N^o d'ordre : 109
N^o bibliothèque : 99ENSL0109

ÉCOLE NORMALE SUPÉRIEURE DE LYON
Laboratoire de l'Informatique du Parallélisme

THÈSE

pour obtenir le grade de
Docteur de l'École Normale Supérieure de Lyon
spécialité : Informatique
au titre de la formation doctorale de : Informatique de Lyon

Complexité algorithmique des systèmes dynamiques continus et hybrides

par Olivier BOURNEZ

Soutenue le 18 janvier 1999

Après avis de : Eugène ASARIN
Max DAUCHET

Devant la commission d'examen formée de :

Eugène ASARIN
Vincent BLONDEL
Michel COSNARD
Max DAUCHET
Pascal KOIRAN
Oded MALER

Résumé

Cette thèse présente une étude de la complexité algorithmique de la vérification automatique de propriétés pour les systèmes dynamiques continus et hybrides.

Dans un premier temps nous étudions la décidabilité de la vérification automatique de propriétés des systèmes dynamiques à espace continu : nous prouvons tout d'abord l'indécidabilité du problème de la stabilité des systèmes dynamiques linéaires seuillés, puis nous étudions la frontière entre décidabilité et indécidabilité pour les systèmes de basses dimensions en discutant l'existence d'un algorithme pour décider la mortalité des matrices deux par deux.

Dans un second temps, nous étudions la représentation des polyèdres par leurs sommets : nous proposons plusieurs représentations pour les polyèdres orthogonaux et prouvons que ces représentations permettent une réalisation efficace des algorithmes classiques de vérification automatique. Nous généralisons ensuite ces représentations aux polyèdres manipulés par les algorithmes de vérification de propriétés des automates temporisés.

Dans un troisième temps, nous présentons une caractérisation complète de la puissance de calcul d'une classe particulière de systèmes dynamiques à espace et à temps continus : nous prouvons que la puissance de calcul des systèmes dynamiques définis par une équation différentielle constante par morceaux se relie aux classes de langages de la hiérarchie hyperarithmétique.

Remerciements

Je tiens tout d'abord à remercier les différents membres de mon jury de thèse :

- Eugène Asarin, pour avoir accepté d'être rapporteur de cette thèse et faire le déplacement depuis Moscou pour assister à ma soutenance de thèse. Je le remercie vivement de l'intérêt qu'il a manifesté pour ce travail dès son début.
- Vincent Blondel, qui m'a fait l'honneur de présider mon jury de thèse. Je le remercie en outre pour toutes les remarques de lecteur avisé qu'il m'a fait parvenir sur ce manuscrit.
- Michel Cosnard, qui est à l'origine de ce travail. Michel, dès le début, a su m'encourager et me guider, tout en me laissant libre d'orienter mes travaux vers les sujets qui m'intéressaient. Malgré des charges administratives grandissantes, il a su garder une grande disponibilité. Ce fut pour moi un grand plaisir que de travailler avec lui.
- Max Dauchet, pour avoir accepté de passer les fêtes de Noël à rapporter sur cette thèse. Je le remercie ici pour tout l'intérêt qu'il a manifesté pour ce travail.
- Pascal Koiran, qui a co-encadré ce travail, en particulier après le départ physique de Michel. Ses nombreuses remarques et suggestions, fruits de son grand dynamisme scientifique, m'ont été de précieux conseils. Toujours prêt à répondre à toutes mes questions, ce fut pour moi un grand plaisir que de travailler avec lui.
- Oded Maler, qui m'a fait l'honneur d'encadrer mon séjour en tant que scientifique du contingent au laboratoire VERIMAG à Grenoble. Je le remercie sincèrement d'avoir contribué à ce que Lieutenant-colonel de Monicault du 22ème RI aille jusqu'à écrire que "j'avais rendu de bons services pendant mon séjour sous les drapeaux".

L'occasion m'est donnée ici de remercier Joseph Sifakis, et les services de la DGA, qui ont permis que je puisse effectuer mon service national en tant que scientifique du contingent au laboratoire VERIMAG.

Je tiens maintenant à remercier le personnel et les membres du LIP et de VERIMAG grâce à qui ce travail s'est passé dans de très bonnes et agréables conditions. Mes pensées vont tout d'abord en particulier vers les différentes secrétaires de ces deux laboratoires pour leur gentillesse et leur efficacité.

Je remercie en outre Hélène Paugam-Moisy et Jacques Mazoyer, et les participants réguliers des groupes de travail CAP, Complexité algébrique et MC2 pour avoir eu la patience d'écouter chacun de mes exposés et la générosité de me faire partager leurs idées.

Pour l'ambiance agréable de travail qui régnait à la fois au LIP et à VERIMAG, je tiens, dans l'ordre inalphabétique, à remercier particulièrement les personnes suivantes : Thierry, pour ses discussions et ses illustrations graphiques de mon travail de thèse, Bernard, grâce à qui nous avions la vérité au fond de l'armoire, Laure, qui faisait partie d'une équipe qui avait poussé la perfidie jusqu'à choisir un nom anagramme, Peter pour la casquette de l'OM, Ahmed et sa bonne humeur constante, Rym, grâce à qui je prends le train sans m'inquiéter des collisions, Bruno et Hervé, qui à l'aide du CRI, me donnent l'illusion que j'ai un bureau entier pour moi tout seul, Sandrine, qui a eu la politesse de ne jamais me faire remarquer que je faisais plus que dix allées et retours par jour dans son bureau, Richard et sa modestie légendaire, Arnaud, qui a toujours su faire le poids face aux conseils de directions, Sylvain, squatteur invétéré mais toujours prêt à m'offrir un chocolat, Pascal, grâce à qui j'ai découvert procmail, Christian, qui a toujours la réponse à mes questions sur Unix, Linux et ses variantes, Bertrand, toujours prêt à animer une discussion, Cécile, Karine, Thao, Conrado, Hassen pour l'ambiance chaleureuse du midi du restaurant universitaire, Emmanuel, qui m'a sauvé la vie à plusieurs reprises et sans qui mes transparents de thèse auraient été bien pâles, Frédéric, boursicoteur exilé dans un bureau désert du LR5, Ahmed, qui a supporté sans broncher une discussion sur son pays natal après ma soutenance de thèse, et enfin Blaise pour les week-ends, Grégory pour les longues soirées, et Pascal pour les matinées de la dernière ligne droite.

Je finirai en remerciant ma famille au sens large, en particulier ma maman pour le pot de thèse, et Johanne qui a supporté mes délires, même très tardifs, sur les trous d'espace/temps et sur les pyramides en dimension cinq.

Ce document est dédié à tous ceux qui l'on lu ou qui le liront.

Table des matières

Table des figures

Introduction

On peut distinguer deux grands types de systèmes dynamiques : les systèmes à temps discret et les systèmes à temps continu [53]. Les trajectoires des premiers correspondent aux itérations d'une fonction, les trajectoires des seconds correspondent aux solutions d'une équation différentielle. On dit qu'un système est *hybride* lorsqu'il mélange à la fois une dynamique continue et des transitions discrètes.

Ce document présente une étude de la complexité algorithmique des systèmes dynamiques qui se situe à la frontière de deux grandes communautés scientifiques : la communauté de la vérification et la communauté des modèles réels.

Les membres de la communauté de la vérification cherchent à proposer des méthodes qui permettent de garantir qu'un système donné vérifie une propriété donnée. Puisque les systèmes dynamiques ou hybrides permettent de modéliser la plupart des systèmes de la vie réelle, ils interviennent naturellement dans ce contexte. Par exemple, la position/altitude d'un avion peut se modéliser par une certaine équation différentielle. Vérifier que l'avion reste à une altitude positive se ramène à vérifier que les solutions de cette équation différentielle restent dans un certain sous-ensemble de l'espace du système dynamique.

Les membres de la communauté des modèles réels cherchent, quant à eux, à étudier les propriétés calculatoires des modèles de calcul algébriques sur les réels, c'est-à-dire à caractériser la puissance des modèles de calcul où, contrairement à ce que l'on fait dans l'analyse récursive, on mesure la complexité des problèmes en termes du nombre d'opérations arithmétiques nécessaires à leur résolution, sans se préoccuper de la façon avec laquelle les réels sont représentés : voir par exemple le modèle des machines de Blum Shub et Smale [14, 59]. Dans ce contexte, les systèmes dynamiques ou hybrides peuvent être considérés comme des modèles de calcul algébriques particuliers : voir [9, 23, 44].

L'étude générale de la vérification de propriétés pour les systèmes dynamiques et hybrides a très vite mené la communauté de la vérification à découvrir que les systèmes dynamiques et hybrides peuvent simuler les machines de Turing [5, 6, 9, 63]. Presque simultanément, dans la seconde communauté, Moore prouvait que l'on peut simuler toute machine de Turing par un système dynamique à fonction de transition affine par morceaux [60, 61] et Siegelmann et Sontag découvraient que les réseaux de neurones ont une puissance de calcul supérieure aux machines

de Turing [79, 81, 82]. Du point de vue de la vérification, ces résultats sont négatifs puisqu'ils impliquent l'indécidabilité de la vérification de propriétés pour les systèmes dynamiques et hybrides [5, 6, 9, 63]. Mais, du point de vue des modèles réels, ces résultats sont positifs et surprenants, puisqu'ils prouvent que les modèles de calcul algébriques sur les réels ont une puissance supérieure à celle des machines de Turing [79, 81, 82]. Il convient donc de ce point de vue de chercher à caractériser leur puissance.

Le travail présenté dans cette thèse peut s'interpréter selon la problématique de chacune des deux communautés précédentes :

Du point de vue de la vérification automatique, cette thèse contribue à prouver que la plupart des propriétés pour les systèmes hybrides sont indécidables. Elle propose des représentations des polyèdres qui peuvent être utilisées dans les algorithmes ou les semi-algorithmes classiques de vérification. Elle prouve que la vérification automatique de propriétés pour les systèmes hybrides est un problème très difficile : la vérification de propriétés d'un système hybride défini par une équation différentielle constante par morceaux de dimension d est Σ_{ω^k}-complet si $d = 2k + 3, k \geq 0$ et Σ_{ω^k+1}-complet si $d = 2k + 4, k \geq 0$. Autrement dit, pour des systèmes très simples comme les systèmes à dérivée constante par morceaux de dimension 4, il n'y a aucun espoir de construire un algorithme de vérification, ni même aucun semi-algorithme.

Du point de vue des modèles réels, cette thèse contribue à prouver que les systèmes dynamiques à description rationnelle ont une puissance de calcul égale à celle des machines de Turing lorsqu'ils sont à temps discret, et une puissance de calcul supérieure aux machines de Turing lorsqu'ils sont à temps continu. Elle discute la puissance de calcul des systèmes de faibles dimensions. Elle caractérise la puissance de calcul d'une classe particulière de systèmes dynamiques à temps continu : elle prouve que les systèmes à dérivée constante par morceaux semi-reconnaissent précisément Σ_{ω^k} en dimension $d = 2k + 3, k \geq 0$, et Σ_{ω^k+1} en dimension $d = 2k + 4, k \geq 0$.

Nous utiliserons le plan suivant : dans les chapitres 1 et 2, nous introduisons les concepts nécessaires à la compréhension de ce document.

Ainsi, dans le chapitre 1, nous précisons ce que nous appellerons une machine de Turing, nous introduisons les hiérarchies arithmétiques et hyperarithmétiques et certaines de leurs propriétés et nous présentons l'analyse récursive et les machines de Blum Shub et Smale.

Dans le chapitre 2, nous introduisons les systèmes dynamiques à temps discret, puis les systèmes dynamiques à temps continu. Nous définissons les systèmes affines par morceaux, les systèmes linéaires commutés, les réseaux de neurones et les systèmes à dérivée constante par morceaux. Nous discutons enfin l'existence et l'unicité des trajectoires dans les systèmes à dérivée constante par morceaux.

Dans le chapitre 3, nous étudions la décidabilité de la vérification automatique de propriétés pour les systèmes dynamiques : nous rappelons qu'il est connu qu'il n'est pas possible de vérifier les propriétés *locales* des systèmes dynamiques. Nous

répondons à une question de Sontag [83] en prouvant que cela reste vrai pour les propriétés *globales* des systèmes dynamiques en prouvant qu'il n'est pas possible de décider si un système affine par morceaux ou un réseau de neurones est stable.

Dans le chapitre 4, nous étudions la frontière entre décidabilité et indécidabilité pour les systèmes de basses dimensions : nous rappelons que le problème de l'atteignabilité est indécidable pour les systèmes affines par morceaux de dimension 2, mais que l'on ne sait pas si le problème est décidable pour les systèmes de dimension 1. Nous étudions le problème de la contrôlabilité des systèmes linéaires commutés de dimension 2 : nous prouvons que ce problème est exactement le problème de la mortalité des matrices 2×2, que ce dernier problème se relie au nombre minimal de règles rendant le problème de la correspondance de Post indécidable, qu'il est indécidable dans le modèle des machines de Blum Shub et Smale, qu'il est par contre Turing-décidable pour 2 matrices 2×2, et qu'il se relie au problème de l'égalité des coefficients étudié par [48], au problème du zéro en haut dans un coin étudié par [29, 54] et au problème de l'atteignabilité pour les systèmes affines par morceaux de dimension 1. Enfin, nous prouvons que les problèmes de la stabilité sont indécidables pour les systèmes affines par morceaux continus de dimension 4 et pour les systèmes affines par morceaux discontinus de dimension 2.

Les chapitres 5 et 6, motivés par le fait que la plupart des méthodes algorithmiques de vérification manipulent des polyèdres, s'intéressent à la représentation des polyèdres orthogonaux et temporisés.

Dans le chapitre 5, nous étudions la représentation des polyèdres orthogonaux par leurs sommets. Nous proposons trois représentations : la représentation par les couleurs, la représentation par les voisinages et la représentation par les sommets extrêmes. Nous prouvons la validité de ces trois représentations pour les polyèdres convexes et non-convexes de dimension quelconque. Nous prouvons que l'on peut, lorsque la dimension d est fixée, et lorsque n est le nombre de sommets, déterminer la couleur d'un point en temps $O(n^d)$ pour la représentation par les couleurs et en temps $O(n \log n)$ pour les deux autres représentations. Enfin, nous proposons des algorithmes pour effectuer des comparaisons entre polyèdres, pour détecter les faces d'un polyèdre, ou pour réaliser des opérations booléennes entre polyèdres avec ces représentations.

Dans le chapitre 6, nous étudions la représentation des polyèdres temporisés, c'est-à-dire la représentation des polyèdres manipulés par les algorithmes de vérification sur les automates temporisés. Nous proposons deux représentations : la représentation par les simplexes de la boîte et la représentation par les simplexes du voisinage, valides pour les polyèdres convexes et non-convexes de dimension quelconque. Nous prouvons que l'on peut, lorsque la dimension d est fixée, et lorsque n est le nombre de sommets du polyèdre, déterminer la couleur d'un point en temps $O(n^d)$ pour la représentation par les simplexes de la boîte, et en temps $O(n \log n)$ pour la représentation par les simplexes du voisinage. Nous proposons ensuite des algorithmes pour réaliser des comparaisons entre polyèdres, pour réa-

liser l'opération "passage du temps", ou pour réaliser des opérations booléennes entre polyèdres avec ces représentations.

Les chapitres 7 et 8 concernent quant à eux l'étude de la puissance des modèles de calcul à temps continu : ces chapitres présentent une extension de [8] en donnant une caractérisation complète de la puissance de calcul des systèmes à dérivée constante par morceaux.

Dans le chapitre 7, grâce au paradoxe de Zénon, c'est-à-dire grâce au fait que l'on peut réaliser un nombre non-borné d'opérations en un temps fini, nous prouvons que les systèmes à dérivée constante par morceaux peuvent reconnaître des langages de la hiérarchie hyperarithmétique.

Dans le chapitre 8, nous prolongeons ces résultats en caractérisant la puissance de calcul des systèmes à dérivée constante par morceaux. Nous caractérisons leur puissance comme Σ_{ω^k} en dimension $d = 2k+3, k \geq 0$, et Σ_{ω^k+1} en dimension $d = 2k+4, k \geq 0$. Autrement dit, nous prouvons que la vérification de propriétés pour un système à dérivée constante par morceaux de dimension d est Σ_{ω^k}-complet si $d = 2k+3, k \geq 0$, et Σ_{ω^k+1}-complet si $d = 2k+4, k \geq 0$.

Enfin, nous terminons par une discussion sur les résultats obtenus.

CHAPITRE

1 Modèles de calcul

1.1 Introduction

Ce chapitre a pour objectif de rappeler les modèles proposés par la littérature pour calculer sur les entiers ou sur les réels.

Tous les modèles de calcul sur les entiers se ramènent au modèle de la machine de Turing. C'est pourquoi nous ne parlerons que de ce modèle pour les entiers. Par contre, il existe deux grandes approches pour calculer sur les réels : l'analyse récursive et les machines de Blum Shub et Smale. Nous évoquons chacune de ces approches : dans la section 1.2, nous précisons ce que nous appellerons une machine de Turing et nous introduisons les hiérarchies arithmétiques et hyper-arithmétiques. Dans la section 1.3, nous introduisons les modèles de calcul utilisés pour calculer sur les réels : l'analyse récursive et les machines de Blum Shub et Smale.

1.2 Machines de Turing

1.2.1 Machines classiques

Lorsque Σ est un alphabet fini, nous écrirons Σ^* pour l'ensemble des mots finis sur l'alphabet Σ et Σ^ω pour l'ensemble des mots infinis sur l'alphabet Σ. Dans tout le reste de ce document, nous considérerons que l'on a $\Sigma^* \subset \Sigma^\omega$, en identifiant tout mot fini $w \in \Sigma^*$ avec le mot infini $w\#^\omega$, où $\#$ désigne une lettre particulière de Σ utilisée comme délimiteur.

Lorsque $w \in \Sigma^\omega$ est un mot et k est un entier, $w[k]$ désignera la $k$ème lettre de w. ϵ désignera le mot vide. Nous supposerons fixé un codage des entiers sur l'alphabet Σ. Nous noterons par $\overline{n}$ l'écriture sur l'alphabet Σ d'un entier n. Nous noterons par $<,>$ une fonction récursive bijective de $\Sigma^* \times \Sigma^*$ vers Σ^*.

Une *machine de Turing* [40] est une machine abstraite de calcul déterministe avec un nombre fini d'états internes $Q = \{0, 1, \ldots, m-1\}$ et un nombre fini k de rubans. Chacun des rubans peut être vu comme une suite indexée par $\mathbb{Z}$ de symboles d'un alphabet fini Σ. En fonction de son état interne et des symboles sur les rubans aux index 0, la machine peut effectuer les opérations suivantes :

remplacer le symbole d'un ruban à l'index 0 par un nouveau symbole, déplacer un ruban vers la droite (chaque symbole à l'index i du ruban est dorénavant à l'index $i+1$), déplacer un ruban vers la gauche (chaque symbole à l'index i du ruban est dorénavant à l'index $i-1$) ou changer son état interne en un nouvel état interne. Les instructions d'une machine de Turing sont donc des $3k+2$-uplets de la forme $[l, s_1, b_1, D_1, \ldots, s_k, b_k, D_k, l']$, où $l \in Q$ représente l'état interne de la machine, s_j représente le symbole sur le ruban j à l'index 0, b_j le symbole à écrire sur le ruban j à la place de s_j, D_j le déplacement du ruban j à effectuer, et l' le nouvel état interne.

La donnée des rubans et d'un état interne est appelée une *configuration de la machine*. Un ruban peut être vu comme deux mots infinis : le premier mot est constitué des symboles d'index positifs ou nuls ; le second mot est constitué des symboles d'index négatifs. Par conséquent, l'espace des configurations peut être considéré comme $(\Sigma^\omega \times \Sigma^\omega)^k \times Q$.

Une configuration en laquelle aucun uplet ne s'applique est dite *terminale*. Lorsqu'une configuration n'est pas terminale, on suppose qu'un unique uplet s'applique à cette configuration, et on appelle *configuration successeur immédiate* la configuration obtenue en appliquant le uplet. On appelle *relation successeur*, et on note $\vdash^*$, la fermeture transitive de la relation successeur immédiate $\vdash$. Une configuration est dite *mortelle* si elle possède une configuration terminale comme configuration successeur.

On dit qu'une machine de Turing *accepte un mot* $w \in \Sigma^\omega$ si la configuration $(w, \epsilon, \ldots, \epsilon, m-1)$ est mortelle. On dit qu'une machine de Turing *accepte un mot* $w \in \Sigma^\omega$ *avec* $w' \in \Sigma^\omega$ *comme ruban* si la configuration $(w', \epsilon, \ldots, \epsilon, 0)$ est une configuration successeur de la configuration $(w, \epsilon, \ldots, \epsilon, m-1)$.

On dit qu'une machine de Turing *semi-reconnaît un langage* $L \subset \Sigma^*$, si L coïncide avec l'ensemble des mots $w \in \Sigma^*$ acceptés par la machine. On dit qu'un langage $L \subset \Sigma^*$ est *récursivement énumérable* s'il est semi-reconnu par une machine de Turing, et on dit qu'il est *récursif* si en outre son complément est récursivement énumérable.

On dit qu'une machine de Turing *calcule une fonction partielle* $f_M : Y_1 \times Y_2 \times \ldots \times Y_k \to \Sigma^*$, où $Y_1, Y_2, \ldots, Y_k \in \{\Sigma^\omega, \Sigma^*\}$ si pour tout $w_1 \in Y_1, \ldots, w_k \in Y_k$ tels que $f(w_1, \ldots, w_k)$ soit défini, la configuration $(f(w), \epsilon, w_2, \epsilon, \ldots, w_k, \epsilon, 0)$ est configuration successeur de la configuration $(w_1, \epsilon, w_2, \epsilon, \ldots, w_k, \epsilon, m-1)$.

On dit qu'une machine de Turing *calcule une fonction partielle* $f_M : Y_1 \times Y_2 \times \ldots \times Y_k \to \Sigma^\omega$, où $Y_1, Y_2, \ldots, Y_k \in \{\Sigma^\omega, \Sigma^*\}$ si pour tout $w_1 \in Y_1, \ldots, w_k \in Y_k$ tels que $f(w_1, \ldots, w_k) = w'$ soit défini et pour tout entier $n \in \mathbb{N}$ la configuration $(w'[n], \epsilon, w_2, \epsilon, \ldots, w_k, \epsilon, 0)$ est configuration successeur de la configuration $(\overline{n}\#w_1, \epsilon, w_2, \epsilon, \ldots, w_k, \epsilon, m-1)$.

1.2.2 Machines avec oracle

Un *oracle* est un langage $X \subset \Sigma^*$. Supposons fixé un ordre récursif sur les mots de Σ^*. Un oracle peut se coder par le mot infini

$$\rho(X) = w_1 w_2 \dots w_k \dots$$

où $w_i \in \{1, 0\}$ indique l'appartenance du $i$ième mot de Σ^* à X.

Une *machine de Turing avec oracle à k rubans* [40] est une machine de Turing à $k+1$ rubans telle que le $k+1$ème ruban soit accessible en lecture uniquement. Les instructions d'une machine de Turing avec oracle sont donc des $3k+4$-uplets de la forme $[l, s_1, b_1, D_1, \dots, s_k, b_k, D_k, s_{k+1}, D_{k+1}, l']$, où $l \in Q$ représente l'état interne de la machine, s_j représente le symbole sur le ruban j à l'index 0, b_j le symbole à écrire sur le ruban j à la place de s_j, D_j le déplacement du ruban j à effectuer, et l' le nouvel état interne.

Soit $X \subset \Sigma^*$ un langage. La machine est dite *avec X comme oracle* si le $k+1$ème ruban contient le mot infini $\rho(X)$. Lorsque l'oracle $X \subset \Sigma^*$ est fixé, une configuration de la machine peut être vue comme dans la section précédente comme un élément de $(\Sigma^\omega \times \Sigma^\omega)^k \times Q$ puisque la valeur du $k+1$ème ruban est constante. Les définitions d'acceptation, de langage semi-reconnu et de fonction calculée sont alors identiques à celles de la section précédente. On parlera de langage *X-récursivement énumérable* ou *X-récursif* pour préciser que la machine semi-reconnaît ou reconnaît le langage avec le langage X comme oracle.

On remarquera qu'une machine de Turing sans oracle est exactement une machine de Turing avec l'ensemble vide comme oracle.

Puisque les programmes des machines de Turing avec oracle sont finis, il est possible de les numéroter. M_n désignera la $n$ème machine de Turing avec oracle, W_n^X désignera le langage reconnu par la machine M_n avec X comme oracle et W_n désignera le langage reconnu par la machine M_n avec l'ensemble vide comme oracle. De même ϕ_n^X désignera la fonction de Σ^* vers Σ^* calculée par M_n avec X comme oracle et ϕ_n désignera la fonction de Σ^* vers Σ^* calculée par la machine M_n avec l'ensemble vide comme oracle.

1.2.3 Hiérarchie arithmétique

La démonstration classique (voir [40] par exemple) de l'existence d'un langage récursivement énumérable non-récursif consiste à prouver que le problème d'arrêt des machines de Turing est récursivement énumérable non-récursif. Si $\emptyset^{(1)}$ désigne ce problème, on peut alors se poser naturellement la question de savoir s'il existe un langage $\emptyset^{(1)}$-récursivement énumérable non-$\emptyset^{(1)}$-récursif. La réponse est simple : le problème de l'arrêt des machines de Turing avec oracle $\emptyset^{(1)}$ est $\emptyset^{(1)}$-récursivement énumérable non-$\emptyset^{(1)}$-récursif.

On peut alors continuer : pour tout entier k appelons $\emptyset^{(k+1)}$ le problème d'arrêt des machines de Turing avec oracle $\emptyset^{(k)}$. Formellement : $\emptyset^{(k+1)} = \{\overline{u} | u \in \mathbb{N} \wedge \overline{u} \in$

$W_u^{\emptyset^{(k)}}\}$. Pour tout entier k, $\emptyset^{(k+1)}$ est $\emptyset^{(k)}$-récursivement énumérable non-$\emptyset^{(k)}$-récursif.

Si l'on considère alors les langages qui sont récursivement énumérables dans les $\emptyset^{(k)}, k \in \mathbb{N}$ on obtient la hiérarchie suivante :

Définition 1.1 (Hiérarchie arithmétique [64, 68]) *Σ_1 est la classe des langages récursivement énumérables.*

Pour tout entier $k \geq 1$, Σ_{k+1} est la classe des langages récursivement énumérables dans $\emptyset^{(k)}$.

Cette hiérarchie est stricte puisque $\emptyset^{(k)} \in \Sigma_k^{\emptyset}$ et $\emptyset^{(k)} \notin \Sigma_{k-1}$ pour tout $k \geq 1$. En outre, pour tout entier k, $\emptyset^{(k)}$ est un langage $\Sigma_k^{\emptyset}$-complet pour la réduction "many-one" : cf. [64, 68].

Il est remarquable que la hiérarchie arithmétique possède aussi une caractérisation logique appelée *correspondance de Tarski-Kuratowski*correspondance de Tarski-Kuratowski :

Proposition 1.1 (Caractérisation logique [68]) *Soit R un langage définissable à partir de relations récursives par une formule prénexe existentielle du premier ordre avec n alternances de quantificateurs. Alors $R \in \Sigma_{n+1}$.*

Réciproquement, tout langage $R \in \Sigma_{n+1}$ est définissable par une formule du premier ordre de ce type.

En outre la dépendance en les index des relations récursives est uniforme : si le langage R est défini à l'aide des relations récursives $R_1, \ldots, R_k$, alors il existe une fonction récursive $h : \mathbb{N}^k \to \mathbb{N}$ telle que pour tout $n_1, \ldots, n_k$, si la machine de Turing numéro n_i reconnaît pleinement R_i, alors la machine de Turing avec oracle $\emptyset^{(n+1)}$ de numéro $h(n_1, \ldots, n_k)$ reconnaît pleinement le langage R.

Par exemple le langage $L = \{\overline{u} | W_u$ est un langage récursif$\}$ constitué des entiers $u \in \mathbb{N}$ tels que la machine de Turing de numéro u s'arrête sur toute entrée est dans Σ_2 car L s'écrit $L = \{\overline{u} | \exists y \in \mathbb{N}\ \forall x \in \Sigma^*\ (x \in W_u \Leftrightarrow x \notin W_y)\}$.

Il est aussi possible de relativiser la hiérarchie arithmétique à un oracle $X \subset \Sigma^*$ arbitraire : on pose $X^{(0)} = X$, et pour $k \geq 1$ on définit $X^{(k+1)}$ comme le problème de l'arrêt des machines de Turing avec oracle X :

$$X^{(k+1)} = \{\overline{u} | u \in \mathbb{N} \wedge \overline{u} \in W_u^{X^{(k)}}\}$$

Définition 1.2 (Hiérarchie arithmétique relative à X [64, 68]) *Soit $X \subset \Sigma^*$ un langage.*

- *Σ_1^X est la classe des langages X-récursivement énumérables.*
- *Pour tout entier $k \geq 1$, Σ_{k+1}^X est la classe des langages récursivement énumérables dans $X^{(k)}$.*

On a évidemment $\Sigma_k^{\emptyset} = \Sigma_k$ pour tout entier k. La caractérisation logique de la correspondance de Tarski-Kuratowski se relativise alors en :

Proposition 1.2 (Caractérisation logique [68]) *Soit $X \subset \Sigma^*$ un langage.*

Soit R un langage définissable à partir de relations X-récursives par une formule prénexe existentielle du premier ordre avec n alternances de quantificateurs. Alors $R \in \Sigma_{n+1}^X$.

Réciproquement, tout langage $R \in \Sigma_{n+1}^X$ est définissable par une formule du premier ordre de ce type.

En outre la dépendance en X et en les index des relations récursives est uniforme : si le langage R est défini à l'aide des relations X-récursives $R_1, \ldots, R_k$, alors il existe une fonction récursive $h : \mathbb{N}^k \to \mathbb{N}$, telle que pour tout $X \subset \Sigma^*$, et pour tout $n_1, \ldots, n_k$, si la machine de Turing avec oracle X de numéro n_i reconnaît pleinement R_i, alors la machine de Turing avec oracle $X^{(n+1)}$ de numéro $h(n_1, \ldots, n_k)$ reconnaît pleinement le langage R.

1.2.4 Hiérarchie hyperarithmétique

Une fois construite la hiérarchie arithmétique, se pose naturellement la question de l'existence d'un langage non-arithmétique. La réponse est simple : pour tout $X \subset \Sigma^*$, le langage $X^{(\omega)} = \{< \overline{u}, v > | u \in \mathbb{N}, v \in X^{(u)}\}$ construit par diagonalisation des classes de la hiérarchie arithmétique n'est pas dans la hiérarchie arithmétique relative à X.

On peut alors considérer le problème $X^{(\omega+1)}$ de l'arrêt des machines de Turing avec ce langage comme oracle, puis le problème $X^{(\omega+2)}$ de l'arrêt des machines de Turing avec $X^{(\omega+1)}$ comme oracle, etc.... On peut alors ensuite considérer une diagonalisation $X^{(\omega 2)}$ des langages précédents, puis le problème $X^{(\omega 2+1)}$ de l'arrêt des machines de Turing avec cette diagonalisation comme oracle, puis $X^{(\omega 2+2)}$, etc....

En répétant ce principe, on obtient une extension naturelle de la hiérarchie arithmétique aux ordinaux. Toutefois, l'extension aux ordinaux quelconques pose problème, et il faut se restreindre aux ordinaux récursifs : voir [68].

Ordinaux récursifs

Formellement, un ordinal est dit *récursif* si l'admet une notation dans le sens suivant : rappelons que ϕ_n désigne la fonction calculée par la $n$ème machine de Turing.

Définition 1.3 (Système de notation [68]) *On appelle* système de notation *une fonction partielle ν_s de $\mathbb{N}$ vers un segment des nombres ordinaux telle que :*

1. *il existe une fonction partielle récursive $k_s : \mathbb{N} \to \{0, 1, 2\}$ telle que :*
   - *$\nu_s(x) = 0$ implique $k_s(x) = 0$*
   - *$\nu_s(x)$ est un ordinal successeur implique $k_s(x) = 1$*
   - *$\nu_s(x)$ est un ordinal limite implique $k_s(x) = 2$.*

2. *il existe une fonction partielle récursive* $p_s : \mathbb{N} \to \mathbb{N}$ *telle que* $\nu_s(x)$ *est un ordinal successeur implique (*$p_s(x)$ *défini et* $\nu_s(x) = \nu_s(p_s(x)) + 1$*)*
3. *il existe une fonction partielle récursive* $q_s : \mathbb{N} \to \mathbb{N}$ *telle que* $\nu_s(x)$ *est un ordinal limite implique (*$q_s(x)$ *défini et* $\phi_{q_s(x)}$ *totale et* $\{\nu_s(\phi_{q_s(x)}(n))\}_{n=1}^{\infty}$ *est une suite croissante d'ordinaux de limite* $\nu_s(x)$*).*

Définition 1.4 (Ordinaux récursifs [68]) *Un ordinal* α *est dit* récursif *s'il existe un système de notation qui lui associe une notation.*

On est alors amené à distinguer le système de notation suivant :

Définition 1.5 (Système de notation O **[68])** *Le système de notation* O *est le système de notation défini comme suit :*

- *nous définissons à la fois* ν_O *et un ordre partiel* $<_O$ *sur* O *:*
  L'ordinal O *reçoit l'entier* 1 *comme notation.*
  Supposons que tous les ordinaux $< \gamma$ *aient reçu une notation et supposons que l'ordre partiel* $<_O$ *ait été défini sur ces notations.*
  1. *Si* $\gamma = \beta + 1$ *est un ordinal successeur, alors pour toute notation* x *de* β*, l'ordinal* γ *reçoit la notation* 2^x*. En outre, on définit* $z <_0 2^x$ *pour tout* z *tel que* $z = x$ *ou tel que* $z <_0 x$ *ait été déjà défini.*
  2. *Si* γ *est un ordinal limite, alors l'ordinal* γ *reçoit la notation* 3.5^y *pour tout* $y \in \mathbb{N}$ *tel que* $\{\phi_y(n)\}_{n=1}^{\infty}$ *soit une suite de notations d'une suite croissante d'ordinaux de limite* γ *avec pour tout* $i, j \in \mathbb{N}$*,* $\phi_y(i) <_0 \phi_y(j)$*. En outre, on définit* $z <_0 3.5^y$ *pour tous les* z *tels qu'il existe* $n \in \mathbb{N}$ *avec* $z <_0 \phi_y(n)$*.*
- *la fonction* k_O *est définie par* $k_O(1) = 0$*,* $k_O(2^x) = 1$*,* $k_O(3.5^y) = 2$ *pour tout* $x, y \in \mathbb{N}$*.*
- *la fonction* p_O *est définie par* $p_O(2^x) = x$ *pour tout* $x \in \mathbb{N}$*.*
- *la fonction* q_O *est définie par* $q_O(3.5^y) = y$ *pour tout* $y \in \mathbb{N}$*.*

Ce système possède l'avantage d'être *maximal* en le sens suivant :

Proposition 1.3 ([68]) *Tout ordinal récursif possède une notation dans le système* O*.*

En outre, pour ce système il existe une fonction récursive $+_0$ addition sur les ordinaux récursifs qui prolonge l'addition sur les ordinaux :

Proposition 1.4 (Addition sur les ordinaux [68]) *Il existe une fonction partielle récursive* $+_0 : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *telle que pour toute notation dans le système* O x *d'un ordinal récursif* α *et pour toute notation* y *dans le système* O *d'un ordinal récursif* β*,*

- $x +_0 y$ *est une notation dans le système* O *de l'ordinal* $\alpha + \beta$
- *si* $y \neq 1$*,* $x <_0 x +_O y$*.*

Définition de la hiérarchie hyperarithmétique

Nous sommes prêts à définir formellement la hiérarchie hyperarithmétique. Le principe est le suivant : en un ordinal successeur $\gamma = \beta + 1$, on définit $X^{(\gamma)}$ comme le problème de l'arrêt des machines de Turing avec oracle $X^{(\beta)}$. En un ordinal limite γ, on définit $X^{(\gamma)}$ comme une diagonalisation des $X^{(\alpha)}$ pour $\alpha < \gamma$. Formellement :

Définition 1.6 (Langage $X^{(}z)$) *Soit $X \subset \Sigma^*$ un langage et z une notation dans le système O d'un ordinal récursif.*

Le langage $X^{(z)} \subset \Sigma^*$ est défini par :

- $X^{(z)} = X$ *si* $z = 1$
- $X^{(z)} = \{\overline{u} | u \in \mathbb{N} \wedge \overline{u} \in W_u^{X^{(x)}}\}$ *si* $z = 2^x$
- $X^{(z)} = \{< \overline{u}, v > | u \in \mathbb{N} \ \wedge \ u <_0 3.5^y \ \wedge \ v \in X^{(u)}\}$ *si* $z = 3.5^y$.

La propriété suivante est prouvée dans [68] :

Proposition 1.5 ([68]) *Soient z_1 et z_2 deux notations dans le système O d'un même ordinal récursif.*

Tout ensemble $X^{(z_1)}$-récursivement énumérable est $X^{(z_2)}$-récursivement énumérable.

La hiérarchie hyperarithmétique est alors définie par :

Définition 1.7 (Hiérarchie hyperarithmétique [68]) *Soit $X \subset \Sigma^*$ un langage et soit α un ordinal récursif.*

Pour $\alpha < \omega$, Σ_α^X est définie comme la classe Σ_α^X de la hiérarchie arithmétique.

Pour $\alpha \geq \omega$, soit y une notation de α dans le système O. Σ_α^X est définie comme la classe des langages $X^{(y)}$–récursivement énumérables.

Nous noterons Σ_α pour $\Sigma_\alpha^\emptyset$.

Un exemple naturel de langage hyperarithmétique est le langage qui code toutes les formules arithmétiques logiques du premier ordre qui sont vraies. Ce langage est dans Σ_ω : voir [68].

On observera qu'il est encore possible d'étendre cette hiérarchie. En effet, en utilisant des quantifications du second ordre, il est possible de construire une nouvelle hiérarchie appelée *la hiérarchie analytique* : voir [64, 68]. Elle se relie à la hiérarchie hyperarithmétique par le résultat suivant : Δ_1^1 est égal à l'union des classes Σ_α de la hiérarchie hyperarithmétique pour α ordinal récursif : cf. [64, 68].

1.2.5 Propriétés

Nous présentons maintenant les propriétés de la hiérarchie hyperarithmétique que nous utiliserons dans le chapitre 8. Nous noterons $u \leq_0 v$ pour $u <_0 v$ ou $u = v$. Lorsque z est une notation dans le système O d'un ordinal α, nous noterons $\alpha = |z|$.

Définition 1.8 *Nous dirons qu'une fonction récursive $h : \mathbb{N} \to \mathbb{N}$ est une* suite croissante récursive de notations d'ordinaux dans le système O *si $h(n)$ est une notation dans le système O pour tout $n \in \mathbb{N}$ et si on a $h(n) \leq_0 h(n+1)$ pour tout $n \in \mathbb{N}$.*

h est une fonction des entiers vers les notations des ordinaux récursifs dans le système O et si.

Nous appellerons limite de cette suite *la notation z^* dans le système O de l'ordinal récursif $sup_{n\in\mathbb{N}}|h(n)|$ qui vérifie $h(n) \leq_0 z^*$ pour tout n.*

Il est prouvé dans [68] qu'à partir d'un langage $X^{(z)}$, il est possible de calculer uniformément sur les index tous les langages $X^{(z')}$ pour $z' <_0 z$:

Proposition 1.6 (Restriction d'oracle [68]) *Il existe une fonction récursive $restric : \mathbb{N}^3 \to \mathbb{N}$ telle que pour tout entier n, pour tout oracle $X \subset \Sigma^*$, pour toute notation z dans le système O d'un ordinal récursif, pour toute notation z' dans le système O avec $z' <_0 z$, on ait :*

$$W_n^{X^{(z')}} = W^{X^{(z)}}_{restric(n,z',z)}$$

Le résultat suivant sur la composition des oracles est aussi prouvé dans [68] :

Proposition 1.7 (Composition d'oracles [68]) *Soit $X \subset \Sigma^*$ un langage. Soit x une notation dans le système O d'un ordinal récursif et $Y \subset \Sigma^*$ un langage $X^{(x)}$-récursivement énumérable. Soit y une notation dans le système O d'un ordinal récursif et $Z \subset \Sigma^*$ un langage $Y^{(y)}$-récursivement énumérable.*

Alors Z est $X^{(x+_0 y)}$-récursivement énumérable.

En outre la dépendance en x, y, X et Y est uniforme : il existe une fonction partielle récursive $h : \mathbb{N}^4 \to \mathbb{N}$ telle que pour tout $x, y, m, n \in \mathbb{N}$ et pour tout $X \subset \Sigma^*$, $Y = W_m^{X^{(x)}}$ et $Z = W_n^{Y^{(y)}}$ implique $Z = W^{X^{(x+_0 y)}}_{h(m,n,x,y)}$.

Enfin nous utiliserons le théorème du point fixe :

Proposition 1.8 (Théorème du point fixe [64, 68]) *Soit $k \geq 0$ un entier. Soit $f : \mathbb{N}^{k+1} \to \mathbb{N}$ une fonction récursive à $k+1$ variables.*

Il existe une fonction g à k variables telle que pour tout $n_1, \ldots, n_k \in \mathbb{N}$ et pour tout oracle $X \subset \Sigma^*$, on ait :

$$W^X_{g(n_1,\ldots,n_k)} = W^X_{f(g(n_1,\ldots,n_k),n_1,\ldots,n_k)}$$

1.3 Modèles réels

Nous discutons maintenant les modèles de calcul sur les réels. Dans la section 1.3.1 nous présentons le modèle de l'analyse récursive. Dans la section 1.3.2, nous présentons le modèle des machines de Blum Shub et Smale. Observons qu'il existe d'autres modèles de fonctions calculables : voir par exemple les travaux de [62, 69, 78].

1.3.1 Analyse récursive

L'analyse récursive considère que les machines qui manipulent des réels le font via leurs représentations. Elle suppose fixée une représentation des réels par des mots infinis. Une fonction réelle est alors dite calculable si une machine de Turing peut calculer une représentation du résultat de la fonction à partir de représentations des arguments de la fonction.

Nous adoptons la présentation proposée par Weihrauch dans [85] : un *système de codage* d'un ensemble M est une notation ou une représentation de l'ensemble M : une *notation* de M est une fonction surjective (éventuellement partielle) $\rho_M : \Sigma^* \to M$; une *représentation* est une fonction surjective (éventuellement partielle) $\rho_M : \Sigma^\omega \to M$.

En particulier, on peut considérer $\rho_{\mathbb{N}} : \Sigma^* \to \mathbb{N}$ la notation de $\mathbb{N}$ qui représente un entier n par son écriture binaire $\overline{n}$ et $\rho_{\mathbb{Q}} : \Sigma^* \to \mathbb{Q}$ la notation de $\mathbb{Q}$ qui représente un rationnel par sa fraction simplifiée.

On peut aussi considérer la représentation suivante des réels :

Définition 1.9 (Représentation $\rho_{\mathbb{R}}$) *La représentation $\rho_{\mathbb{R}} : \Sigma^\omega \to \mathbb{R}$ de $\mathbb{R}$ consiste à représenter un réel x par un mot infini $p \in \Sigma^\omega$ avec $p = \#u_0\#u_1\#\ldots$ tel que*

1. *$u_0, u_1, \ldots \in dom(\rho_{\mathbb{Q}})$ sont des codages de rationnels*
2. *$\forall k\ \forall i, j > k\ |\rho_{\mathbb{Q}}(u_i) - \rho_{\mathbb{Q}}(u_j)| < 2^{-k}$*
3. *$x = \lim_{i\to\infty} \rho_{\mathbb{Q}}(u_i)$.*

D'autres représentations sont possibles pour les réels. Mais nous renvoyons notre lecteur à [85] pour une discussion des équivalences qui existent entre les représentations et pour une argumentation visant à prouver que cette représentation est une "bonne représentation".

On dit alors qu'une fonction ou un réel est calculable s'il vérifie :

Définition 1.10 (Fonctions et réels calculables) – *Un réel x est dit* calculable *s'il existe une fonction $f : \Sigma^* \to \Sigma^\omega$ calculable par une machine de Turing avec $\rho_{\mathbb{R}}(f(\epsilon)) = x$.*

– *Une fonction partielle* $F : X_1 \times \ldots \times X_k \to X_0$*, avec pour* $i \in \{0, \ldots, k\}$ $X_i \in \{\mathbb{N}, \mathbb{R}\}$*, est dite* calculable *si et seulement s'il existe une fonction* f *calculable par une machine de Turing telle que*

$$F(\rho_{X_1}(y_1), \ldots, (\rho_{X_k}(y_k))) = \rho_{X_0}(f(y_1, \ldots, y_k))$$

lorsque $F(\rho_{X_1}(y_1), \ldots, \rho_{X_k}(y_k))$ *existe.*

Un des résultats importants vérifiés par ce modèle est le suivant :

Proposition 1.9 *Une fonction* $F : \mathbb{R}^k \to \mathbb{R}$ *calculable est nécessairement continue.*

La plupart des fonctions usuelles comme les fonctions somme, exponentielle, logarithme, sinus, etc. . .sont calculables dans ce modèle. Toutefois, en accord avec la proposition précédente, les fonctions discontinues comme les fonctions "signes" ne sont pas calculables.

Nous renvoyons notre lecteur à [85] pour une introduction plus détaillée et pour de nombreuses références.

1.3.2 Modèle de Blum Shub et Smale

Le modèle de Blum Shub et Smale [14] considère quant à lui des machines qui réalisent des opérations exactes en précision non-bornée. Nous adoptons une présentation inspirée de celle utilisée pour présenter les machines de Turing.

Une *machine BSS* [14] est une machine abstraite de calcul déterministe avec un nombre fini d'états internes $Q = \{0, 1, \ldots, m-1\}$ et un ruban infini. Ce ruban peut être vu comme une suite indexée par $\mathbb{Z}$ de nombres réels. En fonction de son état interne et du signe du réel sur le ruban à l'index 0, la machine peut effectuer les opérations suivantes : déplacer le ruban vers la droite (chaque réel à l'index i du ruban est dorénavant à l'index $i + 1$), déplacer un ruban vers la gauche (chaque réel à l'index i du ruban est dorénavant à l'index $i - 1$), changer son état interne en un nouvel état interne, ou remplacer le réel du ruban d'index 0 par k, par $-x_0$, par $k.x_0$, par $x_0 + x_1$, ou par $x_0 * x_1$ où x_0 et x_1 désignent respectivement les valeurs des réels du ruban d'index 0 et 1, et k désigne une constante du programme : voir [14] pour une description plus formelle.

Les programmes des machines BSS sont constitués d'un nombre fini de telles instructions. Posons $\mathbb{R}^\infty = \cup_{i \in \mathbb{N}} \mathbb{R}^i$. Un *langage réel* est une partie de $\mathbb{R}^\infty$. Un langage réel $L \subset \mathbb{R}^\infty$ est dit *BSS-récursivement énumérable* s'il existe une machine BSS telle que L coïncide avec l'ensemble des mots $y \in \mathbb{R}^\infty$ tels que M partant dans l'état interne $m - 1$ et avec le ruban $0^\omega y 0^\omega$ arrive ultimement dans une configuration où plus aucune instruction ne s'applique. L est dit *BSS-récursif* si le complémentaire de L dans $\mathbb{R}^\infty$ est aussi BSS-récursivement énumérable.

Dans ce modèle, la fonction signe est calculable. Les fonctions polynômiales et polynômiales par morceaux sont facilement calculables. Mais les fonctions exponentielles, logarithmes, sinus, etc...ne sont pas calculables.

Rappelons qu'un ensemble $S \subset \mathbb{R}^n, n \in \mathbb{N}$, est appelé un *ensemble semi-algébrique de base* s'il existe un nombre fini de polynômes à n variables p_1, p_2, ..., p_{n_1}, p'_1, ..., p'_{n_2} tels que S s'écrive $S = \{(x_1, \ldots, x_n) \mid p_1(x_1, \ldots, x_n) > 0 \wedge \ldots \wedge p_{n_1}(x_1, \ldots, x_n) > 0 \wedge p'_1(x_1, \ldots, x_n) = 0 \wedge \ldots \wedge p'_{n_2}(x_1, \ldots, x_n) = 0\}$, et qu'un ensemble $S \subset \mathbb{R}^n, n \in \mathbb{N}$, est dit *semi-algébrique* s'il s'écrit comme une union finie d'ensembles semi-algébriques de base.

Nous utiliserons dans le chapitre 4 le résultat suivant prouvé dans [14] :

Proposition 1.10 ([14]) *Soit $L \subset \mathbb{R}^\infty$ un langage réel BSS-récursivement énumérable. L s'écrit comme une union dénombrable d'ensembles semi-algébriques.*

Nous renvoyons notre lecteur à [59] pour une synthèse des résultats connus dans ce modèle, ou à [16, 27] pour des tentatives de comparaison de ce modèle avec le modèle de l'analyse récursive.

CHAPITRE

2 Systèmes dynamiques et hybrides

2.1 Introduction

Ce chapitre a pour objectif de définir formellement les classes de systèmes dynamiques et hybrides qui seront utilisées dans le reste du document.

Les systèmes dynamiques que nous étudierons seront tous à espace continu. La section 2.2 introduit les systèmes à temps discret. La section 2.3 introduit les systèmes à temps continu. Enfin la section 2.4 précise comment les systèmes dynamiques peuvent être vus comme des modèles de calcul.

Nous appelons *polyèdre* de $\mathbb{R}^d$ tout sous-ensemble de $\mathbb{R}^d$ qui s'écrit comme une union finie d'intersections finies entre demi-espaces ouverts ou fermés. Formellement :

Définition 2.1 (Polyèdre) *Un* demi-espace rationnel S *de* $\mathbb{R}^d$ *est un sous-ensemble de* $\mathbb{R}^d$ *qui admet une équation du type* $S = \{(x_1, \ldots, x_d) | a_1 x_1 + \ldots + a_d x_d \# b\}$ *pour* $\# \in \{<, \leq\}$ *et pour certains nombres rationnels* $a_1, \ldots, a_d$ *et* b.

Un polyèdre convexe rationnel *de* $\mathbb{R}^d$ *est une intersection finie entre demi-espaces rationnels.*

Un polyèdre rationnel *de* $\mathbb{R}^d$ *est une union finie de polyèdres convexes rationnels de* $\mathbb{R}^d$.

Nous considérons donc que les polyèdres dégénérés sont des polyèdres particuliers : par exemple, un ensemble réduit à un point de coordonnées rationnelles, ou une droite joignant deux points de coordonnées rationnelles sont pour nous des polyèdres de $\mathbb{R}^d$.

Définition 2.2 (Dimension affine d'un polyèdre) *Un polyèdre* P *est* de dimension affine d' *si* P *est inclus dans un sous-espace affine de dimension* d'*, et si* P *n'est pas inclus dans un sous-espace affine de dimension* $d' - 1$.

Une fonction affine ou linéaire sera dite *rationnelle* si elle vérifie :

Définition 2.3 (Fonction affine ou linéaire rationnelle) *Une* fonction linéaire rationnelle *de* $\mathbb{R}^d$ *vers* $\mathbb{R}^{d'}$ *est une fonction du type* $x \mapsto Ax$ *où* A *est une matrice* $d' \times d$ *à coefficients rationnels.*

Une fonction affine rationnelle *de* $\mathbb{R}^d$ *vers* $\mathbb{R}^{d'}$ *est une fonction du type* $x \mapsto Ax + b$ *où* A *est une matrice* $d' \times d$ *à coefficients rationnels et* b *est un vecteur* $d' \times 1$ *à coefficients rationnels.*

Enfin, une fonction sera dite *constante par morceaux*, ou *affine par morceaux* si elle est constante ou affine sur un nombre fini de polyèdres :

Définition 2.4 (Fonction affine ou constante par morceaux) *Une fonction* $f : \mathbb{R}^d \to \mathbb{R}^{d'}$ *est dite* affine par morceaux *si l'on peut partitionner l'espace* $\mathbb{R}^d$ *en un nombre fini de polyèdres rationnels tels que* f *soit affine rationnelle sur chacun des polyèdres.*

Une fonction $f : \mathbb{R}^d \to \mathbb{R}^{d'}$ *est dite* constante par morceaux *si l'on peut partitionner l'espace* $\mathbb{R}^d$ *en un nombre fini de polyèdres rationnels tels que* f *soit constante rationnelle sur chacun des polyèdres.*

2.2 Systèmes dynamiques à temps discret

Nous introduisons maintenant les systèmes dynamiques à temps discret.

2.2.1 Définitions

Traditionnellement, on définit un système dynamique à temps discret sur $\mathbb{R}^d$ de la façon suivante :

Définition 2.5 (Système dynamique à temps discret) *Nous appelons* système dynamique à temps discret *un couple* (X, f) *où* $X \subset \mathbb{R}^d$ *et* f *est une fonction de* X *vers* X*. L'ensemble* X *est appelé* l'espace du système *et l'entier* d *est appelé* la dimension du système*.*

Soit $x_0 \in X$ *un point de l'espace. La* trajectoire du système partant de x_0 *est la suite des itérés de* f *en* x_0 *: c'est-à-dire la suite à valeur dans* X *définie par* $x_{i+1} = f(x_i)$ *pour tout* $i \in \mathbb{N}$*.*

Il est aussi possible d'autoriser l'intervention d'un contrôle extérieur :

Définition 2.6 (Système dynamique contrôlé) *Nous appelons* système dynamique à temps discret contrôlé *un triplet* (X, Y, f) *où* $X \subset \mathbb{R}^d$*,* Y *est un ensemble, et* f *est une fonction de* $X \times Y$ *vers* X*. L'ensemble* X *est appelé* l'espace du système*, l'ensemble* Y *est appelé* l'espace du contrôle*, et l'entier* d *est appelé* la dimension du système*. Un* contrôle du système *est une fonction* u *de* $\mathbb{N}$ *vers* Y*.*

Soit $x_0 \in X$ *un point de l'espace et* $u : \mathbb{N} \to Y$ *un contrôle. La* trajectoire du système partant de x_0 pour le contrôle u *est la suite à valeur dans* X *définie par* $x_{i+1} = f(x_i, u(i))$ *pour tout* $i \in \mathbb{N}$*.*

Nous étudierons les classes suivantes de systèmes à temps discret : les systèmes affines par morceaux, les réseaux de neurones et les systèmes linéaires commutés.

Nous présentons maintenant chacun de ces systèmes.

2.2.2 Systèmes affines par morceaux

Formellement, un système affine par morceaux se définit par (voir [45] par exemple) :

Définition 2.7 (Système affine par morceaux) *Un* système affine par morceaux *est un système dynamique à temps discret* $(\mathbb{R}^d, f)$ *où* f *est une fonction affine par morceaux.*

Le système est dit continu *si la fonction* f *est continue.*

Par exemple, le système dynamique $(\mathbb{R}, f)$ défini par

$$f : x \mapsto \begin{cases} -4x & \text{si } x < 0 \\ x + 1 & \text{si } x \in [0, 1[\\ 1/2 + x/2 & \text{si } x \geq 1 \end{cases}$$

est un système affine par morceaux de dimension 1 dont toutes les trajectoires convergent vers le point 1.

2.2.3 Systèmes linéaires commutés

Un système linéaire commuté se définit par (voir [13] par exemple) :

Définition 2.8 (Système linéaire commuté) *Un* système linéaire commuté *est un système dynamique* $(\mathbb{R}^d, Y, f)$ *à temps discret contrôlé tel que* Y *s'écrive* $Y = \{1, 2, \ldots, k\}$*, pour* $k \in \mathbb{N}$*, et tel que pour tout* $u \in \{1, \ldots, k\}$*,* $f(., u)$ *soit une fonction linéaire rationnelle.*

Par exemple, le système dynamique $(\mathbb{R}, \{1, 2\}, f)$ à temps discret contrôlé défini par $f(x, 1) = x/2$, $f(x, 2) = 3x$ est un système linéaire commuté de dimension 1.

2.2.4 Réseaux de neurones

Les réseaux de neurones sont des systèmes qui sont très utilisés pour résoudre des problèmes de classification ou pour les problèmes qui nécessitent un apprentissage : voir [25] par exemple pour une introduction.

Un réseau de neurones peut être vu comme un système dynamique à temps discret avec une fonction de transition du type $f(x) = \sigma(Ax + b)$: la matrice A correspond aux poids des connexions du réseau, le vecteur b correspond aux seuils

en chacun des neurones du réseau, et σ correspond à la fonction de transition du réseau. Traditionnellement, on utilise une fonction de transition σ "sigmoïde" comme la fonction arc tangente : voir [25].

Dans ce document, nous n'autoriserons pas des fonctions de transition aussi générales. Nous nous limiterons aux fonctions de transition linéaires seuillées. Ainsi, un réseau de neurones sera pour nous défini par (voir [82] pour une restriction similaire) :

Définition 2.9 (Réseau de neurones) *Soit $\sigma : \mathbb{R} \to \mathbb{R}$ la fonction continue définie par*

$$\sigma : x \mapsto \begin{cases} 1 & \text{lorsque } x \geq 1 \\ x & \text{lorsque } |x| < 1 \\ -1 & \text{lorsque } x \leq -1 \end{cases}$$

Lorsque $x = (x_1, \ldots, x_d)$ est un point de $\mathbb{R}^d$, on notera $\sigma(x)$ pour $(\sigma(x_1), \ldots, \sigma(x_d))$. Une σ-fonction *de $\mathbb{R}^d$ vers $\mathbb{R}^{d'}$ est une fonction $f : \mathbb{R}^d \to \mathbb{R}^{d'}$ du type $x \mapsto \sigma(Ax+b)$ où A est une matrice $d' \times d$ à coefficients rationnels, et b est un vecteur $d' \times 1$ à coefficients rationnels.*

Un réseau de neurones *est un système dynamique à temps discret $(\mathbb{R}^d, f)$ où f est une σ-fonction.*

Observons qu'un réseau de neurones est un système affine par morceaux particulier.

2.3 Systèmes dynamiques à temps continu

2.3.1 Définitions

Nous définissons un système dynamique à temps continu sur $\mathbb{R}^d$ comme un couple (X, f) où $X \subset \mathbb{R}^d$ et $f : X \to \mathbb{R}^d$ définit les trajectoires du système par l'équation différentielle $\dot{x} = f(x)$.

Cependant, on suppose généralement la fonction f lipschitizienne ou continue : supposer la fonction f continue permet de garantir l'existence des trajectoires, et supposer la fonction f lipschitizienne permet en outre de garantir l'unicité des trajectoires : voir [30] par exemple.

Nous ne supposons pas que la fonction f est nécessairement continue. Ainsi, pour nous, un système dynamique à temps continu est :

Définition 2.10 (Système dynamique à temps continu) *Nous appelons* système dynamique à temps continu *un couple (X, f) où $X \subset \mathbb{R}^d$ et f est une fonction de X vers $\mathbb{R}^d$. L'ensemble X est appelé* l'espace du système *et l'entier d est appelé* la dimension du système.

Soit $x_0 \in X$ un point de l'espace. Une trajectoire du système partant de x_0 *est une solution ϕ de l'équation différentielle $\dot{x} = f(x)$ dans le sens suivant :*

1. *ϕ est une fonction continue de I vers X où I est un intervalle non vide du type*
   $I = [0, t^*[$, $I = [0, t^*]$ ou $I = [0, \infty[$
2. *$\phi(0) = x_0$*
3. *ϕ est dérivable à droite sur I*
4. *$\phi'_d(t) = f(\phi(t))$ pour tout $t \in [0, t^*[$ si $I = [0, t^*[$ (resp. $I = [0, t^*]$) et pour tout $t \in [0, \infty[$ si $I = [0, \infty[$: $\phi'_d(t)$ désigne la dérivée à droite de Φ en t.*

2.3.2 Systèmes à dérivée constante par morceaux

Dans ce document, nous étudierons longuement les systèmes à dérivée constante par morceaux. Un système à dérivée constante par morceaux (DCPM) se définit par :

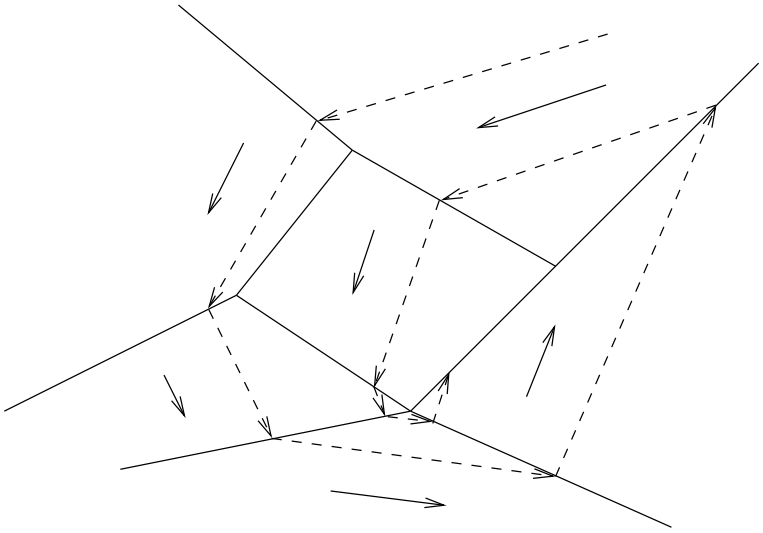


fig :pcdUn système DCPM de dimension 2

Définition 2.11 (Système DCPM) *Un* système à dérivée constante par morceaux
(DCPM) *est un système dynamique à temps continu $(\mathbb{R}^d, f)$ où f est une fonction constante par morceaux.*

En d'autres termes, un système DCPM est obtenu en partitionnant l'espace $\mathbb{R}^d$ en un nombre fini de régions polyédrales et en fixant un *vecteur pente* pour chacune des régions. Les trajectoires du système sont des lignes brisées joignant les frontières des régions suivant les pentes : observez l'exemple de la figure **??**.

2.3.3 Existence et unicité des trajectoires

Etudions un peu plus l'existence et l'unicité des trajectoires dans les systèmes dynamiques à temps continu. Commençons par discuter le domaine des trajectoires maximales dans les systèmes dynamiques à temps continu génériques :

Proposition 2.1 (Domaine des trajectoires maximales) *Soit $(\mathbb{R}^d, f)$ un système dynamique à temps continu tel que la fonction f soit bornée.*

Soit ϕ une trajectoire maximale *du système : c'est-à-dire une trajectoire qui ne s'étend pas en une trajectoire définie sur un intervalle plus grand.*

Alors

- *soit ϕ est définie sur $I = \mathbb{R}$: on dit que la trajectoire* continue à jamais
- *soit ϕ est définie sur un intervalle fermé du type $I = [0, t^*]$, et il n'y a aucune trajectoire qui parte du point $\phi(t^*)$.*

Preuve: Supposons que ϕ soit définie sur un intervalle de type $I = [0, t^*[$. Puisque la fonction f est bornée, par le théorème des accroissements finis, l'image par ϕ de toute suite de Cauchy de I est une suite de Cauchy. Puisque $\mathbb{R}^d$ est un espace complet, $lim_{t \to t^*} \phi(t)$ doit exister. Φ se prolonge donc en une fonction continue définie sur $I = [0, t^*]$ qui vaut $\phi(t)$ sur $[0, t^*[$ et $lim_{t \to t^*} \phi(t)$ en t^*. Cette fonction est, par définition, une trajectoire du système, et donc ϕ n'est pas maximale.

Maintenant, lorsque ϕ est définie sur $I = [0, t^*]$, il est clair que, s'il existe une trajectoire partant de $\phi(t^*)$, alors ϕ peut s'étendre en une trajectoire définie sur un intervalle plus grand. □

Discutons maintenant l'unicité des trajectoires pour les systèmes dynamiques à temps continu génériques :

Proposition 2.2 (Existence et unicité des trajectoires) *Soit $(\mathbb{R}^d, f)$ un système dynamique à temps continu avec existence et unicité locale des trajectoires : c'est-à-dire tel qu'en tout point $x \in \mathbb{R}^d$*

1. *il existe une trajectoire partant du point x*
2. *si ϕ et ϕ' sont des trajectoires partant de x, alors il existe $\epsilon > 0$ tel que pour tout $t \in [0, \epsilon]$, $\phi(t) = \phi'(t)$.*

Alors par tout point de l'espace passe exactement une solution maximale.

Preuve: Soient $\phi : I \to \mathbb{R}^d$ et $\phi' : I' \to \mathbb{R}^d$ deux trajectoires maximales partant du point x. Supposons $I' \subset I$ sans perte de généralité. Il suffit de prouver que $t_{sup} = \sup_{t \in I \cap I'} \{t | \phi(t) = \phi(t')\}$ est la borne supérieure de I'. Les fonctions ϕ et ϕ' étant continues, on doit avoir $\phi(t_{sup}) = \phi'(t_{sup})$. Or si t_{sup} n'était pas la borne supérieure de I', en appliquant l'hypothèse au point $\phi(t_{sup})$, il existerait $\epsilon > 0$ avec $\phi(t) = \phi'(t)$ pour $t \in [t_{sup}, t_{sup} + \epsilon]$ et la trajectoire ϕ' ne serait pas maximale. □

Si l'on particularise ces résultats aux systèmes à dérivée constante par morceaux, on est amené à faire la remarque suivante :

Observation 2.1 (Existence et unicité locale pour les systèmes DCPM) *Soit $H = (\mathbb{R}^d, f)$ un système DCPM.*

Il existe une trajectoire partant d'un point $x \in \mathbb{R}^d$ si et seulement si le point x est tel qu'il existe $\epsilon > 0$ avec $f(x') = f(x)$ pour tout point x' du segment $[x, x + \epsilon f(x)]$ de $\mathbb{R}^d$.

En outre, lorsque tel est le cas, il y a unicité locale des trajectoires : si ϕ et ϕ' sont des trajectoires partant de x, alors il existe $\epsilon > 0$ tel que pour tout $t \in [0, \epsilon]$ on ait $\phi(t) = \phi(t')$.

Posons alors :

Définition 2.12 (Points et systèmes bien définis) *Soit H un système DCPM de dimension d.*

Un point $x \in \mathbb{R}^d$ de l'espace de $H = (\mathbb{R}^d, f)$ est dit bien défini *s'il existe $\epsilon > 0$ tel que $f(x') = f(x)$ pour tout x' du segment $[x, x + \epsilon f(x)]$ de $\mathbb{R}^d$.*

On note $BienDefini(H)$ l'ensemble des points $x \in \mathbb{R}^d$ de l'espace du système qui sont bien définis. Le système H est dit bien défini *s'il vérifie $BienDefini(H) = X$.*

On obtient alors :

Corollaire 2.1 *Soit $H = (\mathbb{R}^d, f)$ un système DCPM de dimension d.*

Soit ϕ une trajectoire maximale de H.

Alors

- *soit ϕ continue à jamais*
- *soit ϕ est définie sur un intervalle fermé du type $I = [0, t^*]$ et $\phi(t^*)$ est un point qui n'est pas bien défini.*

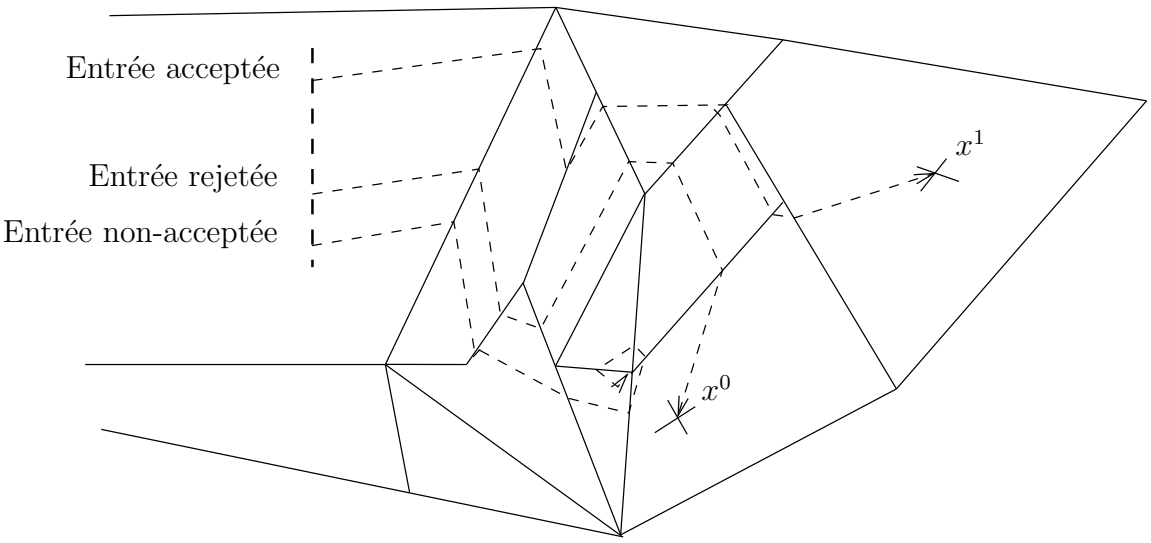
Et :

Corollaire 2.2 *Soit $H = (\mathbb{R}^d, f)$ un système DCPM de dimension d bien défini. Alors par tout point de l'espace de H passe exactement une trajectoire maximale et cette trajectoire maximale continue à jamais.*

Notons que l'on peut faire la remarque suivante :

Observation 2.2 *Soit $H = (\mathbb{R}^d, f)$ un système DCPM de dimension d.*

L'ensemble $BienDefini(H)$ est un polyèdre rationnel calculable.



2.4 Systèmes dynamiques en tant que modèles de calcul

Dans le chapitre 7 et dans le chapitre 8 nous considérerons les systèmes dynamiques comme des modèles de calcul et nous chercherons à caractériser leur puissance de calcul. Nous le ferons de la façon suivante : tout d'abord, nous fixons un moyen de coder un mot sur un alphabet fini Σ comme un point de $\mathbb{R}^d$.

fig :excalExemples de calculs par un système DCPM

Définition 2.13 (Encodage Ξ) *Supposons que l'on se fixe l'alphabet fini $\Sigma = \{0, 1\}$.*

On appelle $\Xi : \Sigma^\omega \to [0, 1]$ la fonction définie par $\Xi(p) = \sum_{j=0}^{\infty}(2a_j)/(4)^{j+1}$ lorsque le mot p s'écrit $p = a_1a_2\ldots$ avec chacun des $a_j \in \Sigma$.

Nous définissons alors : voir la figure **??**.

Définition 2.14 (Calcul par un système dynamique [9, 23, 44]) *Soit $(\mathbb{R}^d, f)$ un système dynamique à temps discret ou continu. Soient x^0 et x^1 deux points de l'espace. On suppose que x^0 et x^1 sont des points fixes du système.*

Soit $w \in \Sigma^\omega$ un mot sur l'alphabet Σ. Un calcul du système *sur w est une trajectoire du système partant du point de coordonnées $(\Xi(w), 0, \ldots, 0)$. Le calcul est dit* acceptant *s'il atteint le point x^1. Le calcul est dit* rejetant *s'il atteint le point x^0.*

Un langage $L \subset \Sigma^*$ est dit semi-reconnu par le système *si L coïncide avec l'ensemble des mots $w \in \Sigma^*$ tels que le calcul du système sur cette entrée soit acceptant.*

Un langage est dit (pleinement-)reconnu par le système *si en outre pour tout* $w \in \Sigma^*$*,* $w \notin L$*, le calcul du système sur* w *est rejetant.*

Le point x^1 *est appelé le* point d'acceptation du système. *Le point* x^0 *est appelé le* point de rejet du système. *Le segment* $[0,1] \times \{0\}^{d-1}$ *est appelé le* port d'encodage du système.

En d'autres termes, une entrée $w \in \Sigma^*$ d'un calcul est encodée par un point rationnel de l'espace, et est dite acceptée (respectivement refusée) si et seulement si la trajectoire partant de ce point atteint le point d'acceptation (respectivement le point de rejet).

Bien entendu, il se peut qu'une entrée soit ni acceptée ni refusée : cela correspond au fait que la machine calcule sans s'arrêter (voir la figure **??**).

CHAPITRE

3 Indécidabilité de la vérification pour les des systèmes dynamiques

3.1 Introduction

Ce chapitre s'intéresse à la décidabilité des propriétés d'atteignabilité, de contrôlabilité, et de stabilité pour chacune des classes de systèmes dynamiques introduites dans le chapitre 2. Il prouve que chacune de ces propriétés est indécidable. En particulier, il prouve l'indécidabilité des problèmes de la stabilité pour les réseaux de neurones et répond à une conjecture exprimée à plusieurs reprises dans la littérature : voir [12, 83].

Le plan de ce chapitre est le suivant : dans la section 3.2, nous étudions la décidabilité du problème de l'atteignabilité. Dans la section 3.3, nous étudions la décidabilité du problème de la contrôlabilité. Enfin, dans la section 3.4, nous étudions la décidabilité des problèmes de la stabilité.

Nous renvoyons notre lecteur au chapitre 2 pour les définitions précises des systèmes dynamiques étudiés.

3.2 Problème de l'atteignabilité

Commençons par discuter le problème de l'atteignabilité : étant donné un sous-ensemble $R \subset \mathbb{R}^d$, nous dirons qu'une trajectoire $x_{i+1} = f(x_i)$ d'un système à temps discret (X, f) *atteint* R s'il existe $t_0 \in \mathbb{N}$ avec $x_{t_0} \in R$. Etant donné un sous-ensemble $R \subset \mathbb{R}^d$, nous dirons qu'une trajectoire ϕ d'un système à temps continu (X, f) *atteint* R s'il existe t_0 dans le domaine de ϕ avec $\phi(t_0) \in R$. L'origine désigne le point de $\mathbb{R}^d$ de coordonnées $(0, \ldots, 0)$ dans la base canonique.

Définition 3.1 (Problème de l'atteignabilité) *On appelle* problème de l'atteignabilité pour une classe $\mathcal{C}$ *de systèmes dynamiques à temps discret ou continu le problème de décision suivant :*

- *Instance : un système dynamique de la classe $\mathcal{C}$, un point $x_0 \in \mathbb{Q}^d$.*
- *Question : la trajectoire partant de x_0 atteint-elle l'origine ?*

3.2.1 Résultats

Le problème de l'atteignabilité pour chacune des classes de systèmes dynamiques à temps discret introduites dans le chapitre 2 est indécidable. En effet, les résultats suivants sont prouvés dans [45, 61, 82].

Théorème 3.1 (Systèmes à temps discret [45, 61, 82]) *Les problèmes de l'atteignabilité*
- *pour les systèmes affines par morceaux*
- *pour les systèmes continus affines par morceaux*
- *pour les réseaux de neurones*

sont indécidables.

Pour les systèmes dynamiques à temps continu, le problème reste indécidable. En effet, le résultat suivant est prouvé dans [9].

Théorème 3.2 (Systèmes à temps continu [9]) *Le problème de l'atteignabilité pour les systèmes DCPM bien définis est indécidable.*

Nous discutons maintenant les preuves de chacun des résultats du théorème 3.1. Le théorème 3.2 sera prouvé dans le chapitre 7.

3.2.2 Systèmes affines par morceaux

Commençons par l'indécidabilité du problème de l'atteignabilité pour les systèmes affines par morceaux. Elle découle immédiatement du résultat suivant de simulation d'une machine de Turing par une fonction affine par morceaux :

Lemme 3.1 (Simulation d'une machine de Turing) *Soit M une machine de Turing à un ruban. Soit $C = \Sigma^\omega \times \Sigma^\omega \times Q$ l'espace de ses configurations.*

Alors il existe une fonction affine par morceaux $g_M : \mathbb{R}^2 \to \mathbb{R}^2$ et une fonction d'encodage $\nu : C \to [0,1]^2$, avec $\nu(\Sigma^* \times \Sigma^* \times Q) \subset \mathbb{Q}^2$, telles que le diagramme suivant commute :

$$\begin{array}{ccc} C & \xrightarrow{\vdash} & C \\ {\scriptstyle\nu}\downarrow & & \downarrow{\scriptstyle\nu} \\ \mathbb{R}^2 & \xrightarrow{g_M} & \mathbb{R}^2 \end{array}$$

(c'est-à-dire telles que $g_M(\nu(c)) = \nu(c')$ pour toutes configurations $c, c' \in C$ avec $c \vdash c'$).

Preuve: Comme nous l'avons fait dans le chapitre 1, nous pouvons supposer que l'alphabet Σ s'écrit $\Sigma = \{0, 1, \ldots, n-1\}$, que l'ensemble Q des états internes

de la machine M s'écrit $Q = \{0, 1, \ldots, m-1\}$ et que 0 est l'unique état interne accepteur de la machine M.

Définissons $\nu : C \to [0,1]^2$ par $\nu(p_1, p_2, q) = (q/m + x_1/m, x_2)$ avec

$$x_i = \sum_{j=0}^{\infty} (2a_i^j)/(2n)^{j+1}$$

si $p_i \in \Sigma^\omega$ s'écrit $p_i = a_i^0 a_i^1 a_i^2 \ldots$ avec les $a_i^j \in \Sigma$. Pour tout $\alpha, \beta \in \Sigma, q \in Q$, définissons le sous-ensemble $B_{\alpha,\beta,q}$ par $B_{\alpha,\beta,q} = [q/m + (2\alpha)/(2mn), q/m + (2\alpha + 1)/(2mn))] \times [(2\beta)/(2n), (2\beta+1)/(2n)]$. Par définition de ν, une configuration du type $(\alpha p_1', \beta p_2', q)$ avec $p_1', p_2' \in \Sigma^\omega$, $q \in Q$, a pour image par ν un point de $B_{\alpha,\beta,q}$. Par conséquent le même quintuplet de la machine de Turing M s'applique à toutes les configurations qui sont envoyées par ν dans un même pavé $B_{\alpha,\beta,q}$.

Un tel quintuplet a pour effet de remplacer le symbole α d'index 0 du ruban par un nouveau symbole α', de déplacer le ruban vers la droite ou vers la gauche et de changer l'état interne q en un nouvel état interne q' : nous définissons alors g_M sur la boîte $B_{\alpha,\beta,q}$ par $g_M(q/m + x_1/m, x_2) = (q'/m + x_1'/m, x_2')$ où $x_1' = ax_1 + b$, $x_2' = cx_2 + d$ avec :

- $a = 2n, b = -2\alpha, c = 1/(2n), d = (2\alpha')/(2n)$ si le ruban est déplacé vers la gauche.
- $a = 1/(2n), b = (2\beta)/(2n) + 2(\alpha' - \alpha)/(2n)^2, c = 2n, d = -2\beta$ si le ruban est déplacé vers la droite.

On a alors $g_M(\nu(c)) = \nu(c')$ pour tout $c, c' \in C$ avec $c \vdash c'$. □

3.2.3 Systèmes continus affines par morceaux et réseaux de neurones

Passons maintenant à la preuve de l'indécidabilité du problème de l'atteignabilité pour les systèmes continus affines par morceaux et pour les réseaux de neurones.

Puisqu'un réseau de neurones est un système continu affine par morceaux particulier, il nous suffit de prouver l'indécidabilité du problème de l'atteignabilité pour les réseaux de neurones.

Rappelons que nous appelons σ-fonction une fonction du type $x \mapsto \sigma(Ax+b)$: voir la définition 2.9. Nous appellerons *σ^*-fonction* une fonction qui s'écrit comme la composée d'un nombre fini de σ-fonctions.

Il est facile de voir que prouver l'indécidabilité du problème de l'atteignabilité pour les réseaux de neurones est équivalent à prouver l'indécidabilité du problème de l'atteignabilité pour les systèmes affines à temps discret du type $(\mathbb{R}^d, f)$ où f est une σ^*-fonction. Or l'indécidabilité de ce dernier problème découle immédiatement de l'observation suivante et de son corollaire.

Observation 3.1 *Soit P une union finie de pavés fermés disjoints de $\mathbb{R}^2$. Soit $f : P \to [-1, 1]$ une fonction affine par morceaux, affine sur chacun des pavés de P.*

Alors il existe une σ^*-fonction $g : \mathbb{R}^2 \to \mathbb{R}^2$ égale à f sur P.

Preuve: Lorsque ϵ est un réel positif et a un réel, observons que la fonction $h^+_\epsilon(x, a) = \sigma(1 + 2/\epsilon(x - a)))$ vaut 1 si $x \geq a$ et -1 si $x \leq a - \epsilon$. La fonction $h^-_\epsilon(x, a) = h^+_\epsilon(-x, -a)$ vaut 1 si $x \leq a$ et -1 si $x \geq a + \epsilon$. Ecrivons alors $P = \cup_{i=1}^n B_i$ avec $B_i = [a^1_i, b^1_i] \times [a^2_i, b^2_i]$. Soit d la distance euclidienne qui sépare les deux boîtes les plus proches l'une de l'autre. La σ^*-fonction χ_i définie par

$$\chi_i(x_1, x_2) = h^+_1(h^+_{d/2}(x_1, a^1_i) + h^-_{d/2}(x_1, b^1_i) + h^+_{d/2}(x_2, a^2_i) + h^-_{d/2}(x_2, b^2_i), 4)$$

vaut 1 sur B_i et 0 sur les $B_j, j \neq i$. Il suffit alors de définir g par

$$g(x_1, x_2) = \sigma(\sum_{i=1}^{n} \sigma(\sigma(\sigma(f_i(x_1, x_2))) + 2\chi_i(x_1, x_2) - 2) + n - 1)$$

où f_i désigne la fonction affine égale à f sur B_i. □

Corollaire 3.1 *On peut supposer que la fonction g_M dans l'énoncé du lemme 3.1 est une σ^*-fonction.*

Preuve: La fonction affine par morceaux g_M construite dans la preuve du lemme 3.1 est affine sur un nombre fini de pavés fermés disjoints : ces pavés sont les $B_{\alpha,\beta,q}$. Elle peut donc s'étendre en une σ^*-fonction hors de ces pavés par l'observation précédente. □

3.3 Problème de la contrôlabilité

Nous nous intéressons maintenant au problème de la contrôlabilité :

Définition 3.2 (Problème de la contrôlabilité) *On appelle* problème de la contrôlabilité *pour une classe $\mathcal{C}$ de systèmes dynamiques à temps discret contrôlés le problème de décision suivant :*

- *Instance : un système dynamique $(\mathbb{R}^d, Y, f)$ de la classe $\mathcal{C}$.*
- *Question : Existe-t'il un contrôle $u : \mathbb{N} \to Y$ tel que pour tout $x_0 \in \mathbb{R}^d$, la trajectoire partant de x_0 pour le contrôle u atteint l'origine ?*

De nouveau ce problème est indécidable :

Théorème 3.3 (Systèmes linéaires commutés) *Le problème de la contrôlabilité pour les systèmes linéaires commutés est indécidable.*

Ce résultat découle clairement de la proposition suivante prouvée par Paterson en 1970 dans [65] :

Proposition 3.1 ([65]) *Le problème de décision suivant est indécidable :*
- *Instance : un ensemble fini $F = \{A_1, \ldots, A_m\}$ de matrices 3×3 à coefficients entiers.*
- *Question : existe-t'il un entier k et des entiers $i_1, \ldots, i_k \in \{1, \ldots, m\}$ vérifiant l'équation matricielle $A_{i_1} \ldots A_{i_k} = 0$?*

Nous discuterons ce problème plus longuement dans le chapitre 4.

3.4 Problèmes de la stabilité

Nous nous intéressons maintenant à la décidabilité des problèmes de la stabilité :

Définition 3.3 (Problèmes de la stabilité) *On appelle* problèmes de la stabilité *pour une classe $\mathcal{C}$ de systèmes dynamiques à temps discret ou continu les problèmes de décision suivants :*
- *Instance : un système dynamique de la classe $\mathcal{C}$.*
- **Problème de la convergence globale** *: est-il vrai que pour tout $x_0 \in \mathbb{R}^d$ la trajectoire partant de x_0 converge vers l'origine ?*
- **Problème de la mortalité** *: est-il vrai que pour tout $x_0 \in \mathbb{R}^d$ la trajectoire partant de x_0 atteint l'origine ?*
- **Problème de la stabilité globale asymptotique** *: le système est-il globalement asymptotiquement stable ? Un système est dit* localement asymptotiquement stable *[84] si pour tout voisinage U de l'origine, il existe un autre voisinage V de l'origine tel que pour tout $x_0 \in V$, la trajectoire partant de x_0 converge vers l'origine sans quitter U. Un système est dit* globalement asymptotiquement stable *s'il est globalement convergent (pour tout $x_0 \in \mathbb{R}^d$ la trajectoire partant de x_0 converge vers l'origine) et s'il est localement asymptotiquement stable [84].*

3.4.1 Résultats

Dans le reste de ce chapitre, nous prouvons les résultats suivants :

Théorème 3.4 (Systèmes à temps discret) *Les problèmes de*
- *la convergence globale*
- *la mortalité*
- *la stabilité globale asymptotique*

pour
- *les systèmes affines par morceaux*

– *les systèmes continus affines par morceaux*
– *les réseaux de neurones*
sont indécidables.

En particulier, nous résolvons une conjecture exprimée à plusieurs reprise dans la littérature (voir [12, 83]) sur l'indécidabilité du problème de la stabilité pour les réseaux de neurones.

Une version préliminaire de ce travail (restreinte aux réseaux de neurones à fonction d'activation discontinue) a mené lieu à la soumission [10] cosignée avec Vincent Blondel, Pascal Koiran, Christos Papadimitriou et John Tsitsiklis. Les résultats présentés ici constituent une extension de ces résultats au problème original.

3.4.2 Principe de la preuve

Nous avons vu dans le lemme 3.1 qu'il était possible de simuler une machine de Turing par une fonction affine par morceaux. La première étape de notre preuve consiste à montrer que l'on peut simuler une machine de Turing par une fonction affine par morceaux dans un sens plus fort que celui du lemme 3.1.

La seconde étape consiste à utiliser un résultat prouvé par Hooper [39], qui stipule qu'il n'est pas possible de déterminer algorithmiquement si une machine de Turing s'arrête sur toute configuration, pour en déduire qu'il n'est pas possible de décider si un système dynamique $(\mathbb{R}^3, f)$ possède une trajectoire qui reste dans le sous-ensemble $\{0\} \times \mathbb{R}^2$.

Enfin la troisième et dernière étape consiste à réduire ce dernier problème aux problèmes de la stabilité pour les réseaux de neurones.

Nous détaillons maintenant chacune de ces étapes.

3.4.3 Simulation d'une machine de Turing

La première étape consiste donc à renforcer le lemme 3.1. Nous avons besoin de prouver le résultat suivant (intuitivement, dans ce lemme ν' est l'inverse de la fonction ν du lemme 3.1 dans le sens où $\nu'(\nu(c)) = c$ pour toute configuration c, $\mathcal{N}^\infty$ est l'image de ν, $\mathcal{N}^1$ correspond à un sous-ensemble de $[0,1]^2$ qui code des configurations correctes, $\mathcal{N}^1_{\neg acc}$ correspond au sous-ensemble de $\mathcal{N}^1$ des configurations non-terminales) ;

Lemme 3.2 *Soit M une machine de Turing à un ruban. Soit $C = \Sigma^\omega \times \Sigma^\omega \times Q$ l'espace de ses configurations.*

Alors il existe une σ^*-fonction $g_M : \mathbb{R}^2 \to \mathbb{R}^2$, une fonction d'encodage $\nu' : [0,1]^2 \to C$, et des sous-ensembles $\mathcal{N}^\infty \subset \mathcal{N}^1 \subset [0,1]^2$, $\mathcal{N}^1_{\neg acc} \subset \mathcal{N}^1$ tels que les conditions suivantes soient vérifiées :

1. *$g_M(\mathcal{N}^\infty) \subset \mathcal{N}^\infty$ et $\nu'(\mathcal{N}^\infty) = C$.*

2. *$\mathcal{N}^1_{\neg acc}$ (respectivement $\mathcal{N}^1$) est un produit cartésien de deux unions finies d'intervalles fermés de $\mathbb{R}$. $\mathcal{N}^1_{\neg acc}$ est à une distance strictement positive du point de coordonnées $(0,0)$ dans $\mathbb{R}^2$.*
3. *Pour $x \in \mathcal{N}^1$, la cconfiguration $\nu'(x)$ de M est non-terminale si et seulement si $x \in \mathcal{N}^1_{\neg acc}$.*
4. *Le diagramme suivant commute :*

$$\begin{array}{ccc} C & \xrightarrow{\vdash} & C \\ {\scriptstyle \nu'}\uparrow & & \uparrow{\scriptstyle \nu'} \\ \mathcal{N}^1 & \xrightarrow{g_M} & [0,1]^2 \end{array}$$

(c'est-à-dire $\nu'(x) \vdash \nu'(g_M(x))$ pour tout $x \in \mathcal{N}^1$).

Preuve: Considérons les notations et fonctions ν et g_M de la preuve du lemme 3.1. Comme mentionné dans le corollaire 3.1, nous pouvons supposer que la fonction g_M est une σ^*-fonction.

Nous voulons définir $\nu' : [0,1]^2 \to C$ de telle sorte que $\nu'(\nu(c)) = c$ pour tout $c \in C$. Nous le faisons de la façon suivante : on définit $pop : [0,1] \times \mathbb{N} \to \Sigma$ par

$$pop(x,j) = \begin{cases} k & \text{s'il existe } l \in \mathbb{N} \text{ et } k \in \Sigma \\ & \text{tels que } x - \frac{l}{(2n)^j} \in [\frac{(2k)}{(2n)^{j+1}}, \frac{(2k+1)}{(2n)^{j+1}}] \\ 0 & \text{sinon} \end{cases}$$

Observons que pour $x_i = \sum_{j=0}^{\infty}(2a_i^j)/(2n)^{j+1}$, on a $pop(x_i, j) = a_i^j$ pour tout $j \in \mathbb{N}$. On définit alors $\nu' : [0,1]^2 \to C$ par $\nu'(y_1/m, y_2) = (p_1, p_2, int(y_1))$, où $p_i = a_i^0 a_i^1 a_i^2 \ldots$ avec les a_i^j définis par $a_i^j = pop(fract(y_i), j)$: int et $fract$ désignent respectivement la partie entière et la partie fractionnaire. On a bien $\nu'(\nu(c)) = c$ pour tout $c \in C$.

Définissons $\mathcal{N}^1$ comme l'union des boîtes $B_{\alpha,\beta,q}$ pour $\alpha, \beta \in \Sigma, q \in Q$. Définissons $\mathcal{N}^1_{\neg acc}$ comme l'union des boîtes $B_{\alpha,\beta,q}$ pour $\alpha, \beta \in \Sigma$ et pour $q \in Q$ différent de l'état interne acceptant 0 de M.

On a bien alors $\nu'(x) \vdash \nu'(g_M(x))$ pour tout $x \in \mathcal{N}^1$.

Posons $\mathcal{N}^\infty = \nu(C)$. On a bien $g_M(\mathcal{N}^\infty) \subset \mathcal{N}^\infty$ et puisque $\nu'(\nu(c)) = c$, on a bien $\nu'(\mathcal{N}^\infty) = C$. $0 \in Q$ étant supposé être un état interne acceptant, le point de coordonnées $(0,0)$ n'appartient pas à $\mathcal{N}^1_{\neg acc}$ et est donc à une distance strictement positive de cet ensemble. □

3.4.4 Résultat de Hooper

Nous utilisons maintenant le résultat prouvé par Hooper [39] pour en déduire qu'il n'est pas possible de décider si un système dynamique $(\mathbb{R}^3, f)$ possède une trajectoire qui reste dans le sous-ensemble $\{0\} \times \mathbb{R}^2$.

Le résultat précis prouvé par Hooper en 1966 est le suivant (voir [39]) :

Théorème 3.5 ([39]) *Le problème de décision suivant est indécidable :*
- *Instance : Une machine de Turing M.*
- *Question : M possède-t'elle une configuration non mortelle ?*

En d'autres termes, il n'est pas possible de déterminer algorithmiquement si une machine de Turing s'arrête sur toute configuration.

Remarquons que ce résultat et le lemme 3.2 suffisent pour prouver l'indécidabilité des problèmes de la stabilité pour les systèmes affines par morceaux lorsque l'on ne leur impose pas d'être continus. En effet :

Corollaire 3.2 *Les problèmes de*
- *la convergence globale*
- *la mortalité*
- *la stabilité globale asymptotique*

pour les systèmes discontinus affines par morceaux de dimension 2 sont indécidables.

Preuve: Nous réduisons tout d'abord le problème du théorème 3.5 au problème de la mortalité. Supposons qu'une machine de Turing M soit donnée. Notons g'_M la fonction discontinue affine par morceaux égale à la fonction g_M du lemme 3.2 sur $\mathcal{N}^1_{\neg acc}$, et égale à 0 ailleurs. Nous affirmons que la machine de Turing M possède une configuration non mortelle si et seulement si la fonction g'_M possède une trajectoire non mortelle : c'est-à-dire si et seulement s'il existe une suite $x_{t+1} = g'_M(x_t)$ avec $x_t \neq 0$ pour tout $t \in \mathbb{N}$.

Supposons qu'une telle trajectoire existe. Puisque g'_M est nulle en dehors de $\mathcal{N}^1_{\neg acc}$, nous avons $x_t \in \mathcal{N}^1_{\neg acc}$ pour tout $t \in \mathbb{N}$. Par le diagramme commutatif du lemme 3.2, la suite $c_t = \nu'(x_t)$ est une suite de configurations consécutives de M. Par la condition 3 du lemme 3.2, aucune des configurations c_t n'est terminale. La configuration c_0 est donc non mortelle.

Réciproquement, supposons que M possède une configuration non mortelle : il existe une suite infinie de configurations non-terminales avec $c_{t+1} \vdash c_t$. Par la condition 1 du lemme 3.2, il existe $x_0 \in \mathcal{N}^\infty$ avec $\nu'(x_0) = c_0$. Nous affirmons que la trajectoire $x_{t+1} = g'_M(x_t)$ est non mortelle : en effet, puisque $g_M(\mathcal{N}^\infty) \subset \mathcal{N}^\infty$, nous avons $x_t \in \mathcal{N}^\infty$ pour tout t, et par le diagramme commutatif, nous obtenons $\nu'(x_t) = c_t$ pour tout t. La configuration c_t étant non-terminale pour tout t, la condition 3 du lemme 3.2 nous dit que $x_t \in \mathcal{N}^1_{\neg acc}$ pour tout t et la condition 2 du lemme 3.2 nous dit que x_t ne saurait s'annuler.

L'indécidabilité des problèmes de la convergence globale et de la stabilité globale asymptotique découle des remarques suivantes : d'une part, par ce qui précède, une trajectoire non mortelle de g'_M ne converge pas vers 0 puisqu'elle reste dans $\mathcal{N}^1_{\neg acc}$ à distance positive de l'origine. D'autre part, toute trajectoire mortelle de g'_M satisfait $x_t = 0$ à partir d'un certain moment, puisque 0 est un point fixe de g'_M. En d'autres termes, pour le système dynamique $(\mathbb{R}^2, g'_M)$ les propriétés

de mortalité, de convergence globale et de stabilité globale asymptotique sont équivalentes. □

Maintenant, lorsque l'on impose aux systèmes dynamiques d'être à fonction de transition continue, le théorème 3.5 n'est pas suffisant pour prouver directement l'indécidabilité des problèmes de la stabilité mais il permet de prouver qu'il n'est pas possible de décider si un système dynamique $(\mathbb{R}^3, f)$ possède une trajectoire qui reste dans le sous-ensemble $\{0\} \times \mathbb{R}^2$.

Avant cela, nous avons besoin d'un résultat technique :

Observation 3.2 *Soit P un sous-ensemble de $\mathbb{R}^2$ qui s'écrit comme un produit cartésien de deux unions finies d'intervalles fermés de $\mathbb{R}$.*

Alors il existe une σ^*-fonction $Z_P : \mathbb{R}^2 \to \mathbb{R}$ telle que, pour tout x,

- *si $x \in P$ alors $Z_P(x) = 0$*
- *si $x \notin P$, alors $Z_P(x) > 0$.*

Preuve: Comme dans l'observation 3.1, lorsque ϵ est un réel positif, et a un réel, appelons $h^+_\epsilon(x, a)$ la fonction définie par $h^+_\epsilon(x, a) = \sigma(1 + 2/\epsilon(x - a)))$ et $h^-_\epsilon(x, a)$ la fonction définie par $h^-_\epsilon(x, a) = h^+_\epsilon(-x, -a)$.

Soit I un intervalle ouvert du type $I =]a, b[$. La fonction $\chi_I(x) = -h^+_{(b-a)/2}(x, b) - h^-_{(b-a)/2}(x, a)$ vaut O pour $x \notin I$ et une valeur strictement positive pour $x \in I$.

Soit I un intervalle ouvert du type $I =]a, \infty[$. La fonction $\chi_I(x) = 1 + h^+_1(x, a + 1)$ vaut 0 pour $x \notin I$ et une valeur strictement positive pour $x \in I$.

Soit I un intervalle ouvert du type $I =]-\infty, a[$. La fonction $\chi_I(x) = 1 + h^-_1(x, a - 1)$ vaut 0 pour $x \notin I$ et une valeur strictement positive pour $x \in I$.

Lorsque $J = \cup^n_{i=1} I_i$ est une union finie d'intervalles fermés de $\mathbb{R}$, le complémentaire J^c de J dans $\mathbb{R}$ s'écrit comme une union finie d'intervalles ouverts : posons $J^c = \cup^n_{i=1} I_i$. Définissons la fonction Z_J par $Z_J(x) = \sum^n_{i=1} \chi_{I_i}(x)$. Cette fonction est nulle pour $x \in J$ et possède une valeur strictement positive pour $x \notin J$.

Maintenant si P s'écrit $P = J_1 \times J_2$ il suffit d'écrire $Z_P(x_1, x_2) = \sigma(Z_{J_1}(x_1) + Z_{J_2}(x_2))$. □

Nous sommes prêts à prouver qu'il n'est pas possible de décider si un système dynamique $(\mathbb{R}^3, f)$ possède une trajectoire qui reste dans le sous-ensemble $\{0\} \times \mathbb{R}^2$:

Lemme 3.3 *Le problème de décision suivant est indécidable :*

- *Instance : un système dynamique $(\mathbb{R}^3, f)$ où f est une σ^*-fonction.*
- *Question : le système possède-t'il une trajectoire $x_{t+1} = f(x_t)$ qui appartienne à $\{0\} \times \mathbb{R}^2$ pour tout t ?*

Preuve: Nous réduisons le problème du théorème 3.5 à ce problème.

Supposons qu'une machine de Turing M soit donnée. Considérons la σ^*-fonction $f : \mathbb{R}^3 \to \mathbb{R}^3$ définie par

$$f(x_1, x_2, x_3) = \begin{pmatrix} \sigma(\sigma(Z_{\mathcal{N}^1_{\neg acc}}(x_2, x_3))) \\ g_M(x_2, x_3) \end{pmatrix}$$

où les fonctions $g_M, \mathcal{N}^1_{\neg acc}$ et $Z_{\mathcal{N}^1_{\neg acc}}$ sont définies dans les lemmes 3.2 et l'observation 3.2 pour $P = \mathcal{N}^1_{\neg acc}$.

Pour éviter certains problèmes de notations, nous écrirons $(x^1, \ldots, x^d)$ les composantes d'un point x de $\mathbb{R}^d$.

Nous prouvons que le système $(\mathbb{R}^3, f)$ possède une trajectoire $x_{t+1} = f(x_t)$ avec $x^1_t = 0$ pour tout t si et seulement si la machine de Turing M possède une configuration non mortelle.

Supposons que le système possède une telle trajectoire. Puisque $Z_{\mathcal{N}^1_{\neg acc}}$, et donc aussi $\sigma(\sigma(Z_{\mathcal{N}^1_{\neg acc}}))$, est strictement positive en dehors de $\mathcal{N}^1_{\neg acc}$, cela signifie que pour tout $t \geq 0$, on doit avoir $(x^2_t, x^3_t) \in \mathcal{N}^1_{\neg acc}$. En utilisant le diagramme commutatif du lemme 3.2, cela signifie que $\nu'(x^2_t, x^3_t)$, $t \in \mathbb{N}$, est une suite de configurations successives de M. Par la condition 3 du lemme 3.2 aucune de ces configurations n'est terminale : c'est-à-dire $c_0 = \nu'(x^2_0, x^3_0)$ est une configuration non mortelle de M.

Réciproquement, supposons que M possède une configuration non mortelle : il existe une suite infinie de configurations non-terminales avec $c_t \vdash c_{t+1}$. Par la condition 1 du lemme 3.2, il existe un point $(x^2_0, x^3_0) \in \mathcal{N}^\infty$ tel que $\nu'(x^2_0, x^3_0) = c_0$. Considérons la suite définie par $(x^2_{t+1}, x^3_{t+1}) = g_M(x^2_t, x^3_t)$ pour tout t. Puisque $g_M(\mathcal{N}^\infty) \subset \mathcal{N}^\infty$, nous avons $(x^2_t, x^3_t) \in \mathcal{N}^\infty$ pour tout $t \geq 0$. La configuration c_t n'étant pas terminale, par la propriété 4 du lemme 3.2 nous avons $(x^2_t, x^3_t) \in \mathcal{N}^1_{\neg acc}$ pour tout $t \geq 0$. Ceci implique précisément que la suite $x_t = (0, x^2_t, x^3_t)$, $t \in \mathbb{N}$, est une trajectoire de $(\mathbb{R}^3, f)$. □

3.4.5 Réduction aux problèmes de la stabilité

Nous en arrivons maintenant à la troisième et dernière étape de notre preuve qui consiste à réduire le problème du lemme 3.3 aux problèmes de la stabilité.

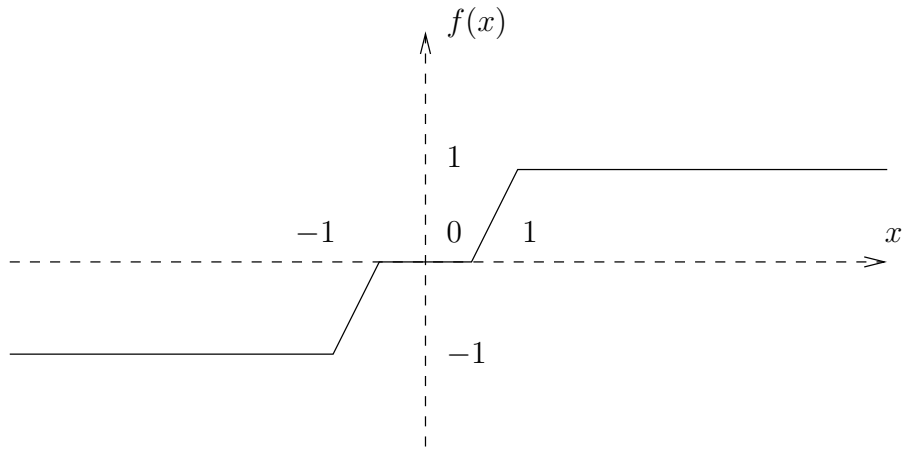
Rappelons qu'une σ-fonction est une fonction du type $x \mapsto \sigma(Ax + b)$: voir la définition 2.9. Nous dirons qu'une telle fonction est une *σ_0-fonction* si b est le vecteur nul : c'est-à-dire si la fonction est la composée de σ et d'une application linéaire. Nous appellerons *σ^*_0-fonction* une fonction qui s'écrit comme la composée d'un nombre fini de σ_0-fonctions.

Commençons par le point technique suivant :

fig :fLa fonction $f(x) = \sigma(2\sigma(x) - \sigma(2x))$

Lemme 3.4 *Il existe deux σ^*_0-fonctions Abs, Sel de $\mathbb{R}^2$ dans $\mathbb{R}$ telles que :*

- *Pour tout $e \in [0, 1]$:*
  - *Pour $z \in [-1, 0]$, on a $Abs(z, e) = 0$.*



  - *Pour $z \in [0,1]$, on a $Abs(z,e) \in [0,1]$.*
  - *Pour $z = 1$, on a $Abs(1,e) = e$.*
- *Pour tout $e \in [-1,1]$:*
  - *$Sel(0,e) = 0$.*
  - *$Sel(1,e) = e$.*
  - *$Sel(-1,e) = e$.*

Preuve: Si l'on pose $f(x) = \sigma(2\sigma(x) - \sigma(2x))$, on peut vérifier que $Abs(z,e) = f(z/2 + e/2)$ et $Sel(z,e) = \sigma(2f(3z/4 + e/4) - z)$ sont solutions : voir la figure **??**. □

Maintenant, la construction suivante est celle qui nous permettra de réduire le problème indécidable du lemme 3.3 aux problèmes de la stabilité :

Lemme 3.5 *Il existe une σ_0^*-fonction $Stab : \mathbb{R} \times [0,1] \to \mathbb{R}$ telle que le système dynamique contrôlé $(\mathbb{R}^2, [0,1], Stab)$ vérifie la propriété suivante : toute trajectoire $z_{t+1} = Stab(z_t, e_t)$, $t \in \mathbb{N}$, du système partant d'un point $z_0 \in [-1,1]$ pour un contrôle $e_t : \mathbb{N} \to [0,1]$ est dans l'un des trois cas exclusifs suivants :*

- *La suite z_t est constante de valeur 1. Ce cas ne se produit que si $z_0 = 1$ et si $e_t = 0$ pour tout t.*
- *La suite z_t est constante de valeur -1. Ce cas ne se produit que si $z_0 = -1$ et si $e_t = 0$ pour tout t.*
- *La suite z_t est ultimement nulle : il existe t_0 tel que $z_t = 0$ pour tout $t \geq t_0$.*

Preuve: Remarque 1 : la fonction $f(x) = \sigma(2\sigma(x) - \sigma(2x))$ possède les points fixes instables -1 et 1 et toute suite du type $x_{t+1} = f(x_t)$ avec $x_0 \notin \{-1, 1\}$ atteint ultimement le point fixe stable 0 : voir la figure **??**.

Remarque 2 : posons $F(z,e) = f(z - Abs(z,e)/2)$. Pour tout $e \in [0,1]$, $z \in [-1,1]$ on a $F(z,e) = f(z)$ pour $z \leq 0$, et $0 \leq F(z,e) \leq f(z)$ pour $z \geq 0$: en effet pour $z \leq 0$ on a $Abs(z,e) = 0$ et pour $z \geq 0$, la fonction f est croissante et on a $-1/2 \leq z - Abs(z,e)/2 \leq z$ avec $f(-1/2) = 0$.

Remarque 3 : posons $Stab(z,e) = F(-F(-z,e),e)$. Alors $0 \leq Stab(z,e) \leq f(f(z))$ pour $z \geq 0$ et $f(f(z)) \leq Stab(z,e) \leq 0$ pour $z \leq 0$: en effet, lorsque z est positif, on a $-F(-z,e) = -f(-z) = f(z) \geq 0$ d'où l'on tire $0 \leq Stab(z,e) \leq f(f(z))$, et lorsque z est négatif, on a $0 \geq -F(-z,e) \geq -f(-z) = f(z)$, d'où l'on tire $Stab(z,e) = f(-F(-z,e)) \geq f(f(z))$ en utilisant la croissance de la fonction f.

Remarque 4 : $z \in \{-1,1\}$, $e \in [0,1]$ et $Stab(z,e) \in \{-1,1\}$ implique $e = 0$: on a d'une part $Stab(1,e) = F(-F(-1,e),e) = F(-f(-1),e) = F(1,e) = f(1 - Abs(1,e)/2) = f(1-e/2)$ et d'autre part $Stab(-1,e) = F(-F(1,e),e) = f(-F(1,e)) = f(-f(1-e/2)) = -f(f(1-e/2))$. La fonction f prenant les valeurs $\{-1,1\}$ seulement en -1 et 1, les deux cas se ramènent à $f(1-e/2) \in \{-1,1\}$. Puisque la fonction f est positive sur $[1/2,1]$, seul le cas $f(1-e/2) = f(1) = 1$ est possible. La fonction f étant strictement croissante sur $[1/2,1]$ ce cas implique $e = 0$.

En utilisant les remarques précédentes, nous prouvons maintenant que la fonction $Stab$ est solution : par la remarque 3 et la remarque 1, il est clair que s'il existe un t avec $|z_t| < 1$, alors la suite z_t est ultimement nulle.

Supposons maintenant que $|z_t| = 1$ pour tout $t \geq 0$. La remarque 4 implique que l'on doive avoir $e_t = 0$ pour tout t. En observant que $Stab(1,0) = 1$ et que $Stab(-1,0) = -1$, cela signifie que la suite z_t est la suite constante $z_t = z_0$. On a alors $z_0 \in \{-1,1\}$ et $e_t = 0$ pour tout t. □

Nous en déduisons alors le résultat suivant :

Lemme 3.6 *Les problèmes de*
- *la convergence globale*
- *la mortalité*
- *la stabilité globale asymptotique*

pour les systèmes dynamiques du type $(\mathbb{R}^4, f)$ où f est une σ_0^*-fonction sont indécidables.

Preuve: Nous réduisons tout d'abord le problème du lemme 3.3 au problème de la mortalité pour ces systèmes.

Supposons qu'un système dynamique $(\mathbb{R}^3, f)$ où f est une σ^*-fonction soit donné. La fonction f, en tant que σ^*-fonction, s'écrit $f = f_k \circ f_{k-1} \circ \ldots \circ f_1$, où les $f_j = \sigma(A_j x + b_j)$ sont des σ-fonctions. Puisque l'on a $\sigma(\sigma(x)) = \sigma(x)$ pour tout x, quitte à composer la fonction f à gauche par des fonctions σ, nous supposer l'entier k supérieur ou égal à 5.

Définissons $f' : \mathbb{R}^4 \to \mathbb{R}^4$ par $f' = f'_k \circ f'_{k-1} \circ \ldots \circ f'_1$ où $f'_j(x,z) = \sigma(A_j x + b_j z)$ (x est un vecteur de $\mathbb{R}^3$ et z est un réel dans cette définition) de telle sorte que l'on ait $f(x) = f'(x,1)$.

Posons $\sigma^{(0)} = \sigma$ et $\sigma^{(n+1)} = \sigma \circ \sigma^{(n)}$ pour tout entier n. Considérons le système dynamique $(\mathbb{R}^4, f'')$ où f'' est la σ_0^*-fonction définie par :

$$f''(x,z) = \begin{pmatrix} Sel(\sigma^{(k)}(z), f'(x,z)) \\ Stab(\sigma^{(k-5)}(z), \sigma^{(k-5)}(x^1)) \end{pmatrix}$$

pour $x = (x^1, x^2, x^3) \in \mathbb{R}^3$ et $z \in \mathbb{R}$.

Nous prétendons que (X'', f'') possède une trajectoire non mortelle (c'est-à-dire qui n'atteint pas l'origine) si et seulement si $(\mathbb{R}^3, f)$ possède une trajectoire $x_{t+1} = f(x_t)$ où la première composante x_t^1 de x_t vérifie $x_t^1 = 0$ pour tout t.

Supposons que $(\mathbb{R}^3, f)$ possède une trajectoire $x_{t+1} = f(x_t)$ avec $x_t^1 = 0$ pour tout t. La trajectoire $(x_t, 1)$ est une trajectoire de $(\mathbb{R}^4, f'')$ car on a

$$\begin{aligned} f''(x_t, 1) &= (Sel(\sigma^{(k)}(1), f'(x_t, 1)), Stab(\sigma^{(k-5)}(1), \sigma^{(k-5)}(x_t^1))) \\ &= (Sel(1, f'(x_t, 1)), Stab(1, x_t^1)) \\ &= (f(x_t), 1) \\ &= (x_{t+1}, 1) \end{aligned}$$

Cette trajectoire est non mortelle puisque sa dernière composante reste toujours égale à 1.

Réciproquement, supposons que $(\mathbb{R}^4, f'')$ possède une trajectoire $x''_{t+1} = f''(x''_t)$ non mortelle. Notons $x''_t = (x_t^{''1}, x_t^{''2}, x_t^{''3}, x_t^{''4})$. Par le lemme 3.5, la suite $x_t^{''4}$ est soit constante de valeur 1, soit constante de valeur -1, soit ultimement nulle. Le dernier cas ne peut pas se produire car s'il existe t tel que $x_t^{''4} = 0$, on a

$$\begin{aligned} x''_{t+1} &= (Sel(\sigma^{(k)}(0), f'(x_t^{''1}, x_t^{''2}, x_t^{''3})), Stab(\sigma^{(k-5)}(0), \sigma^{(k-5)}(x_t^{''1}))) \\ &= (Sel(0, f'(x_t^{''1}, x_t^{''2}, x_t^{''3})), Stab(0, \sigma^{(k-5)}(x_t^{''1}))) \\ &= (0, 0) \end{aligned}$$

La suite $x_t^{''4}$ est donc constante de valeur 1 ou -1, et par le lemme 3.5, on sait que pour que cela se produise, il faut que $x_t^{''1} = 0$ pour tout t. On peut supposer sans perte de généralité que $x_t^{''4} = 1$ pour tout t (sinon il suffit de considérer la suite $-x''_t$ en lieu et place de x''_t qui est aussi une trajectoire de (X'', f'') puisque la fonction f'', en tant que σ_0^*-fonction, est nécessairement impaire.). Le système $(\mathbb{R}^3, f)$ possède la trajectoire $x_t = (x^{''1}, x^{''2}, x^{''3})$ avec $x_t^1 = 0$ pour tout t : en effet, $x_{t+1} = Sel(\sigma^{(k)}(x^{''4}), f'(x_t, x^{''4})) = Sel(1, f'(x_t, 1)) = f'(x_t, 1) = f(x_t)$ et $x_t^1 = x^{''1}$ est nul pour tout $t \geq 0$.

L'indécidabilité des problèmes de la convergence globale et de la stabilité globale asymptotique découle des remarques suivantes : d'une part, par ce qui précède une trajectoire non mortelle de $(\mathbb{R}^4, f'')$ ne converge pas vers 0 puisque sa dernière composante reste dans $\{-1, 1\}$. D'autre part, toute trajectoire mortelle de $(\mathbb{R}^4, f'')$ satisfait $x_t = 0$ à partir d'un certain moment, puisque l'origine est un point fixe de f''. En d'autres termes, pour le système dynamique $(\mathbb{R}^4, f'')$, les propriétés de mortalité, de convergence globale et de stabilité globale asymptotique sont équivalentes.

□

Nous obtenons alors le résultat principal de cette section :

Théorème 3.6 (Systèmes à temps discret) *Les problèmes de*
- *la convergence globale*
- *la mortalité*
- *la stabilité globale asymptotique*

pour
- *les systèmes affines par morceaux*
- *les systèmes continus affines par morceaux*
- *les réseaux de neurones*

sont indécidables.

Preuve: L'indécidabilité de ces problèmes pour les systèmes (continus) affines par morceaux est clairement le lemme précédent.

Il nous reste à montrer l'indécidabilité des problèmes de la stabilité pour les réseaux de neurones : nous réduisons les problèmes du lemme 3.6 aux problèmes équivalents pour les réseaux de neurones.

Soit $(\mathbb{R}^4, f)$ un système dynamique où f est une σ_0^*-fonction. $f : \mathbb{R}^4 \to \mathbb{R}^4$, en tant que σ_0^*-fonction s'écrit $f = f_k \circ f_{k-1} \circ \ldots \circ f_1$, où les fonctions $f_j(x) = \sigma(A_j x)$ sont des σ_0-fonctions de $\mathbb{R}^{d_j}$ dans $\mathbb{R}^{d_{j-1}}$, pour des entiers $d_0, d_1, \ldots, d_k$ avec $d_0 = d_k = 4$.

Considérons le réseau de neurones $(\mathbb{R}^d, f')$ avec $d = d_0 + d_1 + \ldots + d_k$ et la fonction f' définie par $f'(x) = \sigma(Ax)$ où :

$$A = \begin{pmatrix} 0 & 0 & \ldots & 0 & A_k \\ A_1 & 0 & \ldots & 0 & 0 \\ 0 & A_2 & \ldots & 0 & 0 \\ \vdots & \vdots & & 0 & 0 \\ 0 & 0 & \ldots & A_{k-1} & 0 \end{pmatrix}$$

Ce réseau de neurones simule clairement les itérations de la fonction f.

Supposons que (X', f') soit mortel (respectivement globalement convergent, globalement asymptotiquement stable). Alors $(\mathbb{R}^4, f)$ l'est aussi : en effet, lorsque $x_{t+1} = f(x_t)$ est une trajectoire de $(\mathbb{R}^4, f)$ alors la suite $(x_t, f_1(x_t), \ldots, f_{k-1} \circ \ldots \circ f_1(x_t))$ est une suite extraite d'une trajectoire de (X', f').

Supposons que $(\mathbb{R}^4, f)$ soit mortel (respectivement globalement convergent, globalement asymptotiquement stable). Alors (X', f') l'est aussi : lorsque $x'_{t+1} = f'(x_t)$ est une trajectoire de (X', f'), écrivons $x'_t = (y_t^1, \ldots, y_t^k)$, où chacun des y^j est dans $\mathbb{R}^{d_{j-1}}$. La suite y_t^1, $t \in \mathbb{N}$ s'annule ultimement (respectivement converge vers 0) car pour chaque $t_0 \in \{0, \ldots, d-1\}$, la suite $t \mapsto y^1_{t_0+dt}$ est une trajectoire de (X, f). Pour $j > 1$, chacune des suites y_t^j, $t \in \mathbb{N}$, s'annule ultimement (respectivement converge vers 0) car y_t^j est l'image de y_t^1 par la fonction continue $f_{j-1} \circ \ldots \circ f_1$ qui envoie 0 sur 0. □

CHAPITRE

4 Problèmes en basses dimensions

4.1 Introduction

Ce chapitre étudie la décidabilité des problèmes de l'atteignabilité, de la contrôlabilité et de la stabilité pour les systèmes dynamiques de basses dimensions. Il précise la frontière connue entre décidabilité et indécidabilité pour les systèmes de faibles dimensions. Dans la section 4.2, nous discutons le problème de l'atteignabilité. Dans la section 4.3, nous étudions plus longuement le problème de la contrôlabilité. Enfin, dans la section 4.4, nous discutons les problèmes de la stabilité.

4.2 Problème de l'atteignabilité

Nous nous intéressons maintenant au problème de l'atteignabilité : dans la section 4.2.1 nous rappelons les résultats connus dans la littérature, puis dans la section 4.2.2 nous discutons le problème de l'atteignabilité pour les systèmes affines par morceaux de dimension 1.

4.2.1 Résultats connus

Dans cette section, nous rappelons les résultats connus dans la littérature pour chacun des systèmes dynamiques introduits dans le chapitre 2 : c'est-à-dire pour les systèmes affines par morceaux, pour les réseaux de neurones et pour les systèmes DCPM.

Commençons par les systèmes affines par morceaux. Le problème de l'atteignabilité est indécidable pour les systèmes affines par morceaux de dimension 2. En effet, la preuve du théorème 3.1 présentée dans le chapitre précédent permet d'affirmer :

Théorème 4.1 (Systèmes affines par morceaux [45]) *Les problèmes de l'atteignabilité*

- *pour les systèmes affines par morceaux de dimension* 2
- *pour les systèmes continus affines par morceaux de dimension* 2

sont indécidables.

Toutefois, on ne connaît pas la réponse à la question suivante (voir la section 4.2.2 ou [44, 45] pour une discussion) :

Problème 4.1 *Le problème de l'atteignabilité pour les systèmes affines par morceaux de dimension* 1 *est-il décidable ?*

Pour les réseaux de neurones, la meilleure borne connue est le résultat suivant présenté dans [45] :

Théorème 4.2 (Réseaux de neurones [45]) *Le problème de l'atteignabilité pour les réseaux de neurones de dimension* 96 *est indécidable.*

Toutefois, on ne connaît pas la réponse à la question suivante :

Problème 4.2 *Le problème de l'atteignabilité pour les réseaux de neurones de dimension* 95 *est-il décidable ?*

Notons que si l'on autorise les *réseaux de neurones d'ordre supérieur*, on sait prouver que la dimension 25 suffit : voir [41]. Si on utilise des neurones *affines-commutés*, on peut prouver que la dimension 9 suffit : voir [56].

Pour les systèmes DCPM, on sait précisément caractériser les dimensions pour lesquelles le problème de l'atteignabilité est indécidable. En effet, les résultats suivants sont prouvés dans [9] :

Théorème 4.3 ([9]) *Le problème de l'atteignabilité pour les systèmes DCPM bien définis est*
- *décidable en dimension* $d \leq 2$
- *indécidable en dimension* $d \geq 3$.

4.2.2 Problème de l'atteignabilité en dimension 1

Nous discutons maintenant brièvement le problème 4.1 : c'est-à-dire la décidabilité du problème de l'atteignabilité pour les systèmes affines par morceaux de dimension 1.

Mentionnons tout d'abord qu'il a été prouvé dans [44] qu'il n'était pas possible de simuler une machine de Turing par une fonction affine par morceaux de dimension 1. Mais la notion de simulation utilisée dans [44] est très forte et ce résultat n'implique en rien la décidabilité du problème 4.1 : voir [44].

Le problème 4.1 peut aussi se formuler de la façon suivante :

Problème 4.3 (Equivalent au problème 4.1) *Le problème de décision suivant est-il décidable ?*

- *Instance : une fonction affine par morceaux de dimension* 1.
- *Question : la trajectoire du système dynamique* $(\mathbb{R}, f)$ *partant de* 0 *atteint-elle le point* 1.

Mais notre méconnaissance des problèmes 4.1 et 4.3 est très grande car nous ne savons même pas répondre à la question suivante : appelons *fonction affine à deux morceaux* une fonction affine par morceaux de $\mathbb{R}$ dans $\mathbb{R}$ affine sur $]-\infty, u]$ et affine sur $]u, +\infty[$ pour un rationnel $u \in \mathbb{Q}$.

Problème 4.4 *Le problème de décision suivant est-il décidable ?*
- *Instance : une fonction affine à deux morceaux.*
- *Question : la trajectoire du système dynamique* $(\mathbb{R}, f)$ *partant de* 0 *atteint-elle le point* 1 *?*

4.3 Problème de la contrôlabilité

Nous étudions maintenant le problème de la contrôlabilité des systèmes linéaires commutés en basses dimensions.

Le travail présenté dans cette section est le fruit d'une collaboration avec Michael S. Branicky qui a mené à la soumission [21] et à la présentation [22] dans un livre de problèmes ouverts.

4.3.1 Problème de la mortalité

Dans le chapitre 3, nous avons prouvé le résultat suivant :

Théorème 4.4 *Le problème de la contrôlabilité pour les systèmes linéaires commutés de dimension* 3 *est indécidable.*

Le problème de la contrôlabilité pour les systèmes linéaires commutés de dimension 1 est facilement décidable. Mais pour la dimension 2 le problème est ouvert :

Problème 4.5 *Le problème de la contrôlabilité des systèmes linéaires commutés de dimension* 2 *est-il décidable ?*

Formulons autrement ce problème : un ensemble fini $F = \{A_1, \ldots, A_m\}$ de matrices $d \times d$ est dit *mortel* s'il existe un entier $k \geq 1$ et des entiers $i_1, \ldots, i_k$ dans $\{1, 2, \ldots, m\}$ tels que $A_{i_1} \ldots A_{i_k} = 0$. Le problème 4.5 peut se reformuler de la façon suivante :

Problème 4.6 (équivalent au problème 4.5) *Le problème de décision* $Mort_{\mathbb{Q}}(2)$ *suivant est-il décidable ?*

– *Instance : un ensemble fini F de matrices 2×2 à coefficients rationnels.*
– *Question : F est-il mortel ?*

La question de la décidabilité éventuelle de $Mort_{\mathbb{Q}}(2)$ a reçu beaucoup d'intérêt dans la littérature : voir par exemple [48, 49] pour des discussions et des références. Historiquement, cette question a été soulevée pour la première fois par [75] en les termes suivants "construire un algorithme qui, étant donné un ensemble fini H de transformations linéaires non-singulières du plan complexe et des droites L et M contenant l'origine, détermine s'il existe une composition des applications de H qui envoie L sur M ?". Mais, à ce jour, aucune preuve de la décidabilité de $Mort_{\mathbb{Q}}(2)$ ou de son indécidabilité n'a été produite.

Observons que dans cette section nous étudierons exclusivement ce problème, mais que plusieurs autres problèmes matriciels ont été prouvés indécidables dans la littérature, que ces résultats concernent tous des ensembles de matrices 3×3 (voir [29, 47] par exemple), mais que leur décidabilité/indécidabilité pour les matrices 2×2 est à chaque fois laissée sans réponse : voir [29]. Ces autres problèmes incluent par exemple la question de savoir si le semi-groupe généré par un ensemble fini de matrices contient une matrice avec un zéro dans le coin en haut à droite [54] ou la question de savoir si le semi-groupe généré par un ensemble fini de matrices est libre [37].

Afin de pouvoir étudier les liens qui existent entre le nombre de matrices et leur dimension, $Mort_K(d)$ et $Mort_K(d, m)$ désigneront les problèmes suivants :

Définition 4.1 (Problèmes $Mort_K(d)$ et $Mort_K(d, m)$) *Soit $K \in \{\mathbb{N}, \mathbb{R}\}$.*
Le problème $Mort_K(d)$ désigne le problème de décision suivant :
– *Instance : un ensemble fini F de matrices $d \times d$ à coefficients dans K.*
– *Question : F est-il mortel ?*
Le problème $Mort_K(d, m)$ désigne le problème de décision suivant :
– *Instance : un ensemble fini F constitué de m matrices $d \times d$ à coefficients dans K.*
– *Question : F est-il mortel ?*

Le plan du reste de la section 4.3 est le suivant : dans la section 4.3.2, nous étudions les liens qui existent entre le nombre de matrices et leur dimension : nous prouvons que les problèmes $Mort_{\mathbb{Q}}(3, 9)$ et $Mort(27, 2)$ sont indécidables.

Dans la section 4.3.3, nous étudions la décidabilité du problème lorsqu'on le restreint à deux matrices : nous prouvons que $Mort_{\mathbb{Q}}(2, 2)$ est décidable mais que $Mort_{\mathbb{R}}(2, 2)$ est indécidable pour le modèle de Blum Shub et Smale.

Dans la section 4.3.4, nous relions les problèmes 4.5 et 4.6 à d'autres problèmes de la littérature : nous relions ces problèmes au problème de l'égalité des coefficients, au problème du zéro en haut à gauche, et aux problèmes de la section 4.2.

4.3.2 Liens entre dimension et nombre de matrices

Nous étudions maintenant les liens qui existent entre le nombre de matrices et la dimension des matrices : nous prouvons que $Mort_{\mathbb{Q}}(3,9)$ et $Mort(27,2)$ sont indécidables.

Le principe consiste à revenir sur la preuve de la proposition 3.1 présentée dans [65]. En effet, en modifiant cette preuve on peut obtenir :

Proposition 4.1 *Supposons que le problème de la correspondance de Post soit indécidable avec p règles. Alors le problème $Mort_{\mathbb{Q}}(3,p+2)$ est indécidable.*

Preuve: Le problème de la correspondance de Post (PCP) est le problème de décision suivant :

- Instance : un ensemble fini de couples de mots $\{< U_i, V_i > | i = 1 \dots p\}$
- Question : existe-t'il une suite non vide d'indices $i_1, i_2, \dots, i_k$ dans $\{1, 2, \dots, p\}$ telle que $U_{i_1}U_{i_2}\dots U_{i_k} = V_{i_1}V_{i_2}\dots V_{i_k}$?

La preuve de [65] prouve que si le problème de la correspondance de Post est indécidable avec p règles alors le problème $Mort_{\mathbb{Q}}(3,2p+2)$ est indécidable : [65] prouve d'une part que pour toute instance $\{< U_i, V_i > | i = 1 \dots p\}$ de PCP on peut construire un ensemble de matrices à coefficients rationnels $F = \{S, T, W(U_j, V_j), W(U_j, \underline{1V_j})$ pour $j = 1, \dots, p\}$ qui soit mortel si et seulement s'il existe des entiers $i_1, i_2, \dots, i_k$ dans $\{1, 2, \dots, p\}$ tels que $SW(U_{i_1}, \underline{1V_{i_1}})W(U_{i_2}, V_{i_2}) \dots W(U_{i_k}, V_{i_k})T = 0$ et que, d'autre part, les matrices de F sont telles que $SW(U_{i_1}, \underline{1V_{i_1}})\ W(U_{i_2}, V_{i_2}) \dots W(U_{i_k}, V_{i_k})T = 0$ si et seulement si $U_{i_1}U_{i_2}\dots U_{i_k} = V_{i_1}V_{i_2}\dots \overline{V_{i_k}}$: cf. [65].

Remplaçons le problème de la correspondance de Post (PCP) par le problème modifié de la correspondance de Post (MPCP[1]) [40]. Ce problème est le suivant :

- Instance : un ensemble fini de couples de mots $\{< U_i, V_i > | i = 1 \dots p\}$
- Question : existe-t'il une suite non vide d'indices $i_2, \dots, i_k$ dans $\{1, 2, \dots, p\}$ telle que $U_1U_{i_2}\dots U_{i_k} = V_1V_{i_1}\dots V_{i_k}$?

Supposons que le problème PCP soit indécidable avec p règles. Puisque PCP peut se résoudre par p appels au problème MPCP, le problème MPCP est aussi indécidable avec p règles. Or le problème MPCP à p règles se réduit à $Mort_{\mathbb{Q}}(3,p+2)$. En effet, puisque dans le problème MPCP il n'y a qu'une chaîne initiale, on peut remplacer les matrices S et les matrices $W_{U_j,\underline{1V_j}}$ de l'ensemble F du paragraphe précédent par l'unique matrice $SW_{U_1,\underline{1V_1}}$: par les résultats du paragraphe précédent, l'ensemble de matrices $F = \{T, \overline{SW}_{U_1,\underline{1V_1}}, W_{U_j,V_j}$ pour $j = 1, \dots, p\}$ est mortel si et seulement s'il existe des entiers $i_2, \dots, i_k$ dans $\{1, 2, \dots, p\}$ tels que $SW(U_1, \underline{1V_1})W(U_{i_2}, V_{i_2}) \dots W(U_{i_k}, V_{i_k})T = 0$. Or, cela réduit le problème MPCP au problème $Mort_{\mathbb{Q}}(3,p+2)$, puisque nous savons que cela se produit si

[1]La différence entre le problème de Post et le problème modifié de Post est que dans ce dernier on impose à la solution de commencer par le premier couple.

et seulement si $\{< U_i, V_i > | i = 1 \dots p\}$ est une instance positive de MPCP. Par conséquent, le problème $Mort_{\mathbb{Q}}(3, p+2)$ est indécidable. □

On ne connaît pas le nombre minimal p de règles rendant le problème PCP indécidable, mais on sait cependant que p est un entier entre 3 et 7 : voir [57].

Il nous suffit maintenant d'utiliser le résultat suivant prouvé dans [11] et dans [29] :

Lemme 4.1 *[11, 29] Soient $n \geq 2, m \geq 1$ deux entiers.*

Si le problème $Mort_{\mathbb{Q}}(d, m)$ est indécidable, alors le problème $Mort_{\mathbb{Q}}(dm, 2)$ est indécidable.

Pour obtenir :

Corollaire 4.1 *Les problèmes suivants sont indécidables :*
- $Mort_{\mathbb{Q}}(3, 9)$.
- $Mort_{\mathbb{Q}}(27, 2)$.

4.3.3 Cas de deux matrices 2×2

Nous revenons maintenant au cas des matrices 2×2 en rapport avec les problèmes 4.5 et 4.6. Nous prouvons dans un premier temps que le problème $Mort_{\mathbb{R}}(2, 2)$ est BSS-indécidable. Puis nous prouverons que $Mort_{\mathbb{Q}}(2, 2)$ est Turing-décidable.

Pour cela, nous utiliserons la remarque suivante à plusieurs reprises :

Lemme 4.2 *Une ensemble fini $F = \{A_1, \dots, A_m\}$ de matrices 2×2 est mortel si et seulement s'il existe un entier $k \geq 1$ et des entiers $i_1, \dots, i_k \in \{1, \dots, m\}$ tels que $A_{i_1} \dots A_{i_k} = 0$ et*

1. *$rang(A_{i_j}) = 2$ pour $1 < j < k$*
2. *$rang(A_{i_j}) < 2$ pour $j \in \{1, k\}$.*

Preuve: Seul le sens direct nécessite une preuve. Supposons que l'ensemble F soit mortel. Il existe un produit nul $A_{i_1} \dots A_{i_k} = 0$ tel que $k \geq 1$ soit minimal. Supposons $k \geq 2$ car sinon l'assertion est immédiate. Les matrices A_{i_1} et A_{i_k} de ce produit sont non-singulières car sinon en multipliant le produit par leur(s) inverse(s) on obtiendrait un produit nul de longueur inférieure.

Soit j le plus petit entier supérieur à 1 tel que $rang(A_{i_j}) < 2$. Puisque $A_{i_1} \dots A_{i_k} = 0$, le produit $A_{i_1} \dots A_{i_j - 1}$ envoie l'image I de $A_{i_j} \dots A_{i_k}$ vers 0. Or I est aussi l'image de A_{i_j} et est de dimension 1 : en effet, premièrement I est clairement inclus dans l'image de A_{i_j}, deuxièmement, par définition de k, I ne saurait être de dimension 0, et troisièmement puisque $rang(A_{i_j}) < 2$ la dimension de l'image de A_{i_j} est au plus 1. On en déduit que l'on doit avoir $A_{i_1} \dots A_{i_j - 1} A_{i_j} = 0$ ce qui implique $j = k$ d'une part, et l'assertion d'autre part. □

BSS-indécidabilité du problème de la mortalité

Nous prouvons maintenant que $Mort_{\mathbb{R}}(2,2)$ est indécidable dans le modèle de Blum Shub et Smale.

Nous commençons par le résultat préliminaire suivant :

Lemme 4.3 *Soient $a, b, \lambda \in \mathbb{R}$ des réels avec $a^2 + b^2 \neq 0, \lambda \neq 0$. Soit θ un argument du nombre complexe $a + ib$. La paire de matrices $F(a, b, \lambda) = \{A_1, A_2\}$ avec*

$$A_1 = \begin{pmatrix} a & -b \\ b & a \end{pmatrix} \qquad A_2 = \begin{pmatrix} -\lambda & 1 \\ 0 & 0 \end{pmatrix}$$

est mortelle si et seulement s'il existe un entier $n \in \mathbb{N}$ tel que $\lambda = tan(n\theta)$.

Preuve: Par le lemme 4.2, $F(a, b, \lambda)$ est mortel si et seulement s'il existe un entier $n \in \mathbb{N}$ tel que $A_2 A_1^n A_2 = 0$. Cela est vrai si et seulement s'il existe une puissance $n$ème de A_1 qui envoie l'image de A_2 vers son noyau. Puisque $Im(A_2) =< (1, 0) >$, $Ker(A_2) =< (1, \lambda) >$, et puisque A_1 est la composée d'une homothétie et d'une rotation d'angle θ, cela se produit si et seulement s'il existe un entier $n \in \mathbb{N}$ tel que $\lambda = tan(n\theta)$. □

La preuve des assertions suivantes est élémentaire :

Lemme 4.4 *Soit θ un réel. Soit $E(\theta)$ le sous-ensemble de $\mathbb{R}$ défini par*

$$E(\theta) = \{\lambda |\ \textit{il existe un entier } n \in \mathbb{N} \textit{ avec } \lambda = tan(n\theta)\}$$

1. *$E(\theta)$ est dense dans $\mathbb{R}$ si et seulement si $\theta/\pi \notin \mathbb{Q}$.*
2. *Il existe deux nombres rationnels $a, b \in \mathbb{Q}$ tels que l'argument θ du nombre complexe $a + ib$ vérifie $\theta/\pi \notin \mathbb{Q}$: par exemple $a = 1$ et $b = 2$ (voir le lemme 4.5 plus loin).*
3. *Lorsque $\theta/\pi \notin \mathbb{Q}$, le complément $E^c(\theta)$ de $E(\theta)$ dans $\mathbb{R}$ possède un nombre non dénombrable de composantes connexes : en effet, tout point de $E^c(\theta)$ est sa propre composante connexe.*

Nous pouvons alors énoncer :

Théorème 4.5 *Le problème $Mort_{\mathbb{R}}(2,2)$ n'est pas BSS-décidable : le problème $Mort_{\mathbb{R}}(2,2)$ est BSS–récursivement énumérable mais n'est pas BSS-récursif.*

Preuve: Le problème est clairement BSS–récursivement énumérable.

Si les matrices sont représentées par leurs coefficients, l'espace des instances du problème $Mort_{\mathbb{R}}(2,2)$ peut être considéré comme l'espace $\mathbb{R}^8$. Notons $Pos \subset \mathbb{R}^8$ (respectivement $Neg \subset \mathbb{R}^8$) le sous-ensemble des instances positives. En utilisant

la proposition 1.10 du chapitre 1, il nous suffit de montrer que Neg n'est pas une union dénombrable d'ensembles semi-algébriques.

Soit $a, b \in \mathbb{Q}$ avec $a + ib = \rho e^{i\theta}$, $\theta/\pi \notin \mathbb{Q}$ comme dans le lemme 4.4. Soit $\gamma : \mathbb{R} \to \mathbb{R}^8$ la fonction qui envoie $\lambda \in \mathbb{R}$ vers la paire de matrices $F(a, b, \lambda)$. Par définition de γ, l'image Im_γ de γ est un sous-ensemble algébrique de $\mathbb{R}^8$ et γ réalise un homéomorphisme entre $\mathbb{R}$ et Im_γ. Par le lemme 4.3, on a $\gamma^{-1}(Pos) = E(\theta)$ et $\gamma^{-1}(Neg) = E^c(\theta)$. Puisque la fonction γ réalise un homéomorphisme, $E^c(\theta)$ et $\gamma(E^c(\theta)) = Neg \cap Im_\gamma$ doivent posséder le même nombre de composantes connexes. En l'occurrence, par la remarque 3 du lemme 4.4, ils doivent posséder un nombre non dénombrable de composantes connexes.

Or supposons par l'absurde que l'on puisse écrire $Neg = \cup_{i\in\mathbb{N}} S_i$ avec chacun des ensembles S_i semi-algébrique. On a alors $Neg \cap Im_\gamma = \cup_{i\in\mathbb{N}}(Im_\gamma \cap S_i)$. Chacun des ensembles $(Im_\gamma \cap S_i)$ est un ensemble semi-algébrique puisqu'il est égal à une intersection entre un ensemble algébrique et un ensemble semi-algébrique. Puisqu'un ensemble semi-algébrique possède un nombre fini de composantes connexes, $Neg \cap Im_\gamma$ doit posséder un nombre dénombrable de composantes connexes. Ceci est en contradiction avec le paragraphe précédent. □

On obtient immédiatement :

Corollaire 4.2 *Le problème $Mort_{\mathbb{R}}(n, m)$ pour $n \geq 2, m \geq 2$ est BSS-récursivement énumérable non BSS-récursif.*

Notons toutefois qu'il est facile de déduire des preuves de la section suivante que :

Théorème 4.6 *Le problème $Mort_{\mathbb{R}}(2, 2)$ restreint aux matrices à valeurs propres réelles est BSS-récursif.*

Interprétons le résultat du théorème 4.5 et du corollaire 4.2 : il n'est pas possible de décider si un ensemble de matrices est mortel en utilisant uniquement des opérations arithmétiques. Mais, cependant, cela ne signifie pas que ce problème ne puisse pas être décidé lorsque l'on autorise d'autres opérations que les opérations arithmétiques ou lorsqu'on autorise l'utilisation de considérations sur l'anneau K des coefficients.

Ainsi dans la sous-section qui suit, nous prouvons que le problème $Mort_{\mathbb{Q}}(2, 2)$ est Turing-décidable :

Turing-décidabilité du problème $Mort_{\mathbb{Q}}(2, 2)$

Nous prouvons maintenant que le problème $Mort_{\mathbb{Q}}(2, 2)$ est Turing-récursif.

Pour cela, nous avons besoin du résultat suivant extrait de la preuve de [77].

Lemme 4.5 ([77]) *Le problème de décision suivant est décidable :*

– *Instance :*

- *un rationnel* $p \in [-1, 1]$
- *un rationnel* $q \in [-1, 1]$.
- *Question : existe-t'il* $\theta \in \mathbb{R}$ *et un entier* $n \in \mathbb{N}$ *avec* $cos(\theta) = p$ *et* $cos(n\theta) = q$ *?*

En outre, pour $p \notin \{0, 1/2\}$*, il y a au plus un nombre fini de tels* n *et on peut les calculer effectivement.*

Preuve: Nous recopions ici la preuve de [77] dans le but de faire comprendre à notre lecteur où se cachent les arguments qui permettent de prouver la Turing-décidabilité du problème $Mort_{\mathbb{Q}}(2, 2)$, et qui ne permettent pas de prouver la BSS-décidabilité du problème $Mort_{\mathbb{R}}(2, 2)$.

Ecrivons $p = r/s$, $q = u/v$ où r, s, u, v sont des entiers vérifiant $pgcd(r, s) = pgcd(u, v) = 1$. La décidabilité du problème lorsque $p = 0$ ou lorsque $p = 1/2$ est triviale. Supposons maintenant $p \neq 0$ et $p \neq 1/2$. $cos(n\theta)$ est un polynôme en $cos(\theta)$ à coefficients entiers. S'il s'écrit $cos(n\theta) = p_n(r/s)$, alors $s^n p_n(r/s)$ est un entier c_n qui vérifie l'équation de récurrence :

$$2rc_{n+1} - s^2 c_n = c_{n+2} \tag{4.1}$$

avec $c_1 = r$ et $c_2 = 2r^2 - s^2$ (cette $a_n = sin(nx)$ et $b_n = cos(nx)$, cette récurrence se déduit facilement de $a_{n+1} = a_1 b_n + b_1 a_n$, $b_{n+1} = b_1 b_n - a_1 a_n$).

Supposons que s ne soit pas une puissance de 2. Ecrivons $s = 2^a b$, $v = 2^{a'} b'$ avec $b > 1$ et $b' \geq 1$ impairs. Nous cherchons un entier n tel que $c_n/(2^{an} b^n) = u/(2^{a'} b')$. Nous avons $pgcd(c_n, b^n) = 1$ pour tout $n \in \mathbb{N}$: en effet si un entier impair d divise simultanément s et c_n, alors, puisque $pgcd(r, s) = 1$, l'assertion $d|c_{n-1}, d|c_{n-2}, \ldots, d|c_2$ implique $d|r^2$, et donc $d = 1$. Par conséquent un entier n candidat doit vérifier $b' = b^n$. Puisque $b \neq 1$, il y a au plus un entier n candidat et cet entier est calculable.

Supposons maintenant que s soit une puissance de 2. Ecrivons $s = 2^k$, $k > 1$ (rappelons que nous avons supposé $r/s \neq 1/2$). Ecrivons chacun des c_n sous la forme $c_n = 2^{\lambda_n} v_n$ où v_n est un entier impair. La récurrence 4.1 devient

$$2^{\lambda_{n+1}+1} r v_{n+1} - 2^{\lambda_n + 2k} v_n = 2^{\lambda_{n+2}} v_{n+2} \tag{4.2}$$

Nous montrons d'abord qu'il existe un entier n avec $\lambda_n + 1 < 2k + \lambda_{n-1}$: si ce n'était pas le cas, nous aurions toujours $\lambda_n + 1 \geq 2k + \lambda_{n-1}$ de telle sorte que $\lambda_n + 1 \geq 2(n-1)k + \lambda_1$. Puisque $|cos(n\theta)| < 1$, nous avons $kn \geq \lambda_n$ qui implique $kn \geq 2(n-1)k + \lambda_1 - 1$ pour tout $n \in \mathbb{N}$. Ceci est clairement impossible.

Soit n_0 le plus petit entier tel que $\lambda_{n_0+1} < 2k + \lambda_{n_0}$: n_0 peut se calculer effectivement en testant pour les entiers n croissants. Nous avons $\lambda_{n_0+2} = \lambda_{n_0+1} + 1$, $\lambda_{n_0+2} + 1 = \lambda_{n_0+1} + 2 < \lambda_{n_0+1} + 2k$. Nous en déduisons que pour tout entier $h \geq 0$, $\lambda_{n_0+2+h} = \lambda_{n_0+1+h} + 1$. Par conséquent, si l'on pose $n_1 = n_0 + 1$, nous avons $\lambda_{n_1+h} = \lambda_{n_1} + h$ pour tout entier positif h.

Retournons maintenant à la question de l'existence d'un $n \in \mathbb{N}$ avec $cos(n\theta) = u/v$. Puisque $cos(\theta)$ a un dénominateur puissance de 2, l'entier v doit être une puissance de 2. Supposons $v = 2^m$. Il peut se produire qu'il existe une solution pour $n \leq n_1$. Pour $n > n_1$, une solution $n = n_1 + h$ doit vérifier $cos((n_1 + h)\theta) = v_{n_1+h}2^{\lambda_{n_1}+h}/2^{k(n_1+h)} = u/2^m$ donc $k(n_1 + h) - \lambda_{n_1} - h = m$, ou encore $h = (m + \lambda_{n_1} - kn_1)/(k - 1)$. C'est-à-dire que le seul entier n candidat supérieur à n_1 est $n_1 + (m + \lambda_{n_1} - kn_1)/(k - 1)$. Il existe donc au total au plus $n_1 + 2$ entiers candidats et ces entiers sont calculables. □

Muni du résultat précédent, nous sommes prêts à prouver que le problème $Mort_{\mathbb{Q}}(2, 2)$ est décidable (on observera que ce résultat a déjà été énoncé dans la littérature, mais que la preuve est soit absente [29], soit incomplète [74]) :

Théorème 4.7 *Le problème $Mort_{\mathbb{Q}}(2, 2)$ est décidable.*

Preuve: Soit $F = \{A_1, A_2\}$ deux matrices 2×2 à coefficients rationnels. Supposons, sans perte de généralité, que le rang de A_2 soit supérieur au rang de A_1. Si A_1 est de rang 2, alors les deux matrices sont non-singulières et F est immortel par le lemme 4.2. Si A_1 est de rang 0 alors F est mortel. Si les deux matrices ont pour rang 1, par le lemme 4.2, il suffit de tester si l'un des produits $A_1^2, A_1A_2, A_2A_1, A_2^2$ est nul.

Il reste donc uniquement le cas où A_2 est non-singulière et où A_1 est de rang 1. Dans ce cas, le lemme 4.2 nous dit que F est mortel si et seulement s'il existe un entier $n \in \mathbb{N}$ avec

$$A_1A_2^nA_1 = 0 \tag{4.3}$$

Nous voulons vérifier ceci algébriquement en utilisant la forme de Jordan de chacune des matrices A_1 et A_2. Ecrivons :

$$A_1 = P_1^{-1}J_1P_1, \qquad A_2 = P_2^{-1}J_2P_2$$

$$J_1 = \begin{pmatrix} \kappa & 0 \\ 0 & 0 \end{pmatrix},$$

et

$$J_2 = \begin{pmatrix} \lambda & 0 \\ 0 & \mu \end{pmatrix} \qquad \text{ou} \qquad \begin{pmatrix} \lambda & 1 \\ 0 & \lambda \end{pmatrix}.$$

avec P_1 et P_2 inversibles. La valeur propre κ, égale à la trace de A_1 est rationnelle. Les valeurs propres λ, μ de J_2 sont les racines, éventuellement complexes, du polynôme caractéristique de la matrice A_2. L'équation 4.3 se réécrit sous la forme

$$P_1^{-1}J_1P_1P_2^{-1}J_2^nP_2P_1^{-1}J_1P_1 = 0$$

ou, puisque la matrice P_1 est non-singulière, sous la forme

$$J_1 P J_2^n P^{-1} J_1 = 0$$

avec

$$P = P_1 P_2^{-1} = \begin{pmatrix} p & q \\ r & s \end{pmatrix}$$

Maintenant, après substitution de P, P^{-1} et de J, le problème se ramène, lorsque J_2 est de la première forme à l'existence d'un entier $n \in \mathbb{N}$ vérifiant $(ps)\lambda^n - (qr)\mu^n = 0$, et lorsque J_2 est de la seconde forme à l'existence d'un entier $n \in \mathbb{N}$ vérifiant $(ps - qr)\lambda^n - (rp)n = 0$.

Supposons que A_2 soit de la seconde forme. Dans ce cas, la valeur propre λ, égale à la trace de A_2 divisée par 2, est un rationnel. Les coefficients κ, p, q, r, s sont donc rationnels et facilement calculables. Tester s'il existe un entier n tel que $(ps - qr)\lambda^n - rpn = 0$ ne pose pas de problème particulier : on peut par exemple calculer numériquement une approximation à $1/2$ près des solutions réelles de l'équation $(ps - qr)\lambda^x - (rp)x = 0$ de façon à en déduire les entiers n candidats.

Supposons maintenant que A_2 soit de la première forme. Nous voulons tester l'existence d'un entier n tel que $ps\lambda^n - qr\mu^n = 0$. Observons que l'on doit avoir $\lambda \neq 0$ et $\mu \neq 0$ puisque A_2 est de rang 2. Les valeurs propres λ, μ et les coefficients p, q, r, s peuvent être complexes mais sont des éléments calculables de $\mathbb{Q}(\lambda)$: ils s'écrivent sous la forme $a + \lambda b$ pour des rationnels $a, b \in \mathbb{Q}$ calculables. En calculant dans $\mathbb{Q}(\lambda)$, les cas $ps = 0$ ou $qr = 0$ sont triviaux. Supposons maintenant $ps \neq 0$ et $qr \neq 0$. Le problème se ramène à l'existence d'un entier n tel que $(\lambda/\mu)^n = (pq)/(rs)$. On doit avoir $|\lambda/\mu|^n = |pq|/|rs|$. Lorsque $|\lambda/\mu| \neq 1$, n doit nécessairement valoir $|pq|/(|rs|log|\lambda/\mu|)$ et il suffit de calculer numériquement une approximation à $1/2$ près de cette quantité réelle pour obtenir au plus 2 entiers candidats, puis en calculant dans $\mathbb{Q}(\lambda)$ de vérifier si ces candidats sont bien solutions. Lorsque $|\lambda/\mu| = 1$ et λ et μ sont réels, on a nécessairement $\lambda = \mu$ ou $\lambda = -\mu$. Dans les deux cas, en calculant dans $\mathbb{Q}(\lambda)$, le problème est trivial. Lorsque $|\lambda/\mu| = 1$ et $|pq|/|rs| \neq 1$ le problème n'admet pas de solution.

Il reste donc uniquement le cas où λ et μ sont deux racines complexes conjuguées et $(pq)/(rs)$ est un complexe de module 1. Il est facile d'observer que puisque λ est racine du polynôme caractéristique de A_2 à coefficients rationnels, λ est un complexe de partie réelle rationnelle facile à calculer. Par conséquent, les complexes λ/μ et $(pq)/(rs)$, du type $a + \lambda b$ avec $a, b \in \mathbb{Q}$ calculables, ont aussi leurs parties réelles respectives p' et q' rationnelles et calculables. Si l'on note θ un argument du nombre complexe λ/μ de module 1, un entier n solution doit vérifier $cos(n\theta) = q'$. Lorsque $p' = 1/2$, on a $\lambda/\mu = 1/2 + i\sqrt{2}/2$ et $n \mapsto (\lambda/\mu)^n$ est une suite périodique de période 6 : il suffit de vérifier l'équation $(\lambda/\mu)^n = (pq)/(rs)$ pour $n = 0, 1, \ldots, 5$. Un raisonnement similaire permet d'écarter le cas $p' = 0$. Enfin, lorsque $p' \neq 1/2$, le lemme 4.5 nous dit qu'il y a au plus un nombre fini d'entiers n vérifiant $cos(n\theta) = q'$ et que ceux-ci sont calculables. Il suffit de vérifier s'ils sont effectivement solutions de l'équation $(\lambda/\mu)^n = (pq)/(rs)$. □

Nous venons de montrer que $Mort_{\mathbb{Q}}(2,2)$ était décidable. Nous ne savons pas ce qu'il en est de $Mort_{\mathbb{Q}}(2,3)$. Nos résultats sur la décidabilité des problèmes 4.5 et 4.6 s'arrêtent donc au théorème précédent.

Nous ne savons donc pas prouver que le problème 4.5 est décidable ni prouver qu'il est indécidable. Toutefois dans la section qui suit, nous allons prouver que ce problème peut se relier à d'autres problèmes ouverts de la littérature.

4.3.4 Formulations équivalentes

Dans cette section, nous prouvons que les problèmes 4.5 et 4.6 peuvent se relier à d'autres problèmes ouverts de la littérature. En effet, nous allons montrer qu'ils se relient au problème de l'égalité des coefficients étudié dans [48], au problème du zéro dans un coin étudié dans [54, 29] et enfin aux problèmes de la section 4.2. Lorsque C est une matrice, $C_{i,j}$ désignera le $j$ème coefficient de la $i$ème ligne de C.

Problème de l'égalité des coefficients

Nous commençons par prouver que le problème 4.6 se relie au problème de l'égalité des coefficients. Nous étendons par là les résultats de [48].

Il nous suffit de présenter la variation suivante du théorème 2 de [48] (par rapport au théorème 2 de [48] nous n'imposons pas à F de contenir uniquement des matrices non-singulières) :

Lemme 4.6 *Soit F un ensemble fini de matrices 2×2 à coefficients rationnels. Il existe un entier $k \geq 1$ et des entiers $i_1, \ldots, i_k$ tels que $A_{i_1} \ldots A_{i_k}$ soit une matrice C vérifiant $C_{2,1} = C_{2,2}$ si et seulement si l'ensemble fini F' constitué de F et de la matrice*

$$H = \begin{pmatrix} 0 & 1 \\ 0 & -1 \end{pmatrix}$$

est mortel.

Preuve: Il est facile de vérifier que la matrice H est telle que, pour toute matrice C, on ait $HCH = 0$ si et seulement si $C_{2,1} = C_{2,2}$. Cela prouve le sens direct.

Réciproquement, par le lemme 4.2, si F est mortel alors il existe des indices $i_1, \ldots, i_k$ tels que $A_{i_1} \ldots A_{i_k} = 0$ avec $A_{i_j} \neq H$ pour $1 < j < k$, et $rang(A_{i_j}) < 2$ pour $j \in \{1, k\}$. Si $A_{i_1} = A_{i_k} = H$, alors la remarque du paragraphe précédent implique que $A_{i_2} \ldots A_{i_{k-1}}$ est une matrice C vérifiant $C_{2,1} = C_{2,2}$. Si $A_{i_1} \neq H$ et $A_{i_k} \neq H$ alors $A_{i_1} \ldots A_{i_k}$ est un produit de matrices de F égal à la matrice nulle et la matrice nulle O vérifie bien $O_{2,1} = O_{2,2}$. Enfin les cas restants découlent de l'observation que, pour toute matrice C, l'équation $HC = 0$ (respectivement $CH = 0$) implique $C_{1,1} = C_{1,2}$. □

Nous pouvons alors relier les problèmes 4.5 et 4.6 au problème de l'égalité des coefficients (ce résultat constitue une extension aux matrices non-singulières de [48]) :

Théorème 4.8 (Egalité des coefficients) *Les problèmes 4.5 et 4.6 sont équivalents au problème de décision suivant :*

- *Instance : un ensemble fini* $F = \{A_1, \ldots, A_m\}$ *de matrices* 2×2 *à coefficients rationnels.*
- *Question : existe-t'il un entier* $k \geq 1$ *et des entiers* $i_1, \ldots, i_k$ *tels que* $A_{i_1} \ldots A_{i_k}$ *soit une matrice* C *vérifiant* $C_{2,1} = C_{2,2}$ *?*

et au problème de décision suivant :

- *Instance : un ensemble fini* $F = \{A_1, \ldots, A_m\}$ *de matrices* 2×2 *à coefficients rationnels non-singulières.*
- *Question : existe-t'il un entier* $k \geq 1$ *et des entiers* $i_1, \ldots, i_k$ *tels que* $A_{i_1} \ldots A_{i_k}$ *soit une matrice* C *vérifiant* $C_{2,1} = C_{2,2}$ *?*

Preuve: Le second problème se réduit facilement au premier puisqu'il en est un cas particulier. Réciproquement, le premier problème se réduit au problème 4.6 par le lemme 4.6 et [48] prouve que le problème 4.6 se réduit au second problème. □

Problème du zéro en haut à gauche

Nous étudions maintenant les liens qui existent entre les problèmes 4.5 et 4.6 avec le problème du zéro dans un coin présenté dans [29].

Il est connu que le problème de savoir si le semi-groupe généré par un ensemble fini de matrices carrées 3×3 inversibles contient un élément avec un zéro en haut à droite est indécidable [54, 29]. Mais la décidabilité du problème pour les matrices 2×2 est un problème ouvert : voir [29].

Nous prouvons que ce problème peut se relier au problème 4.6 par :

Théorème 4.9 (Zéro en haut à gauche) *Les problèmes 4.5 et 4.6 sont équivalents au problème de décision suivant :*

- *Instance : un ensemble fini* $F = \{A_1, \ldots, A_m\}$ *de matrices* 2×2 *à coefficients rationnels.*
- *Question : existe-t'il un entier* $k \geq 1$ *et des entiers* $i_1, \ldots, i_k$ *tels que* $A_{i_1} \ldots A_{i_k}$ *soit une matrice* C *vérifiant* $C_{1,1} = 0$ *?*

et au problème de décision suivant :

- *Instance : un ensemble fini* $F = \{A_1, \ldots, A_m\}$ *de matrices* 2×2 *à coefficients rationnels non-singulières.*
- *Question : existe-t'il un entier* $k \geq 1$ *et des entiers* $i_1, \ldots, i_k$ *tels que* $A_{i_1} \ldots A_{i_k}$ *soit une matrice* C *vérifiant* $C_{1,1} = 0$ *?*

Preuve: Il est facile de vérifier que, pour toute matrice C, si l'on pose

$$P = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$$

et $C' = PCP^{-1}$, on a $C'_{1,1} = 0$ si et seulement si $C_{2,1} = C_{2,2}$.

Par conséquent chacun des deux problèmes est équivalent au problème correspondant du théorème 4.8 par conjugaison des matrices de F par P ou par P^{-1}. □

Problèmes de l'atteignabilité pour les systèmes affines par morceaux de dimension 1

Nous prouvons maintenant que les problèmes 4.5 et 4.6 se relient au problème de l'atteignabilité pour les systèmes affines par morceaux de dimension 1 étudié dans la section 4.2.

Nous allons utiliser pour cela la restriction du problème 4.6 aux matrices triangulaires inférieures. En effet, il a été proposé dans [48] de restreindre les problèmes précédents aux matrices triangulaires inférieures. La motivation étant le fait que la preuve de la proposition 3.1 de [65] prouve aussi l'indécidabilité du problème de l'égalité des coefficients pour les matrices 3×3 triangulaires inférieures à coefficients rationnels.

Il s'avère que le problème $Mort_{\mathbb{Q}}(2)$ restreint aux matrices triangulaires inférieures est facilement décidable [48] : déterminer si un ensemble fini F de matrices triangulaires inférieures est mortel est équivalent à tester l'existence de deux matrices A, B de F telles que $A_{1,1} = 0$ et $B_{2,2} = 0$. Le problème du zéro en haut à gauche est lui aussi trivial.

Par contre, la question suivante est ouverte : voir [48].

Problème 4.7 (Matrices triangulaires inférieures [48]) *Le problème de décision suivant est-il décidable ?*

- *Instance : un ensemble fini $F = \{A_1, \ldots, A_m\}$ de matrices 2×2 à coefficients rationnels triangulaires inférieures non-singulières.*
- *Question : existe-t'il un entier $k \geq 1$ et des entiers $i_1, \ldots, i_k$ tels que $A_{i_1} \ldots A_{i_k}$ soit une matrice C vérifiant $C_{2,1} = C_{2,2}$?*

Ce problème nous permet de relier les problèmes 4.5 et 4.6 au problème de l'atteignabilité pour les systèmes affines par morceaux de la section 4.2 (voir en particulier le problème 4.3 de la section 4.2) :

Théorème 4.10 *Le problème 4.7 est équivalent au problème de la décidabilité du problème de décision suivant :*

- *Instance : un ensemble fini $F = \{f_1, \ldots, f_m\}$ de fonctions affines rationnelles non-constantes de dimension 1.*

– *Question : existe-t'il une composition* $f_{i_1} \circ f_{i_2} \circ \ldots \circ f_{i_k}$ *de ces fonctions qui envoie le point* 0 *sur le point* 1 *?*

Preuve: Appelons *problème de la composition* ce problème. Soit un ensemble fini $F = \{A_1, \ldots, A_m\}$ de matrices triangulaires inférieures non-singulières. Sans perte de généralité, nous pouvons nous ramener au cas où chacune des matrices $A \in F$ vérifie $A_{2,2} = 1$: en effet, puisque chacune des matrices $A \in F$ est supposée non-singulière, on a $A_{2,2} \neq 0$, et remplacer la matrice A par $A/A_{2,2}$ dans F ne change pas la mortalité de l'ensemble F.

Le problème 4.7 se réduit alors au problème de la composition pour $F' = \{f_1, \ldots, f_m\}$ où $f_i : x \mapsto A_{i_{1,1}} x + A_{i_{2,1}}$: il suffit de remarquer que dans tout produit $C = A_{i_1} \ldots A_{i_k}$ de matrices triangulaires inférieures avec $A_{i_{j_{2,2}}} = 1$ on a $C_{2,2} = 1$ et $C_{2,1} = f_{i_1} \circ f_{i_2} \ldots f_{i_k}(0)$.

Réciproquement le problème de la composition se réduit au problème 4.7 : lorsqu'un ensemble fini $F = \{f_1, \ldots, f_m\}$ de fonctions affines rationnelles non-constantes est donné, si chacune des fonctions f_i s'écrit $f_i : x \mapsto a_i x + b_i$, il suffit de considérer $F' = \{A_1, \ldots, A_m\}$ avec

$$A_i = \begin{pmatrix} a_i & 0 \\ b_i & 1 \end{pmatrix}$$

et de remarquer que, dans tout produit $C = A_{i_1} \ldots A_{i_k}$ de matrices de ce type, on a $C_{2,2} = 1$ et $C_{2,1} = f_{i_1} \circ f_{i_2} \ldots f_{i_k}(0)$. □

4.4 Problèmes de la stabilité

Nous passons maintenant à l'étude de la décidabilité des problèmes de la stabilité en basses dimensions. Dans la section 4.4.1, nous discutons les problèmes de la stabilité pour les systèmes affines par morceaux discontinus de basses dimensions. Dans la section 4.4.2, nous discutons les mêmes problèmes pour les systèmes affines par morceaux continus.

4.4.1 Cas discontinu

Intéressons-nous aux problèmes de la stabilité pour les systèmes affines par morceaux discontinus. Nous avons prouvé le résultat suivant dans le chapitre 3 :

Théorème 4.11 *Les problèmes de*
– *la convergence globale*
– *la mortalité*
– *la stabilité globale asymptotique*
pour les systèmes discontinus affines par morceaux de dimension 2 *sont indécidables.*

Cependant, nous ne savons pas répondre à la question suivante :

Problème 4.8 *Les problèmes de*
- *la convergence globale*
- *la mortalité*
- *la stabilité globale asymptotique*

pour les systèmes discontinus affines par morceaux de dimension 1 *sont-ils décidables ?*

4.4.2 Cas continu

Intéressons-nous maintenant aux problèmes de la stabilité pour les systèmes affines par morceaux continus. Nous avons établi le résultat suivant dans le chapitre 3 :

Théorème 4.12 *Les problèmes de*
- *la convergence globale*
- *la mortalité*
- *la stabilité globale asymptotique*

pour les systèmes continus affines par morceaux de dimension 4 *sont indécidables.*

Le résultat suivant a été prouvé par Koiran et Tsitsiklis dans notre soumission [10].

Théorème 4.13 ([10]) *Les problèmes de*
- *la convergence globale*
- *la mortalité*
- *la stabilité globale asymptotique*

pour les systèmes continus affines par morceaux de dimension 1 *sont décidables.*

Toutefois, nous ne savons pas répondre à la question suivante.

Problème 4.9 *Les problèmes de*
- *la convergence globale*
- *la mortalité*
- *la stabilité globale asymptotique*

pour les systèmes continus affines par morceaux de dimension $d = 2$ *ou* $d = 3$ *sont-ils décidables ?*

CHAPITRE

5 Représentation des polyèdres orthogonaux

5.1 Introduction

Ce chapitre étudie la représentation des polyèdres orthogonaux par leurs sommets : dans le reste de cette section, nous discutons la motivation de cette étude. Dans la section 5.2, nous introduisons les trois représentations que nous étudierons : la représentation par les couleurs, la représentation par les voisinages, et la représentation par les sommets extrêmes et nous prouvons la validité de ces représentations. Dans la section 5.3, nous proposons pour ces représentations des algorithmes pour détecter les faces d'un polyèdre, pour effectuer des comparaisons entre polyèdres et pour réaliser des opérations booléennes entre polyèdres. Enfin, dans la section 5.4, nous discutons les résultats obtenus.

Nous discutons maintenant les raisons qui nous ont amenés à nous intéresser à la représentation des polyèdres.

5.1.1 Vérification automatique de propriétés

Les *systèmes hybrides* ont été introduits pour modéliser les systèmes où un programme discret interagit avec un environnement continu : par exemple, les centrales nucléaires, les usines chimiques, ou les systèmes gérés par de l'informatique embarquée comme les avions ou les métros automatisés. Il est donc important de posséder des outils qui permettent de garantir que ces systèmes fonctionnent correctement.

Malheureusement, par les résultats d'indécidabilité du chapitre 3, il n'y a aucun espoir de concevoir un algorithme général qui prendrait en entrée un système dynamique quelconque et qui déciderait si ce système vérifie une propriété arbitraire.

Pour contourner ce problème, une première solution consiste à restreindre la classe des systèmes dynamiques et hybrides étudiés : par exemple, le problème de l'atteignabilité est décidable pour les automates temporisés [4, 7] ou pour certaines classes particulières de systèmes hybrides linéaires : voir [17, 36, 38, 43, 58].

Une deuxième solution consiste à accepter que les algorithmes puissent diver-

ger dans certains cas, c'est-à-dire à utiliser des *semi-algorithmes* au lieu d'algorithmes. Ces semi-algorithmes fonctionnent alors généralement en calculant l'ensemble des points atteints par les trajectoires des systèmes. On peut distinguer deux grands types de méthodes pour calculer cet ensemble : les *méthodes purement algorithmiques* et les *méthodes déductives*. Les premières fonctionnent sans intervention de l'utilisateur. Les secondes demandent à l'utilisateur de proposer un ensemble ou une famille d'ensembles qu'il pense solution, et se contentent de vérifier que cet ensemble est correct ou se contentent de déterminer un ensemble de la famille qui est solution.

Dans la catégorie des méthodes déductives, on peut ranger la méthode de l'ellipsoïde [51], les méthodes qui fonctionnent grâce à l'élimination des quantificateurs [42], ou celles qui fonctionnent en utilisant les méthodes de la théorie du contrôle comme les fonctions de Liapunov : voir [26, 53].

Dans la catégorie des méthodes purement algorithmiques, on peut ranger les méthodes pour les systèmes linéaires de [5, 6, 63], la méthode "Face-Lifting" de [33, 50, 35], ou les méthodes de l'analyse numérique : voir [76, 28]. On peut aussi ranger dans cette catégorie les programmes d'analyse qualitative automatisée des systèmes dynamiques : voir [71, 72, 73]

La plupart des méthodes purement algorithmiques requièrent la manipulation de polyèdres : les méthodes de [5, 6, 63] manipulent des polyèdres quelconques, [35] manipule des polyèdres produits cartésiens de polygones, [33] manipule des polyèdres orthogonaux, et [7] manipule des polyèdres unions de simplexes.

Il s'avère que la manipulation de polyèdres non nécessairement convexes est très coûteuse en temps. Si l'on veut manipuler des polyèdres convexes, il est important de posséder des représentations efficaces de ces polyèdres. C'est ce qui a motivé le travail présenté dans ce chapitre et dans le suivant.

Ce chapitre a pour objet l'étude de la représentation des polyèdres orthogonaux manipulés par [33]. Le chapitre 6 généralisera nos résultats aux polyèdres unions de simplexes manipulés par [7].

5.1.2 Représentations usuelles des polyèdres

La représentation des polyèdres non-convexes n'est pas un problème nouveau : voir [15, 34]. Classiquement les polyèdres sont représentés soit comme une union de polyèdres convexes (représentations par énumération), soit par une suite d'opérations booléennes entre polyèdres convexes (représentations CSG, représentations par octree), soit par une description de leur topologie (représentations par le graphe d'ajacence entre sommets, faces, arêtes) : voir [55] par exemple.

Toutes ces représentations ont l'avantage de permettre de déterminer très facilement l'appartenance d'un point donné à un polyèdre. Toutefois elles ne sont pas adaptées aux opérations requises par les algorithmes comme ceux de [7, 33]. En effet, dans ces algorithmes, on a besoin d'effectuer de nombreux tests de comparaisons et d'opérations booléennes entre polyèdres et de savoir détecter

les faces des polyèdres. Pour les représentations par énumération, comme pour les représentations CSG, tester si deux polyèdres sont égaux est un problème très difficile : tester si deux unions d'hyperrectangles sont égales est un problème co-NP-complet [34]. Il en est de même pour la détection des faces des polyèdres pour ces représentations. Pour les représentations par octree le test d'égalité entre polyèdres est trivial, mais le problème de la détection des faces reste difficile. Enfin les représentations topologiques ne sont pas non plus adaptées en raison de la difficulté de la gestion des opérations booléennes lorsque la dimension augmente.

Dans ce chapitre, nous proposons de représenter les polyèdres orthogonaux par leurs sommets. Nous proposons plusieurs représentations et nous prouvons que ces représentations conviennent parfaitement aux algorithmes de [33], dans le sens où, il existe des algorithmes relativement efficaces de détection des faces, de comparaison et de réalisation d'opérations booléennes entre polyèdres.

Au moment même de l'écriture de la version finale de ce document, nous avons découvert que nos résultats avaient de fortes similitudes avec ceux publiés très récemment par Aguilera dans sa thèse : voir [1, 2, 3]. Dans sa thèse, Aguilera prouve qu'un polyèdre orthogonal peut se représenter par ses sommets extrêmes. Toutefois le résultat n'est prouvé que pour les dimensions 1, 2 et 3. Dans ce chapitre, nous prouvons qu'un polyèdre orthogonal peut se représenter par ses sommets extrêmes mais en dimension quelconque. Notre point de vue, plus algébrique que le point de vue géométrique de [1], nous permet de donner une définition de sommet extrême qui coïncide avec celle de [1] en dimension $1, 2$ et 3, mais qui est valide en dimension quelconque. En outre nos preuves sont basées sur des arguments purement combinatoires. Nous renvoyons toutefois notre lecteur à [1] pour un développement beaucoup plus important que le nôtre de la complexité "pratique" des algorithmes, et pour de nombreux exemples d'applications des représentations des polyèdres par leurs sommets.

Le travail présenté dans ce chapitre est le fruit d'une collaboration avec Oded Maler et Amir Pnueli qui a mené lieu à la publication [24].

5.2 Représentation des polyèdres orthogonaux

Dans cette section, nous définissons les représentations que nous étudierons : dans la section 5.2.1, nous définissons les représentations que nous proposons. Dans les sections 5.2.2, 5.2.3, et 5.2.4, nous prouvons la validité de ces représentations.

5.2.1 Définitions

Nous introduisons quelques définitions : un *polyèdre orthogonal* est une union finie d'hyperrectangles bornés de pleines dimensions à coefficients rationnels. Autrement dit, à un changement de coordonnées près, un polyèdre orthogonal peut

se définir comme :

Définition 5.1 (Polyèdre orthogonal) *Une* boîte élémentaire *est un sous-ensemble de* $X = (\mathbb{R}^+)^d$ *de la forme* $B = [x_1, x_1+1] \times [x_2, x_2+1] \times \ldots [x_d, x_d+1]$ *avec* $x_i \in \mathbb{N}$ *pour tout* i. *Le point* $\mathbf{x} = (x_1, \ldots, x_d)$ *est appelé le* coin canonique de la boîte.

Un polyèdre orthogonal P *est une union de boîtes élémentaires.*

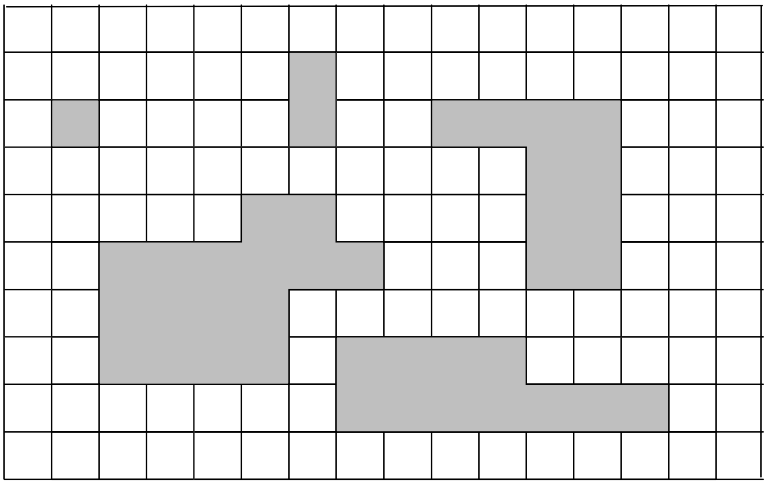


fig :exampleUn polyèdre orthogonal de dimension 2

Nous dénoterons par $\mathcal{G}(X)$ l'ensemble des polyèdres orthogonaux. Puisqu'il y a clairement correspondance entre les boîtes élémentaires et leurs coins canoniques, un polyèdre orthogonal sera aussi vu comme un sous-ensemble de $\mathbb{N}^d$. Nous appellerons *fonction couleur* la fonction $c : \mathbb{N}^d \to \{0, 1\}$ qui vaut 1 en $\mathbf{x}$ si et seulement si la boîte de coin canonique $\mathbf{x}$ est incluse dans P : un point $\mathbf{x}$ sera dit *noir* ou *blanc* suivant la valeur de $c(\mathbf{x})$.

Remarque. $\mathcal{G}$(X) n'est pas close par les opérations booléennes contrairement à $2^{\mathbb{N}^d}$. Si l'on veut que $\mathcal{G}$(X) soit une algèbre booléenne on doit redéfinir la complément de A comme $cl(\neg A)$ et l'intersection de A et de B comme $cl(int(A) \cap int(B))$, où $\neg$ est le complément standard et *cl* et *int* désignent l'adhérence et l'intérieur. C'est ce que nous ferons implicitement dans la suite. Avec ces définitions, on peut vérifier que l'algèbre sur $\mathcal{G}$(X) est isomorphe à l'algèbre sur $2^{\mathbb{N}^d}$.

Les définitions suivantes capturent la signification intuitive de face et de sommet : lorsque F est un sous-ensemble d'un sous-espace affine H, notons $rc_H(F)$ son adhérence dans la topologie induite sur H.

Définition 5.2 (Hyperplans, Faces, Sommets) – *Lorsque* $\mathbf{x} \in \mathbb{N}^d$ *est un point de l'espace et lorsque* $i \in \{1, \ldots, d\}$ *est une direction, le* i-prédécesseur *de* $\mathbf{x}$, *désigne le point* $\mathbf{x}^{i-} = (x_1, \ldots, x_{i-1}, x_i - 1, x_{i+1}, \ldots, x_d)$. *Le point* $\mathbf{x}$ *est dit* i-traversé *si* $c(\mathbf{x}) \neq c(\mathbf{x}^{i-})$.

– *Lorsque z est un entier quelconque et lorsque $i \in \{1, \ldots, d\}$ est une direction, le* i-hyperplan $\mathcal{H}_{i,z}$ *désigne l'hyperplan d'équation* $x_i = z$. *La* face $F_{i,z}$ *désigne le sous-ensemble de* $\mathcal{H}_{i,z}$ *défini par*

$$F_{i,z} = rc_{\mathcal{H}_{i,z}}\{\mathbf{x} = (x_1, \ldots, x_{i-1}, z, x_{i+1}, \ldots, x_d) | \mathbf{x} \text{ est } i\text{-traversé}\}$$

– *Une* arête *est une intersection non vide entre* $(d-1)$ *faces distinctes. Elle est dite une* i-arête *si elle est dans la direction* i, *c'est-à-dire si elle est du type* $\cap_{j \neq i} F_{j,z_j}$.
– *Un point* $\mathbf{x}$ *est un* sommet *s'il s'écrit comme l'intersection non vide de* d *faces distinctes.*

Nous aurons besoin de la notion de voisinage : voir la figure 5.1.

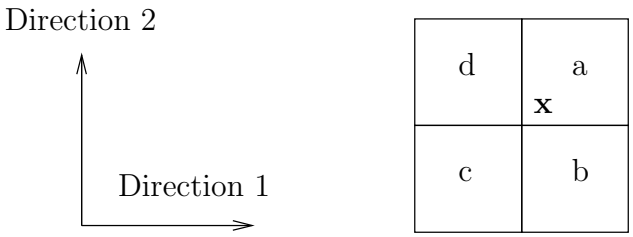


FIG. 5.1 – Les voisinages positifs et négatifs d'un point $\mathbf{x}$ en dimension 2 : $\mathcal{N}(\mathbf{x}) = \{a, b, c, d\}$, $\mathcal{N}^{1+}(\mathbf{x}) = \{a, b\}$, $\mathcal{N}^{1-}(\mathbf{x}) = \{c, d\}$, $\mathcal{N}^{2+}(\mathbf{x}) = \{a, d\}$, $\mathcal{N}^{1-}(\mathbf{x}) = \{b, c\}$

Définition 5.3 (Voisinage) *Le* voisinage du point $\mathbf{x} \in \mathbb{N}^d$, *dénoté par* $\mathcal{N}(\mathbf{x})$, *est l'ensemble* $\{x_1 - 1, x_1\} \times \ldots \times \{x_d - 1, x_d\}$ *: c'est-à-dire les* 2^d *sommets de la boîte d'extrémités* $\mathbf{x} - (1, \ldots, 1)$ *et* $\mathbf{x}$.

Le i-voisinage positif *de* $\mathbf{x} \in \mathbb{N}^d$, *dénoté par* $\mathcal{N}^{i+}(\mathbf{x})$, *est l'ensemble* $\{x_1 - 1, x_1\} \times \ldots \times \{x_{i-1} - 1, x_{i-1}\} \times \{x_i\} \times \{x_{i+1} - 1, x_{i+1}\} \times \ldots \times \{x_d - 1, x_d\}$ *: c'est-à-dire les* 2^{d-1} *sommets de* $\mathcal{N}(\mathbf{x})$ *du côté positif de* $\mathcal{H}_{i,x_i}$.

Le i-voisinage négatif *de* $\mathbf{x} \in \mathbb{N}^d$, *dénoté par* $\mathcal{N}^{i-}(\mathbf{x})$, *est l'ensemble* $\{x_1 - 1, x_1\} \times \ldots \times \{x_{i-1} - 1, x_{i-1}\} \times \{x_i - 1\} \times \{x_{i+1} - 1, x_{i+1}\} \times \ldots \times \{x_d - 1, x_d\}$ *: c'est-à-dire les* 2^{d-1} *sommets de* $\mathcal{N}(\mathbf{x})$ *du côté négatif de* $\mathcal{H}_{i,x_i}$.

Afin d'alléger les notations, lorsque $\mathcal{N}_1$ et $\mathcal{N}_2$ désigneront des sous-ensembles de $\mathbb{N}^d$ identiques à une translation de vecteur v près, nous écrirons $c(\mathcal{N}_1) = c(\mathcal{N}_2)$ pour signifier $c(\mathbf{x}) = c(\mathbf{x} + v)$ pour tout $\mathbf{x} \in \mathcal{N}_1$. Nous écrirons $c(\mathcal{N}_1) \neq c(\mathcal{N}_2)$ sinon.

Une *représentation de* $\mathcal{G}(X)$ est une fonction surjective d'un ensemble d'objets syntaxiques $\mathcal{T}$ vers $\mathcal{G}(X)$: c'est-à-dire une fonction qui à un objet syntaxique associe au plus un polyèdre et telle que chaque polyèdre ait au moins une écriture syntaxique. Une représentation est dite *canonique* lorsque cette fonction est injective : c'est-à-dire lorsqu'un polyèdre admet une unique représentation.

Nous nous intéresserons aux représentations suivantes :

1. La *représentation par les couleurs* représente un polyèdre orthogonal par l'ensemble
$$\{(\mathbf{x}, c(\mathbf{x}))|\ \mathbf{x} \text{ est un sommet du polyèdre}\}$$
Ainsi, dans cette représentation, un polyèdre est représenté par la liste de ses sommets avec pour chaque sommet les informations suivantes : ses coordonnées et la couleur de sa boîte associée.
2. La *représentation par les voisinages* consiste à représenter un polyèdre orthogonal par l'ensemble
$$\{(\mathbf{x}, c(\mathcal{N}(\mathbf{x})))|\ \mathbf{x} \text{ est un sommet du polyèdre}\}$$
Ainsi, dans cette représentation, un polyèdre est représenté par la liste de ses sommets avec pour chaque sommet : ses coordonnées et la couleur des boîtes dans son voisinage.
3. Un sommet $\mathbf{x}$ est dit *extrême* si le voisinage $\mathcal{N}(\mathbf{x})$ de $\mathbf{x}$ possède un nombre impair de boîtes de couleur noire : voir la figure 5.2. La *représentation par les sommets extrêmes* consiste à représenter un polyèdre orthogonal par l'ensemble
$$\{\mathbf{x}|\ \mathbf{x} \text{ est un sommet extrême du polyèdre}\}$$
Ainsi, dans cette représentation, un polyèdre est représenté par la liste de ses sommets extrêmes avec leurs coordonnées : cette représentation a été proposée par [1] pour les dimensions 1, 2 et 3.

On remarquera que la liste des sommets d'un polyèdre ne suffit pas à le représenter sans ambiguïtés : voir la figure 5.2.

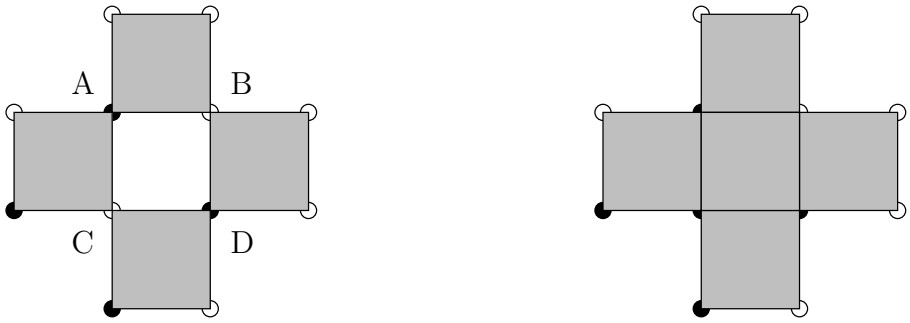


FIG. 5.2 – Deux polyèdres orthogonaux distincts mais avec les mêmes sommets. En chacun des sommets, on a représenté par un disque coloré la couleur de la boîte associée. Les sommets A, B, C et D sont les seuls sommets qui ne sont pas extrêmes

Nous prouvons maintenant que chacune des représentations précédentes est valide. Pour cela, il nous suffit de décrire pour chacune de ces représentations un

algorithme qui résout le *problème de l'appartenance*[1], c'est-à-dire un algorithme qui prend en entrée un polyèdre P et un point $\mathbf{x}$ et retourne $c(\mathbf{x})$.

5.2.2 Validité de la représentation par les couleurs.

On tire immédiatement des définitions de sommet et de face : voir la figure 5.3.

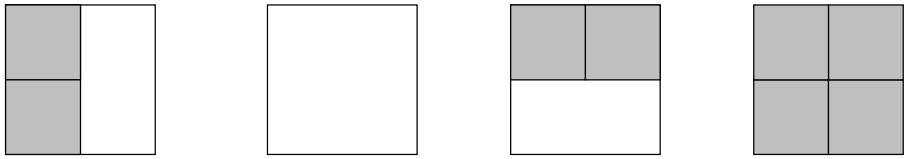


FIG. 5.3 – Illustration du lemme 5.1 : un point $\mathbf{x}$ n'est pas un sommet si et seulement s'il existe une direction $i \in \{1, \ldots, d\}$ telle que $c(\mathcal{N}^{i+}(\mathbf{x})) = c(\mathcal{N}^{i-}(\mathbf{x}))$. Illustration du lemme 5.2 : lorsqu'il existe une direction j avec $c(\mathcal{N}^{j+}(\mathbf{x}) - \{\mathbf{x}\}) = c(\mathcal{N}^{j-}(\mathbf{x}) - \{\mathbf{x}^{j-}\})$, alors $c(\mathbf{x}) = c(\mathbf{x}^{j-})$

Lemme 5.1 *1. Un point $\mathbf{x} \in \mathbb{N}^d$ est sur une i-face ssi $c(\mathcal{N}^{i+}(\mathbf{x})) \neq c(\mathcal{N}^{i-}(\mathbf{x}))$.*

2. Un point $\mathbf{x}$ n'est pas un sommet ssi $\exists i \in \{1, \ldots, d\}$ t.q. $c(\mathcal{N}^{i+}(\mathbf{x})) = c(\mathcal{N}^{i-}(\mathbf{x}))$.

On peut alors en déduire le résultat suivant (voir la figure 5.3) :

Lemme 5.2 (Couleur d'un non-sommet) *Soient $\mathbf{x}$ un point non-sommet et j une direction telle que $c(\mathcal{N}^{j+}(\mathbf{x}) - \{\mathbf{x}\}) = c(\mathcal{N}^{j-}(\mathbf{x}) - \{\mathbf{x}^{j-}\})$. Alors $c(\mathbf{x}) = c(\mathbf{x}^{j-})$.*

Preuve: Puisque $\mathbf{x}$ n'est pas un sommet, il existe une direction i telle que, pour tout $\mathbf{x}' \in \mathcal{N}^{i+}(\mathbf{x})$, $c(\mathbf{x}') = c(\mathbf{x}'^{i-})$. Si $j = i$ nous avons terminé et $c(\mathbf{x}) = c(\mathbf{x}^{j-})$. Sinon, nous avons en particulier $c((\mathbf{x}^{-i})^{-j}) = c(\mathbf{x}^{j-})$ et $c(\mathbf{x}^{i-}) = c(\mathbf{x})$. Par hypothèse nous avons $c((\mathbf{x}^{-i})^{-j}) = c(\mathbf{x}^{i-})$ d'où nous tirons $c(\mathbf{x}) = c(\mathbf{x}^{-j})$. □

Ceci nous permet de prouver la validité de la représentation par les couleurs :

Théorème 5.1 (Représentation par les couleurs) *La représentation par les couleurs*
est une représentation correcte et canonique des polyèdres orthogonaux.

[1]Le problème de l'appartenance est aussi appelé problème de la classification d'un point vis-à-vis d'un polyèdre [1].

Preuve: Puisque la canonicité est évidente, il suffit de prouver que la représentation est valide. Or l'algorithme récursif suivant détermine la couleur d'un point $\mathbf{x}$:

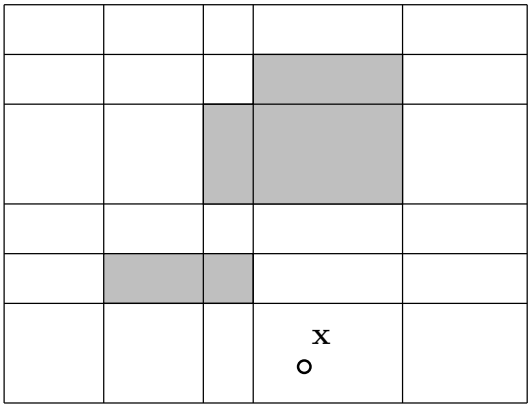
Algorithme $Couleur\_du\_point(\mathbf{x})$:

1. Si x est un sommet alors retourner sa couleur $c(\mathbf{x})$
en utilisant la représentation du polyèdre.
2. Si x possède une coordonnée négative retourner $c(\mathbf{x}) = blanc$.
3. Sinon
   3.1 Pour chacun des points $y \in \mathcal{N}(\mathbf{x}) - \{\mathbf{x}\}$
   Appeler $Couleur\_du\_point(y)$
   pour déterminer récursivement la couleur de y.
   3.2 Rechercher une direction j telle que
   $c(\mathcal{N}^{j+}(\mathbf{x}) - \{\mathbf{x}\}) = c(\mathcal{N}^{j-}(\mathbf{x}) - \{\mathbf{x}^{j-}\})$.
   3.3 Retourner $c(\mathbf{x}) = c(\mathbf{x}^{j-})$.

L'algorithme termine toujours car il définit la couleur d'un point $\mathbf{x}$ en fonction des couleurs de points inférieurs dans l'ordre partiel canonique sur $\mathbb{N}^d$. D'autre part, il est valide, puisque lorsque $\mathbf{x}$ n'est pas un sommet, par l'assertion 3 du lemme 5.1, il existe une direction j telle que $c(\mathcal{N}^{j+}(\mathbf{x}) - \{\mathbf{x}\}) = c(\mathcal{N}^{j-}(\mathbf{x}) - \{\mathbf{x}^{j-}\})$ et, par le lemme 5.2, pour une telle direction, on a nécessairement $c(\mathbf{x}) = c(\mathbf{x}^{j-})$. □

On peut améliorer l'algorithme ci-dessus si l'on veut réduire sa complexité : on peut modifier l'algorithme de telle sorte qu'il stocke la couleur des points dont il a déjà déterminé la couleur. Il fonctionne alors en temps majoré par $O(Nd2^d)$ et en espace N où N est le nombre de points de la grille $\mathbb{N}^d$ entre $(0, \ldots, 0)$ et $\mathbf{x}$. On peut remplacer la grille $\mathbb{N}^d$ par la *grille induite par le polyèdre* de telle sorte que N soit de l'ordre de n^d, où n est le nombre de sommets : si l'on appelle *i-échelle* l'ensemble des $i$èmes coordonnées des sommets du polyèdre, la grille induite par le polyèdre est définie comme le produit cartésien des i-échelles : voir la figure **??**. La grille induite se construit facilement en temps $O(dn^d)$. On obtient alors une complexité majorée par $O(n^d d2^d)$.

fig :ind-gridUn polyèdre et sa grille induite



5.2.3 Validité de la représentation par les voisinages.

Le théorème 5.1 prouve clairement que la représentation par les voisinages est une représentation correcte et canonique des polyèdres orthogonaux puisque cette dernière contient plus d'informations que la représentation par les couleurs.

Nous allons montrer que, lorsque la dimension d est fixée, la représentation par les voisinages est meilleure que la représentation par les couleurs puisqu'elle permet de résoudre le problème de l'appartenance en un temps de l'ordre de $O(nlog(n))$.

L'idée est d'effectuer $d-1$ projections orthogonales judicieuses afin de réduire le problème au cas trivial de la dimension 1.

Nous commençons par définir ce que nous appelons une section : voir la figure **??**.

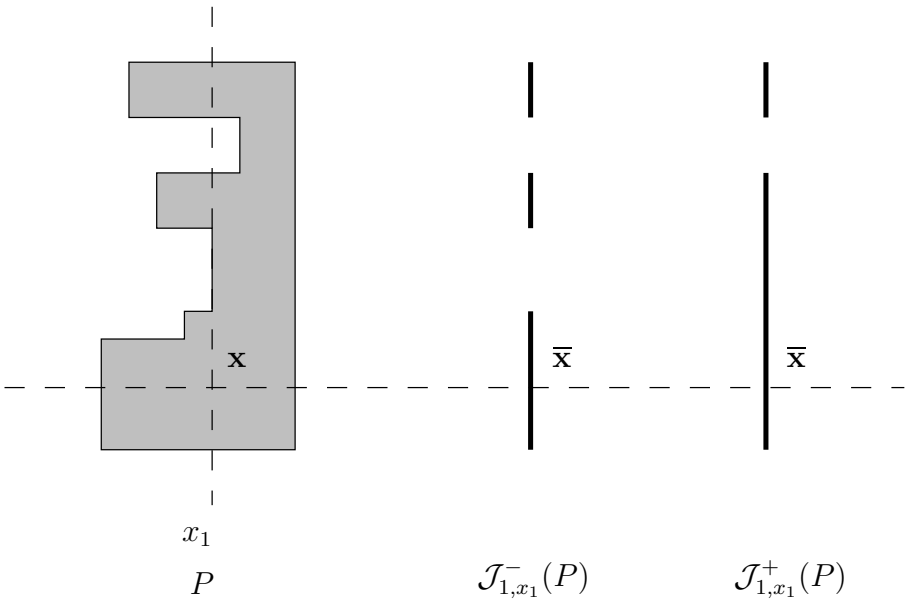


fig :sliceUn polyèdre et ses 1-sections positives et négatives

Définition 5.4 (i-section) *Soient un polyèdre orthogonal P, un entier z, et $i \in \{1, \ldots, d\}$ une direction. La* i-section positive de P en z *, dénotée par $\mathcal{J}^+_{i,z}(P)$, est le polyèdre orthogonal de dimension $d-1$ obtenu en intersectant P avec l'hyperplan $\mathcal{H}_{i,z+\epsilon}$, pour $\epsilon > 0$ très petit.*

Intéressons-nous au calcul des i-sections positives. Des définitions et résultats précédents on peut facilement tirer :

Observation 5.1 *Soient P un polyèdre et $\overline{P} = \mathcal{J}^+_{i,z}(P)$ une i-section positive de P.*

Les assertions suivantes sont équivalentes :

1. *Le point $\overline{\mathbf{x}} = (x_1, \ldots, x_{d-1})$ de $\overline{P}$ est un sommet.*
2. *Le i-voisinage positif du point $\mathbf{x} = (x_1, \ldots, x_{i-1}, z, x_i, \ldots, x_{d-1})$ de P vérifie $\forall j \neq i$, $c(\mathcal{N}^{i+}(\mathbf{x}) \cap \mathcal{N}^{j+}(\mathbf{x})) \neq c(\mathcal{N}^{i+}(\mathbf{x}) \cap \mathcal{N}^{j-}(\mathbf{x}))$.*
3. *Le point $\mathbf{x}^+ = (x_1, \ldots, x_{i-1}, z+\epsilon, x_i, \ldots, x_{d-1})$ de P, pour $\epsilon > 0$ très petit, appartient à une i-arête.*

Le point 2 de l'observation précédente nous invite à définir : voir la figure 5.4.

Définition 5.5 (Voisinage de sommet en dimension $d-1$) *Le i-voisinage positif d'un point $\mathbf{x}$ sera dit* voisinage de sommet en dimension $d-1$ *s'il vérifie la condition 2 de l'observation précédente : c'est-à-dire s'il vérifie $\forall j \neq i$, $c(\mathcal{N}^{i+}(\mathbf{x}) \cap \mathcal{N}^{j+}(\mathbf{x})) \neq c(\mathcal{N}^{i+}(\mathbf{x}) \cap \mathcal{N}^{j-}(\mathbf{x}))$*

Le i-voisinage négatif d'un point $\mathbf{x}$ sera dit voisinage de sommet en dimension $d-1$ *s'il vérifie $\forall j \neq i$, $c(\mathcal{N}^{i-}(\mathbf{x}) \cap \mathcal{N}^{j+}(\mathbf{x})) \neq c(\mathcal{N}^{i-}(\mathbf{x}) \cap \mathcal{N}^{j-}(\mathbf{x}))$*

Appelons *sommet i-prédécesseur d'un point $\mathbf{x}$* un sommet que l'on rencontre en partant de $\mathbf{x}$ et en se déplaçant dans la direction i décroissante : en d'autres termes, un sommet du type $(x_1, \ldots, x_{i-1}, z, x_{i+1}, \ldots, x_d)$ avec $z \leq x_i$. Appelons *premier sommet i-prédécesseur d'un point $\mathbf{x}$*, noté $\mathbf{x}^{\leftarrow i}$, le premier, quand il existe, de ces sommets que l'on rencontre.

Lemme 5.3 (Sommet d'une section) *Soient P un polyèdre et $\overline{P} = \mathcal{J}^+_{i,z}(P)$ une i-section positive de P.*

Un point $\overline{\mathbf{x}} = (x_1, \ldots, x_{d-1})$ est un sommet du polyèdre $\overline{P}$ si et seulement si le point $\mathbf{x} = (x_1, \ldots, x_{i-1}, z, x_i, \ldots, x_d)$ de P possède un sommet i-prédécesseur et le i-voisinage positif de $\mathbf{y} = \mathbf{x}^{\leftarrow i}$ est un voisinage de sommet en dimension $d-1$.

Lorsque tel est le cas, le voisinage du sommet $\overline{\mathbf{x}}$ dans $\overline{P}$ est égal au i-voisinage positif du sommet $\mathbf{y} = \mathbf{x}^{\leftarrow i}$ dans P.

Preuve: Par l'observation précédente $\overline{\mathbf{x}}$ est un sommet de $\overline{P}$ si et seulement si le i-voisinage positif de $\mathbf{x} = (x_1, \ldots, x_{i-1}, z, x_i, \ldots, x_{d-1})$ est un voisinage de sommet en dimension $d-1$.

Supposons que tel soit le cas. Il existe un point $\mathbf{y} = (x_1, \ldots, x_{i-1}, z, x_{i+1}, \ldots, x_d)$ tel que $c(\mathcal{N}^{i+}(\mathbf{y})) = c(\mathcal{N}^{i+}(\mathbf{x}))$ et $c(\mathcal{N}^{i-}\mathbf{y}) \neq c(\mathcal{N}^{i+}(\mathbf{y}))$ avec l'entier z maximal avec cette propriété. Les points $\mathbf{z}$ entre $\mathbf{y}$ et $\mathbf{x}$ doivent vérifier $c(\mathcal{N}^{i+}(\mathbf{z})) = c(\mathcal{N}^{i-}(\mathbf{z}))$ et ne peuvent donc pas être des sommets. Maintenant, puisque $c(\mathcal{N}^{i+}(\mathbf{y})) = c(\mathcal{N}^{i+}(\mathbf{x}))$, le i-voisinage positif de $\mathbf{y}$ est un voisinage de sommet en dimension $d-1$. De la définition même de cette propriété et du fait que l'on doive avoir $c(\mathcal{N}^{i-}\mathbf{y}) \neq c(\mathcal{N}^{i+}(\mathbf{y}))$ on en déduit que $\forall i \in \{1, \ldots, d\}$ $c(\mathcal{N}^{i+}(\mathbf{y})) \neq c(\mathcal{N}^{i-}(\mathbf{y}))$, c'est-à-dire que le point $\mathbf{y}$ est le sommet de P qui s'écrit $\mathbf{y} = \mathbf{x}^{\leftarrow i}$.

Réciproquement, supposons $\mathbf{y} = \mathbf{x}^{\leftarrow i}$ existe et que le i-voisinage positif de $\mathbf{y}$ soit un voisinage de sommet en dimension $d-1$. Supposons $\mathbf{x} \neq \mathbf{y}$ car sinon il n'y a rien à prouver. On doit avoir $c(\mathcal{N}^{i+}(\mathbf{y})) = c(\mathcal{N}^{i+}(\mathbf{x}))$ puisque sinon, par un raisonnement symétrique à celui utilisé dans le paragraphe précédent, $\mathbf{y}$ admettrait un sommet i-successeur entre $\mathbf{y}$ et $\mathbf{x}$. Donc le i-voisinage positif de $\mathbf{x}$ doit aussi être un voisinage de sommet en dimension $d-1$. □

On en déduit que l'on peut facilement calculer les sections d'un polyèdre donné : voir la figure 5.4.

Proposition 5.1 (Calcul des sections) *Etant donné un polyèdre orthogonal P, une direction i et un entier z, on peut calculer en temps $O(nd(\log n + 2^d))$ une représentation par les voisinages de $\overline{P} = \mathcal{J}^+_{i,z}(P)$ à partir d'une représentation par les voisinages de P.*

Preuve: Par le lemme précédent, il suffit d'utiliser l'algorithme suivant (l'étape 2, équivalente à un tri des sommets se réalise en $O(nd \log n)$, l'étape 3.1 se réalise en temps $O(d2^d)$, et donc l'algorithme s'exécute au total en moins de $O(nd(\log n + 2^d))$ étapes) :

Algorithme $Calcul\_Section(P, i, z)$:

1. Faire $\overline{P} = \emptyset$.
2. Déterminer les sommets de P qui sont sommets i-prédécesseurs de points de l'hyperplan $\mathcal{H}_{i,z}$.
3. Pour chacun de ces sommets $\mathbf{x} = (x_1, \ldots, x_d)$
   3.1 Si le voisinage i-positif du point est un voisinage de sommet en dimension $d-1$, alors ajouter le couple $(\overline{\mathbf{x}}, \mathcal{N}^{i+}(\mathbf{x}))$ à $\overline{P}$ avec $\overline{\mathbf{x}} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_d)$.
4. Retourner $\overline{P}$.

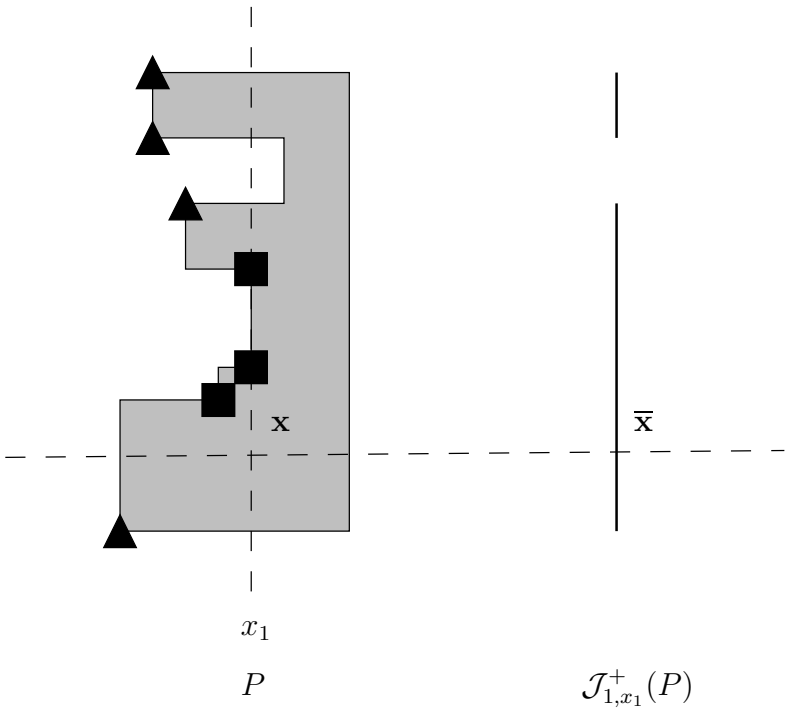


FIG. 5.4 – Illustration de la proposition 5.1 : pour calculer la 1-section positive de P en $z = x_1$, il suffit de déterminer les sommets de P qui sont 1-prédécesseurs de points de $\mathcal{H}_{1,x_1}$ (ceux marqués ici d'un triangle ou d'un carré), et de ne garder que ceux qui ont leurs i-voisinages positifs voisinage de sommet en dimension $d-1$ (ceux marqués ici d'un triangle)

□

Cela nous permet de prouver que pour la représentation par les voisinages, on peut déterminer la couleur d'un point $\mathbf{x}$ en temps $O(n \log n)$ lorsque la dimension d est fixée :

Théorème 5.2 *En utilisant la représentation par les voisinages, on peut déterminer la couleur d'un point* $\mathbf{x}$ *en temps* $O(nd^2(\log n + 2^d))$ *où* n *est le nombre de sommets du polyèdre.*

Preuve: Clairement déterminer la couleur du point $\mathbf{x} = (x_1, \ldots, x_d)$ d'un polyèdre P de dimension d est équivalent à déterminer la couleur du point $\overline{\mathbf{x}} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_d)$ du polyèdre $\overline{P} = \mathcal{J}^+_{i,x_i}(P)$ de dimension $d-1$. En répétant ce principe $d-1$ fois, le problème se réduit au cas trivial de la dimension 1. □

5.2.4 Validité de la représentation par les sommets extrêmes

Rappelons la définition d'un point extrême :

Définition 5.6 (Point extrême) *Un point* $\mathbf{x}$ *est dit* extrême *si* $\mathcal{N}(x)$ *possède un nombre impair de boîtes de couleur noire.*

Cette définition coïncide avec celle de [1] en dimension $1, 2, 3$: ceci peut se vérifier par exemple par l'étude exhaustive des configurations en un sommet présentée dans [1].

Nous avons la propriété remarquable suivante :

Lemme 5.4 *Un point extrême est nécessairement un sommet.*

Preuve: Nous prouvons cette assertion par récurrence sur la dimension. En dimension 1 l'assertion est facile à vérifier. Supposons $d > 1$. Fixons une direction i arbitraire. Puisque $\mathcal{N}(\mathbf{x})$ possède un nombre impair de boîtes noires, ceci doit être aussi le cas pour $\mathcal{N}^{i+}(\mathbf{x})$ ou pour $\mathcal{N}^{i-}(\mathbf{x})$. Donc, par hypothèse de récurrence, $\mathcal{N}^{i+}(\mathbf{x})$ ou $\mathcal{N}^{i-}(\mathbf{x})$ doit être un voisinage de sommet en dimension $d-1$. Dans tous les cas, on tire facilement des définitions que l'on doit avoir $c(\mathcal{N}^{j+}(\mathbf{x})) \neq c(\mathcal{N}^{j-}(\mathbf{x}))$ pour tout $j \neq i$. D'autre part, on doit avoir $c(\mathcal{N}^{i+}(\mathbf{x})) \neq c(\mathcal{N}^{i-}(\mathbf{x}))$, puisque sinon il y aurait un nombre pair de boîtes noires dans $\mathcal{N}(\mathbf{x})$. □

Lorsque $\mathbf{x} \in \mathbb{N}^d$ est un point et $i \in \{1, \ldots, d\}$ est une direction, nous noterons par $\pi^{i+}(\mathbf{x})$ la parité du nombre de boîtes élémentaires de couleur noire dans le i-voisinage positif de $\mathbf{x}$. Nous noterons par $\pi^{i-}(\mathbf{x})$ la parité du nombre de boîtes élémentaires de couleur noire dans le i-voisinage négatif de $\mathbf{x}$.

Nous prouvons :

Lemme 5.5 *Soient* $\mathbf{x} = (x_1, \ldots, x_{i-1}, z, x_{i+1}, \ldots, x_d)$ *un point et* $i \in \{1, \ldots, d\}$ *une direction. Soit* $\mathbf{y} = ((\mathbf{x})^{i-})^{\leftarrow i}$ *le premier sommet* i*-prédécesseur strict de* $\mathbf{x}$*.*

Alors $\pi^{i-}(\mathbf{x}) = \pi^{i+}(\mathbf{y})$*.*

Preuve: Le lemme 5.4 prouve que si l'on a $\pi^{i-}(\mathbf{x}) = 1$ alors le i-voisinage positif de $\mathbf{x}^{i-}$ est voisinage de sommet en dimension $d-1$. Par le lemme 5.3, on a alors $\mathcal{N}^{i-}(\mathbf{x}) = \mathcal{N}^{i+}(\mathbf{y})$.

Réciproquement, le lemme 5.4 implique que si l'on a $\pi^{i+}(\mathbf{y}) = 1$ alors le i-voisinage positif de $\mathbf{y}$ est un voisinage de sommet en dimension $d-1$. Par le lemme 5.3, on a alors $\mathcal{N}^{i-}(\mathbf{x}) = \mathcal{N}^{i+}(\mathbf{y})$. □

Remarque. Lorsque $\pi^{i-}(\mathbf{x}) = \pi^{i+}(\mathbf{y}) = 0$, on a pas nécessairement $\mathcal{N}^{i-}(\mathbf{x}) = \mathcal{N}^{i+}(\mathbf{y})$.

On déduit alors :

Lemme 5.6 *Soient* P *un polyèdre et* $\overline{P} = \mathcal{J}^+_{i,z}(P)$ *une* i*-section positive de* P*.*

Un point $\overline{\mathbf{x}} = (x_1, \ldots, x_{d-1})$ *est un sommet extrême de* $\overline{P}$ *si et seulement si le point* $\mathbf{x} = (x_1, \ldots, x_{i-1}, z, x_i, \ldots, x_d)$ *de* P *possède un nombre impair de sommets* i*-prédécesseurs extrêmes.*

Preuve: Observons tout d'abord que le point $\mathbf{x}$ est extrême si et seulement si $\pi^{i-}(\mathbf{x}) \neq \pi^{i+}(\mathbf{x})$. Nous prouvons le lemme par récurrence sur le nombre de i-sommets prédécesseurs de $\mathbf{x}$. Supposons que $\mathbf{x}$ ne possède pas de sommet i-prédécesseur. Dans ce cas, $\pi^{i-}(\mathbf{x}) = 0$ et $\pi^{i+}(\mathbf{x}) = 1$ si et seulement si le point $\mathbf{x}$ est extrême. Supposons le lemme vrai pour $n-1$ i-sommets prédécesseurs stricts. Supposons que le point $\mathbf{x}$ possède les sommets i-prédécesseurs stricts $\mathbf{y}^1, \ldots, \mathbf{y}^n$. Par hypothèse d'induction, $\pi^{i+}(\mathbf{y}^n)$ est égal à la parité du nombre de sommets extrêmes parmi les sommets $\mathbf{y}^1, \ldots, \mathbf{y}^n$. Par le lemme précédent, $\pi^{i+}(\mathbf{y}^n) = \pi^{i-}(\mathbf{x})$, et nous avons $\mathbf{x}$ non-extrême si $\pi^{i+}(\mathbf{x}) = \pi^{i-}(\mathbf{x}) = \pi^{i+}(\mathbf{y}^n)$, et $\mathbf{x}$ extrême si $\pi^{i+}(\mathbf{x}) \neq \pi^{i-}(\mathbf{x}) = \pi^{i+}(\mathbf{y}^n)$. Dans tous les cas, $\pi^{i+}(\mathbf{x})$ correspond à la parité du nombre des sommets i-prédécesseurs de $\mathbf{x}$. □

On obtient alors (voir la figure 5.5) :

Proposition 5.2 (Calcul des sections) *Etant donné un polyèdre orthogonal* P*, une direction* i *et un entier* z*, on peut calculer en temps* $O(nd \log n)$ *une représentation par les sommets extrêmes de* $\overline{P} = \mathcal{J}^+_{i,z}(P)$ *à partir d'une représentation par les sommets extrêmes de* P*.*

Preuve: Il suffit, par le lemme précédent, de déterminer les points de $\mathcal{H}_{i,z}$ qui possèdent un nombre impair de sommets extrêmes i-prédécesseurs. Cela peut se réaliser en temps $O(nd \log n)$ en triant convenablement les sommets puis en les parcourant. □

Et :

Théorème 5.3 *En utilisant la représentation par les sommets extrêmes, on peut déterminer la couleur d'un point* $\mathbf{x}$ *d'un polyèdre à* n*-sommets en temps* $O(nd^2 \log n)$*.*

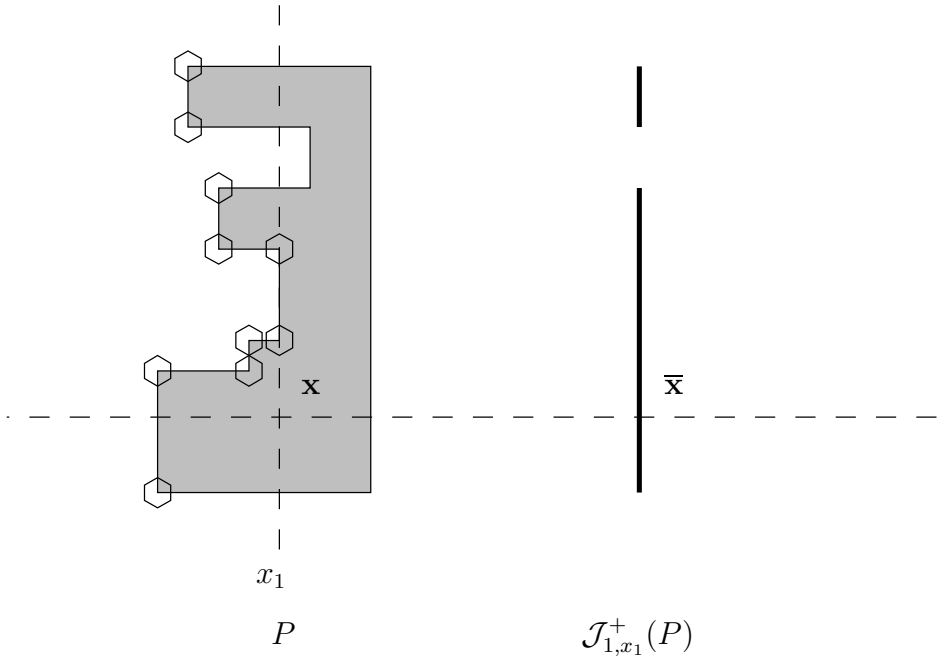


FIG. 5.5 – Illustration de la proposition 5.2 : pour calculer la 1-section positive de P en $z = x_1$, il suffit de déterminer les points de $\mathcal{H}_{1,x_1}$ qui possèdent un nombre impair de sommets extrêmes 1-prédécesseurs

Preuve: Il suffit d'utiliser le principe de la preuve du théorème 5.2 en utilisant l'algorithme précédent pour calculer les sections successives. □

5.3 Algorithmes sur les polyèdres orthogonaux

Nous présentons maintenant des algorithmes pour comparer deux polyèdres, pour détecter les faces d'un polyèdre, ou pour effectuer des opérations booléennes entre polyèdres.

Discutons toutefois la complexité d'un changement de représentation : on peut calculer la représentation par les sommets extrêmes d'un polyèdre à partir de sa représentation par les voisinages en testant si chacun des sommets est extrême : c'est-à-dire en temps $O(n2^d)$.

On peut calculer la représentation par les voisinages d'un polyèdre à partir de sa représentation par les couleurs en effectuant $n2^d$ calculs de couleurs : c'est-à-dire en temps $O(n^{d+1}d4^d)$. Par conséquent, on peut calculer la représentation par les sommets extrêmes d'un polyèdre à partir de sa représentation par les couleurs en temps $O(n2^d + n^{d+1}d4^d) = O(n^{d+1}d4^d)$.

Les autres conversions entre représentations ne nécessitent aucun calcul.

5.3.1 Tests d'égalité

Puisque les représentations étudiées sont canoniques, tester l'égalité entre deux polyèdres est un problème trivial :

Proposition 5.3 *Pour tester si deux polyèdres sont égaux, il suffit de comparer leurs représentations.*

5.3.2 Détection des faces

Le *problème de la détection des faces* est le suivant : étant donnés une représentation d'un polyèdre orthogonal P, une direction i et un entier z, calculer une représentation de la face $F_{i,z}$.

Soient $\mathcal{N}$ un sous-ensemble de $\mathbb{N}^d$ et i une direction. Nous noterons $c_i(\mathcal{N})$ le résultat de l'opération suivante sur les couleurs des boîtes de $\mathcal{N}$: la couleur de chaque boîte $\mathbf{y} \in \mathcal{N}$ est changée en la couleur blanche si $c(\mathbf{y}) = c(\mathbf{y}^{i-})$ et en la couleur noire si $c(\mathbf{y}) \neq c(\mathbf{y}^{i-})$.

Il est facile de déduire des définitions :

Observation 5.2 (Sommets des faces) *Soient P un polyèdre et $F_{i,z}$ une i-face de P.*

Alors $\overline{\mathbf{x}} = (x_1, \ldots, x_{d-1})$ est un sommet de $F_{i,z}$ si et seulement le voisinage de $\mathbf{x} = (x_1, \ldots, x_{i-1}, z, x_i, \ldots, x_d)$ vérifie $\forall j \neq i$ $c_i(\mathcal{N}^{i+}(\mathbf{x}) \cap \mathcal{N}^{j+}(\mathbf{x})) \neq c_i(\mathcal{N}^{i+}(\mathbf{x}) \cap \mathcal{N}^{j-}(\mathbf{x}))$.

Lorsque tel est le cas :

1. $\mathbf{x}$ *est un sommet de* P
2. *le voisinage de* $\overline{\mathbf{x}}$ *est donné par* $c_i(\mathcal{N}^{i+}(\mathbf{x}))$.

Théorème 5.4 (Détection des faces) *En utilisant la représentation par les voisinages, le problème de la détection des faces peut être résolu en temps* $O(nd2^d)$.

Preuve: Pour calculer la face $F_{i,z}$ il suffit de parcourir les sommets de P qui appartiennent à $\mathcal{H}_{i,z}$ et de tester si leurs voisinages vérifient la formule logique de l'observation 5.2. □

Remarquons qu'il est facile de modifier la condition testée en chaque sommet si l'on veut calculer les faces orientées.

Si l'on ne veut pas orienter les faces, le problème est encore plus facile avec la représentation par les sommets extrêmes :

Lemme 5.7 *Soient* P *un polyèdre et* $\overline{P} = F_{i,z}$ *une face de* P.

Alors $\overline{\mathbf{x}} = (x_1, \ldots, x_{d-1})$ *est un sommet extrême de* $F_{i,z}$ *si et seulement si le point* $\mathbf{x} = (x_1, \ldots, x_{i-1}, z, x_i, \ldots, x_d)$ *est un sommet extrême de* P.

Preuve: Le voisinage du point $\overline{\mathbf{x}}$ dans $\overline{P}$ est donné par $c_i(\mathcal{N}^{i+}(\mathbf{x}))$. Puisque l'opération qui définit $c_i(\mathcal{N}^{i+}(\mathbf{x}))$ à partir de $\mathcal{N}^{i+}(\mathbf{x})$ ne modifie pas la parité du nombre de boîtes noires, la parité du nombre de boîtes noires dans le voisinage du point $\overline{\mathbf{x}}$ dans $\overline{P}$ est égale à celle du nombre de boîtes noires dans $\mathcal{N}(\mathbf{x})$. □

Théorème 5.5 (Détection des faces) *En utilisant la représentation par les sommets extrêmes, le problème de la détection des faces peut être résolu en temps* $O(n)$.

Preuve: Pour calculer la face $F_{i,z}$ il suffit de retourner les sommets de la représentation de P qui appartiennent à $\mathcal{H}_{i,z}$. □

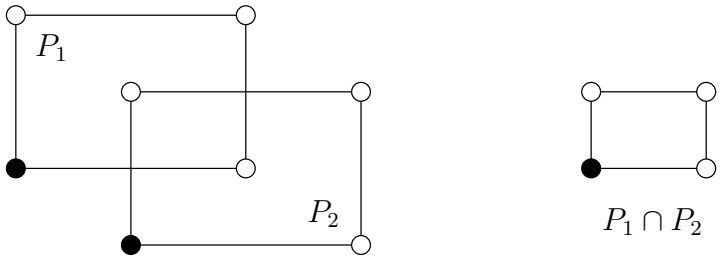
5.3.3 Opérations booléennes

Nous discutons maintenant la réalisation des opérations booléennes entre polyèdres.

Calculer le complément d'un polyèdre est facile pour les trois représentations. Nous décrivons maintenant comment effectuer une intersection booléenne entre deux polyèdres. Par les lois de Morgan, cela donnera le moyen de réaliser aussi une union entre polyèdres pour ces représentations.

Après intersection de deux polyèdres, certains sommets disparaissent et d'autres apparaissent : voir la figure **??**. Cependant tout point n'est pas candidat pour être un sommet de l'intersection :

fig :boolIntersection entre deux polyèdres



Lemme 5.8 *Un point* $\mathbf{x}$ *est un sommet de* $P_1 \cap P_2$ *seulement si, pour toute direction* i, $\mathbf{x}$ *est sur une* i*-face de* P_1 *ou sur une* i*-face de* P_2.

Preuve: Sinon il y aurait une direction i telle que $c(\mathcal{N}^{i+}(\mathbf{x})) = c(\mathcal{N}^{i-}(\mathbf{x}))$ à la fois dans P_1 et dans P_2, cela resterait le cas après intersection. □

Lorsque $\mathbf{y} = (y_1, \ldots, y_d)$ et $\mathbf{y}' = (y'_1, \ldots, y'_d)$ sont deux points de $\mathbb{R}^d$, nous écrivons $\mathbf{y} \leq \mathbf{y}'$ pour signifier $y_i \leq y'_i$ pour toute i. Nous dénotons par $max(\mathbf{y}, \mathbf{y}')$ le point $\mathbf{z} = (z_1, \ldots, z_d)$ défini par $z_i = max(y_i, y'_i)$ pour tout i.

Lemme 5.9 *Soit* $\mathbf{x} = (x_1, \ldots, x_d)$ *un point d'un polyèdre* P. *Soit* I *l'ensemble des directions* i *telles que* $\mathbf{x}$ *soit sur une* i*-face. Si* $I \neq \emptyset$, *alors il existe un sommet* $\mathbf{y} = (y_1, \ldots, y_d)$ *de* P *tel que* $\mathbf{y} \leq \mathbf{x}$ *et tel que* $y_i = x_i$ *pour tout* $i \in I$.

Preuve: Nous prouvons l'assertion par récurrence sur $k = d - |I|$ où $|I|$ est la cardinalité de I. Lorsque $k = 0$, l'assertion est triviale. Supposons $k \geq 1$. Soit i une direction n'appartenant pas à I. On doit avoir $\mathcal{N}^{i+}(\mathbf{x}) = \mathcal{N}^{i-}(\mathbf{x})$. Soit $\mathbf{y}' = (x_1, \ldots, x_{i-1}, z, x_i, \ldots, x_d)$ le point avec z maximal tel que $\mathcal{N}^{i+}(\mathbf{y}') = \mathcal{N}^{i-}(\mathbf{x})$ et $\mathcal{N}^{i+}(\mathbf{y}') \neq \mathcal{N}^{i-}(\mathbf{y}')$. De $\mathcal{N}^{i+}(\mathbf{x}) = \mathcal{N}^{i-}(\mathbf{x}) = \mathcal{N}^{i+}(\mathbf{y}')$, on déduit que, pour tout $j \in I$, $\mathbf{y}'$ appartient à une j-face. De $\mathcal{N}^{i+}(\mathbf{y}') \neq \mathcal{N}^{i-}(\mathbf{y}')$, on déduit que $\mathbf{y}'$ appartient à une i-face. Il suffit alors d'appliquer l'hypothèse de récurrence à $\mathbf{y}'$: il existe un sommet $\mathbf{y} \leq \mathbf{y}' \leq \mathbf{x}$ avec $y_j = y'_j = x_j$ pour tout $j \in I$. □

On obtient alors :

Lemme 5.10 *Soit* $\mathbf{x}$ *un sommet de* $P_1 \cap P_2$, *qui n'est ni un sommet de* P_1 *ni un sommet de* P_2. *Alors il existe un sommet* $\mathbf{y}^1$ *de* P_1 *et un sommet* $\mathbf{y}^2$ *de* P_2 *tels que* $\mathbf{x} = \max(\mathbf{y}^1, \mathbf{y}^2)$.

Preuve: Soit I_1 l'ensemble des directions i telles que $\mathbf{x}$ soit sur une i-face de P_1 et soit I_2 l'ensemble des directions i telles que $\mathbf{x}$ soit sur une i-face de P_2. Par le lemme 5.9 il existe un sommet $\mathbf{y}^1$ de P_1 et un sommet $\mathbf{y}^2$ de P_2 tels que, pour tout $j \in \{1, 2\}$, on ait $\mathbf{y}^j \leq \mathbf{x}$ et $y^j_i = x_i$ dès que $i \in I_j$. De l'égalité $I_1 \cup I_2 = \{1, 2, \ldots, d\}$ donnée par le lemme 5.8, on déduit que l'on doit avoir $\mathbf{x} = max(\mathbf{y}^1, \mathbf{y}^2)$. □

Nous obtenons (voir la figure 5.6) :

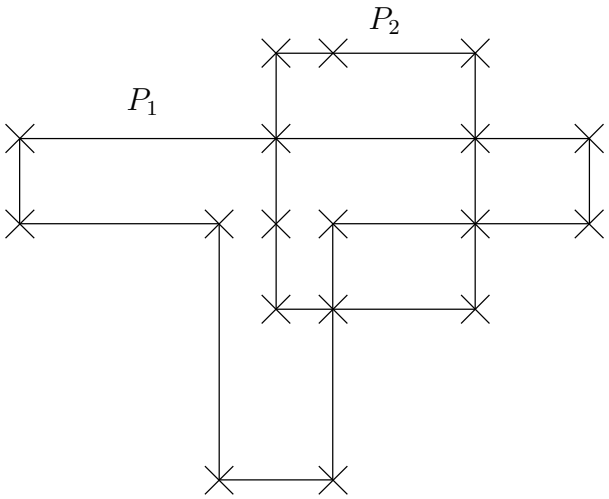


FIG. 5.6 – Illustration du théorème 5.6 : pour calculer l'intersection de deux polyèdres P_1 et P_2, il suffit de calculer le voisinage des points marqués d'une croix

Théorème 5.6 (Opérations booléennes) *L'intersection et l'union de deux polyèdres orthogonaux avec respectivement n_1 et n_2 sommets peuvent être calculées en utilisant la représentation par les sommets extrêmes en temps $O(n_1 n_2 d^2 2^d (n_1 + n_2))$.*

Preuve: Les candidats pour être des sommets de $P_1 \cap P_2$ sont les points du sous-ensemble suivant ($S(P_i)$ désigne l'ensemble des sommets de P_i) :

$$M(P_1, P_2) = S(P_1) \cup S(P_2) \cup \{\mathbf{x} : \exists \mathbf{y}^1 \in S(P_1) \exists \mathbf{y}^2 \in S(P_2) \mathbf{x} = \max(\mathbf{y}^1, \mathbf{y}^2)\}$$

Il suffit pour chaque point de cet ensemble de calculer son voisinage dans P_1 (complexité= 2^d calculs de couleurs), de calculer son voisinage dans P_2 (complexité= 2^d calculs de couleurs), d'intersecter les voisinages (complexité = 2^d) et de ne garder le point que si le voisinage obtenu compte un nombre impair de boîtes noires (complexité = 2^d).

Si l'on trie convenablement une fois pour toutes les sommets de P_1 et de P_2, le calcul des sommets qui sont les sommets i-prédécesseurs d'un point d'un hyperplan donné se fait en temps linéaire. Déterminer la couleur d'un point de P_i peut alors se réaliser en temps $O(n_i d^2)$. Il suffit d'observer que la cardinalité de $M(P_1, P_2)$ est majorée par $n_1 + n_2 + n_1 n_2$ pour obtenir la complexité annoncée. □

Remarque. On peut aussi calculer l'intersection ou l'union de deux polyèdres récursivement section par section. C'est la méthode utilisée par [1] en dimension 1, 2 et 3.

5.4 Discussion

Dans ce chapitre, nous avons donc proposé trois représentations des polyèdres orthogonaux par leurs sommets : la représentation par les couleurs, la représentation par les voisinages et la représentation par les sommets extrêmes.

Nous avons présenté des algorithmes pour résoudre le problème de l'appartenance, pour comparer les polyèdres, pour détecter les faces des polyèdres, et pour réaliser des opérations booléennes entre polyèdres. Nous avons en outre donné des bornes supérieures sur la complexité au pire cas de chacun de ces algorithmes.

Les algorithmes décrits dans ce document ont été implémentés dans une application "jouet" qui permet de manipuler des polyèdres orthogonaux aléatoires de dimension quelconque.

Les algorithmes sont en cours d'intégration dans le système décrit dans [33]. Mais l'avancement de ce travail ne nous permet pas de donner ici des mesures significatives de la performance des algorithmes sur des exemples réels de vérification.

CHAPITRE

6 Représentation des polyèdres temporisés

6.1 Introduction

Dans [7], Alur et Dill ont introduit les automates temporisés : un *automate temporisé* est un automate muni de variables qui croissent à vitesse 1 et qui peuvent être remises à zéro par des transitions discrètes. Les automates temporisés s'avèrent être de très puissants outils de modélisation très utilisés pratiquement : voir toutes les références de la synthèse [4]. Alur et Dill ont prouvé que le problème de l'atteignabilité est décidable pour ces systèmes : en effet, tout automate temporisé possède un nombre fini de classes de bissimulation et, par conséquent, l'étude des trajectoires d'un automate temporisé se ramène à l'étude du graphe fini de ses classes de bissimulation, appelé *graphe des régions* : voir [7].

Les classes de bissimulation des automates temporisés sont des polyèdres qui s'écrivent comme des unions finies de simplexes d'un type particulier. C'est pourquoi, les algorithmes sur les automates temporisés ont besoin de manipuler de tels polyèdres : ces algorithmes ont besoin de savoir comparer ces polyèdres, de savoir réaliser des opérations booléennes entre ces polyèdres, et de savoir réaliser une opération particulière appelée *passage du temps* : voir [4]. Puisque ces opérations sont très coûteuses pour les polyèdres non-convexes avec les représentations usuelles, les algorithmes se restreignent généralement aux polyèdres convexes et les représentent alors par des matrices de différences bornées : voir la synthèse [4].

Dans ce chapitre, nous proposons plusieurs représentations pour une sous-classe de ces polyèdres, que nous appelons les *polyèdres temporisés* : ces polyèdres correspondent aux classes de bissimulation des automates temporisés où l'on interdit les conditions de transition strictes. Nous prouvons la validité de ces représentations pour les polyèdres temporisés convexes et non-convexes de dimension quelconque. Nous proposons en outre des algorithmes qui permettent de réaliser des tests d'égalité, des opérations booléennes et l'opération "passage du temps" pour ces représentations.

Le plan de ce chapitre est le suivant : dans la section 6.2, nous introduisons les polyèdres temporisés et les deux représentations que nous étudierons : la

représentation par les simplexes de la boîte et la *représentation par les simplexes du voisinage*. Dans la section 6.3, nous introduisons quelques définitions. Dans les sections 6.4 et 6.5, nous prouvons la validité de ces représentations. Enfin, dans la section 6.6, nous décrivons comment l'on peut réaliser des tests d'égalité, des opérations booléennes, ou réaliser l'opération "passage du temps" en utilisant ces représentations.

6.2 Polyèdres temporisés

Nous allons maintenant définir précisément ce que nous appelons un polyèdre temporisé : lorsque x est un réel, nous écrirons $< x >$ pour la partie fractionnaire du réel x. Formellement : voir l'exemple de la figure **??**.

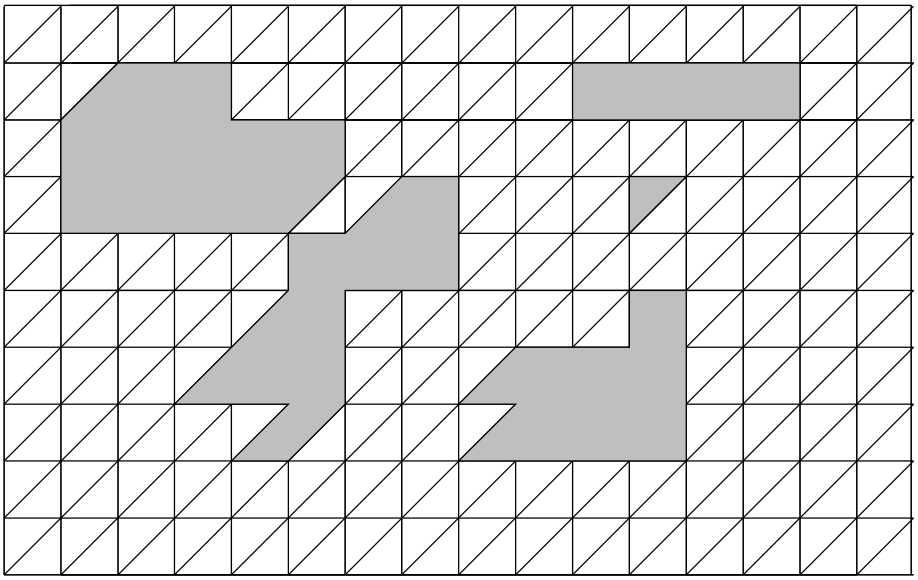


fig :exampletUn polyèdre temporisé de dimension 2

Définition 6.1 (Polyèdre temporisé) – *Une* boîte élémentaire *est un sous-ensemble de $X = (\mathbb{R}^+)^d$ de la forme $B = [x_1, x_1+1]\times[x_2, x_2+1]\times\ldots[x_d, x_d+1]$ avec $x_i \in \mathbb{N}$ pour tout i. Le point $\mathbf{x} = (x_1, \ldots, x_d)$ est appelé le* coin canonique de la boîte.

– *Un* simplexe élémentaire *est un sous-ensemble de $X = (\mathbb{R}^+)^d$ qui s'écrit comme l'intersection d'une boîte élémentaire et d'un sous-ensemble du type*

$$\{\mathbf{x}| < x_{\sigma(1)} > \leq < x_{\sigma(2)} > \leq \ldots \leq < x_{\sigma(d)} >\}$$

pour une permutation σ de $\{1, 2, \ldots, d\}$.

– *Un* polyèdre temporisé *P est une union finie de simplexes élémentaires.*

Nous dénoterons par $\mathcal{G}(X)$ l'ensemble des polyèdres temporisés. Un polyèdre temporisé sera aussi vu comme un sous-ensemble de $\overline{X} = \mathbb{N}^d \times \Pi$, où Π désigne l'ensemble des permutations de $\{1, \ldots, d\}$: $(\mathbf{x}, \sigma)$ dénotera le simplexe élémentaire inclus dans la boîte élémentaire de coin canonique $\mathbf{x}$ correspondant à la permutation σ. Comme dans le chapitre précédent, nous désignerons par $c : \overline{X} \to \{0, 1\}$ la fonction couleur, et nous supposerons redéfinies les opérations booléennes de telle sorte que l'algèbre sur $\mathcal{G}(X)$ soit isomorphe à l'algèbre sur $2^{\overline{X}}$.

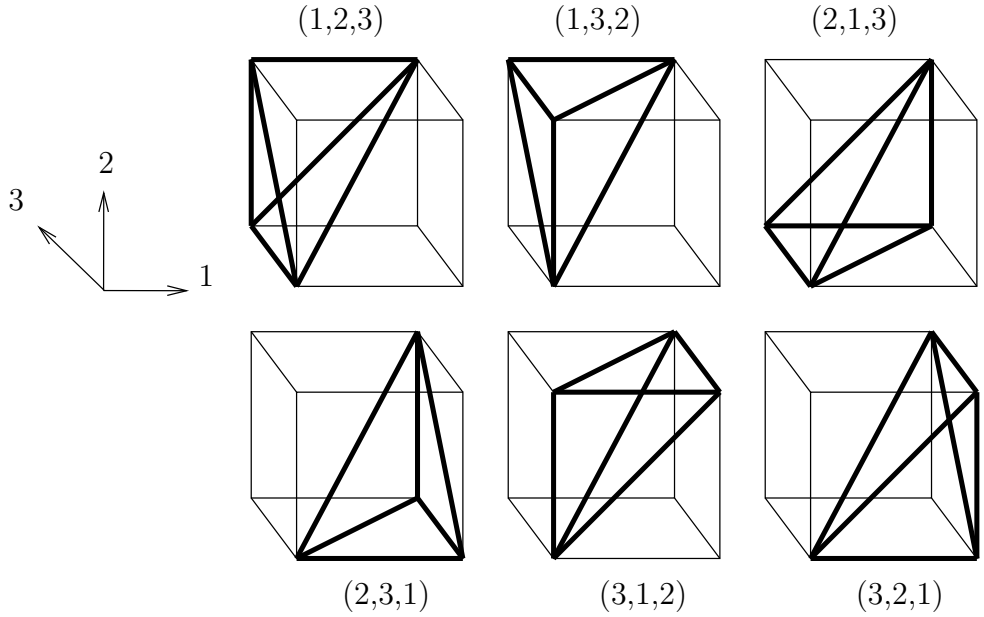


FIG. 6.1 – Les 6 simplexes d'une boîte élémentaire de dimension 3 avec leurs permutations associées

Nous présentons maintenant quelques remarques générales sur la géométrie des polyèdres temporisés : voir la figure 6.1.

Observation 6.1 *En dimension d :*

1. *Une boîte élémentaire contient $d!$ simplexes élémentaires.*
2. *Un point $\mathbf{x} \in \mathbb{N}^d$ appartient à $(d+1)!$ simplexes élémentaires : ces simplexes sont les simplexes $(\mathbf{y}, \sigma)$ tels que*

$$y_{\sigma(j)} = \begin{cases} x_{\sigma(j)} & \text{pour } j \leq k \\ x_{\sigma(j)} - 1 & \text{sinon} \end{cases}$$

   pour $k \in \{0, 1, \ldots, d\}$ et $\sigma \in \Pi$.
3. *Chaque simplexe élémentaire $(\mathbf{y}, \sigma)$*

(a) *est convexe*

(b) *possède* $d+1$ *sommets : ces sommets sont des sommets de la boîte élémentaire de coin canonique* $\mathbf{y}$

(c) *possède* $d(d+1)/2$ *arêtes*

(d) *possède* $d+1$ *faces : ces faces ont pour supports les hyperplans d'équations* $x_{\sigma(1)} = y_{\sigma(1)}$, $x_{\sigma(d)} = y_{\sigma(d)} + 1$, *et* $x_{\sigma(i)} - y_{\sigma(i)} = x_{\sigma(i+1)} - y_{\sigma(i+1)}$, $i \in \{1, 2 \ldots, d-1\}$.

On peut facilement se convaincre que les définitions suivantes capturent la signification intuitive de face et de sommet : comme dans le chapitre précédent, lorsque F est un sous-ensemble d'un sous-espace affine H, $rc_H(F)$ désigne l'adhérence de F dans la topologie induite sur H.

Définition 6.2 (Hyperplans et faces) *Soit* P *un polyèdre temporisé. Soit* z *un entier quelconque et soient* i *et* j *deux entiers distincts de* $\{1, \ldots, d\}$.

Hyperplans :

- $\mathcal{H}_{i,j,z}$ *désigne l'hyperplan d'équation* $x_i - x_j = z$.
- $\mathcal{H}_{i,z}$ *désigne l'hyperplan d'équation* $x_i = z$.

Faces :

- *La* face $F_{i,j,z}$ *désigne le sous-ensemble de* $\mathcal{H}_{i,j,z}$ *défini par* $F_{i,j,z} = rc_{\mathcal{H}_{i,j,z}}\{(\mathbf{x}, \sigma) \mid$ *le simplexe* $(\mathbf{x}, \sigma)$ *possède* $\mathcal{H}_{i,j,z}$ *comme face et est de couleur différente de son symétrique par cet hyperplan*$\}$.
- *La* face $F_{i,z}$ *désigne le sous-ensemble de* $\mathcal{H}_{i,z}$ *défini par* $F_{i,z} = rc_{\mathcal{H}_{i,z}}\{(\mathbf{x}, \sigma) \mid$ *le simplexe* $(\mathbf{x}, \sigma)$ *possède* $\mathcal{H}_{i,z}$ *comme face et est de couleur différente de son symétrique par cet hyperplan*$\}$.
- *Une* face *est soit une* i,j-face *(c'est-à-dire une face* $F_{i,j,z}$*), soit une* i-face *(c'est-à-dire une face* $F_{i,z}$*).*

Arêtes et sommets :

- *Une* arête *est une intersection entre faces de dimension affine* 1.
- *Un* sommet *est une intersection entre faces de dimension affine* 0.

Nous présentons maintenant les représentations que nous étudierons : il est naturel de définir les notions suivantes de voisinages : voir la figure 6.2.

Définition 6.3 (Voisinage-boîte et voisinage-simplexe) *Soit* $\mathbf{x} \in \mathbb{N}^d$ *un point.*

Le voisinage-boîte *du point* $\mathbf{x}$ *est constitué des* $d!$ *simplexes inclus dans la boîte élémentaire de coin canonique* $\mathbf{x}$.

Le voisinage-simplexe *du point* $\mathbf{x}$ *est constitué des* $(d+1)!$ *simplexes élémentaires qui contiennent* $\mathbf{x}$.

Nous nous intéresserons aux représentations suivantes :

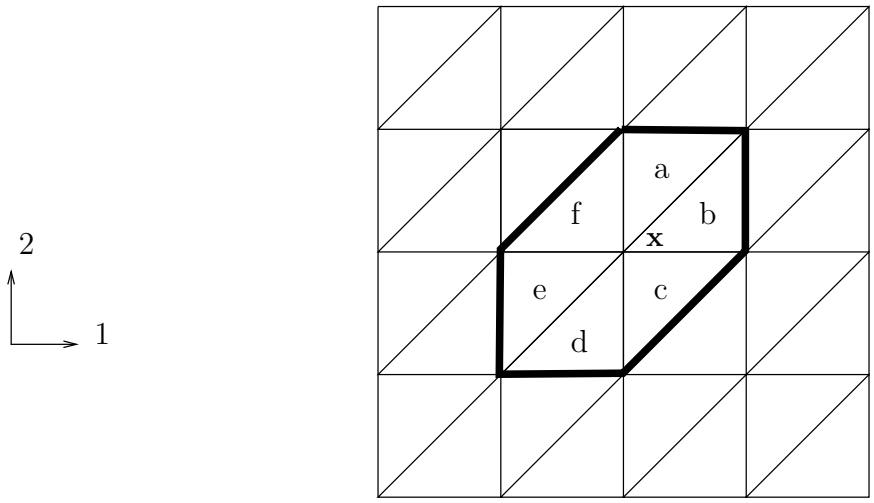


FIG. 6.2 – Le voisinage-boîte et le voisinage-simplexe d'un point $\mathbf{x}$ en dimension 2 : le voisinage-boîte est constitué des simplexes a et b ; le voisinage-simplexe est constitué des simplexes a, b, c, d, e, f

1. La *représentation par les simplexes de la boîte* (généralisation de la représentation par les couleurs du chapitre précédent) : un polyèdre temporisé est représenté par la liste des sommets du polyèdre avec pour chaque sommet $\mathbf{x}$ du polyèdre les informations suivantes : les coordonnées de $\mathbf{x}$ et la couleur de chacun des simplexes du voisinage-boîte de $\mathbf{x}$.

2. La *représentation par les simplexes du voisinage* (généralisation de la représentation par les voisinages du chapitre précédent) : un polyèdre temporisé est représenté par la liste des sommets du polyèdre avec pour chaque sommet $\mathbf{x}$ du polyèdre les informations suivantes : les coordonnées de $\mathbf{x}$ et la couleur de chacun des simplexes du voisinage-simplexe de $\mathbf{x}$.

Observons que nous ne possédons pas de généralisation satisfaisante de la représentation par les sommets extrêmes du chapitre précédent : en effet, les arguments de parité utilisés dans le chapitre précédent ne semblent pas se généraliser très facilement au cadre des polyèdres temporisés.

6.3 Préliminaires

Avant de prouver la validité des représentations précédentes, nous introduisons quelques définitions

6.3.1 Notation $(\mathbf{x}, \sigma, k)$

Afin de noter aisément les simplexes du voisinage-simplexe d'un point donné, nous représenterons de la façon suivante les simplexes du voisinage-simplexe d'un point $\mathbf{x}$: observez la figure 6.3 (et le point 2 de l'observation 6.1).

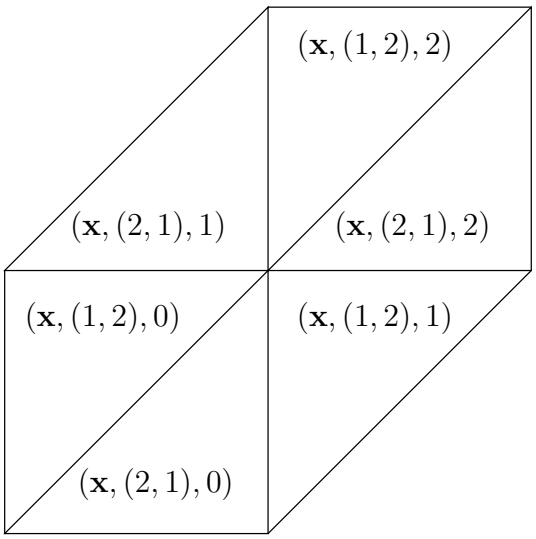


FIG. 6.3 – Notation des simplexes du voisinage-simplexe du point $\mathbf{x}$

Définition 6.4 (Notation $(\mathbf{x}, \sigma, k)$) *Lorsque $0 \leq k \leq d$ est un entier et σ est une permutation, $(\mathbf{x}, \sigma, k)$ dénotera le simplexe élémentaire $(\mathbf{y}, \sigma)$ du voisinage-simplexe de $\mathbf{x}$ défini par*

$$y_{\sigma(j)} = \begin{cases} x_{\sigma(j)} & \text{pour } j \leq k \\ x_{\sigma(j)} - 1 & \text{sinon} \end{cases}$$

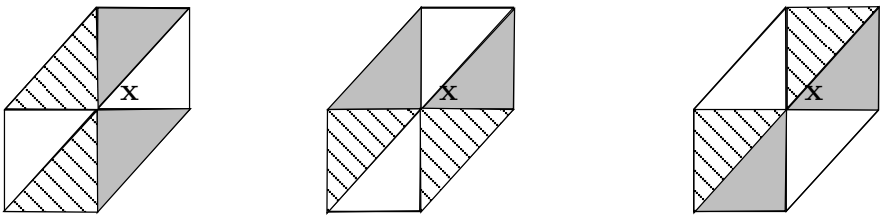
6.3.2 i-voisinages et i, j-voisinages

Nous utiliserons la terminologie suivante : voir la figure 6.4.

Définition 6.5 (i-voisinage et i, j-voisinage) *Soient $\mathbf{x} \in \mathbb{N}^d$ un point et $i \in \{1, \ldots, d\}$ une direction.*

- *Le* i-voisinage *de $\mathbf{x}$ est le sous-ensemble du voisinage-simplexe de $\mathbf{x}$ constitué des simplexes dont l'une des faces est supportée par l'hyperplan $\mathcal{H}_{i,x_i}$.*
- *Le* i-voisinage positif *(resp.* i-voisinage négatif*) de $\mathbf{x}$ est constitué des simplexes du i-voisinage de $\mathbf{x}$ qui sont inclus dans le demi-espace d'équation $y_i \geq x_i$ (resp. $y_i \leq x_i$).*

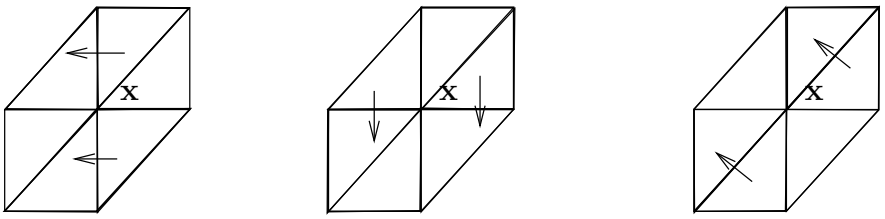
Soient $\mathbf{x} \in \mathbb{N}^d$ un point et $i, j \in \{1, \ldots, d\}$, $i < j$, deux directions.



1-voisinage 2-voisinage $1,2$-voisinage

FIG. 6.4 – Figure de gauche (resp. du milieu, de droite) : les simplexes du 1-voisinage positif (resp. 2-voisinage positif, $1,2$-voisinage positif) de $\mathbf{x}$ sont grisés ; les simplexes du 1-voisinage négatif (resp. 2-voisinage négatif, $1,2$-voisinage négatif) de $\mathbf{x}$ sont hachurés

- *Le* i,j-voisinage *de* $\mathbf{x}$ *est le sous-ensemble du voisinage-simplexe de* $\mathbf{x}$ *constitué des simplexes dont l'une des faces est supportée par l'hyperplan* $\mathcal{H}_{i,j,x_i-x_j}$.
- *Le* i,j-voisinage positif *(resp.* i,j-voisinage négatif*) de* $\mathbf{x}$ *est constitué des simplexes du* i,j*-voisinage de* $\mathbf{x}$ *qui sont inclus dans le demi-espace d'équation* $y_i - y_j \geq x_i - x_j$ *(resp.* $y_i - y_j \leq x_i - x_j$*).*



1-prédécesseur 2-prédécesseur $1,2$-prédécesseur

FIG. 6.5 – Figure de gauche (resp. du milieu, de droite) : le simplexe 1-prédécesseur (resp. 2-prédécesseur, $1,2$-prédécesseur) de chacun des simplexes du voisinage 1-positif (resp. 2-positif, $1,2$-positif) de $\mathbf{x}$ est indiqué par une flèche

On peut alors définir : voir la figure 6.5.

Définition 6.6 (Simplexe i et i,j-prédécesseur) *Soit* $\mathbf{x} \in \mathbb{N}^d$ *un point et soit une direction* $i \in \{1,\ldots,d\}$*. Soit* $(\mathbf{x},\sigma,k)$ *un simplexe du* i*-voisinage positif de* $\mathbf{x}$*.*

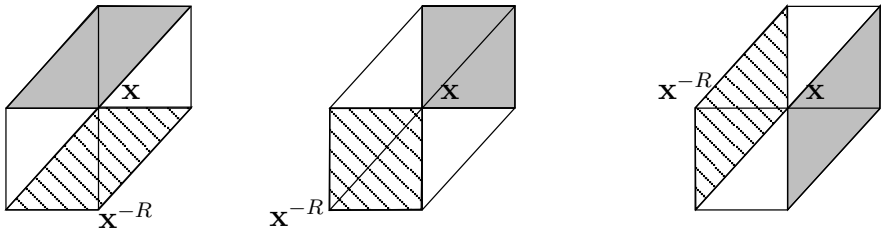
Le simplexe i-prédécesseur *de* $(\mathbf{x},\sigma,k)$*, noté* $(\mathbf{x},\sigma,k)^{-i}$*, est le symétrique du simplexe* $(\mathbf{x},\sigma,k)$ *par rapport à l'hyperplan* $\mathcal{H}_{i,x_i}$*.*

Soient $\mathbf{x} \in \mathbb{N}^d$ *un point et* $i,j \in \{1,\ldots,d\}$ *deux directions avec* $i < j$*. Soit* $(\mathbf{x},\sigma,k)$ *un simplexe du* i,j*-voisinage positif de* $\mathbf{x}$*.*

Le simplexe i,j-prédécesseur *de* $(\mathbf{x},\sigma,k)$*, noté* $(\mathbf{x},\sigma,k)^{-i,j}$*, est le symétrique du simplexe* $(\mathbf{x},\sigma,k)$ *par rapport à l'hyperplan* $\mathcal{H}_{i,j,x_i-x_j}$*.*

6.3.3 R-voisinages

Nous définissons maintenant la notion de R-voisinage, lorsque $R \in \{0,1\}^d$ est un vecteur non nul.



$R=(0,1)$ $R=(1,1)$ $R=(1,0)$

FIG. 6.6 – Les simplexes du R-voisinage positif (resp. négatif) de $\mathbf{x}$ sont grisés (resp. hachurés) pour les directions $R=(0,1)$, $R=(1,1)$ et $R=(1,0)$

Définition 6.7 (R-voisinage) *Soient* $\mathbf{x}=(x_1,\ldots,x_d)\in\mathbb{N}^d$ *un point et* $R\in\{0,1\}^d$ *un vecteur non nul.*

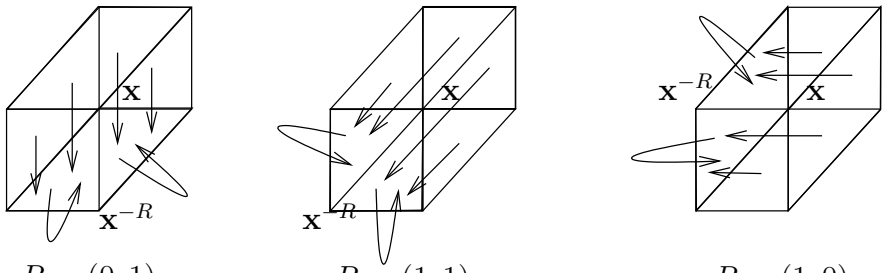
- *Le point* R-prédécesseur *du point* $\mathbf{x}$*, noté par* $\mathbf{x}^{-R}$*, est le point de coordonnées* $\mathbf{x}^{-R}=(x_1-R_1,\ldots,x_d-R_d)$*.*
- *Le* R-voisinage négatif *du point* $\mathbf{x}$*, noté* $\mathcal{N}^{R-}(\mathbf{x})$*, est constitué des simplexes qui sont à la fois dans le voisinage-simplexe de* $\mathbf{x}$ *et dans le voisinage-simplexe de* $\mathbf{x}^{-R}$*.*
- *Le* R-voisinage positif *du point* $\mathbf{x}$*, noté* $\mathcal{N}^{R+}(\mathbf{x})$*, est constitué des simplexes qui sont à la fois dans le voisinage-simplexe de* $\mathbf{x}$ *et dans le voisinage-simplexe du point dont* $\mathbf{x}$ *est le* R*-prédécesseur.*

Nous pouvons alors définir : observez la figure 6.7.

Définition 6.8 (Simplexe R-prédécesseur) *Soient* $(\mathbf{x},\sigma,k)$ *un simplexe du voisinage-simplexe de* $\mathbf{x}$ *et* $R\in\{0,1\}^d$ *un vecteur non nul.*

Le simplexe R-prédécesseur *du simplexe* $(\mathbf{x},\sigma,k)$*, noté* $(\mathbf{x},\sigma,k)^{-R}$*, est le simplexe du* R*-voisinage négatif de* $\mathbf{x}$ *qui peut être joint à partir du simplexe* $(\mathbf{x},\sigma,k)$ *par un segment de vecteur directeur* $-R$ *sans quitter le voisinage-simplexe de* $\mathbf{x}$ *: voir la figure 6.7.*

Nous utiliserons la caractérisation suivante :



$R = (0,1)$ $R = (1,1)$ $R = (1,0)$

FIG. 6.7 – Le simplexe R-prédécesseur de chacun des simplexes du voisinage-simplexe de $\mathbf{x}$ est indiqué par une flèche, pour les directions $R = (0,1)$, $R = (1,1)$ et $R = (1,0)$

Lemme 6.1 *Le simplexe R-prédécesseur d'un simplexe $(\mathbf{x}, \sigma, k)$ est donné par l'algorithme suivant :*

Algorithme $Simplexe - R - predecesseur\_de\ (\mathbf{x}, \sigma, k)$:

1. S'il existe une direction i, avec $R_i = 1$, telle que $(\mathbf{x}, \sigma, k)$ appartienne au i-voisinage positif de $\mathbf{x}$, alors calculer récursivement le simplexe R-predecesseur de $(\mathbf{x}, \sigma, k)$ par $(\mathbf{x}, \sigma, k)^{-R} = Simplexe - R - predecesseur\_de((\mathbf{x}, \sigma, k)^{-i})$.

2. S'il existe deux directions $i < j$, avec $R_i = 1$, $R_j = 0$, telles que $(\mathbf{x}, \sigma, k)$ appartienne au i,j-voisinage positif de $\mathbf{x}$, alors calculer récursivement le simplexe R-prédécesseur de $(\mathbf{x}, \sigma, k)$ par $(\mathbf{x}, \sigma, k)^{-R} = Simplexe - R - predecesseur\_de((\mathbf{x}, \sigma, k)^{-i,j})$.

3. Sinon, retourner $(\mathbf{x}, \sigma, k)$ car $(\mathbf{x}, \sigma, k)$ est son propre simplexe R-prédécesseur.

Preuve: S'il existe une direction i avec $R_i = 1$ telle que $(\mathbf{x}, \sigma, k)$ soit un simplexe du i-voisinage positif de $\mathbf{x}$, alors tout segment de vecteur directeur R qui joint $(\mathbf{x}, \sigma, k)^{-i}$ à son simplexe R-prédécesseur peut se prolonger en un segment partant du simplexe $(\mathbf{x}, \sigma, k)$. Par conséquent, $(\mathbf{x}, \sigma, k)$ et $(\mathbf{x}, \sigma, k)^{-i}$ ont même simplexe R-prédécesseur. Un raisonnement similaire prouve que, s'il existe deux directions $i < j$ avec $R_i = 1$, $R_j = 0$ telles que $(\mathbf{x}, \sigma, k)$ soit un simplexe du i,j-voisinage positif de $\mathbf{x}$, alors $(\mathbf{x}, \sigma, k)$ et $(\mathbf{x}, \sigma, k)^{-i,j}$ ont même simplexe R-prédécesseur. Enfin, si l'on n'est pas dans l'un des deux cas précédents, il est

facile de vérifier que le simplexe $(\mathbf{x}, \sigma, k)$ est nécessairement un simplexe du R-voisinage négatif de $\mathbf{x}$, et donc qu'il est son propre simplexe R-prédécesseur. □

6.4 Représentation par les simplexes de la boîte

Nous prouvons maintenant que la représentation par les simplexes de la boîte est une représentation valide des polyèdres temporisés. Nous allons, pour cela, généraliser les arguments du chapitre précédent pour la représentation par les voisinages.

On tire immédiatement des définitions le résultat suivant :

Lemme 6.2 *Un point $\mathbf{x} \in \mathbb{N}^d$ appartient à une i-face si et seulement s'il existe un simplexe $(\mathbf{x}, \sigma, k)$ du i-voisinage positif de $\mathbf{x}$ tel que $c((\mathbf{x}, \sigma, k)) \neq c((\mathbf{x}, \sigma, k)^{-i})$.*

Un point $\mathbf{x} \in \mathbb{N}^d$ appartient à une i, j-face si et seulement s'il existe un simplexe $(\mathbf{x}, \sigma, k)$ du i, j-voisinage positif de $\mathbf{x}$ tel que $c((\mathbf{x}, \sigma, k)) \neq c((\mathbf{x}, \sigma, k)^{-i,j})$.

Utilisons la terminologie suivante :

Définition 6.9 (Point R-invariant) *Soient $\mathbf{x} \in \mathbb{N}^d$ un point et $R \in \{0, 1\}^d$ un vecteur non nul.*

Le point $\mathbf{x}$ est dit R-invariant *si son voisinage-simplexe est invariant par passage au R-prédécesseur : c'est-à-dire, si l'on a $c(\mathbf{x}, \sigma, k) = c((\mathbf{x}, \sigma, k)^{-R})$ pour tout $\sigma \in \Pi$, $0 \leq k \leq d$.*

On obtient alors : voir la figure 6.8.

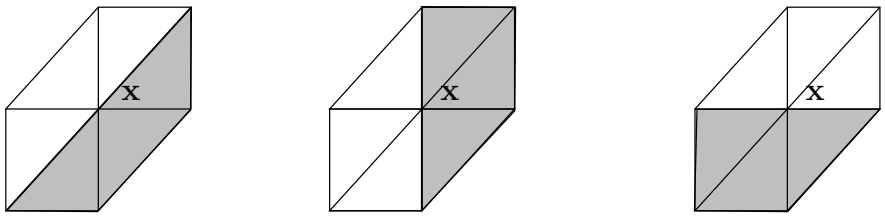


FIG. 6.8 – Illustration du lemme 6.3 : un point $\mathbf{x}$ non-sommet est R-invariant pour un certain vecteur $R \in \{0, 1\}^d$ non nul : considérez $R = (1, 1)$ pour la figure de gauche, $R = (0, 1)$ pour la figure du milieu, et $R = (1, 0)$ pour la figure de droite

Lemme 6.3 (Généralisation du lemme 5.1) *Un point $\mathbf{x}$ n'est pas un sommet si et seulement s'il est R-invariant pour un certain vecteur $R \in \{0, 1\}^d$ non nul.*

Preuve: Supposons que $\mathbf{x}$ ne soit pas un sommet. L'intersection de toutes les faces F_{i,x_i} et F_{i,j,x_i-x_j} est de dimension affine supérieure ou égale à 1 et doit donc au moins contenir un segment. Vue la forme des équations des hyperplans $\mathcal{H}_{i,j,z}$ et $\mathcal{H}_{i,z}$ qui supportent les faces F_{i,x_i} et F_{i,j,x_i-x_j}, on peut supposer ce segment du type $[\mathbf{x}, \mathbf{x}^{-R}]$ pour un certain vecteur $R \in \{0,1\}^d$ non nul. On a nécessairement $\mathbf{x} \notin F_{i,j,x_i-x_j}$ dès que $R_i = 1$ et $R_j = 0$ puisque le segment doit être contenu dans la face F_{i,j,x_i-x_j}. De même, on a $\mathbf{x} \notin F_{i,x_i}$ dès que $R_i = 1$ puisque le segment doit être contenu dans la face F_{i,x_i}. Maintenant, par le lemme 6.1, le simplexe R-prédécesseur de tout simplexe $(\mathbf{x}, \sigma, k)$ s'obtient en partant de $(\mathbf{x}, \sigma, k)$ et en itérant soit des passages aux simplexe i-prédécesseur pour des directions i avec $R_i = 1$, soit des passages au simplexe i,j-prédécesseur pour des directions $i < j$ avec $R_i = 1$ et $R_j = 0$. Par le lemme 6.2, ces opérations préservent les couleurs.

Réciproquement, supposons que le point $\mathbf{x}$ soit R-invariant pour un certain vecteur $R \in \{0,1\}^d$ non nul. Soit i une direction telle que $R_i = 1$. On doit avoir $\mathbf{x} \notin F_{i,x_i}$ car sinon, par le lemme 6.2, il existerait un simplexe $(\mathbf{x}, \sigma, k)$ du i-voisinage positif de $\mathbf{x}$ tel que $c((\mathbf{x}, \sigma, k)^{-i}) \neq c(\mathbf{x}, \sigma, k)$: on obtiendrait une contradiction car l'algorithme du lemme 6.1 prouve que ces deux simplexes ont même simplexe R-prédécesseur. Un raisonnement similaire prouve que l'on doit avoir $\mathbf{x} \notin F_{i,j,x_i-x_j}$ dès que $R_i = 1$ et $R_j = 0$. Vues les équations des hyperplans $\mathcal{H}_{i,j,z}$ et $\mathcal{H}_{i,z}$, ceci implique que le segment $[\mathbf{x}^{-R}, \mathbf{x}]$ doit appartenir à chacune des faces F_{i,x_i} et à chacune des faces F_{i,j,x_i-x_j}. Par conséquent la dimension affine de l'intersection de toutes les faces F_{i,x_i} et F_{i,j,x_i-x_j} est au moins égale à un. □

Le lemme 5.2 du chapitre précédent se généralise alors en :

Lemme 6.4 (Généralisation du lemme 5.2) *Soient* $\mathbf{x}$ *un point non-sommet et* $R' \in \{0,1\}^d$ *un vecteur non nul. Supposons* $c(\mathbf{x}, \sigma, k) = c((\mathbf{x}, \sigma, k)^{-R'})$ *pour tout* $\sigma \in \Pi$, $0 \leq k < d$.

Alors le point $\mathbf{x}$ *est* R'*-invariant.*

Preuve: Puisque $\mathbf{x}$ n'est pas un sommet, $\mathbf{x}$ est R-invariant pour un certain vecteur $R \in \{0,1\}^d$ non nul. Soit $(\mathbf{x}, \sigma, k)$ un simplexe tel que $k = d$. En utilisant le lemme 6.1, il est facile de vérifier qu'un tel simplexe vérifie $(((\mathbf{x}, \sigma, k)^{-R})^{-R'})^{-R} = ((\mathbf{x}, \sigma, k)^{-R'})^{-R}$. Il suffit alors d'écrire $c(\mathbf{x}, \sigma, k) = c((\mathbf{x}, \sigma, k)^{-R}) = c(((\mathbf{x}, \sigma, k)^{-R})^{-R'})$ $= c((((\mathbf{x}, \sigma, k)^{-R})^{-R'})^{-R}) = c(((\mathbf{x}, \sigma, k)^{-R'})^{-R}) = c((\mathbf{x}, \sigma, k)^{-R'})$. □

Nous pouvons alors prouver la validité de la représentation par les simplexes de la boîte (observons qu'un simplexe $(\mathbf{x}, \sigma, k)$ appartient au voisinage-boîte de $\mathbf{x}$ si et seulement si $k = d$) :

Théorème 6.1 (Représentation par les simplexes de la boîte) *La représentation par les simplexes de la boîte est une représentation correcte et canonique des polyèdres temporisés.*

Preuve: Pour calculer la couleur des simplexes du voisinage-boîte d'un point $\mathbf{x}$, il suffit d'utiliser l'algorithme suivant : voir la figure 6.9.

Algorithme *Couleur_des_simplexes_de_la_boite_de*($\mathbf{x}$):

1. Si le point $\mathbf{x}$ est un sommet alors retourner la couleur de chacun des simplexes $(\mathbf{x}, \sigma)$, $\sigma \in \Pi$, en utilisant la représentation du polyèdre.
2. Si $\mathbf{x}$ a une coordonnée négative, alors retourner $c(\mathbf{x}, \sigma) = \textit{blanc}$ pour tout $\sigma \in \Pi$.
3. Sinon
   - 3.1 Pour chacune des boîtes élémentaires $\mathbf{y} \in \mathbb{N}^d$, $\mathbf{y} \neq \mathbf{x}$ voisines de celle de $\mathbf{x}$, appeler récursivement *Couleur_des_simplexes_de_la_boite_de*($\mathbf{y}$) pour déterminer la couleur des simplexes de la boîte élémentaire de $\mathbf{y}$.
   - 3.2 Rechercher un vecteur $R \in \{0,1\}^d$ non nul tel que $c(\mathbf{y}, \sigma, k) = c((\mathbf{y}, \sigma, k)^{-R})$ pour tout $\sigma \in \Pi, 0 \leq k < d$.
   - 3.3 Retourner $c(\mathbf{x}, \sigma) = c((\mathbf{x}, \sigma)^{-R})$ pour tout $\sigma \in \Pi$.

□

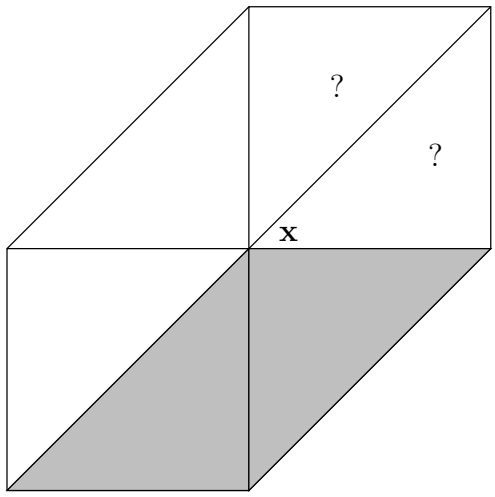


FIG. 6.9 – Illustration de l'algorithme du théorème 6.1. Si l'on sait que le point $\mathbf{x}$ n'est pas un sommet, et si l'on a déterminé récursivement la couleur de chacun des simplexes non-marqués d'un point d'interrogation, alors on peut déterminer la couleur des simplexes du voisinage-boîte de $\mathbf{x}$.

Si l'on utilise des techniques similaires à celles discutées dans le chapitre précédent, la procédure précédente s'effectue en moins de $O(2^d d(d!)n^d)$ étapes, c'est-à-dire en moins de $O(n^d)$ étapes lorsque la dimension d est fixée.

6.5 Représentation par les simplexes du voisinage

Nous allons maintenant prouver que la représentation par les simplexes du voisinage permet de déterminer la couleur d'un simplexe en temps $O(n \log n)$ lorsque la dimension d est fixée. Le principe consiste à généraliser les arguments que nous avions utilisés dans le chapitre précédent pour la représentation par les voisinages.

Contrairement à ce qui se passe pour les polyèdres orthogonaux, lorsque l'on intersecte un polyèdre temporisé avec un hyperplan $\mathcal{H}_{i,z}$, on n'obtient pas toujours un polyèdre temporisé. Il nous faut donc définir la notion de section plus subtilement : voir la figure 6.10.

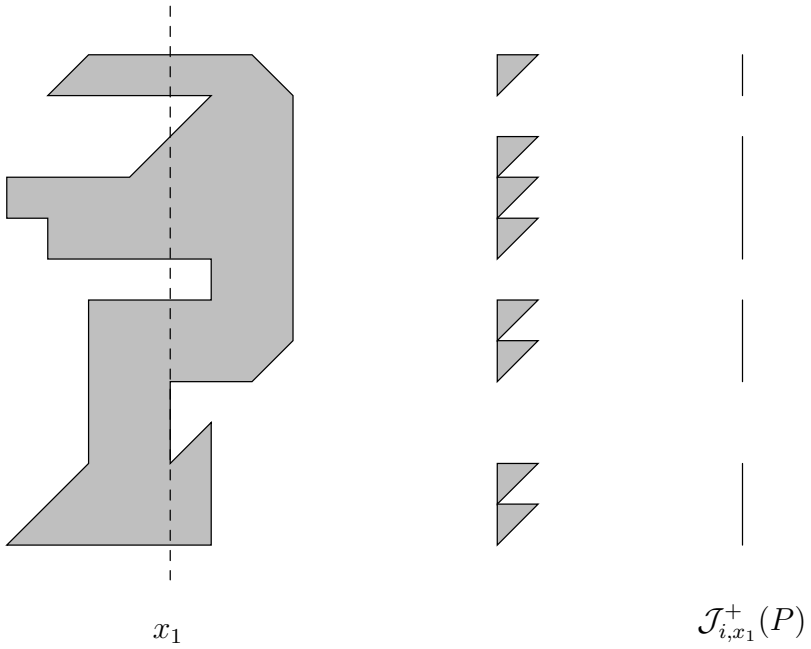


FIG. 6.10 – Un polyèdre temporisé de dimension 2, les simplexes des i-voisinages positifs des points de $\mathcal{H}_{i,z}$, et la 1-section positive en z du polyèdre.

Définition 6.10 (i-section) *Soient P un polyèdre temporisé, z un entier, $i \in \{1, \ldots, d\}$ une direction et $\mathbf{x}$ un point.*

La i-section positive en z de P, *dénotée par $\mathcal{J}^+_{i,z}(P)$, est le polyèdre temporisé de dimension $d-1$ obtenu en projetant orthogonalement sur $\mathcal{H}_{i,z}$ les simplexes des i-voisinages positifs des points de $\mathcal{H}_{i,z}$.*

Il est alors facile d'établir :

Observation 6.2 (Généralisation de l'observation 5.1) *Soit P un polyèdre. Soit $\overline{P} = \mathcal{J}^+_{i,z}(P)$ une i-section positive de P.*

Les assertions suivantes sont équivalentes :

1. *Le point $\overline{\mathbf{x}} = (x_1, \ldots, x_{d-1})$ de $\overline{P}$ est un sommet*
2. *Le i-voisinage positif du point $\mathbf{x} = (x_1, \ldots, x_{i-1}, z, x_i, \ldots, x_{d-1})$ de P vérifie l'assertion : pour tout vecteur $R' \in \{0,1\}^d$ non nul tel que $R'_i = 0$, il existe un simplexe $(\mathbf{x}, \sigma, k)$ du i-voisinage positif de $\mathbf{x}$ vérifiant $c(\mathbf{x}, \sigma, k) \neq c((\mathbf{x}, \sigma, k)^{-R'})$.*

Comme dans le chapitre précédent, nous définissons :

Définition 6.11 (Voisinage de sommet en dimension d-1) *Le i–voisinage positif d'un point $\mathbf{x} \in \mathbb{N}^d$ sera dit* voisinage de sommet en dimension $d-1$ *s'il vérifie la condition 2 de l'observation précédente : c'est-à-dire si, pour tout vecteur $R' \in \{0,1\}^d$ non nul tel que $R'_i = 0$, il existe un simplexe $(\mathbf{x}, \sigma, k)$ du i-voisinage de $\mathbf{x}$ vérifiant $c(\mathbf{x}, \sigma, k) \neq c((\mathbf{x}, \sigma, k)^{-R'})$.*

Nous voulons maintenant généraliser le lemme 5.3 du chapitre précédent. Ceci est plus problématique : en effet, nous sommes obligés de faire les remarques suivantes que l'on tire facilement des définitions : voir la figure 6.11.

Observation 6.3 *Soient $\mathbf{y} \in \mathbb{N}^d$ un point, $i \in \{1, \ldots, d\}$ une direction, et $R \in \{0,1\}^d$ un vecteur avec $R_i = 1$.*

Les assertions suivantes sont équivalentes :

1. *Le point $\mathbf{y} + \epsilon R$, pour $\epsilon > 0$ très petit, appartient à une arête de direction R.*
2. *Le R-voisinage positif de $\mathbf{y}$ vérifie la condition suivante : si l'on appelle $\mathbf{y}'$ le point fictif dont les coordonnées seraient données par $\mathbf{y}' = \mathbf{y} + R$ et dont le voisinage-simplexe serait donné par $\mathcal{N}^{-R}(\mathbf{y}') = \mathcal{N}^{+R}(\mathbf{y})$ et par $c((\mathbf{x}, \sigma, k)) = c((\mathbf{x}, \sigma, k)^{-R})$ pour tout autre simplexe, alors le i-voisinage positif de $\mathbf{y}'$ est un voisinage de sommet en dimension $d-1$.*

On a aussi besoin du résultat suivant qui se déduit immédiatement des définitions : observez la figure 6.11.

Observation 6.4 *Soient $\mathbf{y} \in \mathbb{N}^d$ un point et $R \in \{0,1\}^d$ un vecteur avec $R_i = 1$. Supposons que le point $\mathbf{y}$ vérifie l'une des deux assertions équivalentes de l'observation précédente.*

Si $\mathbf{y}$ n'est pas un sommet, alors $\mathbf{y}$ est R-invariant.

Nous appellerons *sommet R-prédécesseur* d'un point $\mathbf{x} \in \mathbb{N}^d$ un sommet que l'on rencontre en partant de $\mathbf{x}$ et en se déplaçant dans la direction $-R$: en d'autres termes, un sommet $\mathbf{y}$ de la forme $\mathbf{y} = \mathbf{x} - R\alpha$ pour $\alpha \in \mathbb{N}$, $\alpha \geq 1$. Appelons *premier sommet R-prédécesseur* du point $\mathbf{x}$, noté $\mathbf{x}^{\leftarrow R}$, le premier de ces sommets que l'on rencontre.

Nous pouvons alors généraliser le lemme 5.3 du chapitre précédent par :

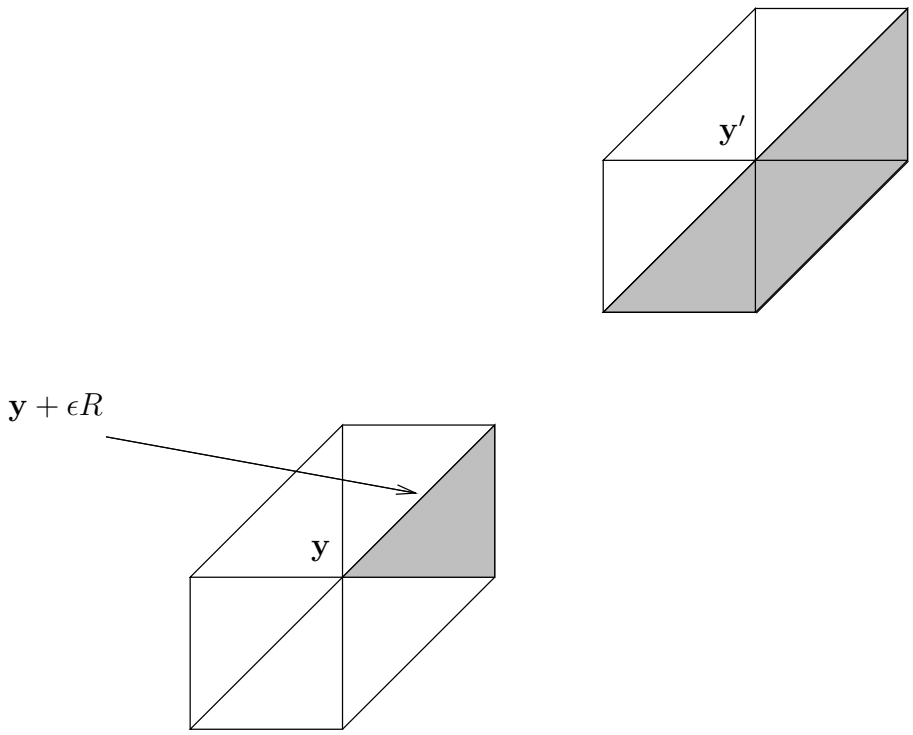


FIG. 6.11 – Illustration de l'observation 6.3 pour $R = (1, 1)$: le point $\mathbf{y} + \epsilon R$ appartient à une arête dans la direction R si et seulement si le i-voisinage positif du point fictif $\mathbf{y}'$ est voisinage de sommet en dimension $d - 1$: voir la définition 6.11

Lemme 6.5 (Généralisation du lemme 5.3) *Soient P un polyèdre et $\overline{P} = \mathcal{J}^+_{i,z}(P)$ une i-section positive de P.*

Un point $\overline{\mathbf{x}} = (x_1, \ldots, x_{d-1})$ est un sommet de $\overline{P}$ si et seulement si au moins l'une des deux conditions suivantes est vérifiée :

1. *Le point $\mathbf{x} = (x_1, \ldots, x_{i-1}, z, x_i, \ldots, x_d)$ est un sommet de P tel que son i-voisinage positif soit un voisinage de sommet en dimension $d-1$.*
2. *Il existe un vecteur $R \in \{0,1\}^d$ avec $R_i = 1$ tel que $\mathbf{x}$ possède un premier sommet R-prédécesseur $\mathbf{y} = \mathbf{x}^{\leftarrow R}$ qui vérifie l'une des deux assertions équivalentes de l'observation 6.3.*

Preuve: Par l'observation 6.2, $\overline{\mathbf{x}}$ est un sommet de $\overline{P}$ si et seulement si le i-voisinage positif du point $\mathbf{x} = (x_1, \ldots, x_{i-1}, z, x_i, \ldots, x_d)$ est un voisinage de sommet en dimension $d-1$.

Nous prouvons maintenant le sens direct. Si $\mathbf{x}$ est un sommet, il n'y a rien à prouver. Sinon, $\mathbf{x}$ est R-invariant pour un certain vecteur $R \in \{0,1\}^d$ non nul. On doit avoir $R_i = 1$ pour ne pas contredire le point 2 de l'observation 6.2. Soit $\mathbf{y} = \mathbf{x} - \alpha R$ le point, avec $\alpha \in \mathbb{N}$, $\alpha > 0$ minimal, tel que $\mathbf{y}$ ne soit pas R-invariant. Les points $\mathbf{z} \neq \mathbf{y}$ entre $\mathbf{x}$ et $\mathbf{y}$ sont R-invariants. On en déduit, d'une part que ces points ne sont donc pas des sommets, et que d'autre part, le i-voisinage positif de $\mathbf{y} + R$ est égal à celui de $\mathbf{x}$. Puisque $\mathbf{y} + R$ est R-invariant, $\mathbf{y}$ vérifie l'assertion 2 de l'observation 6.3. Puisque $\mathbf{y}$ n'est pas R-invariant, on déduit de l'observation 6.4 que $\mathbf{y}$ est le sommet qui s'écrit $\mathbf{y} = \mathbf{x}^{\leftarrow R}$.

Réciproquement, supposons que $\mathbf{x}$ ne soit pas un sommet, et qu'il existe un vecteur $R \in \{0,1\}^d$, $R_i = 1$ tel que si l'on pose $\mathbf{y} = \mathbf{x}^{\leftarrow R}$, alors $\mathbf{y}$ vérifie l'une des deux assertions équivalentes de l'observation 6.3. Par un raisonnement similaire au paragraphe précédent, les sommets entre $\mathbf{y}$ et $\mathbf{x}$, y compris $\mathbf{x}$ doivent être R-invariants puisque sinon il existerait un sommet entre $\mathbf{y}$ et $\mathbf{x}$. $\mathbf{y} + R$ et $\mathbf{x}$ doivent donc avoir même i-voisinage positif. Par conséquent, le i-voisinage positif de $\mathbf{x}$ doit être un voisinage de sommet en dimension $d-1$. □

Comme nous l'avons fait dans le chapitre précédent, nous en déduisons maintenant qu'il est possible de calculer les sections d'un polyèdre temporisé donné :

Proposition 6.1 (Calcul des sections) *Supposons la dimension d fixée.*

Etant donné un polyèdre temporisé P, une direction i et un entier z, on peut calculer en temps $O(n \log(n))$ une représentation par les simplexes du voisinage de $\overline{P} = \mathcal{J}^+_{i,z}(P)$ à partir d'une représentation par les simplexes du voisinage de P.

Preuve: C'est une généralisation facile de la proposition 5.1 : les sommets de $\overline{P}$ sont, d'une part les sommets de P de $\mathcal{H}_{i,z}$ dont le i-voisinage positif est voisinage de sommet en dimension $d-1$, et d'autre part les points de $\mathcal{H}_{i,z}$ qui possèdent un premier sommet R-prédécesseur qui vérifie la condition du point 2 de l'observation 6.3, pour un certain vecteur $R \in \{0,1\}^d, R_i = 1$. □

On peut alors obtenir le résultat attendu : l'existence d'un algorithme qui calcule la couleur d'un point donné en temps $O(n \log n)$:

Théorème 6.2 *Supposons la dimension d fixée.*

En utilisant la représentation par les simplexes du voisinage, on peut déterminer la couleur d'un simplexe $(\mathbf{x}, \sigma)$ *en temps* $O(n \log n)$ *où* n *est le nombre de sommets du polyèdre.*

Preuve: Déterminer la couleur du simplexe $(\mathbf{x}, \sigma)$ d'un polyèdre P de dimension d est équivalent à déterminer la couleur de sa projection sur $\mathcal{H}_{\sigma(1),x_{\sigma(1)}}$ dans le polyèdre $\overline{P} = \mathcal{J}^{+}_{\sigma(1),x_{\sigma(1)}}(P)$ de dimension $d-1$. Si l'on répète ce principe $d-1$-fois le problème se réduit au cas trivial de la dimension 1. □

Il n'a pas échappé à notre lecteur que la complexité dépend non linéairement de la dimension : lorsque l'on prend en compte la dimension d, la complexité de l'algorithme précédent est en $O(n(d^2 2^d \log n + 4^d(d+1)!))$. Cette complexité peut sembler inacceptable. Mais observons, d'une part qu'une boîte élémentaire possède un nombre de simplexes élémentaire qui dépend factoriellement de la dimension, et d'autre part, que les méthodes usuelles de représentation des polyèdres temporisés (diagrammes de décisions binaires, matrices de différences bornées : voir [4]) se limitent aux polyèdres convexes.

6.6 Algorithmes sur les polyèdres

Nous présentons maintenant comment on peut implémenter les opérations requises par les algorithmes de vérification et de synthèse sur les automates temporisés : tests de comparaisons et opérations booléennes entre polyèdres temporisés, et opération "passage du temps" : voir [4].

6.6.1 Tests de comparaison

Puisque les représentations étudiées sont canoniques, tester si deux polyèdres sont égaux est un problème trivial : il suffit de comparer les représentations des deux polyèdres.

6.6.2 Détection des faces

Comme dans le chapitre précédent pour les polyèdres orthogonaux, il est facile de prouver que les sommets des faces des polyèdres temporisés sont nécessairement des sommets du polyèdre. Par conséquent, pour détecter les faces d'un polyèdre temporisé, il suffit de parcourir les sommets et d'examiner leurs voisinages.

6.6.3 Opérations booléennes

Comme dans le chapitre précédent pour les polyèdres orthogonaux, il est facile de prouver qu'il n'y a qu'un nombre fini de points candidats pour être sommets d'une intersection entre deux polyèdres temporisés. Par conséquent, pour calculer l'intersection de deux polyèdres temporisés, il suffit de calculer les voisinages de chacun des candidats et de ne garder que les candidats qui sont bien des sommets.

6.6.4 Passage du temps

L'opération "passage du temps" requise par les algorithmes sur les automates temporisés (voir [4, 7]) peut se traiter avec une technique similaire : il n'y a qu'un nombre fini de points candidats pour être sommets du polyèdre résultat. Il suffit de calculer les voisinages de chacun des candidats, et de ne garder que les candidats qui sont effectivement des sommets.

6.7 Implémentation

Dans ce chapitre, nous avons proposé des schémas de représentation pour les polyèdres temporisés. Nous avons décrit ou esquissé des algorithmes qui permettent leur utilisation pratique dans les méthodes de la vérification.

Ces résultats sont à l'heure actuel purement théoriques. En effet, les algorithmes présentés dans ce chapitre n'ont pas été implémentés à ce jour.

Nous espérons pouvoir mesurer la performance pratiques des algorithmes dans un futur proche.

CHAPITRE

7 Programmation avec des systèmes DCPM

7.1 Introduction

La puissance de calcul des systèmes dynamiques à temps discret a été étudiée intensément ces dernières années : voir par exemple [23, 44, 45, 46, 60, 81]. Il a été prouvé que ces systèmes ont une puissance de calcul égale à celle des machines de Turing lorsqu'ils admettent une description rationnelle [23, 60, 61, 80], et égale aux classes de complexité non-uniformes dans le cas contraire [23, 31, 32, 81].

La puissance de calcul des systèmes dynamiques à temps continu est beaucoup moins connue. Plusieurs modèles de machines de calcul à temps continu ont été proposés : Shannon a introduit en 1941 le "General Purpose Analog Computer" [78] qui est capable de réaliser des additions, des multiplications et des intégrations de fonctions arbitraires. En 1993, Rubel a proposé une extension de ce modèle appelée "Extended Analog Computer"[69] qui est capable de passer à la limite ou de résoudre certaines équations aux limites. Il existe des caractérisations mathématiques des fonctions calculables dans ces modèles mais ces caractérisations ne permettent pas de comparer la puissance de calcul de ces modèles avec celle des machines de Turing : voir [52, 62, 66, 69, 78].

Récemment, Moore a introduit une "Théorie de la récursivité pour les calculs sur les réels" [62] et a proposé des premiers résultats de comparaison entre la puissance de calcul des systèmes à temps continu et la puissance de calcul des machines de Turing. Asarin et Maler ont présenté dans [8] des résultats similaires pour les systèmes DCPM. Toutefois tous ces résultats ne donnent que des bornes inférieures sur la puissance de calcul des systèmes à temps continu et ne présentent pas de caractérisation de la puissance de calcul des systèmes étudiés.

Dans ce chapitre et le suivant, nous présentons une caractérisation complète de la puissance de calcul des systèmes DCPM étudiés par [8]. Nous prouvons que la puissance de ces systèmes se relie aux classes de langages de la hiérarchie hyperarithmétique : les langages semi-reconnus par les systèmes DCPM en dimension $d = 2k + 3$, $k \geq 0$, sont exactement les langages du ω^kème niveau de la hiérarchie hyperarithmétique. Les langages semi-reconnus par les systèmes DCPM en dimension $d = 2k + 4$, $k \geq 0$, sont exactement les langages du $\omega^k + 1$ème niveau de la hiérarchie hyperarithmétique.

Du point de vue de la vérification automatique, nos résultats impliquent qu'il existe des systèmes DCPM de dimension 4 pour lesquels le problème de l'atteignabilité ne peut être résolu par aucun algorithme ni même par aucun semi-algorithme. Ce résultat met en évidence les limites des techniques classiques de vérification automatique de propriétés, même si l'on se restreint à des systèmes très simples.

Du point de vue des modèles réels, nous présentons une caractérisation complète de la puissance de calcul des systèmes DCPM. C'est, à notre connaissance, la première fois qu'une telle caractérisation est donnée pour une classe de systèmes/machines à temps continu.

Dans ce chapitre, nous prouvons que l'on peut reconnaître des langages hyperarithmétiques avec des systèmes DCPM. Dans le chapitre suivant nous prouverons que tout ensemble reconnu par un système DCPM est hyperarithmétique. Les résultats présentés dans ces deux chapitres ont donné lieu aux publications [19, 20, 18].

Le plan de ce chapitre est le suivant : dans la section 7.2, nous prouvons le théorème 3.2 du chapitre 3. Autrement dit, nous prouvons que l'on peut simuler une machine de Turing par un système DCPM. Dans la section 7.3, nous prouvons que tout langage du second niveau de la hiérarchie arithmétique est semi-reconnu par un système DCPM de dimension 4. Dans la section 7.4, nous prouvons que tout langage de la hiérarchie arithmétique peut être reconnu en dimension 5. Nous généralisons ensuite ce résultat de façon à prouver que, pour $k \geq 0$, tout langage du ω^kème niveau de la hiérarchie hyperarithmétique est semi-reconnu par un système DCPM de dimension $2k+3$ et que tout langage du ω^k+1ème niveau est semi-reconnu par un système DCPM de dimension $2k+4$.

Rappelons que les systèmes DCPM ont été définis dans le chapitre 2.

7.2 Simulation d'une machine de Turing

Dans cette section, nous prouvons le théorème 3.2 du chapitre 3. En d'autres termes, nous prouvons que toute machine de Turing peut être simulée par un système DCPM.

Nous allons utiliser pour cela le lemme 3.1 prouvé dans le chapitre 3. Reformulons le résultat que nous avions obtenu : nous appelons *pavé de* $\mathbb{R}^d$ tout produit cartésien de d intervalles ouverts ou fermés de $\mathbb{R}$.

Observation 7.1 (Reformulation du lemme 3.1) *Soit M une machine de Turing à un ruban. Soit $C = \Sigma^\omega \times \Sigma^\omega \times Q$ l'espace de ses configurations.*

Il existe

- *une fonction $g_M : [0,1]^2 \to [0,1]^2$ du type suivant :*
  1. *g_M est affine par morceaux sur des pavés : il existe un nombre fini de pavés inclus dans $[0,1]^2$ sur lesquels g_M est affine*

2. *sur chacun de ces pavés, g est du type $g : x \mapsto (\lambda_1 x_1 + \beta_1, \lambda_2 x_2 + \beta_2)$ avec $\lambda_1, \lambda_2 \in \mathbb{Q}^+$, $\beta_1, \beta_2 \in \mathbb{Q}$*

– *une fonction d'encodage $\nu : C \to [0,1]^2$ telle que :*

1. *$\nu(\Sigma^* \times \Sigma^* \times Q) \subset \mathbb{Q}^2$*
2. *il existe une constante $\beta_M \in \mathbb{Q}$ telle que $\nu(w_1, w_2, q) \in [0, \beta_M] \times [0,1]$ si et seulement si q est un état interne acceptant de M*

telles que le diagramme suivant commute :

$$\begin{array}{ccc} C & \xrightarrow{\vdash} & C \\ {\scriptstyle\nu}\downarrow & & \downarrow{\scriptstyle\nu} \\ [0,1]^2 & \xrightarrow{g_M} & [0,1]^2 \end{array}$$

(c'est-à-dire telles que $g_M(\nu(c)) = \nu(c')$ pour toutes configurations $c, c' \in C$ vérifiant $c \vdash c'$).

Pour prouver le théorème 3.2, nous allons prouver dans un premier temps que les fonctions du type de la fonction g_M de l'observation précédente sont calculables par un système DCPM, puis dans un second temps que l'on peut simuler les itérations d'une fonction par un système DCPM.

Dans la section 7.2.1 nous précisons formellement ce que nous appelons une fonction DCPM-calculable et nous prouvons que les fonctions du type $x \mapsto \lambda x + \beta$ sont DCPM-calculables. Dans la section 7.2.2 nous discutons la possibilité de construire des chemins dans les systèmes DCPM et nous en déduisons que la fonction g_M est DCPM-calculable. Nous en déduisons alors le théorème 3.2. Dans la section 7.2.3 nous discutons la dimension des systèmes DCPM impliqués.

Dans ce chapitre, $e_1, \ldots, e_d$ désigneront les vecteurs de la base canonique de $\mathbb{R}^d$ et 0 désignera le point de coordonnées $(0, \ldots, 0)$ dans cette base.

7.2.1 Notion de fonction DCPM-calculable

Dans cette sous-section, nous définissons ce que nous appelons une fonction DCPM-calculable. Rappelons que les systèmes DCPM ont été introduits dans le chapitre 2. Nous utiliserons à vrai dire uniquement des systèmes DCPM bornés dans ce chapitre. C'est-à-dire des systèmes définis par :

Définition 7.1 (Système DCPM borné) *Un* système à dérivée constante par morceaux (DCPM) borné *est un système dynamique à temps continu (X, f) où $f : X \to \mathbb{R}^d$ est une fonction constante par morceaux et X est un polyèdre rationnel* borné *de $\mathbb{R}^d$.*

Nous définissons alors la notion de *port d'un système DCPM borné* comme : observez la figure 7.1.

Définition 7.2 (Port d'un système DCPM) *Soit $H = (X, f)$ un système DCPM borné de dimension d. Soit p un entier avec $p < d$.*

Un port de dimension p de H *est la donnée d'un polyèdre P borné de dimension p inclus dans la frontière de X, et d'une base affine orthonormale B de P.*

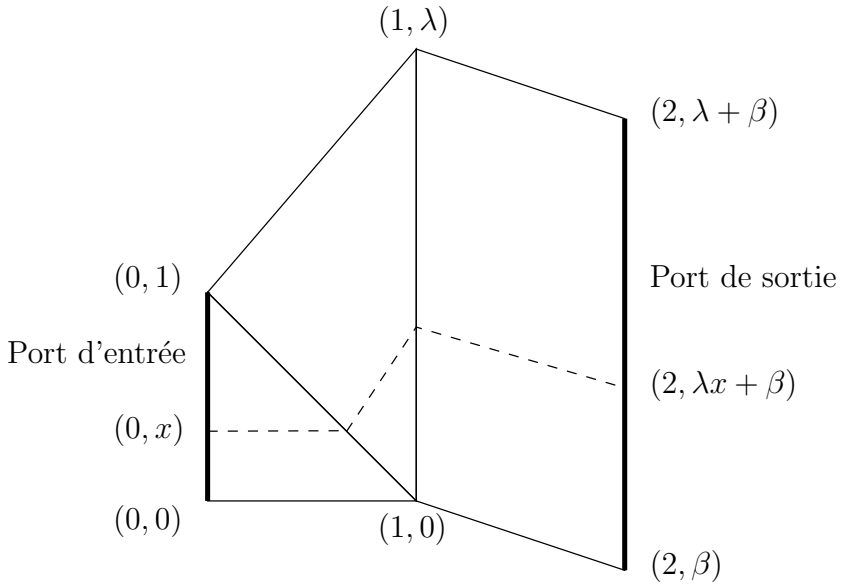


FIG. 7.1 – Un système DCPM borné de dimension 2 qui calcule la fonction $g : [0, 1] \to [\beta, \lambda + \beta]$, $g : x \mapsto \lambda x + \beta$. Le port d'entrée est défini par la base $(0, e_2)$ et le polyèdre d'équation $\{0\} \times [0, 1]$ dans la base canonique de $\mathbb{R}^2$. Le port de sortie est défini par la base $(0', e_2)$ où O' est le point de coordonnées $0' = (2, 0)$ et par le polyèdre d'équation $\{2\} \times [\beta, \lambda + \beta]$ dans la base canonique de $\mathbb{R}^2$

Nous pouvons alors définir ce que nous appelons une fonction DCPM-calculable : voir la figure 7.1.

Définition 7.3 (Fonction DCPM-calculable) *Une fonction (éventuellement partielle) g d'un polyèdre borné S de dimension p vers un polyèdre borné S' de dimension q est dite* DCPM-calculable *en dimension d si :*

- *il existe un système DCPM $H = (X, f)$ bien défini*[1] *borné de dimension d.*
- *il existe un port (P_1, B_1) de H de dimension p.*
- *il existe un port (P_2, B_2) de H de dimension q.*

tels que, pour tout x appartenant au domaine de f, la trajectoire de H partant du point du polyèdre P_1 de coordonnées x dans la base B_1 atteint à un temps fini le polyèdre P_2 au point de coordonnées $g(x)$ dans la base B_2.

[1] C'est-à-dire tel que pour tout point $x \in X$ il existe $\epsilon > 0$ avec $f(x') = f(x)$ pour tout x' du segment $[x, x + \epsilon f(x)] \cap X$ de $\mathbb{R}^d$: voir la définition 2.12 du chapitre 2.

Par exemple, la figure 7.1 montre que la fonction $x \mapsto \lambda x + \beta$ est DCPM calculable :

Lemme 7.1 (Réalisation de $x \mapsto \lambda x + \beta$) *Soient $\lambda \in \mathbb{Q}^+, \beta \in \mathbb{Q}$ deux constantes.*

La fonction $g : x \in [0,1] \mapsto \lambda x + \beta$ est DCPM-calculable en dimension 2.

Preuve: Il suffit d'utiliser le système DCPM de dimension 2 représenté sur la figure 7.1. Formellement, la fonction g est calculée par le système DCPM $H = (X, f)$ de dimension 2 défini par

$$f(x_1, x_2) = \begin{cases} (0,1) & \text{si } 0 \leq x_2 < x_1 \leq 1 \\ (1-\lambda, 1) & \text{si } 0 \leq x_1 \leq 1,\ x_1 \leq x_2 \leq 1 + (\lambda - 1)x_1,\ (x_1, x_2) \neq (1,0) \\ (\beta, 1) & \text{si } 1 < x_2 \leq 2,\ \beta(x_1 - 1) \leq x_2 \leq \lambda + \beta(x_1 - 1) \\ (\beta, 1) & \text{si } (x_1, x_2) = (1,0) \end{cases}$$

□

En plongeant ce système en dimension p on obtient aussi :

Corollaire 7.1 (Réalisation de $x_i \mapsto \lambda x_i + \beta$) *Soient $\lambda \in \mathbb{Q}^+$ et $\beta \in \mathbb{Q}$ deux constantes et $1 \leq i \leq p$ deux entiers.*

La fonction $g : (x_1, \ldots, x_p) \in [0,1]^p \mapsto (x_1, \ldots, x_{i-1}, \lambda x_i + \beta, x_{i+1}, \ldots, x_p)$ est DCPM-calculable en dimension $p + 1$.

Preuve: Si $H = (X, f)$ est le système DCPM du lemme 7.1 qui calcule $x \in [0,1] \mapsto \lambda x + \beta$, alors le système DCPM défini par $H' = (X', f')$ avec $X' = [0,1]^{i-1} \times X \times [0,1]^{p-i}$ et $f'(x_1, \ldots, x_{p+1}) = (0, \ldots, 0, f(x_i, x_{i+1}), 0, \ldots, 0)$ calcule g. □

7.2.2 Construction de chemins et conséquences

Après ces définitions et premiers exemples de fonctions DCPM-calculables, nous discutons maintenant la possibilité d'ajouter à un système DCPM des chemins.

La remarque de base est la suivante : étant donné un système DCPM borné H et deux ports (P_1, B_1) et (P_2, B_2) de H de mêmes dimensions, on peut ajouter à H un *chemin* de (P_1, B_1) vers (P_2, B_2). C'est-à-dire, on peut ajouter à H des régions telles que toute trajectoire partant d'un point du polyèdre P_1 de coordonnées x dans la base B_1 atteigne à un temps borné le polyèdre P_2 en le point de coordonnées x dans la base B_2 : voir la figure 7.2.

En observant la figure 7.2 ou la figure 7.3, il est facile de se convaincre intuitivement de ce fait. Si l'on veut être formel, le résultat est le suivant :

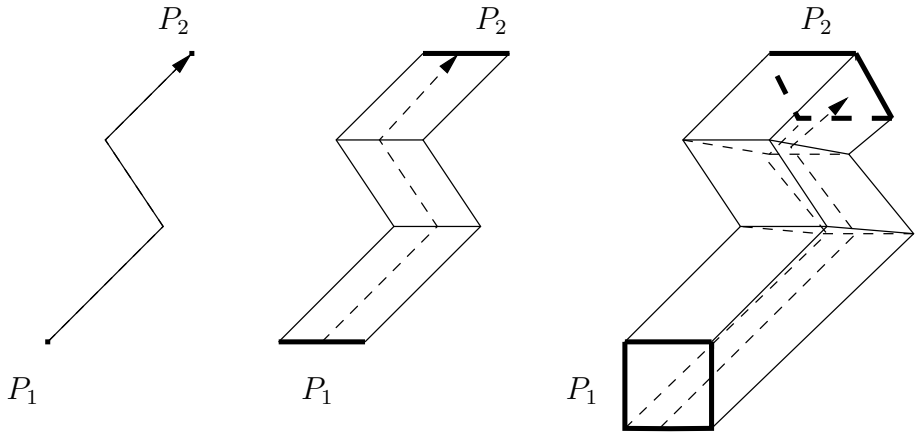


FIG. 7.2 – Réalisation d'un chemin entre un port d-dimensionnel (P_1, B_1) et un port d-dimensionnel (P_2, B_2) pour $d = 1, 2, 3$

Lemme 7.2 (Ajout d'un chemin à un système DCPM) *Soit $H = (X, f)$ un système DCPM borné de dimension d. Soient (P_1, B_1) et (P_2, B_2) deux ports de dimension p de H avec $p < d$.*

Supposons que le complémentaire du polyèdre X dans $\mathbb{R}^d$ possède une unique composante connexe. Supposons que X privé de P_1 et de P_2 soit un sous-ensemble fermé de $\mathbb{R}^d$.

Alors il est possible d'ajouter au système DCPM H des régions bornées telles que toute trajectoire partant d'un point du polyèdre P_1 de coordonnées $(x_1, \ldots, x_p)$ dans la base B_1 atteigne à un temps borné le point du polyèdre P_2 de même coordonnées dans la base B_2 : voir la figure 7.2.

Preuve: Puisqu'un sous-ensemble connexe de $\mathbb{R}^d$ est connexe par arc (voir [67]) il existe un arc totalement inclus dans le complémentaire de X privé de P_1 et de P_2 qui joint un sommet v_1 du polyèdre P_1 à un sommet v_2 du polyèdre P_2. Par la preuve même de la connexité par arc de $\mathbb{R}^d$, on peut supposer cet arc égal à une ligne brisée [67]. En ajoutant éventuellement p régions de sommet v_1 et p régions de sommet v_2 comme sur la figure 7.3, on peut réaliser une contraction dans le voisinage de P_1, annulée par une dilatation dans le voisinage de P_2, de façon à rester arbitrairement proche de la ligne brisée. Il suffit alors de construire des régions qui obligent les trajectoires à suivre cette ligne brisée comme sur les figures 7.2 et 7.3. □

En utilisant ce résultat, nous sommes prêts à montrer que l'ensemble des fonctions DCPM-calculables est clos par composition : voir la figure 7.4 ou la représentation symbolique de la figure 7.5.

Lemme 7.3 (Clôture par composition) *Soit $g : [0, 1]^p \to [0, 1]^q$ une fonction*

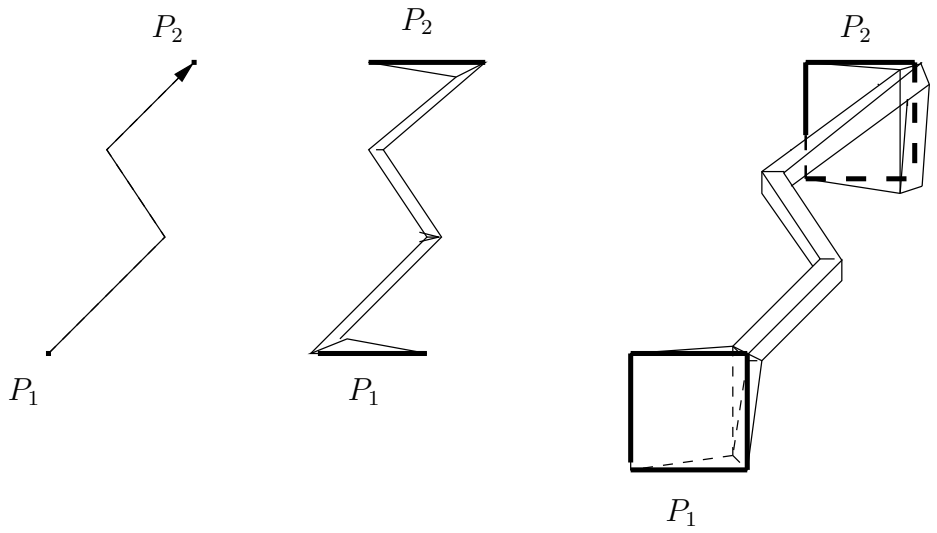


FIG. 7.3 – Preuve du lemme 7.2 : on réalise un chemin entre un port (P_1, B_1) et un port (P_2, B_2) par l'ajout de régions qui obligent les trajectoires à suivre une ligne brisée qui joint deux sommets des polyèdres P_1 et P_2. En insérant p régions près de P_1 qui réalisent une contraction, et p régions près de P_2 qui annulent cette contraction on peut rendre le chemin arbitrairement proche de la ligne brisée

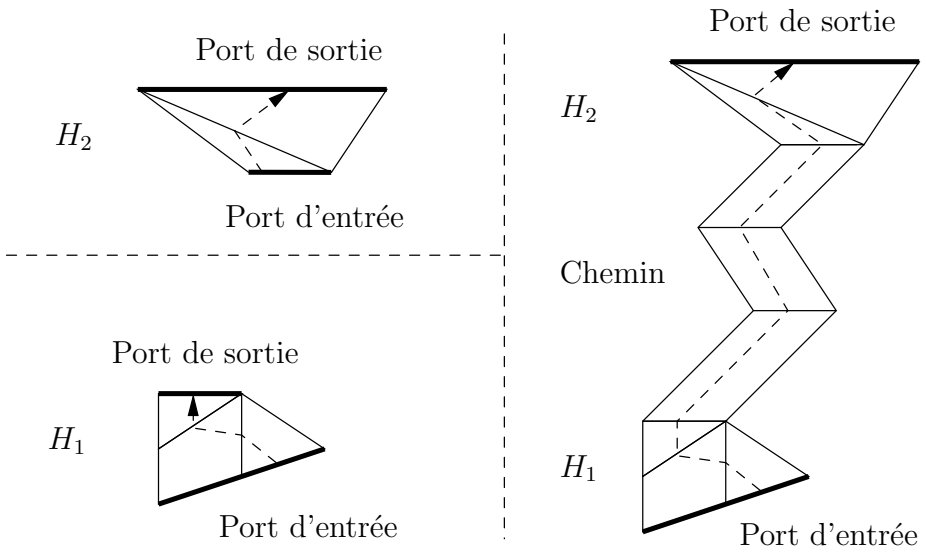


FIG. 7.4 – A partir d'un système DCPM H_1 calculant une fonction g et d'un système DCPM H_2 calculant une fonction g', on peut construire un système DCPM qui calcule $g' \circ g$

DCPM-calculable en dimension d. Soit $g' : [0,1]^q \to [0,1]^r$ *une fonction DCPM-calculable en dimension* d'.

Alors la fonction composée $g' \circ g : [0,1]^p \to [0,1]^r$ *est DCPM-calculable en dimension* $max(d, d')$.

Preuve: Il suffit de construire un système DCPM borné qui contient une copie du système DCPM qui calcule g, une copie du système DCPM qui calcule g', et un chemin qui relie le port de sortie du premier au port d'entrée du second : voir la figure 7.4 ou la figure 7.5. □

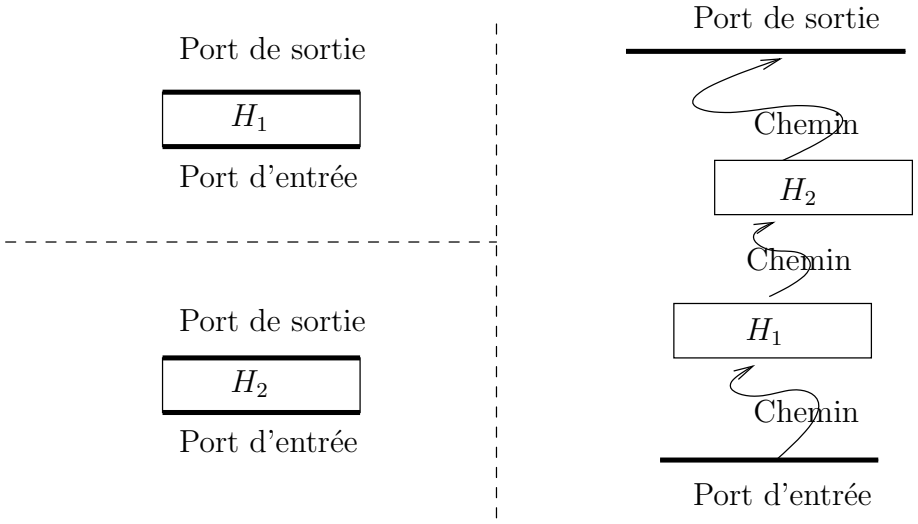


FIG. 7.5 – Représentation symbolique du système DCPM qui calcule la composée de deux fonctions DCPM-calculables

En utilisant le corollaire 7.1, on obtient alors :

Corollaire 7.2 (Réalisation de $x \mapsto \Lambda x + B$ **)** *Soient* $\lambda_1, \ldots, \lambda_p \in \mathbb{Q}^+$, $\beta_1, \ldots, \beta_p \in \mathbb{Q}$ *des constantes.*

La fonction $g : (x_1, \ldots, x_p) \in [0,1]^p \mapsto (\lambda_1 x_1 + \beta_1, \ldots, \lambda_i x_i + \beta_i, \ldots \lambda_p x_p + \beta_p)$ *est DCPM-calculable en dimension* $p + 1$.

De la possibilité de construire des chemins, on peut aussi obtenir aussi la clôture par définition par morceaux des fonctions DCMP-calculables : voir la figure 7.6 ou la représentation symbolique de la figure 7.7.

Lemme 7.4 (Clôture par définition par deux morceaux) *Soit un signe de comparaison* $\# \in \{<, \leq\}$. *Soit* $\beta \in \mathbb{Q}$ *une constante, et soient* $1 \leq i \leq p$ *deux entiers.*

Soient $g_1 : [0,1]^p \to S_1$ *et* $g_2 : [0,1]^p \to S_2$ *deux fonctions DCPM-calculables en dimension d. Soit q la dimension du polyèdre* $S_1 \cap S_2$.

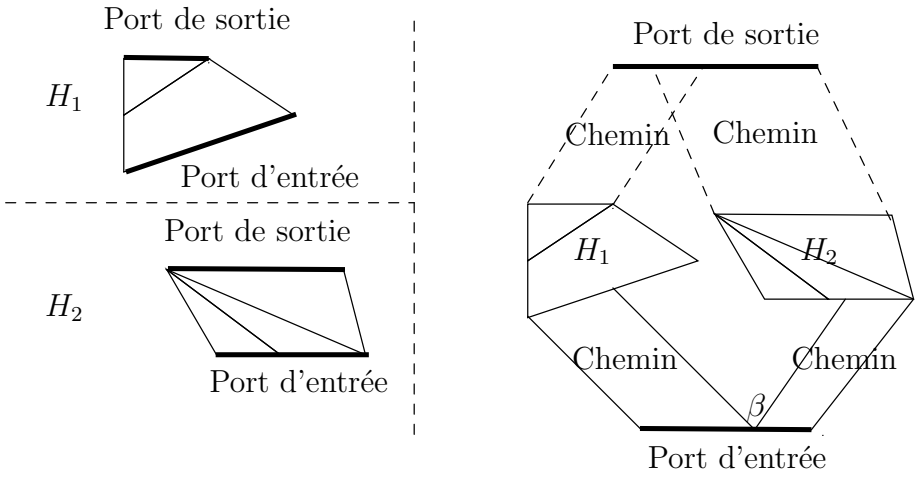


FIG. 7.6 – A partir d'un système DCPM H_1 calculant une fonction g_1 et d'un système DCPM H_2 calculant une fonction g_2, on peut construire un système DCPM qui calcule la fonction égale à g_1 pour $x_1 \leq \beta$ (respectivement $x_1 < \beta$) et égale à g_2 sinon

Alors, la fonction

$$g : x \in [0,1]^p \mapsto \begin{cases} g_1(x) & si\ x_i \# \beta \\ g_2(x) & sinon \end{cases}$$

est DCPM-calculable en dimension $max(d, q+2)$.

Preuve: Nous construisons un système DCPM borné H de dimension $max(d, q+2)$ qui calcule g comme sur les figures 7.6 et 7.7. Formellement, nous obtenons ce système par la suite d'opérations suivante : tout d'abord, nous insérons dans H une copie H_1 du système DCPM qui calcule g_1, une copie H_2 du système DCPM qui calcule g_2, un port d'entrée (P, B) de dimension p, un port de sortie (P', B') de dimension égale au maximum des dimensions de S_1 et de S_2. Appelons P^+ le sous-ensemble des points du polyèdre P dont les coordonnées $(x_1, \ldots, x_p)$ dans la base B vérifient $x_i \# \beta$. Appelons P^- le complémentaire de P^+ dans P. Pour $i \in \{1, 2\}$, désignons par (P_i, B_i) le port de sortie de H_i. Appelons P_i^+ le sous-ensemble des points du polyèdre P_i dont les coordonnées x dans la base B_i vérifient $x \in S_1 \cap S_2$, et appelons P_i^- le complémentaire de P_i^+ dans P_i. De même, appelons $P^{'+}$ le sous-ensemble des points du polyèdre P' dont les coordonnées x dans la base B' vérifient $x \in S_1 \cap S_2$, et appelons $P^{'-}$ le complémentaire de $P^{'+}$ dans P'. Nous ajoutons à H un chemin du port (P^+, B) vers le port d'entrée de H_1, un chemin du port (P^-, B) vers le port d'entrée de H_2, un chemin du port (P_1^-, B_1) vers le port $(P^{'-}, B')$, un chemin du port (P_2^-, B_2) vers le port $(P^{'-}, B')$, un chemin du port (P_1^+, B_1) vers le port $(P^{'+}, B')$, et enfin un chemin du port (P_2^+, B_2) vers le port $(P^{'+}, B')$. □

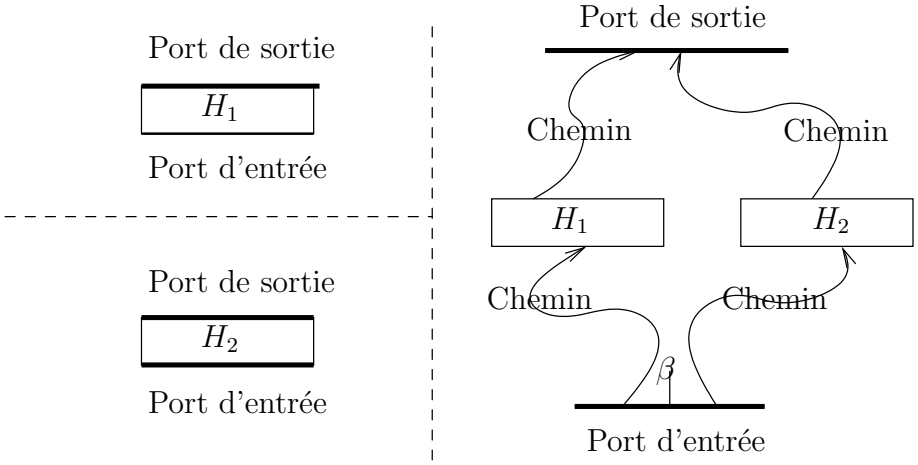


FIG. 7.7 – Représentation symbolique du système DCPM défini par morceaux à partir de deux systèmes DCPM H_1 et H_2

Remarque. La raison qui explique que nous avons supposé la dimension de H supérieure à $q+2$ dans la preuve précédente est la suivante : tous les chemins que nous construisons ont des destinations disjointes deux à deux à l'exception des deux derniers chemins qui ont pour destination commune le polyèdre P'^{+}. Si la dimension de H était inférieure à $q+1$, l'ajout de l'avant dernier chemin du port (P_1^+, B_1) vers le port (P'^{+}, B') aurait rendu impossible l'ajout du dernier chemin du port (P_2^+, B_2) vers le port (P'^{+}, B') puisque le port (P'^{+}, B') n'existerait plus : voir la figure 7.8 à gauche pour $q = 2$. Par contre, en dimension $q+2$ le problème n'apparaît pas : voir la figure 7.8 à droite pour $q = 1$.

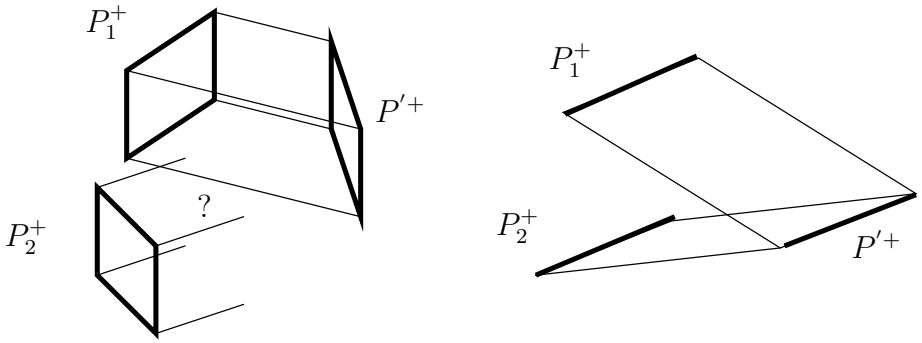


FIG. 7.8 – Illustration de la nécessité d'une dimension supérieure à $q+2$ dans la preuve du lemme 7.4

On obtient alors :

Corollaire 7.3 (Clôture par définition par morceaux) *Soient $1 \leq p < d$ deux entiers. Soient J_1, J_2, ..., J_n des pavés de $[0,1]^p$ disjoints deux à deux. Soient g_1, ..., g_n des fonctions DCPM-calculables en dimension d du type $g_j : [0,1]^p \to S_j$ pour des polyèdres S_j.*

La fonction

$$g : x \in [0,1]^p \mapsto \ g_i(x) \quad si\ x \in J_j$$

définie par morceaux à partir des fonctions g_j est DCPM-calculable en dimension $max(d, q+2)$, où q est le maximum des dimensions des intersections deux à deux des polyèdres S_j.

Preuve: Une telle fonction g s'écrit facilement comme la composition d'un nombre fini d'homothéties de rapports positifs et de fonctions définies à partir des fonctions g_j à l'aide du lemme 7.4. □

Du corollaire 7.1 et de la forme de la fonction g_M donnée par l'observation 7.1, on déduit :

Corollaire 7.4 *La fonction g_M associée à une machine de Turing M par l'observation 7.1 est DCPM-calculable.*

Nous sommes prêts à prouver le théorème 3.2, c'est-à-dire à prouver que toute machine de Turing est simulable par un système DCPM :

Théorème 7.1 ([9]) *Soit M une machine de Turing.*

Il existe un système DCPM H bien défini qui simule M.

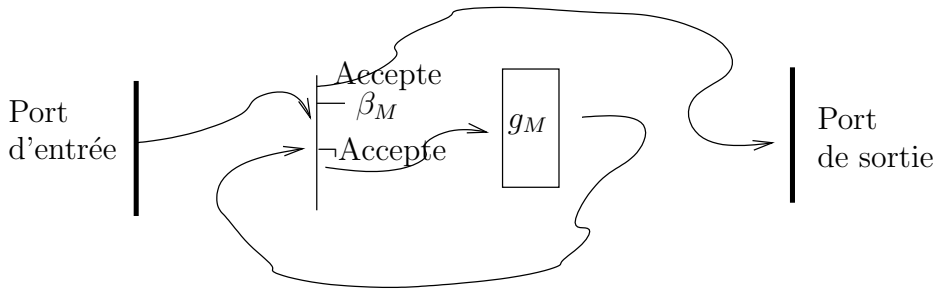


FIG. 7.9 – Représentation symbolique du système DCPM qui simule une machine de Turing M à partir du système DCPM qui calcule sa fonction de transition g_M

Preuve: Il suffit de construire comme sur la figure 7.9 un système DCPM H qui itère la fonction g_M jusqu'à atteindre une configuration acceptante, c'est-à-dire, avec les notations de l'observation 7.1, jusqu'à atteindre un point du sous-ensemble $[0, \beta_M] \times [0,1]$. Formellement, nous obtenons le système DCPM H par la suite d'opérations suivante : tout d'abord, nous insérons dans H une copie H_1

du système DCPM qui calcule g_M et trois ports (P, B), (P', B') et (P'', B'') de dimension 2. Appelons $P'_{Accepte}$ le sous-ensemble de P' constitué des points dont les coordonnées (x_1, x_2) dans la base B' vérifient $x_1 \leq \beta_M$. Notons $P'_{\neg Accepte}$ le complémentaire de $P'_{Accepte}$ dans P. Nous ajoutons un chemin de $(P'_{Accepte}, B')$ vers le port de sortie (P'', B''), un chemin de $(P'_{\neg Accepte}, B')$ vers le port d'entrée de H_1, un chemin du port de sortie de H_1 vers le port (P', B') et un chemin du port d'entrée (P, B) vers le port (P', B'). □

7.2.3 Dimension des systèmes simulant une machine de Turing.

Dans cette section, nous discutons la dimension des systèmes DCPM qui simulent les machines de Turing. Par la remarque page 104, la dimension du système DCPM H construit dans la preuve du théorème 7.1 vaut $max(d, p+2)$ où d est la dimension nécessaire pour calculer g_M et p est la dimension de l'intersection de l'image de g_M avec le port d'entrée. Par le corollaire 7.3, nous avons $d = max(3, q+2)$ où q est la dimension des intersections deux à deux des images par g_M des pavés sur lesquels g_M est affine. Le système DCPM H est donc de dimension $max(3, q+2, p+2)$.

Ceci nous amène à distinguer les cas où les entiers p et q sont inférieurs à 1 du cas général où ils peuvent valoir 2. En revenant à la définition même de la fonction g_M, on est assuré qu'une machine de Turing qui vérifie la définition suivante vérifie $p \leq 1$ et $q \leq 1$.

Définition 7.4 (Machine de Turing régulière [9]) *Une machine de Turing (respectivement avec oracle) M est dite* régulière *si :*

1. *pour toute paire de configurations (w_1, w_2, q), (w'_1, w'_2, q') de M, si l'on note $(\overline{w_1}, \overline{w_2}, \overline{q})$ et $(\overline{w'_1}, \overline{w'_2}, \overline{q'})$ leurs configurations successeurs immédiates respectives, alors $\overline{q} = \overline{q'}$ implique $w_2 = w'_2 = \epsilon$ (ϵ désigne le mot vide).*
2. *Si l'on note q_0 l'état interne initial de la machine, M ne possède aucune configuration (w_1, w_2, q) qui ait pour configuration successeur une configuration (w'_1, w'_2, q') avec $q' = q_0$.*

Récapitulons alors le résultat obtenu :

Théorème 7.2 (Reformulation du théorème 7.1 [9]) *Toute machine de Turing M est simulable par un système DCPM bien défini*
- *de dimension 4 dans le cas général.*
- *de dimension 3 si M est régulière.*

Observons qu'il n'est pas restrictif de supposer les machines de Turing régulières :

Observation 7.2 ([9]) *Toute machine de Turing (respectivement avec oracle) M est simulable par une machine de Turing (respectivement avec oracle) régulière.*

Preuve: Il suffit de considérer une machine de Turing M' qui simule M, mais qui, à chaque fois que M effectue une transition non régulière d'un état interne q vers un état interne q', marque le ruban de façon à mémoriser la position de la tête de lecture, déplace sa tête de lecture vers la gauche tant que possible de façon à atteindre une configuration du type (w, ϵ, q''), $w \in \Sigma^\omega$, $q'' \in Q$, puis transite dans l'état q', et déplace à nouveau sa tête de lecture vers la droite jusqu'à la remettre à la position marquée. □

Corollaire 7.5 *Soit $L \in \Sigma^*$ un langage de Σ_1.*
L est semi-reconnu par un système DCPM de dimension 3.

Preuve: Soit M une machine de Turing régulière à un ruban d'alphabet $\Sigma = \{0, 1\}$ qui semi-reconnaît L. Supposons, sans perte de généralité, que M possède une unique configuration acceptante. Le système DCPM donné par le théorème 7.1 qui simule M semi-reconnaît L : le point d'acceptation est choisi comme le point du port de sortie qui encode la configuration acceptante de M. □

7.3 Ascension de la hiérarchie arithmétique

Dans cette section, nous prouvons que tout langage du second niveau de la hiérarchie arithmétique est semi-reconnu par un système DCPM de dimension 4.

Pour ceci, nous allons utiliser le principe de la construction de Asarin et Maler [8] mais en réduisant[2] la dimension des systèmes utilisés.

Le plan de cette section est le suivant : dans la section 7.3.1, nous exposons le principe de la construction de Asarin et Maler. Dans la section 7.3.2, nous définissons ce que nous appelons un système DCPM homogène. Dans la section 7.3.3, nous prouvons qu'il est possible de construire un système DCPM homogène qui a pour effet de réaliser un "trou d'espace/temps". Dans la section 7.3.4, nous en déduisons que le problème de l'arrêt des machines de Turing est pleinement reconnu par un système DCPM de dimension 4. Dans la section 7.3.5, nous prouvons que tout langage du second niveau de la hiérarchie arithmétique est semi-reconnu par un système DCPM de dimension 4. Enfin, dans la section 7.3.6, nous présentons une légère extension de ce résultat.

[2]La preuve présentée dans [8] prouve que tout langage de Σ_k est reconnu par un système DCPM de dimension $5k + 1$.

7.3.1 Principe de la construction

Dans cette sous-section, nous présentons le principe de la construction que nous allons utiliser.

Le principe se base sur la remarque élémentaire suivante : à partir d'un système DCPM H borné de dimension d, on peut construire un système DCPM $\overline{H}$ borné de dimension $d+1$ qui est construit comme une "pyramide" de H. Nous appelons ce système $\overline{H}$ *l'homogénéisation de* H. Formellement :

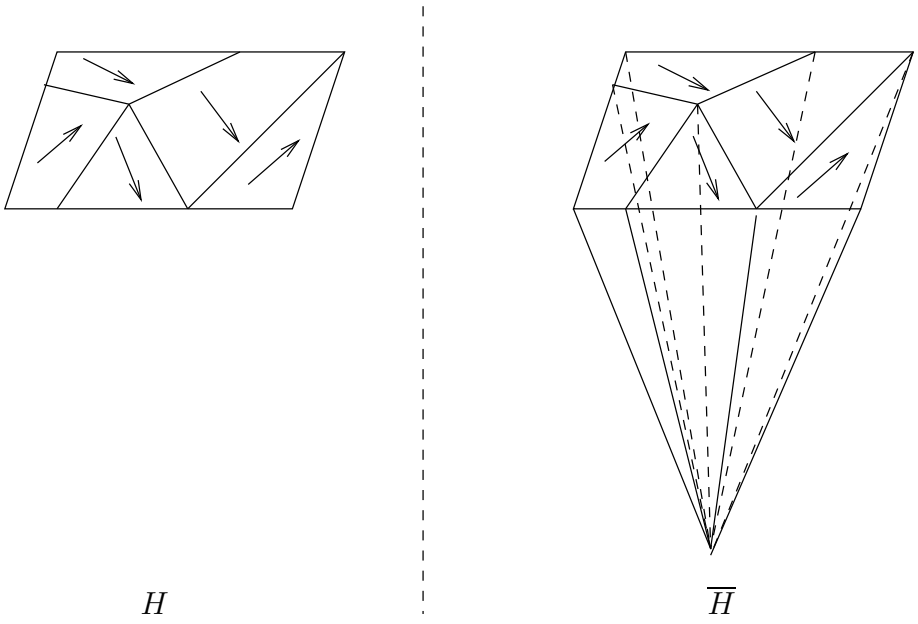


FIG. 7.10 – À partir d'un système DCPM H borné de dimension d, on peut construire un système DCPM $\overline{H}$ borné de dimension $d+1$ construit comme une "pyramide" de H

Définition 7.5 (Homogénéisation d'un système DCPM) *Soit* $H = (X, f)$ *un système DCPM borné de dimension* d.

L' homogénéisation de H, *notée* $\overline{H}$, *est le système DCPM* $(\overline{X}, \overline{f})$ *borné de dimension* $d+1$ *défini par (voir la figure 7.10) :*

- $\overline{X} = \{(x_1, \ldots, x_{d+1}) | 0 < x_{d+1} \leq 1$ *et* $(x_1/x_{d+1}, \ldots, x_d/x_{d+1}) \in X\}$
- $f'(x_1, \ldots, x_{d+1}) = (f(x_1/x_{d+1}, \ldots, x_d/x_{d+1}), 0)$

Considérons maintenant l'homogénéisation $\overline{H}$ d'un système H de dimension d. Considérons une section de $\overline{H}$ par l'hyperplan $x_{d+1} = z$, pour $0 < z \leq 1$: voir la figure 7.11. On obtient un système DCPM H_z de dimension d qui possède la propriété remarquable suivante : les régions de ce système sont obtenues en appliquant une homothétie de rapport z aux régions de H, mais les vecteurs

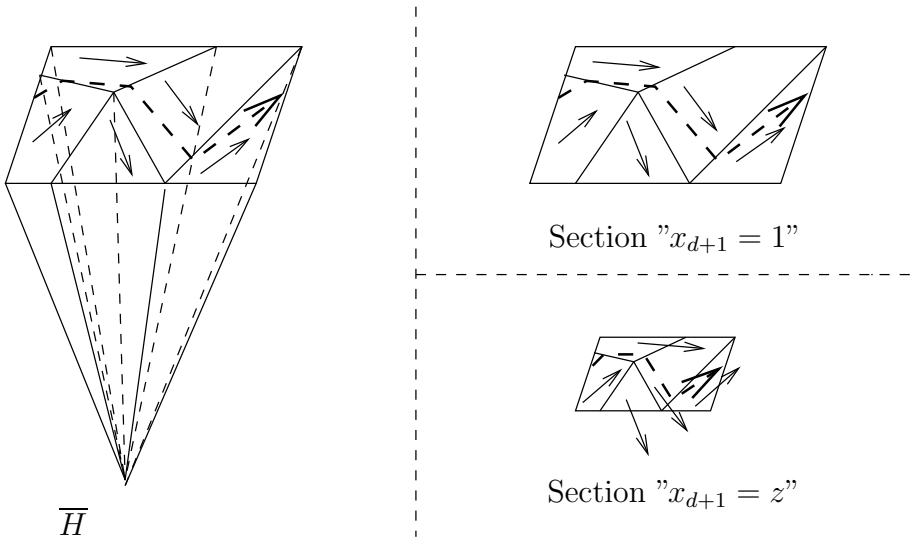


FIG. 7.11 – Homogénéisation $\overline{H}$ d'un système DCPM H et ses sections par les hyperplans d'équations $x_d = 1$ et $x_d = z$ pour $0 < z \leq 1$

pentes de ce système sont donnés par les vecteurs pentes de H. En d'autres termes, ce système est obtenu en contractant l'espace, mais sans contracter les vitesses : voir la figure 7.11.

Considérons maintenant une trajectoire ϕ du système DCPM H qui part d'un port d'entrée (P, B) et qui atteint un port de sortie (P', B'). L'homothétique $\overline{\phi}$ de rapport z de la trajectoire ϕ est une trajectoire de H_z qui part de l'homothétique du port d'entrée (P, B) vers l'homothétique du port de sortie (P', B'). $\overline{\phi}$ réalise les mêmes "opérations" que ϕ puisque $\overline{\phi}$ traverse les (homothétiques des) mêmes régions que ϕ.

Toutefois, supposons que le calcul de ϕ prend un temps t. Puisque l'espace a été dilaté d'un rapport z sans changer les vitesses, $\overline{\phi}$ prend un temps égal au produit de t par z. En d'autres termes, $\overline{\phi}$ réalise les mêmes opérations que ϕ mais $1/z$ fois plus vite !

Formellement :

Observation 7.3 *Soit H un système DCPM de dimension d qui calcule en temps t une fonction $g : [0,1]^p \to [0,1]^q$.*

Alors $\overline{H}$ calcule la fonction partielle $\overline{g} : [0,1]^{p+1} \to [0,1]^{q+1}$ définie pour $0 < x_{p+1} \leq 1$ par

$$\overline{g}(x_1, \ldots, x_{p+1}) = (g(x_1/x_{p+1}, \ldots, x_p/x_{p+1})/x_{p+1}, x_{p+1})$$

en temps t/x_{p+1}.

Considérons maintenant l'effet d'une division par deux de toutes les variables. Cette opération a pour conséquence d'accélérer virtuellement le temps d'un facteur 2 puisqu'un calcul de ϕ ou de $\overline{\phi}$ qui prenait un temps t prend maintenant un temps $t/2$. Appelons " *passer à travers un trou d'espace/temps*" le fait de diviser toutes les variables par deux.

Il est alors possible de simuler un nombre infini d'opérations en temps fini : en effet, il suffit par exemple d'effectuer la première opération, de passer à travers un trou d'espace/temps, d'effectuer la seconde opération, de passer à travers un trou d'espace/temps, et ainsi de suite. On a alors simulé un nombre infini d'opérations en le temps fini $2 = \sum_{j=0}^{\infty} 1/2^j$ en utilisant le *paradoxe de Zénon*, c'est-à-dire en utilisant le fait que l'on peut effectuer une infinité d'opérations discrètes en un temps fini.

C'est ce principe que nous allons utiliser pour reconnaître pleinement des langages non-récursifs à l'aide de systèmes DCPM.

7.3.2 Terminologie

Avant de prouver formellement que tout langage du second niveau de la hiérarchie arithmétique est semi-reconnu par un système DCPM de dimension 4, nous introduisons quelques définitions. Nous dirons qu'un polyèdre est homogène s'il vérifie :

Définition 7.6 (Polyèdre homogène) *Un polyèdre $\overline{P} \subset \mathbb{R}^{d+1}$ est dit* homogène *s'il existe un polyèdre P inclus dans l'hyperplan d'équation $x_{d+1} = 1$ tel que*

$$\overline{P} = \{(x_1, \ldots, x_{d+1}) | 0 < x_{d+1} \leq 1 \text{ et } (x_1/x_{d+1}, \ldots, x_d/x_{d+1}) \in P\}$$

Nous dirons que le polyèdre $\overline{P}$ est l' homogénéisation du polyèdre P. *Nous appellerons P* la projection de $\overline{P}$.

Un système DCPM sera alors dit *homogène* s'il vérifie :

Définition 7.7 (Système DCPM homogène) *Un système DCPM $\overline{H}$ est dit* homogène *si chacune de ses régions est homogène.*

Remarque. L'homogénéisation d'un système DCPM est toujours un système DCPM homogène. Toutefois un système DCPM homogène n'est pas nécessairement l'homogénéisation d'un système DCPM. En effet, dans un système DCPM homogène, il est possible que les vecteurs pentes ne soient pas parallèles à l'hyperplan d'équation $x_{d+1} = 1$. Le système DCPM de la figure 7.12 en est une illustration.

De même, nous dirons qu'un port $(\overline{P}, \overline{B})$ est *homogène* s'il peut s'écrire comme l'homogénéisation d'un port d'un système DCPM. Formellement :

Définition 7.8 *Un port $(\overline{P}, \overline{B})$ de dimension $p+1$ de H est dit* homogène *si d'une part le polyèdre $\overline{P}$ est homogène et si d'autre part la base $\overline{B} = (b_1, b_2, \ldots, b_{p+1})$ est telle que $B = (b_1, b_2, \ldots, b_p)$ soit une base affine de la projection P de $\overline{P}$.*

Le port (P, B) est appelé la projection du port $(\overline{P}, \overline{B})$.

7.3.3 Réalisation d'un trou d'espace/temps

Nous prouvons maintenant qu'il est possible de réaliser un "*trou d'espace/temps*" : c'est-à-dire un système homogène qui a pour effet de diviser toutes les variables par deux.

La première étape pour réaliser un "trou d'espace/temps" est de savoir construire un système homogène qui divise la variable x_{d+1} par deux :

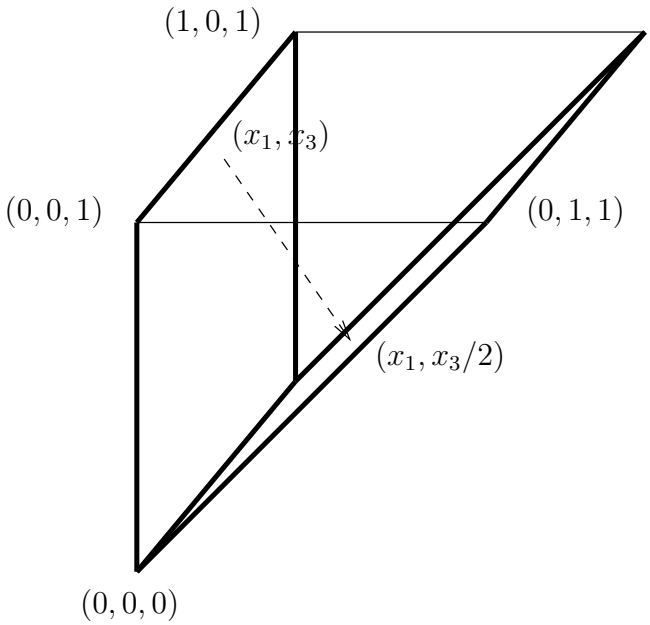


FIG. 7.12 – Un système DCPM homogène de dimension 2 qui calcule la fonction $g : (x_1, x_3) \mapsto (x_1, x_3/2)$ en temps x_3

Observation 7.4 (Réalisation de $x_{d+1} \leftarrow x_{d+1}/2$) *Soit d un entier.*

Il existe un système DCPM homogène borné de dimension $d+1$ qui calcule la fonction $g : [0,1]^d \to [0,1]^d$ définie par

$$g : (x_1, \ldots, x_{d-1}, x_{d+1}) \mapsto (x_1, \ldots, x_{d-1}, x_{d+1}/2)$$

en temps x_{d+1}.

Preuve: Le système DCPM $H = (X, f)$ défini par

$$X = \{(x_1, \ldots, x_{d+1}) | 0 < x_{d+1} \leq 1, 0 \leq x_i/x_{d+1} \leq 1 \text{ pour } i = 1, 2, \ldots, d\}$$

et $f(x) = (0, \ldots, 0, 1, -1/2)$ pour tout $x \in X$ est solution : voir la figure 7.12. □

Nous appellerons *chemin homogène* un chemin obtenu de la façon suivante :

Définition 7.9 (Chemin homogène) *Soit $\overline{H}$ un système DCPM homogène de dimension $d+1$. Soient $(\overline{P_1}, \overline{B_1})$ et $(\overline{P_2}, \overline{B_2})$ deux ports homogènes de H de dimension $p+1$.*

Nous appelons chemin homogène *entre $(\overline{P_1}, \overline{B_1})$ et $(\overline{P_2}, \overline{B_2})$ l'homogénéisation d'un chemin dans l'hyperplan d'équation $x_{d+1} = 1$ entre la projection (P_1, B_1) de $(\overline{P_1}, \overline{B_1})$ et la projection (P_2, B_2) de $(\overline{P_2}, \overline{B_2})$.*

L'intérêt d'un tel chemin est donné par la remarque suivante qui découle immédiatement de l'observation 7.3 :

Observation 7.5 *Un chemin homogène entre deux ports homogènes $(\overline{P_1}, \overline{B_1})$ et $(\overline{P_2}, \overline{B_2})$ de dimension $p+1$ d'un système DCPM homogène de dimension $d+1$ est un chemin tel qu'il existe une constante $k > 0$ telle que toute trajectoire partant du point du polyèdre $\overline{P_1}$ de coordonnées $(x_1, \ldots, x_{p+1})$ dans la base $\overline{B_1}$ atteint le point du polyèdre $\overline{P_2}$ de coordonnées $(x_1, \ldots, x_{p+1})$ dans la base $\overline{B_2}$ à un temps inférieur à kx_{p+1}.*

Nous pouvons alors prouver qu'il est possible de construire un système DCPM qui réalise un "trou d'espace/temps" : voir la figure **??**.

fig :trouRéalisation d'un trou d'espace/temps

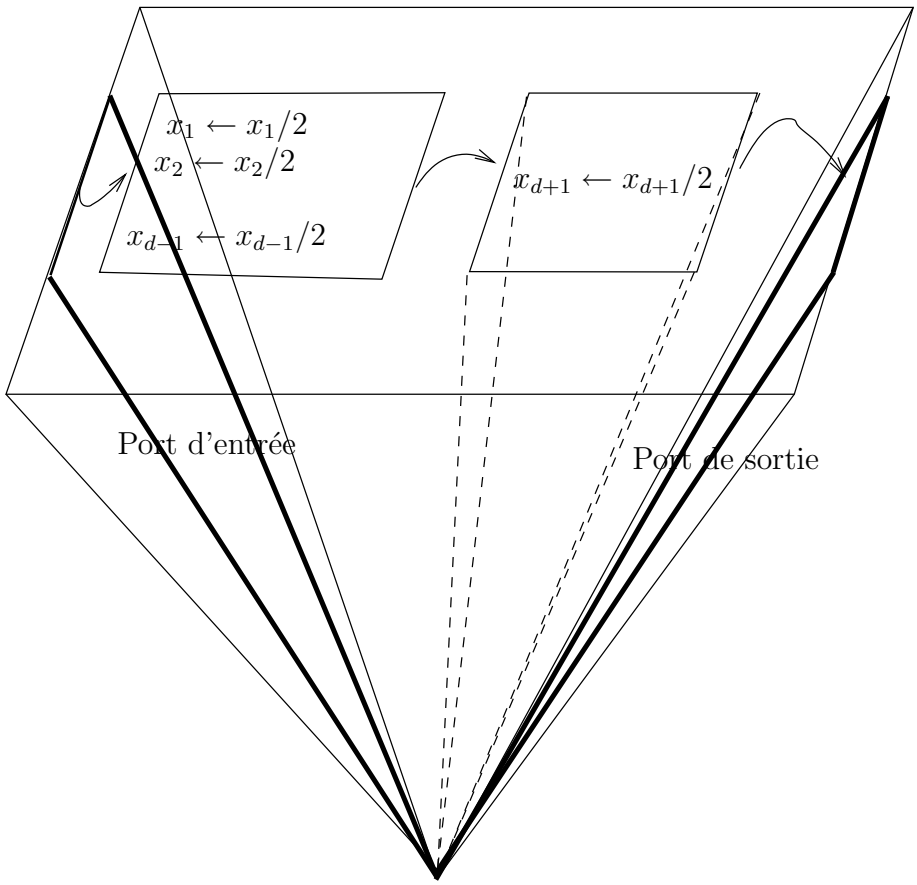
Lemme 7.5 (Réalisation d'un trou d'espace/temps) *Soit d un entier.*

Il existe un système DCPM homogène borné $\overline{H} = (\overline{X}, \overline{f})$ de dimension $d+1$ qui calcule en temps proportionnel à x_{d+1} la fonction partielle $g : [0,1]^d \to [0,1]^d$ définie par

$$g : (x_1, \ldots, x_{d-1}, x_{d+1}) \mapsto (x_1/2, \ldots, x_{d-1}/2, x_{d+1}/2)$$

pour $0 < x_{d+1} \leq 1$, $0 \leq x_i/x_{d+1} \leq 1$.

Preuve: Nous savons que la fonction $g_1 : [0,1]^{d-1} \to [0,1]^{d-1}$ qui à $(x_1, \ldots, x_{d-1})$ associe $(x_1/2, \ldots, x_{d-1}/2)$ est DCPM-calculable en dimension d : voir le corollaire 7.1. Soient H_1 le système DCPM de dimension d qui la calcule et $\overline{H_1}$ l'homogénéisation de H_1. Soit $\overline{H_2}$ le système DCPM de dimension $d+1$ de l'observation 7.4 qui calcule $g_1 : (x_1, \ldots, x_{d-1}, x_{d+1}) \mapsto (x_1, \ldots, x_{d-1}, x_{d+1}/2)$. Il suffit de composer $\overline{H_1}$ avec $\overline{H_2}$ comme sur la figure **??** pour obtenir un système DCPM H de dimension $d+1$ qui vérifie les assertions du lemme. Formellement, nous obtenons H par la suite d'opérations suivante : nous insérons dans H une copie du système


$x_1 \leftarrow x_1/2$
$x_2 \leftarrow x_2/2$
$x_{d-1} \leftarrow x_{d-1}/2$
$x_{d+1} \leftarrow x_{d+1}/2$
Port d'entrée
Port de sortie

DCPM $\overline{H_1}$, une copie du système DCPM $\overline{H_2}$, un port d'entrée homogène $(\overline{P}, \overline{B})$ de dimension d, un port de sortie homogène $(\overline{P'}, \overline{B'})$ de dimension d, un chemin homogène entre $(\overline{P}, \overline{B})$ et le port d'entrée de $\overline{H_1}$, un chemin homogène entre le port de sortie de $\overline{H_1}$ et le port d'entrée de $\overline{H_2}$ et un chemin entre le port de sortie de $\overline{H_2}$ et le port de sortie $(\overline{P'}, \overline{B'})$: voir la figure **??**. □

Il est bien entendu aussi possible de réaliser un " *trou d'espace/temps inverse*" c'est à dire un système qui multiplie toutes les variables par deux :

Lemme 7.6 (Réalisation d'un trou d'espace/temps inverse) *Soit d un entier.*

Il existe un système DCPM homogène borné $\overline{H} = (\overline{X}, \overline{f})$ de dimension $d+1$ qui calcule en temps proportionnel à x_{d+1} la fonction partielle $g : [0,1]^d \to [0,1]^d$ définie par

$$g : (x_1, \ldots, x_{d-1}, x_{d+1}) \mapsto (2x_1, \ldots, 2x_{d-1}, 2x_{d+1})$$

pour $0 < x_{d+1} \leq 1/2$, $0 \leq x_i/x_{d+1} \leq 1$.

Preuve: La preuve est similaire à celle du lemme précédent à la différence près que les chemins homogènes du port de sortie de $\overline{H_1}$ vers le port d'entrée de $\overline{H_2}$ et du port de sortie de $\overline{H_2}$ vers le port $(\overline{P'}, \overline{B'})$ sont remplacés par des chemins homogènes du port de sortie de $\overline{H_1}$ vers le port de sortie de $\overline{H_2}$ et du port d'entrée de $\overline{H_2}$ vers le port $(\overline{P'}, \overline{B'})$. □

7.3.4 Reconnaissance du problème de l'arrêt en dimension 4

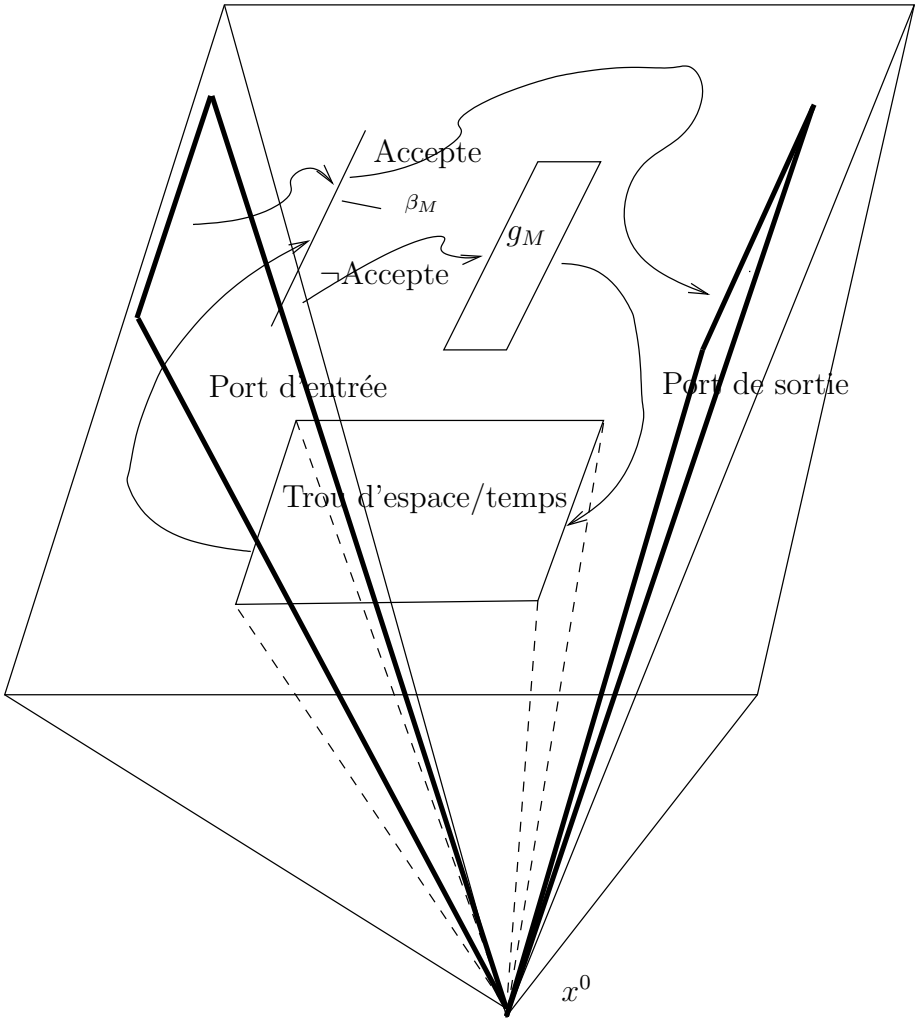
Nous allons maintenant utiliser les observations précédentes pour construire un système DCPM de dimension 4 qui reconnaît pleinement le problème de l'arrêt des machines de Turing en temps fini.

Le principe est le suivant : le problème de l'arrêt des machines de Turing est un langage semi-reconnu par une machine de Turing M. Pour reconnaître pleinement L, nous allons modifier le système DCPM du théorème 7.1 qui simule M de façon à introduire le "trou d'espace/temps" de la section précédente pour accélérer les calculs.

fig :whileborneSimulation d'une machine de Turing en temps fini

Formellement, nous prouvons le résultat suivant : rappelons que la fonction Ξ est la fonction de codage de l'ensemble des mots finis ou infinis sur l'alphabet $\Sigma = \{0,1\}$ vers l'ensemble $\mathbb{R}$ des réels définie dans le chapitre 2.

Lemme 7.7 *Soit M une machine de Turing régulière à un ruban d'alphabet $\Sigma = \{0,1\}$.*


Accepte
β_M
g_M
¬Accepte
Port d'entrée
Port de sortie
Trou d'espace/temps
x^0

Il existe un système DCPM homogène $\overline{H}$ *de dimension* 4 *qui calcule une fonction partielle* $g : [0,1]^2 \to [0,1]^2$ *telle que*

$$g : (\frac{\Xi(w)}{x_{d+1}}, x_{d+1}) \mapsto \begin{cases} (0,0) & \text{si } w \text{ n'est pas accepté par } M \\ (\frac{\Xi(w')}{2^n x_{d+1}}, \frac{x_{d+1}}{2^n}) & \text{si } M \text{ accepte } w \text{ au temps } n \in \mathbb{N} \\ & \text{avec } w' \in \Sigma^\omega \text{ comme ruban} \end{cases}$$

pour tout mot $w \in \Sigma^\omega$ *et pour tout* $0 < x_{d+1} \leq 1$.

Preuve: Il suffit de considérer le système DCPM $\overline{H}$ représenté sur la figure **??** (comparez cette figure à la figure 7.9). Formellement : soit g_M la fonction DCPM-calculable associée à M par l'observation 7.1. Nous construisons $\overline{H}$ par la suite d'opérations suivante : tout d'abord, nous insérons dans $\overline{H}$ l'homogénéisation $\overline{H_1}$ du système DCPM qui calcule g_M. Nous insérons aussi le système DCPM $\overline{H_2}$ du lemme 7.5 qui réalise un "trou d'espace/temps". Nous ajoutons un port homogène $(\overline{P}, \overline{B})$ de dimension 2, et deux ports homogènes $(\overline{P'}, \overline{B'})$, $(\overline{P''}, \overline{B''})$ de dimension 3. Appelons $\overline{P_{Accepte}}$ le sous-ensemble de $\overline{P'}$ constitué des points dont les coordonnées (x_1, x_2, x_3) dans la base $\overline{B'}$ vérifient $x_1/x_3 \leq \beta_M$. Notons $\overline{P'_{\neg Accepte}}$ le complémentaire de $\overline{P_{Accepte}}$ dans $\overline{P'}$. Nous ajoutons un chemin homogène de $(\overline{P'_{Accepte}}, \overline{B'})$ vers le port de sortie $(\overline{P''}, \overline{B''})$, un chemin homogène de $(\overline{P'_{\neg Accepte}}, \overline{B'})$ vers le port d'entrée de $\overline{H_1}$, un chemin homogène du port de sortie de $\overline{H_1}$ vers le port d'entrée de $\overline{H_2}$, un chemin homogène du port de sortie de $\overline{H_2}$ vers $(\overline{P'}, \overline{B'})$ et enfin un chemin homogène du port d'entrée $(\overline{P}, \overline{B})$ vers le port $(\overline{P}', \overline{B}')$: voir la figure **??**.

Discutons maintenant les propriétés du système $\overline{H}$ que nous venons de construire : soient $w \in \Sigma^\omega$ un mot et $0 < x_{d+1} \leq 1$ un réel. Si le mot w est accepté par la machine M au temps n, la trajectoire de H partant de $(\frac{\Xi(w)}{x_{d+1}}, x_{d+1})$ simulera le calcul de M sur w, traversera n fois le système DCPM $\overline{H_2}$ et atteindra ultimement le port de sortie (P'', B'') au point de coordonnées $(\frac{\Xi(w')}{2^n x_{d+1}}, \frac{x_{d+1}}{2^n})$ dans la base B''.

Maintenant si w n'appartient pas à L, M n'acceptera jamais w. Par conséquent, le calcul de H correspondant à w traversa une infinité de fois le système DCPM $\overline{H_2}$. Puisque chaque traversée de ce système à pour effet de diviser toutes les variables par deux le calcul convergera vers le point $x^0 = (0, \ldots, 0)$. Or chaque traversée du système $\overline{H_2}$ a aussi pour effet de réduire d'un facteur deux le temps nécessaire pour calculer l'itération suivante de la fonction g_M. Si l'on note t_0 le temps nécessaire lorsque x_{d+1} vaut 1, la trajectoire sera en $x^0 = (0, \ldots, 0)$ au temps $t_0(1 + 1/2 + \ldots + 1/2^k + \ldots) = 2t_0$, c'est-à-dire à un temps fini. □

Nous pouvons alors prouver que le problème de l'arrêt des machines de Turing est pleinement reconnu par un système DCPM de dimension 4 :

Théorème 7.3 *Soit* L *le problème de l'arrêt des machines de Turing ou tout autre langage récursivement énumérable.*

Alors L *est pleinement reconnu par un système DCPM bien défini de dimension* 4.

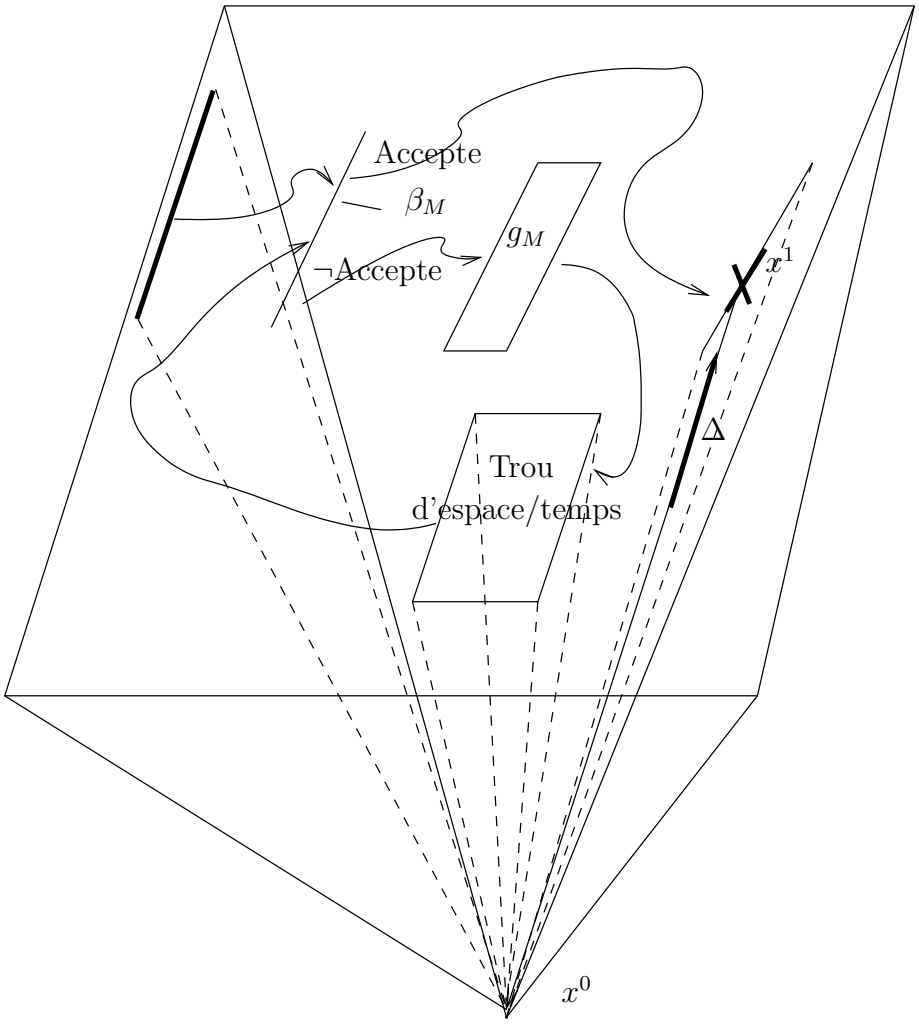


FIG. 7.13 – Modification du système sur le segment Δ de façon à reconnaître le problème de l'arrêt des machines de Turing

Preuve: L est semi-reconnu par une machine de Turing M que nous pouvons supposer régulière et avec une unique configuration acceptante. Considérons le système $\overline{H}$ construit dans le lemme précédent pour cette machine de Turing M. Soit x^0 le point de coordonnées $(0, \ldots, 0)$. Soit x^1 le point de la projection du polyèdre $\overline{P''}$ qui encode l'unique configuration acceptante de M. L'homogénéisation de la région réduite au point x_1 est un segment Δ. Nous modifions le vecteur pente des trajectoires de $\overline{H}$ sur Δ : sur le segment Δ ouvert en x^0, nous fixons le vecteur pente au vecteur $\vec{x^0x^1}$: voir la figure 7.13. Le système DCPM obtenu reconnaît pleinement le langage L avec la projection du port $(\overline{P}, \overline{B})$ comme port d'encodage et les points x^0 et x^1 comme points d'acceptation et de rejet. □

7.3.5 Ascension d'un niveau

Nous allons maintenant prouver que tous les langages du second niveau de la hiérarchie arithmétique sont semi-reconnus par un système DCPM de dimension 4.

Pour ceci, nous allons utiliser la construction présentée dans le lemme 7.7 ou sur la figure **??**. Toutefois, la construction présentée dans ce lemme ou sur cette figure offre l'inconvénient de retourner un résultat qui dépend du nombre n de passages à travers le "trou d'espace/temps" du système. Nous allons commencer par prouver que l'on peut se passer de cette dépendance en annulant l'effet des passages à travers les trous d'espace/temps : observez la figure **??**.

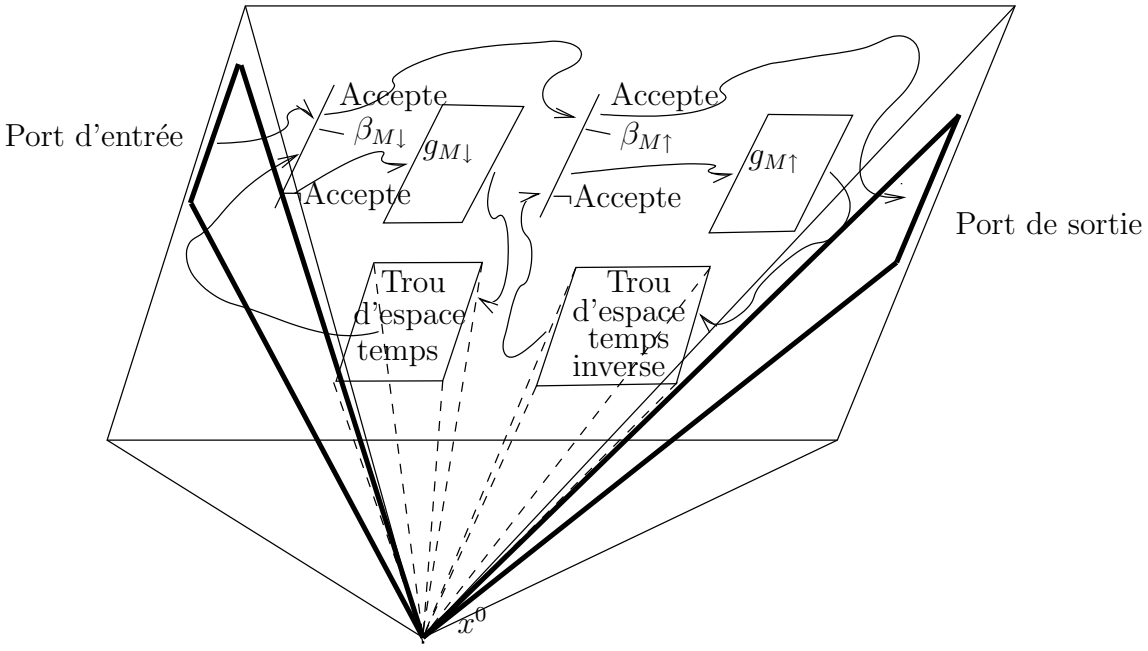


fig :whilebornecstLe système DCPM du lemme 7.8

Nous nous basons pour cela sur l'observation suivante :

Observation 7.6 *Soit M une machine de Turing (respectivement avec oracle $X \subset \Sigma^*$).*

Il existe deux machines de Turing $M\uparrow$ et $M\downarrow$ (respectivement avec oracle X) régulières à un ruban d'alphabet $\Sigma = \{0,1\}$ telles que, pour tout mot $w \in \Sigma^\omega$ accepté par M, si l'on note $w' \in \Sigma^\omega$ le ruban avec lequel M accepte,

- *$M\downarrow$ accepte w avec w comme ruban*
- *$M\uparrow$ accepte w avec w' comme ruban*
- *$M\downarrow$ et $M\uparrow$ acceptent w en exactement le même temps.*

Preuve: Soit $w \in \Sigma^\omega$ un mot accepté par M. M accepte ce mot avec un certain mot $w' \in \Sigma^\omega$ comme ruban. Puisqu'en temps fini M ne peut modifier qu'un nombre fini et calculable de cases de son ruban, il existe des préfixes calculables $w_1 \in \Sigma^*$ et $w'_1 \in \Sigma^*$ de w et de w' respectivement, tels que w_1 et w'_1 soient de même longueur, et tels que w et w' s'écrive $w = w_1 w_2$ et $w' = w'_1 w_2$ pour un même suffixe $w_2 \in \Sigma^\omega$.

Soit M' une machine de Turing (respectivement avec oracle X) régulière à un ruban qui accepte tout mot $w \in \Sigma^\omega$ accepté par M mais avec $w_1\#w'_1\#w_2$ comme ruban : $\#$ désigne une lettre de l'alphabet utilisée comme délimiteur.

Il suffit de construire $M\uparrow$ comme la machine M' à laquelle sont ajoutées des instructions qui transforment le ruban $w_1\#w'_1\#w_2$ en $w'_1 w_2$ de la façon suivante : le préfixe $w_1\#w'_1\#$ est balayé successivement de gauche à droite et de droite à gauche. Chaque balayage supprime le caractère le plus à droite de w_1. Lorsqu'il ne reste plus de caractère dans w_1, un dernier balayage supprime les deux symboles $\#$ restants.

Il suffit de construire $M\downarrow$ comme la machine M' à laquelle sont ajoutées des instructions qui transforme le ruban $w_1\#w'_1\#w_2$ en $w_1 w_2$ de la façon suivante : le préfixe $w_1\#w'_1\#$ est balayé successivement de gauche à droite et de droite à gauche. Chaque balayage supprime le caractère le plus à droite de w'_1. Lorsqu'il ne reste plus de caractère dans w'_1, un dernier balayage supprime les deux symboles $\#$ restants. □

Nous en déduisons : voir la figure **??**.

Lemme 7.8 *Soit M une machine de Turing régulière à un ruban d'alphabet $\Sigma = \{0,1\}$.*

Il existe un système DCPM homogène $\overline{H}$ de dimension 4 qui calcule une fonction partielle $g : [0,1]^2 \to [0,1]^2$ telle que

$$g : (\frac{\Xi(w)}{x_{d+1}}, x_{d+1}) \mapsto \begin{cases} (0,0) & \text{si } w \text{ n'est pas accepté par } M \\ (\frac{\Xi(w')}{x_{d+1}}, x_{d+1}) & \text{si } M \text{ accepte } w \text{ avec } w' \in \Sigma^\omega \text{ comme ruban} \end{cases}$$

pour tout mot $w \in \Sigma^\omega$ et pour tout $0 < x_{d+1} \leq 1$.

Preuve: Notons $M\uparrow$ et $M\downarrow$ les machines de Turing régulières associées à M par l'observation précédente. Soit $\overline{H_1}$ le système obtenu en appliquant le lemme

7.7 à la machine $M \downarrow$. Soit $\overline{H_2}$ le système obtenu en appliquant le lemme 7.7 à la machine $M \uparrow$ mais en remplaçant le trou d'espace/temps par un trou d'espace/temps inverse. Il suffit de relier ces deux systèmes par un chemin homogène pour obtenir un système DCPM solution : voir la figure **??**. □

Nous sommes alors prêts à prouver que tous les langages de Σ_2 sont semi-reconnus par un système DCPM de dimension 4 :

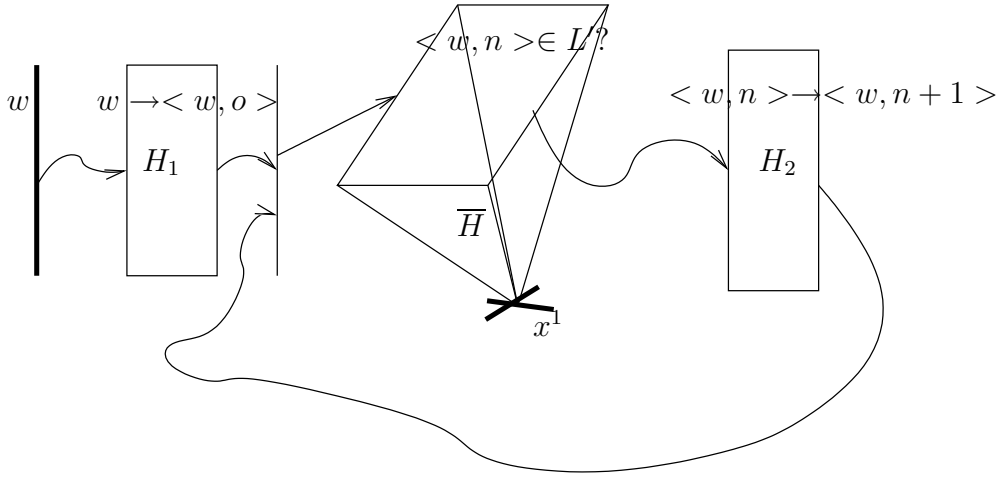


fig :sigmadeuxSystème DCPM qui semi-reconnaît un langage de Σ_2

Théorème 7.4 (Semi-reconnaissance de Σ_2) *Tout langage L de Σ_2 est semi-reconnu par un système DCPM bien défini de dimension* 4.

Preuve: Soit L un langage deΣ_2. Notons par $<,>$ une fonction récursive bijective de $\Sigma^* \times \mathbb{N} \rightarrow \Sigma^*$. Il existe un langage récursivement énumérable L' tel que L s'écrive $L = \{w \in \Sigma^* | \exists n \in \mathbb{N} \ < w, n > \in L'\}$: voir [64] ou [68]. Soit M une machine de Turing qui semi-reconnaît L'. Nous pouvons supposer sans perte de généralité que M accepte en retournant son entrée. Soit $\overline{H}$ le système DCPM qui lui correspond par le lemme 7.8. Soit H_1 le système DCPM de dimension 3 qui simule la machine de Turing qui transforme un mot $w \in \Sigma^*$ en $< w, 0 >$, et soit H_2 le système DCPM de dimension 3 qui simule la machine de Turing qui transforme $< w, n >$ en $< w, n+1 >$ pour tout $n \in \mathbb{N}$. L est semi-reconnu par le système DCPM H de dimension 4 représenté sur la figure **??**. Formellement H est obtenu en prenant une copie des systèmes $\overline{H}$, H_1 et H_2, en insérant deux ports (P, B) et (P', B') de dimension 1, en ajoutant un chemin du port (P, B) vers le port d'entrée de H_1, un chemin du port de sortie de H_1 vers (P', B'), un chemin de (P', B') vers la projection du port d'entrée de $\overline{H}$, un chemin de la projection du port de sortie de $\overline{H}$ vers le port d'entrée de H_2 et enfin un chemin du port de sortie de H_2 vers (P', B'). Le point d'acceptation x^1 du système est le point de coordonnées $(0, 0)$ dans les bases associées aux ports homogènes de $\overline{H}$: voir la figure **??**. □

7.3.6 Généralisation aux dimensions supérieures

Dans cette sous-section nous présentons une légère extension du théorème 7.4. Nous introduisons pour ceci la terminologie suivante : nous dirons qu'une machine de Turing avec oracle est *DCPM-simulable en dimension d* s'il existe l'équivalent de l'observation 7.1 pour cette machine. Formellement :

Définition 7.10 (Machine DCPM-simulable) *Soit M une machine de Turing à un ruban avec oracle $X \subset \Sigma^*$. Soit $C = \Sigma^\omega \times \Sigma^\omega \times Q$ l'espace de ses configurations.*

M est dite DCPM-simulable en dimension *d s'il existe*

- *une fonction $g_M : [0,1]^2 \to [0,1]^2$ DCPM-calculable en dimension d*
- *une fonction d'encodage $\nu : C \to [0,1]^2$ telle que :*
  1. *$\nu(\Sigma^* \times \Sigma^* \times Q) \subset \mathbb{Q}^2$*
  2. *il existe une constante $\beta_M \in \mathbb{Q}$ telle que $\nu(w_1, w_2, q) \in [0, \beta_M] \times [0,1]$ si et seulement si q est un état interne acceptant de M.*

telles que le diagramme suivant commute :

$$\begin{array}{ccc} C & \xrightarrow{\vdash} & C \\ {\scriptstyle\nu}\downarrow & & \downarrow{\scriptstyle\nu} \\ [0,1]^2 & \xrightarrow{g_M} & [0,1]^2 \end{array}$$

(c'est-à-dire telles que $g_M(\nu(c)) = \nu(c')$ pour toutes configurations $c, c' \in C$ avec $c \vdash c'$).

Le lemme 7.8 se généralise alors en :

Lemme 7.9 *Soit $X \subset \Sigma^*$ un langage. Soit $d \geq 3$ un entier tel que toute machine de Turing avec oracle X soit DCPM-simulable en dimension d. Soit M une machine de Turing avec oracle X régulière à un ruban d'alphabet $\Sigma = \{0,1\}$.*

Il existe un système DCPM homogène $\overline{H}$ de dimension $d+1$ qui calcule une fonction partielle $g : [0,1]^2 \to [0,1]^2$ telle que

$$g : (\frac{\Xi(w)}{x_{d+1}}, x_{d+1}) \mapsto \begin{cases} (0,0) & \text{si } w \text{ n'est pas accepté par } M \\ (\frac{\Xi(w')}{x_{d+1}}, x_{d+1}) & \text{si } M \text{ accepte } w \text{ avec } w' \in \Sigma^\omega \text{ comme ruban} \end{cases}$$

pour tout mot $w \in \Sigma^\omega$ et pour tout $0 < x_{d+1} \leq 1$.

Preuve: La preuve est identique à celle du lemme 7.8 si ce n'est que les fonctions $g_{M\downarrow}$ et $g_{M\uparrow}$ ne sont pas données directement par l'observation 7.1 mais en utilisant la définition précédente.

Toutefois, pour palier au fait que la définition précédente ne précise pas que le temps de calcul d'une fonction g_M est indépendant de la machine M et donc

qu'à priori les temps de calculs utilisés pour calculer $g_{M\downarrow}$ et $g_{M\uparrow}$ peuvent différer, on peut utiliser l'astuce suivante : lorsque M est une machine avec oracle X, les machines $M\downarrow$ et $M\uparrow$ données par l'observation 7.6 sont construites de telle sorte qu'il existe une même machine M' avec oracle X et des machines M_1 et M_2 sans oracle telles que $M\downarrow$ s'écrive comme la composée de M' et de M_1 et telles que $M\uparrow$ s'écrive comme la composée de M' et de M_2. On peut alors remplacer le système $\overline{H_1}$ de la preuve du lemme 7.8 par la composition des systèmes DCPM donnés par le lemme 7.7 appliqué à $g_{M'\downarrow}$ puis à $g_{M_1\downarrow}$, et remplacé le système $\overline{H_2}$ de la preuve du lemme 7.8 par la composition des systèmes DCPM donnés par le lemme 7.7 appliqué à $g_{M'\uparrow}$ puis à $g_{M_2\uparrow}$. Il est alors facile de vérifier que les temps de calculs utilisés pour calculer $g_{M\downarrow}$ et $g_{M\uparrow}$ doivent alors nécessairement coïncider. □

Le théorème 7.4 se généralise alors immédiatement en :

Théorème 7.5 *Soit $X \subset \Sigma^*$ un langage. Soit $d \geq 3$ un entier tel que toute machine de Turing avec oracle X soit DCPM-simulable en dimension d.*

Alors tout langage L de Σ_2^X est semi-reconnu par un système DCPM bien défini de dimension $d+1$.

7.4 Ascension de la hiérarchie hyperarithmétique

Dans cette section, nous généralisons les constructions de la section précédente pour prouver que tout langage de la hiérarchie arithmétique peut être reconnu par un système DCPM de dimension 5. Puis nous généralisons nos résultats pour prouver que, pour $k \geq 0$, tout langage du ω^kème niveau de la hiérarchie hyperarithmétique est semi-reconnu par un système DCPM de dimension $2k+3$ et que tout langage du $\omega^k + 1$ème niveau est semi-reconnu par un système DCPM de dimension $2k+4$.

Le plan de cette section est le suivant : dans la section 7.4.1, nous décrivons le principe de la construction que nous allons utiliser. Dans la section 7.4.2, nous prouvons que tous les langages de la hiérarchie arithmétique peuvent être reconnus en dimension 5. Enfin, dans la section 7.4.3, nous généralisons ce résultat aux systèmes DCPM de dimensions supérieures.

7.4.1 Principe général

Le principe de la construction est le suivant : supposons que nous ayons un système DCPM $\overline{H} = (\overline{X}, \overline{f})$ homogène de dimension $d+1$. Soit $\overline{R}$ une région particulière de ce système.

Considérons le système DCPM $H' = (X', f')$ de dimension $d+2$ défini par

$$X' = \overline{X} \times [0,1]$$

$$f'(x_1,\ldots,x_{d+2}) = \begin{cases} (\overline{f}(x_1,\ldots,x_{d+1}),0) & \text{si } (x_1,\ldots,x_{d+1}) \notin \overline{R} \text{ et } x_{d+2} \in [0,1] \\ (\overline{f}(x_1,\ldots,x_{d+1}),1) & \text{si } (x_1,\ldots,x_{d+1}) \in \overline{R} \text{ et } x_{d+2} \in [0,1] \end{cases}$$

Soit $\phi = (\phi_1,\ldots,\phi_{d+2})$ une trajectoire de H'. Notons $\overline{\phi} = (\phi_1,\ldots,\phi_{d+1})$. Lorsque la projection $\overline{\phi}$ de ϕ n'est pas dans la région $\overline{R}$, la composante ϕ_{d+2} de la trajectoire ϕ selon l'axe de coordonnées x_{d+2} reste inchangée. Lorsque la projection $\overline{\phi}$ de la trajectoire ϕ est dans la région $\overline{R}$, ϕ_{d+2} augmente à la vitesse 1 : voir la figure **??**.

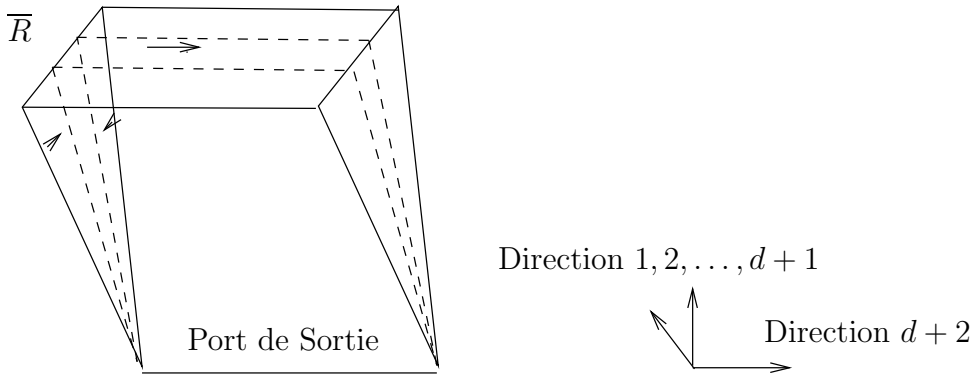


fig :dim5Principe de la construction

Supposons maintenant que le système $\overline{H}$ est construit de telle sorte que toutes les trajectoires de $\overline{H}$ convergent vers le point de coordonnées $(0,\ldots,0)$. Toutes les trajectoires de H' qui ne quittent pas $\overline{X}\times[0,1]$ convergent alors vers le polyèdre de $\mathbb{R}^{d+2}$ d'équation $\{0\}^{d+1} \times [0,1]$ dans la base canonique de $\mathbb{R}^{d+2}$. Une trajectoire ϕ partant du point de coordonnées $(x_1,\ldots,x_{d+1},0)$ converge vers le point de coordonnées $x^* = (0,\ldots,0,x^*_{d+2})$ où x^*_{d+2} est la somme totale des temps pendant lesquels la projection $\overline{\phi}$ de ϕ appartenait à la région $\overline{R}$: en d'autres termes, le système H' calcule la fonction qui à $(x,0)$, $x \in \mathbb{R}^{d+1}$, associe le temps total passé dans la région $\overline{R}$ par la trajectoire $\overline{\phi}$ de $\overline{H}$ partant de x.

Nous allons utiliser ce principe pour construire un système DCPM de dimension 5 qui prenne en entrée un point réel codant un langage $X \subset \Sigma^*$ et qui converge alors vers le point réel qui code le problème de l'arrêt des machines de Turing avec oracle X.

Grâce à ce système, il sera alors facile de reconnaître tous les langages de la hiérarchie arithmétique : si l'on pose $\emptyset^{(0)} = \emptyset$, pour calculer pour $k \geq 0$, le codage du problème $\emptyset^{(k+1)}$ de l'arrêt des machines de Turing avec oracle $\emptyset^{(k)}$, il suffira d'itérer k fois ce système sur le codage de l'ensemble vide.

7.4.2 Reconnaissance de la hiérarchie arithmétique en dimension 5

Dans cette section, nous utilisons le principe exposé dans la section précédente pour prouver formellement que tout langage de la hiérarchie arithmétique est semi-reconnu par un système DCPM de dimension 5.

Observons tout d'abord qu'il est facile de généraliser l'observation 7.6 en :

Observation 7.7 *Soit $h : \Sigma^\omega \to \mathbb{N}$ une fonction partielle récursive et soit M une machine de Turing (respectivement avec oracle $X \subset \Sigma^*$).*

Il existe deux machines de Turing $M\uparrow$ et $M\downarrow$ (respectivement avec oracle X) régulières à un ruban d'alphabet $\Sigma = \{0,1\}$ telles que, pour tout mot $w \in \Sigma^\omega$ à la fois accepté par M et dans le domaine de h, si l'on note $w' \in \Sigma^\omega$ le ruban avec lequel M accepte,

- *$M\downarrow$ accepte w avec w comme ruban.*
- *$M\uparrow$ accepte w avec w' comme ruban.*
- *$M\downarrow$ et $M\uparrow$ acceptent w en des temps $t_{M\downarrow}$ et $t_{M\uparrow}$ tels que*

$$t_{M\downarrow} = t_{M\uparrow} + h(w)$$

Preuve: Soit M' une machine de Turing (respectivement avec oracle X) régulière à un ruban qui accepte tout mot $w \in \Sigma^\omega$ accepté par M avec $\#^{h(w)/2-2}w_1\#w_1'\#w_2$ comme ruban : $\#$ désigne une lettre de l'alphabet utilisée comme délimiteur et w_1, w_1', w_2 désignent les mots associés au mot w comme dans la preuve de l'observation 7.6.

Il suffit de construire $M\uparrow$ comme la machine M' à laquelle sont ajoutées des instructions qui transforment le ruban $\#^{h(w)/2-2}w_1\#w_1'\#w_2$ en $w_1'w_2$ de la façon suivante : le préfixe $\#^{h(w)/2-2}w_1\#w_1'\#$ est balayé successivement de gauche à droite et de droite à gauche. Chaque balayage supprime le caractère le plus à droite de w_1. Lorsqu'il ne reste plus de caractère dans w_1, un dernier balayage supprime les $h(w)/2$ symboles $\#$ restants.

Il suffit de construire $M\downarrow$ comme la machine M' à laquelle sont ajoutées des instructions qui transforme le ruban $\#^{h(w)/2-2}w_1\#w_1'\#w_2$ en w_1w_2 de la façon suivante : le préfixe $\#^{h(w)}w_1\#w_1'\#$ est balayé successivement de gauche à droite et de droite à gauche. Chaque balayage supprime le caractère le plus à droite de w_1'. Lorsqu'il ne reste plus de caractère dans w_1', les $h(\omega)/2$ symboles $\#$ restants sont parcourus une fois de gauche à droite et de droite à gauche sans être modifiés. Enfin un dernier balayage supprime les $h(\omega)/2$ symboles $\#$ restants. □

Maintenant si l'on utilise ces machines $M\downarrow$ et $M\uparrow$ dans la preuve du lemme 7.8 au lieu des machines de l'observation 7.6, on obtient immédiatement : voir la figure **??** page **??**.

Lemme 7.10 *Soient $h : \Sigma^\omega \to \mathbb{N}$ une fonction partielle récursive et M une machine de Turing régulière à un ruban d'alphabet $\Sigma = \{0,1\}$.*

Il existe un système DCPM homogène de dimension 4 *qui calcule une fonction partielle* $g : [0,1]^2 \to [0,1]^2$ *telle que*

$$g : (\frac{\Xi(w)}{x_{d+1}}, x_{d+1}) \mapsto \begin{cases} (0,0) & \text{si } w \text{ n'est pas accepté par } M \\ & \text{ou si } w \text{ n'est pas dans le domaine de } h \\ (\frac{\Xi(w')}{2^{h(w)}x_{d+1}}, \frac{1}{x_{d+1}2^{h(w)}}) & \text{si } M \text{ accepte } w \text{ au temps } n \in \mathbb{N} \\ & \text{avec } w' \in \Sigma^\omega \text{ comme ruban.} \end{cases}$$

pour tout mot $w \in \Sigma^\omega$ *et pour tout* $0 < x_{d+1} \leq 1$.

En outre, il existe une constante $k > 0$ *telle que pour tout* $w \in \Sigma^\omega$ *et pour tout* $0 < x_{d+1} \leq 1$ *cette fonction soit calculée en temps inférieur à* kx_{d+1}.

Nous allons utiliser la région $\overline{R}$ construite comme l'homogénéisation de la région particulière R suivante :

Observation 7.8 (Région R) *Soit* d *un entier.*

Il existe un système DCPM homogène borné R *de dimension* d *réduit à une seule région qui calcule la fonction identité de* $[0,1]^{p-1}$ *dans* $[0,1]^{p-1}$ *en temps exactement* 1.

Preuve: Il suffit de poser $R = (X, f)$ avec $X = [0,1]^d$ et $f(x) = (0, \ldots, 0, 1)$ pour tout $x \in X$. □

Nous obtenons alors : voir la figure **??**.

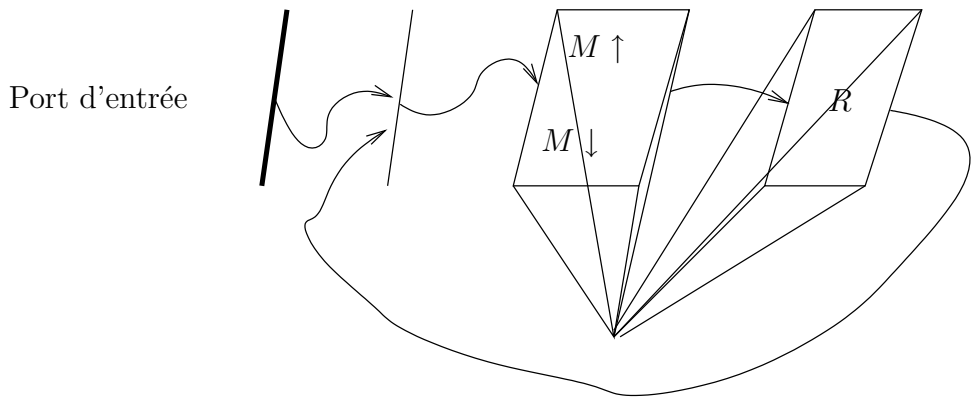


fig :hyperarithmLe système construit dans le lemme 7.11

Lemme 7.11 *Soit* $r : \Sigma^\omega \times \mathbb{N} \to \mathbb{N}$ *une fonction récursive.*

Il existe un système DCPM homogène borné de dimension 5 *qui calcule une fonction partielle* $g : [0,1] \to [0,1]$ *telle que*

$$g(\Xi(w)) = \sum_{j=0}^{\infty} \frac{1}{2^{r(w,j)}}$$

pour tout mot $w \in \Sigma^\omega$ *tel que cette série soit définie, convergente et de limite dans* $[0,1]$.

Preuve: Posons $r(w,-1)=0$. Notons $l:\Sigma^\omega\times\mathbb{N}\to\mathbb{N}$ la fonction partielle récursive définie par $l(w,n)=r(w,n)-r(w,n-1)$ pour tout $w\in\Sigma^\omega$ et pour tout $n\in\mathbb{N}$.

Soit M_0 une machine de Turing qui accepte tout mot $w\in\Sigma^\omega$ avec $\overline{0}\#\overline{l(w,0)}\#w$ comme ruban : $\#$ désigne une lettre de l'alphabet Σ utilisée comme délimiteur et $\overline{n}$ désigne l'écriture sur l'alphabet Σ de l'entier n pour tout $n\in\mathbb{N}$.

Soit M une machine de Turing qui accepte tout mot du type $\overline{n}\#\overline{n'}\#w$ avec le mot $\overline{n+1}\#\overline{l(w,n+1)}\#w$ comme ruban, pour tout $n,n'\in\mathbb{N}$, $w\in\Sigma^\omega$.

Soit $h:\Sigma^\omega\to\mathbb{N}$ une fonction partielle récursive qui à tout mot du type $\overline{n}\#\overline{n'}\#w$ associe n', pour tout $n,n'\in\mathbb{N}$, $w\in\Sigma^\omega$.

Nous commençons par construire un système DCPM homogène $\overline{H}=(\overline{X},\overline{f})$ de dimension 4 de la façon suivante : nous insérons dans $\overline{H}$ un système DCPM H_0 qui simule la machine de Turing M_0, le système DCPM $\overline{H_1}$ obtenu par le lemme 7.10 pour la machine M précédente et pour la fonction h précédente. Nous ajoutons l'homogénéisation $\overline{R}$ du système DCPM R de l'observation 7.8. Nous insérons un port (P,B) de dimension 1 et un port homogène (P',B') de dimension 2. Nous ajoutons un chemin homogène du port d'entrée (P,B) vers le port d'entrée de H_0, un chemin homogène du port de sortie de H_0 vers la projection du port (P',B'), un chemin homogène du port (P',B') vers le port d'entrée de $\overline{H_1}$, un chemin homogène du port de sortie de $\overline{H_1}$ vers le port d'entrée de $\overline{R}$, un chemin homogène du port de sortie de $\overline{R}$ vers (P',B') : voir la figure **??**.

Soit $w\in\Sigma^\omega$ un mot. La trajectoire partant du point du port d'entrée (P,B) de coordonnées $(\Xi(w),1)$ dans la base B passe à travers H_0 puis alternativement à travers $\overline{H_1}$ et à travers $\overline{R}$: voir la figure **??**. Par construction, le $n$ème passage par le port d'entrée de $\overline{H_1}$ se fait au point dont les coordonnées sont $(\Xi(w')/2^{r(w,n-2)},1/2^{r(w,n-2)})$ dans la base associée à ce port où $w'=\overline{n-1}\#\overline{l(w,n-1)}\#w$. Le $n$ème passage par le port de sortie de $\overline{H_1}$ ou par les ports d'entrée et de sortie de $\overline{R}$ se font aux points dont les coordonnées sont $(\Xi(w'')/2^{r(w,n-1)},1/2^{r(w,n-1)})$ dans les bases associées à ces ports où $w''=\overline{n}\#\overline{l(w,n)}\#w$.

Supposons que le mot $w\in\Sigma^\omega$ soit tel que la série $\sum_{j=0}^{\infty}\frac{1}{2^{r(w,j)}}$ soit définie et convergente. Cela implique que la suite qui à $j\in\mathbb{N}$ associe $\frac{1}{2^{r(w,j)}}$ converge vers 0. La trajectoire correspondante à w converge donc dans $\mathbb{R}^4$ vers le point de coordonnées $(0,\ldots,0)$.

Construisons maintenant le système DCPM $H'=(X',f')$ de dimension 5 en posant

$$X'=\overline{X}\times[0,1]$$

$$f'(x_1,\ldots,x_{d+2})=\begin{cases}(\overline{f}(x_1,\ldots,x_{d+1}),0) & \text{si } (x_1,\ldots,x_{d+1})\notin\overline{R} \text{ et } x_{d+2}\in[0,1]\\ (\overline{f}(x_1,\ldots,x_{d+1}),1) & \text{si } (x_1,\ldots,x_{d+1})\in\overline{R} \text{ et } x_{d+2}\in[0,1]\end{cases}$$

La composante selon le 5ème axe de coordonnées d'une trajectoire partant du point de $P\times\{0\}$ correspondant au mot w reste inchangée sauf lorsque la projection sur $\mathbb{R}^4$ de la trajectoire appartient à $\overline{R}$. Or au $n$ème passage à travers $\overline{R}$

elle augmente à vitesse 1 pendant un temps égal à $1/2^{r(w,n-1)}$. Elle converge donc vers le point de coordonnées $(0,\ldots,0,g(w))$ du polyèdre d'équation $\{0\}^4 \times [0,1]$ dans la base canonique de $\mathbb{R}^5$. □

Nous avons vu dans le chapitre 1 qu'il était possible de coder un langage $L \subset \Sigma^*$ par un mot infini :

Définition 7.11 (Codage ρ) *Soit Σ un alphabet.*

Le codage ρ des parties de Σ^* vers Σ^ω est défini pour tout $L \subset \Sigma^*$ par :

$$\rho(L) = w_1 w_2 \ldots w_k \ldots$$

où $w_i \in \{1,0\}$ indique l'appartenance du i ième mot de Σ^* à L.

Du lemme 7.11 nous obtenons alors :

Corollaire 7.6 (Réalisation de $L \mapsto L^{(1)}$) *Il existe un système DCPM de dimension 5 qui calcule une fonction partielle $g : [0,1] \to [0,1]$ telle que*

$$g : \Xi(\rho(L)) \mapsto \Xi(\rho(L^{(1)}))$$

pour tout langage $L \subset \Sigma^*$, où $L^{(1)}$ désigne le problème de l'arrêt des machines de Turing avec oracle L, c'est-à-dire le langage

$$L^{(1)} = \{\overline{u} | u \in \mathbb{N} \wedge \overline{u} \in W_u^L\}$$

Preuve: Il est facile de construire une fonction récursive $r : \Sigma^\omega \times \mathbb{N} \to \mathbb{N}$ telle que pour tout $L \subset \Sigma^*$, $\Xi(\rho(L^{(1)})) = \sum_{j=0}^{\infty} \frac{1}{2^{r(\rho(L),j)}}$. □

Maintenant puisqu'il est possible d'obtenir le codage du problème $L^{(1)}$ à partir du codage de L, pour tout langage $L \subset \Sigma^*$ arbitraire, en itérant ce système, il est facile de simuler toute machine de Turing avec oracle $\emptyset^{(\omega)} = \{< \overline{u}, v > | u \in \mathbb{N}, v \in X^{(u)}\}$.

Si l'on veut être formel, il nous faut modifier le corollaire 7.6 en (dans toute la suite de ce chapitre # désigne une lettre de l'alphabet Σ utilisable comme délimiteur et $<,>$ désigne une fonction récursive bijective de $\Sigma^* \times \Sigma^* \to \Sigma^*$) :

Corollaire 7.7 (Réalisation de $L \mapsto L^{(1)}$) *Il existe un système DCPM de dimension 5 qui calcule une fonction partielle $g : [0,1] \to [0,1]$ telle que*

$$g : \Xi(w\#\rho(L)) \mapsto \Xi(w\#\rho(L^{(1)}))$$

pour tout mot $w \in \Sigma^*$, et pour tout langage $L \subset \Sigma^*$, où $L^{(1)}$ désigne le problème de l'arrêt des machines de Turing avec oracle L.

On obtient alors :

Lemme 7.12 (Réalisation de $L \mapsto L^{(u)}$) *Il existe un système DCPM de dimension* 5 *qui calcule une fonction partielle* $g : [0,1] \to [0,1]$ *telle que*

$$g : \Xi(w\#\overline{u}\#\rho(L)) \mapsto \Xi(w\#\rho(L^{(u)}))$$

pour tout mot $w \in \Sigma^*$*, pour tout entier* $u \in \mathbb{N}$ *et pour tout langage* $L \subset \Sigma^*$*, où, pour tout entier* $u \in \mathbb{N}$*,* $L^{(u+1)}$ *désigne le problème de l'arrêt des machines de Turing avec oracle* $L^{(u)}$*.*

Preuve: Soit M_0 une machine de Turing qui transforme tout mot du type $w\#\overline{u}\#\rho(L)$ en le mot $< w, \overline{u} > \#\rho(L)$ pour tout $w \in \Sigma^*, u \in \mathbb{N}, L \subset \Sigma^*$. Soit M une machine qui transforme tout mot du type $< w, \overline{n} > \#\rho(L)$ en $< w, \overline{n-1} > \#\rho(L)$ pour tout $w \in \Sigma^*, u \in \mathbb{N}, L \subset \Sigma^*$. Soit M_1 une machine qui transforme tout mot du type $< w, \overline{0} > \#\rho(L)$ en $w\#\rho(L)$ pour tout $w \in \Sigma^*, u \in \mathbb{N}, L \subset \Sigma^*$.

Le système DCPM de dimension 5 est alors construit de telle sorte que qu'il simule M_0, qu'il itère ensuite la composition de la simulation de la machine M avec le système DCPM du corollaire 7.7 jusqu'à obtenir une configuration qui encode un mot du type $< w, \overline{0} > \#\rho(L)$. Lorsqu'une telle configuration est atteinte, le système simule alors la machine M_1 sur cette configuration. □

En construisant une fonction $r : \Sigma^\omega \times \mathbb{N} \to \mathbb{N}$ adéquate et en utilisant le lemme 7.11 on a alors facilement :

Lemme 7.13 (Codage de l'ensemble vide) *Il existe un système DCPM de dimension* 5 *qui calcule une fonction partielle* $g : [0,1] \to [0,1]$ *telle que* $g : \Xi(w) \mapsto \Xi(w\#\rho(\emptyset))$ *pour tout mot* $w \in \Sigma^*$*.*

De même si l'on utilise la proposition 1.6 du chapitre 1 on a :

Lemme 7.14 (Codage de X à partir de $X^{(u)}$) *Il existe un système DCPM de dimension* 5 *qui calcule une fonction partielle* $g : [0,1] \to [0,1]$ *telle que* $g : \Xi(w\#\overline{u}\#\rho(L^{(u)})) \mapsto \Xi(w\#\rho(L))$ *pour tout mot* $w \in \Sigma^*$*, pour tout entier* $u \in \mathbb{N}$ *et pour tout langage* $L \subset \Sigma^*$*.*

On obtient alors : rappelons que le système O de notation des ordinaux récursifs a été introduit dans le chapitre 1.

Théorème 7.6 *Soit* z *une notation dans le système* O *de l'ordinal* ω*.*

Toute machine de Turing avec oracle $\emptyset^{(z)}$ *est DCPM-simulable en dimension* 5*.*

Preuve: Une machine de Turing M avec oracle $\emptyset^{(z)}$ est une machine de Turing classique si ce n'est qu'elle peut se déplacer sur son second ruban pour répondre à des questions du type "est ce que le mot v est dans le langage $\emptyset^{(u)}$?" : voir le chapitre 1.

Il suffit de construire un système DCPM de dimension 5 qui utilise les fonctions des lemmes 7.12, 7.13 et 7.14 pour simuler la machine M de la façon suivante : sur un mot $w \in \Sigma^*$, le système commence par appeler la fonction du lemme 7.13 pour obtenir le mot infini $w\#\rho(\emptyset)$. Le système simule alors la machine M sur w tant que celle-ci ne fait pas appel à son oracle. Lorsque M fait appel à son oracle et pose une question du type $v \in \emptyset^{(u)}$?, M encode dans un mot w' son état actuel et la valeur de u et de v, appelle la fonction du lemme 7.12 sur le mot infini $w'\#\overline{u}\#\rho(\emptyset)$ de façon à obtenir le mot infini $w'\#\rho(\emptyset^{(u)})$, lit alors la $v$ème lettre de $\emptyset^{(u)}$ pour déterminer si v est un mot de $\emptyset^{(u)}$, puis appelle alors la fonction du lemme 7.14 sur le mot $w'\#\overline{u}\#\emptyset^{(u)}$ de façon à revenir dans l'état $w'\#\emptyset$ et pouvoir poursuivre la simulation de M. □

On obtient alors :

Corollaire 7.8 (Reconnaissance de Σ_ω en dimension 5) *Soit L un langage de Σ_ω.*

L est semi-reconnu par un système DCPM de dimension 5.

7.4.3 Généralisation aux dimensions supérieures

Nous généralisons maintenant nos résultats aux systèmes DCPM de dimensions supérieures.

Le lemme 7.11 se généralise facilement en :

Lemme 7.15 *Soit $d \geq 3$ un entier.*

Soit $r : \Sigma^\omega \times \mathbb{N} \to \mathbb{N}$ une fonction calculable en dimension d dans le sens suivant : il existe une fonction $g_M : [0,1] \to [0,1]^2$ DCPM-calculable en dimension d, une constante $\beta_M \in \mathbb{Q}$ telle que, pour tout $w \in \Sigma^*$, $n \in \mathbb{N}$, $w' \in \Sigma^\omega$, si l'on note n_0 le plus petit entier n tel que la $n$ème itération de g_M sur $\Xi(w\#\overline{n}\#w')$ soit dans $[0, \beta_M] \times [0,1]$, alors $g_M^{n_0}(\Xi(w\#\overline{n}\#w')) = (\Xi(w\#\overline{r(w,n)}\#w'))$.

Alors il existe un système DCPM homogène borné de dimension $d+2$ qui calcule une fonction partielle $g : [0,1] \to [0,1]$ telle que

$$g(\Xi(w)) = \sum_{j=0}^{\infty} \frac{1}{2^{r(w,j)}}$$

pour tout mot $w \in \Sigma^\omega$ tel que cette série soit définie, convergente et de limite dans $[0,1]$.

Les corollaires 7.6 et 7.7 se généralisent alors immédiatement en :

Corollaire 7.9 (Réalisation de $L \mapsto L^{(\omega^k)}$) *Soient $d \geq 3$ et $k \geq 1$ deux entiers.*

Supposons que, pour toute notation z dans le système O de l'ordinal ω^k, il existe un système DCPM de dimension d qui calcule une fonction partielle $g : [0,1] \to [0,1]$ telle que

$$g : \Xi(w\#\overline{u}\#\rho(L)) \mapsto \Xi(w\#\rho(L^{(u)}))$$

pour tout mot $w \in \Sigma^*$, pour tout langage $L \subset \Sigma^*$ et pour toute notation $u <_0 z$ dans le système 0. .

Soit z une notation dans le système O de l'ordinal ω^k. Il existe un système DCPM de dimension $d+2$ qui calcule une fonction partielle $g : [0,1] \to [0,1]$ telle que

$$g : \Xi(w\#\rho(L)) \mapsto \Xi(w\#\rho(L^{(z)}))$$

pour tout $w \in \Sigma^*$ et pour tout langage $L \subset \Sigma^*$.

Le lemme 7.12 se généralise en :

Lemme 7.16 (Réalisation de $L \mapsto L^{(\omega^{k+1}u)}$) *Soient $d \geq 3$ et $k \geq 1$ deux entiers.*

Supposons que, pour toute notation z dans le système O de l'ordinal ω^k, il existe un système DCPM de dimension d qui calcule une fonction partielle $g : [0,1] \to [0,1]$ telle que

$$g : \Xi(w\#\overline{u}\#\rho(L)) \mapsto \Xi(w\#\rho(L^{(u)}))$$

pour tout mot $w \in \Sigma^*$, pour tout langage $L \subset \Sigma^*$ et pour toute notation $u <_0 z$ dans le système 0.

Soit z' une notation dans le système O de l'ordinal ω^{k+1}. Il existe un système DCPM de dimension $d+2$ qui calcule une fonction partielle $g : [0,1] \to [0,1]$ telle que

$$g : \Xi(w\#\overline{u}\#\rho(L)) \mapsto \Xi(w\#\rho(L^{(u)}))$$

pour tout mot $w \in \Sigma^*$, pour tout langage $L \subset \Sigma^*$ et pour toute notation $u <_0 z'$ dans le système 0.

Le théorème 7.6 se généralise alors en :

Proposition 7.1 *Soient $d \geq 3$ et $k \geq 1$ deux entiers.*

Supposons que, pour toute notation z dans le système O de l'ordinal ω^k, il existe un système DCPM de dimension d qui calcule une fonction partielle $g : [0,1] \to [0,1]$ telle que

$$g : \Xi(w\#\overline{u}\#\rho(L)) \mapsto \Xi(w\#\rho(L^{(u)}))$$

pour tout mot $w \in \Sigma^*$, pour tout langage $L \subset \Sigma^*$ et pour toute notation $u <_0 z$ dans le système 0.

Soit z une notation dans le système O de l'ordinal ω^{k+1}. Toute machine de Turing avec oracle $\emptyset^{(z)}$ est DCPM-simulable en dimension $d+2$.

Nous obtenons alors :

Théorème 7.7 (Puissance de calcul des systèmes DCPM) *Soit $k \geq 0$ un entier.*

1. *Tout langage L de Σ^k_ω est semi-reconnu par un système DCPM bien défini de dimension $2k+3$.*
2. *Tout langage L de Σ_{ω^k+1} est semi-reconnu par un système DCPM bien défini de dimension $2k+4$.*

Preuve: Les assertions pour $k = 0$ sont données par le corollaire 7.5 et par le théorème 7.4.

Le lemme 7.16 prouve par récurrence sur k que pour toute notation z dans le système O de l'ordinal ω^k, il existe un système DCPM de dimension $2k+3$ qui calcule une fonction partielle $g : [0,1] \to [0,1]$ telle que $g : \Xi(w\#\overline{u}\#\rho(L)) \mapsto \Xi(w\#\rho(L^{(u)}))$ pour tout mot $w \in \Sigma^*$, pour tout langage $L \subset \Sigma^*$ et pour toute notation $u <_0 z$ dans le système 0. Les assertions pour $k \geq 1$ découlent alors du théorème 7.5 et de la proposition 7.1. □

CHAPITRE

8 Bornes sur la puissance des systèmes DCPM

8.1 Introduction

Dans le chapitre 7, nous avons prouvé que les systèmes DCPM pouvaient reconnaître certains langages de la hiérarchie hyperarithmétique. En particulier, dans le théorème 7.7, nous avons prouvé que, pour tout entier $k \geq 0$, tout langage de Σ_{ω^k} pouvait être semi-reconnu par un système DCPM de dimension $2k+3$, et que tout langage de Σ_{ω^k+1} pouvait être semi-reconnu par un système DCPM de dimension $2k+4$.

Dans ce chapitre nous prouvons que ces résultats sont optimaux : nous prouvons que tout langage semi-reconnu par un système DCPM de dimension $2k+3$ est dans Σ_{ω^k} et que tout langage semi-reconnu par un système DCPM de dimension $2k+4$ est dans Σ_{ω^k+1}.

Nous obtenons ainsi une caractérisation complète de la puissance de calcul des systèmes DCPM en fonction de leurs dimensions.

Le plan de ce chapitre est le suivant : dans la section 8.2, nous présentons une première majoration de la puissance de calcul des systèmes DCPM en prouvant que tout langage semi-reconnu par un système DCPM est dans la hiérarchie hyperarithmétique. Dans la section 8.3, nous affinons ce résultat en prouvant que tout langage semi-reconnu par un système DCPM de dimension 3 est récursivement énumérable. Enfin, dans la section 8.4, nous caractérisons la puissance de calcul des systèmes DCPM de dimensions arbitraires.

8.2 Première majoration

8.2.1 Principe

Dans la section 8.2.2, nous introduisons deux notions de temps : le *temps continu* qui correspond à la notion de temps du chapitre précédent, et le *temps discret* qui correspond au nombre de transitions discrètes effectuées par la trajectoire. Dans la section 8.2.3, nous montrons que l'on peut associer à tout point de l'espace d'un système DCPM une dimension que nous appelons *dimension locale*. Dans la section 8.2.4, nous utilisons cette notion pour majorer le temps

discret d'une trajectoire de temps continu fini. Dans les sections 8.2.5 et 8.2.6, nous en déduisons que tout langage reconnu ou semi-reconnu par un système DCPM est dans la hiérarchie hyperarithmétique.

8.2.2 Temps discret et temps continu

Nous introduisons maintenant les deux notions de temps que nous allons utiliser : le temps discret et le temps continu.

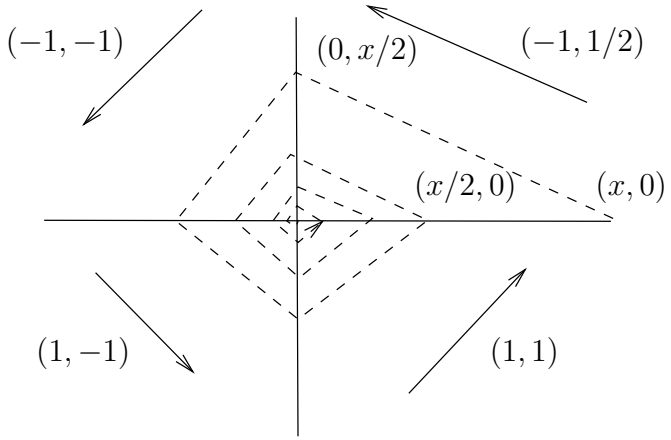


FIG. 8.1 – Paradoxe de Zénon : une trajectoire de temps continu 5 et de temps discret ω entre le point $(x,0)$ et le point $(0,0)$

Le temps continu correspond à la notion de temps utilisée dans le chapitre précédent :

Définition 8.1 (Temps continu) *Soit H un système DCPM et soit ϕ une trajectoire de H définie sur un intervalle I.*

La trajectoire ϕ est dite de temps continu fini *si I est un intervalle fermé borné. Le* temps continu *de la trajectoire est défini comme le réel $t_* - t_0$ si I s'écrit $I = [t_0, t^*]$.*

Remarque. Rappelons que toute trajectoire définie sur un intervalle semi-ouvert à droite borné s'étend en une trajectoire définie sur l'adhérence de l'intervalle : voir la proposition 2.1 du chapitre 2.

Le temps discret correspond quant à lui intuitivement au nombre de transitions discrètes effectuées par la trajectoire : voir la figure 8.1. Pour le définir proprement, il nous faut passer par l'observation suivante :

Observation 8.1 *Soient H un système DCPM et ϕ une trajectoire de H définie sur un intervalle du type $I = [t_0, t^*]$ ou $I = [t_0, \infty[$.*

Soit T_ϕ l'ensemble constitué des temps $t \in I$ en lesquels ϕ n'est pas dérivable à gauche ou en lesquels la valeur de la dérivée à gauche de ϕ diffère de la valeur de sa dérivée à droite.

T_ϕ est un ensemble bien ordonné.

Preuve: Rappelons qu'un ensemble est *bien ordonné* si toute partie non vide de l'ensemble possède un plus petit élément. Soit J une partie de T_ϕ. Puisque J est un sous-ensemble minoré de $\mathbb{R}$ nous pouvons considérer l'infimum t de J. À ce temps t, la dérivée à droite $\phi_d(t)$ de ϕ doit vérifier $\phi_d(t) = f(\phi(t))$. Il doit exister un $\epsilon > 0$ tel que $\phi(t')$ appartienne au segment $[\phi(t), \phi(t) + f(\phi(t))]$ pour tout $t' \in [t, t + \epsilon]$. L'intervalle $]t, t + \epsilon[$ ne contient alors aucun point de J. Par conséquent, t est un minimum de J. □

Le temps discret se définit alors par :

Définition 8.2 (Temps discret) *Le* temps discret *de la trajectoire ϕ est défini comme l'ordinal isomorphe à l'ensemble bien ordonné T_ϕ.*

Pour tout ordinal β inférieur à cet ordinal, nous pouvons aussi légitimement parler de la position de la trajectoire au temps discret β :

Définition 8.3 (Position à un temps discret β) *Soit α le temps discret de la trajectoire ϕ. Soit β un ordinal vérifiant $\beta < \alpha$ si $I = [t_0, \infty[$ et $\beta \leq \alpha$ si $I = [t_0, t^*]$.*

Nous appelons temps continu de ϕ au temps discret β *le réel t_β tel que T_ϕ restreint à l'intervalle $[0, t_\beta]$ soit isomorphe à l'ordinal β. Nous appelons* position de ϕ au temps discret β *le point x_β défini par $x_\beta = \phi(t_\beta)$.*

Des définitions et de la continuité de ϕ on tire immédiatement :

Observation 8.2 *Supposons que β soit un ordinal limite.*
Alors :
– *t_β est le suprémum des t_γ pour $\gamma < \beta$*
– *x_β est la limite des x_γ pour $\gamma < \beta$.*

8.2.3 Dimension locale

Dans cette section, nous prouvons que l'on peut associer à tout point de l'espace d'un système DCPM une dimension que nous appelons *dimension locale.*

Nous fixons une distance sur $\mathbb{R}^d$: nous choisissons la distance du maximum. Nous notons par *dist* cette distance. Pour $x \in \mathbb{R}^d$ et pour $\epsilon > 0$, nous notons $B(x, \epsilon)$ la boule de centre x et de rayon ϵ, c'est-à-dire l'ensemble des points $y \in \mathbb{R}^d$ tels que $dist(y, x) < \epsilon$.

Commençons par l'observation suivante : voir la figure 8.2.

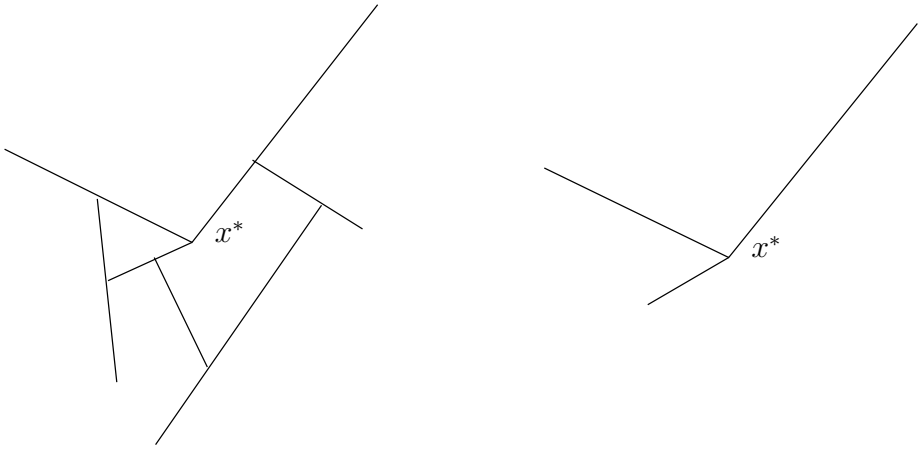


FIG. 8.2 – Illustration de l'observation 8.3 : à gauche un système DCPM ; à droite les régions de ce système qui intersectent un voisinage de x^* assez petit

Observation 8.3 *Soit x^* un point de l'espace d'un système DCPM $H = (X, f)$ de dimension d.*

Il existe $\epsilon > 0$ tel que toute région de H qui intersecte la boule $B(x^*, \epsilon)$ possède le point x^* dans son adhérence.

Preuve: Il suffit de prendre $\epsilon = d/2$ où d est la distance qui sépare x^* des régions de H qui ne contiennent pas le point x^* dans leurs adhérences. □

Intéressons-nous maintenant à la dimension de l'intersection des adhérences des régions qui intersectent cette boule. Si d'' dénote cette dimension, nous définissons la *dimension locale du point x^** comme l'entier $d - d''$. Formellement : voir la figure 8.3.

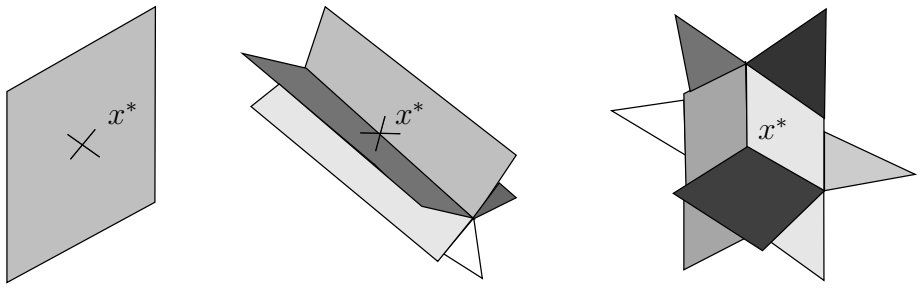


FIG. 8.3 – Des points de dimension locale $1, 2$ et 3 dans un système DCPM de dimension 3

Définition 8.4 (Dimension locale) *Soit x^* un point de l'espace d'un système DCPM $H = (X, f)$ de dimension d.*

Appelons Δ_{x^*} l'intersection des adhérences des régions de H qui intersectent la boule $B(x^*, \epsilon)$ de l'observation précédente.

La dimension locale *d' du point x^* est définie comme l'entier $d - d''$ où d'' est la dimension du polyèdre Δ_{x^*}.*

Cette définition est motivée par l'observation suivante : voir la figure 8.4.

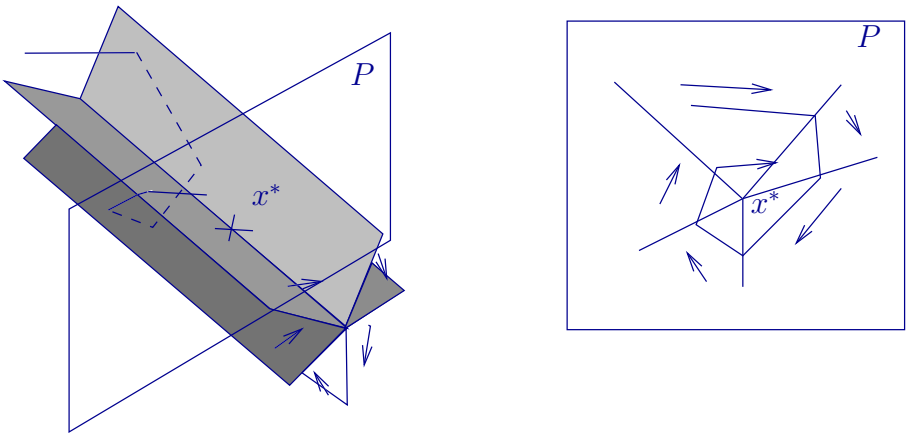


FIG. 8.4 – Si x^* est dimension locale d', alors les trajectoires sont localement données par un système DCPM de dimension d'

Observation 8.4 (Etude locale en un point) *Soit x^* un point de l'espace d'un système DCPM $H = (X, f)$ de dimension d. Soit $B(x^*, \epsilon)$ la boule donnée par l'observation 8.3. Soit Δ_{x^*} le polyèdre de la définition précédente. Supposons que la dimension locale d' du point x^* vérifie $d' < d$. Appelons P_{x^*} le sous-espace affine de dimension d' orthogonal en x^* au polyèdre Δ_{x^*}. Soit ϕ une trajectoire de H, définie sur un intervalle I, telle que $\phi(t) \in B(x^*, \epsilon)$ pour tout $t \in I$.*

Alors la projection orthogonale de ϕ sur P_{x^*} est une trajectoire d'un système DCPM de dimension d'.

Preuve: Notons p l'application qui, à tout point x de $\mathbb{R}^d$, associe sa projection orthogonale sur P_{x^*}. Puisque toute région qui intersecte $B(x^*, \epsilon)$ contient le polyèdre Δ_{x^*} dans son adhérence, deux points de $B(x^*, \epsilon)$ qui ont même image par p ont même image par f. Par conséquent, la projection $p(\phi)$ de ϕ est une trajectoire du système DCPM $H' = (X', f')$ défini par $X' = p(V)$ et $f'(p(x)) = f(x)$ pour tout $x \in X'$: voir la figure 8.4. □

8.2.4 Lien entre temps discret et temps continu

L'étude des systèmes DCPM planaires effectuée dans [9] prouve le résultat suivant : voir l'exemple de la figure 8.1.

Lemme 8.1 ([9]) *Toute trajectoire de temps continu fini d'un système DCPM $H = (X, f)$ de dimension 2 possède un temps discret $T_d \leq \omega$.*

Lorsque T_d vaut ω, la trajectoire est convergente vers un point x^* du système DCPM qui est

1. *soit un point qui n'est pas bien défini : voir la définition 2.12 du chapitre 2.*
2. *soit un point tel que $f(x^*) = 0$.*

Nous utilisons alors ce résultat pour prouver :

Théorème 8.1 *Toute trajectoire de temps continu fini d'un système DCPM de dimension d possède un temps discret T_d qui vérifie*

- *$T_d \leq \omega$ si $d = 2$.*
- *$T_d < \omega^{d-1}$ si $d > 2$.*

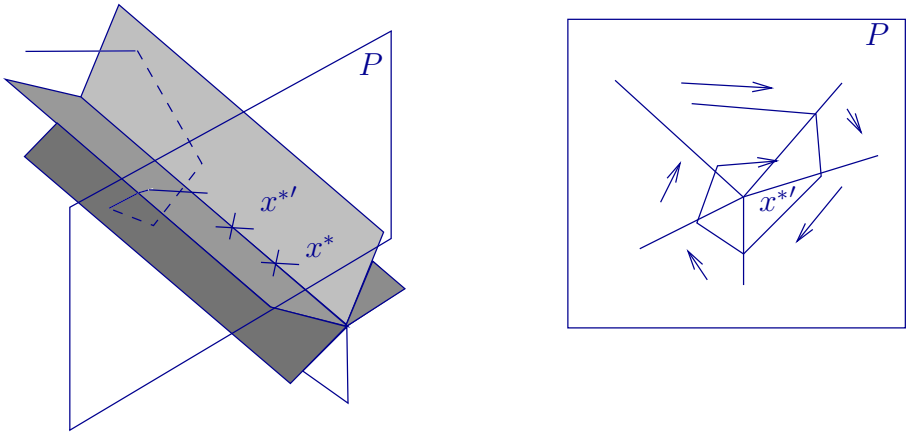
Preuve: Nous prouvons le théorème par récurrence sur la dimension d. Le cas $d = 2$ est le lemme précédent prouvé dans [9] en utilisant le théorème de Jordan.

Supposons maintenant que le théorème soit vrai pour toute dimension $d' < d$. Supposons par l'absurde que H possède une trajectoire ϕ de temps continu fini et de temps discret $T_d \geq \omega^{d-1}$. Pour tout ordinal $\beta \leq T_d$ dénotons par x_β et par t_β la position et le temps continu de la trajectoire au temps discret β.

Considérons $x^* = x_{\omega^{d-1}}$ et $t^* = t_{\omega^{d-1}}$. Nommons $B(x^*, \epsilon)$ la boule donnée par l'observation 8.3. Puisque la fonction ϕ est continue, il existe un réel $t^0 < t^*$ avec $\phi(t) \in B(x^*, \epsilon)$ pour tout $t \in [t^0, t^*]$. Puisque t^* est le suprémum des $t_{\omega^{d-2}i}$, pour $i \in \mathbb{N}$, nous pouvons supposer t^0 du type $t_{\omega^{d-2}i_0}$ pour $i_0 \in \mathbb{N}$.

Considérons alors $x^{*\prime} = x_{\omega^{d-2}(i_0+1)}$ et $t^{*\prime} = t_{\omega^{d-2}(i_0+1)}$. Nommons $B(x^{*\prime}, \epsilon')$ la boule donnée par l'observation 8.3 pour le point $x^{*\prime}$. Puisque ϕ est continue, il existe un réel $t^0 < t^1 < t^{*\prime}$ avec $\phi(t) \in B(x^{*\prime}, \epsilon') \cap B(x^*, \epsilon)$ pour tout $t \in [t^1, t^{*\prime}]$. Puisque $t^{*\prime}$ est le suprémum des $t_{\omega^{d-2}i_0+\omega^{d-3}i}$, $i \in \mathbb{N}$, nous pouvons supposer t^1 du type $t_{\omega^{d-2}i_0+\omega^{d-3}i_1}$ pour $i_1 \in \mathbb{N}$.

Toutes les régions du système DCPM H qui intersectent $B(x^{*\prime}, \epsilon') \cap B(x^*, \epsilon)$ contiennent chacune les points x^* et $x^{*\prime}$ dans leurs adhérences. Puisque les régions d'un système DCPM sont convexes, chacune de ces régions doit aussi contenir le segment $[x^*, x^{*\prime}]$ dans son adhérence. Par conséquent, la dimension locale d' de $x^{*\prime}$ doit être inférieure à $d-1$. Considérons alors la restriction de la trajectoire ϕ à l'intervalle $[t^1, t^{*\prime}]$. Cette trajectoire est de temps discret ω^{d-2}. Par l'observation 8.4 appliquée au point $x^{*\prime}$, la projection de cette trajectoire sur un certain sous-espace affine P de dimension d' doit être une trajectoire d'un système DCPM de dimension d'. Puisque $d' \leq d-1$ et puisque cette trajectoire ϕ' est de temps



discret ω^{d-2} nous obtenons une contradiction avec l'hypothèse de récurrence sauf si $d = 3$.

fig :pliagemodifIllustration du cas $d = 3$

Supposons alors $d = 3$. La projection de la trajectoire sur le sous-espace affine P est une trajectoire d'un système DCPM de dimension 2 de temps discret ω. Par le lemme 8.1, sa limite $x^{*\prime}$ doit nécessairement être un point fixe du système DCPM défini sur P. Cela signifie que le vecteur pente en $x^{*\prime}$ du système DCPM original de dimension 3 doit nécessairement être colinéaire au vecteur $x^*\vec{x}^{*\prime}$: voir la figure **??**. Puisqu'aux temps $t^{*\prime} < t < t^*$ la trajectoire doit rester dans la boule $B(x^*, \epsilon)$, ce vecteur est nécessairement dans la même direction que le vecteur $x^*\vec{x}^{*\prime}$. Par conséquent, il doit exister $t^{*\prime} < t^2 \leq t_{\omega^{d-2}(i_0+1)+1} < t^*$ avec $\phi(t^2) = x^*$. La restriction de la trajectoire ϕ à l'intervalle $[t^2, t^*]$ est une trajectoire de temps continu fini et de même temps discret que ϕ. Si l'on reprend l'étude précédente en remplaçant ϕ par cette nouvelle trajectoire, on obtient qu'il doit exister $t^2 < t^3 < t^*$ avec $\phi(t^3) = x^*$. Puisque l'on a $\phi(t^2) = \phi(t^3)$ la trajectoire ϕ doit être cyclique : pour tout $t \in [t^2, t^3]$ et pour tout $n \in \mathbb{N}$ tel que $t + n(t^3 - t^2)$ appartienne à I on doit avoir $\phi(t + n(t^3 - t^2)) = \phi(t)$. Par conséquent, si α est le temps discret de la restriction de la trajectoire ϕ à l'intervalle $[t^2, t^3]$ et si n_0 est le plus petit entier tel que $t^2 + n_0(t^3 - t^2)$ n'appartienne plus à l'intervalle borné I, alors le temps discret de ϕ doit être inférieur à $\omega^{d-2}(i_0 + 1) + 1 + \alpha n_0$. Nous obtenons une contradiction car α est par définition tel que $\alpha < \omega^2$. □

Observons que les bornes présentées dans le théorème précédent sont optimales. En effet, pour toute dimension d et pour tout ordinal $\alpha < \omega^{d-1}$, il est facile de construire un système DCPM qui possède une trajectoire de temps continu fini et de temps discret α : observez la figure 8.5 pour $d = 2$.

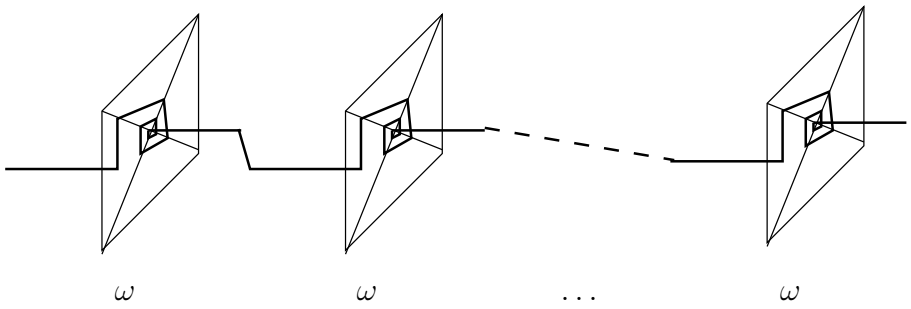


FIG. 8.5 – Pour tout $p \in \mathbb{N}$, on peut construire un système DCPM de dimension 3 avec une trajectoire de temps continu fini et de temps discret ωp

8.2.5 Modèle de calcul

Nous allons maintenant en déduire que tout langage semi-reconnu par un système DCPM est dans la hiérarchie hyperarithmétique.

Avant cela, nous allons préciser le modèle de calcul que nous utiliserons. Nous allons utiliser un modèle analogue à celui de l'analyse récursive introduit dans le chapitre 1. Toutefois, puisque les systèmes DCPM peuvent effectuer des comparaisons à 0 et que ces opérations ne sont pas décidables dans le modèle de l'analyse récursive, nous n'allons pas utiliser la représentation classique $\rho_{\mathbb{R}}$ des réels par une suite de Cauchy convergente (voir la définition 1.9 du chapitre 1) mais une représentation ad hoc. En effet, nous définissons :

Définition 8.5 (Langage $[x]$) *Soit $d \geq 1$ un entier. Supposons que l'on se fixe un codage des polyèdres rationnels de $\mathbb{R}^d$ sur les mots de Σ^*. Soit x un point de $\mathbb{R}^d$. Nous notons $[x]$ le langage constitué des mots $w \in \Sigma^*$ qui codent un polyèdre rationnel P de $\mathbb{R}^d$ contenant x.*

Nous utiliserons alors les représentations suivantes :

Définition 8.6 (Représentations) *Soit $d \geq 1$ un entier. Soit ρ la fonction des parties de Σ^* vers Σ^ω qui associe à un langage $X \subset \Sigma^*$ le mot infini $\rho(X) = w_1 w_2 \dots w_k \dots$ où $w_i \in \{1, 0\}$ indique l'appartenance du iième mot de Σ^* à X : voir le chapitre 1.*

- *La représentation $\rho_{\mathbb{R}^d} : \Sigma^\omega \to \mathbb{R}^d$ de $\mathbb{R}^d$ consiste à représenter un point x de $\mathbb{R}^d$ par le mot infini $\rho([x])$.*
- *$\rho_{\mathbb{N}} : \Sigma^* \to \mathbb{N}$ désigne la notation qui représente un entier n par son écriture $\overline{n}$ sur l'alphabet Σ.*
- *$\rho_{\Sigma^*} : \Sigma^* \to \Sigma^*$ désigne la fonction identité.*

Une fonction sera alors dite calculable si elle l'est pour ces représentations. C'est-à-dire si :

Définition 8.7 (Fonctions calculables) *Une fonction partielle $F : X_1 \times \ldots \times X_k \to X_0$, avec $X_i \in \{\Sigma^*, \mathbb{N}, \mathbb{R}^d\}$ pour $i \in \{0, \ldots, k\}$, est dite* calculable *si et seulement s'il existe une fonction f calculable par une machine de Turing telle que*

$$F(\rho_{X_1}(y_1), \ldots, (\rho_{X_k}(y_k))) = \rho_{X_0}(f(y_1, \ldots, y_k))$$

lorsque $F(\rho_{X_1}(y_1), \ldots, (\rho_{X_k}(y_k)))$ existe.

Ce modèle permet d'avoir le résultat suivant qui n'est pas vrai avec la représentation classique des réels de l'analyse récursive :

Observation 8.5 (Fonction successeur) *Soit H un système DCPM de dimension d.*

La fonction Succ : $\mathbb{R}^d \to \mathbb{R}^d$ qui à un point $x \in \mathbb{R}^d$ associe la position au temps discret 1 de la trajectoire partant de x au temps 0 est calculable.

En outre, il existe une fonction récursive succ : $\mathbb{N} \to \mathbb{N}$ telle que pour $X \subset \Sigma^*$ et pour tout $n \in \mathbb{N}$, si machine de Turing avec oracle X de numéro n reconnaît $[x]$, alors la machine de Turing avec oracle X de numéro $succ(n)$ reconnaît $[Succ(x)]$.

Nous introduisons la notation suivante :

Définition 8.8 (Langage $[f]$) *Supposons fixée une fonction récursive bijective $<,>$ de $\mathbb{N} \times \Sigma^*$ vers Σ^*. Soit $f : \mathbb{N} \to \mathbb{R}^d$ une fonction. Nous notons $[f]$ le langage constitué des mots de Σ^* du type $< n, w >$ tels que n est un entier et w code un polyèdre rationnel P contenant $f(n)$.*

8.2.6 Résultat de majoration

Nous pouvons maintenant majorer la puissance de calcul des systèmes DCPM. Le principe consiste simplement à prouver que le résultat du théorème 8.1 se traduit directement, via l'observation 8.2, en le fait que tout langage semi-reconnu par un système DCPM de dimension d est dans le ω^{d-2}ème niveau de la hiérarchie hyperarithmétique.

Dans le reste de ce chapitre, par abus de notation, nous ne distinguerons pas les entiers de leurs notations dans le système O : ainsi, lorsque n est un entier, nous noterons abusivement $X^{(n)}$ pour le langage $X^{(z)}$ où z est la notation de l'entier n dans le système O.

Nous observons tout d'abord :

Observation 8.6 (Définition de la limite) *Soit $f : \mathbb{N} \to \mathbb{R}^d$ une fonction.*

Il existe une formule du premier ordre définie à l'aide de la relation $[f]$ et de relations récursives qui est vraie si et seulement si la suite $f(n)$, $n \in \mathbb{N}$ est convergente.

Lorsque la suite est convergente, si f^* dénote sa limite, alors le langage $[f^*]$ est définissable par une formule du premier ordre à partir de la relation $[f]$ et de relations récursives.

Ces formules sont équivalentes à des formules existentielles de moins de 3 *alternances de quantificateurs.*

Preuve: La suite $f(n)$, $n \in \mathbb{N}$, est convergente si et seulement si elle est de Cauchy, c'est-à-dire si et seulement si $\forall n_1\ \exists n_2\ \exists P\ \forall n_3\ \forall n_4\ n_3 \geq n_2, n_4 \geq n_2 \rightarrow f(n_3) \in P, f(n_4) \in P, diam(P) < 1/n_1$, où n_1, n_2, n_3, n_4 sont des entiers, P est polyèdre rationnel et $diam(P)$ désigne le diamètre de P. De même, $[f^*]$ est définissable par une formule du premier ordre à l'aide du langage $[f]$ et de relations récursives : tester si f^* appartient à un polyèdre rationnel P est équivalent à tester chacune des inégalités qui définissent le polyèdre : pour vérifier par exemple $a_1x_1 + \ldots + a_dx_d > 0$ il suffit d'écrire $\exists n_1\ \forall n_2\ n_2 \geq n_1 \rightarrow f(n_2) \in P(1/n_1)$, où $P(1/n_1)$ est le polyèdre constitué des points $(x_1, \ldots, x_d)$ avec $a_1x_1 + \ldots + a_dx_d > 1/n_1$. □

Nous déduisons alors de la proposition 1.2 du chapitre 1 :

Corollaire 8.1 *Il existe une fonction récursive $lim : \mathbb{N} \rightarrow \mathbb{N}$ telle que, pour tout oracle $X \subset \Sigma^*$ et pour tout entier u, si la machine de Turing avec oracle X de numéro n reconnaît pleinement $[f]$ et si la suite $f(n)$, $n \in \mathbb{N}$, est convergente, alors la machine de Turing numéro $lim(u)$ avec oracle $X^{(4)}$ reconnaît pleinement le langage $[f^*]$.*

Et aussi :

Lemme 8.2 (Reconnaissance de $[f^*]$) *Soit $f : \mathbb{N} \rightarrow \mathbb{R}^d$ une fonction convergente vers f^*. Soit $X \subset \Sigma^*$ un oracle. Supposons qu'il existe une fonction récursive $g_1 : \mathbb{N} \rightarrow \mathbb{N}$ et une suite croissante récursive $g_2 : \mathbb{N} \rightarrow \mathbb{N}$ de notations dans le système O telles que pour tout entier n, la machine de Turing de numéro $g_1(n)$ avec oracle $X^{(g_2(n))}$ reconnaisse le langage $[f(n)]$. Soit z la notation dans le système O définie par $z = z^* +_0 4$ où z^* est la limite de la suite g_2 : voir la définition 1.8 du chapitre 1.*

Alors le langage $[f^*]$ est $X^{(z)}$-récursif. En outre la dépendance en f, X et en les index de g_1 et de g_2 est uniforme : il existe une fonction récursive limite : $\mathbb{N}^2 \rightarrow \mathbb{N}$ telle que pour tout $X \subset \Sigma^*$, pour toute fonction $f : \mathbb{N} \rightarrow \mathbb{R}^d$ convergente, pour tout $n_1, n_2 \in \mathbb{N}$, si n_1 est le numéro de la machine de Turing qui calcule g_1 et si n_2 est le numéro de la machine de Turing qui calcule g_2, et si pour tout n la machine de Turing de numéro $g_1(n)$ avec oracle $X^{(g_2(n))}$ reconnaît $[f(n)]$, alors $limite(n_1, n_2)$ est le numéro de la machine de Turing avec oracle $X^{(z)}$ qui reconnaît pleinement $[f^*]$.

Preuve: $[f]$ est $X^{(z^*)}$-récursif par la proposition 1.6 du chapitre 1. Il suffit alors d'utiliser le corollaire 8.1 et la proposition 1.7 du chapitre 1. □

Nous pouvons alors prouver : comme dans le chapitre 1, M_n^X dénote la machine de Turing avec oracle X de numéro n.

Proposition 8.1 *Soit H un système DCPM de dimension d.*

Notons $\Phi : \Sigma^* \times \mathbb{N} \to \mathbb{R}^d$ la fonction qui à un mot $w \in \Sigma^*$ et à une notation z dans le système 0 d'un ordinal récursif β associe la position au temps discret β, lorsqu'elle existe, du calcul de H sur le mot w : voir la définition 2.14 du chapitre 2.

Alors il existe une fonction récursive $p : \Sigma^* \times \mathbb{N}^2 \to \mathbb{N}$ et une fonction récursive $q : \mathbb{N} \to \mathbb{N}$ telles que pour tout $w \in \Sigma^*$, pour toute notation z dans le système O, et pour toute notation $u \leq_0 z$ dans le système O d'un ordinal récursif β on ait

$$M_{p(w,u,z)}^{\emptyset^{(q(z))}} = [\Phi(w,u)]$$

Preuve: Notons $<,>$ une fonction récursive bijective de $\mathbb{N}^2$ dans $\mathbb{N}$. Pour tout entier $e \in \mathbb{N}$, notons $\phi_e : \mathbb{N} \to \mathbb{N}$ la fonction calculée par la machine de Turing numéro e et notons $p_e : \mathbb{N}^2 \to \mathbb{N}$ et $q_e : \mathbb{N} \to \mathbb{N}$ les fonctions définies par $\phi_e(< u, z >) = p_e(u,z)$ et $q_e(z) = \phi_e(z)$ pour tout $u, z \in \mathbb{N}$.

Supposons qu'un entier e et un mot w soient fixés. Définissons les fonctions partielles $p : \mathbb{N}^2 \to \mathbb{N}$ et $q : \mathbb{N} \to \mathbb{N}$ par :

- Pour $z = u = 1$ posons $q(z) = 1$ et $p(u,z) = m$ où m est le numéro d'une machine de Turing telle que M_m reconnaisse $[\Phi(w,1)]$.
- Pour $z = 2^x$ posons $q(z) = q_e(x)$.
  - Lorsque $u \neq z$ posons $p(u,z) = p_e(u,x)$.
  - Lorsque $u = z$, posons $p(u,z) = succ(p_e(x,x))$ où $succ$ est la fonction récursive de l'observation 8.5.
- Pour $z = 3.5^y$, soit z^* la limite de la suite $q_e(\phi_y(n))$, $n \in \mathbb{N}$ lorsque cette suite est une suite croissante récursive de notations dans le système O. Posons alors $q(z) = z^* +_o 4$.
  - Lorsque $u \neq z$, posons $p(u,z) = restric(p_e(u,u), q_e(u), q(z))$ où $restric$ est la fonction récursive de la proposition 1.6 du chapitre 1.
  - Lorsque $u = z$, si n_1 désigne le numéro de la machine de Turing qui calcule la fonction qui à $n \in \mathbb{N}$ associe $p_e(\phi_y(n), \phi_y(n))$ et n_2 désigne le numéro de la machine de Turing qui calcule la fonction qui à $n \in \mathbb{N}$ associe $q_e(\phi_y(n))$, posons $p(u,z) = limite(n_1, n_2)$ où $limite$ est la fonction récursive du lemme 8.2.

La fonction q est une fonction partielle récursive et le numéro de la machine de Turing qui la calcule dépend uniformément de e. En d'autres termes, il existe une fonction récursive $r : \mathbb{N} \to \mathbb{N}$ avec $q = \phi_{r(e)}$ pour tout e. Appliquons alors le théorème du point fixe, c'est-à-dire la proposition 1.8 du chapitre 1 : il existe un entier e tel que l'on ait $\phi_e = \phi_{r(e)}$.

Définissons $\psi : \mathbb{N} \to \mathbb{N}$ par $\psi(< u, z >) = p(u,z)$ pour tout u, z. La fonction ψ est une fonction partielle récursive, et le numéro de la machine de Turing qui

la calcule dépend uniformément de e et de w. En d'autres termes, il existe une fonction récursive $r' : \Sigma^* \times \mathbb{N} \to \mathbb{N}$ telle que $\psi = \phi_{r'(w,e)}$. Appliquons alors le théorème du point fixe c'est-à-dire la proposition 1.8 du chapitre 1 : il existe une fonction récursive $g : \Sigma^* \to \mathbb{N}$ telle que $\phi_{g(w)} = \phi_{r'(w,g(w))}$ pour tout $w \in \Sigma^*$.

Il suffit alors de choisir $p : \Sigma^* \times \mathbb{N}^2 \to \mathbb{N}$ comme la fonction qui à w, u, z associe $p_{g(w)}(u, z)$ et $q : \mathbb{N} \to \mathbb{N}$ comme la fonction qui à z associe $q_e(z)$. □

On obtient :

Théorème 8.2 *Soit $L \subset \Sigma^*$ un langage semi-reconnu par un système DCPM de dimension d.*

Alors L est dans la hiérarchie hyperarithmétique : L appartient à $\Sigma_{\omega^{d-2}}$.

Preuve: Par le théorème 8.1, toute trajectoire de temps continu fini d'un système DCPM de dimension d possède un temps discret inférieur à ω^{d-1}. Fixons alors une notation z dans le système O de l'ordinal ω^{d-1}. La proposition précédente nous dit qu'il existe une fonction récursive $p' : \Sigma^* \to \mathbb{N}$ et une notation z' dans le système O d'un ordinal récursif tels que pour tout $w \in \Sigma^*$ et pour toute notation $u \leq_0 z$ dans le système O d'un ordinal récursif β, la machine $M^{\emptyset^{(z')}}_{p'(w,u)}$ reconnaisse le langage $[x_\beta]$ où $[x_\beta]$ est la position au temps discret β du calcul du système DCPM sur le mot w. Le langage L est donc semi-reconnu par la machine de Turing avec oracle $\emptyset^{(z')}$ qui vérifie pour toutes les notations $u \leq_0 z$ dans le système O d'un ordinal $\beta < \omega^{d-1}$ si le point d'acceptation du système DCPM est atteint au temps discret β. Il suffit alors de remarquer que $z' = q(z)$ est par construction une notation de l'ordinal ω^{d-2}. □

8.3 Etude des systèmes DCPM de dimension 3

Nous allons maintenant améliorer les bornes données par le théorème 8.2 en observant géométriquement de plus près ce qui peut se passer en chaque dimension : dans cette section nous prouvons que tout ensemble semi-reconnu en dimension 3 est récursivement énumérable.

8.3.1 Signature des trajectoires

Nous allons tout d'abord prouver que la signature d'une trajectoire de temps continu fini de temps discret ω d'un système DCPM de dimension 3 est nécessairement de signature ultimement périodique : rappelons que la *signature d'une trajectoire* est la suite des régions traversées par la trajectoire.

Ce résultat découle du fait que la sphère de $\mathbb{R}^2$ vérifie le théorème de Jordan et donc que dans le voisinage d'un point de dimension locale 3 on peut appliquer les arguments de [9].

Pour éviter d'avoir à reprendre un à un les arguments de [9], nous allons nous ramener à un cadre étudié par [9]. En effet, les auteurs de [9] mentionnent que

leurs résultats s'étendent à la classe suivante de systèmes dynamiques à temps continu :

Définition 8.9 (Pseudo-système DCPM [9]) *Un* pseudo-système DCPM planaire *est un système dynamique $H = (X, f)$ à temps continu tel que*

1. *$X = \mathbb{R}^2$.*
2. *La fonction f est bornée sur X.*
3. *L'espace X peut se partitionner en un nombre fini de régions connexes.*
4. *Pour toute droite Δ de $\mathbb{R}^2$, si l'on appelle $I_1, \ldots, I_n$ la partition de Δ induite par la partition de X, chacun des segments $I_1, \ldots, I_n$ ne peut être traversé que dans une unique direction.*

Remarque. Un système DCPM de dimension 2 est un pseudo-système DCPM planaire.

Le résultat suivant est alors prouvé dans [9] :

Lemme 8.3 (Trajectoires d'un pseudo-système DCPM planaire [9]) *Soit $H = (X, f)$ un pseudo-système DCPM planaire.*

Toute trajectoire de H possède une signature ultimement cyclique du type

$$R_1 \ldots R_p (R_{p+1} \ldots R_{p+q})^\omega$$

pour $p, q \in \mathbb{N}$.

Nous prouvons alors :

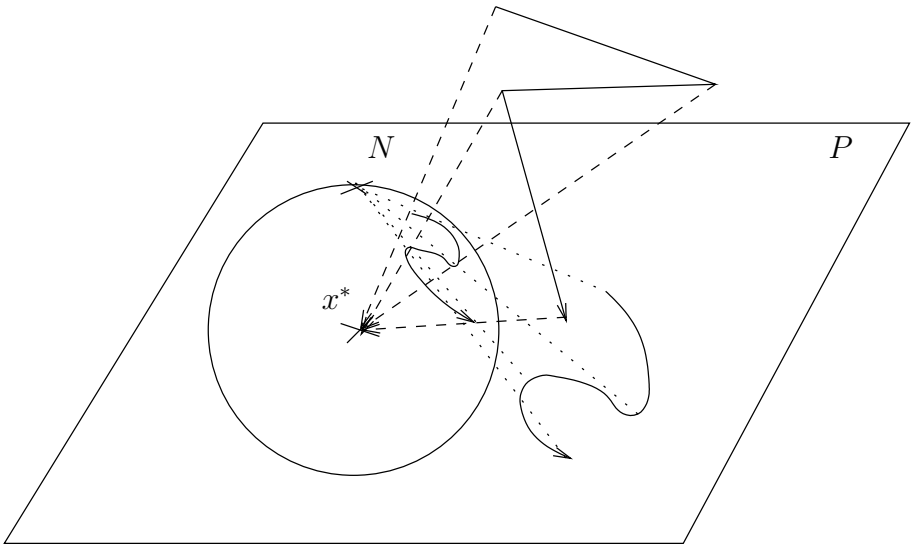
Proposition 8.2 *Soit $H = (X, f)$ un système DCPM de dimension d. Soit ϕ une trajectoire de temps continu fini et de temps discret ω telle que la position atteinte par la trajectoire au temps discret ω soit de dimension locale $d' \leq 3$.*

Alors la signature de ϕ est ultimement cyclique du type $R_1 \ldots R_p (R_{p+1} \ldots R_{p+q})^\omega$ pour $p, q \in \mathbb{N}$.

fig :balldim3Illustration de la preuve de la proposition 8.2

Preuve: Par l'observation 8.4, nous pouvons supposer $d = d'$. Le cas $d = d' = 2$ découle immédiatement du lemme précédent. Supposons donc $d = d' = 3$. Notons x^* la position atteinte par la trajectoire au temps discret ω. Soit $B(x^*, \epsilon)$ la boule donnée par l'observation 8.3. Puisque la fonction ϕ est continue, il existe un temps $t^0 < t^*$ avec $\phi(t) \in B(x^*, \epsilon)$ pour tout $t \in [t^0, t^*]$. Soit S la sphère centrée en x^* de rayon ϵ. Appelons θ la projection centrale de $\mathbb{R}^3 - \{x^*\}$ sur la sphère S : voir la figure **??**.

La remarque essentielle est la suivante : puisque toute région qui intersecte $B(x^*, \epsilon)$ contient le point x^* dans son adhérence, deux points x et x' de $B(x^*, \epsilon)$ de même image par θ ont même image par f. Par conséquent, l'image par θ de



la restriction de la trajectoire ϕ à $[t^0, t^*[$ est une trajectoire d'un certain système dynamique sur la sphère S : voir la figure **??**.

Pour nous ramener au lemme 8.3 nous allons ajouter un difféomorphisme entre la sphère S et un plan P comme sur la figure **??**. Soit N un point[1] de S non atteint par l' image par θ de la restriction de ϕ à $[t^0, t^*[$. Soient P le plan orthogonal au segment $[x^*, N]$ en x^* et ψ la projection stéréographique qui envoie $S - \{N\}$ sur P. L'image par $\psi \circ \theta$ de la restriction de ϕ à $[t^0, t^*[$ est une trajectoire d'un pseudo-système DCPM planaire sur P. Par conséquent, elle doit avoir une signature ultimement périodique du type $R_1 \dots R_p (R_{p+1} \dots R_{p+q})^\omega$ et ceci doit être aussi le cas pour ϕ. □

8.3.2 Décidabilité d'une convergence cyclique

Nous venons de prouver que toute trajectoire convergente vers un point de dimension locale 3 a une signature cyclique. Nous allons maintenant prouver qu'il est possible de calculer la limite d'une trajectoire de signature cyclique. Nous en déduirons ensuite qu'il est possible de simuler toutes les trajectoires d'un système DCPM de dimension 3.

Nous avons besoin des résultats suivants faciles à établir :

Observation 8.7 *Le problème de décision suivant est décidable*[2]

[1] Un tel point existe car sinon $\theta(\phi)$ réaliserait un homéomorphisme entre un intervalle de $\mathbb{R}$ et la sphère S et il est connu que la sphère de $\mathbb{R}^2$ est isomorphe à aucun segment de $\mathbb{R}$: voir [70].

[2] Lorsque $P(I_1, \dots, I_k)$ est un problème de décision dont les instances peuvent être des réels, des mots ou des entiers, le problème P est dit *décidable* si la fonction caractéristique de cette propriété est calculable dans le sens de la définition 8.7.

- *Instance : un point $x_0 \in \mathbb{R}^2$, une matrice A 2×2 à coefficients rationnels.*
- *Question : la suite définie par $x_{n+1} = Ax_n$ pour tout $n \in \mathbb{N}$ converge-t'elle vers 0 ?*

Observation 8.8 *Lorsque x_0, A est une instance positive de l'observation 8.7, la série*
$\sum_{i=0}^{\infty} A^i x_0$ est nécessairement convergente et de limite calculable.

Observation 8.9 *Les problèmes de décision suivants sont décidables :*

- *Instance :*
  - *Une instance positive x_0, A de l'observation 8.7.*
  - *Un polyèdre rationnel Q.*
- *Question 1 : Existe-t'il un entier n tel que l'on ait $A^n x_0 \in Q$?*
- *Question 2 : Existe-t'il un entier n tel que l'on ait $\sum_{i=1}^{n} A^i x_0 \in Q$?*

Nous pouvons alors prouver :

Lemme 8.4 *Soit $H = (X, f)$ un système DCPM de dimension d.*
Soit $d' \leq 3$ un entier.
Le problème de décision suivant est décidable :

- *Instance :*
  - *un point $x_0 \in \mathbb{R}^3$*
  - *un nombre fini de régions $R_1, \ldots, R_n$ de H.*
- *Question : La trajectoire ϕ partant de x_0 au temps 0 vérifie-t'elle simultanément les assertions suivantes ?*
  1. *Il existe un réel $t^* > 0$ et un point $x^* = \phi(t^*)$ de dimension locale d' tel que la restriction de ϕ à l'intervalle $[0, t^*]$ soit de temps discret ω.*
  2. *La restriction de ϕ à l'intervalle $[0, t^*[$ a pour signature $(R_1, \ldots, R_n)^\omega$.*
  3. *$\phi(t) \in B(x^*, \epsilon)$ pour tout $t \in [0, t^*]$ où $B(x^*, \epsilon)$ dénote la boule donnée par l'observation 8.3 pour le point x^*.*

En outre, la fonction qui à une instance $x_0, R_1, \ldots, R_n$ positive de ce problème associe x^* est calculable.

Preuve: Soit Δ l'intersection des adhérences des régions R_i. Le point x^*, s'il existe, doit nécessairement appartenir à Δ et la dimension d_Δ de Δ doit vérifier $d_\Delta \geq d - d'$. Posons $d'_\Delta = d - d_\Delta - 1$. Soit H un hyperplan affine de $\mathbb{R}^d$ contenant x_0 et Δ. Fixons une base $B = (x, e_1, \ldots, e_d)$ de $\mathbb{R}^d$ telle que $B_\Delta = (x, e_1, \ldots, e_{d_\Delta})$ soit une base affine de Δ et $B_H = (x, e_1, \ldots, e_{d_\Delta}, e_{d_\Delta+2}, \ldots, e_d)$ soit une base affine de H. Lorsque la signature de la trajectoire partant d'un point x de H débute par $(R_1, \ldots, R_n)$, notons x^+ la prochaine intersection de la trajectoire avec H. Puisque chacune des régions R_i contient le polyèdre Δ dans son adhérence, un raisonnement similaire à l'observation 8.4 montre qu'il

doit exister une application linéaire rationnelle calculable $A : \mathbb{R}^{d'_\Delta} \to \mathbb{R}^{d'_\Delta}$ et des applications linéaires rationnelles calculables $B_1, \ldots, B_{d_\Delta} : \mathbb{R}^{d'_\Delta} \to \mathbb{R}$ telles que l'on ait pour tout pour tout $x \in H$ $(x^+_{d-d'_\Delta}, \ldots, x^+_d) = A(x_{d-d'_\Delta}, \ldots, x_d)$ et pour tout $1 \leq i \leq d_\Delta$ $x^+_i = x_i + B_i(x_{d-d'_\Delta}, \ldots, x_d)$ où $(x_1, \ldots, x_{d_\Delta}, x_{d_\Delta+2}, \ldots, x_d)$ et $(x^+_1, \ldots, x^+_{d_\Delta}, x^+_{d_\Delta+2}, \ldots, x^+_d)$ sont les coordonnées de x et de x^+ dans la base B_H. Si x_0 a pour coordonnées $(y_1, \ldots, y_{d_\Delta}, y_{d_\Delta+1}, \ldots, y_d)$ dans cette base, la suite des itérés de A en $(y_{d-d'_\Delta}, \ldots, y_d)$ doit être convergente. Lorsque tel est le cas, le point x^* ne peut être que le point de Δ de coordonnées $(x^*_1, \ldots, x^*_{d_\Delta})$ dans la base B_Δ avec $x^*_i = y_i + \sum_{i=1}^{\infty} B_i A^i(y_{d-d'_\Delta}, \ldots, y_d)$. Maintenant, le sous-ensemble Q des points x de H tels que la trajectoire partant de x ne débute pas par $R_1, \ldots, R_n$ ou quitte la boule $B(x^*, \epsilon)$ avant d'atteindre x^+ est un polyèdre calculable.

Le problème se ramène à décider si la suite des itérés de A en $(y_{d-d'_\Delta}, \ldots, y_d)$ est convergente, et à tester si la suite définie par $x_{i+1} = x^+_i$ pour tout i atteint le polyèdre Q. Ces problèmes sont décidables par les observations précédentes. □

8.3.3 Résultat de majoration

Nous pouvons alors prouver :

Théorème 8.3 *Tout langage $L \subset \Sigma^*$ semi-reconnu par un système DCPM de dimension 3 est récursivement énumérable.*

Preuve: Appelons $BienDefini(H)$ le sous-ensemble des points de l'espace du système DCPM qui sont bien définis : voir la définition 2.12 du chapitre 2. Rappelons que ce sous-ensemble est un polyèdre. Nous pouvons supposer sans perte de généralité que ce polyèdre est une région du système DCPM. Le langage L est semi-reconnu par la machine de Turing M suivante : en un mot $w \in \Sigma^*$, M pose $i = 0$, $x_0 = (\Xi(w), 0, \ldots, 0)$ et détermine la région R_0 qui contient x_0. Ensuite, M évolue de la façon suivante.

1. M accepte si la trajectoire partant de x_i atteint le point d'acceptation du système DCPM à un temps discret inférieur ou égal à 1.
2. M s'arrête sans accepter si la trajectoire partant de x_i atteint un point du complémentaire de $BienDefini(H)$ à un temps discret inférieur ou égal à 1.
3. Sinon M incrémente i et définit x_{i+1} comme la position au temps discret 1 de la trajectoire partant de x_i au temps 0. M définit R_{i+1} comme la région qui contient x_{i+1}.
4. M vérifie s'il existe un entier $1 \leq p \leq i$ avec $R_p = R_{i+1}$. Si tel est le cas, M utilise l'algorithme qui décide le problème du lemme 8.4 pour l'instance donnée par le point x_{i+1} et par les régions $R_p, \ldots, R_i$. Si cet algorithme répond positivement, M effectue les opérations suivantes :

(a) Soit x^* le point calculable donné par le lemme 8.4 pour cette instance et soit $B(x^*, \epsilon)$ la boule donnée par l'observation 8.3 pour ce point.

(b) Si x^* est le point d'acceptation du système DCPM, M accepte.

(c) Sinon, si $B(x^*, \epsilon)$ ne contient pas le point d'acceptation du système, M pose $i = 0$, $x_0 = x^*$ et définit R_0 comme la région qui contient x^*

5. M retourne dans l'étape 1.

Chacune des positions $x_i = \phi(t_i)$ calculées par l'algorithme est une position atteinte par le calcul ϕ du système DCPM sur le mot w. Il suffit de prouver que la suite croissante t_i, $i \in \mathbb{N}$, n'est pas majorée. Supposons par l'absurde que $t^* = \sup_{i\in\mathbb{N}} t_i$ soit fini. Posons $x^* = \phi(t^*)$. Soit $B(x^*, \epsilon)$ la boule donnée par l'observation 8.3. Le point x^* doit être de dimension locale $d' = 2$ ou 3. Par continuité de ϕ et par la proposition 8.2, il doit exister $t^0 < t^*$ tel que la signature de la restriction de ϕ à $[t^0, t^*]$ soit périodique et tel que l'on ait $\phi(t) \in B(x^*, \epsilon)$ pour tout $t \in [t^0, t^*]$. Par définition de t^*, nous pouvons supposer $t^0 = t_{i_0}$ pour $i_0 \in \mathbb{N}$. Nous obtenons une contradiction car nous avons alors nécessairement $\phi(t_{i_0+1}) = x^*$ et $t_{i_0} + 2 > t^*$. □

8.4 Etude des systèmes DCPM de dimension d

Nous étudions maintenant les systèmes DCPM de dimension quelconque. Dans la section 8.4.1, nous introduisons la notion d' *échantillonnage d'une trajectoire*. Dans la section 8.4.2, nous prouvons qu'il est possible à partir d'un échantillonnage de dimension d' de construire un échantillonnage de dimension $d' + 1$. Dans la section 8.4.3, nous améliorons ce résultat en prouvant qu'il est possible à partir d'un échantillonnage de dimension d' de construire un échantillonnage de dimension $d' + 2$. Enfin dans la section 8.4.4, nous en déduisons une caractérisation de la puissance de calcul des systèmes DCPM.

8.4.1 Echantillonnage d'une trajectoire

La construction effectuée dans la preuve du théorème 8.3 nous invite à poser la définition suivante :

Définition 8.10 (Echantillonnage d'un système DCPM) *Soit H un système DCPM de dimension d. Notons $\mathcal{P}$ le sous-ensemble des mots de Σ^* qui codent un polyèdre rationnel.*

Nous appelons échantillonnage de H *une fonction Echant* : $\mathbb{R}^d \times \mathbb{R} \times \mathcal{P} \times \mathbb{N} \to \mathbb{R}^d \times \mathbb{R}$ *telle que pour tout point x_0 de $\mathbb{R}^d$, pour tout $t_0 \in \mathbb{R}$ et pour tout polyèdre rationnel Q, lorsque x_0, t_0 et Q sont fixés on a les propriétés suivantes :*

1. $Echant_{x,t_0,Q}(0) = (x_0, t_0)$.

2. *Pour tout* $n \in \mathbb{N}$, $Echant_{x_0,t_0,Q}(n)$ *s'écrit* (x_n, t_n) *où* x_n *est un point atteint au temps* t_n *par la trajectoire* ϕ *partant de* x_0 *au temps* t_0.
3. *La suite* $(t_n)_{n\in\mathbb{N}}$ *vérifie :*
   - $t_n < t_{n+1}$ *pour tout* n *s'il n'existe pas d'entier* n_0 *avec* $x_{n_0} \in Q$
   - $t_n < t_{n+1}$ *pour* $n < n_0$ *et* $t_n = t_{n_0}$ *pour tout* $n \geq n_0$ *si* n_0 *est le plus petit entier avec* $x_{n_0} \in Q$.

L'échantillonnage est dit Zénon *en* x_0, t_0, Q *si la suite* $(t_n)_{n\in\mathbb{N}}$ *correspondante est majorée. Dans ce cas, puisque toute trajectoire définie sur un intervalle semi-ouvert s'étend en une trajectoire définie sur l'adhérence de cet intervalle (voir la proposition 2.1 du chapitre 2), on peut supposer* ϕ *définie en* $t^* = sup_{n\in\mathbb{N}} t_n$. *On définit alors* la limite de l'échantillonnage *en* x_0, t_0, Q *comme le couple* (x^*, t^*) *avec* $t^* = sup_{n\in\mathbb{N}} t_n$ *et* $x^* = \phi(t^*)$.

Nous définissons alors :

Définition 8.11 (Dimension d'un échantillonnage) *Soient* H *un système DCPM de dimension* d *et* d' *un entier.*

Un échantillonnage de H *est dit de* dimension d' *si pour tout point* x_0, *pour tout réel* t_0 *et pour tout polyèdre* Q *l'une des deux conditions suivantes est vérifiée :*

1. *l'échantillonnage n'est pas Zénon en* x_0, t_0, Q
2. *l'échantillonnage est Zénon en* x_0, t_0, Q *et la limite* (x^*, t^*) *de l'échantillonnage en* x_0, t_0, Q *est telle que* x^* *soit de dimension locale supérieure ou égale à* $d' + 1$.

On a alors :

Proposition 8.3 *Soit* H *un système DCPM de dimension* d.

Il existe un échantillonnage de H *de dimension* 3 *qui est calculable.*

Preuve: Etant donné $x_0 \in \mathbb{R}^d$, $t_0 \in \mathbb{R}$ et $Q \in \mathcal{P}$ considérons l'algorithme donné par le théorème 8.3 qui part avec $x_0 = 0$ et $t_0 = 0$ mais où le point d'acceptation est remplacé par le polyèdre Q. La suite des positions calculées par l'algorithme donne un échantillonnage de H. Le raisonnement à la fin de la preuve du théorème 8.3 prouve que lorsque celui-ci est Zénon, il ne saurait converger vers un point de dimension locale inférieure ou égale à 3. □

8.4.2 Itération d'un échantillonnage

Nous prouvons maintenant qu'il est possible à partir d'un échantillonnage de dimension d' de construire un échantillonnage de dimension $d' + 1$. Le principe consiste simplement à prouver que l'échantillonnage obtenu en itérant des passages à la limite sur un échantillonnage donné est un échantillonnage de dimension strictement supérieure.

Formellement, nous définissons :

Définition 8.12 (Itération d'un échantillonnage) *Soient H un système DCPM de dimension d et $Echant : \mathbb{R}^d \times \mathbb{R} \times \mathcal{P} \times \mathbb{N} \to \mathbb{R}^d \times \mathbb{R}$ un échantillonnage de H.*

Nous appelons itération de cette échantillonnage *l'échantillonnage $Iter : \mathbb{R}^d \times \mathbb{R} \times \mathcal{P} \times \mathbb{N} \to \mathbb{R}^d \times \mathbb{R}$ défini pour tout point $x_0 \in \mathbb{R}^d$, pour tout $t_0 \in \mathbb{R}$, pour tout polyèdre rationnel Q et pour tout entier n par :*

- *$Iter_{x_0,t_0,Q}(0) = (x_0, t_0)$*
- *$Iter_{x_0,t_0,Q}(n+1)$ est la limite de l'échantillonnage $Echant$ en $(Iter_{x_0,t_0,Q}(n)), Q$ si l'échantillonnage $Echant$ est Zénon en $(Iter_{x_0,t_0,Q}(n)), Q$.*
- *$Iter_{x_0,t_0,Q}(n+1) = Echant_{(Iter_{x_0,t_0,Q}(n),Q)}(n)$ sinon.*

Il est clair que l'itération d'un échantillonnage est un échantillonnage.

Nous prouvons maintenant que, lorsque $Echant$ est de dimension d', alors $Iter$ est de dimension $d' + 1$. Nous allons étudier pour cela la dimension locale de la limite d'une suite de points de dimension locale fixée.

Commençons par la remarque élémentaire suivante :

Observation 8.10 *Soit x^* un point de l'espace d'un système DCPM de dimension d. Soient $B(x^*, \epsilon)$ et Δ_{x^*} la boule et le polyèdre de la définition 8.4 pour ce point. Notons par d' la dimension locale de x^*.*

Alors :

- *Tout point de $B(x^*, \epsilon)$ est de dimension locale inférieure ou égale à d'.*
- *Les points de $B(x^*, \epsilon)$ de dimension locale d' sont exactement les points de $\Delta_{x^*} \cap B(x^*, \epsilon)$.*

Nous prouvons maintenant :

Lemme 8.5 *Soit x^* un point de l'espace d'un système DCPM $H = (X, f)$ de dimension d. Supposons que la dimension locale d' de x^* vérifie $d' < d$. Notons $B(x^*, \epsilon)$, P_{x^*} et Δ_{x^*} la boule, le sous-espace affine et le polyèdre définis dans l'observation 8.4. Soit ϕ une trajectoire de H définie sur un intervalle I avec $\phi(t) \in B(x^*, \epsilon)$ pour tout $t \in I$. Supposons qu'il existe $t^1 < t^2$ dans I tels que $\phi(t^1)$ et $\phi(t^2)$ aient même projection orthogonale sur P_{x^*}.*

Alors la trajectoire ϕ est ultimement cyclique dans le sens suivant : si l'on note δ le vecteur $\vec{\phi(t^1)\phi(t^2)}$ alors pour tout $t \in [t^1, t^2]$ et pour tout entier $n \in \mathbb{N}$ tel que $t + n(t^2 - t^1)$ appartienne à I on a $\phi(t + n(t^2 - t^1)) = \phi(t) + n\delta$.

Preuve: Il suffit d'utiliser l'observation 8.4 en remarquant que lorsque ϕ' est une trajectoire d'un système DCPM $H' = (X', f')$ de dimension d' qui vérifie $\phi'(t^1) = \phi'(t^2)$ pour des temps $t^1 < t^2$, alors pour tout $t \in [t^1, t^2]$ et pour tout entier $n \in \mathbb{N}$ on a $\phi'(t + n(t^2 - t^1)) = \phi'(t)$. □

On en déduit alors :

Lemme 8.6 (Limite de points de dimension locale d') *Soit H un système DCPM de dimension d. Soit ϕ une trajectoire de H de temps continu fini. Soit α le temps discret de ϕ. Supposons que α soit un ordinal limite qui s'écrive $\alpha = sup_{i\in\mathbb{N}}\beta_i$. Supposons qu'il existe un entier d' tel que pour tout i la position atteinte par la trajectoire au temps discret β_i soit de dimension locale supérieure ou égale à d'.*

Alors la position atteinte par la trajectoire au temps discret α est de dimension locale supérieure ou égale à $d'+1$.

Preuve: Pour tout $\beta \leq \alpha$ notons $x_\beta = \phi(t_\beta)$ la position atteinte par la trajectoire au temps discret β. Posons $x^* = x_\alpha$ et $t^* = t_\alpha$. Soit $B(x^*, \epsilon)$ la boule donnée par l'observation 8.3 pour le point x^*. Par continuité de ϕ il doit exister $t^0 < t^*$ tel que l'on ait $\phi(t) \in B(x^*, \epsilon)$ pour tout $t \in [t^0, t^*]$. Puisque t^* est le suprémum des t_{β_i}, $i \in \mathbb{N}$, nous pouvons supposer $t^0 = t_{\beta_{i_0}}$ pour $i_0 \in \mathbb{N}$. L'observation 8.10 nous dit que la dimension locale d'' du point x^* doit être supérieure ou égale à la dimension locale de chacun des x_{β_i} pour $i \geq i_0$. On a donc nécessairement $d'' \geq d'$. Supposons maintenant par l'absurde que l'on ait $d'' = d'$. Chacun des x_{β_i} pour $i \geq i_0$ doit être de dimension locale d'. Si l'on appelle Δ_{x^*} le polyèdre de la définition 8.4, l'observation 8.10 nous dit que chacun des x_{β_i}, pour $i \geq i_0$ doit appartenir au polyèdre Δ_{x^*}. On peut alors utiliser l'observation 8.5 au point x^* avec $t^1 = t_{i_0}$ et avec t^2 défini comme l'infimum des temps $t > t^1$ pour lesquels $\phi(t) \in \Delta_{x^*}$: chacun des t_{β_i} pour $i \geq i_0$ doit alors s'écrire sous la forme $t^1 + n_i(t^2 - t^1)$ pour un certain entier $n_i \in \mathbb{N}$ avec $t^1 + n_i(t^2 - t^1) \in I$. Nous obtenons une contradiction car la suite t_{β_i} est supposée strictement croissante et il y a au plus un nombre fini de points du type $t^1 + n(t^2 - t^1)$, $n \in \mathbb{N}$, dans l'intervalle borné I. □

Corollaire 8.2 (Dimension de l'itération) *Soit H un système DCPM de dimension d. Soit Echant un échantillonnage de H de dimension d'.*

Alors l'itération de cet échantillonnage est un échantillonnage de dimension $d'+1$.

Preuve: Dénotons par $Iter : \mathbb{R}^d \times \mathbb{R} \times \mathcal{P} \times \mathbb{N} \to \mathbb{R}^d \times \mathbb{R}$ l'itération de l'échantillonnage *Echant*. Si *Iter* est Zénon en x_0, t_0, Q alors l'échantillonnage *Echant* doit être Zénon en $Iter_{x_0,t_0,Q}(n), Q$ pour tout n. Par conséquent, pour tout n, $Iter_{x_0,t_0,Q}(n+1)$ doit s'écrire sous la forme (x_{n+1}, t_{n+1}) avec x_{n+1} de dimension locale supérieure ou égale à $d'+1$. Par le lemme 8.6, la limite de l'échantillonnage *Iter* en x_0, t_0, Q doit donc s'écrire sous la forme (x^*, t^*) avec x^* de dimension locale supérieure ou égale à $d'+2$. □

8.4.3 Suppression des cycles

Nous allons maintenant améliorer ce résultat en prouvant, qu'à partir d'un échantillonnage de dimension d', il est aussi possible de construire un échantillonnage de dimension $d'+2$. Le principe consiste à observer de plus près ce qui se

passe lorsqu'une suite de points de dimension locale d' converge vers un point de dimension locale $d'+1$.

Rappelons que *dist* dénote la distance du maximum. Nous commençons par la remarque suivante : observez l'exemple de la figure 8.1.

Observation 8.11 *Soit $H=(X,f)$ un système DCPM de dimension d et soit ϕ une trajectoire de H, définie sur un intervalle I, telle qu'il existe un point $x^* \in X$ et deux réels $t^1 < t^2$ dans I avec :*

1. *$dist(\phi(t^2),x^*) < dist(\phi(t^1),x^*)$*
2. *$\phi(t) \in B(x^*,\epsilon)$ pour tout $t \in [t^1,t^2]$*

où $B(x^*,\epsilon)$ est la boule donnée par l'observation 8.3.

Alors la trajectoire ϕ atteint le point x^* au temps $t^* = t^1 + \sum_{j=0}^{\infty} \lambda^j (t^2-t^1)$ où λ est le réel donné par $dist(\phi(t^2),x^*) = \lambda dist(\phi(t^1),x^*)$.

Posons alors la définition suivante :

Définition 8.13 (Problème Cycle) *Soit $H=(X,f)$ un système DCPM de dimension d.*

Le problème Cycle est le problème de décision suivant :

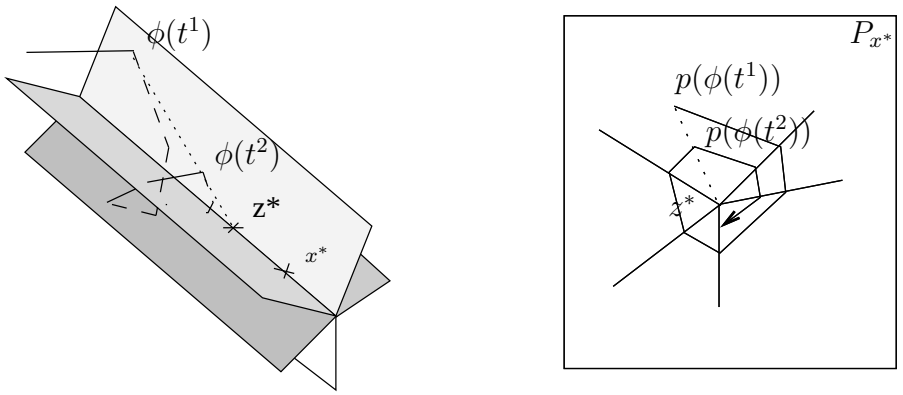
- *Instance :*
  - *Un point $x^* \in \mathbb{Q}^d$.*
  - *Un point x^1 et un réel t^1.*
  - *Un point x^2 et un réel t^2 tel que la trajectoire ϕ partant de x^1 au temps t^1 atteigne le point x^2 au temps t^2 et vérifie $\phi(t) \in B(x^*,\epsilon)$ pour tout $t \in [t^1,t^2]$ où $B(x^*,\epsilon)$ désigne la boule donnée par l'observation 8.3 pour le point x^*.*

  Notons P_{x^*} et Δ_{x^*} le sous-espace affine et le polyèdre définis dans l'observation 8.4 pour le point x^*. Notons p la projection orthogonale de $\mathbb{R}^d$ sur P_{x^*}.
- *Question : Les assertions suivantes sont-elles simultanément vérifiées ?*
  1. *$dist(p(x^2),p(x^*)) < dist(p(x^1),p(x^*))$*
  2. *la droite définie par les deux points x^1 et x^2 intersecte Δ_{x^*} en un point z^**
  3. *z^* appartient à $B(x^*,\epsilon)$.*

Lorsqu'une instance positive $\mathcal{I} = x^*, x^1, t^1, x^2, t^2$ est donnée, le point z^* est appelé le point limite correspondant à l'instance $\mathcal{I}$ *et le réel t^* défini par $t^* = t^1 + \sum_{j=0}^{\infty} \lambda^j (t^2-t^1)$, avec le réel λ donné par $dist(p(x^2),p(x^*)) = \lambda dist(p(x^1),p(x^*))$, est appelé* le temps limite correspondant à l'instance $\mathcal{I}$.

On a alors : voir la figure **??**.

fig :limiteIllustration du corollaire 8.3



Corollaire 8.3 *Soit x^* un point de l'espace d'un système DCPM $H = (X, f)$ de dimension d. Supposons que la dimension locale d' de x^* vérifie $d' < d$. Soit $B(x^*, \epsilon)$ la boule donnée par l'observation 8.3. Soit ϕ une trajectoire de H définie sur un intervalle I et deux réels $t^1 < t^2$ dans I avec $\phi(t) \in B(x^*, \epsilon)$ pour tout $t \in [t^1, t^2]$ tels que l'instance du problème Cycle donnée par $x^*, \phi(t^1), t^1, \phi(t^2), t^2$ soit positive.*

Alors la trajectoire ϕ atteint le point z^* au temps t^* où z^* et t^* sont respectivement le point limite et le temps limite de la définition 8.13 pour l'instance $x^*, \phi(t^1), t^1, \phi(t^2), t^2$.

Preuve: Ce n'est rien d'autre qu'une traduction de l'observation 8.11 via l'observation 8.4. □

Maintenant on peut observer :

Observation 8.12 *Soit x^* un point de l'espace d'un système DCPM de dimension d. Soit $B(x^*, \epsilon)$ la boule donnée par l'observation 8.3 pour ce point.*

Si x^* est de dimension locale d, alors l'ensemble des points de $B(x^*, \epsilon)$ de dimension locale $d-1$ est constitué d'un nombre fini de segments ouverts contenant chacun le point x^* dans leurs adhérences.

On déduit alors l'extension suivante du lemme 8.6 :

Lemme 8.7 (Limite de points de dimension locale d') *Soit H un système DCPM de dimension d. Soit ϕ une trajectoire de H de temps continu fini et α son temps discret. Pour tout $\beta \leq \alpha$, notons $x_\beta = \phi(t_\beta)$ la position atteinte par la trajectoire au temps discret β. Supposons que α soit un ordinal limite qui s'écrive $\alpha = sup_{i \in \mathbb{N}} \beta_i$ et supposons qu'il existe un entier d' tel que pour tout i la position atteinte par la trajectoire au temps discret β_i soit de dimension locale supérieure ou égale à d'.*

Alors la position atteinte par la trajectoire au temps discret α est de dimension locale d'' supérieure ou égale à $d'+1$. En outre, si d'' vaut $d'+1$, alors il doit exister

un point rationnel $x \in \mathbb{Q}^d$ et deux entiers $i \leq i'$ tels que $x, x_{\beta_i}, t_{\beta_i}, x_{\beta_{i'}}, t_{\beta'_i}$ constituent une instance positive du problème Cycle. Dans ce cas, x_α et t_β sont donnés par le point limite et le temps limite correspondant à l'instance $x, x_{\beta_i}, t_{\beta_i}, x_{\beta_{i'}}, t_{\beta'_i}$: voir la définition 8.13.

Preuve: Posons $x^* = x_\alpha$ et $t^* = t_\alpha$. Nous savons par le lemme 8.6 que la dimension locale d'' de x^* doit vérifier $d'' \geq d' + 1$. Supposons $d'' = d' + 1$. Il doit exister un entier i_0 tel que x_{β_i} soit de dimension locale d' pour tout $i \geq i_0$ car sinon nous pourrions extraire de la suite x_{β_i} une suite croissante de points de dimension locale supérieure ou égale à $d' + 1$ et nous obtiendrions une contradiction avec le lemme 8.6 appliqué à cette suite. Soit $B(x^*, \epsilon)$ la boule donnée par l'observation 8.3. Par continuité de ϕ, il doit exister $t^0 < t^*$ avec $\phi(t) \in B(x^*, \epsilon)$ pour tout $t \in [t^0, t^*]$. Puisque t^* s'écrit $t^* = sup_{i \in \mathbb{N}} t_{\beta_i}$, nous pouvons supposer $t^0 = t_{\beta_{i_1}}$ pour $i_1 \geq i_0$. Soient P_{x^*} et Δ_{x^*} le sous-espace affine et le polyèdre définis dans l'observation 8.4. Appelons p la projection orthogonale de $\mathbb{R}^d$ sur P_{x^*}. Nommons H_{x^*} le système DCPM de dimension d'' sur P_{x^*} construit par l'observation 8.4. Chacun des points $p(x_{\beta_i})$ pour $i \geq i_1$ doit être un point de dimension locale d' dans H_{x^*}. L'observation 8.12 nous dit que chacun des points $p(x_{\beta_i})$ pour $i \geq i_1$ doit appartenir à un nombre fini de segments de P_{x^*} contenant $p(x^*)$ dans leurs adhérences. Puisque ces segments sont en nombre fini et qu'il y a un nombre infini de points $p(x_{\beta_i})$ pour $i \geq i_1$, il doit exister deux entiers $i_1 \leq i < i'$ tels que $p(x_{\beta_{i'}})$ et $p(x_{\beta_{i'}})$ appartiennent à un même segment. Nous pouvons supposer sans perte de généralité que $dist(p(x^*), p(x_{\beta_{i'}})) < dist(p(x^*), p(x_{\beta_i}))$. Soit x un point rationnel quelconque de $\Delta_{x^*} \cap B(x^*, \epsilon)$. L'instance donnée par $x, x_{\beta_i}, t_{\beta_i}, x_{\beta_{i'}}, t_{\beta'_i}$ est alors une instance positive du problème *Cycle*. Par le lemme 8.7, le point x^* et le temps t^* sont alors nécessairement donnés par le point limite et le temps limite correspondant à cette instance : voir la définition 8.13. □

Cela nous invite à étendre la notion d'itération d'un échantillonnage en :

Définition 8.14 (Itération sans-cycle d'un échantillonnage) *Soit Echant : $\mathbb{R}^d \times \mathbb{R} \times \mathcal{P} \times \mathbb{N} \to \mathbb{R}^d \times \mathbb{R}$ un échantillonnage d'un système DCPM de dimension d.*

Nous appelons itération sans-cycle de cet échantillonnage *l'échantillonnage $SsCycle : \mathbb{R}^d \times \mathbb{R} \times \mathcal{P} \times \mathbb{N} \to \mathbb{R}^d \times \mathbb{R}$ défini pour tout $x_0 \in \mathbb{R}^d$, pour tout $t_0 \in \mathbb{R}$, pour tout polyèdre rationnel Q et pour tout entier n par (Iter dénote l'itération de l'échantillonnage Echant) :*

1. $SsCycle_{x_0,t_0,Q}(0) = (x_0, t_0)$
2. $SsCycle_{x_0,t_0,Q}(n+1)$ *est le couple (x^*, t^*) de la définition 8.13 pour l'instance y, $(SsCycle_{x_0,t_0,Q}(n))$,$(Iter_{(SsCycle_{x_0,t_0,Q}(n), Q \cup B(y,\epsilon)^c)}(n'))$ s'il existe un point $y \in \mathbb{Q}^d$ et un entier $n' < n$ tels que cette instance soit une instance positive du problème Cycle, où $B(y, \epsilon)^c$ désigne le complémentaire de la boule $B(y, \epsilon)$ donnée par l'observation 8.3 pour le point y.*

3. $SsCycle_{x_0,t_0,Q}(n+1) = Iter_{(SsCycle_{x_0,t_0,Q}(n),Q)}(n)$ *sinon.*

On a alors :

Corollaire 8.4 (Dimension de l'itération sans-cycle) *Soit H un système DCPM de dimension d. Soit un échantillonnage de H de dimension d' pour $1 \leq d' \leq d$.*

L'itération sans-cycle de cet échantillonnage est un échantillonnage de H de dimension $d'+2$.

8.4.4 Résultat de majoration

En partant de l'échantillonnage de la proposition 8.3 et en itérant l'application qui à un échantillonnage associe son itération sans-cycle on obtient immédiatement :

Corollaire 8.5 *Soient H un système DCPM de dimension d et $k \geq 0$ un entier.*

Il existe un échantillonnage de H de dimension $3k+2$.

Nous allons maintenant majorer la puissance de calcul des systèmes DCPM en prouvant que l'échantillonnage du corollaire précédent est calculable avec un oracle du ω^kème niveau de la hiérarchie hyperarithmétique.

Nous proposons la généralisation naturelle suivante de la définition 8.8 :

Définition 8.15 (Langage $[f]$) *Supposons fixée une fonction récursive bijective $<,,>$ de $\mathcal{P} \times \mathbb{N} \times \Sigma^*$ vers Σ^*. Soit $f : \mathcal{P} \times \mathbb{N} \to \mathbb{R}^d$ une fonction.*

Nous notons $[f]$ le langage constitué des mots de Σ^* du type $< Q, n, w >$ tels que w code un polyèdre rationnel P de $\mathbb{R}^d$ contenant $f(Q,n)$.

Pour tout entier k, nous notons $[f]_k$ le langage constitué des mots de Σ^* du type $< Q, n, w >$ tels que n soit un entier inférieur ou égal à k et tels que w code un polyèdre rationnel P de $\mathbb{R}^d$ contenant $f(Q,n)$.

Définition 8.16 (Langage $[f](x)$) *Soit $f : \mathbb{R}^d \times \mathbb{R} \times \mathcal{P} \times \mathbb{N} \to \mathbb{R}^d \times \mathbb{R}$ une fonction. Lorsque x, t sont donnés, notons $f_{(x,t)}$ la fonction définie par $f_{(x,t)}(Q,n) = f(x,t,Q,n)$ pour tout Q, n.*

Pour tout $(x,t) \in \mathbb{R}^{d+1}$ nous notons $[f](x,t)$ pour le langage $[f_{(x,t)}]$.

Pour tout $(x,t) \in \mathbb{R}^{d+1}$ et pour tout entier k nous notons $[f]_k(x,t)$ pour le langage $[f_{(x,t)}]_k$.

La définition 8.12 montre facilement que l'on a :

Observation 8.13 *Soient H un système DCPM de dimension d et Echant : $\mathbb{R}^d \times \mathbb{R} \times \mathcal{P} \times \mathbb{N} \to \mathbb{R}^d \times \mathbb{R}$ un échantillonnage de H. Appelons Iter l'itération de cet échantillonnage.*

Il existe une formule fixe F_1 du premier ordre telle que pour tout $x_0 \in \mathbb{R}^d$, $t_0 \in \mathbb{R}$, $Q \in \mathcal{P}$, $n \in \mathbb{N}$, le langage $[Iter_{x_0,t_0,Q}(n+1)]$ soit défini par la formule F_1 à partir de relations récursives et du langage $[Echant](Iter_{x_0,t_0,Q}(n))$.

Cette formule F_1 est équivalente à une formule existentielle du premier ordre de moins de 3 alternances de quantificateurs.

La proposition 1.2 donne alors :

Corollaire 8.6 *Il existe une fonction récursive $g_1 : \mathbb{N} \to \mathbb{N}$ telle que pour tout oracle $X \subset \Sigma^*$ et pour tout entier u, si la machine de Turing avec oracle X de numéro u reconnaît pleinement $[Echant](Iter_{x_0,t_0,Q}(n))$, alors la machine de Turing de numéro $g_1(u)$ avec oracle $X^{(4)}$ reconnaît pleinement $[Iter_{x_0,t_0,Q}(n+1)]$*

Lorsque u est une notation dans le système O d'un ordinal récursif, notons $|u|$ l'ordinal dénoté par u. De la proposition 1.7 on déduit :

Corollaire 8.7 *Soit z une notation dans le système O d'un ordinal récursif.*

Il existe une fonction récursive $g_2 : \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ et une fonction récursive g_3 des entiers vers les notations dans le système O telles que, pour tout oracle $X \subset \Sigma^*$, pour tout $x_0 \in \mathbb{R}^d$, pour tout $t_0 \in \mathbb{R}$, pour tout entier n et pour tout entier u, si la machine de Turing avec oracle $X^{(z)}$ de numéro u reconnaît pleinement $[Echant](x_0, t_0)$, alors la machine de Turing numéro $g_2(n, u)$ avec oracle $X^{(g_3(n))}$ reconnaît pleinement $[Iter]_n(x_0, t_0)$.

La fonction g_3 vérifie $|g_3(n)| = (|z| + 4)n$ pour tout n.

La définition 8.14 montre que l'on a :

Observation 8.14 *Soient H un système DCPM de dimension d et $Echant$: $\mathbb{R}^d \times \mathbb{R} \times \mathcal{P} \times \mathbb{N} \to \mathbb{R}^d \times \mathbb{R}$ un échantillonnage de H. Appelons $Iter$ l'itération de cet échantillonnage et $SsCycle$ l'itération sans-cycle de cet échantillonnage.*

Il existe une formule fixe F_2 du premier ordre telle que pour tout $x_0 \in \mathbb{R}^d$, $t_0 \in \mathbb{R}$, $Q \in \mathcal{P}$, $n \in \mathbb{N}$, le langage $[SsCycle_{x_0,t_0,Q}(n+1)]$ soit défini par la formule F_2 à partir de relations récursives et du langage $[Iter]_n(SsCycle_{x_0,t_0,Q}(n))$.

Cette formule F_2 est équivalente à une formule existentielle de moins de 3 alternances de quantificateurs.

La proposition 1.2 du chapitre 1 donne alors :

Corollaire 8.8 *Il existe une fonction récursive $g_4 : \mathbb{N} \to \mathbb{N}$ telle que, pour tout oracle $X \subset \Sigma^*$, pour tout $x_0 \in \mathbb{R}^d$, pour tout $t_0 \in \mathbb{R}$ et pour tout entier u, si la machine de Turing avec oracle X de numéro u reconnaît pleinement $[Iter]_n[SsCycle_{x_0,t_0,Q}(n)]$, alors la machine de Turing numéro $g_4(u)$ avec oracle $X^{(4)}$ reconnaît pleinement $[SsCycle_{x_0,t_0,Q}(n+1)]$*

Par la proposition 1.7 du chapitre 1 on déduit :

Corollaire 8.9 *Soit z une notation dans le système O d'un ordinal récursif.*

Il existe une fonction récursive $g_5 : \mathbb{N} \to \mathbb{N}$ et une notation z' dans le système O d'un ordinal récursif telles que, pour tout oracle $X \subset \Sigma^*$, pour tout $x_0 \in \mathbb{R}^d$, pour tout $t_0 \in \mathbb{R}$, pour tout entier n et pour tout entier u, si la machine de Turing avec oracle $X^{(z)}$ de numéro u reconnaît pleinement $[Echant](x_0, t_0)$, alors la machine de Turing de numéro $g_5(u)$ avec oracle $X^{(z')}$ reconnaît pleinement $[SsCycle](x_0, t_0)$.

La notation z' vérifie $|z'| = (|z| + 4)\omega$.

Par une récurrence immédiate sur k, le corollaire précédent montre que les échantillonnages construits dans le corollaire 8.5 sont tels que :

Corollaire 8.10 *Soient H un système DCPM de dimension d et $k \geq 0$ un entier.*

Il existe un échantillonnage $Echant$ de H de dimension $3 + 2k$, une fonction récursive $g : \mathbb{N} \to \mathbb{N}$ et une notation z dans le système O de l'ordinal récursif ω^k tels que, pour tout oracle $X \subset \Sigma^*$, pour tout $x_0 \in \mathbb{R}^d$, pour tout $t_0 \in \mathbb{R}$, pour tout entier n et pour tout entier u, si la machine de Turing avec oracle X de numéro u reconnaît pleinement $[(x_0, t_0)]$, alors la machine de Turing de numéro $g(u)$ avec oracle $X^{(z)}$ reconnaît pleinement $[Echant](x_0, t_0)$.

D'où nous déduisons :

Théorème 8.4 *Soit L un langage semi-reconnu par un système DCPM de dimension $d = 3 + 2k$ pour $k \geq 0$.*

Alors L est dans Σ_{ω^k}.

Soit L un langage semi-reconnu par un système DCPM de dimension $d = 4 + 2k$ pour $k \geq 0$.

Alors L est dans Σ_{ω^k+1}.

Preuve: Soit $Echant : \mathbb{R}^d \times \mathbb{R} \times \mathcal{P} \times \mathbb{N} \to \mathbb{R}^d \times \mathbb{R}$ l'échantillonnage de dimension $3 + 2k$ du corollaire précédent. Soit x^1 le point d'acceptation du système DCPM. Soit z la notation dans le système O de l'ordinal ω^k et $g : \mathbb{N} \to \mathbb{N}$ la fonction récursive du corollaire précédent pour cet entier k.

Supposons $d = 3 + 2k$. Le langage L est semi-reconnu par la machine avec oracle $\emptyset^{(z)}$ qui sur un mot $w \in \Sigma^*$, calcule le numéro u d'une machine de Turing qui reconnaît le langage $[x_w]$ défini par $x_w = (\Xi(w), 0, \ldots, 0)$, puis simule alors la machine de Turing avec oracle $\emptyset^{(z)}$ de numéro $g(u)$ de façon à tester s'il existe un entier n avec $Echant(x_w, 0, x^1, n) = x^1$.

Supposons $d = 4 + 2k$. Etant donné un mot $w \in \Sigma^*$, une machine de Turing avec oracle $\emptyset^{(z)}$ peut reconnaître pleinement le langage $[Echant](x_w, 0)$: il lui suffit comme dans le paragraphe précédent de calculer le numéro u d'une machine de Turing qui reconnaît le langage $[x_w]$ puis de simuler la machine de Turing avec oracle $\emptyset^{(z)}$ de numéro $g(u)$. En utilisant la proposition 1.2, il nous suffit de prouver

qu'il existe une formule existentielle fixe du premier ordre avec une alternance de quantificateurs, définie à l'aide de la relation $[Echant](x_w, 0)$, qui soit vraie si et seulement si w appartient à L.

Il suffit d'écrire qu'un mot w appartient à L si et seulement s'il existe un entier n tel que l'on ait $Echant(x_w, 0, x^1, n) = x^1$ ou s'il existe deux rationnels $t^0 < t^1$, un entier n_0 et un point de coordonnées rationnelles x^*, choisi parmi une liste finie de points à coordonnées rationnelles, tels que l'on ait $t_n < t^1$ pour tout n et $t_n \in B(x^*, \epsilon)$ pour tout $n \geq n_0$.

En effet, le mot w appartient à L si et seulement s'il existe un entier n tel que l'on ait $Echant(x_w, 0, x^1, n) = x^1$ ou si l'échantillonnage $Echant$ est Zénon en $x_w, 0, x^1$ et converge vers un point x^* de dimension locale d tel que la trajectoire partant de ce point atteigne le point x^1. L'échantillonnage est Zénon en $x_w, 0, x^1$ si et seulement s'il existe un rationnel t^1 avec $t_n < t^1$ pour tout n. Il y a un nombre fini de points de dimension locale d dans un système de dimension d et ceux-ci ont des coordonnées rationnelles. Maintenant étant donné un de ces points x^* de dimension locale d, si l'on appelle $B(x^*, \epsilon)$ la boule donnée par l'observation 8.3, l'échantillonnage $Echant$ converge vers x^* si et seulement s'il existe un rationnel $t^0 < t^1$ et un entier n_0 tel que l'on a $t_n \in B(x^*, \epsilon)$ pour tout $n \geq n_0$: il est clair que si la trajectoire converge vers x^* alors cette propriété doit être vérifiée. Réciproquement lorsqu'il existe un rationnel $t^0 < t^1$ et un entier n_0 tel que l'on ait $t_n \in B(x^*, \epsilon)$ pour tout $n \geq n_0$, on déduit que l'échantillonnage ne peut converger que vers un point de $B(x^*, \epsilon)$, qui ne peut être que le point x^* puisque l'échantillonnage est de dimension $2k + 3$ et puisque x^* est le seul point de dimension locale $d = 2k + 3$ dans $B(x^*, \epsilon)$. □

Du théorème 7.7 du chapitre précédent nous déduisons :

Corollaire 8.11 *Les langages semi-reconnus par les systèmes DCPM de dimension d sont exactement les langages de*

- Σ_{ω^k} *pour* $d = 2k + 3$, $k \geq 0$.
- Σ_{ω^k+1} *pour* $d = 2k + 4$, $k \geq 0$.

Conclusion

Dans cette thèse, nous avons présenté une étude de la complexité algorithmique des systèmes dynamiques à espace continu. Nous nous sommes particulièrement intéressés aux classes suivantes de systèmes dynamiques : les systèmes affines par morceaux, les systèmes linéaires commutés, les réseaux de neurones et les systèmes à dérivée constante par morceaux (DCPM). Après avoir introduit formellement chacune de ces classes, nous avons caractérisé les propriétés qui permettent de garantir l'existence et l'unicité des trajectoires dans un système DCPM. Cette caractérisation, nouvelle à notre connaissance, permet de légitimer formellement les systèmes DCPM tels que proposés dans [9].

Nous nous sommes ensuite intéressés à la décidabilité de la vérification automatique de propriétés pour les systèmes dynamiques. Lorsque nous avons débuté ce travail, il était connu que ces systèmes pouvaient simuler les machines de Turing [9, 45, 60, 61, 80]. Par conséquent, il était clair que toute propriété *locale* de ces systèmes, comme les propriétés d'atteignabilité, était indécidable pour ces systèmes. Toutefois, la décidabilité de propriétés *globales* comme les propriétés de stabilité était une question ouverte, mentionnée à plusieurs reprises dans la littérature : voir [12, 83]. Dans cette thèse, nous avons répondu à cette question en prouvant que l'on ne peut pas décider la stabilité des systèmes dynamiques. Plus précisément, nous avons proposé trois formulations possibles de la stabilité (mortalité, stabilité globale et stabilité globale asymptotique), et nous avons prouvé, que pour chacune de ces notions, il n'est pas possible de décider si un système dynamique affine par morceaux ou un réseau de neurones est stable.

Nous avons ensuite étudié la frontière entre décidabilité et indécidabilité pour les systèmes dynamiques de basses dimensions, en présentant une synthèse des résultats connus à ce jour. Nous avons mis en évidence que la décidabilité du problème de l'atteignabilité pour les systèmes affines par morceaux de dimension 1, de la contrôlabilité pour les systèmes linéaires commutés de dimension 2, ou de la stabilité pour les systèmes affines par morceaux continus (resp. discontinus) de dimension 2 et 3 (resp. 1) est une question ouverte. Nous avons en particulier développé plus longuement le problème de la contrôlabilité des systèmes linéaires commutés de dimension 2 : nous avons prouvé que ce problème est exactement le problème de la mortalité des matrices 2×2, problème qui est ouvert depuis de nombreuses années [75]. Nous avons dressé un panorama des résultats connus sur

ce problème et proposé quelques extensions : nous avons prouvé que le problème de la mortalité pour 9 matrices 3×3 ou pour 2 matrices 27×27 est indécidable, et que le nombre de matrices rendant le problème de la mortalité pour les matrices 3×3 indécidable se relie au nombre de règles rendant le problème de la correspondance de Post indécidable. Nous avons prouvé l'indécidabilité dans le modèle de Blum Shub et Smale du problème pour les matrices réelles 2×2, et la décidabilité du problème pour 2 matrices 2×2 pour la calculabilité classique. Par conséquent, nous avons prouvé que le problème de la mortalité pour les matrices 2×2 ne peut être résolu en utilisant uniquement des opérations arithmétiques, et donc relève de la théorie des nombres. Nous avons ensuite souligné la difficulté de ce problème en prouvant qu'il est équivalent à d'autres problèmes mentionnés comme ouverts dans la littérature, comme le problème de l'égalité des coefficients étudié par [48], ou le problème du zéro en haut dans un coin étudié par [29, 54]. Nous l'avons en outre relié au problème de l'atteignabilité pour les systèmes affines par morceaux de dimension 1.

Motivés par le fait que la plupart des méthodes algorithmiques de vérification manipulent des polyèdres, nous avons étudié la représentation des polyèdres orthogonaux et temporisés par leurs sommets.

Nous avons proposé trois représentations des polyèdres orthogonaux : représentation par les couleurs, représentation par les voisinages, et représentation par les sommets extrêmes. Nous avons prouvé la validité de ces représentations pour les polyèdres convexes et non-convexes de dimension quelconque. En particulier, notre représentation par les sommets extrêmes est une extension à la dimension quelconque de la représentation proposée par Aguilera et Ayala dans [1, 2, 3] pour les dimensions 1, 2 et 3. Nous avons prouvé que, lorsque la dimension d est fixée, et lorsque n est le nombre de sommets, on peut déterminer la couleur d'un point en temps $O(n^d)$ pour la représentation par les couleurs et en temps $O(n \log n)$ pour les deux autres représentations. Nous avons en outre proposé des algorithmes pour réaliser les opérations spécifiques requises par les algorithmes de vérification : comparaisons entre polyèdres, détections des faces d'un polyèdre, opérations booléennes entre polyèdres.

Nous avons ensuite généralisé ces représentations aux polyèdres temporisés. Nous avons proposé deux représentations : la représentation par les simplexes de la boîte et la représentation par les simplexes du voisinages. Nous avons prouvé que ces représentations, valides pour les polyèdres convexes et non-convexes de dimension quelconque, permettent, lorsque la dimension d est fixée, et lorsque n est le nombre de sommets du polyèdre, de déterminer la couleur d'un point en temps $O(n^d)$ pour la représentation par les simplexes de la boîte, et en temps $O(n \log n)$ pour la représentation par les simplexes du voisinage. Nous avons ensuite esquissé des algorithmes pour réaliser les opérations spécifiques requises par les algorithmes de vérification sur ces polyèdres : comparaisons entre polyèdres, opérations "passage du temps", opérations booléennes entre polyèdres.

Nous nous sommes ensuite intéressés à l'étude de la puissance de calcul des

modèles à temps continu, par l'intermédiaire d'une caractérisation de la puissance de calcul des systèmes à dérivée constante par morceaux. Nous avons caractérisé la puissance de calcul des systèmes à dérivée constante par morceaux de dimension d comme Σ_{ω^k} lorsque $d = 2k + 3, k \geq 0$, et Σ_{ω^k+1} lorsque $d = 2k + 4, k \geq 0$.

Cherchons maintenant à interpréter tous ces résultats : du point de vue de la vérification, nos résultats, exception faite des résultats des chapitres 5 et 6, sont essentiellement négatifs. En effet, nos résultats impliquent que la vérification de la plupart des propriétés des systèmes dynamiques mène à des problèmes indécidables, même pour des systèmes de très basses dimensions. Par exemple, il n'est pas possible de décider si un système affine par morceaux est stable, même en dimension 2. En outre, nos résultats montrent que ces problèmes indécidables sont relativement hauts dans les hiérarchies arithmétiques et hyperarithmétiques : la vérification de propriétés pour un système à dérivée constante par morceaux de dimension d est Σ_{ω^k}-complet si $d = 2k + 3, k \geq 0$, et Σ_{ω^k+1}-complet si $d = 2k + 4, k \geq 0$. Autrement dit, par exemple, vérifier une propriété d'un système à dérivée constante par morceaux de dimension 5 est plus difficile que de vérifier la satisfaction d'une formule arithmétique arbitraire, ou encore, vérifier une propriété d'un système à dérivée constante par morceaux de dimension 4 ne peut être résolu par aucun algorithme ni même par aucun semi-algorithme.

D'un point de vue plus pratique, nous avons proposé des représentations des polyèdres orthogonaux et temporisés qui doivent permettre des implémentations efficaces des méthodes algorithmiques de vérification.

Du point de vue des modèles réels, nos résultats contribuent à mettre en évidence la non-trivialité des propriétés calculatoires des systèmes dynamiques et hybrides. Les résultats du chapitre 3 prouvent que les systèmes dynamiques ont une puissance supérieure au calculable. Les résultats du chapitre 4 prouvent que cela reste vrai même pour des systèmes de très basses dimensions, et que la question de savoir si cela est le cas pour les dimensions 1 ou 2 est un problème très difficile. Enfin, les résultats des chapitres 7 et 8 ont permis d'obtenir, pour la première fois à notre connaissance, une comparaison entre la puissance de calcul des modèles à temps continu et la puissance des machines de Turing. En l'occurrence, nous avons caractérisé la puissance de calcul des systèmes à dérivée constante par morceaux comme Σ_{ω^k} en dimension $d = 2k + 3, k \geq 0$, et Σ_{ω^k+1} en dimension $d = 2k + 4, k \geq 0$.

Il serait important de poursuivre nos travaux de différentes façons : notre preuve de l'indécidabilité des problèmes de la stabilité posent les questions suivantes : les problèmes de la stabilité sont-ils indécidables pour les réseaux de neurones de dimension fixée ? Les problèmes de la stabilité sont-ils indécidables pour des réseaux de neurones dont les matrices ne contiennent que des coefficients dans $\{-1, 0, 1\}$? Nos preuves utilisent l'équivalence des problèmes de la mortalité, de la stabilité globale et de la stabilité globale asymptotique pour les systèmes construits dans nos réductions. Pour quelles classes de matrices ce cas se produit-il ?

Au niveau de l'étude de la décidabilité de la vérification de propriétés pour les systèmes de basses dimensions, il pourrait être intéressant de poursuivre l'étude des liens qui existent entre les différents problèmes. En particulier, existe-t'il un lien entre le problème de la stabilité en dimension 1 pour les systèmes affines par morceaux et le problème de la mortalité pour les matrices 2×2 ?

Pour les représentations des polyèdres orthogonaux, il reste bien entendu à insérer dans les outils de [33] les algorithmes proposés, actuellement utilisables uniquement dans une application jouet qui manipule des polyèdres aléatoires. Il reste aussi à évaluer en pratique la pertinence de ces représentations sur des exemples réels de vérification.

Pour les polyèdres temporisés, il reste, là aussi bien entendu, à implémenter les algorithmes. Toutefois, nous ne sommes pas encore totalement convaincus qu'il n'est pas possible de construire une représentation qui soit le pendant de la représentation par les sommets extrêmes pour les polyèdres orthogonaux. Peut-on construire une telle représentation ?

En ce qui concerne notre étude de la puissance de calcul des systèmes à dérivée constante par morceaux, nous pensons qu'il est peut être envisageable de construire une théorie de la récursivité pour les fonctions DCPM-calculables similaire à celle qui existe pour les fonctions récursives : il est connu que les fonctions récursives sont les fonctions engendrées par composition, récursion primitive, et mu-récursion à partir des fonctions de base [64, 68]. En effet, le chapitre 7 construit des fonctions DCPM-calculables en utilisant uniquement des compositions, des définitions par cas, des homogénéisations à partir de certaines fonctions de base affines et le chapitre 8 semble prouver que toute fonction DCPM-calculable est de ce type. Une telle théorie de la récursivité permettrait de relier nos résultats à la théorie proposée par Moore dans [62]. Pour atteindre ce but, une autre solution peut être d'étudier le lien qui existe entre les constructions que nous utilisons pour contracter un nombre transfini d'étapes discrètes en un temps borné et celles utilisées par Moore pour contracter un calcul continu en un temps borné : voir [62]. L'intérêt serait d'obtenir à long terme une théorie générale des calculs à temps continu qui engloberait toutes les machines "raisonnables" de calcul à temps continu.

Bibliographie

[1] A. Aguilera. *Orthogonal Polyhedra : Study and Application*. PhD thesis, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Barcelone, Espagne, April 1998.

[2] A. Aguilera and D. Ayala. Orthogonal polyhedra as geometric bounds in constructive solid geometry. In *SMA '97 : Proceedings of the Fourth Symposium on Solid Modeling and Applications*, pages 56–67, Atlanta (GE), USA, 14–16 May 1997.

[3] A. Aguilera and D. Ayala. Domain extension for the extreme vertices model (EVM) and set-membership classification. In *CSG'98. Set-Theoretical Solid Modeling : Techniques and Applications*, pages 33–47, 1998.

[4] R. Alur. Timed automata. In *NATO-ASI Summer School on Verification of Digital and Hybrid Systems*, 1998.

[5] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P. H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1) :3–34, 6 February 1995.

[6] R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho. Hybrid automata : an algorithmic approach to the specification and verification of hybrid systems. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer-Verlag, 1993.

[7] R. Alur and D. L. Dill. Automata for modeling real-time systems. In *Automata, Languages and Programming, 17th International Colloquium*, volume 443 of *Lecture Notes in Computer Science*, pages 322–335. Springer-Verlag, 16–20 July 1990.

[8] E. Asarin and O. Maler. Achilles and the tortoise climbing up the arithmetical hierarchy. *Journal of Computer and System Sciences*, 57(3) :389–398, December 1998.

[9] E. Asarin, O. Maler, and A. Pnueli. Reachability analysis of dynamical systems having piecewise-constant derivatives. *Theoretical Computer Science*, 138(1) :35–65, February 1995.

[10] V. D. Blondel, O. Bournez, P. Koiran, C. Papadimitriou, and J. N. Tsitsiklis. Deciding stability and mortality of piecewise affine dynamical systems.

Technical Report 1999-05, Laboratoire de L'Informatique du Parallélisme, École Normale Supérieure de Lyon, 1999. Soumis.

[11] V. D. Blondel and J. N. Tsitsiklis. When is a pair of matrices mortal ? *Information Processing Letters*, 63(5) :283–286, September 1997.

[12] V. D. Blondel and J. N. Tsitsiklis. Three problems on the decidability and complexity of stability. In V. D. Blondel, E. D. Sontag, M. Vidyasagar, and J. C. Willems, editors, *Open Problems in Mathematical Systems and Control Theory*, pages 45–52. Springer-Verlag, London, 1999.

[13] V.D Blondel and J.N. Tsitsiklis. Complexity of stability and controllability of elementary hybrid systems. *Automatica*, 35(3), 1999.

[14] L. Blum, M. Shub, and 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(1) :1–46, July 1989.

[15] J. Boissonat and O. Devillers. *Géometrie algorithmique*. Ediscience International, 1995.

[16] P. Boldi and S. Vigna. Equality is a jump. *Theoretical Computer Science*, à paraître.

[17] A. Bouajjani and R. Robbana. Verifying omega-regular properties for a subclass of linear hybrid systems. In P. Wolper, editor, *Proceedings of the 7th International Conference On Computer Aided Verification*, volume 939 of *Lecture Notes in Computer Science*, pages 437–450, Liège, Belgique, July 1995. Springer Verlag.

[18] O. Bournez. Some bounds on the computational power of piecewise constant derivative systems. In Pierpaolo Degano, Robert Gorrieri, and Alberto Marchetti-Spaccamela, editors, *Automata, Languages and Programming, 24th International Colloquium*, volume 1256 of *Lecture Notes in Computer Science*, pages 143–153, Bologne, Italie, 7–11 July 1997. Springer-Verlag.

[19] O. Bournez. Achilles and the Tortoise climbing up the hyper-arithmetical hierarchy. *Theoretical Computer Science*, 210(1) :21–71, 6 January 1999.

[20] O. Bournez. Some bounds on the computational power of piecewise constant derivative systems. *Theory of Computing Systems*, à paraître.

[21] O. Bournez and M. Branicky. On the mortality problem for matrices of low dimensions. Technical Report 98-01, Laboratoire VERIMAG, Grenoble, 1998. Soumis.

[22] O. Bournez and M. Branicky. On matrix mortality in low dimensions. In V. D. Blondel, E. D. Sontag, M. Vidyasagar, and J. C. Willems, editors, *Open Problems in Mathematical Systems and Control Theory*. Springer-Verlag, London, 1999.

[23] O. Bournez and M. Cosnard. On the computational power of dynamical systems and hybrid systems. *Theoretical Computer Science*, 168(2) :417–459, November 1996.

[24] O. Bournez, O. Maler, and A. Pnueli. Orthogonal polyhedra : Representation and computation. In *Hybrid Systems : Computation and Control*, Nijmegen, Pays-Bas, 29–31 March 1999.

[25] P. Bourret, J. Reggia, and M. Samuelides. *Réseaux neuronaux*. Tecknea, October 1991.

[26] M. S. Branicky. *Studies in Hybrid Systems : Modeling, Analysis, and Control*. PhD thesis, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA, USA, June 1995.

[27] V. Brattka and P. Herling. Feasible real random access machines. In *2nd Conference on Real Numbers and Computers*, Marseille, 1996.

[28] P. Caspi and O. Maler. Global simulation via differential equations. Communication personnelle.

[29] J. Cassaigne and J. Karhumäki. Examples of undecidable problems for 2-generator matrix semigroups. *Theoretical Computer Science*, 204(1–2) :29–34, September 1998.

[30] E. A. Coddington and N. Levinson. *Theory of Ordinary Differentiel Equations*. McGraw-Hill, April 1972.

[31] F. Cucker, M. Karpinski, P. Koiran, T. Lickteig, and K. Werther. On real Turing machines that toss coins. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on the Theory of Computing*, pages 335–342, Las Vegas, Nevada, 29 May–1 June 1995.

[32] F. Cucker and P. Koiran. Computing over the reals with addition and order : Higher complexity classes. *Journal of Complexity*, 11(3) :358–376, September 1995.

[33] T. Dang and O. Maler. Reachability analysis via face-lifting. In T. A. Henzinger and S. Sastry, editors, *Hybrid Systems : Computation and Control*, volume 1386 of *Lecture Notes in Computer Science*, pages 96–109. Springer-Verlag, 1998.

[34] J. E. Goodman and J. O'Rourke. *Handbook of discrete and computational geometry*. CRC Press, August 1997.

[35] M. R. Greenstreet. Verifying safety properties of differential equations. In R. Alur and T. A. Henzinger, editors, *Proceedings of the Eighth International Conference on Computer Aided Verification CAV*, volume 1102 of *Lecture Notes in Computer Science*, pages 277–287, New Brunswick, NJ, USA, July/August 1996. Springer Verlag.

[36] V. Gupta, T. A. Henzinger, and R. Jagadeesan. Robust timed automata. In O. Maler, editor, *Hybrid and Real Time Systems*, volume 1201 of *Lecture Notes in Computer Science*, pages 331–345. Springer-Verlag, 1997.

[37] T. Harju and J. Karhumaki. Morphisms. In Rozenberg, editor, *Handbook of Formal Languages*. Springer-Verlag, February 1997.

[38] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata ? *Journal of Computer and System Sciences*, 57(1) :94–124, August 1998.

[39] P. K. Hooper. The undecidability of the turing machine immortality problem. *The journal of symbolic logic*, 31(2), June 1966.

[40] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory Languages and Computation*. Addison-Wesley, October 1979.

[41] P. Indyk. Optimal simulation of automata by neural nets. In *12th Annual Symposium on Theoretical Aspects of Computer Science*, volume 900 of *Lecture Notes in Computer Science*, pages 337–348, Munich, Allemagne, 2–4 March 1995. Springer-Verlag.

[42] M. Jirstrand. Nonlinear control system design by quantifier elimination. *Journal of Symbolic Computation*, 24(2) :137–152, August 1997.

[43] Y. Kesten, A. Pnueli, J. Sifakis, and S. Yovine. Integration graphs : A class of decidable hybrid systems. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 179–208. Springer-Verlag, 1993.

[44] P. Koiran. *Puissance de calcul des réseaux de neurones artificiels*. PhD thesis, Ecole Normale Supérieure de Lyon, June 1993.

[45] P. Koiran, M. Cosnard, and M. Garzon. Computability with low-dimensional dynamical systems. *Theoretical Computer Science*, 132(1-2) :113–128, September 1994.

[46] P. Koiran and C. Moore. Closed-form of analytic maps in one and two dimensions can simulate universal Turing machines. *Theoretical Computer Science*, 210(1) :217–223, 6 January 1999.

[47] M. Krom. An unsolvable problem with products of matrices. *Mathematical System Theory*, 14 :335–337, 1981.

[48] M. Krom and M. Krom. Recursive solvability of problems with matrices. *Zwitschr. f. math. Logik und Grundlagen d. Math*, 35 :437–442, 1989.

[49] M. Krom and M. Krom. More on mortality. *American Mathematical Monthly*, 97 :37–38, 1990.

[50] R. P. Kurshan and K. L. McMillan. Analysis of digital circuits through symbolic reduction. *IEEE transactions on computer-aided design*, 10(11) :1356–1371, November 1991.

[51] A. Kurzhanski and I. Vályi. *Ellipsoidal calculus for estimation and control*. Birkhaüser, November 1996.

[52] L. Lipshitz and L. A. Rubel. A differentially algebraic replacement theorem, and analog computability. *Proceedings of the American Mathematical Society*, 99(2) :367–372, February 1987.

[53] D. G. Luenberger. *Introduction to Dynamic Systems : Theory, Models, and Applications*. Wiley, New York, May 1979.

[54] Z. Manna. *Mathematical Theory of Computations*. McGraw-Hill, USA, January 1974.

[55] M. Mantyla. *An introduction to solid modeling*. Computer science press, July 1988.

[56] M. Margenstern and H. Siegelmann. Nine switch-affine neurons suffice for turing universality. Soumis.

[57] Y. Matiyasevich and G. Sénizergues. Decision problems for semi-Thue systems with a few rules. In *Proceedings, 11$^{\text{th}}$ Annual IEEE Symposium on Logic in Computer Science*, pages 523–531. IEEE Computer Society Press, 27–30 July 1996.

[58] J. McManis and P. Varaiya. Suspension automata : A decidable class of hybrid automata. In D. L. Dill, editor, *Proceedings of the 6th International Conference on Computer Aided Verification*, volume 818 of *Lecture Notes in Computer Science*, pages 105–117, Stanford, USA, 1994. Springer-Verlag.

[59] K. Meer and C. Michaux. A survey on real structural complexity theory. *Bulletin of the Belgian Mathematical Society - Simon Stevin*, 4(1), 1997.

[60] C. Moore. Unpredictability and undecidability in dynamical systems. *Physical Review Letters*, 64(20) :2354–2357, May 1990.

[61] C. Moore. Generalized shifts : unpredictability and undecidability in dynamical systems. *Nonlinearity*, 4 :199–230, 1991.

[62] C. Moore. Recursion theory on the reals and continuous-time computation. *Theoretical Computer Science*, 162(1) :23–44, 5 August 1996.

[63] X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. An approach to the description and analysis of hybrid systems. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 149–178. Springer-Verlag, 1993.

[64] P. Odifreddi. *Classical Recursion Theory*, volume 125 of *Studies in Logic and the foundations of mathematics*. North-Holland, April 1992.

[65] M. S. Paterson. Unsolvability in 3x3 matrices. *Studies in Applied Mathematics*, XLIX(1) :105–107, March 1970.

[66] 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 :1–28, 1974.

[67] E. Ramis, C. Deschamp, and J. Odoux. *Cours de Mathématiques Spéciales, Tome 3, Topologie et éléments d'analyse*. Masson, February 1995.

[68] H. Rogers. *Theory of Recursive Functions and Effective Computability*. MIT Press, April 1987.

[69] Lee A. Rubel. The extended analog computer. *Advances in Applied Mathematics*, 14 :39–50, 1993.

[70] W. Rudin. *Real and Complex Analysis, 3rd edition*. McGraw Hills, USA, March 1987.

[71] E. P. Sacks. A dynamic systems perspective on qualitative simulation. *Artificial Intelligence*, 42 :349–362, 1990.

[72] E. P. Sacks. Automatic analysis of one-parameter planar ordinary differential equations by intelligent numeric simulation. *Artificial Intelligence*, 48 :27–56, 1991.

[73] E. P. Sacks. Automatic qualitative analysis of dynamic systems using piecewise linear approximations. *Artificial Intelligence*, 41 :313–364, 1991.

[74] Y. Saouter. The mortality of a pair of 2x2 matrices is decidable. Technical Report RR-2842, Institut National de Recherche en Informatique et en Automatique, 1996.

[75] P. Schultz. Mortality of 2x2 matrices. *American Mathematical Monthly*, 84(2) :463–464, 1977. Correction 85 :p. 263,1978.

[76] J. A. Sethian. Theory, algorithms, and applications of level set methods for propagating interfaces. *Acta Numerica*, pages 309–395, 1996.

[77] H. S. Shank. The rational case of a matrix problem of harrison. *Discrete Mathematics*, 28 :207–212, 1979.

[78] C. E. Shannon. Mathematical theory of the differential analyser. *Journal of Mathematics and Physics MIT*, 20 :337–354, 1941.

[79] H. T. Siegelmann. Computation beyond the Turing limit. *Science*, 268 :545–548, 1995.

[80] H. T. Siegelmann and E. D. Sontag. Turing computability with neural nets. *Applied Mathematics Letters*, 4(6) :77–80, 1991.

[81] H. T. Siegelmann and E. D. Sontag. Analog computation via neural networks. *Theoretical Computer Science*, 131(2) :331–360, September 1994.

[82] H. T. Siegelmann and E. D. Sontag. On the computational power of neural nets. *Journal of Computer and System Sciences*, 50(1) :132–150, February 1995.

[83] E. Sontag. From linear to nonlinear : Some complexity comparisons. In *IEEE Conference on Decision and Control*, pages 2916–2920, New Orleans, December 1995.

[84] E. D. Sontag. *Mathematical Control Theory : Deterministic Finite Dimensional Systems*. Springer-Verlag, New York, second edition, August 1998.

[85] K. Weihrauch. A simple introduction to computable analysis. Technical Report 171–2/1995, Fernuniversität Gesamthochschule in Hagen, Fachbereich Informatik, Hagen, Allemagne, 1995.

Bibliographie personnelle

Revues internationales

[1] O. Bournez. Achilles and the Tortoise climbing up the hyper-arithmetical hierarchy. *Theoretical Computer Science*, 210(1) :21–71, 6 janvier 1999.

[2] O. Bournez. Some bounds on the computational power of piecewise constant derivative systems. *Theory of Computing Systems*, à paraître.

[3] O. Bournez et M. Cosnard. On the computational power of dynamical systems and hybrid systems. *Theoretical Computer Science*, 168(2) :417–459, novembre 1996.

[4] P. Gros, O. Bournez, et E. Boyer. Using local planar geometric invariants to match and model images of line segments. *Computer Vision and Image Understanding : CVIU*, 69(2) :135–155, février 1998.

Conférences internationales

[5] O. Bournez, O. Maler, et A. Pnueli. Orthogonal polyhedra : Representation and computation. Dans *Hybrid Systems : Computation and Control*, Nijmegen, Pays-Bas, 29–31 mars 1999.

[6] O. Bournez. Some bounds on the computational power of piecewise constant derivative systems. Dans *Automata, Languages and Programming, 24th International Colloquium*, édité par Pierpaolo Degano, Robert Gorrieri, et Alberto Marchetti-Spaccamela, volume 1256 de *Lecture Notes in Computer Science*, pages 143–153, Bologne, Italie, 7–11 juillet 1997. Springer-Verlag.

[7] J. Mundy, C. Huang, J. Liu, W. Hoffman, D. Forsyth, C. Rothwell, A. Zisserman, S. Utcke, et O. Bournez. MORSE : A 3D object recognition system based on geometric invariants. Dans *ARPA Image Understanding Workshop*, pages 1393–1402, Monterey (CA), USA, 13–16 novembre 1994.

Chapitre de livre

[8] O. Bournez et M. Branicky. On matrix mortality in low dimensions. Dans *Open Problems in Mathematical Systems and Control Theory*, édité par V. D.

Blondel, E. D. Sontag, M. Vidyasagar, et J. C. Willems. Springer-Verlag, 1999.

Rapports de recherche

[9] V. D. Blondel, O. Bournez, P. Koiran, C. Papadimitriou, et J. N. Tsitsiklis. Deciding stability and mortality of piecewise affine dynamical systems. Rapport de recherche 1999-05, Laboratoire de L'Informatique du Parallélisme, École Normale Supérieure de Lyon, 1999. Soumis.

[10] O. Bournez et M. Branicky. On the mortality problem for matrices of low dimensions. Rapport de recherche 98-01, Laboratoire VERIMAG, Grenoble, 1998. Soumis.