcal{M}$.ajouter_règle({ toggle, μ' } $\rightarrow$ { toggle, μ })
10: **fin pour**
{ *pour les règles* }
11: **pour tout** $\{\mu_i^-\}_i \rightarrow \{\mu_j^+\}_j$ règle initiale de $\mathcal{M}$ **faire**
12: $\mu' \leftarrow \mathcal{M}$.ajouter_nouveau_méta-signal_pente(θ)
13: $\mathcal{M}$.ajouter_règle({ toggle } $\cup \{\mu_i^-\}_i \rightarrow$ { toggle, μ' })
14: $\mathcal{M}$.ajouter_règle({ toggle, μ' } $\rightarrow$ { toggle } $\cup \{\mu_j^+\}_j$)
15: **fin pour**
Sortie :
16: toggle : méta-signal faisant la bascule

FIG. 6.3 – Algorithme pour créer une translation.

Il faut considérer le cas où toggle passe précisément par une collision. Les signaux entrent dans la collision au passage du premier toggle et en ressortent au passage du second. La collision est translatée. Pour chaque règle, un nouveau méta-signal μ' de pente θ est ajouté ainsi que deux règles faisant la bascule entre l'entrée et la sortie de la collision, cela correspond aux lignes 11 à 14 de l'algorithme de la Fig. 6.3.

Pour que la translation fonctionne, il faut que les deux signaux toggle déclenchant et arrêtant la translation soient présents. La mise en place de la translation se fait en mettant deux toggle du bon côté, *i.e.* à gauche si β est la plus grande de toutes les vitesses, à droite si c'est la plus petite.

6.1.2 Correction

Théorème 35 *Tout calcul s'emboîte dans tout calcul obtenu en lui faisant subir un translation. Ils sont similaires en dessous, comme au-dessus, de* toggle.

►► Preuve.
Si aucun signal n'est ajouté, il se passe exactement la même chose car il n'y a eu aucune modification des règles originelles. Si les signaux toggle sont ajoutés du « mauvais » côté, ils ne rencontrent jamais les autres signaux car leur vitesse est plus grande (ou plus petite).

Plaçons nous dans le cas où β vaut 1 et est plus grande que toutes les vitesses présentes, θ est strictement plus petite que 1, la configuration initiale se trouve dans $]0, +\infty[$ et les deux signaux toggle se trouvent aux positions -1 et 0 (le chapitre précédent prouve que ce cas est général). Le cas vitesse de toggle plus petite que le minimum et signaux à droite se traite par symétrie.

Le signal toggle partant de 0 n'est jamais dévié, il rencontre tous les signaux ou collisions (présents sur $x = t$) et les met tous en translation, comme on peut le voir avec les règles de la Fig. 6.3.

Toute la partie du calcul se trouvant dans la portion du diagramme espace-temps délimitée par $t < x$ est identique. La partie se trouvant sur $t = x$ est mise en translation. Tous ses signaux sont parallèles ; il n'y a donc pas de collisions avant de rencontrer le second toggle.

Le second signal toggle avance lui aussi de façon rectiligne uniforme, sa trajectoire est : $x = t - 1$. Rencontrer d'autres signaux ne le dévie pas, comme le premier.

Un signal basculé à la position (a, a) (sur $t = x$) a une vitesse de translation θ et rencontre le second toggle à la position $(a + \frac{1}{1-\theta}, a + \frac{\theta}{1-\theta})$. De là il reprend sa course, après avoir été translaté de $(\frac{1}{1-\theta}, \frac{\theta}{1-\theta})$, quelle qu'ait été sa position initiale sur sur $t = x$.

S'il se produit une collision sur $t = x$, celle-ci est codée par un des nouveaux méta-signaux. Elle part aussi en translation et rencontre le second toggle après une translation identique. Les signaux sortant de la collision apparaissent alors.

Le calcul dans la portion $t > x$ se retrouve donc translaté de $(\frac{1}{1-\theta}, \frac{\theta}{1-\theta})$ et se continue normalement.

Le fait que toutes les vitesses soient plus petites que celles de toggle et que les signaux toggle partent de gauche garantit que le premier toggle atteindra tous les signaux mais qu'une fois translaté, aucun ne peut rattraper le second toggle.

Au niveau de l'ordre du calcul, il y a uniquement ajout de collisions et de liens de causalité. Les collisions situées sur la frontière ont été remplacées par deux collisions, de part et d'autre de la translation. L'ordre de causalité originel est un sous-ordre de l'ordre de causalité engendré. Par contre le nouvel ordre de causalité possède beaucoup de relations dues à la translation mais sans rapport avec le calcul originel.

◁◀◀

En faisant varier β, θ, et l'espace entre les signaux toggle, on peut déplacer la suite du calcul n'importe où sur le diagramme espace-temps, tant que cela se trouve « plus tard ».

On peut composer autant de translations que l'on veut, comme l'illustrent les deux exemples de la Fig. 6.4. On y reconnaît les morceaux du diagramme espace-temps originel découpé par les translations. L'ordre de création des translations et les positions initiales des différents signaux toggle sont très importants car pour faire la seconde translation, il y a déjà eu l'ajout de tous les méta-signaux et règles de la première.

Remarquons au passage que le calcul pour deux translations en l'absence de tout autre signal est aussi emboîté dans n'importe quel calcul obtenu en ajoutant des signaux de la machine originelle.

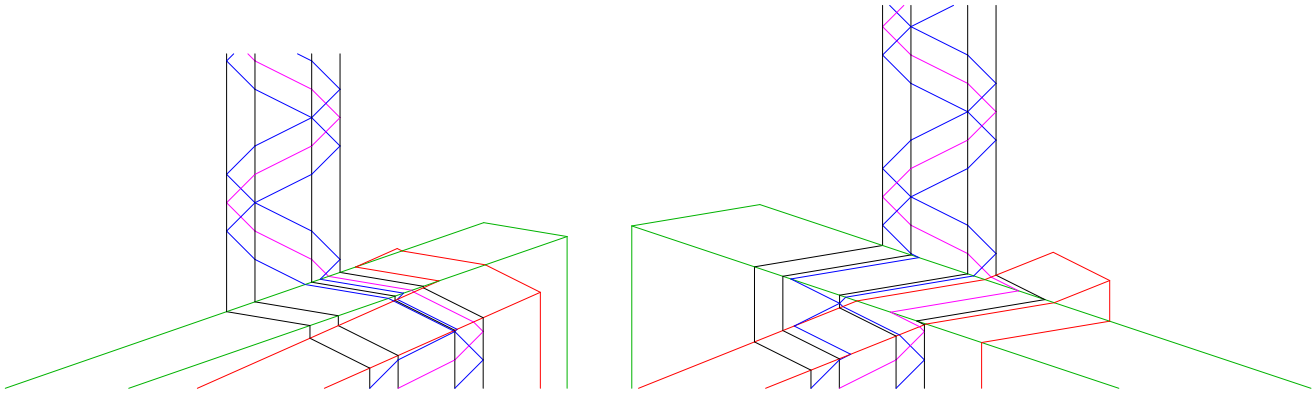


FIG. 6.4 – Exemples de translations l'une sur l'autre.

6.2 Contraction simple du volume du calcul

Avec le mécanisme de gel du calcul et le théorème de Thalès, il est possible de réduire la taille du calcul, *e.g.* de diviser par deux sa durée et sa largeur, et donc, par quatre sa superficie.

L'idée est d'intercaler un changement de direction au cours de la translation de manière à faire une homothétie des signaux avant de dégeler le calcul. Le principe de cet opérateur est illustré par la Fig. 6.5.

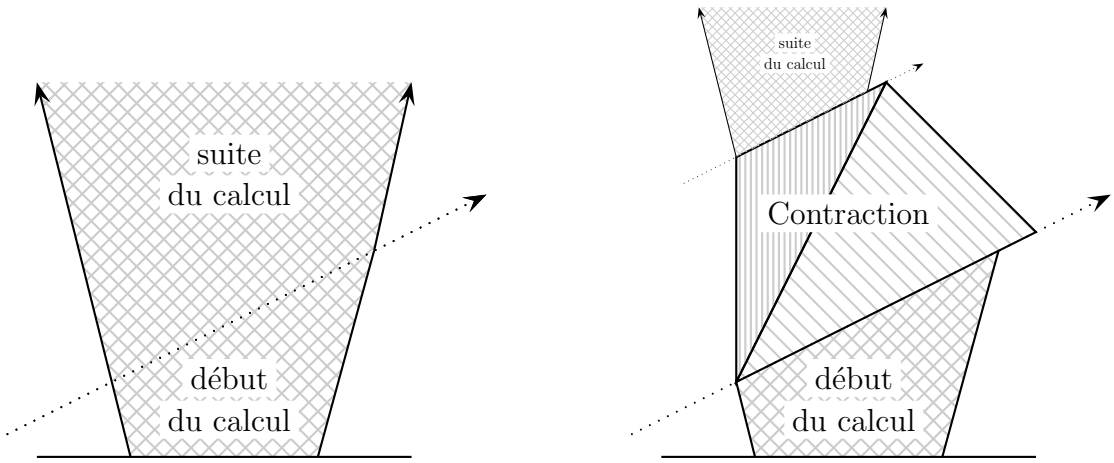


FIG. 6.5 – Principe de contraction du volume.

La Fig. 6.6 montre deux exemples de contraction : la première sur une configuration vide pour montrer la structure du contracteur et la seconde avec notre « bande témoin ».

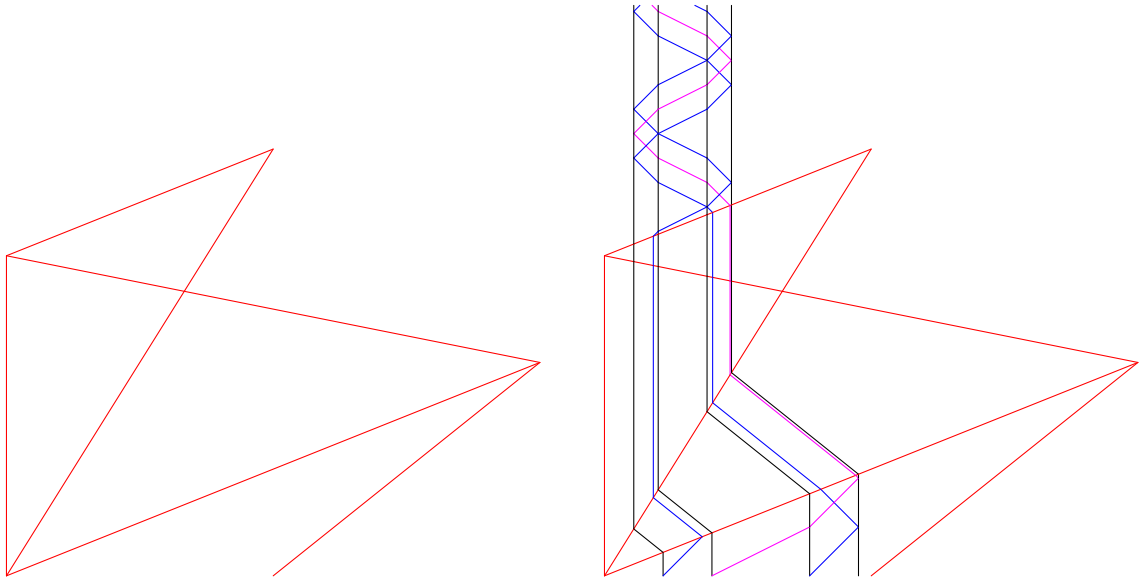


FIG. 6.6 – Structure et exemple de contraction.

6.2.1 Construction

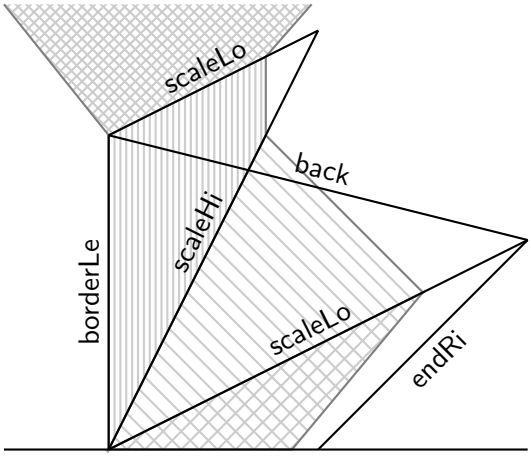
La contraction ne demande qu'un paramètre : une vitesse ν_0. celle-ci doit vérifier la condition suivante :

$$\big(\ (0 < \nu_0) \wedge (\nu_{max}^{\mathcal{M}} < \nu_0)\ \big)\ \vee\ \big(\ (\nu_0 < 0) \wedge (\nu_0 < \nu_{min}^{\mathcal{M}})\ \big) \tag{6.2}$$

Cette condition garantit d'avoir une vitesse supérieure (resp. inférieure) à toutes les autres pouvant marquer la limite du calcul. Être strictement positive (resp. négative) est important car cette vitesse est multipliée par des rationnels strictement positifs ; si elle est nulle toutes ces vitesses sont nulles ; si elles ne sont pas du bon signe, alors la construction devrait être dans le passé ! Par la suite, nous restons dans le cas où ν_0 est positive, l'autre cas étant symétrique.

La contraction simple correspond à la structure de la Fig. 6.7. Tous les signaux subissant la contraction sont représentés en gris. Les parties hachurées sont celles où le calcul originel a lieu ; celles où il n'y a que des lignes parallèles sont les lieux de translation. Détaillons les signaux présents :

- endRi sert à borner la configuration à droite avant que le calcul ne soit entièrement gelé, ainsi qu'à arrêter scaleLo ;
- scaleLo sert à geler le calcul et à le mettre en translation en biais au début de la contraction, et, à dégeler le calcul en translation verticale à la fin ;
- scaleHi sert à changer de translation, ainsi qu'à mettre fin au second scaleLo ;
- borderLe sert à marquer l'extrémité gauche du contracteur, permettant à back d'engendrer le second scaleLo ;
- back est engendré quand scaleLo atteint endRi (tout le calcul est alors gelé), il repart jusqu'à l'autre bord (marqué par borderLe) pour engendrer le second scaleLo qui dégèle le calcul contracté.



Méta-Signal	Vitesse
back	$-4\nu_0$
borderLe	0
endRi	ν_0
scaleHi	$\nu_0/2$
scaleLo	$2\nu_0$

ν_0 : vitesse strictement positive supérieure ou égale à toutes les autres, ici 1.

FIG. 6.7 – Structure et méta-signaux de la contraction simple.

Le calcul est gelé entre les deux scaleLo. Du premier scaleLo à scaleHi, les signaux sont en translations de pente $-\nu_0$; entre scaleHi et le second scaleLo, les signaux sont en translation de vitesse nulle.

L'algorithme pour ajouter à une machine à signaux tous les mécanismes de contraction est donné par Fig. 6.8. Pour la mettre en place, il suffit d'ajouter à la configuration initiale, à gauche, à la même position spatiale, borderLe, scaleHi et scaleLo et, à droite, endRi.

6.2.2 Correction

Théorème 36 *Tout calcul s'emboîte dans tout calcul obtenu en lui faisant subir une contraction. Ces calculs sont similaires en dessous, comme au-dessus, de la partie gelée au calcul originel.*

►► Preuve.

En toute généralité, prenons $\nu_0 = 1$ (et donc toutes les vitesses strictement inférieures à 1) et la configuration initiale incluse dans $]0, 1[$. La translation est valide puisque la condition (6.1) est vérifiée pour $\beta = 2\nu_0$ et $\theta = -\nu_0$. Les signaux borderLe, scaleHi et scaleLo sont ajoutés à gauche en position 0 et, endRi à droite en position 1. Le calcul des positions des collisions de la structure donne :

– $(2, 1)$ pour la collision entre scaleLo et endRi,
– $(0, \frac{3}{2}$ pour la collision entre borderLe et back, et
– $(1, 2)$ pour la collision entre scaleLo et scaleHi.

Le calcul est entièrement gelé, en effet, la contrainte sur ν_0 empêche que endRi soit rattrapé, il borne bien le calcul. Le calcul ne peut non plus déborder à gauche car les signaux sont mis en translation dès qu'ils rencontrent le premier scaleLo. Le premier signal scaleLo étant deux fois plus rapide que tout signal du calcul, il rattrape forcément tous les signaux et gèle le calcul au passage. Tout le calcul se trouve donc gelé et mis en translation suivant la direction $(-1, 1)$ au passage de la droite d'équation $x = 2t$.

Entrée :
1: $\mathcal{M}$: machine à signaux
2: ν_0 : rationnel { *vitesse bornant* }
Antécédent :
3: (($0 < \nu_0$) $\wedge$ ($\nu^{\mathcal{M}}_{max} < \nu_0$)) $\vee$ (($\nu_0 < 0$) $\wedge$ ($\nu_0 < \nu^{\mathcal{M}}_{min}$))
Faire :
{ *pour la structure* }
4: back $\leftarrow$ $\mathcal{M}$.ajouter_nouveau_méta-signal_pente($-4\nu_0$)
5: borderLe $\leftarrow$ $\mathcal{M}$.ajouter_nouveau_méta-signal_pente(0)
6: endRi $\leftarrow$ $\mathcal{M}$.ajouter_nouveau_méta-signal_pente(ν_0)
7: scaleHi $\leftarrow$ $\mathcal{M}$.ajouter_nouveau_méta-signal_pente($\nu_0/2$)
8: scaleLo $\leftarrow$ $\mathcal{M}$.ajouter_nouveau_méta-signal_pente($2\nu_0$)
9: $\mathcal{M}$.ajouter_règle({ scaleLo, endRi } $\rightarrow$ { back })
10: $\mathcal{M}$.ajouter_règle({ scaleHi, back } $\rightarrow$ { back, scaleHi })
11: $\mathcal{M}$.ajouter_règle({ borderLe, back } $\rightarrow$ { scaleLo })
12: $\mathcal{M}$.ajouter_règle({ scaleLo, scaleHi } $\rightarrow$ { })
{ *pour les méta-signaux* }
13: **pour tout** μ méta-signal initial de $\mathcal{M}$ **faire**
14: $\mu' \leftarrow \mathcal{M}$.ajouter_nouveau_méta-signal_pente(-ν_0)
15: $\mu'' \leftarrow \mathcal{M}$.ajouter_nouveau_méta-signal_pente(0)
16: $\mathcal{M}$.ajouter_règle({ scaleLo, μ } $\rightarrow$ { scaleLo, μ' })
17: $\mathcal{M}$.ajouter_règle({ scaleLo, μ'' } $\rightarrow$ { scaleLo, μ })
18: **fin pour**
{ *pour les règles* }
19: **pour tout** $\{\mu_i^-\}_i \rightarrow \{\mu_j^+\}_j$ règle initiale de $\mathcal{M}$ **faire**
20: $\mu' \leftarrow \mathcal{M}$.ajouter_nouveau_méta-signal_pente(-ν_0)
21: $\mu'' \leftarrow \mathcal{M}$.ajouter_nouveau_méta-signal_pente(0)
22: $\mathcal{M}$.ajouter_règle({ scaleLo } $\cup \{\mu_i^-\}_i \rightarrow$ { scaleLo, μ' })
23: $\mathcal{M}$.ajouter_règle({ scaleLo, μ'' } $\rightarrow$ { scaleLo } $\cup \{\mu_j^+\}_j$)
24: **fin pour**
{ *pour le changement de translation* }
25: **pour tout** μ' **faire**
26: $\mathcal{M}$.ajouter_règle({ back, μ' } $\rightarrow$ { back, μ' })
27: $\mathcal{M}$.ajouter_règle({ back, μ'' } $\rightarrow$ { back, μ'' })
28: $\mathcal{M}$.ajouter_règle({ scaleHi, μ' } $\rightarrow$ { scaleHi, μ'' })
29: $\mathcal{M}$.ajouter_règle({ scaleHi, back, μ' } $\rightarrow$ { scaleHi, back, μ'' })
30: **fin pour**
Sortie :
31: (back, borderLe, endRi, scaleHi, scaleLo)

FIG. 6.8 – Algorithme pour créer une contraction.

Suivons un signal (ou collision) gelé en $(2t_0, t_0)$. La première translation l'amène en $(t_0, 2t_0)$ où il continue à vitesse nulle et est dégelé en $(t_0, \frac{t_0}{2}+\frac{3}{2}$ (sur la droite $x = 2t - 3$). Le passage d'une position à l'autre correspond à une homothétie de rapport $\frac{1}{2}$ et de centre $(0, 0)$ suivi d'une translation de $(0, \frac{3}{2}$.

Aucun signal en translation ne passe après la collision du second scaleLo et scaleHi, car pour cela, il faudrait qu'il ait été mis en translation après la collision du premier scaleLo et de endRi.

Une fois dégelé, le calcul reprend, la seule différence est l'homothétie (de même rapport suivant l'espace et le temps) et une translation. Les résultats du chapitre précédent prouvent que le calcul est identique ensuite. La condition (6.2) sur ν_0 assure qu'au fur et à mesure du dégel, les signaux ne peuvent rattraper scaleLo. Aucune des modifications statiques du chapitre précédent ne change le coefficient d'homothétie ($\frac{1}{2}$), mais le vecteur de translation dépend de la largeur de la configuration initiale et de ν_0.

◁◀◀

Par symétrie, on peut faire une contraction qui commence sur la droite, avec ν_0 soit strictement négative plus petite que toute vitesse originelle, les positions des signaux de structure doivent aussi être modifiées par symétrie.

En augmentant ν_0 il est possible d'avoir des contracteurs aussi plats que l'on souhaite.

La construction est robuste dans le sens où il est possible d'éloigner borderLe et scaleHi autant que l'on veut sur le côté gauche, tant qu'ils restent dans l'ordre, de gauche à droite, borderLe, scaleHi puis scaleLo.

6.3 Contraction du volume à une bande

Il est possible d'itérer les contractions les unes après les autres de manière à ne jamais laisser le calcul s'étendre sur le côté. Le principe est de relancer la contraction à chaque fois. Une translation recale le calcul car celui-ci s'étend aussi sur la gauche dès qu'on le dégèle.

La Fig. 6.9 montre comment un calcul se trouve découpé et enchâssé dans l'itération de contractions. La portion calculante semble occuper une partie restreinte de l'espace, mais comme elle y est avec une échelle de plus en plus réduite, il y a, à chaque fois, deux fois plus de « temps originel » écoulé. Cela se voit aussi avec le découpage du calcul originel : la troisième partie s'achève dans le diagramme espace-temps contracté avant l'originel.

Tous les temps sont accomplis puisque la série de 2^n diverge. Si le calcul originel ne s'arrête pas, le contracté non plus. Mais il n'y a pas non plus de phénomène de type Zénon qui mettrait tous les temps avant une date finie.

La Figure 6.10 montre, à droite, la structure de notre itération et, à gauche, son application sur notre bande témoin. La largeur de la bande est divisée par deux à chaque fois, elle n'est dégelée que dans les parties trapézoïdales où il y a deux fois plus d'aller-retours à chaque fois.

6.3.1 Construction

En partant de la contraction simple, pour la structure, quatre signaux sont ajoutés, ainsi que quatre règles de collisions ; trois règles de la contraction simple sont modifiées. Une série de signaux est ajoutée pour la translation de pente ν_0 ainsi que les règles afférentes.

La condition sur ν_0 est renforcée car cette vitesse devra limiter des deux côtés l'expansion du diagramme espace-temps :

$$\nu_{abs}^{\mathcal{M}} < |\nu_0| \ . \tag{6.3}$$

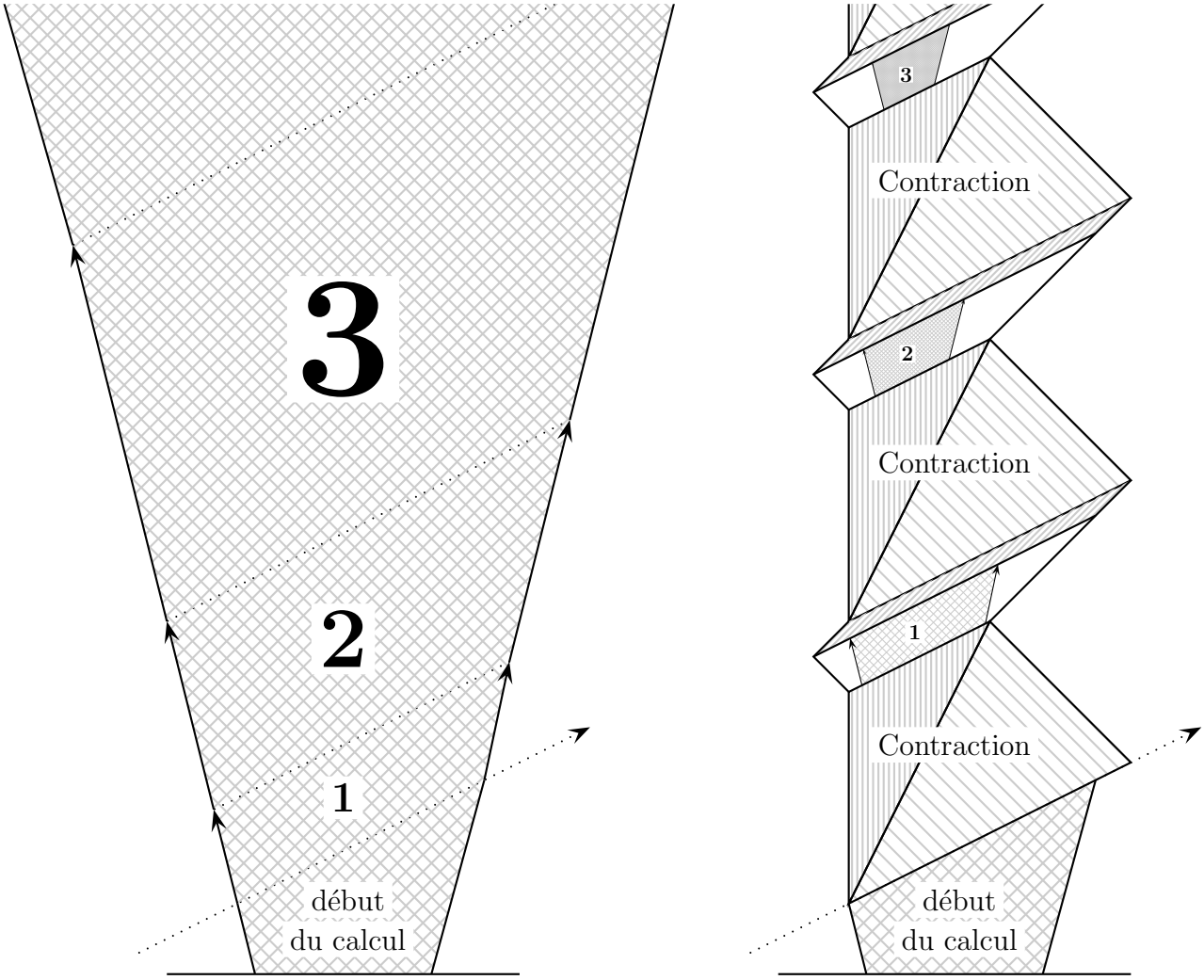


FIG. 6.9 – Principe de contraction itérée du volume.

Comme précédemment, nous ne considérons que le cas positif, l'autre étant symétrique. Nous ne cherchons pas la condition la plus fine mais une qui soit simple à exprimer et à tester statiquement. Les vitesses pouvant prendre n'importe quelle valeur rationnelle, il y a toujours une valeur possible.

La structure de la contraction est décrite sur la 6.11 et l'algorithme correspondant est donné par la 6.12. Par rapport à la contraction simple, les méta-signaux ajoutés sont :

- endLe sert à borner la configuration à gauche ; (6.3) assure qu'il n'est pas dépassé par le calcul ;
- backHi sert à relancer la contraction, sa vitesse un peu particulière ($-\frac{14}{5}\nu_0$) assure que toggle parcourt la même distance à chaque fois[1]. Sa collision avec scaleLo le transforme en back qui plus rapide rattrape endLe et déclenche le gel du calcul ;

[1] On peut bien sûr s'amuser à changer ce coefficient, un peu plus faible, la bande s'élargit ; plus important, elle se contracte avec apparition d'un point d'accumulation. Nous n'avons pas exploré ce cas ici, car dans le chapitre suivant nous fournissons une contraction à une partie finie du diagramme.

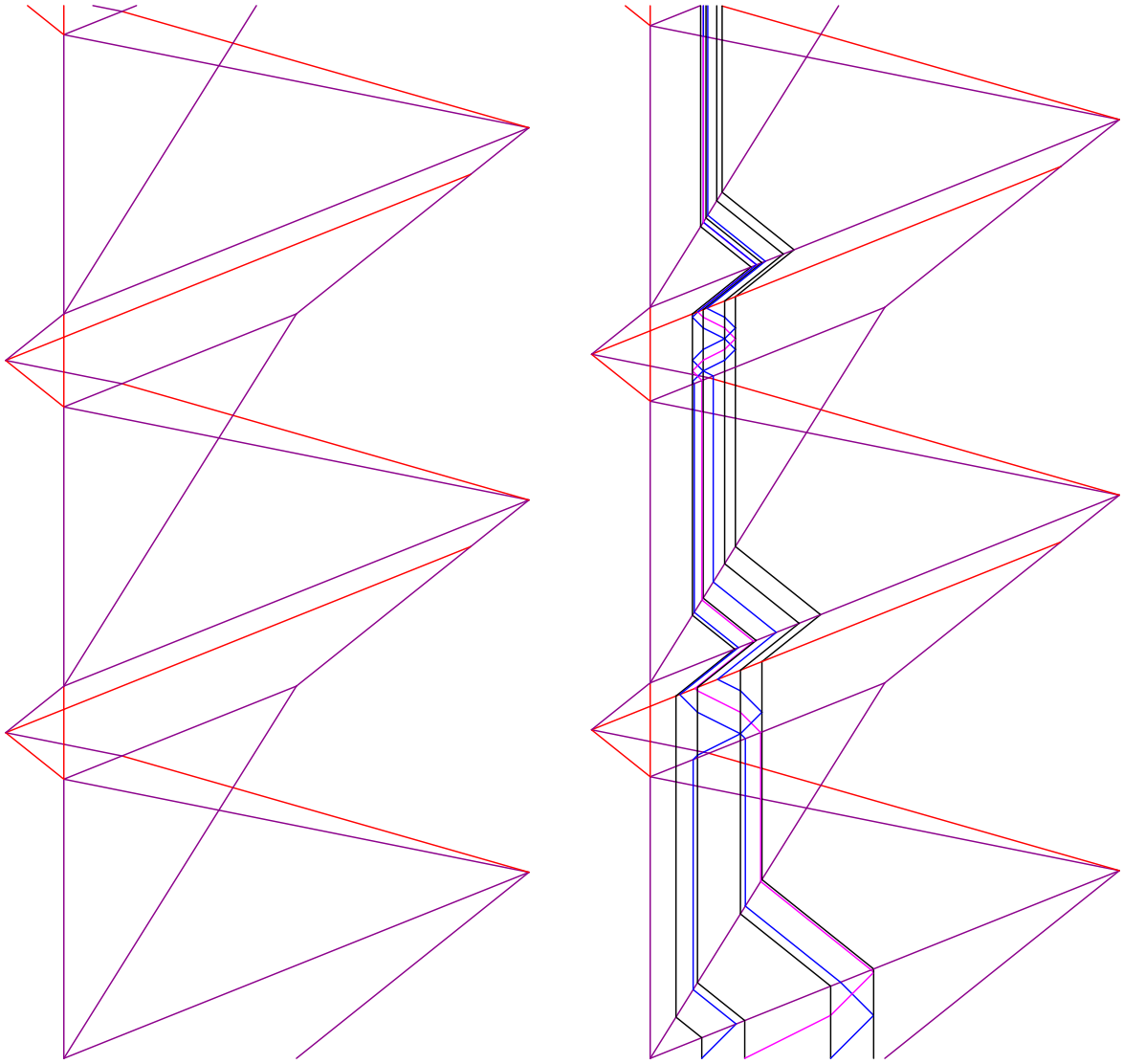


FIG. 6.10 – Structure et exemple de contraction itérée.

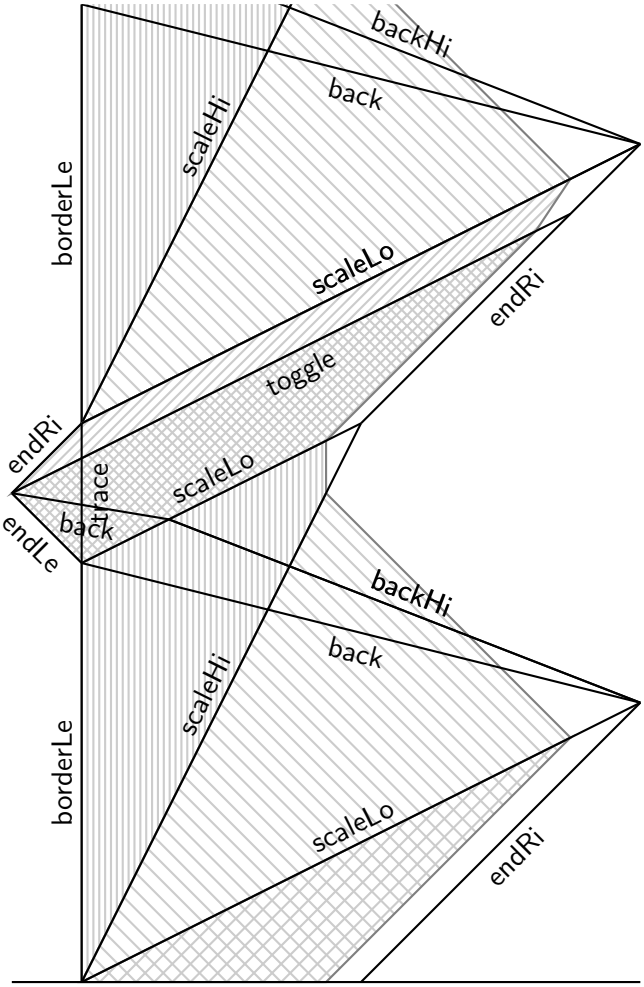
- trace sert à conserver la position de borderLe ; on ne peut garder borderLe sinon le croisement avec back ne pourrait être cohérent avec l'autre collision de ces signaux ;
- toggle sert à geler le calcul et à le mettre en translation, il est différent de scaleLo car la translation n'est pas dans le même sens[2]. La condition (6.1) pour la translation est vérifiée.

À certains signaux de la contraction simple, nous ajoutons des rôles dans notre structure :

- endRi sert à délimiter la translation de chaque côté, à gauche, il relance un scaleLo qui remet les signaux en translation de vitesse $-\nu_0$ relançant ainsi le contacteur ;
- scaleLo sert à passer de la translation de pente ν_0 à celle de pente $-\nu_0$.

Les signaux sont de plus en plus proches, mais comme l'espace est continu, il y a toujours

[2]Il est possible de redéfinir scaleLo de manière à ne plus avoir besoin de toggle ; mais cela compliquerait inutilement la présentation.



Méta-Signal	Vitesse
back	$-4\nu_0$
borderLe	0
endRi	ν_0
scaleHi	$\nu_0/2$
scaleLo	$2\nu_0$
backHi	$-\frac{14}{5}\nu_0$
endLe	$-\nu_0$
toggle	$2\nu_0$
trace	0

ν_0 : vitesse supérieure à toutes les valeurs absolues des autres, ici 1.

FIG. 6.11 – Structure et méta-signaux de la contraction itérée.

de la place. Ce n'est qu'à l'infini qu'il sont « confondus » (quoi que cela puisse signifier). La structure ne provoque nullement l'apparition de points d'accumulation.

6.3.2 Correction

Théorème 37 *Tout calcul s'emboîte dans tout calcul obtenu en lui faisant subir une contraction itérée.*

►► Preuve.

Démontrons cela par récurrence dans le cas général où la configuration initiale est dans $]0,1[$ et $\nu_0 = 1$. Le cas initial correspond à la contraction simple. Montrons que la partie qui relance la contraction fait bien avancer le calcul et le relance correctement.

Un calcul simple donne les positions suivantes pour les collisions de la première relance :

– borderLe et back : $(0, 12)$,

– endLe et back : $(-2, 14)$,
– endRi et trace : $(0, 16)$,
– scaleLo et scaleHi : $(8, 16)$,
– toggle et endRi : $(14, 22)$, et,
– scaleLo et endRi : $(16, 24)$.

Les suivantes sont à chaque fois décalées de $(0, 16)$.

De scaleLo à toggle, le calcul est dégelé, il reste entre endLe et endRi puisque ce sont des vitesses plus petites (resp. grandes) que toutes celles du calcul. De toggle à scaleLo, il y a une translation suivant endRi.

Le calcul gelé se retrouve donc, sous la forme μ'', en entrée de scaleLo où la direction de la translation change pour correspondre à celle du contracteur.

À chaque fois, tout le calcul progresse de $3 * 2^i$ unités de temps puis est dirigé sur un nouveau contracteur déplacé de $(0, 16)$ aux dimensions identiques.

◁◀◀

Une configuration originelle à un instant donné se retrouve tronçonnée entre plusieurs trapèzes mais tout le diagramme originel est emboîté.

Pour finir, une dernière figure pour bien montrer que la méthode ne fonctionne pas uniquement sur des diagrammes espace-temps spatialement bornés (comme notre bande de référence). En partant du diagramme de la Fig. 5.1 de la page 57 qui fait un cône dont les vitesses extrêmes sont -2 et 2, avec $\nu_0 = \frac{11}{10}$, on obtient le diagramme de la Fig. 6.13. Sur cette figure, les quadrillages, deux fois plus petits à chaque fois, se distinguent bien.

La condition (6.3) n'est pas remplie, mais on sait que le bord droit de la configuration avance à vitesse 1, ce qui est en fait la vraie contrainte (on doit alors ajouter celle pour la translation avec $2\nu_0$ ce qui explique le $\frac{11}{10}$). Malheureusement, cette vitesse d'expansion sur la droite est difficilement calculable, mais elle est bornée par la vitesse maximum.

Corollaire 38 *Tout calcul s'emboîte dans un calcul dont le diagramme espace-temps est contenu dans une bande d'espace fini. La construction du diagramme englobant est effective.*

Entrée :
1: $\mathcal{M}$: machine à signaux
2: ν_0 : rationnel { *vitesse bornant* }
Antécédent :
3: ($\nu_{abs}^{\mathcal{M}} < |\nu_0$ |)
Faire :
4: faire_contracteur($\mathcal{M}$, ν_0)
{ *pour la structure* }
5: backHi $\leftarrow$ $\mathcal{M}$.ajouter_nouveau_méta-signal_pente($-\frac{14}{5}\nu_0$)
6: endLe $\leftarrow$ $\mathcal{M}$.ajouter_nouveau_méta-signal_pente($-\nu_0$)
7: trace $\leftarrow$ $\mathcal{M}$.ajouter_nouveau_méta-signal_pente(0)
8: toggle $\leftarrow$ $\mathcal{M}$.ajouter_nouveau_méta-signal_pente($2\nu_0$)
9: $\mathcal{M}$.ajouter_règle({ scaleHi, backHi } $\rightarrow$ { back, scaleHi })
10: $\mathcal{M}$.ajouter_règle({ scaleLo, backHi } $\rightarrow$ { back, scaleLo })
11: $\mathcal{M}$.ajouter_règle({ endLe, back } $\rightarrow$ { endRi, toggle })
12: $\mathcal{M}$.ajouter_règle({ endRi, trace } $\rightarrow$ { borderLe, scaleHi, scaleLo })
13: $\mathcal{M}$.ajouter_règle({ trace, back } $\rightarrow$ { back, trace })
14: $\mathcal{M}$.ajouter_règle({ toggle, trace } $\rightarrow$ { trace, toggle })
15: $\mathcal{M}$.ajouter_règle({ toggle, endRi } $\rightarrow$ { endRi })
16: $\mathcal{M}$.changer_règle({ scaleLo, endRi } $\rightarrow$ { back, backHi })
17: $\mathcal{M}$.changer_règle({ scaleLo, scaleHi } $\rightarrow$ { endRi })
18: $\mathcal{M}$.changer_règle({ borderLe, back } $\rightarrow$ { endLe, trace, scaleLo })
{ *pour les méta-signaux* }
19: **pour tout** μ méta-signal initial de $\mathcal{M}$ **faire**
20: $\mathcal{M}$.ajouter_règle({ back, μ } $\rightarrow$ { back, μ })
21: $\mathcal{M}$.ajouter_règle({ trace, μ } $\rightarrow$ { trace, μ })
22: $\mathcal{M}$.ajouter_règle({ trace, back, μ } $\rightarrow$ { trace, back, μ })
23: $\mathcal{M}$.ajouter_règle({ scaleLo, backHi, μ'' } $\rightarrow$ { back, scaleLo, μ })
24: $\mu''' \leftarrow$ $\mathcal{M}$.ajouter_nouveau_méta-signal_pente(ν_0)
25: $\mathcal{M}$.ajouter_règle({ toggle, μ } $\rightarrow$ { toggle, μ''' })
26: $\mathcal{M}$.ajouter_règle({ toggle, trace, μ } $\rightarrow$ { toggle, trace, μ''' })
27: **fin pour**
{ *pour les règles* }
28: **pour tout** $\{\mu_i^-\}_i \rightarrow \{\mu_j^+\}_j$ règle initiale de $\mathcal{M}$ **faire**
29: $\mathcal{M}$.ajouter_règle({ back } $\cup \{\mu_i^-\}_i \rightarrow$ { back } $\cup \{\mu_j^+\}_j$)
30: $\mathcal{M}$.ajouter_règle({ trace } $\cup \{\mu_i^-\}_i \rightarrow$ { trace } $\cup \{\mu_j^+\}_j$)
31: $\mathcal{M}$.ajouter_règle({ trace, back } $\cup \{\mu_i^-\}_i \rightarrow$ { trace, back } $\cup \{\mu_j^+\}_j$)
32: $\mathcal{M}$.ajouter_règle({ scaleLo, backHi, μ'' } $\rightarrow$ { scaleLo, back } $\cup \{\mu_j^+\}_j$)
33: $\mu''' \leftarrow$ $\mathcal{M}$.ajouter_nouveau_méta-signal_pente(ν_0)
34: $\mathcal{M}$.ajouter_règle({ toggle } $\cup \{\mu_i^-\}_i \rightarrow$ { toggle, μ''' })
35: $\mathcal{M}$.ajouter_règle({ toggle, trace } $\cup \{\mu_i^-\}_i \rightarrow$ { toggle, trace, μ''' })
36: **fin pour**
{ *pour le changement de translation* }
37: **pour tout** μ' engendré **faire**
38: $\mathcal{M}$.ajouter_règle({ backHi, μ' } $\rightarrow$ { backHi, μ' })
39: $\mathcal{M}$.ajouter_règle({ scaleHi, backHi, μ' } $\rightarrow$ { scaleHi, backHi, μ'' })
40: $\mathcal{M}$.ajouter_règle({ backHi, μ'' } $\rightarrow$ { backHi, μ'' })
41: $\mathcal{M}$.ajouter_règle({ trace, μ''' } $\rightarrow$ { trace, μ''' })
42: $\mathcal{M}$.ajouter_règle({ scaleLo, μ''' } $\rightarrow$ { scaleLo, μ' })
43: **fin pour**
Sortie :
44: (back, backHi, borderLe, endLe, endRi, scaleHi, scaleLo, toggle, trace)

FIG. 6.12 – Algorithme pour créer une contraction itérative.

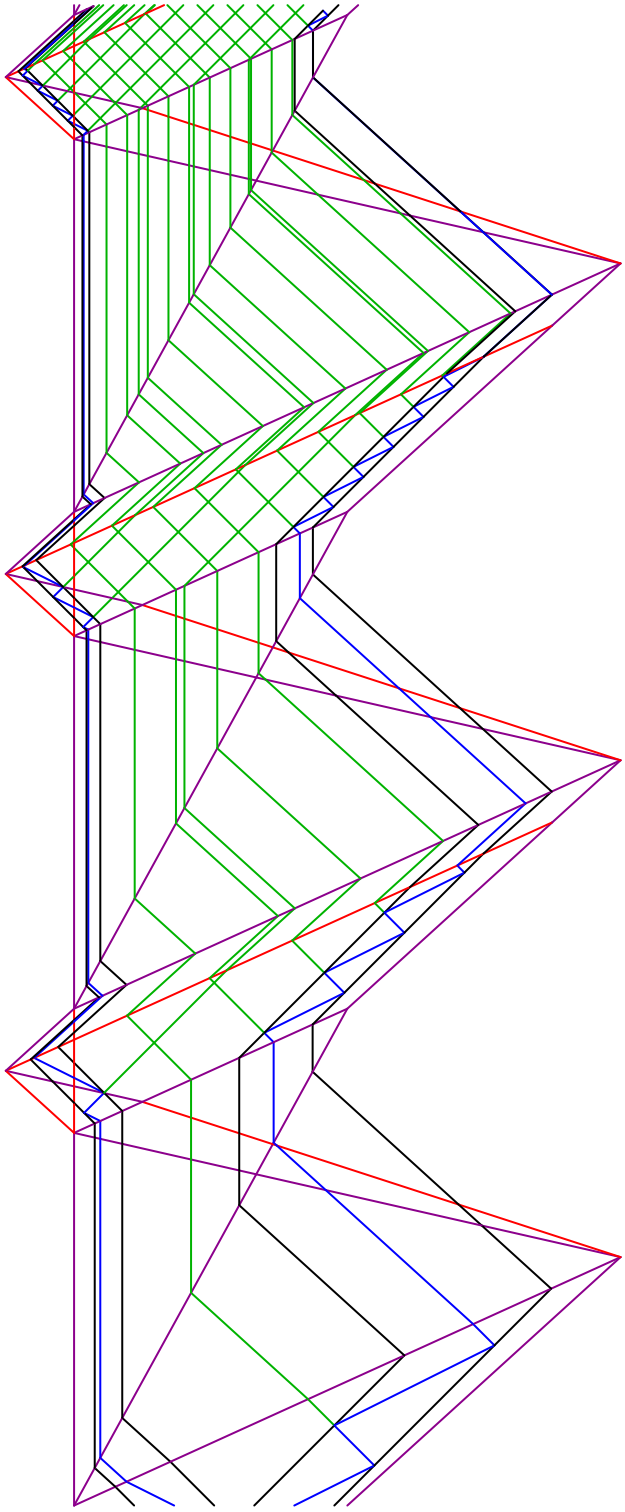


FIG. 6.13 – Contraction itérée sur la « grille » de la Fig. 5.1 de la page 57.

Chapitre 7

Homothéties

Nous continuons l'étude de déformations partielles et dynamiques de l'espace-temps commencée au chapitre précédent.

Dans ce chapitre, nous présentons les homothéties qui permettent de modifier le diagramme espace-temps sans que le calcul soit gelé. Grâce à cela, nous construisons une translation qui ne gèle pas le calcul, *i.e.* il se continue durant la translation.

Le construction la plus complexe que nous présentons est la contraction d'un calcul occupant tout l'espace à une partie finie de l'espace.

7.1 Homothéties simples

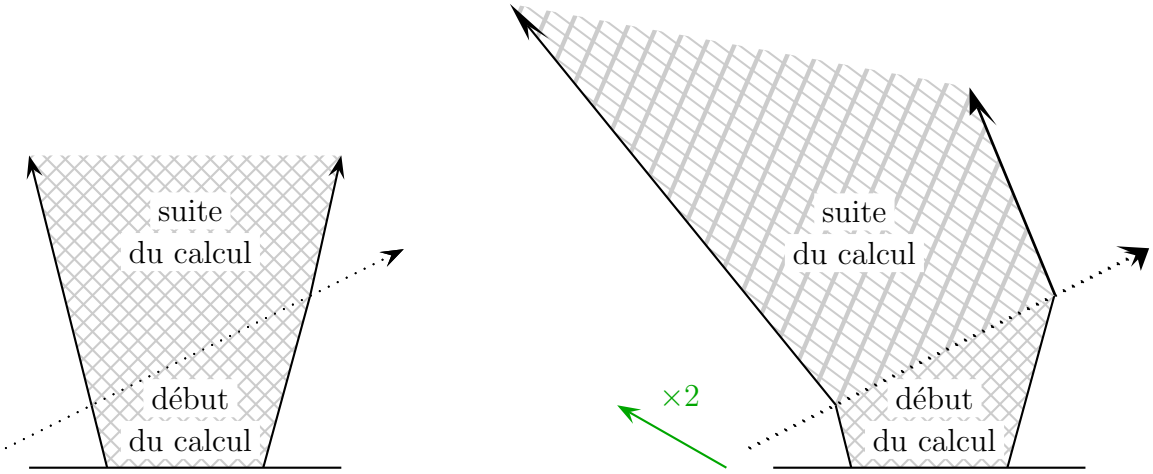
Ce que nous appelons homothétie est une homothétie par rapport à une droite et selon une direction. La droite correspond au passage d'un signal qui va modifier tous les signaux rencontrés de manière à ce qu'au-dessus, le diagramme espace-temps soit l'homothétique du diagramme originel. La partie « sous la droite », antérieure au passage du signal, n'est pas affectée, la partie « au-dessus », postérieure, est déformée mais l'ordre de causalité est respecté.

La Fig. 7.1 décrit le principe de l'homothétie. Nous appelons bascule, toggle, ce signal qui met l'homothétie en place et dont la trace marque la frontière sur le diagramme espace-temps. La suite du calcul est déformée, mais les mouvements sont toujours rectilignes, les règles de collision sont les mêmes au-dessus et en dessous et aucun signal ne va à « rebrousse-temps ».

7.1.1 Construction

Une homothétie se définit par trois paramètres : α, la direction selon laquelle on fait l'homothétie, β, la vitesse du signal bascule et, k, le coefficient d'homothétie (strictement positif). au-dessus de la trace de toggle, la vitesse d'un signal correspond à la vitesse, sous la bascule, décomposée selon (α,1) et (β,1) ayant sa première coordonnée multipliée par k. Cela correspond à :

$$\begin{aligned}(\nu, 1) &= a(\alpha, 1) + b(\beta, 1) \ , \\ \lambda(\nu', 1) &= ka(\alpha, 1) + b(\beta, 1) \ .\end{aligned} \tag{7.1}$$



Le rapport est 2 et la direction est indiquée par le vecteur de gauche.

FIG. 7.1 – Principe d'homothétie.

Un coefficient λ est présent car il faut normaliser la vitesse. Ceci se résout en :

$$\begin{aligned}\lambda(\nu',1) &= \left(\frac{(k\alpha-\beta)\nu+\alpha\beta(1-k)}{\alpha-\beta}, \frac{(k-1)\nu+\alpha-k\beta}{\alpha-\beta}\right) \ , \\ (\nu',1) &= \left(\frac{(k\alpha-\beta)\nu+\alpha\beta(1-k)}{(k-1)\nu+\alpha-k\beta}, 1\right) \ . \end{aligned} \tag{7.2}$$

Les valeurs de α, β et k ne peuvent être quelconques. La première contrainte est que α et β soient différentes. La seconde est qu'aucun signal ne parte à « rebrousse-temps » : λ doit donc être toujours strictement positif. La troisième est que le signal continue bien de l'autre côté du signal bascule ; attention, un signal au-dessus du signal bascule peut repasser en dessous ; cela se traduit par ($\nu < \beta$ et $\nu' < \beta$) ou ($\beta < \nu$ et $\beta < \nu'$). Les inégalités sont strictes car on refuse de superposer avec la bascule[1].

Ces conditions correspondent à la formule suivante :

$$\left\{\begin{array}{l} 0 < k, \\ \alpha \neq \beta \ , \\ \\ \forall \nu \text{ présent dans la machine, } \left\{\begin{array}{rcl} \nu & \neq & \beta \ , \\ 0 & < & \dfrac{(k-1)\nu+\alpha-k\beta}{\alpha-\beta} \ , \\ \beta < \nu & \Leftrightarrow & \beta < \dfrac{(k\alpha-\beta)\nu+\alpha\beta(1-k)}{(k-1)\nu+\alpha-k\beta} \ . \end{array}\right. \end{array}\right. \tag{7.3}$$

[1]Si on le désire, cela est possible car notre décomposition implique alors $\nu = \beta = \nu'$ ce qui est cohérent avec les deux côtés de la bascule. De tels méta-signaux pourraient être produits par des collisions sur la bascule, il faut alors prévoir de nouveaux méta-signaux correspondant à la superposition d'un ou plusieurs méta-signaux avec la bascule. Ces nouveaux méta-signaux servant aussi de bascule, il faut alors définir les règles de bascule comme pour toggle. C'est fastidieux, mais heureusement le nombre de méta-signaux supplémentaires est fini, au maximum exponentiel.

Nous ne cherchons pas à connaître exactement l'ensemble des solutions de (7.3) cela ne présentant que peu d'intérêt pour notre propos. Cette condition doit être vérifiée et peut être testée simplement en machine (cela a été implanté). Nous nous bornons à signaler un cas, assez vaste, où elle est forcément vérifiée : $0 < k$, α (β) plus petite (grande) que toutes les vitesses. Les coefficients a et b de (7.1) étant nécessairement positifs dans ce cas.

La Fig. 7.2 montre deux exemples où la vitesse de bascule est plus grande que toutes les autres, mais pas la direction ainsi qu'un exemple où aucune des deux vitesses n'est extrémale. Ce dernier exemple illustre de plus une collision sur le passage de la bascule ainsi que le passage dans la partie du dessous d'un signal bleu au milieu, ce qui ne gène pas la poursuite du calcul.

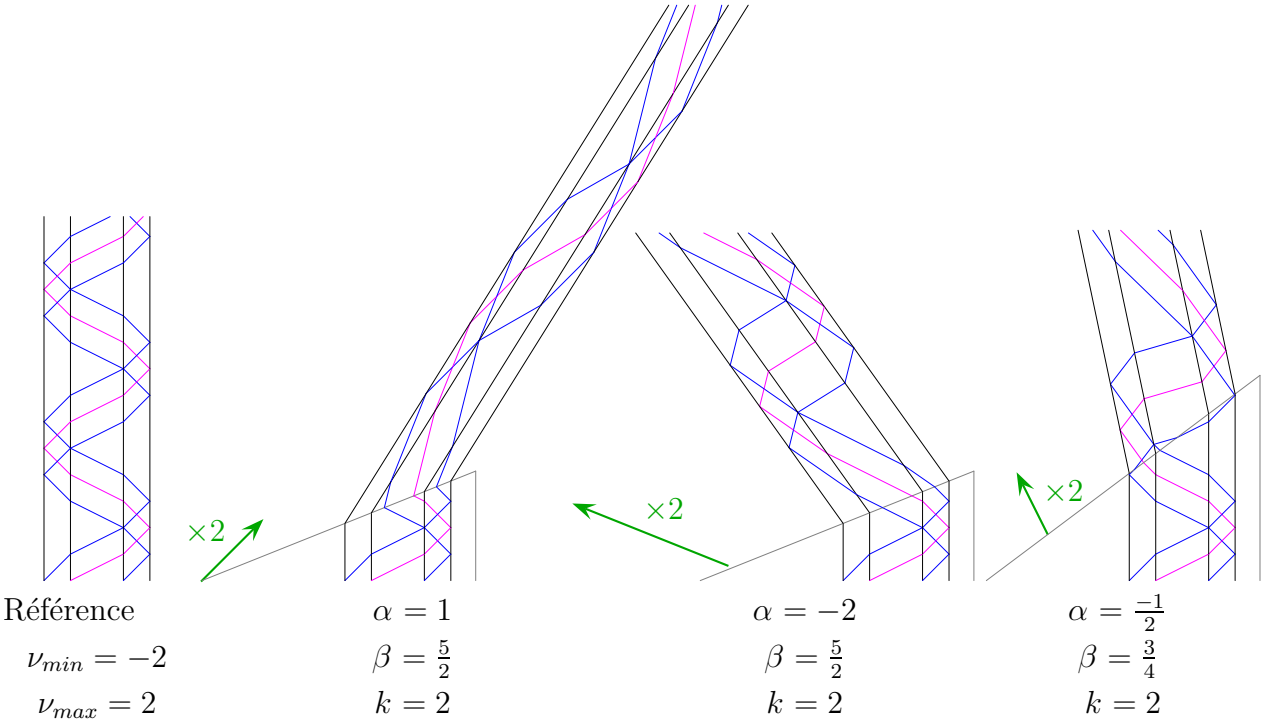


FIG. 7.2 – Différentes homothéties.

Dans la suite, nous considérons le cas général, pas seulement le cas où α et β encadrent les autres vitesses et, nous supposons (7.3) toujours vérifiée.

7.1.2 Modification de la machine

Par rapport à la translation, hors du cas des vitesses extrémales, il faut considérer les cas où un signal repasse en dessous du signal bascule. L'adaptation de la machine pour mettre en place une homothétie se fait selon l'algorithme de la Fig. 7.3.

Pour chaque méta-signal μ, un nouveau méta-signal μ' de pente ν' calculée suivant (7.2) ainsi que deux règles faisant la bascule entre μ et μ' sont ajoutés. Cela correspond aux lignes 7 à 10 de l'algorithme. Les deux règles sont présentes car un signal du haut peut repasser en dessous pour continuer le calcul comme sur l'exemple de droite de la Fig. 7.2.

Entrée :
1: $\mathcal{M}$: machine à signaux
2: α : rationnel { *vitesse de le direction de l'homothétie* }
3: β : rationnel { *vitesse de la bascule* }
4: k : rationnel { *rapport de l'homothétie* }
Antécédent :
5: L'équation (7.3) est vérifiée
Faire :
6: toggle $\leftarrow \mathcal{M}$.ajouter_nouveau_méta-signal_de_pente(β)
{ *ajout de méta-signaux* }
7: **pour tout** μ méta-signal de $\mathcal{M}$ **faire**
8: $\mu' \leftarrow \mathcal{M}$.ajouter_nouveau_méta-signal_pente($((k\alpha-\beta)\nu + \alpha\beta(1-k)) \,/\, ((k-1)\nu + \alpha - k\beta)$)
9: $\mathcal{M}$.ajouter_règle({ toggle, μ } $\rightarrow$ { toggle, μ' })
10: $\mathcal{M}$.ajouter_règle({ toggle, μ' } $\rightarrow$ { toggle, μ })
11: **fin pour**
{ *ajout de règles* }
12: **pour tout** règle $\{\mu_i^-\}_i \rightarrow \{\mu_j^+\}_j$ de $\mathcal{M}$ **faire**
13: $\mathcal{M}$.ajouter_règle($\{\mu_i^{-\prime}\}_i \rightarrow \{\mu_j^{+\prime}\}_j$)
14: $\mathcal{M}$.ajouter_règle($\{\text{toggle}\} \cup \{\mu_i^-\}_{\mu_i^-.\nu<\beta} \cup \{\mu_i^{-\prime}\}_{\mu_i^-.\nu>\beta} \rightarrow \{\text{toggle}\} \cup \{\mu_j^+\}_{\mu_j^+.\nu>\beta} \cup \{\mu_j^{+\prime}\}_{\mu_j^+.\nu<\beta}$)
15: $\mathcal{M}$.ajouter_règle($\{\text{toggle}\} \cup \{\mu_i^{-\prime}\}_{\mu_i^-.\nu<\beta} \cup \{\mu_i^-\}_{\mu_i^-.\nu>\beta} \rightarrow \{\text{toggle}\} \cup \{\mu_j^{+\prime}\}_{\mu_j^+.\nu>\beta} \cup \{\mu_j^+\}_{\mu_j^+.\nu<\beta}$)
16: **fin pour**
Sortie :
17: toggle : méta-signal faisant la bascule

FIG. 7.3 – Algorithme pour créer une homothétie.

La partie homothétique du calcul doit suivre les mêmes règles. Pour chaque règle, une règle est donc ajoutée dans laquelle tous les méta-signaux sont remplacés par les nouveaux correspondants.

Il faut aussi considérer le cas où toggle passe précisément par une collision. La collision peut se produire avec des signaux venant du dessous comme du dessus ; les signaux issus pouvant continuer au-dessus comme en dessous. Supposons que β soit positive, tous les signaux ayant une vitesse inférieure viennent du dessous, mais ceux de vitesse supérieure viennent du dessus, de même, les sortants ayant une vitesse inférieure passent au-dessus, les autres en dessous. Pour chaque règle, il faut ajouter une règle correspondante où sont remplacés, en entrée, les méta-signaux de vitesse supérieure par les nouveaux correspondants et, en sortie, ceux de vitesse inférieure par les nouveaux correspondants ; toggle est ajouté en entrée comme en sortie. De même si β est négative, ce sont ceux de vitesse inférieure en entrée et supérieure en sortie que l'on remplace par les nouveaux correspondants. Pour simplifier, et fournir en même temps l'homothétie de rapport inverse, les deux règles sont ajoutées. Les lignes 15 et 14 de l'algorithme montrent le traitement particulier des règles sur la bascule.

7.1.3 Correction

Théorème 39 *Tout calcul s'emboîte dans tout calcul obtenu en lui faisant subir une homothétie.*

►► Preuve.

On se place dans le cas où α vaut -1 et β, 1 et où, toggle se trouve à gauche de la configuration, à la position 0. Grâce aux résultats de la Sect. 5.2 et par symétrie, ce cas est bien le plus général. Par construction de l'homothétie, (7.3) reste vérifiée quand toutes les vitesses (α et β incluses) sont remplacées par des combinaisons linéaires. On note toujours avec une apostrophe les méta-signaux et signaux qui se trouvent au-dessus de toggle.

La portion sous toggle est identique à la portion originelle.

Pour la portion au-dessus, un point de coordonnée (x,t) se décompose :

$$\begin{pmatrix} x \\ t \end{pmatrix} = a \begin{pmatrix} -1 \\ 1 \end{pmatrix} + b \begin{pmatrix} 1 \\ 1 \end{pmatrix} \ ,$$
$$a = \frac{-x+t}{2} \ ,$$
$$b = \frac{x+t}{2} \ ,$$
$$\begin{pmatrix} x' \\ t' \end{pmatrix} = \begin{pmatrix} \dfrac{(1+k)x+(1-k)t}{2} \\ \dfrac{(1-k)x+(1+k)t}{2} \end{pmatrix} \ .$$

Les nouvelles vitesses se calculent par :

$$\nu \ \longmapsto \ \frac{(1+k)\nu+1-k}{(1-k)\nu+1+k} \ .$$

Ce qui correspond bien à une homothétie du diagramme espace-temps.

La construction et (7.3) garantissent qu'au-dessus de toggle, il n'y a que des signaux primés et, en dessous, uniquement des signaux originels. Les collisions à cheval sur toggle respectent l'homothétie.

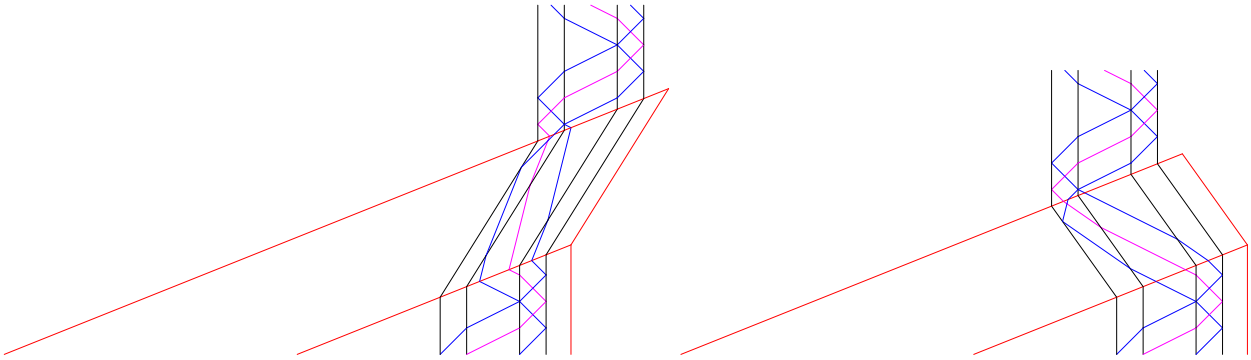
◁◀◀

Comme pour les translations, on peut construire des homothéties sur des homothéties, le diagramme obtenu englobe toujours le calcul originel. Il est possible de faire des constructions utilisant plusieurs homothéties en même temps.

7.2 Translations sans arrêt du calcul

Une homothétie peut être faite puis défaite par l'homothétie de coefficient inverse. Les signaux primés redeviennent les signaux originels. Notre construction produit aussi l'homothétie inverse puisqu'elle permet de passer, avec toggle, des signaux originaux aux primés

et vice-versa. C'est une des raisons pour lesquelles il n'y a pas de condition du type « si β est positif » dans l'algorithme de la Fig. 7.3. On peut donc « redresser » la suite de l'homothétie par un autre toggle comme l'illustrent les exemples de la Fig. 7.4.



$\alpha = 1$ $\beta = \frac{5}{2}$ $k = 2$ — $\alpha = -2$ $\beta = \frac{5}{2}$ $k = 2$

FIG. 7.4 – Translations sans arrêt du calcul par homothétie et redressement.

7.2.1 Correction

Théorème 40 *Avec un second* toggle*, une translation du calcul est réalisée, mais le calcul se poursuit durant la translation.*

►► Preuve.

En toute généralité, on se place dans le cas où α vaut -1 et β, 1 et où, deux toggle se trouvent à gauche de la configuration, aux positions -1 et 0. Les positions sur $x' + 1 = t'$ correspondent aux positions sur $x' + \frac{1}{k} = t'$. On passe de celles du calcul originel à celles après homothétie par une translation de $\frac{k-1}{2k}(-1, 1)$. Comme toutes les vitesses ont été rétablies après le second toggle, on retrouve tout le calcul originel au-dessus de $x' + \frac{1}{k} = t'$ translaté de $\frac{k-1}{2k}(-1, 1)$.

◁◀◀

Si β n'est pas extrémale, il est possible qu'il y ait une partie infinie du diagramme espace-temps qui ne rentre pas dans la translation ou ne la termine jamais.

On peut translater le calcul où l'on veut en prenant des α et β extrémales ; mais la frontière (la trace des toggle) ne peut être quelconque à cause de la contrainte (7.3).

7.3 Plusieurs zones d'homothéties

Il est possible d'avoir plusieurs zones d'homothéties différentes sans que ce soit une homothétie construite sur une autre. Mais, à chaque fois, les frontières doivent être parallèles

aux directions des homothéties sinon les diagrammes espace-temps ne se recollent pas. La Fig. 7.5 montre un tel mélange. La frontière entre deux zones d'homothéties est la direction commune à ces deux homothéties.

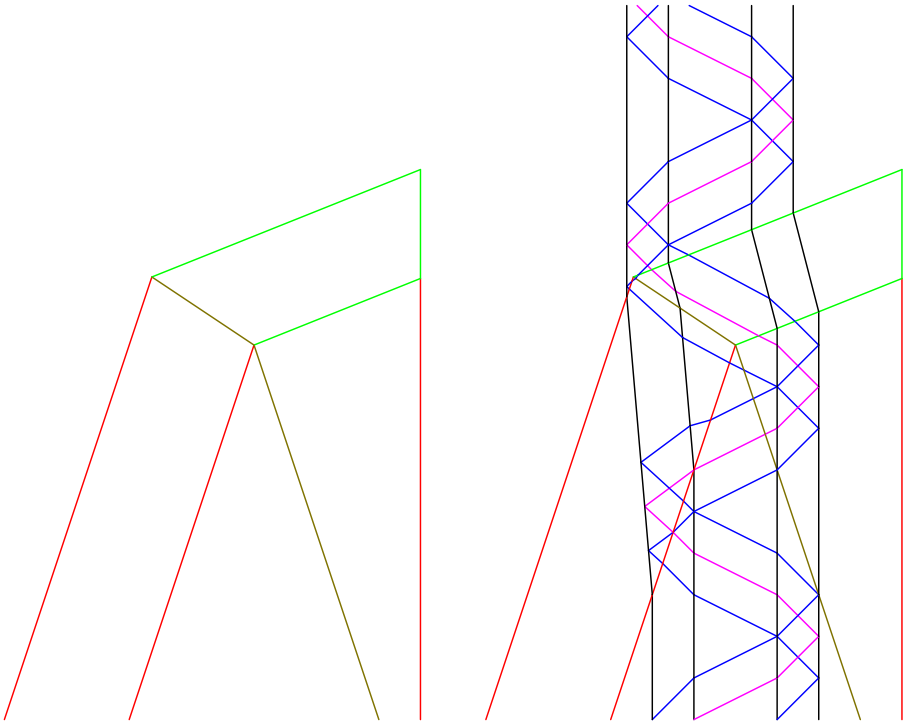


FIG. 7.5 – Deux zones d'homothétie.

7.4 Contraction simple continue

Géométriquement, il est possible de faire une homothétie centrale à partir de deux homothéties selon des droites. Nous pouvons également faire cela avec nos homothéties.

Nous nous plaçons ici dans un cadre très particulier : le diagramme espace-temps est continu dans une bande, *i.e.* les positions spatiales sont bornées. Il est toujours possible d'emboîter un calcul dans une telle bande comme le prouve le Th. 37 du chapitre précédent. Cette condition n'est pas obligatoire, mais elle simplifie la construction et permet d'itérer la construction de la section suivante.

7.4.1 Construction

Elle est relativement simple : deux directions sont prises, on fait tout d'abord une homothétie de rapport $\frac{1}{2}$ selon l'une et suivant l'autre puis une seconde homothétie où l'on a permuté les directions.

La Fig. 7.6 montre la structure et l'utilisation de deux homothéties pour diminuer par deux la taille de la configuration et donc accélérer par deux le temps de calcul. Le nombre de collisions n'est pas pour autant diminué.

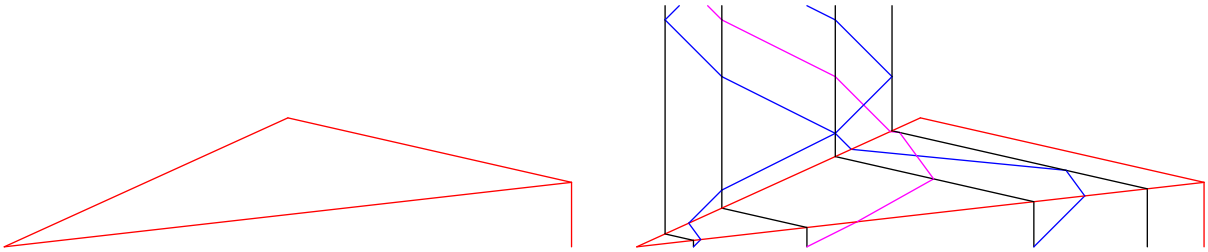
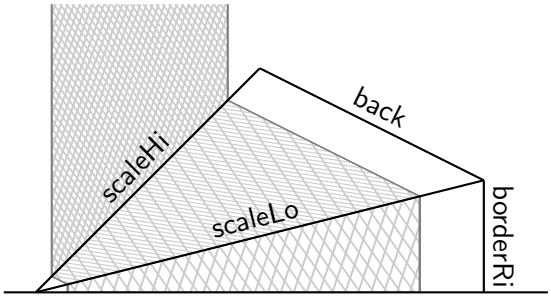


FIG. 7.6 – Structure et exemple de contraction continue.

La Fig. 7.7 donne la structure de cet opérateur. Quatre méta-signaux sont ajoutés à la machine :

- borderRi borne à droite le calcul,
- back borne à droite le calcul durant l'homothétie,
- scaleHi et scaleLo marquent les frontières des homothéties.

Les deux premiers méta-signaux ne sont importants que pour que la structure soit finie. Dans la section suivante, ils sont très importants pour itérer la construction.



Méta-Signal	Vitesse
back	$-2\nu_0$
borderRi	0
scaleHi	ν_0
scaleLo	$4\nu_0$

ν_0 : vitesse strictement positive, strictement supérieure à toutes les valeurs absolues des autres, ici 1.

FIG. 7.7 – Structure et méta-signaux de la contraction continue simple.

La première homothétie est donc selon scaleHi et suivant scaleLo avec un rapport de $\frac{1}{2}$ (sur scaleHi). Le seconde est selon scaleLo et suivant scaleHi avec un rapport de $\frac{1}{2}$ (sur scaleLo).

Pour que cela fonctionne, il faut que (7.3) soit remplie. Le plus simple pour cela est que ν_0 soit suffisamment grande, en particulier :

$$\nu_{abs}^{\mathcal{M}} < \nu_0 \quad . \tag{7.4}$$

Avec $\beta = 4\nu_0$, $\alpha = \nu_0$ et $k = \frac{1}{2}$, (7.2) devient :

$$(\nu', 1) = \left(\nu_0 \frac{7\nu - 4\nu_0}{\nu + 2\nu_0}, 1\right) \quad . \tag{7.5}$$

Cela correspond à l'algorithme de la Fig. 7.8. Le code est écrit en considérant cela : scaleLo démarre l'homothétie et scaleHi la termine. Nous sommes dans un des cas où les homothéties se recollent bien car scaleHi est aussi la direction de l'homothétie.

Entrée :
1: $\mathcal{M}$: machine à signaux
2: ν_0 : rationnel { *vitesse de la contraction* }
Antécédent :
3: L'équation (7.4) est vérifiée
Faire :
4: back $\leftarrow \mathcal{M}$.ajouter_nouveau_méta-signal_de_pente($-2\nu_0$)
5: borderRi $\leftarrow \mathcal{M}$.ajouter_nouveau_méta-signal_de_pente(0)
6: scaleHi $\leftarrow \mathcal{M}$.ajouter_nouveau_méta-signal_de_pente(ν_0)
7: scaleLo $\leftarrow \mathcal{M}$.ajouter_nouveau_méta-signal_de_pente($4\nu_0$)
8: $\mathcal{M}$.ajouter_règle({ scaleLo, borderRi } $\rightarrow$ { back })
9: $\mathcal{M}$.ajouter_règle({ scaleHi, back } $\rightarrow$ { })
{ *ajout de méta-signaux* }
10: **pour tout** μ méta-signal de $\mathcal{M}$ **faire**
11: $\mu' \leftarrow \mathcal{M}$.ajouter_nouveau_méta-signal_pente($\nu_0\,(7\nu - 4\nu_0)\,/\,(\nu + 2\nu_0)$)
12: $\mathcal{M}$.ajouter_règle({ scaleLo, μ } $\rightarrow$ { scaleLo, μ' })
13: $\mathcal{M}$.ajouter_règle({ scaleHi, μ' } $\rightarrow$ { scaleHi, μ })
14: **fin pour**
{ *ajout de règles* }
15: **pour tout** règle $\{\mu_i^-\}_i \rightarrow \{\mu_j^+\}_j$ de $\mathcal{M}$ **faire**
16: $\mathcal{M}$.ajouter_règle($\{\mu_i^{-\prime}\}_i \rightarrow \{\mu_j^{+\prime}\}_j$)
17: $\mathcal{M}$.ajouter_règle($\{\text{scaleLo}\} \cup \{\mu_i^-\}_{\mu_i^-.\nu<\beta} \cup \{\mu_i^{-\prime}\}_{\mu_i^-.\nu>\beta} \rightarrow \{\text{scaleLo}\} \cup \{\mu_j^+\}_{\mu_j^+.\nu>\beta} \cup \{\mu_j^{+\prime}\}_{\mu_j^+.\nu<\beta}$)
18: $\mathcal{M}$.ajouter_règle($\{\text{scaleHi}\} \cup \{\mu_i^{-\prime}\}_{\mu_i^-.\nu<\beta} \cup \{\mu_i^-\}_{\mu_i^-.\nu>\beta} \rightarrow \{\text{scaleHi}\} \cup \{\mu_j^{+\prime}\}_{\mu_j^+.\nu>\beta} \cup \{\mu_j^+\}_{\mu_j^+.\nu<\beta}$)
19: **fin pour**
Sortie :
20: (back, borderRi, scaleLo, scaleHi)

FIG. 7.8 – Algorithme pour créer une contraction continue simple.

La condition sur ν_0 garantit que les signaux ne peuvent repasser ni sous scaleLo ni sous scaleHi. En effet, une étude rapide de la fonction (7.5) indique que si $\nu \in]-\nu_0, \nu_0[$ alors $\nu' \in]-11\nu_0, \nu_0[$. Les vitesses ν et ν' sont donc toujours inférieures à ν_0 (comme à $2\nu_0$). La condition (7.3) est donc toujours vérifiée.

Pour mettre en place l'homothétie, il faut placer des signaux scaleHi puis scaleLo à gauche de la configuration et, borderRi à droite.

7.4.2 Correction

Théorème 41 *Tout calcul s'emboîte dans tout calcul obtenu en lui faisant subir une contraction continue simple.*

►► Preuve.

En toute généralité, prenons $\nu_0 = \frac{1}{2}$ (et donc toutes les valeurs absolues des vitesses sont strictement inférieures à 1) et toutes les coordonnées spatiales du calcul originel incluses

dans $]0,2[$. Les signaux scaleHi et scaleLo sont ajoutés à gauche en position 0 et borderRi à droite en position 2. Le calcul des positions des collisions de la structure donne :

– $(1,1)$ pour la collision entre scaleLo et borderRi, et
– $(1,2)$ pour la collision entre scaleHi et back.

Dans un premier temps, nous ne considérons pas l'action de la seconde homothétie. Ce qu'il y a à (x,t) dans le diagramme originel avec $(x < 2t)$ se retrouve à :

$$\begin{pmatrix} x \\ t \end{pmatrix} = a \begin{pmatrix} 1 \\ 2 \end{pmatrix} + b \begin{pmatrix} 2 \\ 1 \end{pmatrix} \ ,$$
$$a = \frac{-x+2t}{3} \ ,$$
$$b = \frac{2x-t}{3} \ ,$$
$$\begin{pmatrix} x' \\ t' \end{pmatrix} = \frac{1}{2}a \begin{pmatrix} 1 \\ 2 \end{pmatrix} + b \begin{pmatrix} 2 \\ 1 \end{pmatrix} \quad = \quad \begin{pmatrix} \frac{7x-2t}{6} \\ \frac{x+t}{3} \end{pmatrix} \ .$$

Les droites verticales, *i.e.* de vitesses nulles ou $x = cte$, se prolongent par des droites de pente $-\nu_0$. La bande qui contient le diagramme espace-temps se continue donc par une barre oblique de pente $-\nu_0$. Donc back borne bien le calcul en homothétie.

Considérons maintenant la seconde homothétie. Les coordonnées deviennent :

$$\begin{pmatrix} x' \\ t' \end{pmatrix} = a' \begin{pmatrix} 1 \\ 2 \end{pmatrix} + b' \begin{pmatrix} 2 \\ 1 \end{pmatrix} \ ,$$
$$a' = \frac{-x'+2t'}{3} \ ,$$
$$b' = \frac{2x'-t'}{3} \ ,$$
$$\begin{pmatrix} x'' \\ t'' \end{pmatrix} = a' \begin{pmatrix} 1 \\ 2 \end{pmatrix} + \frac{1}{2}b' \begin{pmatrix} 2 \\ 1 \end{pmatrix} \quad = \quad \begin{pmatrix} \frac{7x'-2t'}{6} \\ \frac{x'+2t'}{6} \end{pmatrix} \quad = \quad \frac{1}{2} \begin{pmatrix} x \\ t \end{pmatrix} \ .$$

On a donc bien obtenu une homothétie de rapport $\frac{1}{2}$ et de centre $(0,0)$.

◁◀◀

Pour redresser le diagramme et faire la seconde homothétie, on profite du fait que la direction de l'une est la frontière de l'autre. En deux homothéties, les coordonnées ont été réduites par deux sur chacun des axes. Ces deux directions ne forment pas un repère orthogonal, elles semblent même peu naturelles puisque plus ν_0 est élevée, plus elles forment un angle aiguë.

7.5 Contraction d'une bande à un triangle

La Fig. 7.9 montre un exemple de contraction continue itérée sur la bande. Le calcul de celle-ci se continue à travers les homothéties alors qu'elle est de plus en plus réduite. Il va de

plus en plus vite, le triangle que l'on voit n'est fermé qu'au bout d'une infinité de collisions. Cela engendre un point d'accumulation.

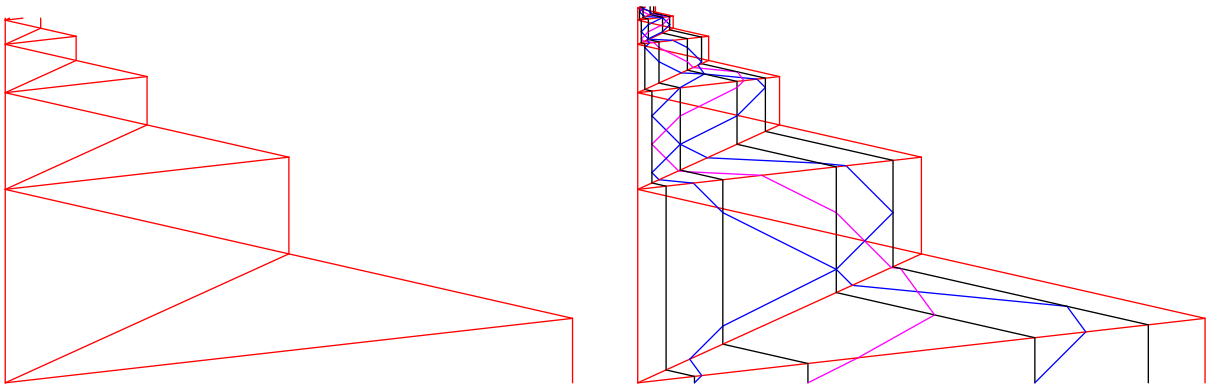
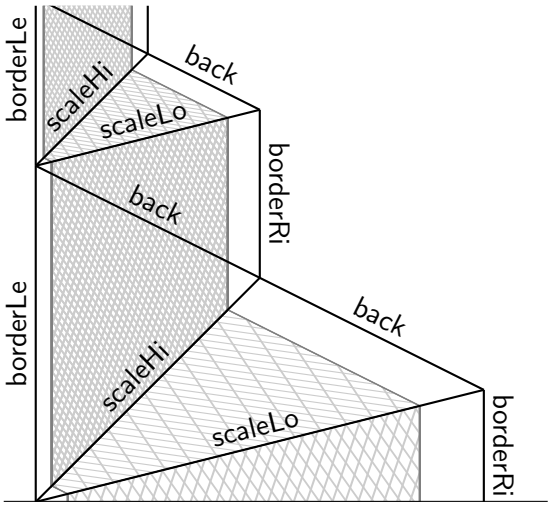


FIG. 7.9 – Contraction continue itérée.

7.5.1 Construction

La Fig. 7.10 montre la structure pour l'itération de la contraction continue simple. En fait, il s'agit de marquer la limite gauche et de laisser continuer back pour qu'il relance la contraction.



Méta-Signal	Vitesse
back	$-\nu_0$
borderRi	0
scaleHi	$\nu_0/2$
scaleLo	$2\nu_0$
borderLe	0

ν_0 : vitesse strictement positive, sa moitié est strictement supérieure à toutes les valeurs absolues des autres, ici 1.

FIG. 7.10 – Structure et méta-signaux de la contraction continue itérée.

Le seul méta-signal ajouté à la contraction simple continue est borderLe qui sert à marquer la limite à gauche. La règle pour la collision de scaleHi et back est changée : non seulement back continue pour relancer la contraction, mais en plus borderRi est engendré pour marquer la nouvelle borne à droite du calcul. Le méta-signal back traverse la bande de calcul sans le modifier. Toutes les règles correspondant à cela sont ajoutées.

À chaque fois, le nouveau borderRi se retrouve positionné à mi-distance entre back et l'ancien ; de sorte que la structure est à chaque fois deux fois plus petite.

La Fig. 7.11 donne l'algorithme pour créer une contraction itérée continue pour un calcul contenu dans une bande.

Entrée :
1: $\mathcal{M}$: machine à signaux
2: ν_0 : rationnel { *vitesse de la contraction* }
Antécédent :
3: L'équation (7.4) est vérifiée
Faire :
4: Créer_contraction_continue($\mathcal{M}$, ν_0)
5: borderLe $\leftarrow$ $\mathcal{M}$.ajouter_nouveau_méta-signal_de_pente(0)
6: $\mathcal{M}$.ajouter_règle({ borderLe, back } $\rightarrow$ { borderLe, scaleHi, scaleLo })
7: $\mathcal{M}$.changer_règle({ scaleHi, back } $\rightarrow$ { back, borderRi })
{ *ajout de méta-signaux* }
8: **pour tout** μ' méta-signal de $\mathcal{M}$ **faire**
9: $\mathcal{M}$.ajouter_règle({ back, μ' } $\rightarrow$ { back, μ' })
10: **fin pour**
{ *ajout de règles* }
11: **pour tout** règle $\{\mu_i^{-\prime}\}_i \rightarrow \{\mu_j^{+\prime}\}_j$ de $\mathcal{M}$ **faire**
12: $\mathcal{M}$.ajouter_règle({ back } $\cup \{\mu_i^{-\prime}\}_i \rightarrow$ { back } $\cup \{\mu_j^{+\prime}\}_j$)
13: **fin pour**
Sortie :
14: (back, borderLe borderRi, scaleLo, scaleHi)

FIG. 7.11 – Algorithme pour créer un contraction itérée continue sur une bande.

7.5.2 Correction

Théorème 42 *Tout calcul contenu dans une bande s'emboîte dans tout calcul obtenu en lui faisant subir une contraction continue itérée.*

►► Preuve.

Par une récurrence immédiate, la construction se répète, tout recolle bien ; la bande reste bien à l'intérieur de la construction.

Nous nous plaçons dans le cas général suivant : $\nu_0 = 1$ et, borderLe et borderRi aux positions initiales respectives 0 et 2.

Il faut vérifier si tous les temps sont bien présents, *i.e.* la structure n'atteint pas son point d'accumulation avant que toute la bande ne soit emboîtée. Pour cela, mesurons l'écoulement du temps originel sur l'axe borderLe[2]. Un rapide calcul montre que la position temporelle de

[2]Ou légèrement à sa droite pour être dans la configuration, mais le temps originel est continu sur le diagramme espace-temps contracté. La seul différence serait l'ajout d'un terme $\frac{1}{2^n}(\epsilon, 2\epsilon)$ où n est le numéro de l'itération. Ce terme ne ferait qu'alourdir la preuve.

la n^{e} collision de borderLe et back est donnée par

$$\sum_{k=1}^{k=n} \frac{3}{2^k} \;=\; 3\left(1 - \frac{1}{2^n}\right)$$

(la position spatiale étant toujours 0). Le temps originel correspondant à ce point (il est le même pour les quatre régions incidentes) est $3n$.

La suite des positions converge, mais la suite $(3n)_{n\in\mathbb{N}}$ tend vers l'infini, tous les temps originels auront donc lieu avant le point d'accumulation.

Le fait de changer d'échelle de temps ou d'écartements ne fait que changer les différentes constantes des suites, mais le temps originel tend toujours vers l'infini quand on se rapproche du point d'accumulation.

◁◀◀

Corollaire 43 *Tout calcul sur une bande s'emboîte dans un calcul tenant dans une portion finie de l'espace-temps.*

7.6 Contraction de tout l'espace à un triangle

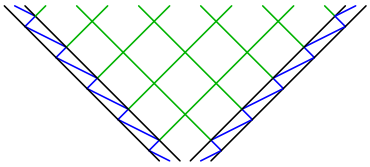
En utilisant la dernière construction du chapitre précédent, on peut contenir tout calcul dans une bande (Cor. 38). Maintenant, il est possible de le réduire encore à un triangle en application du Cor. 43 :

Théorème 44 *Tout calcul est emboîté dans un calcul contenu dans une partie finie de l'espace-temps. La construction du calcul englobant est effective.*

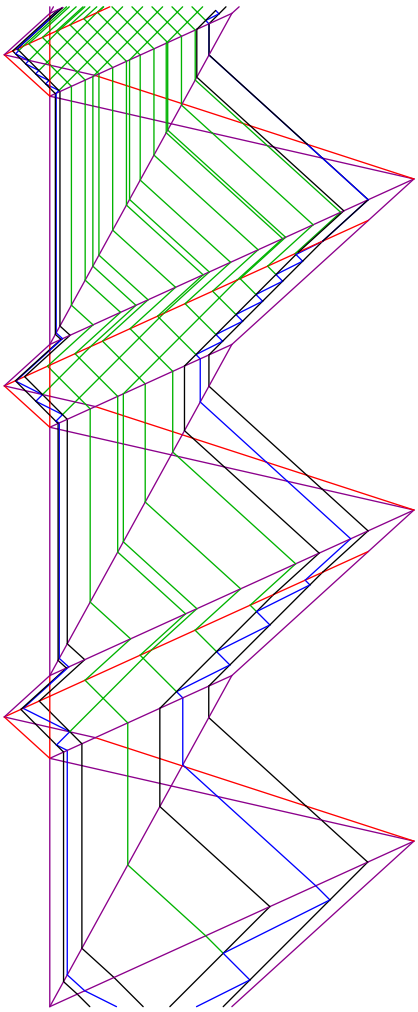
Nous ne décrivons pas la construction ni l'algorithme, car cela correspond à appliquer séquentiellement l'algorithme de la Fig. 6.12 puis celui de la Fig. 7.11.

La Fig. 7.12 montre un exemple de cette construction. En haut, le diagramme s'étend des deux côtés, en bas à gauche, il est restreint à une bande et, en bas à droite, il tient dans un triangle fini.

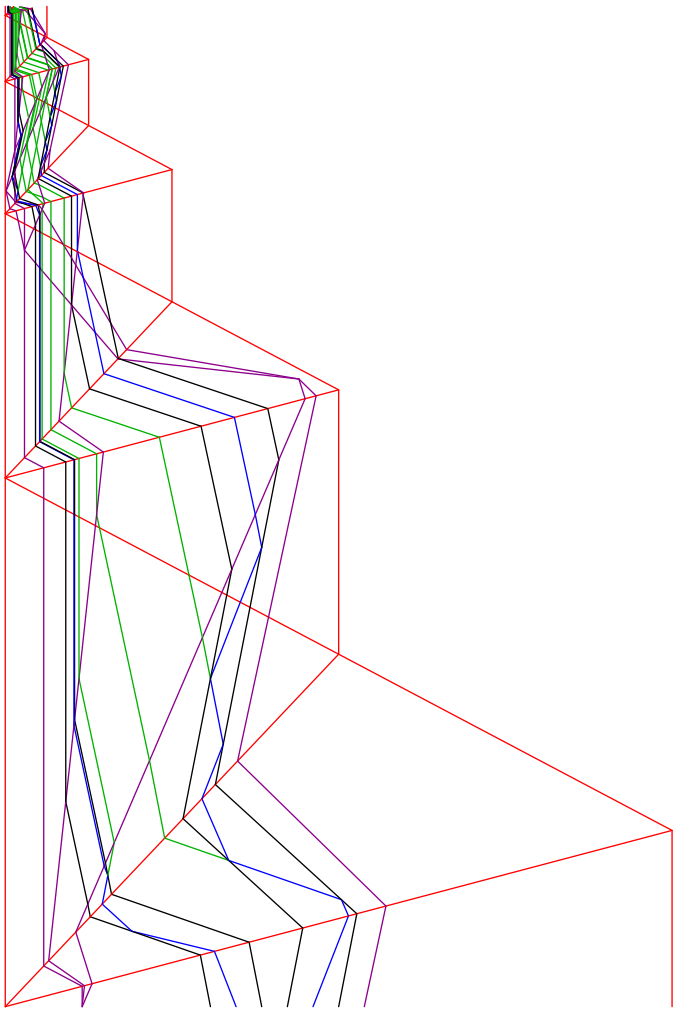
Le diagramme espace-temps de droite de la Fig. 7.12 tient dans un triangle. Que se passe-t-il dans le sommet du haut ? Il y a un point d'accumulation... ce à quoi est consacré la seconde partie de ce mémoire.



(a) calcul



(b) contraction à une bande



(c) contraction à un triangle

FIG. 7.12 – Contraction d'un calcul à un triangle.

Deuxième partie

Accumulations (de collisions et de signaux)

Chapitre 8

Reformulation topologique

L'accumulation est une notion topologique. Pour bien l'appréhender, nous reformulons nos diagrammes espace-temps de manière topologique ; la définition des points d'accumulation et leurs traitements dans les chapitres suivants sont alors aisés.

Les machines à signaux ne savent pas gérer les accumulations. Elles ne définissent pas la valeur d'un diagramme espace-temps à une position d'accumulation (ni ce qui pourrait se trouver dans son cône d'influences émises). Nous appelons une telle position une *singularité*.

Pour nous, une singularité est toute position où le diagramme n'est pas défini par le fonctionnement de la machine. Les valeurs du diagramme en de telles positions ne sont pas encore définies (le chapitre suivant en définit quelques unes). Nous supposons uniquement qu'elles appartiennent à un ensemble fini $\mathcal{S}$ et que cet ensemble contient un élément $\Diamond$ (non défini et sans contrainte).

Le but de ce chapitre est de formuler une définition topologique des diagrammes qui correspondent aux traces de la première partie (Chap. 3) pour toutes les positions non singulières et, de prouver cette concordance. Dans tout ce chapitre, on peut considérer que $\mathcal{S}$ est réduit à $\Diamond$.

8.1 Définition topologique

Nous notons $\mathbb{D}_{\mathcal{M}}$ les diagrammes espace-temps pour une machine $\mathcal{M}$ tels que définis dans le Chap. 3 et $\mathbb{D}_{\mathcal{T}}$ les diagrammes tels que nous les définissons ici. Une fois leur équivalence montrée, ou si le contexte est clair, nous les notons $\mathbb{D}$.

8.1.1 Topologie sur l'espace-temps

L'espace-temps est $\mathbb{R}^2_{0\leq t}$ (les temps commencent à 0), les éléments de cet ensemble sont appelés des *positions* (conformément aux définitions de la Sect. 3.2). Nous munissons cet ensemble de la distance euclidienne :

$$\forall (x,t),(x',t') \in \mathbb{R}^2_{0\leq t},\ d((x,t),(x',t')) = \sqrt{(x-x')^2+(t-t')^2}$$

et de la topologie associée (*i.e.* la trace de la topologie usuelle de $\mathbb{R}^2$ sur $\mathbb{R}^2_{0\leq t}$). Nous notons cet espace topologique $(\mathbb{R}^2_{0\leq t}, \mathcal{O}_{0\leq t})$ ($\mathcal{O}_{0\leq t}$ étant l'ensemble des ouverts)[1].

Soit $\mathcal{S}$ un ensemble fini quelconque, mais possédant un élément $\Diamond$, représentant les valeurs possibles d'un diagramme aux positions de singularité. Nous nommons ces valeurs des *méta-singularités*. Nous en définissons dans le chapitre suivant pour appréhender leur nature et leur signification. Ces définitions sont alors ajoutées à celle de la machine.

Nous sommes maintenant en mesure de définir topologiquement les diagrammes.

Définition 45 Pour la machine $\mathcal{M} = (\{\mu_i\}_i, \{\rho_k\}_k)$, un *diagramme espace-temps (topologique)*, $\mathbb{D}_\mathcal{T}$, est une fonction de $\mathbb{R}^2_{0\leq t}$ dans $\{\oslash\} \cup \{\mu_i\}_i \cup \{\rho_k\}_k \cup \mathcal{S}$ telle que, pour toute position (x_0, t_0) :

1. si $\mathbb{D}_\mathcal{T}(x_0, t_0) = \oslash$ alors il existe un ouvert o de $\mathcal{O}_{0\leq t}$ contenant (x_0, t_0) tel que :
   - $\mathbb{D}_\mathcal{T}(o) = \{\oslash\}$;
2. si $\mathbb{D}_\mathcal{T}(x_0, t_0) = \mu$ alors il existe un ouvert o de $\mathcal{O}_{0\leq t}$ contenant (x_0, t_0) tel que :
   - $\mathbb{D}_\mathcal{T}(o) = \{\oslash, \mu\}$,
   - $\forall (x, t) \in o$,
     - $\mathbb{D}_\mathcal{T}(x, t) = \mu$ est équivalent à $x - x_0 = \nu_\mu(t - t_0)$;
3. si $\mathbb{D}_\mathcal{T}(x_0, t_0) = \rho = \{\mu_i^-\}_i \rightarrow \{\mu_j^+\}_j$ alors il existe un ouvert o de $\mathcal{O}_{0\leq t}$ contenant (x_0, t_0) tel que :
   - $\mathbb{D}_\mathcal{T}(o) = \{\oslash, \rho\} \cup \{\mu_i^-\}_i \cup \{\mu_j^+\}_j$,
   - $\forall (x, t) \in o$,
     - $\mathbb{D}_\mathcal{T}(x, t) = \rho$ est équivalent à $(x, t) = (x_0, t_0)$,
     - $\mathbb{D}_\mathcal{T}(x, t) = \mu_i^-$ est équivalent à ($t < t_0$ et $x - x_0 = \nu_{\mu_i^-}(t - t_0)$),
     - $\mathbb{D}_\mathcal{T}(x, t) = \mu_j^+$ est équivalent à ($t_0 < t$ et $x - x_0 = \nu_{\mu_j^+}(t - t_0)$).
4. si $\mathbb{D}_\mathcal{T}(x_0, t_0) \in \mathcal{S}$, la position (x_0, t_0) est dite *singulière* ou *de singularité* ou une *singularité*[2].

Les valeurs atteintes dans ces ouverts sont aussi l'ensemble minimal (pour l'inclusion) des ensembles des valeurs atteintes sur tout ouvert contenant la position[3]. Nous utilisons cette approche dans le chapitre suivant.

Remarque 46 Une valeur non singulière fixe le diagramme sur tout un ouvert contenant la position. Les conditions restent valables pour tout ouvert plus petit, tant qu'il contient la position.

Le temps n'est plus qu'une dimension ; les notions de postérieur ou d'antérieur ne sont plus que le reflet de l'ordre sur cet axe.

[1] Nous aurions pu prendre une topologie exotique, par exemple en construisant les ouverts à partir des cônes d'influences reçues / émises ou, de la relation d'influence. Mais comme on le voit dans le chapitre suivant, les cônes ne signifient plus grand-chose en cas d'accumulation.

[2] Il n'y a pas de conditions pour l'instant ; dans le chapitre suivant, nous définirons quelques singularités et les conditions correspondantes. La valeur $\Diamond$ ne doit jamais avoir de condition associée ; elle sert à compléter les diagrammes correspondant à des traces.

[3] La finitude de $\mathcal{S}$ implique que tout point d'accumulation des singularités est aussi un point d'accumulation d'une singularité.

Remarque 47 À l'exclusion des singularités, les valeurs atteintes pour les positions antérieures à l'intérieur de l'ouvert fixent la valeur. En effet, s'il n'y a que $\oslash$ alors c'est $\oslash$, s'il y a $\oslash$ et un seul μ, alors c'est μ, s'il y a plusieurs μ alors c'est la règle correspondante (elle est unique puisque la machine est déterministe).

Comme il n'y a aucune contrainte sur les singularités, il est possible de transformer autant de positions[4] que l'on souhaite en singularités sans changer la validité du diagramme.

Notre définition topologique est très proche de celle donnée par Jacopini et Sontacchi dans [JS90]. Les principales différences sont :

- notre travail est limité à la dimension deux[5] ;
- ils considèrent aussi des surfaces et des (hyper)-volumes ;
- leurs polyèdres occupent une partie bornée de l'espace, alors que nous considérons aussi les diagrammes infinis ;
- ils excluent explicitement les singularités ; nous tentons de les traiter.

8.1.2 Définition des signaux

Un sous-ensemble E de $\mathbb{R}^2_{0\leq t}$ est dit *connexe par arcs* si pour tout couple de positions de E, il existe une fonction continue de $[0,1]$ dans E les ayant pour images en 0 et 1. Rappelons qu'un sous-ensemble est *connexe*, s'il n'est pas partitionable en deux parties non vides incluses dans des ouverts disjoints. La connexité par arcs implique la connexité alors que la réciproque est fausse.

Définition 48 Nous appelons *signal* de méta-signal μ d'un diagramme $\mathbb{D}_\mathcal{T}$ toute partie non vide connexe par arcs maximale (pour l'inclusion) de ${\mathbb{D}_\mathcal{T}}^{-1}(\mu)$.

Montrons que cela correspond aux traces de signaux de la première partie.

Lemme 49 *Pour tout méta-signal μ, si (x_0,t_0) est un point d'accumulation de ${\mathbb{D}_\mathcal{T}}^{-1}(\mu)$ alors $\mathbb{D}_\mathcal{T}(x_0,t_0)$ appartient à $\{\mu\} \cup \{\, \rho \,|\, \rho = \{\mu_i^-\}_i \rightarrow \{\mu_j^+\}_j \wedge \mu \in \{\mu_i^-\}_i \cup \{\mu_j^+\}_j) \,\} \cup \mathcal{S}$.*

►► Preuve.

Ce sont les seules valeurs pour lesquelles il est possible d'avoir μ dans l'image de tout ouvert (Déf. 45).

◁◀◀

[4] Il faut toutefois que toute position d'accumulation de positions singulières soit singulière. La Définition 45 le laisse deviner et le Prop. 70 le justifie.

[5] Notre définition peut s'étendre à toute dimension en remplaçant chaque égalité par la colinéarité avec la vitesse correspondante. Tous les résultats de cette partie s'étendent à toute dimension finie, mais cela n'apporte rien à notre propos.

Lemme 50 *Tout signal σ de méta-signal μ, est un ensemble de points de l'une des formes suivantes :*

- $\{\ (x_0 + \nu_\mu(t - t_0),\ t) \mid 0 \leq t < b\ \}$, *et*
- $\{\ (x_0 + \nu_\mu(t - t_0),\ t) \mid a < t < b\ \}$, *avec* $a \in [0, b[$ [6],

où $(x_0, t_0) \in \sigma$ *et* $b \in]0, +\infty[\cup \{+\infty\}$.

►► Preuve.

La preuve est en deux temps, nous montrons que tous les points sont de la forme $(x_0 + \nu_\mu(t-t_0), t)$ puis nous considérons les extrémités.

Soient (x_0, t_0) et (x, t) deux positions appartenant à σ. Comme σ est connexe par arcs, il existe une fonction f telle que :

$$\begin{array}{rcll} f : [0,1] & \longrightarrow & \sigma \subseteq \mathbb{R}^2_{0\leq t} & \text{continue ,} \\ 0 & \longmapsto & (x_0, t_0) & , \\ \forall \lambda \in [0,1],\ \lambda & \longmapsto & (x_\lambda, t_\lambda) & , \\ 1 & \longmapsto & (x, t) & . \end{array}$$

Définissons une fonction $g : [0,1] \longrightarrow \mathbb{R}$ par $g(\lambda) = x_0 - x_\lambda + \nu_\mu(t_\lambda - t_0)$. Elle est continue et $g(0) = 0$. Montrons qu'elle est nulle partout par l'absurde. Soit λ_0 dans $[0,1]$ la borne inférieure des points où elle ne vaut pas 0. La valeur de λ_0 ne peut pas être 1, sinon g serait déjà nulle partout. Si $\lambda_0 = 0$, $g(\lambda_0) = 0$, sinon, par continuité à gauche, $g(\lambda_0) = 0$.

Puisque $f(\lambda_0) \in \sigma$, $\mathbb{D}_\mathcal{T}(f(\lambda_0)) = \mu$. Soit o un ouvert de $\mathbb{R}^2_{0\leq t}$ correspondant à la Cond. 2 de la Déf. 45. La fonction f étant continue, il existe un ouvert o' de $[0,1]$ contenant λ_0 tel que $f(o') \subseteq o$. Comme λ_0 est strictement inférieur à 1, il existe $0 < \varepsilon$ tel que $[\lambda_0, \lambda_0 + \varepsilon] \subseteq o'$. Pour tout $\lambda \in [\lambda_0, \lambda_0 + \varepsilon]$, la Cond. 2 de la Déf. 45 implique que $g(\lambda_0) - g(\lambda) = x_\lambda - x_{\lambda_0} + \nu_\mu(t_{\lambda_0} - t_\lambda) = 0$. Donc $g(\lambda_0) = g(\lambda) = 0$, et $g([\lambda_0, \lambda_0 + \varepsilon]) = \{0\}$ avec $0 < \varepsilon$; λ_0 n'est donc pas la borne inférieure. La fonction g est donc nulle sur tout l'intervalle $[0,1]$, donc $g(1)$ vaut 0 et (x, t) est bien de la forme souhaitée.

Les positions sont donc bien toutes de la forme souhaitée, la connexité impose que les t forment un intervalle. Il reste à considérer les bornes de ces intervalles. Pour chaque borne différente de 0 et de $+\infty$, le Lem. 49 indique que cette extrémité a pour valeur soit μ, soit ρ, soit un élément de $\mathcal{S}$. Dans le premier cas, de nouveau grâce à la Cond. 2 de la Déf. 45, le signal s'étendrait après l'extrémité et ne serait pas maximal. Dans les deux derniers cas, la position ne peut appartenir au signal, donc l'intervalle est ouvert de ce côté. L'intervalle a donc bien la forme souhaitée.

◁◄◄

Avec ces lemmes nous pouvons affirmer :

Théorème 51 *Les signaux définis par la Déf. 48 correspondent aux traces de la Déf. 2 du Chap. 3.*

[6] La possibilité d'avoir $a = 0$ correspond à accepter des collisions à $t = 0$. Nous aurions pu le refuser, mais cela n'aurait fait que compliquer les choses car il faut alors le faire apparaître partout. Par contre, $a = +\infty$ et b=0 n'ont pas de sens.

En effet, chaque signal (au sens la Déf. 48) correspond à la trace d'un signal de méta-signal μ et de vitesse ν_μ de son origine ($t = 0$ ou collision) à sa fin (collision ou sans fin). Dans le contexte topologique, son origine, comme sa fin, peut être une singularité.

8.1.3 Définition des collisions

Définition 52 Une *collision* correspondant à la règle $\rho = \{\mu_i^-\}_i \to \{\mu_j^+\}_j$ est une position p telle que $\mathbb{D}_{\mathcal{T}}(p) = \rho$.

Théorème 53 *La définition des collisions correspond à la Déf. 4 du Chap. 3.*

►► Preuve.
La Cond. 3 de la Déf. 45 impose que les seuls signaux présents soient exactement $\{\mu_i^-\}_i$ et $\{\mu_j^+\}_j$ et, de plus, que les $\{\mu_i^-\}_i$ se terminent à cette position et que les $\{\mu_j^+\}_j$ y commencent. Nous retrouvons l'arrivée de signaux, leur disparition et l'apparition des signaux engendrés. Comme les signaux correspondent, les collisions aussi.

◁◀◀

8.2 Équivalence des définitions de diagramme

Lemme 54 *Tout diagramme tel que défini dans la première partie vérifie la Déf. 45 en complétant sa définition par $\Diamond$ là où il n'est pas défini.*

►► Preuve.
Les seules valeurs, avant complétion, sont $\oslash$, μ ou ρ. On vérifie ces différents cas sans problème.

◁◀◀

Soient $\nu_{min}^{\mathcal{M}}$ et $\nu_{max}^{\mathcal{M}}$ les plus petite et plus grande vitesses présentes dans une machine $\mathcal{M}$.

Définition 55 (Complément de la Sect. 3.5) Nous appelons *cône épointé (d'influences reçues)* en (x_0, t_0) et, notons $\mathcal{C}_-^*(x_0, t_0)$ (ou simplement $\mathcal{C}_-^*$) l'ensemble des positions suivant :

$$\mathcal{C}_-^*(x_0, t_0) \;=\; \left\{ (x,t) \in \mathbb{R}^2_{0\le t} \;\middle|\; t < t_0 \;\wedge\; \nu_{min}^{\mathcal{M}} \le \frac{x_0 - x}{t_0 - t} \le \nu_{max}^{\mathcal{M}} \right\} .$$

Le cône est épointé ; (x_0, t_0) n'y appartient pas. C'est la seule utilité de la première inégalité. Le terme « d'influences reçues » est souvent sous-entendu car ce sont les seuls cônes que l'on épointe.

Remarque 56 Nous pouvons étendre la Rem. 47 en ne considérant plus les positions antérieures mais les positions dans le cône car les égalités imposent aux positions non $\oslash$ d'être dans le cône (éventuellement sur les frontières).

Nous complétons la Prop. 13 sur les influences réciproques :

Propriété 57 *Pour tout couple de positions p et q*

$$\mathcal{C}^*_-(p) \subsetneq \mathcal{C}^*_-(q) \quad \Leftrightarrow \quad p \in \mathcal{C}^*_-(q) \quad \Leftrightarrow \quad \left\{ \begin{array}{l} q \in \mathcal{C}_+(p) \\ p \neq q \end{array} \right\} \quad \Leftrightarrow \quad \mathcal{C}_+(q) \subsetneq \mathcal{C}_+(p) \ .$$

Nous caractérisons les diagrammes sans singularités « inutiles » et les parties maximales sans singularités dans leurs cônes d'influences reçues.

Définition 58 Un diagramme est *sans singularités inutiles* si, et seulement si, il n'existe pas de diagramme ayant la même définition sur les positions non singulières et ayant strictement moins de singularités, *i.e.* :

$$\mathbb{D} \textit{ sans singularités inutiles } \Leftrightarrow \forall \mathbb{D}', \left(\mathbb{D}|_{\mathbb{R}^2_{0\leq t} \setminus \mathbb{D}^{-1}(\mathcal{S})} = \mathbb{D}'|_{\mathbb{R}^2_{0\leq t} \setminus \mathbb{D}^{-1}(\mathcal{S})} \Rightarrow \mathbb{D}^{-1}(\mathcal{S}) = \mathbb{D}'^{-1}(\mathcal{S}) \right) \ .$$

La partie *déterministe* d'un diagramme, $det(\mathbb{D})$, est l'union de tous ses cônes d'influences reçues sans singularités.

$$det(\mathbb{D}) \ = \bigcup_{\mathbb{D}(\mathcal{C}_-) \cap \mathcal{S} = \emptyset} \mathcal{C}_-$$

Un diagramme est *à partie déterministe maximale* si, et seulement si, il n'existe pas de diagramme ayant la même définition sur la partie déterministe du premier et ayant une partie déterministe strictement plus grande, *i.e.* :

$$\mathbb{D} \textit{ à partie déterministe maximale } \Longleftrightarrow \forall \mathbb{D}', \left(\mathbb{D}|_{det(\mathbb{D})} = \mathbb{D}'|_{det(\mathbb{D})} \Rightarrow det(\mathbb{D}') \subseteq det(\mathbb{D}) \right) \ .$$

Remarque 59 Attention, un diagramme sans singularités inutiles n'est pas forcément à partie déterministe maximale, l'inverse est lui aussi faux ! La première notion correspond à ne pas avoir de singularités en des points où l'on peut s'en passer alors que la seconde caractérise la partie correspondant à la machine ! Le terme déterministe se comprend aussi par la possibilité d'avoir plusieurs continuations différentes après une singularité comme nous le montrons dans le chapitre suivant.

Définition 60 Nous appelons *triangle de base* (x_l, x_r, t_d), et notons $\mathcal{T}(x_l, x_r, t_d)$ (ou simplement $\mathcal{T}$), l'ensemble des points suivants :

$$(x,t) \in \mathcal{T}(x_l, x_r, t_d) \iff \left\{ \begin{array}{rcl} t_d & \leq & t \\ \nu^{\mathcal{M}}_{max}(t - t_d) & \leq & (x - x_l) \\ (x - x_r) & \leq & \nu^{\mathcal{M}}_{min}(t - t_d) \end{array} \right. \ .$$

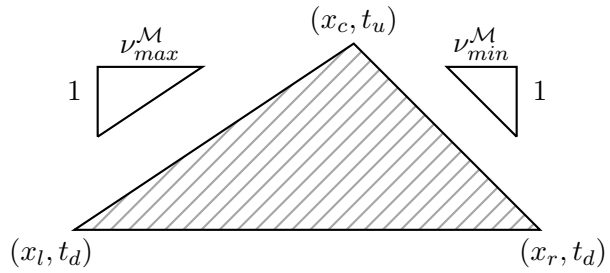


FIG. 8.1 – Triangle de base (x_l, x_r, t_d).

Le triangle n'est pas épointé. Il correspond à la Fig. 8.1 où (x_c, t_u) se calculent simplement comme l'intersection de deux droites. Un triangle correspond à un cône d'influences reçues restreint à une valeur minimale de t. Cet ensemble est fermé et compact.

Pour toute position d'un triangle $\mathcal{T}(x_l, x_r, t_d)$, le cône d'influences reçues y aboutissant est inclus dans le triangle pour toutes les dates postérieures ou égales à t_d.

Lemme 61 *Si deux diagrammes sont identiques sur la base d'un triangle et n'ont pas de singularités dans ce triangle alors, ils sont égaux sur tout celui-ci.*

►► Preuve.

Par l'absurde, supposons qu'il existe deux diagrammes, $\mathbb{D}_{\mathcal{T}}$ et $\mathbb{D}'_{\mathcal{T}}$, et un triangle $\mathcal{T}(x_l, x_r, t_d)$ sans singularités sur les deux diagrammes, tels qu'il y a égalité sur la base du triangle mais pas sur le triangle entier. Notons $\mathcal{T}^{\neq}$ la partie du triangle où les diagrammes diffèrent. Soit t_0 la borne inférieure des instants de $\mathcal{T}^{\neq}$. Nous montrons qu'il y a égalité à t_0 et même après.

Si $t_0 = t_d$ alors, il y a égalité sur t_0 puisque c'est la base du triangle. Considérons le cas $t_d < t_0$. La Rem. 56 montre que pour toute position (x, t_0), la valeur (hormis les singularités) est définie par ce qui se trouve dans l'intersection d'un ouvert contenant le point et du cône épointé. Comme il y a égalité pour $t < t_0$ sur $\mathcal{T}$ et sur les intersections des cônes avec des ouverts suffisamment petits pour être dans le triangle, il y a forcément égalité en (x, t_0).

Pour toute position (x, t_0) du triangle, il y a un ouvert $o_{(x,t_0)}$ ($o'_{(x,t_0)}$) pour $\mathbb{D}_{\mathcal{T}}$ ($\mathbb{D}'_{\mathcal{T}}$) correspondant à la Déf. 45. Les deux diagrammes sont donc égaux sur l'ouvert $o''_{(x,t_0)} = o_{(x,t_0)} \cap o'_{(x,t_0)}$. Soit $o'''_{(x,t_0)}$ un ouvert inclus dans $o''_{(x,t_0)}$ de la forme $]x-\varepsilon_{(x,t_0)}, x+\varepsilon_{(x,t_0)}[\times]t_0-\varepsilon_{(x,t_0)}, t_0+\varepsilon_{(x,t_0)}[$ pour $\varepsilon_{(x,t_0)}$ strictement positif et suffisamment petit (la topologie est engendrée par les produits d'ouverts de $\mathbb{R}$ et $\mathbb{R}_{0\leq}$).

L'ensemble $\mathcal{T}$ est compact, sa restriction à $t = t_0$ est fermée, donc compacte elle aussi. Elle est recouverte par les ouverts $o'''_{(x,t_0)}$ pour $(x, t_0) \in \mathcal{T}$, donc par une sous-famille finie. Soit ε_0 le plus petit des $\varepsilon_{(x,t_0)}$ de cette sous-famille finie, alors les deux diagrammes sont aussi égaux sur le triangle jusqu'à $t_0 + \varepsilon_0$.

La valeur t_0 ne peut donc être une borne inférieure des instants de $\mathcal{T}^{\neq}$. Il y a donc égalité sur tout le triangle.

◁◀◀

Les diagrammes sont également égaux sur un ouvert contenant le triangle grâce à l'égalité sur les frontières. Par une preuve similaire à celle du Lem. 61, en utilisant la compacité de la frontière, on obtient :

Lemme 62 *Tout cône d'influences reçues sans singularités est inclus dans l'intérieur d'un autre cône d'influences reçues sans singularités.*

La pointe du cône ne pouvant être dans l'intérieur, en épointant le cône on obtient :

Corollaire 63 *Tout cône d'influences reçues sans singularités est inclus dans un cône épointé sans singularités.*

Définition 64 Un diagramme (au sens topologique) a une configuration (au sens de la Sect. 3.5) c_0 comme *configuration initiale* s'il vérifie :

$$\forall x \in \mathbb{R}, \begin{cases} \mathbb{D}_{\mathcal{T}}(x,0) \in \{\oslash\} \cup \{\mu_i\}_i \cup \{\rho_k\}_k \ , \\ \mathbb{D}_{\mathcal{T}}(x,0) = \mu \iff (\mu,(x,0)) \in c_0 \ , \\ \mathbb{D}_{\mathcal{T}}(x,0) = \rho \iff (\rho,(x,0)) \in c_0 \ . \end{cases}$$

Sachant qu'un cône est un triangle épointé ayant sa base à $t = 0$ et que les diagrammes de la première partie le sont aussi par rapport à la définition topologique (Lem. 54) :

Théorème 65 *Pour une machine M et une configuration initiale c_0, tout diagramme correspondant à la Sect. 3.5 est identique, sur tout cône d'influences reçues sans singularités, au diagramme suivant la Déf. 45 ayant pour configuration initiale c_0.*

La restriction aux cônes est très importante car d'une part, transformer un nombre quelconque[7] de positions en singularités ne change pas la validité du diagramme, mais crée des positions où les « contraintes ne suivent plus ». D'autre part, cela est très important car le modèle de base n'incluant pas les singularités, même si elles apparaissent dans les deux diagrammes, ce qui se trouve dans leurs cônes d'influences émises n'est pas défini (même si nous tentons de le faire partiellement dans le chapitre suivant).

Corollaire 66 *Tous les diagrammes à partie déterministe maximale ayant la même configuration initiale correspondent sur la partie déterministe. Cette partie correspond à l'exécution de la machine selon la première partie. Elle est unique et déterministe.*

Cherchons à caractériser plus la partie déterministe.

Définition 67 Nous définissons les ensembles suivants :

$$det^*(\mathbb{D}) \ = \ \bigcup_{\mathbb{D}(\mathcal{C}^*_-)\cap\mathcal{S}=\emptyset} \mathcal{C}^*_- \qquad \text{et} \qquad co{-}det(\mathbb{D}) \ = \ \bigcup_{\mathbb{D}(p)\in\mathcal{S}} \mathcal{C}_+(p) \ .$$

Théorème 68 *Ces ensembles vérifient*

$$det(\mathbb{D}) \ = \ det^*(\mathbb{D}) \ = \ \mathbb{R}^2_{0\leq t} \setminus co{-}det(\mathbb{D}) \ .$$

[7] Même remarque que la Note 4 de la page page 95.

►► **Preuve.**

Montrons que toute position, p, appartient soit à $co{-}det(\mathbb{D})$, soit aux deux autres ensembles.

Considérons le cas où p est une singularité. Comme $p \in \mathcal{C}_+(p)$ et $\mathbb{D}(p) \in \mathcal{S}$ alors $p \in co{-}det(\mathbb{D})$. Si p appartient à $det(\mathbb{D})$ ($det^*(\mathbb{D})$), alors il appartient à un $\mathcal{C}_-$ ($\mathcal{C}^*_-$) sans singularités ce qui n'est pas possible puisque p en est justement une.

Considérons l'autre cas : p n'est pas une singularité.

Si p appartient à $co{-}det(\mathbb{D})$, alors il existe q telle que q est une singularité et p appartient à $\mathcal{C}_+(q)$. Donc q appartient à $\mathcal{C}^*_-(p)$ (Prop. 57 puisque p et q ne peuvent être égales). Tout cône $\mathcal{C}_-$ ($\mathcal{C}^*_-$) contenant p inclut donc $\mathcal{C}^*_-(p)$ et donc contient la singularité q. Aucun des cônes définissant $det(\mathbb{D})$ ($det^*(\mathbb{D})$) ne peut donc contenir p.

Si p n'appartient pas à $co{-}det(\mathbb{D})$, alors aucune singularité ne peut appartenir à $\mathcal{C}_-(p)$; ce cône est donc inclus dans $det(\mathbb{D})$ et p y appartient donc. Grâce au Cor. 63, p est aussi dans un cône épointé sans singularités et, appartient donc à $det^*(\mathbb{D})$.

◅◄◄

8.3 Propriétés topologiques

Propriété 69 *Pour tout ensemble de méta-signaux M et de règles R, les ensembles*

- $\mathbb{D}^{-1}(\{\oslash\})$,
- $\mathbb{D}^{-1}(\{\oslash\} \cup M)$ *et,*
- $\mathbb{D}^{-1}\left(\{\oslash\} \cup R \cup \bigcup_{\{\mu_i^-\}_i \to \{\mu_j^+\}_j \in R} (\{\mu_i^-\}_i \cup \{\mu_j^+\}_j)\right)$

sont ouverts.

►► **Preuve.**

Pour toute (x,t) telle que $\mathbb{D}(x,t) = \oslash$, il y a un ouvert $o_{(x,t)}$ tel que $\mathbb{D}(o_{(x,t)}) = \{\oslash\}$, donc :

$$\mathbb{D}^{-1}(\{\oslash\}) = \bigcup_{\mathbb{D}(x,t)=\oslash} o_{(x,t)},$$

est donc ouvert comme union (quelconque) d'ouverts.

Pour tout méta-signal μ de M et pour toute (x,t) telle que $\mathbb{D}(x,t) = \mu$, alors il existe un ouvert o contenant (x,t) tel que $\mathbb{D}(o) = \{\oslash, \mu\}$, donc $o \subseteq \mathbb{D}^{-1}(\{\oslash\} \cup M)$. Le cas $\mathbb{D}(x,t) = \oslash$ est identique.

Pour le troisième cas, il faut ajouter les points correspondant aux règles, ils sont eux aussi contenus dans des ouverts inclus dans la pré-image.

◅◄◄

Par complémentarité :

Propriété 70 *L'ensemble $\mathbb{D}^{-1}(\mathcal{S})$ est fermé.*

De la Cond. 3 de la Déf. 45 découle

Propriété 71 *L'ensemble* $\mathbb{D}^{-1}(\{\rho_k\}_k)$ *est formé de points isolés.*

Si l'on change l'espace topologique pour supprimer les positions sans information (*i.e.* $\oslash$), on obtient :

Propriété 72 *Sur la topologie trace de* $\mathbb{R}^2_{0\leq t}$ *sur* $\mathbb{R}^2_{0\leq t} \setminus {\mathbb{D}_{\mathcal{T}}}^{-1}(\{\oslash\})$*, les signaux sont des ouverts.*

►► Preuve.
Soit σ un signal de méta-signal μ, pour chacune de ses positions (x,t), il existe un ouvert $o_{(x,t)}$ correspondant à la Cond. 2 de la Déf. 45. Toutes les positions de l'ouvert ayant μ sont dans le signal. Soit E l'union de ces ouverts ; E est donc ouvert et sa trace sur $\mathbb{R}^2_{0\leq t} \setminus {\mathbb{D}_{\mathcal{T}}}^{-1}(\{\oslash\})$ l'est donc aussi. Sa trace est le signal puisque seules les valeurs μ et $\oslash$ sont atteintes et tout le signal est ouvert.

◁◀◀

Propriété 73 *L'ensemble* $det(\mathbb{D})$ *(*$co{-}det(\mathbb{D})$*) est ouvert (fermé).*

►► Preuve.
Du Lem. 62, il découle que $det(\mathbb{D})$ est aussi l'union de tous les intérieurs des cônes sans singularités et est donc ouvert. La fermeture de $co{-}det(\mathbb{D})$ découle de la complémentarité des deux ensembles (Th. 68).

◁◀◀

Nous avons donc mis en place notre reformulation topologique et montré qu'elle correspondait en dehors des singularités. Nous avons tous les éléments nécessaires pour nous pencher sur les singularités.

Chapitre 9

Singularités isolées : taxonomie et non déterminisme

La première partie se termine par un diagramme espace-temps (Fig. 7.12(c)) où il y a une infinité de collisions et de signaux dans une portion finie du diagramme espace-temps. Dans le chapitre précédent, nous avons établi un cadre topologique pour les diagrammes espace-temps.

Dans ce chapitre, nous considérons le cas des singularités isolées, *i.e.* il existe un ouvert la contenant et ne contenant aucune autre singularité. Nous mettons en évidence tous les cas possibles et montrons comment définir des méta-singularités pour prolonger les diagrammes espace-temps.

Dans toute ce chapitre, nous ne considérons que des diagrammes espace-temps maximaux, *i.e.* sans singularité que l'on pourrait remplacer par une valeur non singulière sans changer la validité du diagramme.

Nous considérons le voisinage d'une singularité isolée[1] : signaux passant par la position, signaux et règles toujours présents (*i.e.* s'y accumulant). Nous montrons comment définir ce qui se passe en une telle position et que, dans certains cas, la suite (*i.e.* le cône d'influences émises) n'est plus uniquement définie (ce qui complète la Rem. 59). Nous définissons un nouveau type de règles correspondant à ces singularités, que nous nommons *méta-singularités*. Celles-ci peuvent être ajoutées à la définitions d'une machine pour étendre son fonctionnement et la validité des diagrammes. Certaines méta-singularités font passer dans un contexte non déterministe.

9.1 Singularités isolées

Nous commençons par définir ce qui se trouve « autour » d'une position, dans un voisinage immédiat. Nous considérons ensuite les différents cas possibles.

[1] L'ensemble des méta-singularités est fini ; il ne peut y avoir une accumulation collective de méta-singularités sans qu'une seule ne s'accumule.

Définition 74 Les *valeurs accumulées* autour d'une position (x_0, t_0) sont :

$$V_1(x_0,t_0) \;=\; \bigcap_{\substack{o \in \mathcal{O}_{0\leq t} \\ (x_0,t_0)\in o}} \mathbb{D}(o \setminus \{(x_0,t_0)\}) \;\subseteq\; \{\oslash\} \cup \{\mu_i\}_i \cup \{\rho_j\}_j \cup \mathcal{S} \ .$$

La valeur de la position est exclue car nous voulons caractériser le voisinage pour exprimer la validité locale du diagramme. Étant dans un contexte topologique, nous ne différencions pas, pour l'instant, entre antérieur et postérieur, mais nous le faisons dans la section suivante.

Pour les positions non singulières, la Déf. 45 impose que $V_1(x_0,t_0) \cap (\{\rho_j\}_j \cup \mathcal{S}) = \emptyset$ et $\oslash \in V_1(x_0,t_0)$.

Définition 75 Une singularité en (x_0, t_0) est *isolée* si, et seulement si, $V_1(x_0,t_0) \cap \mathcal{S} = \emptyset$.

L'ensemble $\{\oslash\} \cup \{\mu_i\}_i \cup \{\rho_j\}_j$ étant fini,

Propriété 76 *Pour toute singularité isolée, il existe un ouvert contenant la singularité sur lequel $V_1(x_0,t_0)$ est atteint.*

Tout ouvert contenant une position non singulière atteint aussi $\oslash$, donc

Propriété 77 *Si $V_1(x_0,t_0) \not\subseteq \mathcal{S}$ alors $\oslash \in V_1(x_0,t_0)$.*

Dans ce chapitre nous ne traitons que des singularités isolées pour des diagrammes maximaux. Regardons les différentes valeurs possibles de $V_1(x,t)$, pour avoir une première vision de la situation.

9.1.1 Cas $V_1(x_0,t_0) = \{\oslash\}$

La singularité est inutile car en changeant la valeur de cette position en $\oslash$, on obtient un diagramme valide, identique en dehors des singularités et ayant un ensemble de singularités strictement plus petit.

9.1.2 Cas $V_1(x_0,t_0) \subseteq \{\oslash\} \cup \{\mu_i\}_i$

Il y a donc des signaux mais pas de collision. Regardons les différents cas possibles.

Tous les signaux se terminent ou commencent en (x_0, t_0)

Il ne peut y avoir deux signaux de même vitesse avant (après) (x_0, t_0), sinon ils occupent les mêmes positions.

Si ces signaux entrants et sortants correspondent à une règle, alors le diagramme a une singularité inutile. S'il n'existe pas de règle avec les signaux entrants, alors la règle peut être définie et le diagramme aurait une singularité inutile. S'il existe déjà une règle avec les signaux entrants, mais que ce ne sont pas les mêmes signaux sortants, il y a en fait une règle

non déterministe ou une utilisation des singularités pour faire correspondre ce qu'il y a avant et après. Nous ne sommes plus dans un contexte déterministe dans les deux cas.

Nous refusons qu'il y ait ce type de singularité (nous le faisons apparaître explicitement plus loin) car : soit c'est une règle cachée, soit elle est due à ce que l'on a déjà défini pour une position postérieure.

Il y a des signaux aussi près que l'on veut ne passant pas par (x_0, t_0)

Cela est-il possible ? À première vue oui : si l'on prend uniquement des signaux de même méta-signal de vitesse nulle aux positions initiales $\frac{1}{n}$ pour tout n alors $(0, 0)$ vérifie cela. Mais il faut un nombre infini de signaux (nous montrons dans la Sect. 9.4 comment cela peut être fait à partir d'un nombre fini de signaux) ; toutes les positions $(0, \varepsilon)$ sont alors singulières et donc la singularité n'est pas isolée. Il doit y avoir d'autres singularités dans les valeurs accumulées.

Plaçons-nous sur un ouvert suffisamment petit pour contenir cette accumulation de signaux de méta-signal μ mais aucune règle. Soit ν_μ la vitesse de ces signaux. Pour tout ε de valeur absolue suffisamment petite pour que $(x_0 + \nu_\mu\varepsilon, t_0 + \varepsilon)$ soit aussi dans l'ouvert, ces signaux sont aussi des points d'accumulation des signaux. Donc, la singularité n'est pas isolée.

Définition 78 Une singularité isolée dont l'ensemble des valeurs accumulées ne contient pas de règle est *invalide*.

Cette définition se fait par la négation car notre étude apporte d'autres critères d'invalidité, une fois ceux-ci mis à jour, la Déf. 85 définit la validité d'une singularité.

9.1.3 Cas $V_1(x_0, t_0) \cap \{\rho_j\}_j \neq \emptyset$

Cela ne peut être qu'une singularité puisqu'aucune position non singulière ne contient de règles dans ses valeurs accumulées.

Il faut affiner la caractérisation car il peut y avoir des signaux s'accumulant mais présents au même instant (*i.e.* ils commencent avant et se terminent après).

9.2 Raffinement de la perception de l'entourage

Les règles sont définies en termes de signaux entrants (antérieurs) et sortants (postérieurs) pour bien marquer la spécificité du temps. Nous affinons notre caractérisation des singularités isolées pour en extraire des règles d'un nouveau type que nous nommons *méta-singularités*.

Commençons par les signaux qui terminent ou aboutissent à la position.

Définition 79 Les *méta-signaux entrants, sortants et incidents* à une position (x_0, t_0) sont :

$$\begin{aligned}
M_0^-(x_0,t_0) &= \left\{ \mu \;\middle|\; \begin{array}{l} \exists o \in \mathcal{O}_{0\leq t}, (x_0,t_0) \in o, \forall (x,t) \in o, \\ (t < t_0 \wedge (x - x_0) = \nu_\mu (t - t_0)) \Rightarrow \mathbb{D}(x,t) = \mu \end{array} \right\} \subseteq \{\mu_i\}_i \ , \\
M_0^+(x_0,t_0) &= \left\{ \mu \;\middle|\; \begin{array}{l} \exists o \in \mathcal{O}_{0\leq t}, (x_0,t_0) \in o, \forall (x,t) \in o, \\ (t_0 < t \wedge (x - x_0) = \nu_\mu (t - t_0)) \Rightarrow \mathbb{D}(x,t) = \mu \end{array} \right\} \subseteq \{\mu_i\}_i \ , \\
M_0(x_0,t_0) &= M_0^-(x_0,t_0) \cup M_0^+(x_0,t_0) \subseteq \{\mu_i\}_i \ .
\end{aligned}$$

Ces conditions pour définir les ensembles correspondent à celles de la Déf. 45 pour les méta-signaux et les règles, mais cette fois l'implication n'est que dans un sens ; μ peut très bien être présent ailleurs.

Attention, une suite de signaux discontinue (parsemée de collisions) n'est pas incidente. En particulier, cette suite de discontinuités correspond à une accumulation de règles (étant en nombre fini certaines doivent s'accumuler). Leurs méta-signaux sont alors déjà inclus dans ces règles, il faut donc les écarter (la sous-section suivante montre cela formellement) et ne considérer que les signaux arrivant directement à la singularité.

Nous notons $\mathbb{R}^2_{0\leq t<t_0}$ et $\mathbb{R}^2_{t_0<t}$ les deux ouverts de $\mathcal{O}_{0\leq t}$ suivants :

$$\mathbb{R}^2_{0\leq t<t_0} = \mathbb{R} \times [0, t_0[\qquad\qquad \mathbb{R}^2_{t_0<t} = \mathbb{R} \times]t_0, \infty[\ .$$

Définition 80 Les ensembles des *valeurs accumulées antérieures, concomitantes et postérieures* d'une position (x_0, t_0) sont les sous-ensembles de $\{\oslash\} \cup \{\mu_i\}_i \cup \{\rho_j\}_j \cup \mathcal{S}$ suivants :

$$\begin{aligned}
V_1^-(x_0,t_0) &= \bigcap_{\substack{o \in \mathcal{O}_{0\leq t} \\ (x_0,t_0) \in o}} \mathbb{D}(o \cap \mathbb{R}^2_{0\leq t<t_0}) \ , \\
V_1^=(x_0,t_0) &= \bigcap_{\substack{o \in \mathcal{O}_{0\leq t} \\ (x_0,t_0) \in o}} \mathbb{D}(o \cap \{\, (x,t_0) \mid x \neq x_0 \,\}) \ , \\
V_1^+(x_0,t_0) &= \bigcap_{\substack{o \in \mathcal{O}_{0\leq t} \\ (x_0,t_0) \in o}} \mathbb{D}(o \cap \mathbb{R}^2_{t_0<t}) \ .
\end{aligned}$$

Nous définissons les ensembles des *méta-signaux accumulés antérieurs, concomitants et postérieurs* par :

$$\begin{aligned}
M_1^-(x_0,t_0) &= V_1^-(x_0,t_0) \cap \{\mu_i\}_i \subseteq \{\mu_i\}_i \ , \\
M_1^=(x_0,t_0) &= V_1^=(x_0,t_0) \cap \{\mu_i\}_i \subseteq \{\mu_i\}_i \ , \\
M_1^+(x_0,t_0) &= V_1^+(x_0,t_0) \cap \{\mu_i\}_i \subseteq \{\mu_i\}_i \ .
\end{aligned}$$

Nous définissons les ensembles des *règles accumulées antérieures, concomitantes et postérieures* par :

$$\begin{aligned} R_1^-(x_0,t_0) &= V_1^-(x_0,t_0) \cap \{\rho_j\}_j \subseteq \{\rho_j\}_j \ , \\ R_1^=(x_0,t_0) &= V_1^=(x_0,t_0) \cap \{\rho_j\}_j \subseteq \{\rho_j\}_j \ , \\ R_1^+(x_0,t_0) &= V_1^+(x_0,t_0) \cap \{\rho_j\}_j \subseteq \{\rho_j\}_j \ . \end{aligned}$$

Comme

$$V_1(x_0,t_0) = V_1^-(x_0,t_0) \cup V_1^=(x_0,t_0) \cup V_1^+(x_0,t_0) \ ,$$

une singularité est isolée si aucun des trois ensembles ne contient d'élément de $\mathcal{S}$.

Pour les singularités isolées, nous faisons une typologie en fonction de la vacuité de ces différents ensembles de méta-signaux et d'accumulations. Mais avant cela regardons les relations entre ces ensembles.

9.2.1 Liens entre ces ensembles

Les valeurs de ces ensembles ne sont pas quelconques comme nous le montrons avec les lemmes suivants.

Lemme 81 *Pour une singularité isolée,*

$$M_1^=(x_0,t_0) \subseteq M_1^-(x_0,t_0) \cap M_1^+(x_0,t_0) \tag{9.1}$$

$$M_1^-(x_0,t_0) = M_0^-(x_0,t_0) \cup M_1^=(x_0,t_0) \cup \bigcup_{\exists \{\mu_j^+\}_j, \{\mu_i^-\}_i \to \{\mu_j^+\}_j \in R_1^-(x_0,t_0) \cup R_1^=(x_0,t_0)} \{\mu_i^-\}_i \ , \tag{9.2}$$

$$M_1^+(x_0,t_0) = M_0^+(x_0,t_0) \cup M_1^=(x_0,t_0) \cup \bigcup_{\exists \{\mu_i^-\}_i, \{\mu_i^-\}_i \to \{\mu_j^+\}_j \in R_1^+(x_0,t_0) \cup R_1^=(x_0,t_0)} \{\mu_j^+\}_j \ . \tag{9.3}$$

►► **Preuve.**

Les signaux correspondant aux méta-signaux de $M_1^=(x_0,t_0)$ doivent se continuer en haut et en bas, ce qui prouve (9.1).

Nous faisons uniquement la preuve pour (9.2), la preuve de (9.3) se faisant de même. Nous démontrons l'égalité en démontrant les deux inclusions.

L'inclusion de $M_0^-(x_0,t_0)$ est automatique, celle de $M_1^=(x_0,t_0)$ vient de (9.1). Chaque ouvert contenant une règle atteint nécessairement les méta-signaux entrants et sortants de cette règle.

Maintenant considérons un méta-signal de $M_1^-(x_0,t_0)$, s'il ne correspond qu'à un nombre fini de signaux dans un ouvert contenant (x_0,t_0) suffisamment petit, alors il passe nécessairement par (x_0,t_0) (sinon ils sont à des distances strictement positives et ne sont pas dans des ouverts suffisamment petits, donc pas dans $M_1^-(x_0,t_0)$).

S'il y a une infinité de signaux de ce méta-signal dans tout ouvert, chacun se termine soit avant, soit à, soit après t_0. Donc il y en a forcément une infinité se terminant avant, à ou après.

S'il y en a une infinité se terminant avant, alors considérons l'ensemble de collisions qui y mettent fin, elles s'accumulent en (x_0, t_0) (sinon, comme le signal est antérieur, il ne peut être dans $M_1^-(x_0, t_0)$). Les règles correspondantes étant en nombre fini, nécessairement il en existe une dans $R_1^-(x_0, t_0)$.

S'il y en a une infinité se terminant à t_0, alors il y doit y avoir une règle dans $R_1^=(x_0, t_0)$.

S'il y en a une infinité se terminant après, alors ceux-ci passent aussi près que l'on veut de (x_0, t_0) et existent à t_0. Le méta-signal est alors dans $M_1^=(x_0, t_0)$.

◁◀◀

Ne perdons pas de vue que les vitesses sont bornées, on ne peut avoir de signaux de plus en plus plats (et pouvant se terminer en dehors d'ouverts) au fur et à mesure que l'on s'approche de la position.

Lemme 82 *Pour une singularité isolée,*

$$M_1^=(x_0,t_0) \subseteq \bigcup_{\exists \{\mu_i^-\}_i, \{\mu_i^-\}_i \to \{\mu_j^+\}_j \in R_1^-(x_0,t_0)} \{\mu_j^+\}_j \cap \bigcup_{\exists \{\mu_j^+\}_j, \{\mu_i^-\}_i \to \{\mu_j^+\}_j \in R_1^+(x_0,t_0)} \{\mu_i^-\}_i \ .$$

►► **Preuve.**

Si les signaux correspondant à un méta-signal de $M_1^=(x_0, t_0)$ ne sont pas dans la première des unions, alors ces signaux existent depuis un temps $t_1 < t_0$ et ils provoquent d'autres singularités sur tout $]t_1, t_0[$ (comme pour le cas 9.1.2). Il en est de même pour l'autre union.

◁◀◀

Nécessairement, les sorties de règles s'accumulant en bas ne peuvent que : continuer après t_0 (donc dans $M_1^=(x_0, t_0)$) ou, se terminer dans des règles en bas (donc dans $R_1^+(x_0, t_0)$ ou $R_1^=(x_0, t_0)$).

Lemme 83 *Pour une singularité isolée,*

$$\bigcup_{\exists \{\mu_i^-\}_i, \{\mu_i^-\}_i \to \{\mu_j^+\}_j \in R_1^-(x_0,t_0)} \{\mu_j^+\}_j = M_1^=(x_0,t_0) \cup \bigcup_{\exists \{\mu_j^+\}_j, \{\mu_i^-\}_i \to \{\mu_j^+\}_j \in R_1^-(x_0,t_0) \cup R_1^=(x_0,t_0)} \{\mu_i^-\}_i \ ,$$

$$\bigcup_{\exists \{\mu_j^+\}_j, \{\mu_i^-\}_i \to \{\mu_j^+\}_j \in R_1^+(x_0,t_0)} \{\mu_i^-\}_i = M_1^=(x_0,t_0) \cup \bigcup_{\exists \{\mu_i^-\}_i, \{\mu_i^-\}_i \to \{\mu_j^+\}_j \in R_1^+(x_0,t_0) \cup R_1^=(x_0,t_0)} \{\mu_j^+\}_j \ .$$

Les signaux présents à t_0 doivent être engendrés (et détruits) par les collisions correspondant aux règles de $R_1^-(x_0, t_0)$ (et $R_1^+(x_0, t_0)$) sinon la singularité n'est pas isolée.

9.3 Vision du voisinage

Parmi les différents ensembles, $M_1^-(x_0, t_0)$ et $M_1^+(x_0, t_0)$ n'ont pas d'intérêt car ils se déduisent des autres (Lem. 81). Nous définissons le contexte d'une singularité, puis les méta-singularités, par :

Définition 84 Le *contexte* d'une singularité en (x_0, t_0) est le sextuple

$$ctx(x_0,t_0) \;=\; \begin{pmatrix} M_0^+(x_0,t_0) & R_1^+(x_0,t_0) \\ M_1^=(x_0,t_0) & R_1^=(x_0,t_0) \\ M_0^-(x_0,t_0) & R_1^-(x_0,t_0) \end{pmatrix} \;.$$

Nous avons opté pour une présentation en matrice : les lignes sont avant, à et, après ; les colonnes les méta-signaux et les règles. Pour avoir une présentation proche de celle des règles, nous utilisons aussi :

$$\big(\, M_0^-(x_0,t_0),\, R_1^-(x_0,t_0) \,\big) \xrightarrow{(\, M_1^=(x_0,t_0),\, R_1^=(x_0,t_0) \,)} \big(\, M_0^+(x_0,t_0),\, R_1^+(x_0,t_0) \,\big) \;.$$

Ces contextes sont en nombre fini et définissent les singularités isolées. La dernière formalisation permet de mettre en évidence ce qui est avant, concurrent et postérieur sous une forme état - condition - résultat.

Un contexte pour lequel $R_1^-(x_0,t_0)$ est vide devrait être, de notre point, de vue une règle, à moins d'accepter qu'une collision entre signaux puisse provoquer une accumulation en sortie.

Définition 85 Un contexte est *valide* si, et seulement si, $R_1^-(x_0,t_0)$ n'est pas vide et s'il vérifie les différents lemmes de la Sous-sect. 9.2.1. L'ensemble des *méta-singularités* isolées est un ensemble de contextes valides. Une méta-singularité isolée est *déterministe* si $R_1^+(x_0,t_0)$ est vide.

Des méta-singularités peuvent être ajoutées à une machine pour en étendre son fonctionnement.

En particulier, pour une méta-singularité isolée déterministe, ces conditions impliquent que $M_1^=(x_0,t_0)$ est vide et, $M_1^+(x_0,t_0) = M_0^+(x_0,t_0)$ (Lem. 81). Postérieurement, il ne peut y avoir que des signaux sortant de la singularité. S'il y avait des signaux issus des règles de $R_1^=(x_0,t_0)$, il y aurait accumulation de collisions postérieures ou accumulation de signaux après (et la singularité ne serait pas isolée).

9.4 Exemples déterministes

Dans cette section nous présentons différents exemples pour bien saisir les composantes du contexte et les liens entre celles-ci.

9.4.1 Accumulation de collisions

Il n'est pas nécessaire d'avoir une machine à signaux compliquée pour obtenir une singularité comme le montre la Fig. 9.1 : quatre méta-signaux et deux règles suffisent.

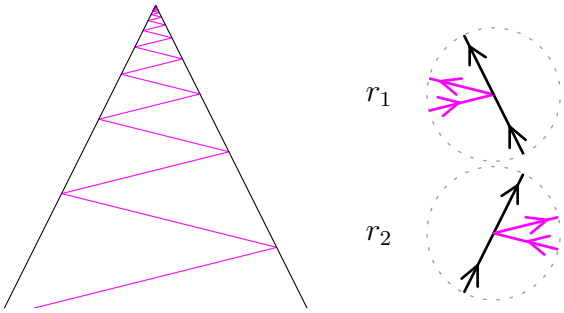


FIG. 9.1 – Exemple de singularité sans signal incident.

Définissons cette méta-singularité par :

$$(\,\emptyset,\,\{r_1, r_2\}\,) \xrightarrow{(\emptyset,\emptyset)} (\,\emptyset,\,\emptyset\,) \quad .$$

Elle signifie que rien ne ressort (comme sur la figure, il n'y a rien au-dessus de la pointe). Elle ne prend pas en compte les signaux présents, ce qui se conçoit puisqu'ils participent tous aux collisions de l'accumulation (Lem. 81).

9.4.2 Présence de signaux incidents

Ajoutons un autre signal sur le côté, comme sur la Fig. 9.2. Ce signal est incident mais ne participe pas aux collisions accumulées, il vient naturellement se joindre à la collision comme pour la première partie.

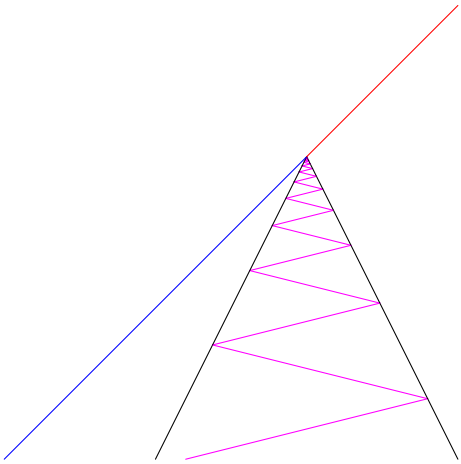


FIG. 9.2 – Exemple de singularité avec un signal incident.

Pour qu'il sorte un signal sur la Fig. 9.2, nous définissons la méta-singularité par :

$$(\,\{\text{bleu}\},\,\{r_1, r_2\}\,) \xrightarrow{(\emptyset,\emptyset)} (\,\{\text{rouge}\},\,\emptyset\,) \quad .$$

9.5 Exemples non déterministes

Cette fois nous imposons des règles dans $R_1^+(x_0, t_0)$.

9.5.1 Accumulation de collisions postérieures

Reprenons l'exemple de la Fig. 9.1 et changeons la définition de la méta-singularité par :

$$(\emptyset,\ \{r_1, r_2\}) \xrightarrow{(\emptyset,\emptyset)} (\emptyset,\ \{r_3, r_4\}) \quad .$$

Les règles additionnelles sont définies sur la Fig. 9.3. Les deux diagrammes de cette figure sont alors valides, bien que différents dans le cône d'influences émises de la singularité.

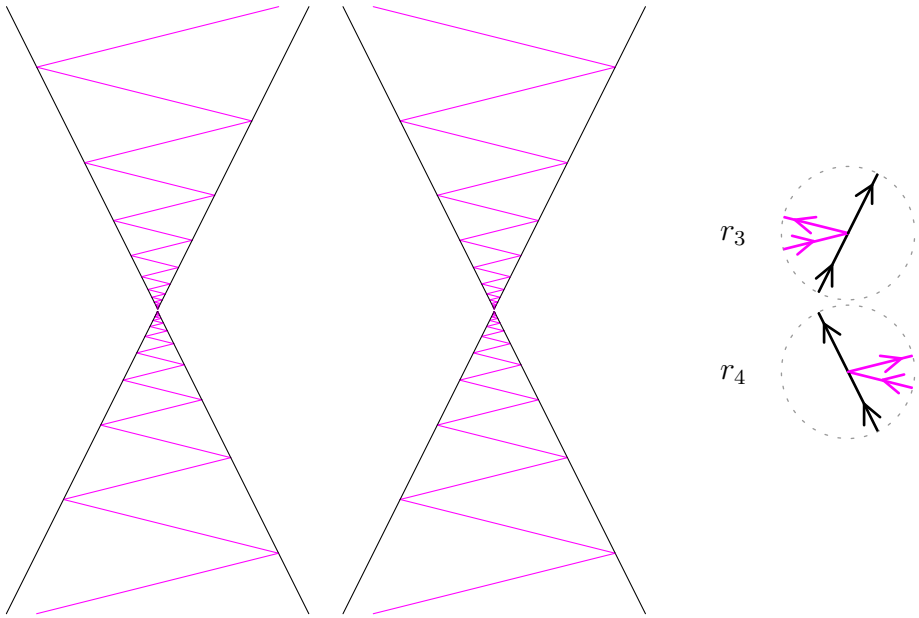


FIG. 9.3 – Exemple de singularité avec $R_1^+(x_0, t_0)$ non vide.

Il y a en fait un nombre non dénombrable (ou dénombrable si l'on se restreint aux positions rationnelles) de possibilités de continuer après la singularité, car l'on peut faire des homothéties de manière à placer le dernier rebond à gauche visible sur la figure, sur tout un intervalle.

9.5.2 Accumulation de signaux

Modifions la règle r_1 pour l'exemple de la Fig. 9.1 de manière à ce qu'elle produise à chaque fois un signal vert de pente 0, on obtient alors la Fig. 9.4. L'ensemble $M_1^=(x_0, t_0)$ n'est pas vide (il se réduit à $\{\text{vert}\}$).

Cette singularité n'est pas isolée. En effet, les positions strictement au-dessus sont aussi des singularités.

Définissons la méta-singularité suivante :

$$(\emptyset,\ \{r_1, r_2\}) \xrightarrow{(\{\text{vert}\},\emptyset)} (\emptyset,\ \{r_5, r_6\}) \quad .$$

Les nouvelle règles étant définies sur la Fig. 9.5. Ces règles permettent de faire disparaître tous les signaux verts. L'alternance des signaux rouges et bleus est forcément sur une même

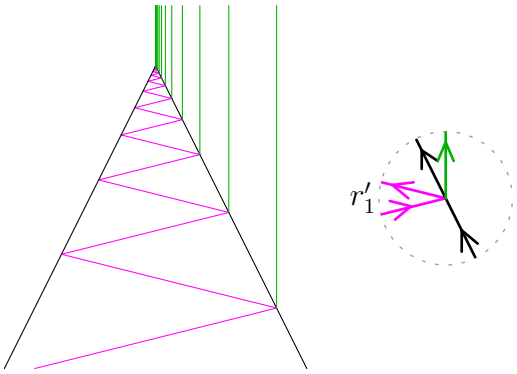


FIG. 9.4 – Exemple engendrant une accumulation de signaux concomitants.

ligne et cette ligne ne peut démarrer qu'à la singularité (preuve en prenant une de ces collisions et en la prolongeant de chaque côté).

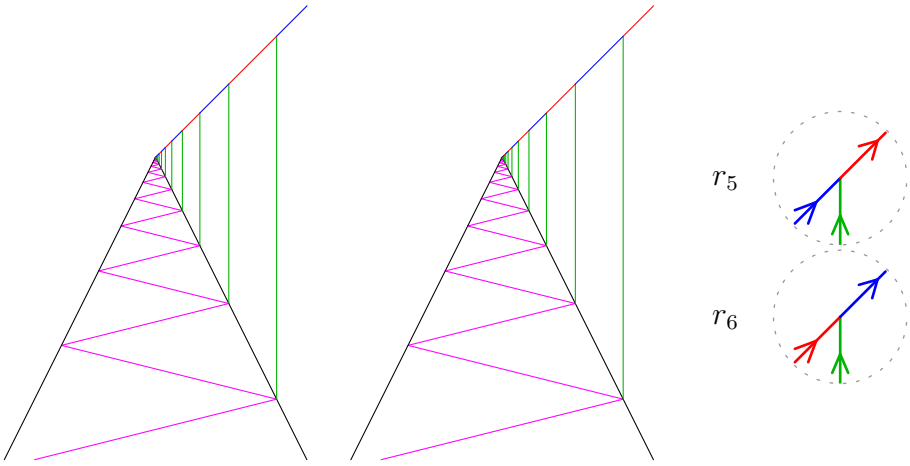


FIG. 9.5 – Accumulation de signaux avec seulement deux continuations possibles.

De la même manière, nous pouvons faire, pour n'importe quel entier n, une construction ayant exactement n continuations possibles. La présence de signaux incidents sur la gauche (*i.e.* le côté où il n'y a pas accumulation de signaux) n'apporte rien.

9.6 Discussion

9.6.1 Singularité d'ordre supérieur

Il est possible d'avoir des accumulations de singularités isolées comme le montre le diagramme de Fig. 9.6 où sont aussi représentées les deux méta-singularités utilisées.

La Fig. 9.6 laisse deviner comment obtenir des singularités d'ordre 3 et plus. On devine aussi quelle figure fractale est engendrée en itérant la construction (son ordre n'est pas fini). On peut imaginer des accumulations de tout ordre (sur les ordinaux).

En s'inspirant de ce qui a été fait sur les mots sur les ordinaux (définitions de BÜCHI [Büc76] et de CHOUEKA [Cho78]), nous proposons des directions pour traiter ces singularités :

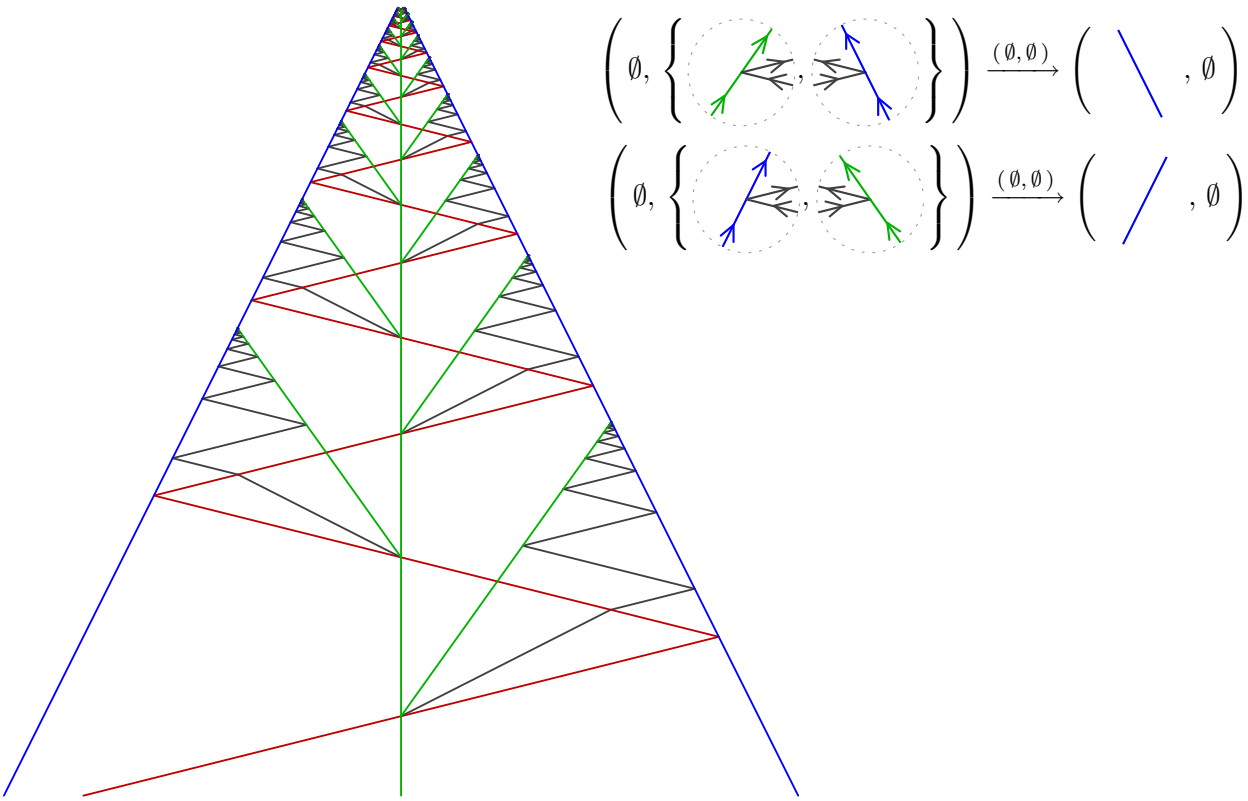


FIG. 9.6 – Singularité d'ordre 2.

- restreindre les contextes en en excluant tout ce qui n'est pas $\oslash$, signal ou règle, cela revient à tout projeter sur les contextes d'ordre 1 (éventuellement en en amendant la définition) ;
- définir des contextes de tout ordre :
  - définir des méta-singularités isolées de tout ordre (*i.e.* $V_1(x,t)$ ne contient que $\oslash$, des méta-signaux, des règles et des méta-singularités isolées d'ordres inférieurs) et, considérer des contextes augmentés des méta-singularités des ordres inférieurs ;
  - omettre les singularités mais en répartissant les autres valeurs selon leurs ordres d'accumulation (*i.e.* des M_α, R_α et V_α pour tout ordre α) ;

  On peut alors définir des singularités isolées de tout ordre par des contextes valides. Il y a de nouveau des relations entre les différents ensembles par type de valeurs accumulées avant, à et après. Mais, le nombre de contextes devient infini[2] ;

Nous ne poursuivons pas plus avant sur le sujet dans ce mémoire. L'Ann. B montre que les singularités isolées ne sont qu'un cas particulier.

[2]Il n'est pas sûr que les contextes puissent être finiment décrits.

Chapitre 10

Décidabilité de l'apparition d'accumulations

Après avoir rappelé la hiérarchie arithmétique, nous nous interrogeons sur la prédiction de l'apparition d'une accumulation. Nous prouvons qu'au maximum ce problème est dans Σ_2^0, puis nous prouvons qu'il est Π_1^0-difficile et Σ_2^0-complet.

Apparition d'une accumulation
Instance
$\mathcal{M}$: machine à signaux rationnelle,
c_0 : configuration initiale où il n'y a qu'un nombre fini de signaux (à des positions rationnelles),
Question
Dans l'évolution de $\mathcal{M}$ sur c_0, y aura-t-il des accumulations ?

10.1 Hiérarchie arithmétique

Nous rappelons et complétons la présentation de la hiérarchie arithmétique de la Sous-sect. 4.1.3.

Définition 86 Un ensemble, ou problème de décision, P, est dans Σ_n^0, s'il existe un prédicat récursif (calculable) total, $\mathcal{P}$, à $n+1$ variables libres, tel que x appartient à P si, et seulement si, $\mathcal{P}(x, y_1, y_2, \dots y_n)$, précédé des quantificateurs suivants, est vrai :

$$\forall x \in \mathbb{N}, \qquad x \in P \Longleftrightarrow \exists y_1 \in \mathbb{N},\ \forall y_2 \in \mathbb{N},\ \exists y_3 \in \mathbb{N}, \dots\ \mathcal{P}(x, y_1, y_2, \dots y_n) \text{ vrai } .$$

Le dernier prédicat est existentiel ($\exists y_n \in \mathbb{N}$) si n est impair ; sinon il est universel ($\forall y_n \in \mathbb{N}$).

On définit aussi la classe des ensembles complémentaires (ce qui correspond à la négation de la formule logique, la négation étant transférée dans le prédicat total). Un problème P est dans Π_n^0 si, et seulement si,

$$\forall x \in \mathbb{N}, \qquad x \in P \Longleftrightarrow \forall y_1 \in \mathbb{N},\ \exists y_2 \in \mathbb{N},\ \forall y_3 \in \mathbb{N}, \dots\ \mathcal{P}(x, y_1, y_2, \dots y_n) \text{ vrai } .$$

Le dernier prédicat est existentiel ($\exists y_n \in \mathbb{N}$) si n est pair ; sinon il est universel ($\forall y_n \in \mathbb{N}$).

On a donc :

$$P \in \Sigma_n^0 \iff \complement P = \mathbb{N} \setminus P \in \Pi_n^0 \ .$$

Les ensembles décidables correspondent au degré zéro : $\mathcal{R} = \Sigma_0^0 = \Pi_0^0$. Les ensembles récursivement énumérables et co-récursivement énumérables correspondent au niveau un : $\mathcal{RE} = \Sigma_1^0$ et co$-\mathcal{RE} = \Pi_1^0$.

De plus, en ajoutant des quantificateurs sur de nouvelles variables inutilisées :

$$\Sigma_n^0 \cup \Pi_n^0 \ \subseteq \ \Sigma_{n+1}^0 \cap \Pi_{n+1}^0 \ .$$

10.2 Apparition d'une accumulation appartient à Σ_2^0

Soit $\mathcal{M}$ et c_0 des données pour ce problème. Nous utilisons notre caractérisation topologique pour cerner l'apparition d'accumulation.

Lemme 87 *Un diagramme sans singularités inutiles à partie déterministe maximale ayant une configuration initiale finie sans singularités, a une accumulation si, et seulement si, il existe un entier t_0 tel qu'il y ait un nombre infini de collisions avant t_0.*

►► Preuve.

S'il n'y a qu'un nombre fini de collisions, il ne peut y avoir de point d'accumulation ; réciproquement, s'il y a un point d'accumulation, il y a forcément un nombre infini de collisions.

Supposons maintenant qu'il y ait un nombre infini de collisions.

Si pour tout entier t, il existe un entier n tel que la n^{e} collision soit postérieure à t alors il ne peut y avoir de point d'accumulation avant t. Ceci étant vrai pour tout t, il n'y a pas de point d'accumulation.

S'il existe un entier t_0 tel qu'une infinité de collisions ait lieu avant t_0, alors les dates des collisions convergent vers une valeur t_1 (suite infinie croissante bornée, on ne s'occupe pas des collisions qui pourraient avoir lieu après la première valeur d'accumulation). Regardons les positions spatiales des collisions avant t_1, il y en a une infinité dans un espace fini. En effet la configuration initiale étant finie, en un temps fini, elle n'a pu s'étendre que de manière finie. L'espace, comme le temps, est (un ensemble) complet ; comme il y a une infinité de collisions dans un compact (les positions sont bornées en espace et en temps), il y a forcément un point d'accumulation.

◁◀◀

Il est possible d'écrire un programme qui calcule la n^{e} collision (ou dit s'il n'y en a pas autant). Le prédicat « *La n^e collision existe et se produit avant t_0* » est récursif. La paire $(\mathcal{M}, c_0)$ est donc une instance positive de Apparition d'une accumulation si, et seulement si,

$$\exists t_0 \in \mathbb{N},\ \forall n \in \mathbb{N},\ \text{la } n^{\text{e}} \text{ collision existe et se produit avant } t_0 \ .$$

Donc :

Lemme 88 Apparition d'une accumulation *appartient à Σ_2^0.*

10.3 Π_1^0-difficile

Pour cela, nous montrons que l'on peut y co-réduire un problème Σ_1^0-complet, *i.e.* y réduire son complémentaire. Le problème Arrêt d'un automate à deux compteurs de la Sect. 4.3 est Σ_1^0-complet.

Arrêt d'un automate à deux compteurs
Instance
$\mathcal{A}$: automate à deux compteurs,
(a_0, b_0) : valeurs initiales pour les deux compteurs
Question
$\mathcal{A}$ démarré avec (a_0, b_0) s'arrête-t-il ?

Nous prouvons que Apparition d'une accumulation est Π_1^0-difficile en y co-réduisant Arrêt d'un automate à deux compteurs.

Nous présentons la construction et prouvons la validité de la réduction.

10.3.1 Construction et réduction

Nous reprenons la construction de la simulation d'un automate à deux compteurs de la Sect. 4.2. Nous la modifions de manière à ce que, si la machine s'arrête, elle efface tous les signaux présents, ce qui est facile puisque les signaux sont bornés par un bord de chaque côté. Nous utilisons ensuite la construction de la Sect. 7.5 pour contracter la bande à un triangle, en modifiant les règles de manière à ce que les signaux de la structure soient aussi effacés si l'automate à deux compteurs s'arrête.

Si l'automate s'arrête, tous les signaux sont effacés, il y a un nombre fini de collisions et donc pas d'accumulations. S'il ne s'arrête pas, alors, la contraction continue itérée n'est pas stoppée et provoque une accumulation (celle de la structure de contraction).

La Fig. 10.1 montre une simulation d'un automate à deux compteurs s'arrêtant toujours et sa simulation pour $a = 4$ et $b = 4$ avec effacement de la simulation. La Fig. 10.3 montre une simulation d'un automate à deux compteurs s'arrêtant, subissant la contraction à un triangle. À gauche, seul l'automate est effacé, à droite, la structure de contraction l'est aussi.

Effacer la simulation de l'automate

Le nettoyage de la simulation quand le signal stopRi apparaît se fait en remplaçant les règles correspondantes de la Fig. 4.10 par celles de Fig. 10.2. Le fonctionnement est le suivant : stopRi continue sur la droite en détruisant tous les signaux qu'il rencontre jusqu'à rencontrer bord qui marque la fin à droite de la simulation de l'automate. Cette collision ne produit qu'un stopLe qui part sur la gauche. Celui-ci détruit tous les signaux qu'il rencontre jusqu'à atteindre l'autre extrémité de la simulation (*i.e.* l'autre bord), les deux disparaissent alors.

Effacer la simulation de l'automate et la contraction

La construction précédente permet de nettoyer la simulation de l'automate, elle le fait toujours quand la contraction est mise en place. Par contre, il faut que la structure de la

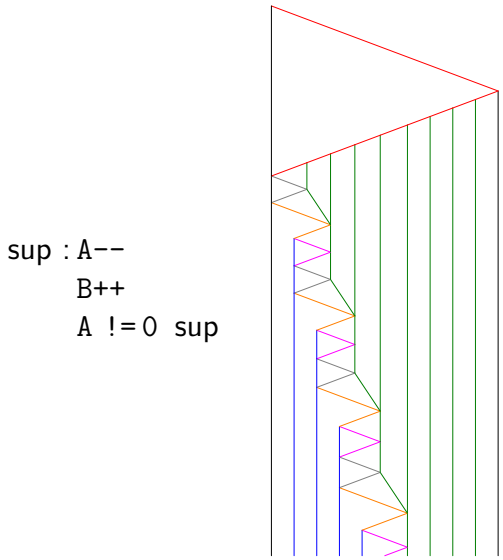


FIG. 10.1 – Fin effaçante seule et dans une contraction itérée.

Méta-signal		Règles
Information	Vitesse	
stopRi	2	{ stopRi, bord } → { stopLe }
		{ stopRi, b } → { stopRi }
		{ stopRi, bMv } → { stopRi }
stopLe	-2	{ bord, stopLe } → { }
		{ a, stopLe } → { stopRi }
		{ aMv, stopLe } → { stopRi }

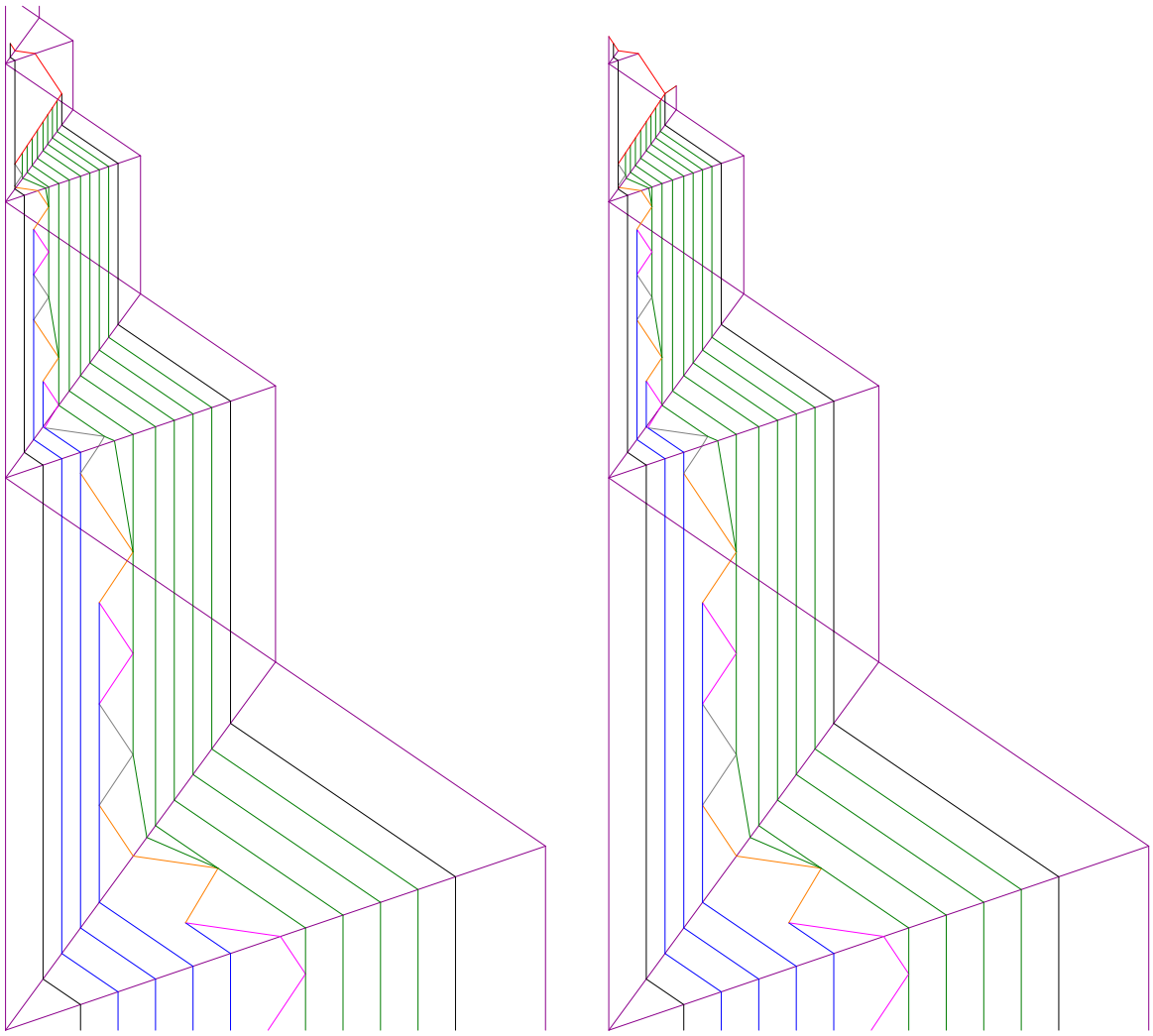
FIG. 10.2 – Règles modifiées pour effacer l'automate.

contraction soit elle aussi effacée. L'idée est d'effacer tous les signaux au-dessus (et donc au passage) de stopLe et d'un autre signal, stopStrRi, que l'on fait partir sur la droite quand stopLe est engendré, comme sur la partie droite de la Fig. 10.3.

Suivant le Chap. 7, nous notons avec une apostrophe tous les signaux homothétiques. Nous parlons de zone droite et de zone penchée pour différencier les zones du diagramme normales (signaux originels) et homothétiques (signaux prime). Nous désignons par *de la simulation* tous les signaux qui prennent part à la simulation, prime inclus, et par *de la structure* tous ceux qui font partie de la structure de contraction continue itérée.

La construction de la contraction continue itérée est faite sur la simulation de l'automate effaçant la simulation, de sorte qu'il ne reste plus qu'à effacer la structure. La Figure 7.10 indique quel signal de structure peut être rencontré dans quelle zone ; nous ne la redonnons pas, mais y référons pour les différents signaux de la structure.

Deux nouveaux méta-signaux stopStrRi et stopStrRi′ sont ajoutés. Ils ont la même vitesse



(a) Seule la simulation est effacée

(b) La structure de contraction est aussi effacée

FIG. 10.3 – Simulation subissant contraction continue itérée.

et celle-ci est incluse entre celles de scaleHi et de scaleLo (en pratique ν_0). Ils ne peuvent rattraper scaleLo (qui atteindra back) mais peuvent rattraper scaleHi (ceci évite de considérer certaines collisions).

L'idée est que lorsque stopRi atteint le bord droit de la simulation, il ne repart que deux signaux : stopStrRi continue à droite pour effacer la structure et stopLe part à droite pour effacer l'automate ainsi que le reste de la structure de contraction. Il faut donc modifier les règles contenant stopLe (et stopLe′) pour qu'il efface la structure sur son passage (sachant que les collisions avec les signaux de la structure peuvent aussi comprendre des signaux de la simulation). Abordons tous les cas depuis l'engendrement de stopLe (ou stopLe′) et stopStrRi (ou stopStrRi′).

Quand stopRi atteint bord, cela peut être fait sur la zone droite ou la zone penchée avec

le passage de back, scaleHi ou scaleLo. Ils sont supprimés, stopLe (ou stopLe′ selon le cas) et stopStrRi (ou stopStrRi′) partent de cette collision. Les signaux de structure éventuellement présents sont détruits. Il est important de différencier stopStrRi et stopStrRi′ car selon la zone, la limite gauche de la structure est marquée par, respectivement, bordRi ou back. Les différentes règles sont dans la partie (a) de la Fig. 10.4.

Considérons stopStrRi et stopStrRi′. Le signal stopStrRi est sur la zone droite, il peut rencontrer back, scaleHi et bordRi. Il efface le premier, efface le second en devenant stopStrRi′ (changement de zone) puis, disparaît avec le dernier (l'extrémité droite de la structure étant atteinte). Le signal stopStrRi′, quant à lui, ne peut rencontrer que back qui marque l'extrémité droite (car c'est la zone penchée), il l'efface et disparaît. Les différentes règles sont dans la partie (b) de la Fig. 10.4.

Regardons stopLe et stopLe′. Quand ils apparaissent, il ne reste plus de b ni de bMv, ils ont été effacés par stopRi. Au retour, ils doivent effacer les signaux de la simulation et de la structure. Pour les collisions ne contenant que des signaux de la simulation, il ne reste rien à faire puisque cela est déjà prévu dans l'effacement de l'automate, avant la construction de la contraction. Il reste donc à voir les cas avec des signaux de la structure, avec ou sans signaux de la simulation. Comme il y a au plus un signal de la simulation dans la collision, nous inscrivons ces règles sans autres commentaires ; nous considérons le cas de bord dans le paragraphe suivant. Le signal stopLe ne peut rencontrer back, car l'équation (7.4) implique que back devrait arriver par au-dessus ce qui n'arrive pas par construction. Le signal stopLe ne peut donc rencontrer que scaleLo, il l'efface et devient stopLe′. Le signal stopLe′ ne peut rencontrer que scaleHi, il l'efface et devient stopLe. Les différentes règles sont dans la partie (d) de la Fig. 10.4.

La rencontre de stopLe avec bord doit être modifiée car stopLe doit continuer pour achever d'effacer la structure. De nouveau, selon la zone, la collision peut se produire avec scaleLo ou scaleHi qui est alors effacé. Les différentes règles sont dans la partie (e) de la Fig. 10.4.

Ensuite, stopLe peut encore rencontrer des signaux de structure, mais leur effacement est déjà donné. Le seul cas restant est la rencontre avec bordLe qui met fin à l'effacement de la structure après celui de la simulation. Là encore, grâce à (7.4), cela ne peut se produire dans la zone penchée. Les différentes règles sont dans la partie (c) de la Fig. 10.4.

Nous avons considéré tous les cas possibles, les signaux marquant la limite du calcul effacent bien tous les signaux de la simulation et de la structure.

Lemme 89 *L'automate à deux compteurs $\mathcal{A}$ s'arrête sur (a_0, b_0) si, et seulement si, le diagramme a un nombre fini de collisions. S'il ne s'arrête pas, alors il y a accumulation. Le problème* Apparition d'une accumulation *est donc Π_1^0-difficile.*

10.4 Σ_2^0-complet

Pour montrer que Apparition d'une accumulation est Σ_2^0-complet, comme il est dans Σ_2^0 (Lem. 88), nous y réduisons le problème Σ_2^0-complet suivant :

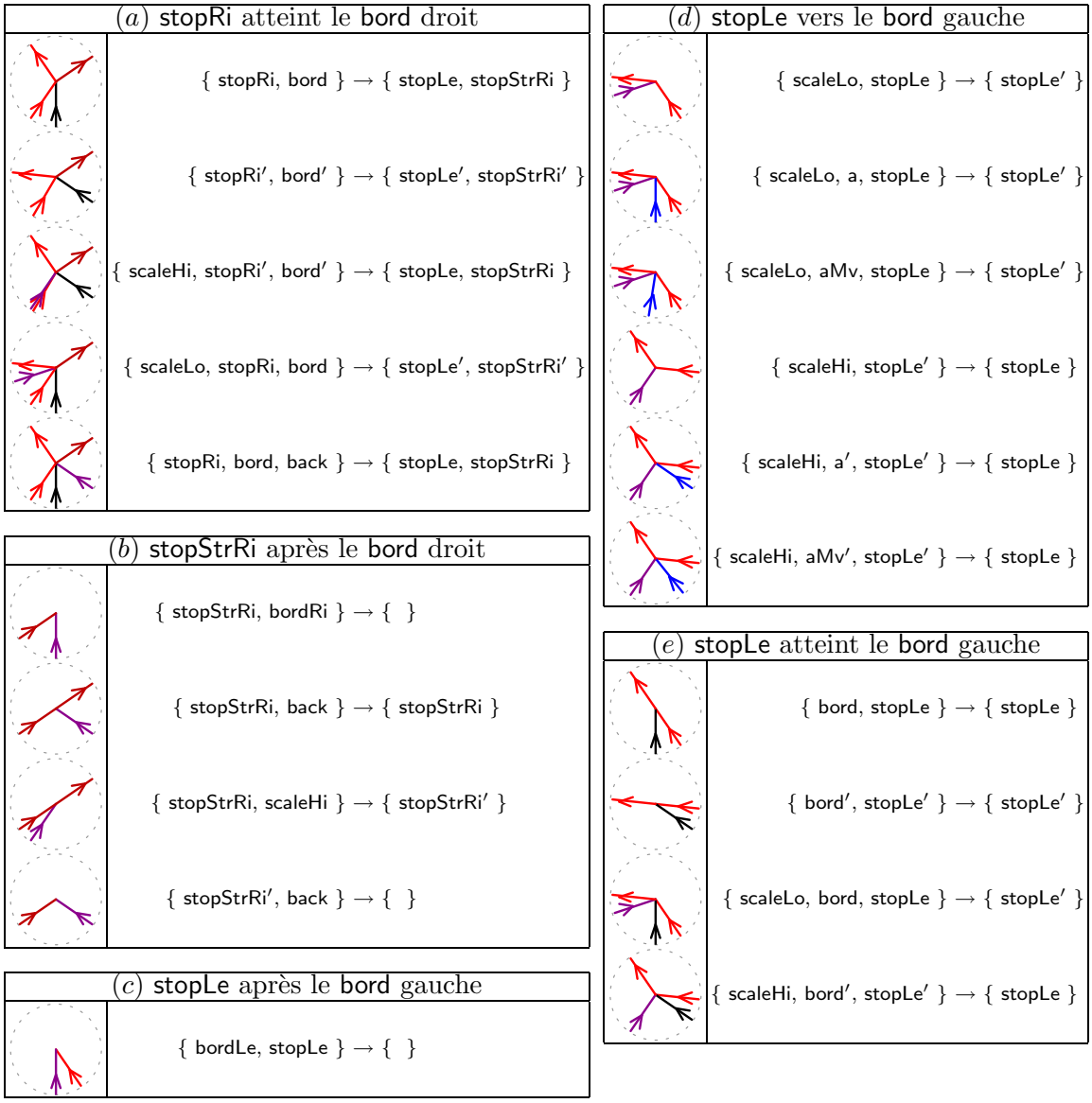


FIG. 10.4 – Règles pour tout effacer.

Existence d'un calcul sans fin pour automate à deux compteurs
Instance
$\mathcal{A}$: automate à deux compteurs,
Question
Existe-t-il des valeurs initiales (a_0, b_0) telles que $\mathcal{A}$ démarré avec (a_0, b_0) ne s'arrête pas ?

Dans [Odi99, p. 621], le problème NTOT est prouvé Σ_2^0-complet. La seule différence avec le problème présenté ci-dessus est que NTOT est formulé à partir des fonctions calculables.

Nous présentons la construction, puis prouvons la validité de la réduction.

10.4.1 Construction

L'idée est de faire faire tous les calculs possibles de l'automate à deux compteurs. Les simulations subissent, comme précédemment, des contractions itérées continues ; de sorte qu'elles n'occupent chacune qu'une partie finie du diagramme et ne provoquent des accumulations que si le calcul correspondant ne se termine pas. Nous obtenons un diagramme sans singularités si aucun calcul n'est infini et avec accumulation si au moins un des calculs ne s'arrête pas.

Partant de la construction précédente, il faut donc fournir un mécanisme faisant démarrer tous ces calculs. L'idée est de reprendre la construction de la Fig. 5.1 du Chap. 5 où l'on fait apparaître un treillis infini correspondant à $\mathbb{N}^2$. À chaque point du treillis, on fait démarrer le calcul correspondant. Les positions des calculs sont décrites dans la Fig. 10.5. Le tout étant paramétré de manière à ce qu'aucun calcul ne déborde sur le treillis ce qui est très facile puisque chaque simulation est bornée pas la taille de la structure de contraction continue itérée.

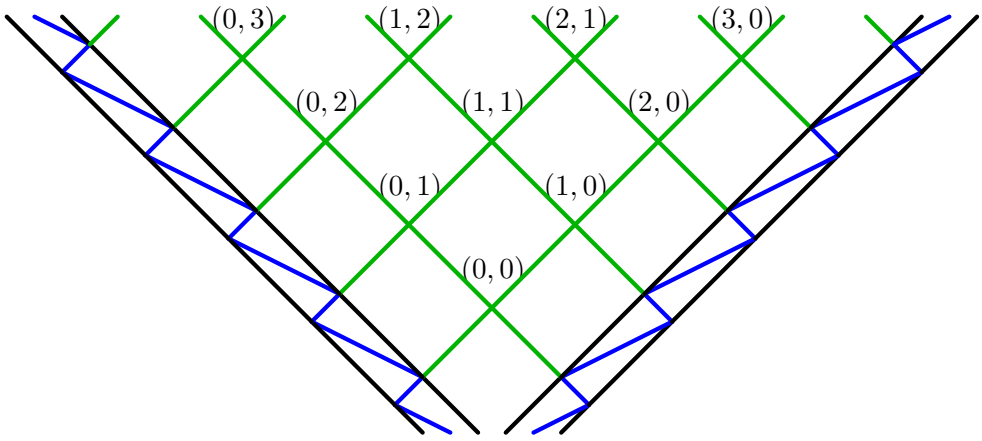


FIG. 10.5 – Positions des calculs dans le treillis.

La Figure 10.6 montre ce qui se passe une fois tout mis en place. L'automate à deux compteurs utilisé est le même que précédemment : il s'arrête toujours.

La construction est présentée en deux parties : comment engendrer le treillis avec toutes les valeurs puis, comment démarrer chacun des calculs. Toutes les collisions qui ne sont pas définies sont blanches, *i.e.*, les méta-signaux sortants sont les méta-signaux entrants.

Création du treillis

Comme on le voit, avec un signal (bleu, en fait deux signaux) qui reste coincé entre deux signaux (noir) parallèles, il est possible de faire partir des signaux (verts) de manière très régulière. Si l'on bloque d'autres signaux avec les mêmes pentes, il font partir des signaux avec la même régularité. L'idée est d'avoir deux signaux qui rebondissent de chaque côté, leurs intersections provoquent le démarrage de la simulation de l'automate.

Il reste à fournir les valeurs des compteurs. Cela se fait en bloquant d'autres signaux et en ajoutant des règles, de manière à ce qu'il y ait un signal de plus bloqué à chaque fois.

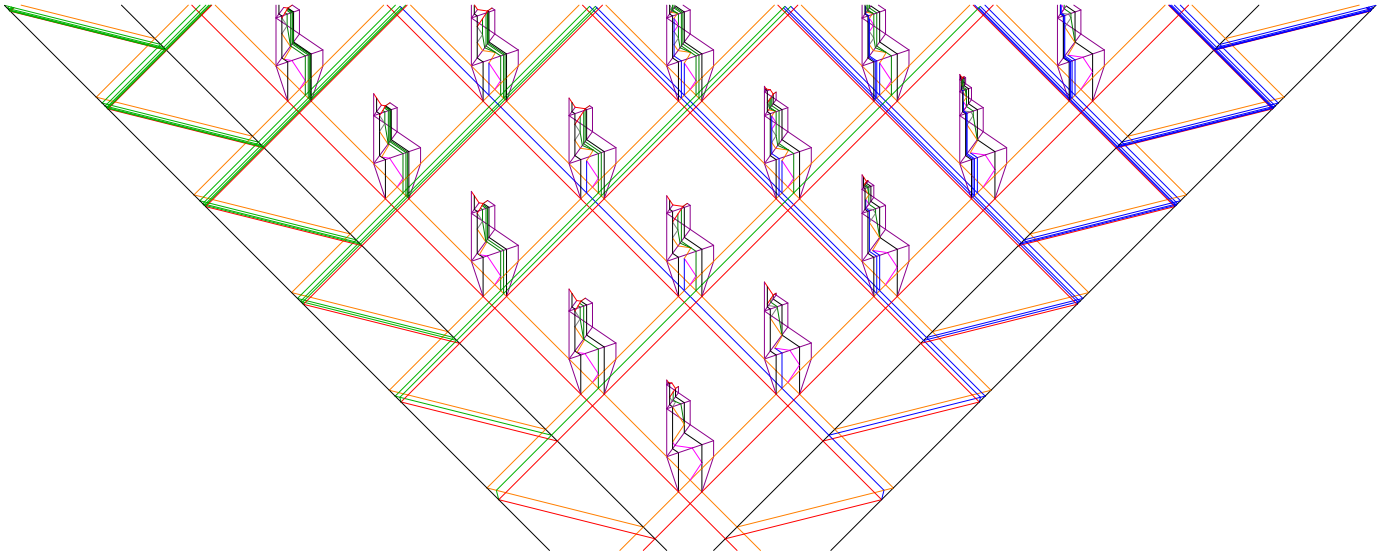


FIG. 10.6 – Simulations de l'automate à deux compteurs pour toutes les valeurs.

Cette construction ne provoque pas de singularité. À chaque instant il n'y a qu'un nombre fini de signaux. La partie droite engendrant la structure est isolée sur la Fig. 10.7.

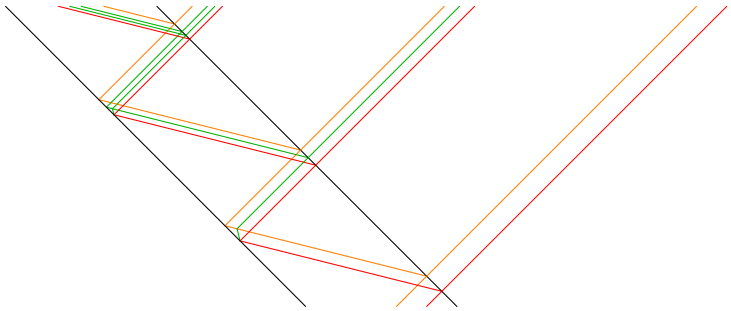


FIG. 10.7 – Engendrer toutes les valeurs de B.

Sur la Fig. 10.8, seules les premières itérations sont considérées et les noms des signaux sont indiqués.

Détaillons le mécanisme de création de signaux à intervalles réguliers, nous verrons ensuite comment le compteur B est augmenté. Regardons comment cela se passe pour hiLe et hiLeBack ; le mécanisme est le même pour loLe et loLeBack et pour bLe et bLeBack. Pour avoir un signal qui revient régulièrement, on fait rebondir les deux signaux hiLe et hiLeBack entre deux bordLe. Quand hiLe atteint le bordLe supérieur, hiLeBack est engendré pour relancer la création et hiLe est aussi engendré pour continuer à l'extérieur. Sur la Fig. 10.9, ce mécanisme est extrait et les règles sont données.

Tous les signaux sont donc engendrés de manière régulière de cette façon avec les mêmes vitesses, comme ils font des aller-retours entre les mêmes bordLe, ils ont donc la même périodicité.

Dans notre construction, les signaux engendrés sont de vitesse 1, les signaux revenant (back) sont de vitesse -4 et, les bordLe, de vitesse -1. Si les bordLe sont écartés de 6 unités

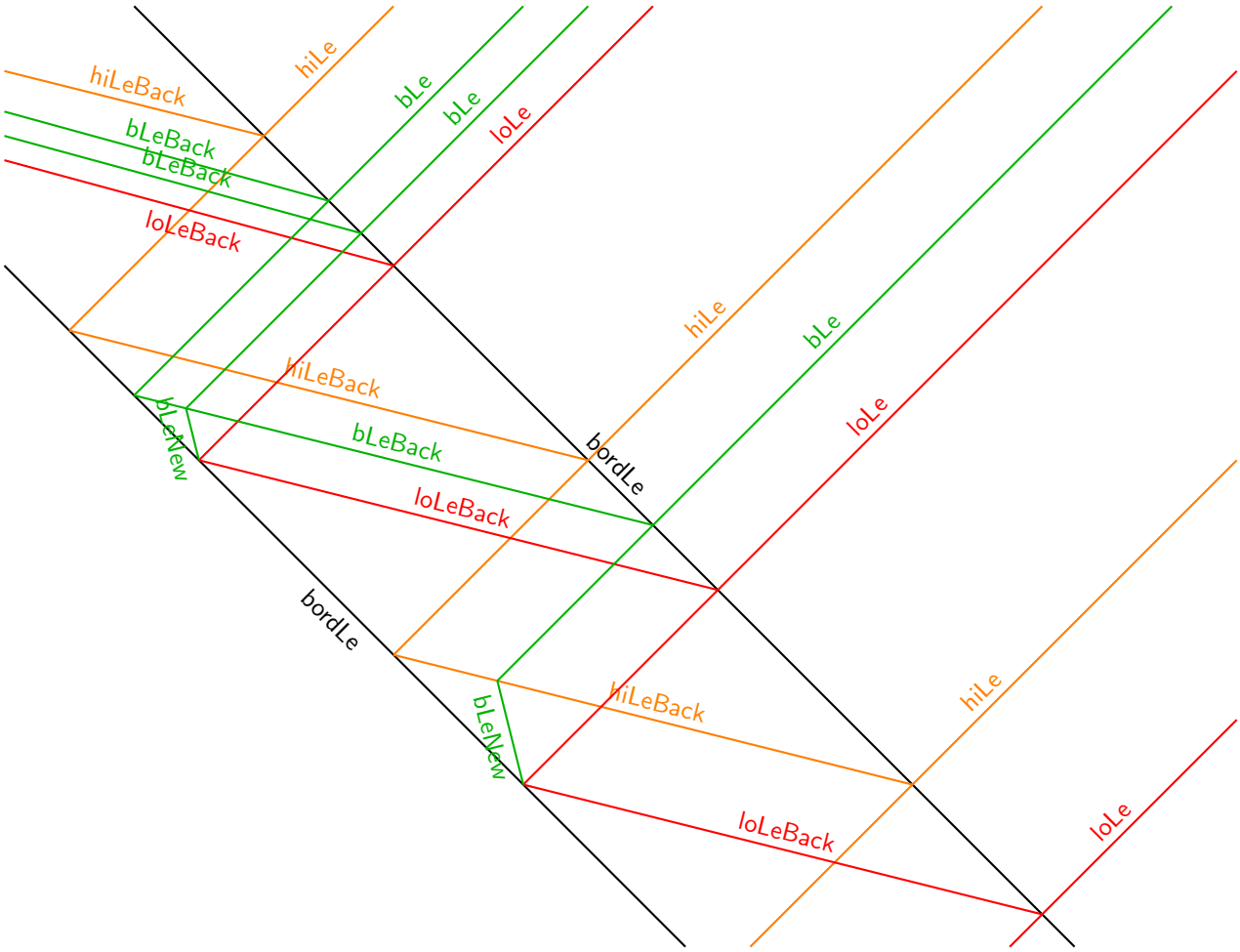


FIG. 10.8 – Structure pour engendrer toutes les valeurs de B.

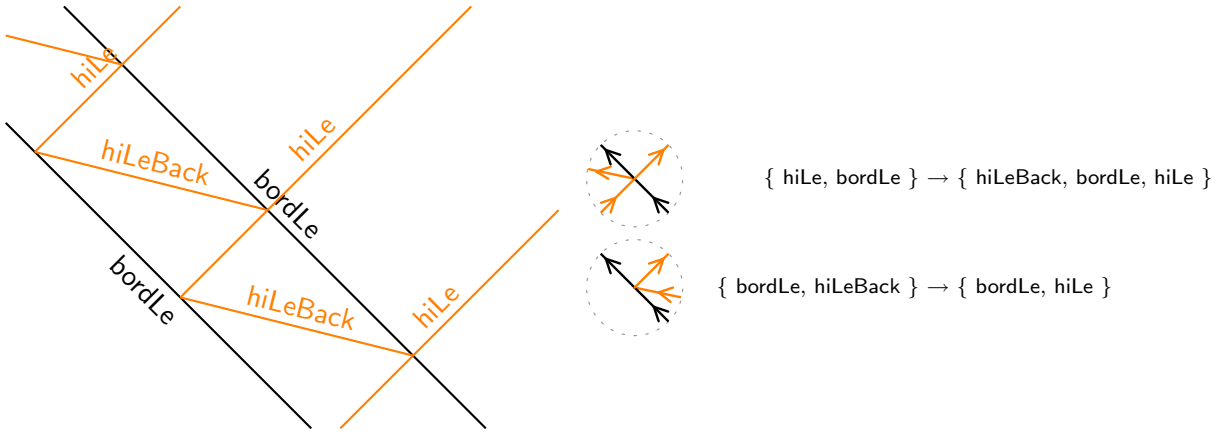


FIG. 10.9 – Structure pour engendrer des signaux régulièrement.

(d'espace), alors les signaux sont émis toutes les 7 unités (de temps) avec, à chaque fois, un décalage de 7 sur la gauche. Il en est de même pour la construction de droite, par symétrie, mais le décalage est sur la droite.

La valeur de B est en unaire ; pour l'augmenter de 1, il suffit donc d'ajouter un signal bLe. Ceci est fait en modifiant la règle du retour de loLeBack comme suit :

$$\{ \text{bordLe}, \text{loLeBack} \} \to \{ \text{bordLe}, \text{bLeNew}, \text{loLe} \} \ .$$

Le signal bLeNew va engendrer un nouveau bLe à la première rencontre de bLeBack ou dehiLeBack (au départ, il n'y a pas de bLeBack). La Figure 10.10 montre ces collisions et les règles correspondantes.

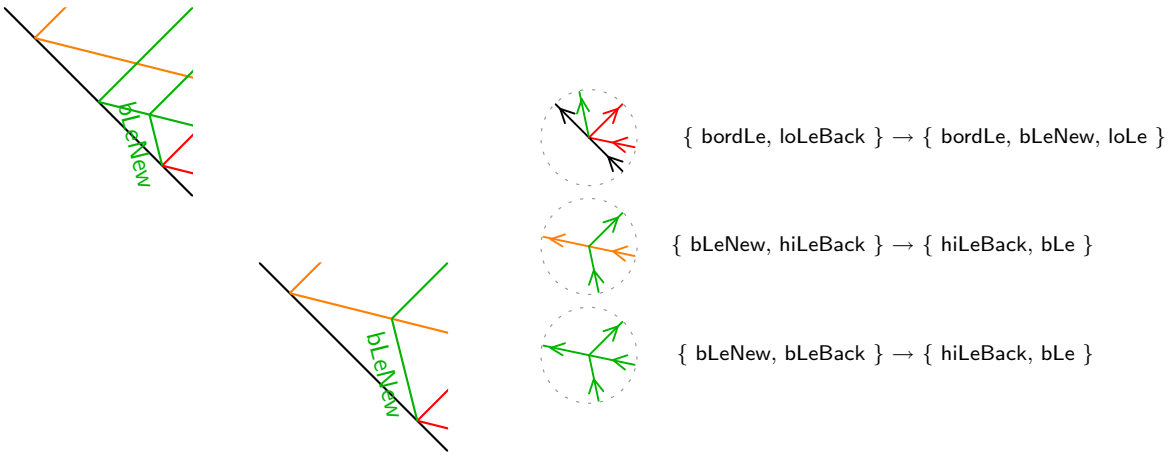


FIG. 10.10 – Règles pour augmenter B de 1 et utilisations.

La vitesse de bLeNew est $\frac{-1}{4}$ ce qui fait apparaître le nouveau bLe à mi-chemin de loLe et bLe (ou hiLeBack).

La partie droite envoie donc les signaux par groupes de :

$$\text{hiLe} \ \ \text{bLe}^b \ \ \text{loLe}$$

pour toutes les valeurs successives de b. De même, la partie droite envoie

$$\text{loRi} \ \ \text{aRi}^a \ \ \text{hiRi}$$

pour toutes les valeurs successives de a dans le sens opposé. Ces groupes vont tous se rencontrer deux à deux.

Démarrages des calculs

Un calcul est démarré à chaque rencontre de hiLe bLe^b loLe venant de gauche et de loRi aRi^a hiRi venant de droite.

Dans un premier temps, nous montrons comment la simulation est lancée et, dans un second temps, comment la contraction itérée continue est démarrée (et est paramétrée pour ne pas se superposer au reste de la construction). À chaque fois, les signaux de ces groupes sont toujours retrouvés en sortie de collision, de manière à ce que la simulation et la structure de contraction ne soient que des « ajouts » sur le diagramme. De nouveau, les collisions non explicitement définies sont blanches.

Pour démarrer la simulation, il faut positionner les deux bords (bord), les signaux codant les compteurs (a et b) et la première instruction. Les bords sont engendrés par la rencontre de hiLe et loRi, à gauche, et hiRi et loLe, à droite (il part aussi des signaux pour engendrer la structure de compression). Les a (resp. b) sont engendrés par les rencontres de aRi avec hiLe (resp. bLe avec hiRi). Tous ces signaux sont de vitesse nulle. La première instruction est engendrée par la rencontre de hiLe et hiRi. La simulation est donc démarrée sur la configuration :

$$\mathsf{bord}\ \ \mathsf{a}^a\ \ \mathsf{aller\_1}\ \ \mathsf{b}^b\ \ \mathsf{bord}\ .$$

Ce qui est bien le calcul que l'on veut démarrer.

La Figure 10.11 montre le démarrage de la simulation pour a=3 et b=4 ainsi que les règles nécessaires. La vitesse des instructions est choisie suffisamment lente pour que toutes les collisions de la simulation aient lieu au-dessus des groupes de signaux.

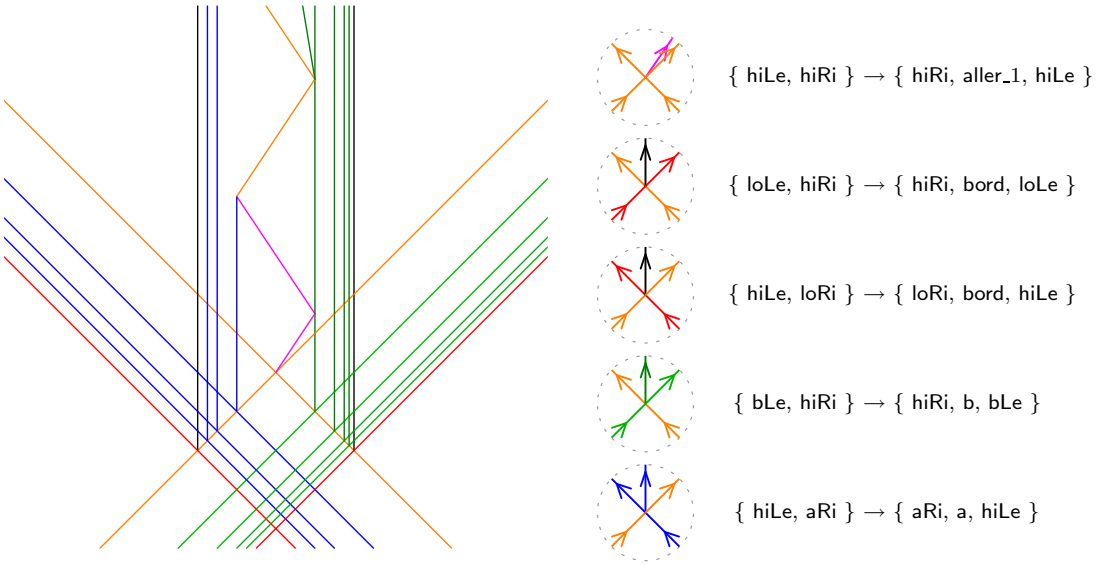


FIG. 10.11 – Exemple et règles pour lancer la simulation.

Démarrage de la contraction itérée continue

Cette contraction est faite, comme dans la section précédente, *i.e.* uniquement pour la simulation ; elle n'a aucun effet sur la structure qui fait démarrer tous les calculs. De plus, elle est positionnée et paramétrée de manière à se « blottir » dans une maille du treillis.

Que faut-il pour démarrer la contraction ? Des signaux bordLe, scaleHi et scaleLo, à gauche, et un signal bordRi, à droite. Ceux-ci doivent encadrer la simulation.

Ils sont engendrés de manière très simple : les rencontres entre hiLe et loRi, à gauche, et, hiRi et loLe, à droite, en plus d'engendrer les bord, engendrent des signaux startIterRi et startIterLe respectivement. Ces signaux sont de vitesse $\frac{1}{3}$ et $\frac{-1}{3}$. Ils s'écartent donc des bords et vont rencontrer hiRi et hiLe. Ces rencontres sont simultanées et engendrent des signaux bordLe, scaleHi et scaleLo à gauche et un signal bordRi à droite.

La Figure 10.12 donne un exemple, toujours avec a=3 et b=4 et décrit toutes ces règles.

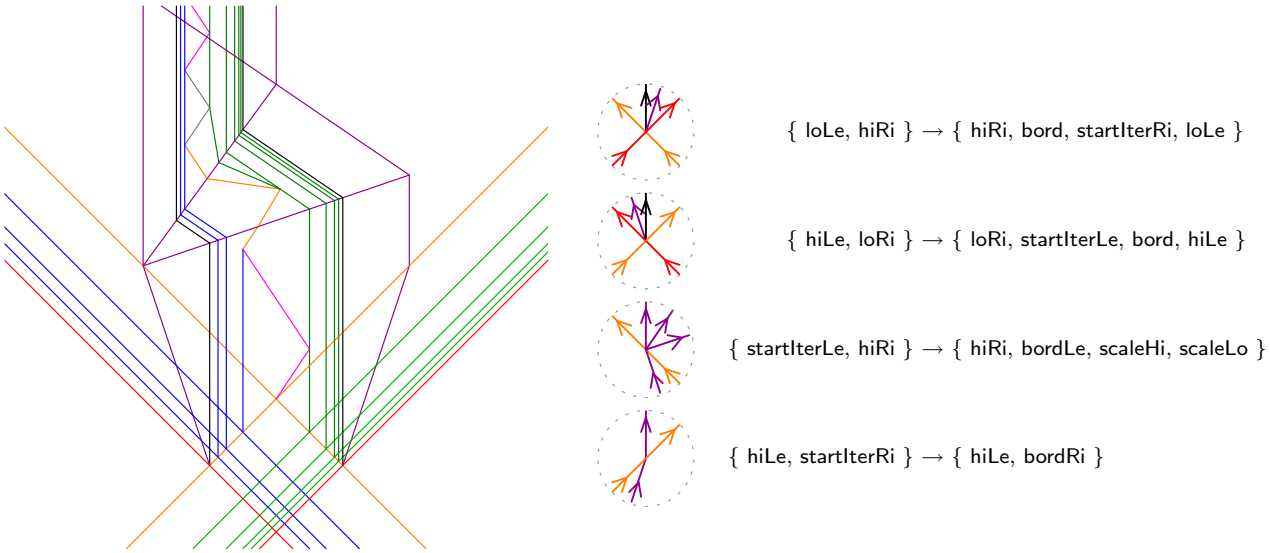


FIG. 10.12 – Exemple et règles pour lancer la compression.

Comment s'assurer que les simulations compressées ne touchent pas la structure lançant tous les calculs ? Observons que toutes les compressions occupent au maximum la même taille (le triangle avec une accumulation dans la pointe du haut) et que cette taille ne dépend que de l'espacement entre loLe et hiLe et, loRi et hiRi. Ces espacements sont toujours les mêmes et ne dépendent que des espacement initiaux des loLe et hiLe et, loRi et hiRi qui engendrent le treillis. L'autre paramètre, qui, cette fois, donne la taille de la maille d'où ne doit pas sortir la simulation, est la distance entre les deux bordLe d'un côté et les deux bordRi de l'autre. Augmenter cette distance ne fait qu'allonger la période de tous les signaux émis sans changer la distance entre les signaux d'un même groupe. On peut donc s'arranger pour que les simulations rentrent dans les mailles[1].

10.4.2 Correction de la réduction

Pour notre construction, si l'automate à deux compteurs s'arrête pour toute valeur de A et B, alors le diagramme engendré n'a pas d'accumulations, car aucune des simulations n'en produit et la structure n'en produit jamais. Par contre, s'il existe un couple (a_0, b_0) pour lequel le calcul ne s'arrête pas alors, comme toutes les simulations pour tous les couples sont engendrées, la simulation correspondante ne s'arrête pas et grâce à la compression continue itérée, il y a une accumulation.

Le problème EXISTENCE D'UN CALCUL SANS FIN POUR AUTOMATE À DEUX COMPTEURS se réduit donc au problème APPARITION D'UNE ACCUMULATION. Le premier étant Σ_2^0-complet :

Lemme 90 *Le problème* APPARITION D'UNE ACCUMULATION *est donc* Σ_2^0*-difficile.*

[1]Dans nos exemples, pour une vitesse $\nu_0 = \frac{22}{15}$ pour la contraction itérée ; la distance entre les paires de bordLe (et bordRi) est 5 fois celle entre loLe et hiLe (loRi et hiRi).

Nous avons déjà montré l'appartenance du problème Apparition d'une accumulation à Σ_2^0 (Lem. 88), donc :

Théorème 91 *Le problème* Apparition d'une accumulation *est donc* Σ_2^0*-complet et en particulier indécidable.*

Chapitre 11

Conclusion et perspectives

Nous avons présenté un nouveau modèle de calcul géométrique.

Dans la première partie, nous l'avons abordé sous l'angle de traces de signaux en définissant des machines à signaux (méta-signaux et règles de collision pour ceux-ci). Nous avons montré qu'elles sont capables de simuler tout automate à deux compteurs et sont donc Turing-universelles. Nous avons aussi montré comment modifier n'importe quelle machine à signaux pour réaliser différentes modifications géométriques (*i.e.* translations et homothéties). Nous avons également fait des constructions itérées utilisant ces modifications de manière à restreindre l'espace et le temps nécessaire au calcul : restriction à un espace borné puis à une durée également bornée. Ces constructions sont possibles grâce à la continuité du temps et de l'espace. Par contre, travailler sur du continu peut donner lieu à des phénomènes du type *Achille et la tortue* : une infinité de collisions (discrètes) pendant une durée (continue) bornée.

La seconde partie du mémoire reformule les diagrammes espace-temps en termes topologiques. Nous avons montré que cette définition correspond à celle de la première partie. Mais elle permet en plus de considérer les singularités (*i.e.* là où il y a accumulation de signaux et de collisions). Nous avons proposé une typologie des singularités isolées et donné un critère de validité pour les diagrammes espace-temps en de tels points. Il apparaît que la sortie d'une singularité peut être non déterministe. En construisant, pour toute automate à deux compteurs, une machine et une configuration initiale qui n'a pas d'accumulation si tout calcul de l'automate s'arrête, nous avons prouvé que la prédiction de l'apparition d'une accumulation est Σ_2^0-complet.

Au cours de notre étude, beaucoup de questions, dépassant le cadre de ce mémoire, nous sont venues à l'esprit. Nous souhaitons les présenter car elles nous semblent prometteuses tant pour les machines à signaux en elles-mêmes que pour les liens et les éclairages que cela apporte sur différents sujets. Nous les avons classées en quatre catégories :

- liens avec les automates cellulaires,
- questions sur le modèle, sans singularité, en lui-même,
- questions sur les singularités,
- liens avec d'autres modèles continus.

Pour nous, les plus importantes sont

- la validation des preuves géométriques par un théorème de transfert (Sous-sect. 11.1.1)

qui sont à l'origine du modèle ;
- les liens avec les autres modèles du calcul (Sect. 11.4) pour situer le modèle ;
- l'universalité intrinsèque pour les machines à signaux (Sous-sect. 11.2.3) pour montrer la cohérence du modèle.

11.1 Liens avec les automates cellulaires

Nous avons introduit les machines à signaux comme, d'une part, une formalisation de toutes les constructions géométriques utilisées pour produire des automates cellulaires correspondant à certaines spécifications [DL97, Fis65, Got66, Maz96a, Maz96b, MT99, VMP70] et, d'autre part, comme des extensions continues des automates cellulaires dont la dynamique se décrit avec des particules ou des signaux discrets [BNR91, HSC01, LN90]. Il est naturel de chercher à approfondir les liens entre les deux modèles, le nôtre étant à temps et espace continus, celui des automates cellulaires étant entièrement discret.

Notons au passage qu'il est aisé de simuler, avec une machine à signaux, un automate cellulaire (ou une machine de Turing) en se servant d'une grille (comme celle de la Fig. 5.1) comme support pour avoir un pas de discrétisation.

11.1.1 Approximation par des automates cellulaires et validité de preuves géométriques discrètes

Le processus de définition continue puis discrétisation dans diverses constructions sur les automates cellulaires demande le passage d'une machine à un automate cellulaire qui la simule tant que les positions restent quantifiées (et donc sans singularité). Peut-on automatiser ce processus ? et garantir une certaine « qualité » sur la simulation. En particulier, quelles propriétés des diagrammes continus peut-on retrouver dans leurs homologues discrets ? Comme cela se passe dans le cas des fusiliers. Peut-on systématiser cela et avoir des théorèmes de *transfert* garantissant la préservation de propriétés à travers la discrétisation ? Le plupart du temps, la construction continue est explicitée pour la compréhension, mais la construction et les démonstrations sont sur les automates cellulaires ; il serait plus commode de prouver en continu puis d'utiliser de tels théorèmes.

L'existence de tels théorèmes de transfert, même avec des conditions restrictives, est très importante : il transforme une preuve heuristique ou combinatoire en calculs de géométrie analytique.

11.1.2 Constructions issues des automates cellulaires

Nous considérons ici le passage inverse : que peuvent apporter les constructions existantes sur les automates cellulaires aux machines à signaux ?

Fonctions constructibles *à la* Fischer Patrick Fischer a construit les graphes de différentes fonctions dans des diagrammes espace-temps d'automates cellulaires [Fis65]. Cette

façon de faire a mené à considérer plus généralement ce type de fonctions [MT99]. Pouvons-nous faire de même ? Sans traitement de points d'accumulation, on ne peut obtenir que des fonctions linéaires par morceaux, peut-on faire plus avec des singularités, sachant que celles-ci ne seraient pas nécessairement isolées ?

Diverses opérations sont possibles sur les courbes dans les diagrammes espace-temps des automates cellulaires [Fis65, Maz96a, MT99], quelles sont leurs contreparties continues ?

Motif de fond Souvent, pour les particules (comme sur la Fig. 1.2 tiré de [BNR91, HSC01]), les signaux se superposent à un motif de fond périodique. Ces motifs sont discrets, ont-ils une contrepartie continue ?

Il est possible de concevoir les constructions contractantes comme des motifs de fond (non plus périodiques mais récursifs), car même s'ils modifient la géométrie du calcul, ils ne le modifient pas. Peut-on en créer d'autres et en tirer parti ? Ou inversement, les extraire pour revenir à un calcul épuré des modifications causées ?

11.2 Machine à signaux

11.2.1 Notion de complexité

Nous n'avons pas défini de notion de complexité. Cela est pourtant nécessaire pour comparer des machines à signaux et les problèmes qu'elles résolvent.

Mesures empiriques de complexité

Nous pouvons affirmer que le temps écoulé ou la largeur maximale ou la surface occupée par un calcul ne signifient pas grand chose, en effet, ces valeurs peuvent varier du tout au tout en changeant la position initiale et des vitesses initiales. Malheureusement, il reste toujours possible d'accélérer autant que l'on veut un calcul comme le montrent les constructions des Chapitres 6 et 7. On peut craindre que la mesure de complexité ne reflète plus celle de l'algorithme mais la qualité d'une accélération géométrique implicite.

Si l'on veut utiliser ce type de mesure, on doit ajouter des contraintes de manière à pouvoir légitimement comparer. Les premières contraintes qui nous viennent à l'esprit sont un espacement minimal des signaux au démarrage, une borne maximale sur les modules des vitesses... Jacopini et Sontacchi définissent de telles complexités pour les *automates de* Mondrian [JS90] ; mais ils imposent à tout calcul d'avoir un nombre borné de fragments (signaux pou nous) dans chaque boule de rayon 1. Cette condition peut se tester sur les calculs finis (mais la finitude comme l'accumulation sont indécidables). La seconde idée qui nous vient à l'esprit est de normaliser ces mesures en les divisant par la plus petite distance (sur $\mathbb{R}^2_{0\leq t}$) entre deux signaux n'entrant pas en collision. Cette distance est nulle dès qu'il y a une accumulation et, peut l'être pour les calculs infinis.

Nombre de collisions

On peut définir une complexité en nombre de collisions, ce qui correspondrait au nombre d'instructions élémentaires. Mais à ce moment-là, il n'y a pas de prise en compte du parallélisme inhérent au modèle. À travers nos constructions, nous avons montré que l'ordre des collisions est fondamental. Pour avoir des notions d'espace et de temps, on peut considérer les plus grandes chaînes et anti-chaînes pour l'ordre de causalité tel que défini par la Déf. 28 du Chap. 5. On peut alors envisager des classes de complexité, leur robustesse par rapport à notre notion de simulation (Chap. 5) étant automatique.

Pour chaque construction, la structure est, elle aussi, emboîtée dans le calcul obtenu. Il est légitime de se demander dans quelle mesure les modifications faites ont un pendant comme une agrégation ou un produit particulier dans la théorie des ordres. Ceci permettrait de caractériser l'influence de constructions sur ces complexités.

Si l'on accepte les singularités, ces ordres sont infinis et pas forcément bien fondés. Si l'on se restreint à des singularités isolées déterministes (sans accumulation en sortie) : on peut alors utiliser les ordinaux pour classer les chaînes maximales. Notons au passage qu'il existe déjà une extension des automates cellulaires itérant sur des ordinaux [Laf02] ; le passage à la limite pourrait correspondre à des « fronts » de singularités.

Dimension fractale

Pour les calculs infinis (sans forcément de singularités), on peut se poser la question de la dimension fractale. Celle-ci peut-elle servir de mesure de complexité ? Est-elle robuste par rapport aux constructions/emboîtements possibles ? Elle est invariante par homothétie, ce qui montre déjà sa robustesse par rapport aux modifications statiques du Chap. 5.

11.2.2 Réversibilité, probabilisme, non déterminisme et quantique

Nous pouvons définir une machine comme réversible si l'ensemble de ses règles forme une bijection (ou une injection) sur les ensembles de signaux. La machine inverse s'obtient en échangeant les entrées et les sorties de règles ; on peut alors faire marcher la machine à « rebrousse temps ». Cet aspect est très important en informatique [Ben88] car il correspond à la conservation de l'information et de l'énergie. Il a été très étudié pour les machines de Turing [Lec63, Ben73] ainsi que les automates cellulaires [Kar90, Mor92b, TM90] et divers autres modèles (*e.g.* [Mor96] pour les automates à deux compteurs). La question que l'on peut se poser est la puissance des machines à signaux réversibles : peut-on simuler toute machine à signaux par une réversible (pour les automates cellulaires, on sait le faire sur des configurations finies [Mor92a] ou en ajoutant une dimension [Tof77]) ? On peut aussi se poser la question pour les accumulations, ce qui demande de préciser ce que sont les accumulations déterministes.

Le non déterminisme peut apparaître avec les singularités, mais on peut considérer un système non déterministe. Il y aurait alors plusieurs règles pour un même ensemble de signaux entrants.

Il est aussi possible de définir des probabilités quand plusieurs règles existent et, obtenir ainsi une version probabiliste du modèle.

On peut même aller plus loin et imaginer une version quantique du modèle où chaque règle serait munie d'une amplitude. On se rapproche alors d'une vision du signal comme une particule élémentaire en mouvement. Cela n'est pas forcément simple à considérer car il apparaît des questions du type : doit-on avoir des règles spéciales correspondant à des observations ? Doit-on / peut-on prévoir un mécanisme de décorrélation comme dans le monde réel ? Que se passe-t-il au niveau des accumulations ?

Les questions classiques se posent alors avec ces différents paradigmes : Peut-on calculer plus (en terme de calculabilité) ? Peut-on calculer mieux (en terme de complexité) ?

11.2.3 Universalité intrinsèque

Il serait intéressant de construire une machine à signaux intrinsèquement universelle, *i.e.* capable de simuler toutes les autres et d'avoir un « théorème Smn ». Malheureusement, avec la Déf. 30 du Chap. 5, un calcul ne peut s'emboîter que dans un calcul ayant au moins autant de règles. Ceci rend donc impossible d'avoir une machine dont les calculs englobent tous les calculs possibles de toutes les machines.

Pour arriver à ce type d'universalité, il faudra d'autres notions d'englobement et de simulation, mais, si possible, étendant nos définitions. Tout cela est certainement possible en utilisant les ordres de causalité et, à la base, de modules (groupe de collisions) et non des éléments (collisions).

Tant que l'on reste avec des rationnels et des diagrammes espace-temps finis, on peut simuler sur une machine de Turing universelle qui, à son tour, est simulable par une machine à signaux, mais ceci ne correspondra vraisemblablement à aucun emboîtement donc ne saurait convenir. De plus, faire intervenir d'autres modèles de calcul comme intermédiaires n'apporte pas grand chose à la compréhension du modèle.

Il est peu probable qu'une machine intrinsèquement universelle puisse exister pour des machines à signaux à valeurs non rationnelles car ce sont des constantes qu'il faudrait arriver à insérer dans les machines et à manipuler, *i.e.* comment pouvoir coder/manipuler des signaux de toutes pentes dans un même machine. Ce problème n'est pas particulier au modèle, il est en fait beaucoup plus général sur tous les modèles pouvant coder des oracles quelconques sur les données/constantes. Dans notre contexte, montrer que l'on pourrait interroger l'oracle serait un pas vers l'impossibilité ; inversement, montrer que l'on peut coder une vitesse par une distance serait un pas vers la possibilité. Ce dernier serait aussi un pas vers une unification du temps et de l'espace.

11.2.4 Espace et temps

Dimension supérieure Tout ce que nous avons présenté est en dimension 1. Les définitions et résultats sont, pour la plupart, valides en toute dimension[1] ? Par contre les constructions de la première partie ne passent plus en dimension supérieure car il faudrait que l'on utilise des hyperplans pour, *e.g.*, geler tous les signaux. Existe-t-il des moyens de réaliser cela autrement ? Doit-on définir des signaux de dimension 2 et plus ?

[1] On peut se poser la question sur d'autres espaces où ont été définis les automates cellulaires (*e.g.* espaces hyperboliques [IAMI02, MM99] et graphes de Cayley [Rók00]).

L'Annexe B montre des signaux formés de singularités que l'on pourrait envisager de deux dimensions. La Sous-section 11.3.3 propose quelques pistes pour aborder des singularités non isolées. On aurait alors la possibilité de créer des signaux de dimension 2 et plus à partir de signaux unidimensionnels. Attention, la Fig. B.3 montre que l'on peut aussi créer des signaux qui ne sont pas de dimension entière (*i.e.* Cantors) ! Ces signaux peuvent-ils être discrétisés sur des automates cellulaires ? C'est le cas pour celui de la Fig. B.3.

Plus généralement, le problème est de voir et concevoir en dimension 2 et plus. Beaucoup d'algorithmes en dimension 2 ne sont que des extensions/produits d'algorithmes en dimension 1 ; existe-t-il des constructions qui soient propres à la dimension 2 ?

Compréhension de l'espace-temps Par ailleurs, nous nous interrogeons sur notre compréhension de l'espace et du temps. Dans la plupart de nos simulations, le calcul est identique, mais la structure de construction modifie sa localisation. On peut se demander si ce n'est pas en quelque sorte l'espace-temps qui est changé par la structure et non plus simplement la localisation des calculs[2].

11.3 Singularités

Nous avons montré que les singularités font partie du modèle et nous savons en engendrer de plusieurs types, mais peut-on les empêcher ? Nous avons défini des singularités pour traiter les singularités isolées. Il serait intéressant, d'un côté, de caractériser d'autres singularités et, de l'autre, d'en tirer parti pour calculer.

11.3.1 Empêcher les singularités

On peut aussi se poser la question de faire des constructions qui garantissent qu'il n'y ait pas de singularité. Par exemple, la construction de compression itérée continue du Chap. 7 ramène dans un espace-temps borné ce qui se passe à l'infini créant ainsi une singularité, l'inverse est-il possible ? Par cela nous entendons faire une construction qui fasse une « extension » (au lieu d'une contraction) rejetant la première singularité à l'infini mais réalisant toutes les collisions la précédant. Cela est-il possible même quand l'existence ou la position de cette singularité est inconnue ?

11.3.2 Singularités d'ordres supérieurs

La Section 9.6 du Chap. 9 aborde la possibilité de définir des singularités d'ordres supérieurs. L'ordre jusqu'auquel les singularités sont définies n'est pas anodin, comme peut le laisser penser le cas des systèmes à dérivée continue par morceaux [Bou99a] où il conditionne le degré de la hiérarchie arithmétique que l'on peut atteindre. Ceci permet d'aller au-delà de la calculabilité au sens de Turing. Notre système aurait-il des propriétés similaires ?

[2] Je n'irai pas jusqu'à dire que les singularités créées par la structure sont des artefacts de l'espace-temps de type trous noirs.

11.3.3 Singularités non isolées

Peut-on les définir et que peut-on en tirer ? Par exemple, pour le problème très étudié de la synchronisation d'une ligne de fusiliers sur les automates cellulaires [Maz96b], toutes les constructions que nous connaissons sont basées sur des constructions géométriques continues. La synchronisation se produit lorsque la granularité de l'espace (*i.e.* les cellules) est atteinte. Ces constructions géométriques se retranscrivent avec signaux ; mais dans le modèle, il n'y a pas de granularité puisque celui-ci est continu. Les processus de découpages géométriques engendrent alors des figures fractales dont la limite est un « signal horizontal » toutes les singularités, hormis les extrémités, étant identiques.

Nous tentons de donner quelques illustrations de ceci et développons quelques perspectives sur le sujet dans l'Ann. B où nous montrons que les singularités peuvent former un Cantor.

11.3.4 Approche pavages continus

Une autre approche qui aurait pu être suivie est celle de dire que toutes les singularités de même valeur doivent être identiques localement sur des ouverts (carrés ou circulaires) à un changement d'échelle près. Cela revient à définir les méta-singularités par des valeurs sur des ouverts. Cette définition est la même pour $\oslash$, les méta-signaux et les règles. Cela se rapproche aussi de la définition de Jacopini et Sontacchi dans [JS90].

Les principaux changements sont la disparition des ordres de singularité, l'exemple de la Fig. 9.4 peut être rendu déterministe (*i.e.* une seule continuation possible), mais celui de la Fig. 9.5 reste non déterministe, les deux continuations étant homothétiques l'une de l'autre.

Nous pourrions voir cela comme une tentative de passer au continu pour les pavages : les pavages usuels proposent d'assembler des copies de pièces les unes avec les autres, ces pièces sont discrètes et leurs positions aussi (au minimum quantifiées), ici les pièces seraient des ouverts et chaque point de $\mathbb{R}^2$ devrait être bien « pavé ».

Cette approche est purement topologique et définie par des critères de validité locale, celle que nous avons préférée prolonge la formulation des machines de la première partie.

11.4 Liens avec d'autres modèles du continu

11.4.1 Valeurs réelles

Dans tout le mémoire, seulement des valeurs rationnelles ont été considérées (pour les vitesses et les positions initiales ; les positions des collisions sont alors rationnelles). Cela a été fait pour rester dans un contexte *classique*, manipulable sur machine de Turing et programmable sur ordinateur.

Comme nous l'avons déjà abordé dans la Sous-sect. 11.2.3, si l'on accepte des valeurs réelles pour les vitesses (les positions initiales), augmente-t-on les capacités à calculer ? En effet, on fournit alors des constantes réelles pouvant coder, par exemple la fonction caractéristique de n'importe quel ensemble d'entiers (*i.e.* pas forcément récursif). A-t-on,

comme pour les Analog recurrent neural network ou les Analog shift maps [Sie96] un puissance super-Turing ? Par exemple, ces modèles correspondent à P/poly (temps polynomial avec un oracle ne dépendant que de la taille de l'entrée), qui contient des fonctions non récursives.

11.4.2 Plus généralement

En introduction, nous avons fait un rapide panorama des divers modèles de calcul utilisant le continu. Ces modèles ont des capacités de calcul très différentes si ce n'est incomparables, à l'inverse des modèles discrets qui ont donné lieu à la thèse de Church-Turing. Il serait intéressant de relier notre modèle à différentes classes continues. On peut déjà imaginer que les machines à signaux rationnelles et à constantes réelles vont donner lieu à des classes différentes. De plus, il risque d'y avoir à définir des valeurs d'entrée et de sortie de manière très différentes, *e.g.* comme la distance entre deux signaux.

Cette direction nous paraît importante car elle permettrait de situer notre modèle, de profiter de résultats sur d'autres modèles et, d'établir de nouveaux liens entre ces modèles.

Nous avons présenté un modèle de calcul géométrique qui peut être défini par les traces d'une machine autant que par des contraintes locales sur les diagrammes espace-temps. Nous avons montré comment calculer, comment faire des modifications géométrique et tirer parti de la continuité du support (*i.e.* temps et espace). Nous avons montré également que les accumulations ont une place dans notre modèle et peuvent être traitées.

Pour nous, tout calcul a une géométrie intrinsèque, et les singularités ont une signification.

L'utilisation de la géométrie plane est un moyen de calculer. Beaucoup de directions de recherches sont ouvertes, tant sur le modèle lui-même (puissance de calcul avec les singularités, universalité intrinsèque), que sur une discrétisation automatique (vers les automates cellulaires) et, que sur les implications de la géométrie en tant que moyen de calculer partout où elle intervient.

Annexes

Annexe A

Logiciel créé

Nous avons développé un logiciel en Java pour simuler les machines rationnelles. Celui-ci nous a permis d'engendrer toutes les figures du mémoire et de valider expérimentalement les constructions. Ses définitions des machines et des configurations sont enregistrées dans des fichiers en XML (Extensible Markup Language). Les diagrammes espace-temps sont sauvegardés en postscritpt. Une petite interface graphique a été développée pour modifier les machines, les configurations et éditer les diagrammes espace-temps.

Le logiciel se limite aux machines de dimension 1. Ce choix est délibéré car il correspond à tout ce qui est dans ce mémoire, la gestion des collisions est facilement optimisée (en dimension supérieure il faudrait tout tester ou passer par des régionalisations de la configuration du type quad-tree), l'affichage et l'impression d'un diagramme espace-temps de dimension 2 sont aisés (en dimension supérieure cela est beaucoup plus problématique).

Un module a été écrit pour la gestion des automates à deux compteurs, il permet entre autres la génération d'une machine à signaux le simulant ainsi que des sorties du code en LaTeX.

Le logiciel ne gère pas les accumulations (rappelons que leur prédiction est indécidable) bien que nous ayons implanté quelques approximations avec des conditions du type signaux à distance, ou durée de vie, inférieure à un paramètre. Aucun des cas ne nous satisfait pleinement ou plutôt, il ne s'adresse qu'à une accumulation / construction particulière.

Actuellement, le logiciel n'est pas suffisamment abouti ni d'une utilisation suffisamment intuitive pour être mis à disposition sur internet. Il le sera dès que nous aurons remédié à ces handicaps. Néanmoins, nous le fournissons sur simple demande.

Nous ne donnons dans la suite de cette annexe, que quelques éléments d'architecture du logiciel et quelques moyens de l'utiliser. Il ne s'agit en aucun cas d'une documentation technique (elle peut être engendrée avec Javadoc ou doxygene), ni d'un manuel d'utilisation.

Nous présentons les différentes composantes du logiciel avec à chaque fois comment l'utiliser. Les présentations ne sont bien évidemment pas exhaustives ; nous nous contentons d'indiquer les attributs et méthodes utiles à la compréhension.

Nous présentons le corps de la machine puis, l'implantation des constructions ensuite, les outils pour les automates à deux compteurs et, finalement, les interfaces graphiques.

A.1 Corps de la machine et entrées / sorties

Toutes ces classes sont dans le package `fr.ensLyon.jdurand.SigMach`.

Les machines simulées étant rationnelles, il nous a fallu faire une classe `Rational` (*c.f.* Fig. A.1) pour manipuler des rationnels de taille quelconque. Cette classe s'appuie sur la classe `java.math.BigInteger` qui permet de manipuler des entiers de taille quelconque. Toutes les opérations de bases sur les rationnels y sont présentes et exactes.

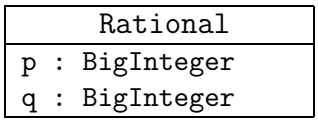


FIG. A.1 – Classe pour les rationnels.

A.1.1 Méta-signaux et signaux

La classe `Kind` correspond aux méta-signaux, en plus de la vitesse elle contient divers identificateurs et informations pour l'affichage (écran ou sortie postscritpt).

La classe `Signal` correspond aux signaux, en plus des dates de naissance et de mort et, de la position à la naissance, elle contient diverses méthodes pour connaître la position du signal à différents instants, la durée de vie,...

Ces deux classes ainsi que le lien d'agrégation qui les unit sont décrits sur la Fig. A.2.

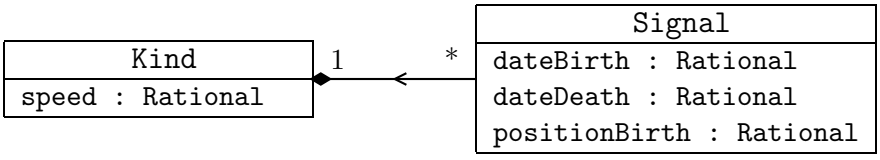


FIG. A.2 – Classes pour les méta-signaux et signaux

A.1.2 Règles et collisions

Les règles sont stockées dans un tableau associatif (`java.util.TreeMap`, le tableau des méta-signaux entrants est associé au tableau des méta-signaux sortants) dans les machines (*c.f.* Sous-sect. suivante). Elles ne sont pas dans une classe particulière ; bien que cela ait un sens conceptuellement, le besoin ne s'en est pas fait sentir et cela simplifie le programme.

Les collisions forment une classe (*c.f.* Fig. A.3). Les instances ne sont utilisées que pour prévoir les collisions et sont donc mises à jour au fur et à mesure que les collisions ont lieu. Les diagrammes espace-temps ne sont pas stockés sous forme de collisions mais uniquement par la liste des signaux présents. Chaque collision contient sa date et sa position (spatiale) ainsi que la liste des signaux entrants. Les signaux sortants sont engendrés lorsque la collision a lieu. En effet, cette classe est utilisée pour enregistrer les collisions possibles.

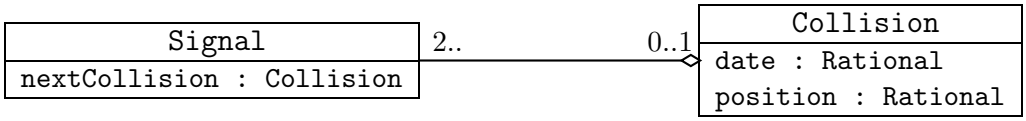


FIG. A.3 – Classe pour les collisions.

A.1.3 Machine

Les machines sont scindées en deux classes : le cœur de la machine d'un côté et, sa manipulation, modification et interrogation de l'autre. La première, `MachineCore`, contient toutes les définitions et les mécanismes de mise à jour. La seconde, `Machine`, hérite de la première et offre en plus toutes les possibilité de changer les définitions, la configuration actuelle, d'obtenir des informations et d'afficher (mode texte). Cette distinction des rôles nous a été très utile pour la mise au point et l'est pour la maintenance et l'évolution.

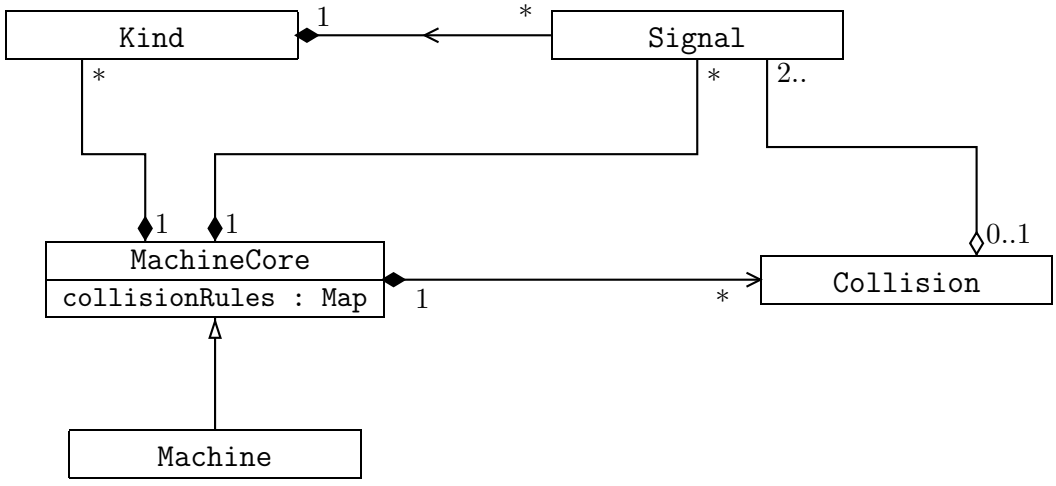


FIG. A.4 – Classes pour les machines.

La machine maintient à jour une liste des signaux encore vivants et un tas, le tas des collisions prévues. Lorsqu'une collision a lieu, la liste et le tas sont mis à jour.

A.1.4 Formats d'entrées / sorties

Codage XML pour les machines et configurations

Les machines sont codées en XML suivant un DTD (Document Type Definition). Le DTD et un exemple de fichier XML sont donnés sur la Fig. A.5.

Il est à noter que la machine doit contenir au plus une configuration et qu'il n'est pas possible de mettre une configuration seule (pour l'instant, il n'y a pas de référence à un fichier externe pour la définition de la machine).

Deux classes servent à lire et écrire dans ce format : `XMLInput` et `XMLOutput`.

```
<?xml version='1.0' encoding='utf-8'?>

<!-- declarations for signal machine -->
<!ELEMENT signalMachine (kind+,rule+,start*)>

<!-- declarations for Kind -->
<!ELEMENT kind (color?)>
<!ATTLIST kind
            id      ID    #REQUIRED
            speed   CDATA    #REQUIRED
>

<!ELEMENT color EMPTY>
<!ATTLIST color
            rgb      CDATA    #IMPLIED
            name     CDATA    #IMPLIED
>

<!-- declarations for rule -->
<!ELEMENT rule (in+,out*)>
<!ELEMENT in EMPTY>
<!ATTLIST in
            id      IDREF    #REQUIRED
>
<!ELEMENT out EMPTY>
<!ATTLIST out
            id      IDREF    #REQUIRED
>

<!-- declarations for initial configuration -->
<!ELEMENT start EMPTY>
<!ATTLIST start
            id    IDREF    #REQUIRED
            pos   CDATA    #REQUIRED
>
```

(a) DTD

```
<?xml version="1.0"?>
<signalMachine>

<kind id="bleuD1" speed="2/1"><color rgb="bleu"/></kind>
<kind id="bleuD2" speed="-1/1"><color rgb="bleu"/></kind>
<kind id="bleuG1" speed="-2/1"><color rgb="bleu"/></kind>
<kind id="bleuG2" speed="1/1"><color rgb="bleu"/></kind>
<kind id="noirD" speed="1/1"><color rgb="noir"/></kind>
<kind id="noirG" speed="-1/1"><color rgb="noir"/></kind>
<kind id="vertD" speed="1/1"><color rgb="vert"/></kind>
<kind id="vertG" speed="-1/1"><color rgb="vert"/></kind>

<rule>
<in id="bleuG1" /><in id="noirG" />
<out id="noirG" /><out id="bleuG2" />
</rule>

<rule>
<in id="noirG" /><in id="bleuG2" />
<out id="bleuG1" /><out id="noirG" /><out id="vertD" />
</rule>

<rule>
<in id="bleuD2" /><in id="noirD" />
<out id="vertG" /><out id="noirD" /><out id="bleuD1" />
</rule>

<rule>
<in id="noirD" /><in id="bleuD1" />
<out id="bleuD2" /><out id="noirD" />
</rule>

<start id="noirG" pos="0/1" />
<start id="bleuG1" pos="1/1" />
<start id="noirG" pos="2/1" />
<start id="noirD" pos="3/1" />
<start id="bleuD1" pos="4/1" />
<start id="noirD" pos="5/1" />

</signalMachine>
```

(b) Exemple

FIG. A.5 – DTD et exemple XML.

Exportation des diagrammes en postscritpt

Cela se fait grâce à la classe `DriverPS`. Une instance doit être associée à une instance de `Machine` ; après il n'y a plus qu'à paramétrer et à demander l'exportation d'un diagramme ou d'une règle en postscritpt.

A.1.5 Paramétrage pour les règles non définies

Il peut être fastidieux de définir toutes les règles possibles. L'interface `IFUndefinedRule` est donc proposée pour servir de paramétrage à `Machinecore` et définir les actions à prendre quand il n'y a pas de règle définie pour la collision (*i.e.* les signaux entrants ne sont pas une clé pour le tableau associatif contenant toutes les règles). Il est possible de programmer n'importe quelle classe implantant l'interface et de l'affecter à `Machinecore`. Une classe par

défaut est fournie, elle considère toutes les collisions non définies comme blanches ; elle peut être paramétrée pour afficher un message à chaque fois ou pour ne pas mettre de signaux en sortie.

Ces règles ne sont pas sauvegardées car ce sont des règles implicites. L'utilisateur doit faire le choix entre une liste exhaustive (qui augmente la taille des fichiers et peut ralentir la machine) et une définition implicite (*i.e.* la méthode `setUndefinedRule` « calcule » alors l'ensemble des méta-signaux sortants).

A.1.6 Utilisation en ligne de commande

La classe `RunXML` sert à lancer en ligne de commande une machine contenue dans un fichier, lui faire un certain nombre d'itérations et faire exporter le résultat en postscritpt.

A.2 Constructions géométriques

Toutes les constructions géométriques ont été programmées. Les algorithmes sont dans les chapitres correspondant. Nous nous bornons donc à présenter la hiérarchie des classes (Fig. A.6) qui reflète les dépendances des constructions et, à expliciter le nom des classes.

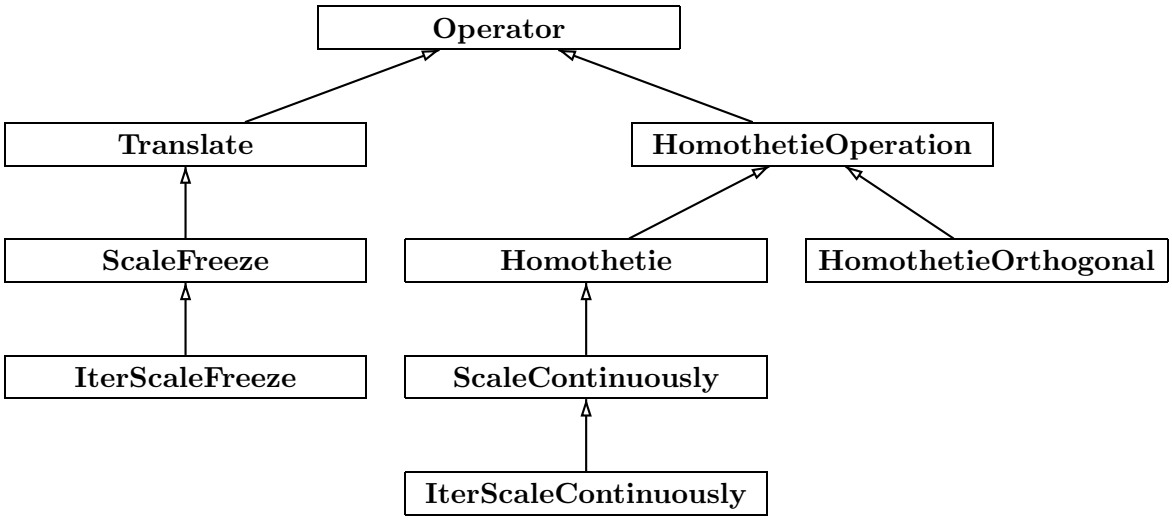


FIG. A.6 – Classes pour les opérations géométriques.

La classe `Operator` regroupe toutes les fonctionnalités communes à toutes les opérations (*e.g.* trouver un nouveau suffixe, tester la présence d'une vitesse, engendrer des signaux échangeant entre deux suffixes –toggle).

La classe `Translate` permet d'engendrer n'importe quelle translation (Sect. 6.1).

La classe `ScaleFreeze` produit une contraction simple avec gel (Sect. 6.2).

La classe `IterScaleFreeze` produit une contraction itérée avec gel (Sect. 6.3).

La classe `HomothetieOperation` regroupe des opérations de base pour les homothéties.

La classe `HomothetieOrthogonal` fait une homothétie de direction orthogonale.

La classe `Homothetie` fait une homothétie quelconque (Sect. 7.1).

La classe `ScaleContinuously` produit une contraction continue pour une configuration bornée (Sect. 7.4).

La classe `IterScaleContinuously` produit une contraction continue itérée pour une configuration bornée (Sect. 7.5).

A.3 Automates à deux compteurs

La simulation des automates à deux compteurs ne faisant pas à proprement parler partie des machines à signaux, nous avons regroupé les classes correspondantes dans un sous-package à part : `fr.ensLyon.jdurand.SigMach.TwoCounter`.

Nous nous contentons de lister les principales classes du package, leurs implantations ne posant pas de difficultés.

La classe `TwoCounter` sert à stocker la définition d'un automate à deux compteurs. Les classes `TwoCounterReader` et `TwoCounterPrinter` servent à lire et écrire des automates à deux compteurs enregistrés sous forme de programme. La classe `TwoCounterLatexWriter` permet d'en exporter une version LaTeX(comme sur la partie droite de la Fig. 4.3).

La classe `TwoCounterMachineWriter` fait la traduction en machines à signaux. Les classes `TwoCounterIterContinuosly` et `TwoCounterAllRun` servent à appliquer directement les itérations continues et le lancement d'un diagramme où l'on fait toutes les itérations (comme celle de la Fig. 10.5). La seconde classe hérite de la première qui est elle-même une sous-classe de `TwoCounterMachineWriter`.

La classe `RunTwoCounterReader` sert à lancer, en ligne de commande, la simulation d'un automate lu dans un fichier par une machine et, à faire engendrer le diagramme en postscritpt.

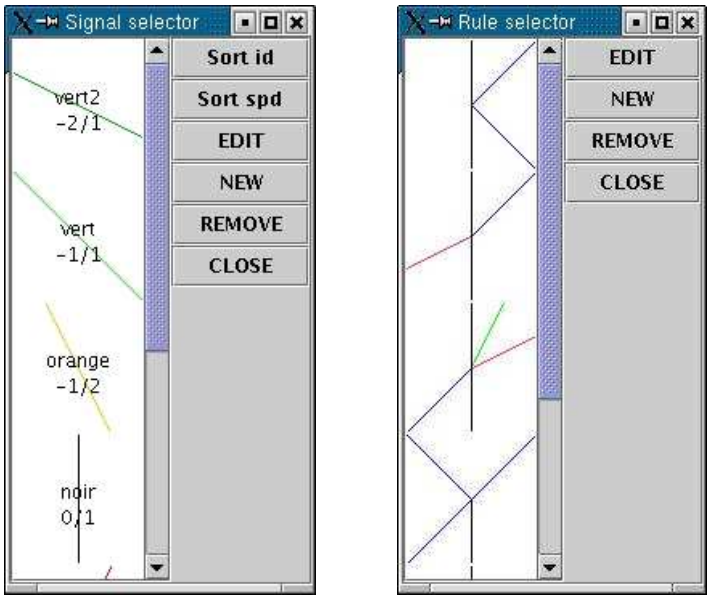
A.4 Interface graphique

Nous ne rentrons pas dans l'architecture logicielle des interfaces graphiques. Elle ne présente pas de difficulté ; mais sa présentation nécessiterait une bonne connaissance de la bibliothèque `swing` de Java.

Toutes les classes sont regroupées dans le package `fr.ensLyon.jdurand.SigMach.Swing`. Les éléments de la machine sont *enveloppés* dans des classes qui servent d'intermédiaires entre la machine et l'interface. Des *drag and drop* sont implantés entre les différentes composantes.

Il y a deux sélecteurs, un pour les `Kind` et un autre pour les règles de collisions. Ils sont visibles sur la Fig. A.7.

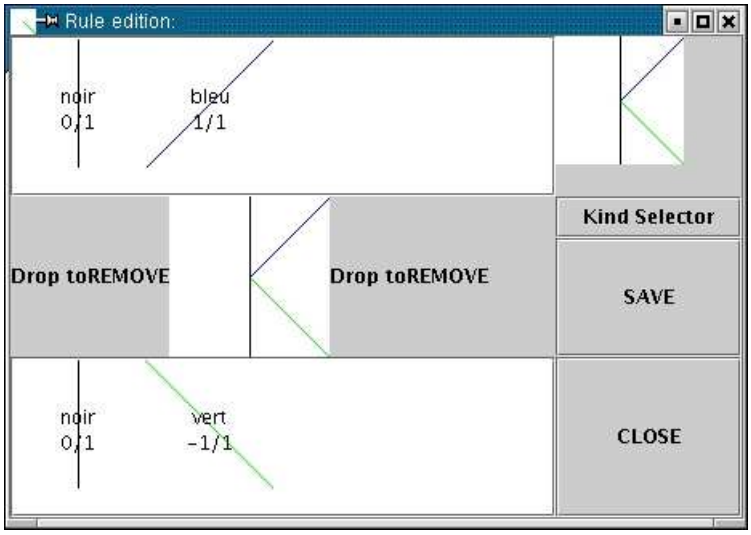
Il est possible d'éditer les signaux comme les règles avec les interfaces de la Fig. A.8. Tout ce qui les définit est modifiable.



(a) de méta-signal (b) de règle

FIG. A.7 – Sélecteurs.

(a) de méta-signal



(b) de règle

FIG. A.8 – Éditeurs.

Annexe B

Singularités non isolées

Dans ce Chapitre, nous regroupons quelques exemples de singularités non isolées et montrons que l'on peut souhaiter y donner un sens. En particulier, nous nous intéressons aux versions continues des exemples de synchronisation de fusiliers tels que nous les avons vus dans l'introduction (Sect. 1.1).

Synchronisation

Nous abordons les deux exemples de l'introduction pour résoudre ce problème. Nous montrons comment le premier exemple reste au continu sans problème si ce n'est le traitement de singularités non isolées, mais il n'y en a que deux types. Par contre pour l'autre exemple, il faudrait une infinité de signaux (déjà présents au début du calcul qui plus est) ; nous expliquons pourquoi cet exemple ne nous semble pas pouvoir être implanté « naturellement » sur une machine à signaux.

Les figures originelles sont « redressées » pour bien montrer la correspondance avec nos diagrammes.

Si l'on prend la solution de Goto [Got66], on obtient le diagramme espace-temps de la Fig. B.1 où l'on voit bien l'arbre binaire infini bleu. Il y a accumulation sur tout un segment du diagramme espace-temps.

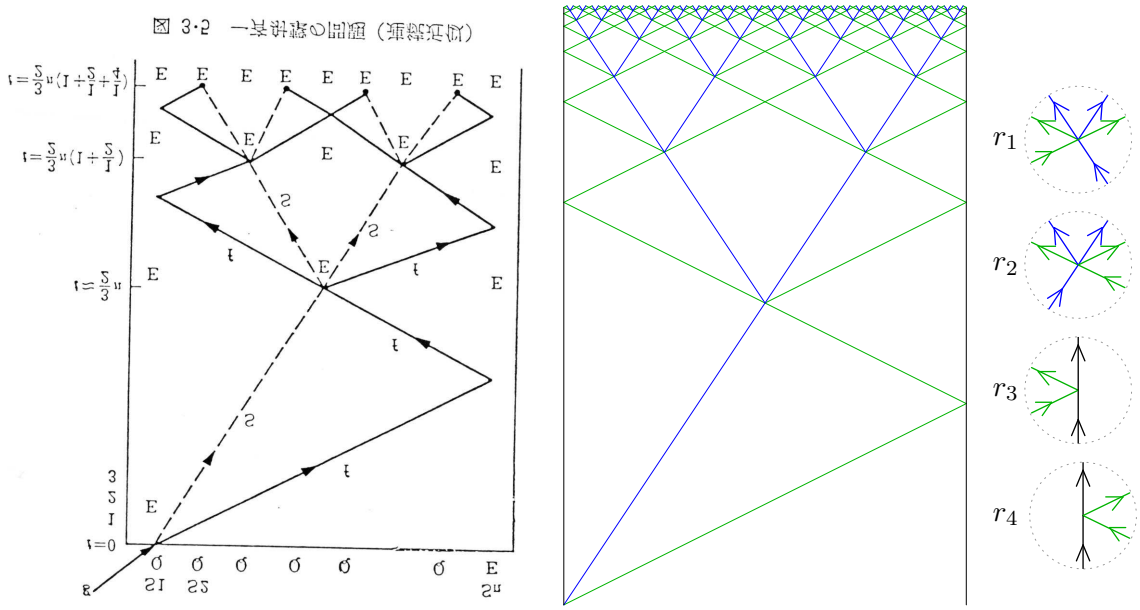
Regardons les valeurs qui s'accumulent avant, en suivant la méthodologie de la Sect. 9.2. Il n'y a ni signal incident ni méta-signal accumulé. Par contre, des règles s'accumulent, si (x,t) est à l'intérieur du segment, alors

$$R_1^-(x,t) = \{r_1, r_2\} \ ,$$

par contre, si (x,t) est l'une des deux extrémités,

$$R_1^-(x,t) = \{r_1, r_2, r_3, r_4\} \ .$$

Nous pourrions là aussi définir des règles et des voisinages ; mais les définitions des contextes devront être plus poussées car on doit ajouter des singularités dans le voisinage (dans la partie concomitante). Mais cela n'est pas tout, si l'on note σ, alors σ doit aussi être dans la partie concomitante de son propre contexte. Nous ne sommes plus dans un contexte de définition, mais bien dans celui de la correction locale.



(a) Version originelle avant discrétisation [Got66, Fig. 3]

(b) Version machine à signaux

FIG. B.1 – Algorithme de GOTO et réalisation en tant que machine à signaux.

La transcription de l'algorithme de [VMP70] (Fig. 1.6) est beaucoup moins aisée. En effet, celui-ci utilise un nombre non borné de signaux au premier abord. En cherchant plus, on s'aperçoit que tous ces signaux sont identiques ; ils se déplacent progressivement en fonction du mouvement des uns et des autres en s'aidant de la granularité de l'espace et du temps. Nous pouvons mimer ce genre de choses, mais attention, la discrétisation doit être adaptée, mais là nous amenons du discret dans le continu alors que l'on souhaiterait quelque chose de purement continu. Néanmoins, nous illustrons cet engendrement de signaux discrets dans un cadre continu dans la Fig. B.2.

Nous n'avons pas regardé plus avant cet algorithme car nous ne pensons pas qu'il soit possible de l'implanter à moins d'imposer une relation (type rapport entier ou puissance de 2) entre la distance des signaux noirs servant de bordure et la distance des deux signaux bleus. De plus il est probable que cette discrétisation provoque une synchronisation discrète ou une impossibilité de « descendre » en dessous d'une certaine taille liée à la distance des deux bleus (ou alors il faut arriver à réduire cette distance en cours de route).

Signaux horizontaux

Si l'on cherche à continuer le diagramme de la Fig. B.1, on doit définir deux contextes (pour les extrémités et l'intérieur du segment). Aucun des points de l'intérieur ne peut donc être distingué.

On peut proposer beaucoup de continuations :

– ne garder que les points extrémités, et ne garder que deux signaux parallèles ;

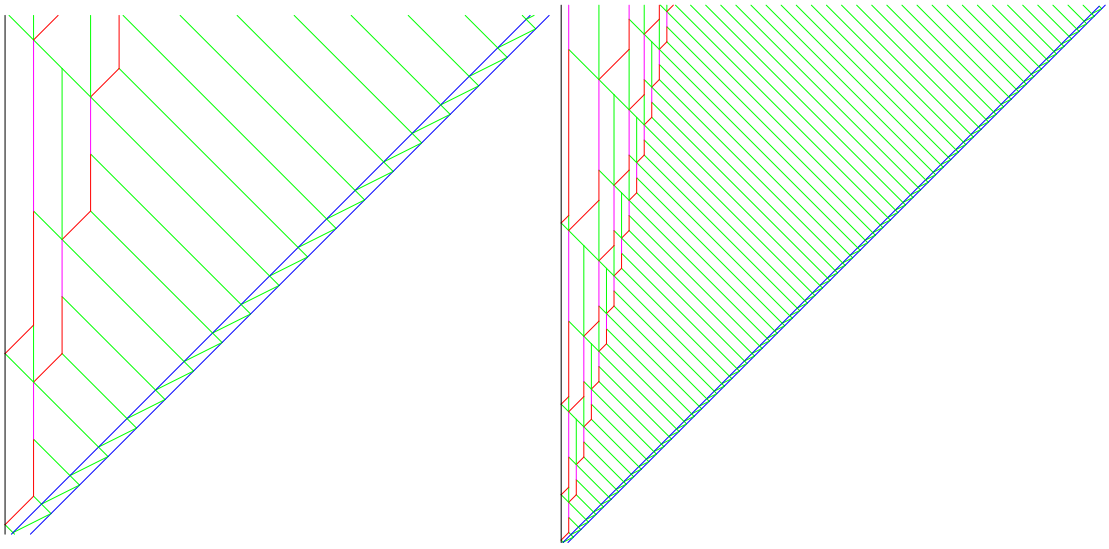


FIG. B.2 – Algorithme pour engendrer des « signaux discrets » de toutes pentes.

- imposer la présence (postérieurement) de différentes règles et signaux, de manière à imposer une « configuration limite initiale », que l'on pourrait « forcer » à l'aide d'autres signaux qui seraient incidents sur le côté ;
- demander à avoir une singularité comme seule valeur d'accumulation postérieure, cette singularité n'étant valide que pour un contexte où elle est la seule valeur accumulée (⊘ compris).

Dans le premier cas, postérieurement, il n'y a plus que des signaux en sortie, nous sommes dans un contexte déterministe.

Dans le second cas, le contexte est fortement déterministe.

Dans le troisième cas, on peut dire qu'il y a création d'un signal de dimension 1 (les signaux étant de dimension 0).

On peut alors imaginer des croisements, des signaux de dimensions supérieures (si l'on travaille en dimensions supérieures).

On peut également se demander si l'on n'a pas, en quelque sorte, des signaux de vitesse infinie, *i.e.* horizontaux.

Attention, les ensembles limites ne sont pas forcément des intervalles, ils peuvent aussi être fractals comme le montre la Fig. B.3 où cet ensemble est un Cantor.

Déformations

Toutes les déformations étudiées dans la première partie s'appliquent bien entendu à ces diagrammes. Mais cette fois, il faut aussi engendrer des singularités prime et des contextes primes. La trace devient une ligne brisée si l'on prend différentes homothéties. Nous n'avons pas inclus de dessins car le logiciel ne s'occupe que piètrement des accumulations. Il permet de continuer un peu le diagramme, mais cela ne peut servir que d'intuition.

En mélangeant différents effets, est-il possible d'obtenir des courbes ? (*e.g.* des paraboles ou arcs de cercles parfaits, qui seraient peut-être le pendant de ceux de DELORME *et*

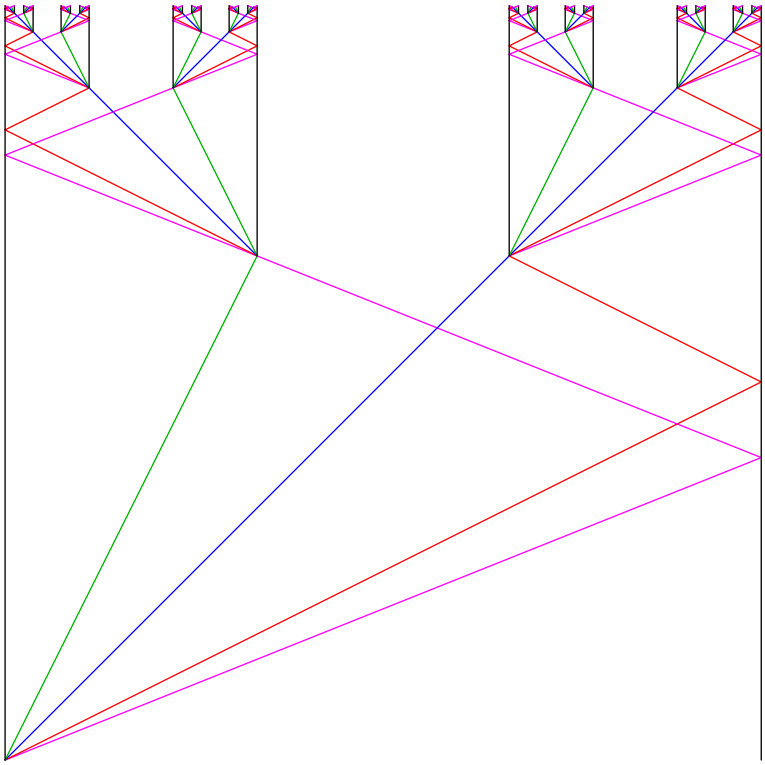


FIG. B.3 – Ensemble des points des singularités formant un Cantor.

al. [DMT99]).

Bibliographie

[AD94] R. Alur et D. L. Dill – « A Theory of timed automata », *Theoretical Computer Science* **126** (1994), no. 2, p. 183–235.

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

[AHU74] A. V. Aho, J. E. Hopcroft et J. Ullman – *The design and analysis of computer algorithms*, Addison Wesley, 1974.

[AM98] E. Asarin et O. Maler – « Achilles and the tortoise climbing up the arithmetical hierarchy », *Journal of Computer and System Sciences* **57** (1998), no. 3, p. 389–398.

[Büc76] R. Büchi – « The monadic second order theory of ω_1 », *The monadic second order theory of all countable ordinals*, LNCS, no. 328, 1976, p. 1–126.

[BCSS98] L. Blum, F. Cucker, M. Shub et S. Smale – *Complexity and real computation*, Springer, New York, 1998.

[Ben73] C. H. Bennett – « Logical reversibility of computation », *IBM Journal of Research and Development* **6** (1973), p. 525–532.

[Ben88] — , « Notes on the history of reversible computation », *IBM Journal of Research and Development* **32** (1988), no. 1, p. 16–23.

[BNR91] N. Boccara, J. Nasser et M. Roger – « Particle-like structures and interactions in spatio-temporal patterns generated by one-dimensional deterministic cellular automaton rules », *Phys. Rev. A* **44** (1991), no. 2, p. 866–875.

[Bou99a] O. Bournez – « Achilles and the Tortoise climbing up the hyper-arithmetical hierarchy », *Theoretical Computer Science* **210** (1999), no. 1, p. 21–71.

[Bou99b] — , « Complexité algorithmique des systèmes dynamiques continus et hybrides », Thèse, Laboratoire de l'Informatique du Parallélisme, École Normale Supérieure de Lyon, 1999.

[BP00] B. Bérard et C. Picaronny – « Accepting zeno words: a way towards timed refinements », *Acta Informatica* **37** (2000), no. 1, p. 45–81.

[Bra95] M. S. Branicky – « Universal computation and other capabilities of hybrid and continuous dynamical systems », *Theoretical Computer Science* **138** (1995), no. 1, p. 67–100.

[BSS89] L. Blum, M. Shub et S. Smale – « On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines », *Bulletin of the American Mathematical Society* **21** (1989), no. 1, p. 1–46.

[Cho78] Y. Choueka – « Finite automata, definable sets, and regular expressions over ω^n-tapes », *Journal of Computer and System Sciences* **17** (1978), no. 1, p. 81–97.

[Chu36a] A. Church – « A note on the Entscheidungsproblem », *Journal of Symbolic Logic* **1** (1936), p. 40–41 and 101–102.

[Chu36b] — , « An unsolvable problem of elementary number theory », *American Journal of Mathematics* **58** (1936), p. 345–363.

[CM01] M. L. Campagnolo et C. Moore – « Upper and lower bounds on continuous-time computation », 2nd International Conference on Unconventional Models of Computation - UMC '2K (I. Antoniou, C. Calude et M. Dinneen, éds.), Springer, 2001, p. 135–153.

[CR73] S. A. Cook et R. A. Reckhow – « Time-bounded random access machines », *Journal of Computer and System Sciences* **7** (1973), p. 354–375.

[Deh93] A. Dehornoy – *Complexité et décidabilité*, Mathématiques et applications, no. 12, Springer, 1993.

[DL97] J. Durand-Lose – « Intrinsic universality of a 1-dimensional reversible cellular automaton », *STACS '97*, LNCS, no. 1200, Springer, 1997, p. 439–450.

[DM02] M. Delorme et J. Mazoyer – « Signals on cellular automata », in [Ada02], pp. 234–275, 2002.

[DMT99] M. Delorme, J. Mazoyer et L. Tougne – « Discrete parabolas and circles on 2D cellular automata », *Theoretical Computer Science* **218** (1999), no. 2, p. 347–417.

[DSW94] M. Davis, R. Signal et E. Weuber – *Computability, complexity and languages*, Academic Press, 1994.

[Ela01] S. Elaydi – « Is the world evolving discretely? », *Interdisciplinary Symposium on Complexity* (K. Nishimura, éd.), Kyoto University Press, 2001, p. 171–186.

[Fis65] P. C. Fischer – « Generation of primes by a one-dimensional real-time iterative array », *Journal of the ACM* **12** (1965), no. 3, p. 388–394.

[For03] L. Fortnow – « A physics-free introduction to the quantum computation model », *Bulletin of the EATCS* **79** (2003), p. 69–85.

[Göd31] K. Gödel – « Uber formal unentscheidbare satze der principia mathematica und verwandter systeme », *Monatsch. fur Mathematik und Physik* **38** (1931), p. 173–198.

[Got66] E. Goto – « Ōtomaton ni kansuru pazuru [Puzzles on automata] », *Jōhōkagaku eno michi [The Road to Information Science]* (T. Kitagawa, éd.), Kyoristu Shuppan Publishing Co., Tokyo, 1966, p. 67–92.

[Gri99] E. R. Griffor (éd.) – *Handbook of computability theory*, Studies in Logic and the Foundations of Mathematics, no. 140, Elsevier, 1999.

[Gru97] J. Gruska – *Foundations of computing*, International Thompson Publishing, 1997.

[Gru99] — , *Quantum computing*, McGraw-Hill, London, 1999.

[Hai03] E. Hainry – *Fonctions réelles calculables et fonctions* $\mathbb{R}$*-récursives*, DÉa, ÉNS Lyon, 2003.

[Hir01] M. Hirvensalo – *Quantum computing*, Natural Computing, Springer, 2001.

[HSC01] W. Hordijk, C. R. Shalizi et J. P. Crutchfield – « An upper bound on the products of particle interactions in cellular automata », *Physica D* **154** (2001), p. 240–258.

[HU79] J. Hopcroft et J. Ullman – *Introduction to automata theory, languages, and computation*, Addison-Wesley, 1979.

[IAMI02] C. Iwamoto, T. Andou, K. Morita et K. Imai – « Computational complexity in the hyperbolic plane », *Symposium on Mathematical Foundations of Computer Science (MFCS '02)*, LNCS, no. 2420, 2002, p. 365–374.

[JS90] G. Jacopini et G. Sontacchi – « Reversible parallel computation: an evolving space-model », *Theoretical Computer Science* **73** (1990), no. 1, p. 1–46.

[JSS02] M. H. Jakubowsky, K. Steiglitz et R. Squier – « Computing with solitons: A review and prospectus », in [Ada02], pp. 277–297, 2002.

[Kar90] J. Kari – « Reversibility of 2D cellular automata is undecidable », *Physica D* **45** (1990), p. 379–385.

[Kle36a] S. C. Kleene – « General recursive functions of natural numbers », *Mathematische Annalen* **112** (1936), p. 727–742.

[Kle36b] — , « l-definability and recursiveness », *Duke Mathematical Journal* **2** (1936), p. 340–353.

[Laf02] G. Lafitte – « Calculs et infinis », Thèse, Laboratoire de l'Informatique du Parallélisme, École Normale Supérieure de Lyon, 2002.

[Lec63] Y. Lecerf – « Machines de Turing réversibles. Récursive insolubilité en $n \in \mathbb{N}$ de l'équation $u = \theta^n u$, où θ est un isomorphisme de codes », *Comptes rendus des séances de l'académie des sciences* **257** (1963), p. 2597–2600.

[LN90] K. Lindgren et M. G. Nordahl – « Universal computation in simple one-dimensional cellular automata », *Complex Systems* **4** (1990), p. 299–318.

[LP88] H. Lewis et C. Papadimitriou – *Elements of the theory of computation*, Prentice Hall, 1988.

[Maz96a] J. Mazoyer – « Computations on one dimensional cellular automata », *Annals of Mathematics and Artificial Intelligence* **16** (1996), p. 285–309.

[Maz96b] — , « On optimal solutions to the Firing squad synchronization problem », *Theoretical Computer Science* **168** (1996), no. 2, p. 367–404.

[Min67] M. Minsky – *Finite and infinite machines*, Prentice Hall, 1967.

[MM99] M. Margenstern et K. Morita – « A Polynomial Solution for 3-SAT in the Space of Cellular Automata in the Hyperbolic Plane », *Journal of Universal Computer Science* **5** (1999), no. 9, p. 563–573.

[Moo91] C. Moore – « Generalized shifts: Unpredictability and undecidability in dynamical systems », *Nonlinearity* **4** (1991), no. 2, p. 199–230.

[Moo96] — , « Recursion theory on the reals and continuous-time computation », *Theoretical Computer Science* **162** (1996), no. 1, p. 23–44.

[Mor92a] K. Morita – « Any irreversible cellular automaton can be simulated by a reversible one having the same dimension (on finite configurations) », *Technical Report of the IEICE, Comp.* **92-45 (1992-10)** (1992), p. 55–64.

[Mor92b] — , « Computation-universality of one-dimensional one-way reversible cellular automata », *Information Processing Letters* **42** (1992), p. 325–329.

[Mor96] K. Morita – « Universality of a reversible two-counter machine », *Theoretical Computer Science* **168** (1996), no. 2, p. 303–320.

[MT99] J. Mazoyer et V. Terrier – « Signals in one-dimensional cellular automata », *Theoretical Computer Science* **217** (1999), no. 1, p. 53–80.

[MY78] M. Matchtey et P. Young – *An Introduction to the general theory of algorithms*, Theory of computation, Elsevier, 1978.

[NW01] T. J. Naughton et D. Woods – « On the computational power of a continuous-space optical model of computation », *International Conference on Machines, Computations, and Universality (MCU '01)* (M. Margenstern, éd.), LNCS, vol. 2055, 2001, p. 288–299.

[Odi99] P. Odifreddi – *Classical recursion theory. volume 2*, Studies in Logic and the Foundations of Mathematics, no. 143, Elsevier, Amsterdam, 1999.

[Orp94] P. Orponen – « A survey of continuous-time computation theory », *Advances in Algorithms, languages and complexity* (D.-Z. Du et K.-J. Ko, éds.), Kluwer Academic Publisher, 1994, p. 209–224.

[PE74] M. B. Pour-El – « Abstract computability and its relation to the general purpose analog computer (some connections between logic, differential equations and analog computers) », *Transactions of the American Mathematical Society* **199** (1974), p. 1–28.

[Pos21] E. L. Post – « On a simple class of deductive systems », *Bulletin of the American Mathematical Society* **27** (1921), p. 396–397.

[Pos36] — , « Finite combinatory processes-formulation », *Journal of Symbolic Logic* **1** (1936), no. 3, p. 103–105.

[Pos46] — , « A variant of a recursively unsolvable problem », *Bulletin of the American Mathematical Society* **52** (1946), p. 264–268.

[PRS98] G. Păun, G. Rozenberg et A. Salomaa – *DNA computing*, Springer, Berlin Heidelberg, 1998.

[Rab98] A. Rabinovich – « On translations of temporal logic of actions into monadic second-order logic », *Theoretical Computer Science* **193** (1998), no. 1–2, p. 197–214.

[Rab03] — , « Automata over continuous time », *Theoretical Computer Science* **300** (2003), no. 1-3, p. 331–363.

[Rók00] Z. Róka – « The firing squad synchronization problem on cayley graphs », *Theoretical Computer Science* **244** (2000), no. 1–2, p. 243–256.

[Sha41] C. E. Shannon – « Mathematical theory of the differential analyzer », *Journal of Mathematics and Physics* **20** (1941), p. 337–354, included in [Sha93].

[Sha42] — , « The theory and design of linear differential equation machines », Tech. Report Services 20, Div. 7-311-M2, Bell Laboratories, 1942, included in [Sha93].

[Sha93] — , *Claude Elwood Shannon: collected papers*, IEEE Press, Piscataway, NJ, 1993.

[Sie96] H. T. Siegelmann – « The simple dynamics of super Turing theories », *Theoretical Computer Science* **168** (1996), no. 2, p. 461–472.

[Sip97] M. Sipser – *Introduction to the theory of computation*, PWS Publishing Co., Boston, Massachusetts, 1997.

[Siw01] P. Siwak – « Soliton-like dynamics of filtrons of cycle automata », *Inverse Problems* **17** (2001), p. 897–918.

[ŠO01] J. Šíma et P. Orponen – « Computing with continuous-time Liapunov systems », *STOC '01*, ACM Press, 2001, p. 722–731.

[Soa99] R. I. Soare – « The history and concept of computability », 1999, Chap. 1 in [Gri99].

[SS90] J.-M. Salanskis et H. Sinaceur (éds.) – *Le labyrinthe du continu*, Springer, 1990.

[SS95] H. T. Siegelmann et E. D. Sontag – « On the computational power of neural nets », *Journal of Computer and System Sciences* **50** (1995), no. 1, p. 132–150.

[TM90] T. Toffoli et N. Margolus – « Invertible cellular automata: a review », *Physica D* **45** (1990), p. 229–253.

[Tof77] T. Toffoli – « Computation and construction universality of reversible cellular automata », *Journal of Computer and System Sciences* **15** (1977), p. 213–231.

[Tra01] B. A. Trakhtenbrot – « Automata, circuits, and hybrids: Facets of continuous time », *ICALP 2001*, LNCS, vol. 2076, 2001, p. 4–23.

[Tur36] A. M. Turing – « On computable numbers, with an application to the entscheidungsproblem », *Proceedings of the London Mathematical Society* **42** (1936), no. 2, p. 230–265.

[Ula52] S. Ulam – « Random processes and transformations », *International Congress of Mathematics 1950*, no. 2, 1952, p. 264–275.

[US93] V. Uspensky et A. Semenov – *Algorithms : main ideas and applications*, Mathematics and its applications, no. 251, Kluwer, 1993.

[vL90] J. van Leeuwen (éd.) – *Handbook of theoretical computer science*, vol. A, MIT Press, 1990.

[VMP70] V. I. Varshavsky, V. B. Marakhovsky et V. A. Peschansky – « Synchronization of interacting automata », *Mathematical System Theory* **4** (1970), no. 3, p. 212–230.

[Vol99] H. Vollmer – *Introduction to circuit complexity - a uniform approach*, Texts in Theoretical Computer Science, Springer, 1999.

[Wei00] K. Weihrauch – *Introduction to computable analysis*, Texts in theoretical computer science, Springer, Berlin, 2000.

[Wol91] P. Wolper – *Introduction à la calculabilité*, InterEditions, 1991.

[Zei01] D. Zeilberger – « "Real" analysis is a degenerate case of discrete analysis », *International Conference on Difference Equations and Applications*, 2001, http://www.math.rutgers.edu/~zeilberg/mamarim/mamarimhtml/real.html.

Liste des symboles

$\mathbb{R}^2_{0\leq t}$	$\mathbb{R} \times \mathbb{R}_{0\leq t}$, positions dans l'espace-temps, page 32
p	Position dans l'espace-temps, page 32
x	Position dans l'espace, page 32
t	Position dans le temps, page 32
μ	Méta-signal, page 32
ι	Information véhiculée par un méta-signal, page 32
ν	Vitesse de toutes les instances d'un méta-signal, page 32
ρ	Règle de collision, page 33
μ^-	Méta-signal entrant en collision, page 33
μ^+	Méta-signal sortant d'une collision, page 33
π	Collision, page 34
c_t	Configuration d'une machine, page 35
c_0	Configuration initiale d'une machine, page 35
$\mathcal{M}$	Machine à signaux, page 35
Γ	Calcul d'une machine à signaux sur une entrée, page 35
$\oslash$	Rien à la position donnée, page 36
$\mathbb{D}$	Diagrammes espace-temps, page 36
$\nu^{\mathcal{M}}_{min}$	Plus petite vitesse présente dans $\mathcal{M}$, page 36
$\nu^{\mathcal{M}}_{max}$	Plus grande vitesse présente dans $\mathcal{M}$, page 36
$\nu^{\mathcal{M}}_{abs}$	Plus grande vitesse absolue présente dans $\mathcal{M}$, page 36
$\mathcal{C}_+(x,t)$	Cône d'influences émises en (x,t), page 36
$\mathcal{C}_-(x,t)$	Cône d'influences reçues en (x,t), page 36
$\perp$	Indéfini comme résultat d'une fonction partielle, page 40
$\mathcal{R}$	Ensemble des ensembles récursifs, page 42
$\mathcal{RE}$	Ensemble des ensembles récursivement énumérables, page 42
co$-\mathcal{RE}$	Ensemble des ensembles co-récursivement énumérables, page 42
`A`	Premier compteur d'un automate à deux compteurs, page 42
`B`	Second compteur d'un automate à deux compteurs, page 42
$\mathcal{A}$	Automate à deux compteurs, page 42
a	Valeur de `A`, page 42
b	Valeur de `B`, page 42

β	Vitesse de signal toggle, page 62
θ	Vitesse des signaux en translation, page 62
α	Direction d'une homothétie, page 77
k	Coefficient d'une homothétie, page 77
$\Diamond$	Indéfinie, comme valeur dans un diagramme, page 93
$\mathbb{D}_{\mathcal{M}}$	Diagramme espace-temps tel que défini pour une machine $\mathcal{M}$ dans le Chap. 3, page 93
$\mathbb{D}_{\mathcal{T}}$	Diagramme espace-temps tel que défini dans le Chap. 8, page 93
$\mathcal{O}_{0\leq t}$	Ensemble des ouverts de $\mathbb{R}^2_{0\leq t}$, page 94
$\mathcal{C}^*_-(x,t)$	Cône épointé d'influences reçues en (x,t), page 97
$det(\mathbb{D})$	Partie déterministe du diagramme $\mathbb{D}$, page 98
$\mathcal{T}(x_l,x_r,t_d)$	Triangle de base (x_l,t_d) et (x_r,t_d), page 98
$det^*(\mathbb{D})$	Partie déterministe du diagramme $\mathbb{D}$ par les cônes épointés, page 100
$co{-}det(\mathbb{D})$	Complémentaire de la partie déterministe du diagramme $\mathbb{D}$, page 100
$V_1(x_0,t_0)$	Valeurs accumulées en (x_0,t_0), page 104
$\mathbb{R}^2_{0\leq t<t_0}$	$\mathbb{R}\times[0,t_0[$, page 106
$\mathbb{R}^2_{t_0<t}$	$\mathbb{R}\times]t_0,\infty[$, page 106
$V_1^-(x_0,t_0)$	Valeurs accumulées antérieures en (x_0,t_0), page 106
$V_1^=(x_0,t_0)$	Valeurs accumulées concomitantes en (x_0,t_0), page 106
$V_1^+(x_0,t_0)$	Valeurs accumulées postérieures en (x_0,t_0), page 106
$M_1^-(x_0,t_0)$	Méta-signaux accumulés antérieurs en (x_0,t_0), page 107
$M_1^=(x_0,t_0)$	Méta-signaux accumulés concomitants en (x_0,t_0), page 107
$M_1^+(x_0,t_0)$	Méta-signaux accumulés postérieurs en (x_0,t_0), page 107
$R_1^-(x_0,t_0)$	Règles accumulées antérieures en (x_0,t_0), page 107
$R_1^=(x_0,t_0)$	Règles accumulées concomitantes en (x_0,t_0), page 107
$R_1^+(x_0,t_0)$	Règles accumulées postérieures en (x_0,t_0), page 107

Index