Modèles Continus. Calculs. Algorithmique Distribuée.

Olivier Bournez

Institut National
Polytechnique de Lorraine

Département de formation doctorale en informatique

École doctorale IAEM Lorraine

Modèles continus.
Calculs.
Algorithmique distribuée.

MÉMOIRE

Jeudi 7 Décembre 2006

pour l'obtention de

l'Habilitation à Diriger les Recherches

par

Olivier Bournez

Composition du jury

Rapporteurs :	Eugène	ASARIN
	S. Barry	COOPER
	Giuseppe	LONGO
Examinateurs :	Eugène	ASARIN
	Vincent	BLONDEL
	José-Félix	COSTA
	Giuseppe	LONGO
	Jean-Yves	MARION

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

A Johanne,

A Pierrick,
A Laëtitia,

Et à Guignol[1].

[1] Je sais, Pierrick, cela ne fait pas trois syllabes, mais on trouvera mieux, promis.

Remerciements

Mes premiers remerciements vont aux rapporteurs de ce mémoire, Eugène Asarin, S. Barry Cooper et Giuseppe Longo, qui m'ont fait l'honneur de lire et de commenter ce travail. Leur vision personnelle, et l'expression de leurs points de vue avant la soutenance, lors, et après la soutenance sont d'ores et déjà pour moi source de stimulation et d'inspiration.

Eugène Asarin avait déjà souffert avec la lecture de ma thèse. Quel courage d'avoir accepté de ressigner. . .. Je crois que les travaux d'Eugène font partie des plus cités dans ce document, en particulier dans le chapitre 3. C'est dire à quel point je les apprécie, et en reconnaît la valeur, et apprécie la valeur de son acception de ce rôle de rapporteur. Je remercie S. Barry Cooper pour avoir spontanément répondu "Of course", malgré toutes ses charges de travail et responsabilités, et sans même savoir à quoi il s'engageait. La réactivité de S. Barry Cooper, et son dynamisme, à tous les niveaux, et en particulier au niveau international pour fédérer tout le réseau Computability in Europe, est un modèle. Enfin, je voudrais remercier Giuseppe Longo, bien entendu pour tout son travail, mais aussi très sincèrement pour m'avoir réellement beaucoup fait progresser dans ma compréhension de certains aspects de mon propre travail par ses écrits, remarques et questions, en particulier sur les liens entre modèles mathématiques, systèmes et lois physiques, et leurs rapports avec le monde qui nous entoure.

Je remercie aussi tous les membres du jury, qui ont accepté de me faire l'honneur de se libérer et de venir à Nancy, malgré toutes leurs contraintes à cette période de l'année. C'est pour moi sincèrement un grand honneur que Vincent Blondel ait accepté de venir écouter, et même présider ce jury. Il avait si bien fait cela lors de ma thèse que je ne pouvais pas faire appel à quiconque d'autre. Plus sérieusement, son recul et sa culture sur un large domaine couvrant ce document me semblaient indispensables à ce jury. Félix Costa, par son rôle moteur dans des créations de forces internationales autour des modèles analogiques, et sa conviction en l'intérêt des travaux autour de ces thématiques, et surtout sa connaissance profonde du domaine, me semblait aussi indispensable. Qu'il ait accepté de venir d'aussi loin, est là encore un extrême honneur. Ses nombreuses remarques constructives sur le document sont aussi encore maintenant une mine d'informations pour des investigations futures. Enfin, le fait que Jean-Yves Marion arrive encore à avoir une opinion positive de moi après tous mes agissements ces dernières années au détriment de sa tranquillité d'esprit de futur chef d'équipe est un réel exploit, qui montre sa valeur et ses qualités. Je tiens à le remercier vivement de m'avoir fait autant confiance scientifiquement ces dernières années. Sans cela, ce mémoire ne serait pas celui-ci. Il était donc lui aussi indispensable bien au-delà du simple rôle de représentation de l'INPL ou du laboratoire.

Je voudrais mettre en avant plusieurs "beta-lecteurs" de ce document, qui en réagissant, parfois spontanément, à ce qui prenait forme sur le web, m'ont réellement beaucoup aidé. Fabienne en fait partie. Kathleen aussi, via Manuel. Daniel et Emmanuel aussi. Mais je voudrais dire un sincère merci à Frédéric, pour toutes les discussions que nous avons eu, ou il a accepté de me laisser délirer à l'heure du café à plusieurs (trop de) reprises. Ses remarques constructives et ses questions ont réellement amélioré la qualité de l'exposition de ce document. J'ai sincèrement beaucoup bénéficié de sa culture, et de sa curiosité d'esprit.

Plusieurs des chapitres de ce document sont relatifs à des résultats obtenus en collaboration. En particulier, le chapitre 3, est le fruit de nombreux échanges de points de vue avec Manuel Campagnolo, au détriment de son été et de sa famille. Sans tous ces échanges, et sans toutes les qualités de Manuel, le survol présenté ne serait pas de cette qualité. Les chapitres 4 et 5 ne seraient rien sans les forces vives Daniel, Emmanuel et Paulin, et aussi sans Felipe et Jean-Yves. Je ne peux que leur être reconnaissant.

Ce document se focalise sur certains de mes travaux, mais il est clair que le choix est parfois cruel,

et qu'oublier Liliana et Florent est oublier aussi injustement la valeur de leur travail et de leurs qualités scientifiques et humaines respectives. Je voudrais les en remercier très sincèrement. Un merci aussi à tous les stagiaires qui, à différents niveaux, ont travaillé avec moi : André, Damien, Djalel Emmanuel, Guillaume, Hassen, Mathieu, Mohamed, Pierre, Régis, Xavier

Plus généralement, je voudrais remercier tous mes collaborateurs au sens large : co-auteurs, membres des équipes protheo et carte, des projets auxquels je participe, ... A tous, un énorme merci. Clairement, au niveau du laboratoire, sans tous ces cafés, ces repas, et ces discussions partagées, la vie aurait été beaucoup plus triste et nettement moins productive. Un grand merci à Bogdan, Daniel, Diane, Didier, Dominique, Dominique, Emmanuel H., Emmanuel H., Fabienne, Florent, Françoise, Frédéric, Guillaume, Isabelle, Jean-Yves, Liliana, Loubna, Luigi, Mathieu, Mathieu, Michaël, Miki, Paulin, Romain, Taoufik, Véronique, Yann, Yannick, Ye-Qiong, et ceux que j'aurais pu oublier.

Un merci particulier, aussi pour Claude et Hélène, pour tout leur support et la liberté qu'ils m'ont laissé dès le départ. Sans tout cela, clairement ce document n'existerait pas. Un merci aussi à Chantal, qui par son efficacité, et son style direct, permet régulièrement de faire avancer les choses, et de revenir sur terre.

Enfin, je voudrais finir par un énorme merci à Johanne, qui continue à me supporter, dans tous les sens du terme. En particulier, qui depuis longtemps à su comprendre que de me laisser délirer parfois à certains moments sur l'intérêt du compteur EDF pour la calculabilité réelle, ou sur l'intérêt des pyramides pour la relativité générale, me faisait du bien.

Grâce à Pierrick, Laëtitia, et au petit numéro encore anonyme, peut-être ne saurais-je pas autant convaincu que ce document ne sera jamais ma meilleure production de l'été 2006.

Préambule

A propos de ce document

On trouvera ce document sur le lien *Document Scientifique* sur `http://www.loria.fr/~bournez/HDR/`.

On trouvera les publications personnelles référencées sur `http://www.loria.fr/~bournez/publications.html`.

Différentes conceptions

À vrai dire, il n'y a pas de définition claire et unanime de ce qu'est, ou ce que doit être, une habilitation à diriger les recherches, à part, bien entendu, les textes légaux, laconiques.

Basons nous sur ce que je vois.

Certains font un bilan, proche d'un rapport d'activité. En fait, le *Curriculum Vitae Commenté*, disponible sur `http://www.loria.fr/~bournez/HDR/` développe sur une cinquantaine de pages, un document qui relève plus de cette conception que celui la.

Certains réalisent un survol sur un domaine. Le chapitre 3 relève de cette conception.

Certains présentent leurs résultats en perspectives. Les chapitres 4, 5, et l'annexe A relèvent de cette conception.

Certains expriment un point de vue. L'annexe *A* relève, au moins dans son début, de cette conception.

Certains insistent sur les perspectives. Les questions, surtout dans les chapitres 1, 2, 3, ou les questions des chapitres 4, et 5 dans une moindre mesure, relèvent de cette conception.

Table des matières

Introduction

Document principal

Motivation Nous avons décidé de consacrer deux chapitres à notre motivation.

Ainsi, le chapitre 1 présente les systèmes dynamiques, avec un point de vue assez mathématique, et classique, ainsi que quelques exemples bien connus de la littérature.

Les exemples du chapitre 2 se distinguent de ceux du chapitre précédant par le fait qu'ils mettent explicitement en jeu une certaine notion de concurrence entre agents.

Ces deux chapitres ne visent pas à réellement produire des résultats nouveaux, mais plutôt à mettre en perspective différents systèmes, et différentes relations entre systèmes, et à mettre en perspectives ce travail.

Chapitre 1 Le premier chapitre vise à argumenter plusieurs faits élémentaires.

Le premier est la richesse, et la subtilité des comportements possibles des systèmes dynamiques continus. Ce point de vue est assez classique, et, sans complexe, est très largement inspiré par l'excellent ouvrage [Hirsch et al., 2003], dont nous reprenons beaucoup d'exemples.

Le second est que différents dispositifs sont intrinsèquement continus, et utilisables comme tels pour réaliser certains calculs.

Enfin, nous voulons insister sur la puissance de modélisation d'une classe de systèmes dynamiques, que nous nommons, en relations avec [Graça, 2006], les problèmes de Cauchy polynomiaux.

Nous discutons aussi, à l'aide du très pédagogique article [Krivine et al., 2006], plusieurs des problèmes qui se posent lorsqu'on cherche à discrétiser des systèmes continus, argumentant par là que l'abstraction continue est souvent la bonne manière de faire.

L'idée que raisonner sur le continu est un moyen très élégant de raisonner sur des systèmes continus ou discrets est initiée dans le chapitre 1, mais se voit essentiellement développée dans le chapitre 2.

Chapitre 2 Nous présentons ainsi dans le chapitre 2, plusieurs modèles, parfois intrinsèquement discrets, dont le bon modèle d'abstraction, de compréhension, et d'étude est le continu.

Certains de ces modèles sont issus de la bioinformatique, pour les réseaux de régulations génétiques, d'autres utilisés en biologie des populations, en virologie biologique, ou en virologie informatique. Nous présentons alors la théorie des jeux, et ses modèles, en nous focalisant sur certains de ses modèles du dynamisme.

Nous cherchons à montrer que ces modèles de dynamiques continus deviennent naturels pour parler d'algorithmique distribuée, en particulier dès que l'on a affaire à des systèmes de grande taille, ou dont on ne contrôle pas les interactions ou la topologie autrement que par des arguments probabilistes ou statistiques.

Ce point de vue nettement moins classique est très récent. Nous discuterons quelques modèles proposés en algorithmique distribuée qui intègrent déjà ces considérations.

Les exemples de ces deux chapitres montrent qu'il est très important d'arriver à comprendre la puissance et la richesse de ces modèles, que ce soit pour les applications citées, mais aussi pour l'algorithmique distribuée et l'informatique actuelle et, nous le pensons, du futur.

Chapitre 3 Ce chapitre constitue un survol de la théorie des calculs pour les modèles à temps continu.

La puissance des modèles de calculs à temps et espace discrets est relativement bien comprise grâce à la thèse de Church : en effet, celle-ci postule que tous les modèles raisonnables et suffisamment puissants ont la même puissance, celle des machines de Turing.

Cependant, on peut considérer des modèles de calculs qui travaillent sur un espace continu. C'est par exemple le cas du modèle de Blum Shub et Smale sur les réels [Blum et al., 1989] (ce modèle est évoqué dans le chapitre 5). Les modèles de l'analyse récursive rentrent aussi dans ce cadre. Ces modèles sont à temps discret.

Mais on peut aussi considérer des modèles où le temps est continu. Les chapitres 1 et 2 en évoquent certains, certains réalistes, certains plutôt folkloriques, et certains futuristes en rapport avec l'algorithmique distribuée. Mais certaines autres grandes classes de modèles ont été considérées. Nous les reprenons dans ce chapitre, en présentant un panorama de ce qui est connu sur leurs propriétés calculatoires.

Chapitre 4 Ce chapitre présente un panorama de quelques-uns de nos résultats personnels à propos de la comparaison de la puissance de plusieurs modèles à temps continu, en relations avec la thèse de Emmanuel Hainry.

Le GPAC (General Purpose Analog Computer) a été introduit en 1941 par Shannon [Shannon, 1941] comme un modèle mathématique d'un dispositif analogique de l'époque : l'Analyseur Différentiel.

Shannon a proposé une caractérisation précise des fonctions calculables dans ce modèle. Les résultats de Shannon ont longtemps été inteprétés comme la preuve que le GPAC est un modèle trop faible, et en tout cas plus faible que l'analyse récursive.

En collaboration avec Manuel Campagnolo, Daniel Graça, et Emmanuel Hainry, nous avons prouvé récemment qu'il n'en est rien.

Ce résultat prend toute sa perspective si l'on comprend que les fonctions calculées par le GPAC correspondent aux problèmes de Cauchy polynomiaux, et que nous argumentons dans le chapitre 1, que presque tous les systèmes dynamiques continus considérés en physique, en biologie, ou en chimie rentrent dans ce cadre.

Parmi les modèles à temps continu, il y a aussi la classe des fonctions $\mathbb{R}$-récursives introduite par Cris Moore dans [Moore, 1996]. L'article de Moore présente des idées fort intéressantes et très originales pour comprendre les calculs sur les réels, qui peuvent se présenter de la façon suivante : puisqu'il n'existe pas de notion de machine universellement acceptée dans le monde continu, pourquoi ne pas contourner le problème en partant des caractérisations des classes de complexité et de calculabilité qui s'affranchissent de notions de machines, en particulier des caractérisations algébriques.

Manuel Campagnolo, dans sa thèse [Campagnolo, 2001], supervisée par Félix Costa et Cris Moore, propose l'idée très intéressante de se limiter aux classes primitives récursives, c'est-à-dire sans opérateur de minimisation, et montre que le remplacement de l'opérateur d'intégration de Moore par un opérateur d'intégration linéaire conduit à une classe de fonctions qui se relie naturellement aux fonctions élémentaires sur les entiers.

Nous avons montré qu'il était possible d'aller plus loin, et de caractériser algébriquement les fonctions élémentairement calculables et calculables au sens de l'analyse récursive.

Nous reprenons nos principaux résultats à ce sujet dans le chapitre 4.

Chapitre 5 Dans ce chapitre, nous reprenons certains de nos résultats à propos de caractérisations logiques de classes de complexité dans le modèle de Blum Shub et Smale, en relations avec la thèse de Paulin Jacobé de Naurois.

Le modèle de Blum Shub et Smale [Blum et al., 1989] constitue un modèle de calcul à temps discret et à espace continu.

Le modèle, défini initialement pour parler de complexité algébrique de problèmes sur le corps des réels, ou plus généralement sur un anneau, a été par la suite été étendu par Poizat dans [Poizat, 1995], [Goode, 1994] en un modèle de calculs sur une structure logique arbitraire.

Par l'intermédiaire de la thèse de Paulin Jacobé de Naurois, aussi supervisée par Felipe Cucker à Hong-Kong et par Jean-Yves Marion, à Nancy, nous avons cherché à comprendre s'il était possible de caractériser syntaxiquement les classes de complexité dans ce modèle sur une structure arbitraire.

Le chapitre 5 reprend nos résultats en droite ligne des caractérisations, dites de la complexité implicite, amorcées par les travaux de Bellantoni et Cook [Bellantoni and Cook, 1992a].

Chapitre 6 Enfin, le dernier chapitre est consacré à une conclusion. Nous y reprenons les principales questions.

Annexes

Il semble de tradition de reprendre en annexe dans une HDR quelques publications.

Annexe A Nous reprenons dans cette annexe l'article [Bournez, 2006].

La question de l'existence de systèmes capables de réaliser des hypercalculs, c'est-à-dire d'effectuer des calculs exploitables qui ne seraient pas réalisables par aucune machine de Turing, fait encore couler de l'encre et des controverses. La situation n'est pas si claire qu'elle n'y parait.

Nous avons été invité à exprimer notre point de vue dans un numéro spécial sur le sujet, dans cet article [Bournez, 2006].

Nous y rappelons, en nous basant sur [Copeland, 2002], plusieurs mauvaises compréhensions fréquentes de ce que dit précisément la thèse de Church, et nous présentons un panorama de plusieurs classes de systèmes mathématiques, avec la caractérisation de leur puissance.

L'article [Bournez, 2006] contient plusieurs résultats originaux, et peut aussi se voir comme une remise à jour avec notre compréhension actuelle, des résultats qui existaient lorsque nous avions débuté notre thèse [Bournez, 1999b].

Chapitre 1

Richesse des systèmes dynamiques continus

1.1 Introduction

Ce chapitre et le suivant présentent quelques systèmes dynamiques issus de modèles de la physique, la biologie, la bioinformatique, la virologie informatique, des jeux, ou de l'informatique distribuée. Ce chapitre se focalise sur les systèmes qui ne font pas intervenir explicitement une certaine concurrence entre agents, alors que le second visera à présenter certains de ces modèles.

Comme nous l'avons annoncé dans le chapitre introductif, le but de notre discussion dans ce chapitre est multiple.

Premièrement, nous cherchons à montrer la richesse des systèmes dynamiques, de leurs comportements obtenus, et des difficultés sous-jacentes à leur étude. Presque tous les exemples de ce chapitre sont tirés de l'excellent ouvrage [Hirsch et al., 2003].

Deuxièmement, nous cherchons à démontrer l'intérêt de la classe des problèmes de Cauchy polynomiaux, à la fois par la calculabilité réelle, et par le fait qu'elle couvre vraiment la plupart des systèmes rencontrés dans la nature.

Enfin, nous cherchons à montrer par tous ces exemples que différents dispositifs sont intrinsèquement continus, et utilisables comme tels pour réaliser certains calculs.

Nous discutons aussi, à l'aide du très pédagogique article [Krivine et al., 2006], plusieurs des problèmes qui se posent lorsqu'on cherche à discrétiser des systèmes continus, argumentant par là que l'abstraction continue est souvent la bonne manière de faire.

1.2 Préambule mathématique

1.2.1 Systèmes dynamiques

Dans ce chapitre, un système dynamique à temps continu correspond à une solution d'une équation différentielle ordinaire, avec une condition initiale. C'est-à-dire, à une solution d'un problème de Cauchy du type

$$y' = f(y), \qquad y(t_0) = x \tag{1.1}$$

où $f : E \to \mathbb{R}^n$, avec $E \subset \mathbb{R}^n$ ouvert.

Une trajectoire $y(t)$ est une fonction différentiable $y : I \subset \mathbb{R} \to E$ qui satisfait l'équation.

Un système dynamique à temps discret correspond à une suite solution d'une équation de récurrence : l'analogue du problème de Cauchy (1.1) pour les systèmes à temps discret est une équation de récurrence du type

$$y_{t+1} = f(y_t), \qquad y_0 = x. \tag{1.2}$$

On trouvera une discussion beaucoup plus complète dans le chapitre 3, de ce qu'on appelle en général un système dynamique.

Un ensemble semi-algébrique est une partie de $\mathbb{R}^n$ qui se représente par un nombre fini d'égalités et d'inégalités polynomiales. Plus précisément, soit $g \in \mathbb{R}[X_1, \ldots, X_n]$ un polynome. Notons $U(g) = \{\mathbf{x} \in \mathbb{R}^n | g(x) > 0\}$. Un ensemble E est semi-algébrique s'il appartient au plus petit ensemble qui contient les $U(g)$ et qui est clos par complémentations, unions et intersections.

Par le théorème de Tarksi-Seidenberg, cela correspond à tous les ensembles définissables dans la logique du premier ordre sur $(\mathbb{R}, +, -, \times)$: tout ensemble qui se définit à partir de constantes réelles, des opérations $+$, $-$, et $\times$, des négations, conjonctions, disjonctions, et des quantifications existentielles et universelles sur des réels, est semi-algébrique.

En guise de préambule à notre discussion, disons que si la modélisation de systèmes par systèmes dynamiques est très ancienne, il faut prendre conscience que la richesse d'au moins une partie des comportements obtenus n'est discutée que depuis peu.

En particulier, comme l'écrivent Hirsch, Smale et Devaney dans la préface de la deuxième édition [Hirsch et al., 2003] de leur ouvrage, en 1970, lors de la première édition, le mot *chaos*, n'avait jamais été utilisé dans un contexte mathématique, pour parler de systèmes dynamiques. Ils écrivent ainsi

> "La découverte de systèmes dynamiques si complexes comme l'application fer à cheval, les enchevêtrements homoclines, le système de Lorenz, et leur analyse mathématique, ont convaincu les scientifiques que les simples mouvements stables comme les solutions d'équilibre ou périodiques ne sont pas les comportements les plus intéressants des équations différentielles."

1.2.2 Systèmes linéaires

Il s'agit des systèmes dynamiques à temps continu $\mathbf{x}' = f(\mathbf{x}, t)$, où f est une fonction linéaire en $\mathbf{x} \in \mathbb{R}^n$. En d'autres termes, il existe une matrice A telle que

$$\mathbf{x}' = A(t)\mathbf{x}. \tag{1.3}$$

Par exemple, une particule de masse m attachée à un ressort voit sa position $x(t)$ au temps t, liée à sa dérivée seconde, par l'équation différentielle du second ordre

$$x'' + p^2 x = 0,$$

pour une certaine constante p représentant le coefficient de raideur du ressort.

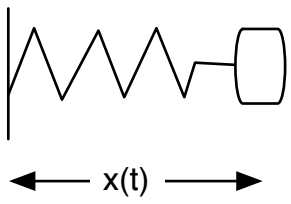


FIG. 1.1 – Oscillateur harmonique

Ce modèle est appelé le modèle de l'oscillateur harmonique, et ses solutions sont du type $x(t) = A\cos(pt + t_0)$.

Il s'agit bien d'un système linéaire, puisque, selon la technique classique, x s'obtient en projetant selon la première coordonnée les solutions du système

$$\left\{ \begin{array}{rcl} x' & = & y \\ y' & = & -p^2 x. \end{array} \right.$$

Ce qui fait la force des systèmes linéaires est leur intérêt en pratique. En effet, il est bien connu que l'étude en un point des solutions d'un système non-linéaire peut se réaliser en raisonnant sur sa linéarisation en ce point [Hirsch et al., 2003].

Cependant, leur pouvoir de modélisation est limité. Une simple dynamique comme $x' = x(1-x)$ n'est pas linéaire par exemple.

Les systèmes linéaires sont relativement bien compris. Par exemple, sur $\mathbb{R}^2$, suivant le déterminant $\det(A)$ et la trace $\mathrm{tr}(A)$ de la matrice A de taille 2×2, on a la classification de comportements possibles de la figure 1.2 : voir [Hirsch et al., 2003].

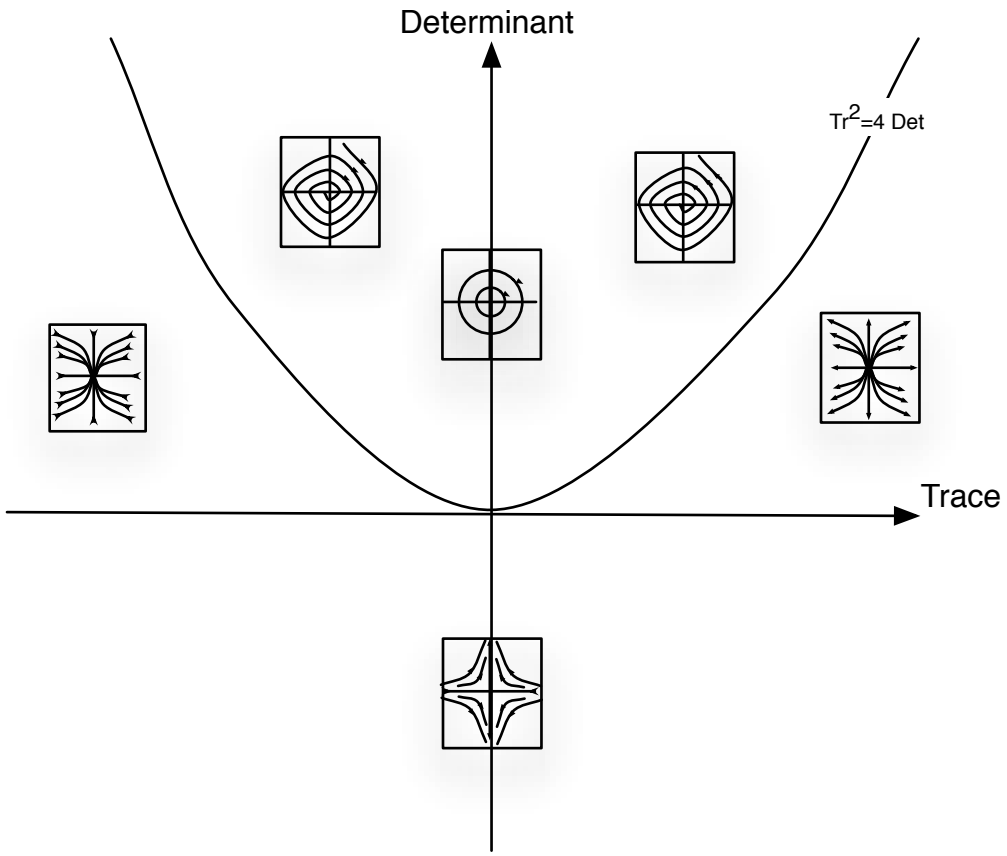


FIG. 1.2 – Représentation symbolique des dynamiques de $\mathbf{x}' = A\mathbf{x}$, en dimension 2, suivant la trace et le déterminant de A : voir [Hirsch et al., 2003].

Cette classification montre du doigt un phénomène intéressant. Tout d'abord, même si le système est linéaire, le type de dynamique obtenu est en fait fonction du déterminant et de la trace de la matrice A, c'est-à-dire une condition semi-algébrique sur les coefficients de A, et non pas une condition linéaire.

En dimension supérieure, l'analyse est plus subtile que la simple considération de la trace et du déterminant, mais l'observation reste vraie : les bons outils pour analyser des dynamiques linéaires ne sont pas des fonctions ou des conditions linéaires, mais des conditions semi-algébriques.

Que des conditions semi-algébriques suffisent peuvent se voir comme une conséquence du théorème de Tarski-Seidenberg.

Mais on peut souvent être plus explicite sur les conditions en question : par exemple, le théorème de Routh-Hurwitz affirme le résultat suivant.

Théorème 1 (Routh-Hurwitz) *L'origine est un point stable attracteur du système linéaire* $\mathbf{x}' = A\mathbf{x}$ *sur* $\mathbb{R}^n$ *ssi* $\Delta_1 > 0$, $\Delta_2 > 0, \cdots, \Delta_n > 0$ *avec*

$$\Delta_k = \begin{vmatrix} b_1 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ b_3 & b_2 & b_1 & 1 & 0 & 0 & \cdots & 0 \\ b_5 & b_4 & b_3 & b_2 & b_1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{2k-1} & b_{2k-2} & b_{2k-3} & b_{2k-4} & b_{2k-5} & b_{2k-6} & \cdots & b_k \end{vmatrix}$$

où le polynôme caractéristique de A *s'écrit* $\lambda^n + b_1\lambda^{n-1} + \cdots + b_{n-1}\lambda + b_n$, *en d'autres termes* $b_i = \frac{1}{i!}\frac{d^i|A-\lambda I|}{d\lambda^i}(0)$

Ainsi, le bon outil pour discuter des systèmes linéaires est en fait la manipulation de conditions de signes sur des polynômes.

1.2.3 Problèmes de Cauchy polynomiaux

Les systèmes

$$\mathbf{x}' = p(\mathbf{x}, t) \tag{1.4}$$

où p est un vecteur de polynômes (en $\mathbf{x}$ et t) ne sont pas des systèmes linéaires, mais la remarque précédente reste vraie appliquée à ces systèmes : l'étude de la stabilité en un point d'un système, peut se réaliser par linéarisation en ce point, et donc avec des polynômes. En quelque sorte, les polynômes permettent bien de parler de polynômes, ce que les fonctions linéaires ne permettent pas. Observons que d'une certaine façon cet argument est à la base[1] du modèle de Blum Shub et Smale [Blum et al., 1989, Blum et al., 1998] sur lequel nous reviendrons (dans le chapitre 5).

Suivant [Graça, 2006], nous proposons de distinguer une classe particulière d'équations différentielles, que nous nommerons les *problèmes de Cauchy polynomiaux*.

Définition 1 (Problème de Cauchy polynomial) *Un problème de Cauchy polynomial est un problème de Cauchy du type*

$$\begin{cases} \mathbf{x}' &= p(\mathbf{x}, t) \\ \mathbf{x}(0) &= \mathbf{x}_0 \end{cases}$$

où $p(\mathbf{x}, t)$ *désigne un vecteur de polynômes, et* $\mathbf{x}_0$ *est une condition initiale.*

Ce qui fait la réelle valeur de cette classe de système est sa généralité, et sa puissance. Immédiatement, tous les systèmes linéaires sont dans cette classe, comme toutes les dynamiques où explicitement l'équation est polynomiale. Mais cela est aussi vrai pour des systèmes qui font apparaître des fonctions comme *sin*, *cos*, ..., et en fait même n'importe quelle fonction qui se définie à son tour par (une projection d') une solution d'un problème de Cauchy polynomial, ce qui permet d'affirmer que tous les exemples de livres comme [Hirsch et al., 2003, Murray, 2002], et en fait tous ceux de ce document sont dans cette classe.

Par exemple, considérons la dynamique d'un pendule. Les lois de la physique donnent immédiatement une dynamique du type

$$x'' + p^2 \sin(x) = 0.$$

Lorsque l'angle $x(t)$ est faible, on peut approcher cette dynamique par l'équation de l'oscillateur harmonique, qui est linéaire. Si on ne fait pas cette hypothèse, à priori la fonction sinus rend la dynamique non directement solution d'une équation différentielle polynomiale.

[1] Au moins de l'existence d'une machine universelle dans le modèle de l'article initial [Blum et al., 1989].

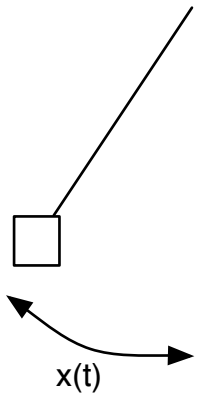


FIG. 1.3 – Pendule

Cependant, posons $y = x'$, $z = \sin(x)$, $u = \cos(x)$. Un simple calcul de dérivée montre que l'on doit avoir

$$\begin{cases} x' &= y \\ y' &= -p^2 z \\ z' &= yu \\ u' &= -yz \end{cases}.$$

Ce qui montre que la solution $x(t)$ s'obtient bien comme la projection d'une solution un système de Cauchy polynomial.

En fait, notre lecteur pourra effectivement se persuader que tous les systèmes considérés dans des livres comme [Hirsch et al., 2003], ou [Murray, 2002] peuvent bien se ramener à des équations différentielles définies par des problèmes de Cauchy polynomiaux, ce qui montre leur intérêt pratique, ne serait-ce que pour la modélisation.

On trouvera dans [Graça, 2006] la preuve de la clôture par addition, soustraction, multiplication, division, composition, différentiation, composition inverse des fonctions (projections de) solutions de problèmes de Cauchy polynomiaux. On y trouvera même la preuve qu'une solution d'une équation du type $\mathbf{x}' = f(\mathbf{x}, t)$, où f est un vecteur de fonctions (projections de) solutions de problèmes de Cauchy polynomiaux se ramène à un problème de Cauchy polynomial.

1.2.4 Problèmes de Cauchy polynomiaux et GPAC

Ces remarques ne sont pas si anodines qu'elle n'y paraissent, car elle nous semble l'incarnation de remarques profondes de calculabilité réelle.

En effet, cette classe de système dynamique prend encore plus d'intérêt si l'on réalise qu'elle capture tout ce qui est calculable par certaines machines à temps continu, comme le General Purpose Analog Computer de Shannon [Shannon, 1941].

Ce modèle de calcul est une abstraction théorique des machines continues qui existaient à l'époque de Shannon, comme l'analyseur différentiel construit pour la première fois en 1931 sous la supervision de Vannevar Bush au MIT [Bush, 1931]. Si l'informatique digitale l'a emporté, n'oublions pas son ancêtre, l'informatique analogique. Nous reviendrons dans un chapitre ultérieur (le chapitre 3) sur l'histoire des machines analogiques.

En effet il a été prouvé récemment (corrigeant et simplifiant les articles [Shannon, 1941], [Pour-El, 1974], [Lipshitz and Rubel, 1987]) que :

Théorème 2 ([Graça and Costa, 2003]) *Une fonction est GPAC-générable (i.e. calculable par le General Purpose Analog Computer de Shannon) si et seulement si elle elle est une projection d'une solution d'un problème de Cauchy polynomial.*

En fait, le GPAC, ou les systèmes polynomiaux permettent de capturer presque toutes les fonctions en pratique. En d'autres termes, Shannon avait bien raison d'appeler son modèle de calcul le "General Purpose" analog computer, puisqu'il possède bien une certaine propriété d'universalité, comme la machine de Turing possède une propriété d'universalité sur les machines discrètes, et les fonctions calculables discrètes.

Il est intéressant de réaliser que nous avons à peu près le même phénomène qu'en calculabilité classique pour certaines classes de fonctions, comme les fonctions élémentaires, introduites par [Kalmár, 1943], ou les fonctions primitives récursives.

En effet, en calculabilité classique, toutes les fonctions usuelles ou presque sont élémentaires (respectivement primitives récursives). Les seuls rares contre-exemples, comme la fonction d'Ackermann, correspondent d'une certaine façon à une diagonalisation sur cette propriété [Kalmár, 1943], [Rose, 1984]. D'autres part, en calculabilité classique, ces classes sont très robustes et stables par de nombreuses opérations.

Ici, presque toutes les fonctions sont GPAC calculables. Les propriétés précédentes attestent de la robustesse de la classe des fonctions GPAC calculables. Les quelques rares exceptions de fonctions non GPAC calculables connues sont obtenues en considérant des fonctions qui ne sont pas différentiellement algébriques : une fonction unaire y est différentiellement algébrique sur l'intervalle I, s'il existe un polynôme non nul p à coefficients réels tel que

$$p\left(t, y, y', ..., y^{(n)}\right) = 0$$

sur I. Une fonction qui n'est pas différentiellement algébrique est dite *transcendentalement transcendentale*.

Question 1 *Est-il possible de formaliser cela ? Peut-on relier fonctions élémentaires, primitives récursives et fonctions calculées par le GPAC, plus formellement que par cette simple analogie ?*

Parmi les fonctions transcendentalement transcendentales on compte :

Théorème 3 *Les fonctions*

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$$

et la fonction zeta de Riemmann

$$\zeta(x) = \sum_{k=0}^{\infty} \frac{1}{k^x}$$

ne sont pas différentiellement algébriques.

Ces fonctions constituent en quelque sorte l'analogue d'Ackermann pour les fonctions élémentaires. La preuve pour la fonction Γ est due à Hölder en 1887 [Hölder, 1887]. Celle de la fonction Γ à Hilbert, rédigée par Stadigh [Stadigh, 1902].

1.3 Discret versus continu

1.3.1 Ne soyons pas trop discrets

Après ces préambules mathématiques, venons en au coeur du sujet.

Certains de nos collègues, souvent informaticiens, ont régulièrement le travers de nous demander pourquoi nous nous intéressons à des systèmes continus, avec des arguments du type

- en informatique, rien n'est continu, tous les processus sont discrets
- le continu n'est qu'une abstraction irréaliste du monde discret
- les systèmes continus il faut de toute façon les discrétiser pour les simuler
- ...

– que fait l'informatique là-dedans
– ...

Notre réponse à ces objections constituera une suite d'exemples, pour motiver cette discussion.

1.3.2 Quelques systèmes dynamiques discrets

Commençons tout d'abord par présenter quelques exemples de classes de systèmes dynamiques à temps discret qui ont fait beaucoup couler d'encre. Le premier plutôt chez les informaticiens. Le second plutôt chez les mathématiciens.

La machine de Turing

Tout d'abord, la machine de Turing. Une machine de Turing sur l'alphabet Σ, avec l'ensemble d'états internes Q, correspond bien à un système dynamique à temps discret. L'état de la machine à un instant donné correspond à la donnée du contenu du ruban, de la position de la tête de lecture, ainsi que de l'état interne de la machine. Tout ceci peut se coder par exemple par un élément de l'ensemble $\Sigma^\omega \times \mathbb{Z} \times Q$. Le programme de la machine correspond à certaines règles d'évolutions, qui se traduisent immédiatement par une fonction de transition discrète qui donne, à partir de l'état de la machine au temps t, l'état de la machine au temps $t+1$.

Voir une machine de Turing comme un système dynamique à temps discret particulier ne fait pas réellement avancer les choses (et est loin d'être original, voir par exemple [Minsky, 1967]), mais il nous semble important de comprendre que lorsqu'on discute de systèmes dynamiques à temps discret en toutes généralités, on a au moins la richesse de cette sous-classe : possibilité d'autosimulation, résultats d'indécidabilité, etc.... La possibilité qu'une machine de Turing puisse calculer toute fonction calculable, et en particulier une autre machine de Turing, implique rapidement que c'est la classe de systèmes dynamiques la pire qu'il soit au niveau des comportements pour tous les systèmes dynamiques à temps discret (au moins du moment que la fonction de transition est calculable). En particulier, la classe des machines de Turing est une classe de systèmes chaotiques, qui peut exhiber des attracteurs étranges, etc...

Ces considérations apparaissent aussi sur les systèmes dynamiques à temps continu, dès que ceux-ci permettent la simulation de systèmes dynamiques à temps discrets comme les machines de Turing.

L'application logistique

Ayant considéré le pire, tentons de considérer le plus simple. Le plus simple serait une dynamique linéaire du type $x(t+1) = ax(t)$, i.e. $x(t) = x(0)a^t$, c'est-à-dire une dynamique d'une suite géométrique, pas vraiment intéressante.

Considérons une dynamique polynomiale sur $\mathbb{R}$ du type

$$x(t+1) = \lambda x(t)(1 - x(t)), \tag{1.5}$$

c'est-à-dire *l'application logistique*. On supposera $\lambda > 0$.

Cette dynamique est en fait motivée par une dynamique de population. En effet, si on suppose que le taux de fécondité de chaque individu est constant à chaque génération, le nombre d'individu au temps t suit la loi de Malthus

$$x(t+1) = ax(t),$$

et donc une croissance exponentielle, sans limites.

Il est nettement plus raisonnable de supposer, comme l'a fait Verhulst en 1838, que la dynamique est en fait du type

$$x(t+1) = \lambda x(t)(1 - x(t)/M),$$

où M est la population maximale que peut supporter l'environnement. En faisant le changement de variable $x(t) = X(t)/M$, on retombe sur la dynamique précédente.

Depuis l'article [May, 1976], il est bien connu qu'une variété extrêmement riche de comportements est générée selon la valeur de la constante λ. Rappelons les propriétés principales.

Pour $1 < \lambda < \lambda_1 = 3$, le point fixe $\lambda^* = 1 - 1/\lambda$ est stable, globalement attracteur et attire toute trajectoire qui part de $x(0) \in]0, 1[$. Pour $\lambda = \lambda_1$, un cycle de longueur 2 apparaît par le biais d'une bifurcation en fourche. Ce cycle de longueur 2 reste stable et globalement attracteur pour $x(0) \in]0, 1[$, et $3 < \lambda < \lambda_2 = 1 + \sqrt{6}$. Cela continue avec une suite $\lambda_3, \lambda_4, \cdots, \lambda_k$ telle que pour $\lambda_k < \lambda < \lambda_{k+1}$ il y a un cycle attracteur de longueur 2^k. La suite $\lambda_1, \cdots, \lambda_k, \cdots$ converge vers $\lambda^\infty \approx 3.5699$. Pour $\lambda^\infty < \lambda$ la dynamique devient chaotique.

La dynamique de ce système est maintenant relativement bien connue, mais avant 1976, en ne se doutait pas qu'un système dynamique en dimension 1 à temps discret, qui plus est aussi simple soit capable d'une telle richesse de comportements possibles.

Le dessin de la figure 1.4 est maintenant très souvent présent dans les livres récents sur les systèmes dynamiques.

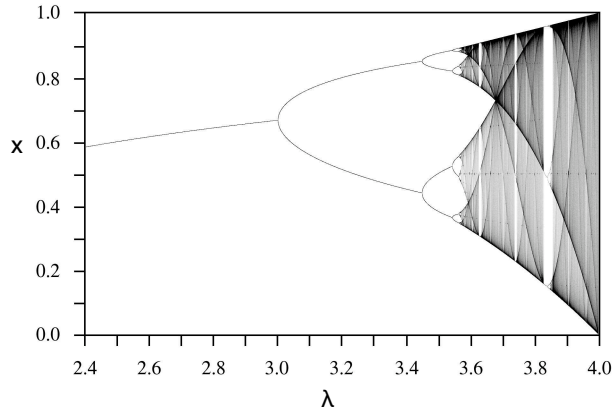


FIG. 1.4 – Diagramme d'orbites pour la famille logistique pour $2.4 < \lambda < 4$ (image provenant de la base commune de Wikipedia)

1.3.3 Sur les méfaits de la discrétisation

Répétons les propos, fort pédagogiques et instructifs de [Krivine et al., 2006]. Il n'est pas possible de donner une solution analytique de la dynamique (1.5) (sauf pour $\lambda = 4$). Aussi, puisque c'est la limite quand t est très grand qui est d'intérêt, il est intéressant de s'intéresser au problème continu correspondant. Le système continu correspondant est

$$y' = y(\lambda(1 - y) - 1) \tag{1.6}$$

Les solutions de (1.5) correspondent à la discrétisation par la méthode d'Euler avec un pas de 1 de l'équation continue (1.6). Cependant, il s'avère que (1.6) possède une solution analytique explicite

$$y(t) = \frac{(\lambda - 1)y_0}{\lambda y_0 + (\lambda(1 - y_0) - 1)e^{-(\lambda-1)t}}$$

ce qui tend vers $\lambda^* = 1 - 1/\lambda$. En d'autres termes, la dynamique continue n'est pas chaotique, et se comporte beaucoup mieux que ses discrétisations.

Les discrétisations par la méthode d'Euler de la version continue reflètent le comportement de la version continue lorsque le pas de discrétisation est plus petit qu'une valeur caractéristique, et deviennent chaotique pour des pas de discrétisations plus grands [Krivine et al., 2006].

Ceci constitue un premier exemple de système, où la discrétisation peut introduire des complications, et rendre l'analyse beaucoup plus ardue qu'en restant dans le monde continu.

Cet exemple peut paraître artificiel, est complètement déconnecté de toute réalité. Il est vrai que l'application logistique peut sembler un jouet pour mathématiciens. Cependant, si des équations aussi simples sont problématiques, on ne peut s'attendre à beaucoup d'optimisme lorsque les équations se compliquent.

Pour les plus sceptiques de l'apport du continu, revenons à des choses bien connues. Commençons par nous persuader qu'on sait calculer autrement qu'avec un ordinateur discret. Pour cela, montrons comment réaliser facilement des opérations continues.

1.4 Réaliser des calculs continus

Chacun s'intéresse à la puissance des machines qu'il possède...Alors voyons ce qu'était la calculabilité du XIXième siècle. Nous ne sommes pas en train de promouvoir l'informatique de deux siècles en arrière, mais juste de souligner que la calculabilité réelle continue ne date nullement d'aujourd'hui, et que cette calculabilité a donné par le passé de très jolis résultats mathématiques.

1.4.1 Mécanismes planaires

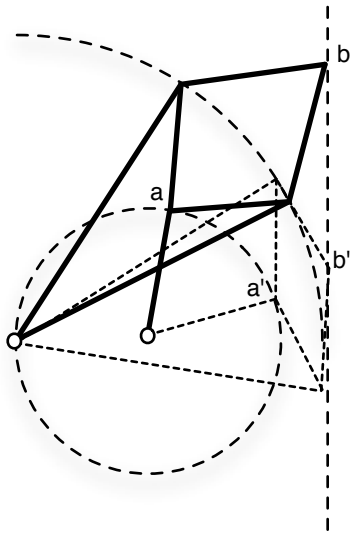


FIG. 1.5 – Mécanisme de Peaucellier. Le mouvement circulaire de a est transformé en un mouvement linéaire de b.

La puissance des mécanismes planaires constitués de barres rigides jointes à leurs extrémités par des rivets a attiré beaucoup d'attentions en Angleterre et en France dans la fin du 19ième siècle, avec un regain d'intérêt en Russie dans les années 1940. Voir par exemple [Artobolevskii, 1964], [Svoboda, 1948].

Tout le monde connaît le pantographe qui permet de réaliser des dilatations. Le mécanisme de Peaucellier permet de transformer un mouvement linéaire en un mouvement circulaire.

Plus généralement, il est naturel de se poser la question de la puissance de calcul de tels dispositifs.

Elle est donnée par le théorème suivant très joli, [Artobolevskii, 1964], [Svoboda, 1948], attribué à Kempe [Kempe, 1876], formulé ici à partir de [Smith, 1998] : la puissance de tels dispositifs correspond aux ensembles semi-algébriques.

Théorème 4 (Complétude des mécanismes planaires) – *Pour tout ensemble semi-algébrique S non-vide, il existe un un mécanisme avec n points qui se promènent sur des segments linéaires, mais libres de se déplacer sur ces segments, et forçant la relation $(x_1, \ldots, x_n) \in S$, où x_i sont les distances sur les segments linéaires.*
– *Réciproquement, le domaine d'évolution de tout mécanisme fini planaire est semi-algébrique.*

On voit réapparaître de nouveau les polynômes, et les ensembles semi-algébriques.

Puisque ces mécanismes peuvent paraître antiques, passons à de l'électronique plus moderne, par plusieurs exemples.

1.4.2 Courbes de Lissajous

Notre premier exemple est purement pédagogique, et est ici essentiellement parce qu'il nous amuse beaucoup. Cela dit, il pose une question de fond, plus profonde qu'il n'y parait.

Question 2 *Qu'est ce qui est raisonnable et qui ne l'est pas quand on parle de calculs sur les réels ?*

Il est facile de générer une sinusoïde avec un circuit R, L, C, et donc une solution à l'équation $x'' = -p^2 x$. La constante p, liée à la période du signal, s'exprime facilement en fonction de R, L, C.

Tout étudiant qui a fait des travaux pratiques de physique en terminale s'est essayé à l'oscilloscope, et a tenté de brancher sur l'entrée qui correspond à l'axe des x un tel signal sinusoïdal, et sur l'axe des y une autre sinusoïde de période distincte, correspondant à une autre constante q.

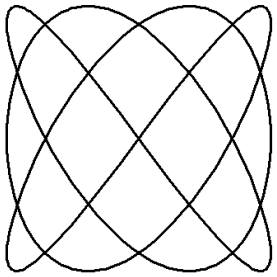


FIG. 1.6 – Une courbe obtenue pour $p/q = 4/3$.

En faisant varier la valeur de cette autre période, tout bon étudiant sait qu'on obtient des courbes de Lissajous : voir la figure 1.6.

Théorème 5 (Courbes de lissajous) – *Pour p/q rationnel, la courbe est stable graphiquement sur l'oscilloscope, et permet de lire la valeur de la fraction p/q, par le nombre nombre d'oscillations de x et de y sur un cycle.*
– *Pour p/q irrationnel, la courbe est dense sur l'écran.*

D'une certaine façon, on a un système physique de calculs capable de déterminer si le rapport de deux valeurs est rationnel ou irrationnel. Théoriquement, cela est impossible, pour des modèles de calculs sur les réels les plus classiques comme l'analyse récursive. N'est-ce pas étonnant ?

Observons que même si l'on ne croit pas au test de rationalité irrationalité, on observera qu'étant donné p, q, le système retourne l'écriture réduite de la fraction p/q, i.e. simplifie les fractions.

Bon, les esprits les plus chagrins n'aimeront pas notre exemple, et notre modèle de calcul par oscilloscope, en discutant le comment détermine t-on si une courbe est dense ou non, ou comment lire le nombre d'oscillations de x et y quand celui-ci est grand, ...etc. Cela dit, sauraient-ils m'expliquer ce qui est "raisonnable"

ou ne l'est pas, formellement pour ce modèle de calcul ? Et caractériser mathématiquement quelle serait sa puissance, si l'on n'autorise que des opérations "raisonnables" de lecture sur l'écran, sans faire référence à une hypothétique thèse de Church pour oscilloscope ?

Bon certes, considérons des choses plus réalistes.

1.4.3 Montage intégrateur

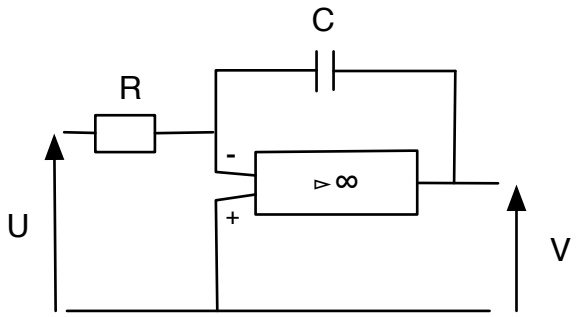


FIG. 1.7 – Un montage intégrateur utilisant un amplificateur opérationnel.

C'est un exercice classique de classe de terminale que d'exprimer la tension V de sortie en fonction de la tension U d'entrée dans le montage de la figure 1.7 comprenant un amplificateur opérationnel.

On a

$$V(t) = -1/RC \int_0^t U(t)dt,$$

i.e. le système réalise un montage intégrateur. La tension en sortie est l'intégrale de la tension en entrée.

Ceci prouve, si besoin, qu'il n'y a nul besoin de discrétiser pour calculer une intégrale. Il est ainsi facile de réaliser des opérations continues, autrement qu'avec des ordinateurs classiques.

1.4.4 Un ordinateur continu

Si l'on se souvient du montage intégrateur de la classe de terminale, on se souvient probablement des montages additionneurs, et on sait comment générer une tension constante.

Autrement dit, on conviendra qu'il est possible de réaliser chacune des opérations de la figure 1.8. Pour chacun des éléments de cette figure, il s'agit de réaliser un montage qui génère (à droite) en sortie la relation indiquée en fonction de ses entrées (à gauche).

On sait alors réaliser des circuits comme sur la figure 1.9, en connectant différents éléments, en autorisant les rétroactions (feedbacks).

Shannon a prouvé que ce qu'on lisait à l'une des sorties d'un tel système correspondait exactement aux fonctions calculables par sa machine, le General Purpose Analog Computer [Shannon, 1941]. Bien entendu, à cette époque, il n'y avait pas d'amplificateurs opérationnels, ni d'électronique d'ailleurs, mais on savait réaliser mécaniquement toutes les opérations de la figure 1.8, et on utilisait effectivement ces mécanismes dans les ordinateurs mécaniques de l'époque comme l'Analyseur Différentiel du MIT.

Autrement dit, on a

Théorème 6 ([Shannon, 1941]) *Les trois conditions suivantes sont équivalentes :*

- *La fonction f est calculable par un tel montage (i.e. se lit en fonction du temps à la sortie d'une unité d'un tel assemblage d'unités)*
- *La fonction f est GPAC-calculable (i.e. calculable par Analyseur Différentiel)*

– *La fonction f est projection d'une solution d'un problème de Cauchy polynomial.*

La preuve de Shannon (corrigée par [Pour-El, 1974, Lipshitz and Rubel, 1987, Graça and Costa, 2003]) est complètement constructive, et donne l'assemblage en fonction de la description du problème de Cauchy polynomial.

En d'autres termes, le théorème peut encore se lire comme : "nous savons calculer toute fonction solution d'un problème de Cauchy polynomial, en temps réel, sans utiliser le moindre ordinateur discret. Réciproquement, on ne peut pas faire plus."

Croisé avec nos remarques sur le fait que toutes les fonctions usuelles sont projections d'une solution d'un problème de Cauchy polynomial, sauf quelques rares exemples ressemblant à une diagonalisation, n'est-on pas en train de dire que le GPAC capture le bon modèle de ce qui est calculable sur les réels ?

Question 3 *Il y'a t'il une thèse de Church pour les calculs en temps continu ?*

Les gens de l'analyse récursive diront que non, puisque la fonction $\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt$ est calculable au sens de l'analyse récursive, alors qu'elle n'est pas GPAC calculable. Mais on compare deux notions de calculs distinctes : dans le modèle de l'analyse récursive, on parle de calculs à la limite, et dans le GPAC de calcul en temps réel. Cela mène à la question suivante, nettement moins ambitieuse, sur laquelle nous reviendrons.

Question 4 *Quelle est la puissance exacte du GPAC, si l'on ne se restreint pas aux fonctions calculées en temps réel ? Le GPAC est-il réellement moins puissant que les machines de Turing ?*

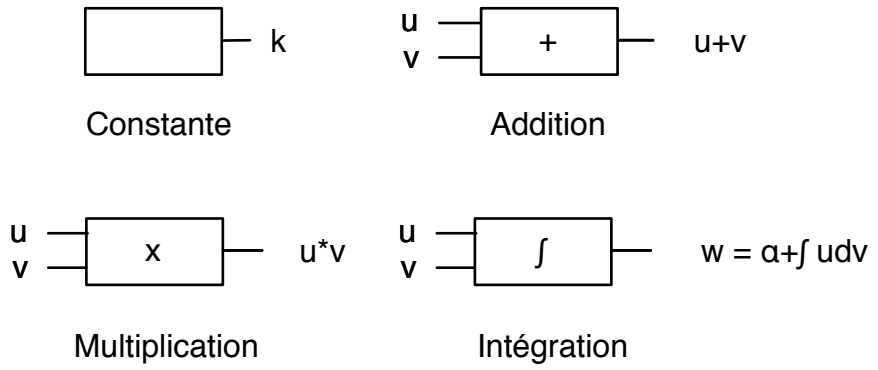


FIG. 1.8 – Les différentes unités de base d'un GPAC (la sortie w d'un opérateur d'intégration satisfait $w'(t) = u(t)v'(t)$, $w(t_0) = \alpha$ pour une condition initiale α).

1.5 Des systèmes dynamiques remarquables

Oublions quelques pages le besoin de voir des systèmes de calculs dans tous les dispositifs, et discutons de différents systèmes dynamiques de la littérature.

Notre discussion vise uniquement à montrer la richesse des comportements possibles pour des systèmes dynamiques, en discutant parfois des problèmes de discrétisation associés. Elle se poursuivra dans le chapitre suivant, en discutant cette fois des modèles faisant intervenir certaines notions de concurrence.

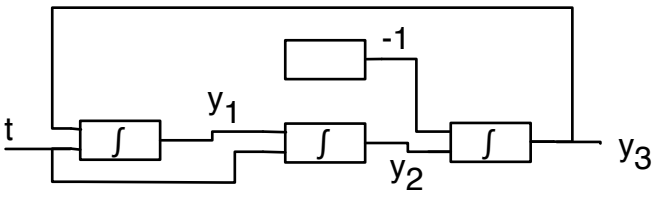


FIG. 1.9 – Génération de cos et sin par un GPAC. Sous forme de système d'équations, on a $y_1' = y_3$, $y_1(0) = 1$, $y_2' = y_1$, $y_2(0) = 0$, $y_3' = -y_1$, $y_3(0) = 0$. Il en suit que $y_1 = \cos$, $y_2 = \sin$, $y_3 = -\sin$.

1.5.1 En météorologie

Le système dynamique chaotique le plus célèbre est sans contestation le système formulé par Lorenz en 1963 comme un modèle (largement simplifié) de la convection atmosphérique

Le système de [Lorenz, 1963] s'écrit

$$\begin{cases} x' &= \sigma(y-x) \\ y' &= \rho x - y - xz \\ z' &= xz - bz \end{cases}$$

où σ, ρ et b sont trois paramètres réels.

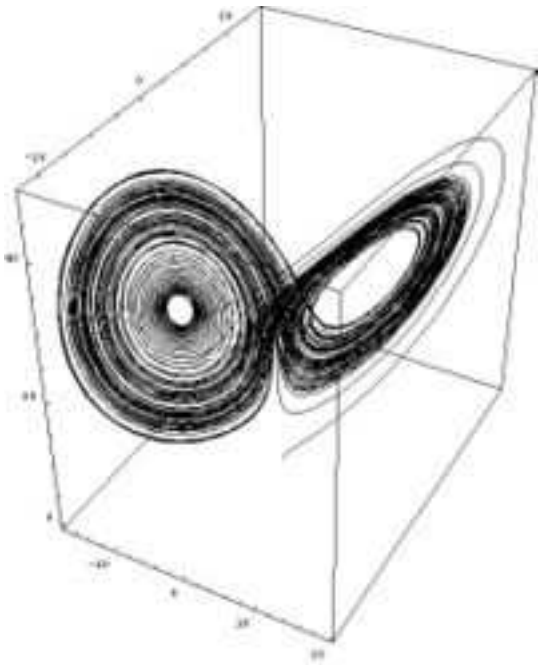


FIG. 1.10 – Une dynamique du système de Lorenz (image provenant de la base commune de Wikipedia).

Pour $\sigma = 10$, $\rho = 28$, $b = 8/3$, le système possède ce que l'on appelle un attracteur étrange. Avant que le modèle ne devienne la proie des scientifiques, les seuls attracteurs connus pour des systèmes dynamiques continus étaient des point fixes et des orbites closes [Hirsch et al., 2003]. Les trajectoires convergent vers l'attracteur, cependant deux conditions initiales arbitrairement proches diffèrent ultimement dans leur manière

de converger vers cet attracteur. De mêmes phénomènes ont été exhibés ensuite pour d'autres dynamiques, comme pour l'attracteur de Rosler.

Nous renvoyons à [Hirsch et al., 2003], pour une introduction. Nous observerons que pour étudier ce système dynamique continu, les auteurs de [Hirsch et al., 2003], le remplacent par une dynamique discrète, en fait une dynamique hybride. Cela montre, si besoin est, que les systèmes discrets, et surtout les systèmes hybrides, sont pertinents pour l'analyse de systèmes continus.

On observera que le système est de dimension 3. Les phénomènes chaotiques du même type en dimension 2 ne sont pas possibles en raison du Théorème de Poincaré-Bendixon.

1.5.2 En chimie

Modèle de Lotka-Volterra

Lotka observa dans [Lotka, 1920] que l'ensemble de réactions couplées, autocatyliques suivant possède des propriétés dynamiques remarquables.

$$\left\{ \begin{array}{lcl} \overline{X} + Y_1 & \rightarrow & 2Y_1 \\ Y_1 + Y_2 & \rightarrow & 2Y_2 \\ Y_3 & \rightarrow & Z \end{array} \right.$$

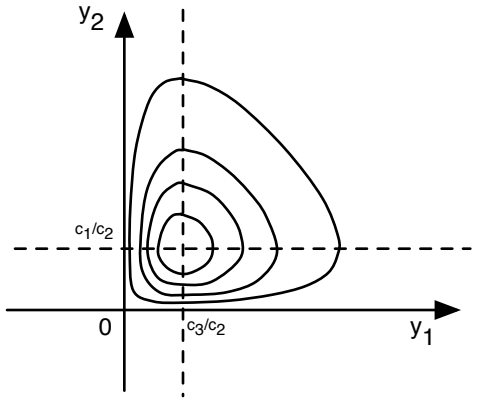


FIG. 1.11 – Dynamiques du modèle de Lotka-Volterra

Mis en équation, la dynamique est simplement

$$\left\{ \begin{array}{lcl} y_1' & = & c_1 y_1 - c_2 y_1 y_2 \\ y_2' & = & c_2 y_1 y_2 - c_3 y_2 \end{array} \right.$$

où c_1, c_2, c_3 sont les constantes cinétiques des réactions.

Cette dynamique coïncide avec la dynamique proposée par Volterra en 1925 [Volterra, 1931] comme un modèle simple des systèmes proies prédateurs sur lesquels nous reviendrons.

Théorème 7 *Le système possède un point stable $y_1 = c_3/c_2$, $y_2 = c_1/c_2$. Toutes les trajectoires partant d'un autre point que celui-ci s'avère former des courbes closes, en laissant invariant la quantité $H(u, v) = \alpha(u - \log u) + (v - \log v)$, où $u = c_2 y_1/c_3$, $v = c_2 y_2/c_1$, $\alpha = c_3/c_2$. En d'autres termes, les trajectoires sont les courbes de niveau de la fonction H : voir la figure 1.11*

Le système est instructif pour plusieurs raisons. D'une part, il montre de façon théorique que certains mécanismes chimiques peuvent provoquer des oscillations. Cependant le modèle est purement formel, sans lien direct avec un système réel, et à vrai dire avant la réaction découverte par Belousov, discutée plus loin, la plupart des chimistes pensaient que cela était impossible d'observer physiquement de telles oscillations, en raison des principes de la thermodynamique.

Surtout, c'est un exemple fort bien discuté dans [Krivine et al., 2006], à propos de problèmes liés à la discrétisation.

En effet, [Krivine et al., 2006] montre que tout schéma d'Euler explicite de simulation de la dynamique de ce système ne conserve pas la fonction H, et donc qu'en pratique les oscillations divergent dans toute simulation numérique par une méthode d'Euler. Il est possible de construire un schéma d'Euler implicite qui fonctionne [Krivine et al., 2006], mais cela montre encore une fois que discrétiser un système continu est souvent très problématique, et donne lieu à des systèmes qui ne simulent pas fidèlement le système sous-jacent.

Réaction de Belousov-Zhabothinksy

La réaction oscillante la plus connue est sans conteste la réaction découverte par Belousov en 1950.

Belousov s'intéressait au cycle de Krebs, et notamment au rôle de l'acide citrique dans ce cycle. Le cycle de Krebs est un mécanisme biochimique complexe intervenant dans le métabolisme des sucres dont le glucose. Belousov cherchait à comprendre le rôle de l'acide citrique qui est oxydé dans ce cycle. Il pensa le doser grâce au bromate de potassium, mais la réaction était trop lente et il ajouta un catalyseur. Mais il constata alors que le milieu réactionnel changeait de couleur périodiquement. Il entreprit alors une étude approfondie en variant le pH de la solution et en ajoutant un indicateur coloré. Cependant toutes ses tentatives de publications furent refusées par des jurys hostiles à l'idée d'oscillations chimiques, argumentant avec la thermodynamique sur l'impossibilité de la chose. Il publia uniquement une note dans le journal méconnu *Sbornik Referatov po Radiaconi Medecine*, reprise dans [Field and Burger, 1985]

En 1961, Anatol Zhabotinsky, étudiant en biophysique à l'Université de Moscou, consacra son travail de thèse à l'étude approfondie de la réaction de Belousov [Zaikin and Zhabotinsky, 1970]. Suivant la suggestion de son professeur S. E. Schnoll, il remplaça l'acide citrique par l'acide malonique et obtint un système dans lequel l'amplitude des oscillations était encore plus grande que dans le système original.

Pendant plusieurs années, la réaction de Belousov-Zhabotinsky resta une curiosité de laboratoires, peu connue en particulier à cause du contexte politique de l'époque[Dupuis and Berland, 2004].

On connaît maintenant plusieurs exemples de réactions oscillantes. Nous renvoyons à [Murray, 2002, Field and Burger, 1985] pour de plus amples discussions de cette réaction.

1.5.3 En physique

Équation de Van der Pol

Considérons un circuit R, L, C comme sur la figure ci-dessous où R n'est pas une résistance parfaite : une résistance parfaite introduirait une relation linéaire entre la tension à ces bornes et l'intensité qui la traverse. Supposons que cette relation soit en fait une certaine fonction $V = f(i)$.

En posant, $L = C = 1$, x l'intensité qui traverse le solénoide, y la tension aux bornes du condensateur, le système doit vérifier

$$\left\{ \begin{array}{rcl} x' & = & y - f(x) \\ y' & = & -x \end{array} \right.$$

ce qu'on appelle *l'équation de Lienard*.

Dans le cas $f(x) = x^3 - x$, on obtient *l'équation de van der Pol*.

$$x' = y - x^3 + x$$

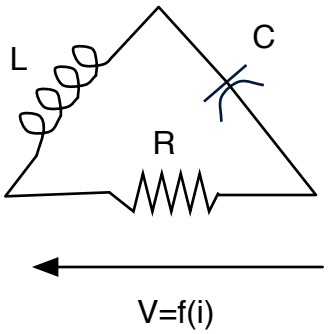


FIG. 1.12 – Circuit R,L,C

Le résultat suivant est un joli exercice classique de mathématiques. Le résultat est très joli, car la preuve ne consiste pas du tout à exhiber la solution périodique, mais à utiliser des arguments de topologie (point fixe) sur une section de Poincaré du système.

Théorème 8 *Il y a une solution non triviale périodique de l'équation de van der Pol, et toute autre solution (sauf le point d'équilibre instable à l'origine) tend vers cette solution. En d'autres termes, le système oscille.*

Si l'on considère

$$\left\{ \begin{array}{rcl} x' & = & y - f_\mu(x) \\ y' & = & -x \end{array} \right.$$

avec $f_\mu(x) = x^3 - x$, $\mu \in [-1, 1]$, on retrouve le système de van der Pol pour $\mu = 1$. On observe une bifurcation de Hopf : pour $\mu < 0$, toutes les solutions convergent vers l'origine : le système est sans vie. Lorsque μ devient positif, la solution périodique apparaît, le système se met à osciller et, d'une certaine façon, prend vie.

Problème des n-corps

Le problème des n-corps consiste à résoudre les équations du mouvement de n corps en interaction gravitationnelle, connaissant leurs masses et leurs positions et vitesses initiales.

Tout étudiant non-amnésique sait que le problème à deux corps est entièrement soluble analytiquement, en utilisant les lois de Newton et de Kepler.

Contrairement à une idée répandue, le problème à trois corps possède une solution analytique exacte, découverte par Sundman en 1909 [Henkel, 2001]. Malheureusement cette solution se présente sous la forme d'une série infinie de convergence très lente, ce qui la rend inutile en pratique.

Dans le cas de n-corps, pour $n > 2$, en dehors de quelques cas rares où une solution exacte est connue, on utilise en général des méthodes de résolution approchées.

Comme le fait remarquer [Krivine et al., 2006], observons que toute discrétisation du schéma de mouvement du problème à 2-corps par une méthode d'Euler explicite ne simule pas correctement le mouvement des 2 corps. En effet, ces schémas ne conservent pas l'énergie du mouvement.

Comme le fait observer, [Coullet et al., 2004], [Krivine et al., 2006], pour l'anecdote observons que Newton dans ses *Principia* en 1687, qui ne connaissait pas le calcul différentiel inventé ultérieurement, a fait un raisonnement géométrique discret pour établir la théorie de l'attraction universelle, en utilisant un schéma de discrétisation implicite, emprunté à Robert Hooke. Il est remarquable que son schéma ait le mérite de conserver l'énergie [Coullet et al., 2004], [Krivine et al., 2006], alors qu'un schéma de discrétisation d'Euler explicite n'aurait pas permis le raisonnement aussi simplement.

Chapitre 2

Populations. Concurrence. Algorithmique distribuée

Nous poursuivons notre panorama de différents modèles de systèmes dynamiques continus, en nous focalisant maintenant sur des modèles faisant intervenir plusieurs agents, à priori plutôt en grand nombre, et donc avec potentiellement une certaine compétition ou une concurrence entre les agents.

Nous visons par la à montrer que l'abstraction continue est naturelle pour parler de populations d'individus, même si parfois ceux-ci, ainsi que leurs comportements sont intrinsèquement discrets (ou même stochastiques).

D'autre part, nous cherchons à montrer que les modèles de populations, ou de la théorie (évolutionnaire) des jeux sont pertinents et naturels pour comprendre l'informatique distribuée actuelle, en particulier lorsque le nombre d'agents devient grand, ou lorsque la topologie n'est pas connue autrement que par des arguments statistiques ou probabilistes.

2.1 Bio-informatique

2.1.1 Réseaux de régulations génétiques

Ne serait-ce que par le séquençage du génome humain et le déluge de données biologiques qui en découle, ou la compréhension récente de nombreuses voies biologiques, il est vital que les outils informatiques et mathématiques s'adaptent pour se montrer à la hauteur de toutes ces données.

Le domaine de ce qu'on appelle la bioinformatique a pris son essor récemment, et vise, par exemple, à identifier les gênes codés par l'ADN, comprendre la structure primaire, secondaire ou tertiaire des protéines, ou encore par exemple à comprendre le fonctionnement des systèmes biologiques.

En fait, sur ce dernier point, on peut aussi dire qu'un nouveau domaine de la biologie, la biologie des systèmes, est né [Kitano, 2001]. Alors que la biologie traditionnelle examine les gênes ou les protéines de façon isolée, la biologie des systèmes étudie de façon simultanée l'interaction de nombreux agents, pour tenter d'expliquer le fonctionnement global du système biologique [Kitano, 2001]. Les réseaux de régulations génétiques, ou plus généralement les réseaux biologiques d'interactions visent ainsi à modéliser, comprendre, les interactions entre les agents impliqués dans différents mécanismes biologiques ou biochimiques. Ces agents peuvent être l'ADN, l'ARN, des protéines, ou encore des molécules lors de phénomènes biochimiques.

2.1.2 Calculs distribués

Comme cela est argumenté par exemple dans [Alur et al., 2001], les circuits génétiques et les réseaux biomoléculaires partagent beaucoup de caractéristiques avec les systèmes informatiques distribués, ou embarqués.

En particulier, au plan intracellulaire, les agents précédents communiquent d'une certaine façon entre eux, en s'influençant l'un et l'autre, comme les noeuds d'un système distribué communiquent et s'influencent. Au plan intercellulaire, certaines communications existent aussi.

La description de l'état d'un système à un instant donné correspond à se donner des caractéristiques comme la concentration de chacune des espèces, ou la probabilité d'être dans tel état. La dynamique est alors souvent décrite en termes ces quantités par certaines équations différentielles [Gibson and Mjolsness, 2000], [de Jong, 2002].

Par la présence de non-linéarités dans la plupart des systèmes considérés, ces dynamiques sont souvent sujettes à des transitions discrètes entre phases continues. D'une certaine façon, ces transitions peuvent être vues comme des changements de phases des agents, comme les agents d'un système distribué peuvent changer d'état en fonction de l'état de leurs proches voisins.

Les modèles considérés sont donc proches des modèles des systèmes distribués actuels, en particulier des systèmes hybrides. Comme cela est mentionné par exemple dans [Alur et al., 2001], [Lincoln and Tiwari, 2004], [Batt et al., 2005], cela ouvre la voie à l'utilisation de méthodes formelles pour la preuve de propriétés sur ces systèmes.

Cependant, par rapport aux problèmes classiques en vérification, où le programme des agents impliqués est fixé et où l'on cherche à déterminer la validité d'une propriété, ici le problème est souvent au niveau de comprendre ou de construire le programme des agents impliqués. De ce point de vue, les méthodes formelles peuvent aider, en écartant les mauvais modèles. C'est pour cela par exemple que les approches symboliques s'avèrent ainsi complètement pertinentes [Batt et al., 2006] pour l'aide à l'analyse de systèmes. On est en fait plus proche du problème de la synthèse de systèmes, que de la vérification de systèmes, pour utiliser la terminologie de la vérification des systèmes continus et hybrides.

2.1.3 Niveaux de description

Au niveau des modèles utilisés pour décrire la dynamique des systèmes, en suivant la classification de [Gibson and Mjolsness, 2000], on peut distinguer les modèles

1. discrets
2. continus
3. stochastiques

Cette classification est grossière, et il existe des approches hybrides intermédiaires entre chacune des classes [Gibson and Mjolsness, 2000], [de Jong, 2002]. On trouvera un très joli et clair panorama de tous les modèles utilisés pour les réseaux de régulations génétiques dans [de Jong, 2002].

Le passage d'une classe à la suivante correspond à une compréhension et modélisation croissante du système. Intrinsèquement les systèmes sont stochastiques. Cependant, l'approche stochastique est souvent difficile à étudier, et l'on y préfère l'approche continue, ou encore l'approche discrète lorsque cette dernière reste difficilement traitable [Gibson and Mjolsness, 2000], [de Jong, 2002].

Focalisons nous sur l'approche continue au sens de la classification précédente [Gibson and Mjolsness, 2000] : les systèmes sont décrits par des équations différentielles sur des quantités comme des concentrations.

Avant d'en parler, faisons juste observer que l'approche stochastique, correspond aussi à des équations différentielles : l'approche stochastique correspond souvent (voir [Gibson and Mjolsness, 2000], [de Jong, 2002]) à utiliser la technique de l'équation maîtresse [Gillespie, 1977], c'est-à-dire à parler de lois de probabilités, et à décrire l'évolution de ces lois de probabilités aussi par des équations différentielles.

2.1.4 Modèles par dynamiques continues

Si nous mettons de côté la question (difficile) de comment nos amis biologistes ou biochimistes arrivent à déterminer les réseaux logiques d'interactions entre tous ces agents, et le problème hautement non-trivial de l'estimation de tous les paramètres quantitatifs impliqués, à un réseau logique d'interactions, on peut associer la dynamique résultante sur les agents, de façon relativement simple.

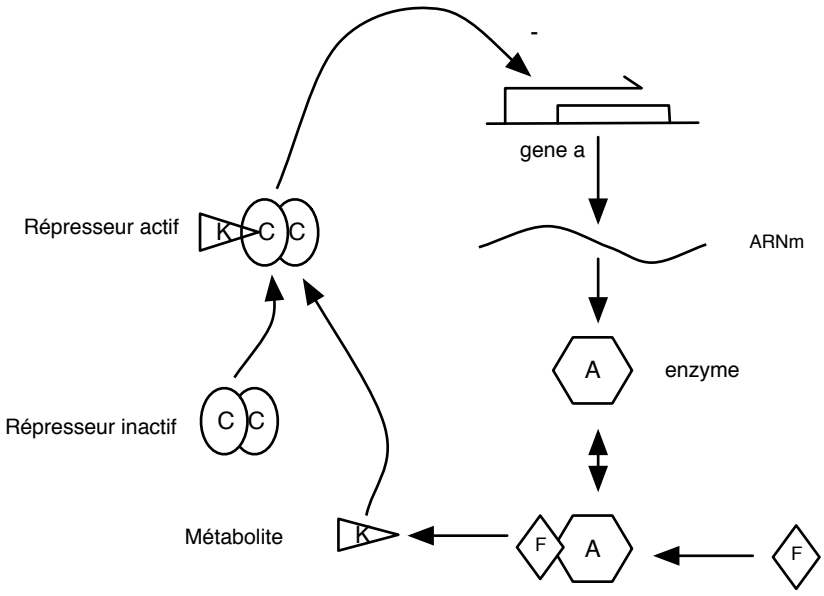


FIG. 2.1 – Exemple de réseau de régulations génétiques impliquant une inhibition produit terminale. Exemple tiré de [de Jong, 2002], adapté de [Goodwin, 1963], [Goodwin, 1965].

Par exemple, au réseau d'interactions de la figure 2.1 (exemple tiré de l'article [de Jong, 2002], adapté de [Goodwin, 1963], [Goodwin, 1965]), on peut associer une dynamique donnée par

$$\left\{ \begin{array}{lcl} x_1' & = & k_1 r(x_3) - \gamma_1 x_1 \\ x_2' & = & k_2 x_1 - \gamma_2 x_2 \\ x_3' & = & k_3 x_2 - \gamma_3 x_3 \end{array} \right. \tag{2.1}$$

où x_1, x_2, x_3 représentent respectivement les concentrations de l'ARNm a, la protéine A, et du métabolite K. Ici $k_1, k_2, k_3, \gamma_1, \gamma_2, \gamma_3$ sont des constantes, $r : \mathbb{R} \to \mathbb{R}$ une fonction décroissante non-linéaire à valeur entre 0 vers 1.

Le passage de la description du réseau à la dynamique n'est pas réellement la partie problématique. D'une certaine façon, la description du réseau est un "programme" d'un système dynamique, et la "compilation" vers un système dynamique n'est pas la partie difficile.

Une vraie difficulté est de décrire ce réseau, i.e. de déterminer le "programme". C'est le problème majeur pour nos amis les biologistes. Pour les informaticiens, un problème majeur est de faire quelque chose du système dynamique obtenu. En effet, lorsque le nombre d'agents devient grand, le système dynamique obtenu devient intraitable, que ce soit pour en faire une simulation, ou toute analyse qualitative ou quantitative. Le nombre de variables est tout simplement trop grand, et la dynamique trop complexe, le nombre d'indéterminations sur les constantes impliquées trop grand pour réellement en faire quelque chose.

Cela est une des raisons des différentes approches pour simplifier ces dynamiques, comme par exemple le remplacement de fonctions non-linéaires par des fonctions linéaires par morceaux [Lincoln and Tiwari, 2004], la simplification de la dynamique en une dynamique affine par morceaux [Ghosh and Tomlin, 2001], éventuellement symboliques [Batt et al., 2006], l'utilisation de modèles plus simples que les systèmes dynamiques à temps continu [de Jong, 2002], etc...

Il est fascinant de voir comment ces modèles de régulations génétiques correspondent à des descriptions logiques de systèmes intrinsèquement continus, et qu'il sera peut être un jour possible de passer de l'étape de compréhension des fonctions, à l'étape utilisation de ces fonctions pour la programmation de fonctions

biologiques, plutôt que l'explication de fonctions biologiques.

2.2 Modèles de populations

Les modèles de la biologie des populations visent à décrire la dynamique des populations. Faisons une rapide présentation de certains de ces modèles, en nous inspirant essentiellement de [Murray, 2002], et quelques considérations de [Hirsch et al., 2003] sur le cas à espèce unique.

2.2.1 Modèles à une espèce

Avant de parler de plusieurs espèces d'individus en interactions, il nous est indispensable de discuter comment croît naturellement une population constituée d'un seul type d'individus.

Le modèle le plus simple consiste à supposer que le taux de croissance de la population (i.e. x'/x) est constant, on obtient l'équation

$$x' = x\lambda. \tag{2.2}$$

C'est le modèle proposé par Malthus en 1798. Cependant, dès que $\lambda > 0$, la population croît sans limites, ce qui n'est pas très raisonnable.

Il est alors plus naturel de supposer que l'environnement à une capacité limitée N, et que lorsque le nombre d'individu est supérieur à cette capacité limite, la mortalité l'emporte sur la natalité.

C'est la version continue du *modèle logistique* de Verhulst en 1838 (rappelons que la version discrète a été discutée dans le chapitre précédent).

$$x' = \lambda x(1 - x/N). \tag{2.3}$$

La population tend asymptotiquement vers N. Nous renverrons notre lecteur à [Murray, 2002] pour la preuve expérimentale de la validité de ce modèle, en relation avec de vraies données issues de la biologie.

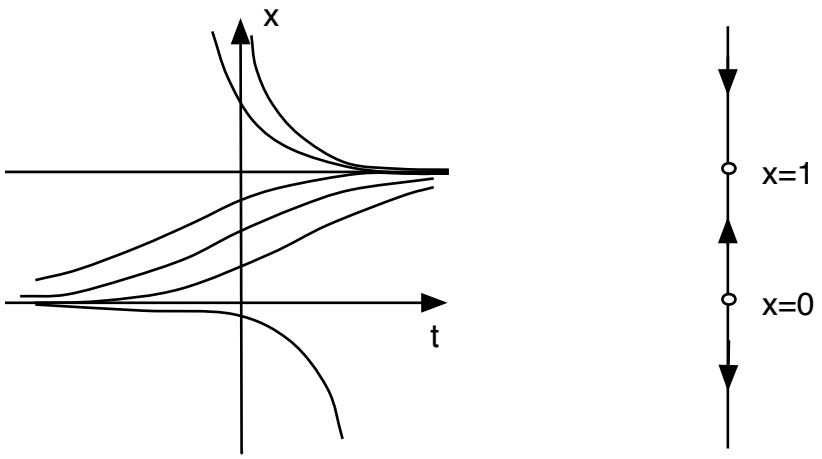


FIG. 2.2 – A gauche : quelques trajectoires de $x' = x(1 - x)$. A droite : ligne de phase.

2.2.2 Présence de chasseurs

Posons $N = 1$. Supposons qu'un adversaire (par exemple un chasseur) tue les individus avec un taux h. On obtient l'équation

$$x' = x(1 - x) - h. \tag{2.4}$$

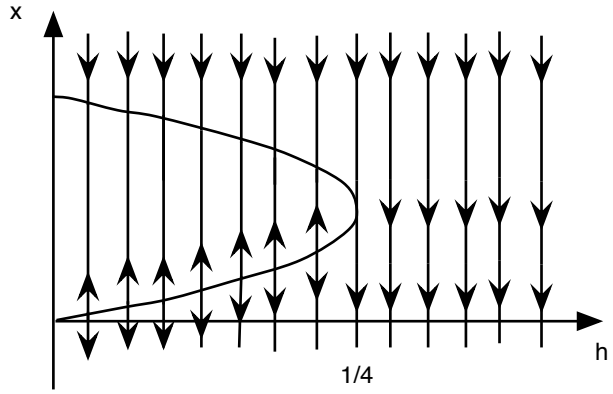


FIG. 2.3 – Diagramme de bifurcation de $x' = x(1 - x) - h$.

Comme observé dans [Hirsch et al., 2003], il se produit un phénomène de bifurcation : pour $0 < h < 1/4$, si la population est en dessous d'un certain seuil elle n'est pas suffisamment nombreuse pour se reproduire. Dès que ce seuil est dépassé, elle converge vers une population limite. Pour $1/4 \leq h$, la population disparaît ultimement.

On observe donc un passage abrupt, à $h = 1/4$ entre une dynamique ou la population subsiste, à la disparition complète de la population. Ce type de phénomène de bifurcations est souvent discuté pour la modélisation de phénomènes de disparitions soudaine d'espèces en biologie, comme la catastrophe écologique qui a résulté de l'introduction de la perche du Nil dans le lac Victoria dans les années 1960 discutée dans [Murray, 2002]. Nous renvoyons notre lecteur à [Murray, 2002] pour d'autres exemples de tels phénomènes.

2.2.3 Phénomènes saisonniers

Continuons l'exercice de modélisation, comme dans [Hirsch et al., 2003]. Il est raisonnable de supposer que la chasse est un phénomène saisonnier. Le modèle devient par exemple

$$x' = x(1 - x) - h(1 + \sin(2\pi t)). \tag{2.5}$$

Le système possède toujours un phénomène de bifurcation avec disparition de la population pour h plus grand qu'un certain seuil. En dessous de ce seuil, il y a apparition de trajectoires closes périodiques, avec une trajectoire périodique attractive, et une trajectoire périodique répulsive [Hirsch et al., 2003] : voir la figure 2.4.

En d'autres termes, la présence de saisons introduit des oscillations.

2.2.4 Phénomènes spatiaux

Supposons maintenant que l'on veuille modéliser le fait que la population n'est pas homogène, et se déplace naturellement.

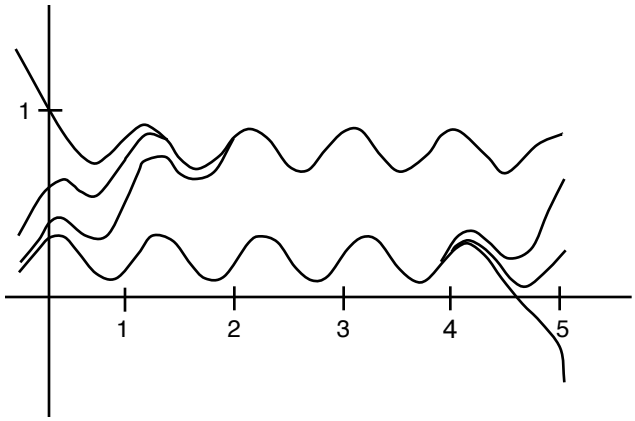


FIG. 2.4 – Quelques solutions de $x' = 5x(1-x) - 0.8(1+\sin(2\pi t))$.

En suivant les arguments classiques, repris par exemple dans l'excellent ouvrage [Murray, 2002], cela correspond à remplacer l'équation différentielle ordinaire précédente $u' = f(u)$ par une équation aux dérivées partielles, avec un terme de diffusion, pour devenir

$$\frac{\delta u}{\delta t} = f(u) + D\frac{\delta^2 u}{\delta u^2}. \tag{2.6}$$

Le système se met à posséder des solutions qui correspondent à des vagues qui se propagent : la population se déplace à une certaine vitesse. Nous renvoyons à [Murray, 2002], [Murray, 2003] pour de très intéressantes discussions sur les phénomènes spatiaux, et leur pendant expérimental observé sur de vraies données.

2.2.5 Modèles à plusieurs espèces

Oublions maintenant les effets spatiaux ou saisonniers, et les chasseurs, et supposons maintenant qu'il y ait deux types d'espèces : une population de prédateurs dénotée par y et une population de proies dénotée par x. Supposons que la population de proies soit la seule nourriture disponible pour les prédateurs, Supposons qu'en l'absence de prédateurs, i.e. quand $y = 0$, les proies croissent selon $x' = ax$. Lorsque des prédateurs sont présents, la population de proies décroît avec un taux proportionnel au nombre de prédateurs proies rencontrés. Le modèle le plus simple pour cela est une décroissance du taux de croissance d'un facteur $-bxy$. Faisons les hypothèses opposées pour les prédateurs. En l'absence de proies, ils décroissent avec un taux $-c$. En présence de proies, ils croissent à un taux proportionnel au nombre de prédateurs proies rencontrés.

On obtient le système suivant

$$\left\{ \begin{array}{rcl} x' & = & x(a-by) \\ y' & = & y(-c+dx). \end{array} \right. \tag{2.7}$$

C'est précisément le modèle proposé par Volterra [Volterra, 1931], déjà discuté précédemment. Qu'il corresponde au modèle de Lotka [Lotka, 1920], motivée par la chimie, n'est pas si étonnant : discuter de population d'individus, ou de molécules est, au niveau de modèles, assez équivalent, nous y reviendrons.

Le système possède donc des trajectoires closes, qui correspondent aux courbes de niveau d'une certaine fonction H.

Plus généralement, les dynamiques de population, dans le cas général de deux populations, mènent à des dynamiques du type

$$\left\{ \begin{array}{rcl} x' & = & M(x,y)x \\ y' & = & N(x,y)y. \end{array} \right. \tag{2.8}$$

où M et N sont des fonctions des deux variables.

Nous laisserons à notre lecteur le soin de deviner le cas de n populations en interactions. Par exemple, l'équation de Lotka-Volterra n-dimensionnelle est un système d'équations du type

$$x_i' = x_i(b_i - \sum_j a_{i,j} N_j). \tag{2.9}$$

2.3 Modèles de la virologie

2.3.1 En biologie

Les modèles de diffusions d'épidémies en biologie sont aussi des modèles de populations particuliers.

Par exemple, le modèle SIR découpe la population d'individus en trois types d'individus. Les individus S, pour susceptibles d'être infectés, I pour infectés, R pour guéris. Il porte le nom de SIR puisqu'un individu susceptible peut devenir infecté, puis un infecté peut devenir guéri. Le modèle $SIRS$ est sur le même principe, si ce n'est qu'on ajoute le fait qu'un individu guéri peut redevenir susceptible, et ainsi de suite pour différents acronymes de modèles comme SIS où un susceptible peut devenir guéri, et un guéri susceptible.

On trouvera un panorama des modèles dans le survol [Hethcote, 2000], ou dans la monographie [Murray, 2002].

Si l'on suppose qu'on a affaire à une maladie comme la malaria où les gens guéris ne peuvent pas retomber malade, on est dans le cas SIR, et on obtient le modèle

$$\left\{ \begin{array}{rcl} S' & = & -\beta SI \\ I' & = & \beta SI - \nu I \\ R' & = & \nu I. \end{array} \right. \tag{2.10}$$

où S, I, R désignent le nombre d'individus dans chaque population, avec $S' + I' + R' = 0$, i.e. une population constante.

Cette façon de formaliser avec des acronymes S, I, R est parfois appelée le modèle de Kermack-McKendrick. Le modèle SIR a été proposé historiquement pour expliquer la diffusion de la peste à Londres en 1665-1666, à Bombay en 1906, ou du choléra à Londres en 1865.

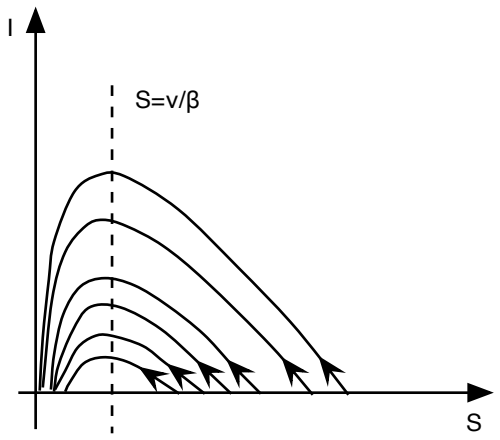


FIG. 2.5 – Portrait de phase du modèle SIR

Une étude mathématique simple du modèle montre que le paramètre important est le *seuil épidémique* donné par $R = \beta S/\gamma$. Si ce seuil est strictement inférieur à 1, l'infection finira par disparaître. S'il est strictement supérieur à 1, l'infection va se répandre sans limites.

Si l'on considère des maladies ou l'on peut retomber malade, on obtient le modèle $SIRS$, dont la dynamique est similaire en introduisant une nouvelle constante modélisant l'intensité des passages de l'état R vers l'état S : voir [Murray, 2002], [Hethcote, 2000].

Les maladies encore plus complexes, comme le SIDA, où il y a des états comme la séropositivé, mène à des modèles plus complexes, mais toujours selon le même principe, avec un nombre de classes plus grand : voir [Murray, 2002], [Hethcote, 2000].

2.3.2 En informatique

En informatique, la modélisation de la diffusion épidémique des virus informatiques se base essentiellement sur les mêmes modèles, au détail près qu'il semble nécessaire de prendre en compte la dynamique de protection résultante d'une infection, ou les phénomènes de surinfection (saturation du système par le virus lui-même) non présents en biologie.

Parmi les premiers modèles, il y a [Kephart and White, 1991], qui illustre bien que pour parler de propagation de virus, comme dans les modèles classiques de populations, le principe est d'abstraire des individus pour parler de proportions.

Cet article qui fait intervenir la topologie du réseau en la supposant probabiliste selon une certaine loi, montre ainsi clairement que choisir un graphe sous-jacent aléatoire revient à abstraire les interactions en termes de probabilités qu'un individu d'un type donné infecte un individu d'un autre type. Au final, l'article retombe sur le modèle SIS de virologie en biologie lorsque les graphes sont de degré moyen constant. Des simulations sont faites pour comprendre ce qui se passe sur des familles de graphes plus proches des graphes rencontrés en pratique sur les réseaux.

Le modèle SIS est assez critiquable en virologie informatique, car une fois infecté, et un antivirus administré, un programme ne redeviendra pas malade du même virus. Le modèle SIR plus haut est donc nettement plus pertinent. L'article [Staniford et al., 2002] modélise ainsi la diffusion du code rouge I par le modèle logistique à temps continu.

Le modèle est ensuite étendu légèrement dans [Zou et al., 2002] pour introduire un modèle à deux facteurs, permettant de modéliser le principe que le taux d'infection β décroît en réalité avec le temps, puisque le nombre de contre-mesures ou de protections fait que les gens ont tendance à se protéger. Cela revient d'une certaine façon à considérer un modèle $SIRQ$, où Q désigne des individus, avec une dynamique du type

$$\left\{\begin{array}{lcl} S' & = & -\beta(t)SI - \mu SJ \\ I' & = & \beta(t)SI - \nu I \\ R' & = & \nu I \\ Q' & = & \mu SJ \\ \beta(t) & = & \beta_0(1 - I/(S+I+R+Q))^\alpha. \end{array}\right. \tag{2.11}$$

En prenant en compte la structure hiérarchique de l'Internet, en systèmes autonomes (AS), les auteurs de [Serazzi and Zanero, 2003] proposent un modèle compartimenté. Si on note a_i la proportion d'hôtes infectés dans l'AS numéro i, et N_i le nombre d'hôtes susceptibles dans cette AS, on obtient une dynamique du type

$$a_i' = (\sum_{j=1}^{n} N_j a_j)(1 - a_i)K/N, \tag{2.12}$$

où K est une constante.

En fait, dans ce modèle, si on pose $a = (\sum_{j=1}^{n} N_j a_j)/N$, on obtient

$$\left\{\begin{array}{lcl} a' & = & aK(1-a) \\ a_i' & = & a(1-a_i)K, \end{array}\right. \tag{2.13}$$

et donc, a suit un modèle logistique. Chacun des a_i est une dynamique indépendante convergente vers la valeur limite 1 de a.

Le modèle est ensuite étendu dans [Serazzi and Zanero, 2003] pour prendre en compte la saturation de la bande passante du réseau, lors de la contamination.

Citons quelques alternatives. L'article [Zou et al., 2003] tente d'étudier un modèle probabiliste de la diffusion de virus par Email, en se basant sur des lois observées en pratique sur la topologie des réseaux sous-jacents, des temps de lectures des mails, etc...L'étude relève plutôt de la simulation, que de modèles analytiques. L'article [Chen et al., 2003] présente un modèle discret de la transmission de vers. Ici encore l'étude relève plutôt de la simulation que de modèles analytiques. D'autre part, plusieurs des hypothèses derrière le modèle, en particulier l'intérêt de discrétiser, semblent discutables : cela est critiqué explicitement par exemple dans [Serazzi and Zanero, 2003].

2.3.3 Phénomènes spatiaux

Dans tous les modèles qui précèdent, la modélisation des phénomènes spatiaux conduit, selon les arguments classiques, à l'ajout d'un terme de diffusion à chacune des équations de dynamique : voir [Murray, 2002], [Murray, 2003].

Cela est en fait vrai dans tous les modèles qui précèdent, que ce soit des modèles de biologie des populations ou de virologie, même si cela n'a pas été fait explicitement à notre connaissance dans le domaine de la virologie informatique.

À ce moment encore, il est très instructif d'observer les phénomènes spatiaux de vagues d'infection qui se propagent, à la fois dans les modèles et dans les données expérimentales : voir [Murray, 2002], [Murray, 2003] pour la biologie. Le volume [Murray, 2003] est consacré entièrement aux phénomènes spatiaux en biologie.

2.4 Modèles de la théorie des jeux

2.4.1 Introduction à la théorie des jeux

La théorie des jeux est une des façons de modéliser les situations de concurrence. Elle vise à prédire vers quelle(s) situation(s) doivent se placer un ensemble de partenaires rationnels en situation de concurrence. Nous renvoyons notre lecteur à la référence classique [Osbourne and Rubinstein, 1994] pour une présentation en anglais, ou à [Binmore, 1999] en français.

On peut faire remonter certaines de ses idées au 18ième siècle, mais le développement principal de la théorie remonte dans les années 1920 avec les travaux d'Emile Borel et de John von Neumann. Un évènement décisif a été la publication du livre [von Neumann and Morgenstern, 1944] en 1944 par John von Neumann et Oskar Morgenstern, qui a fixé la terminologie et la présentation des problèmes encore utilisées à ce jour. À cette époque, l'attention était essentiellement sur les jeux à somme nulle, à deux joueurs. Une certaine attention était portée sur les jeux coopératifs, mais les jeux non coopératifs étaient largement ignorés.

Dans les années 50, John Nash prouvait dans [Nash, 1950] que tout jeu possède une situation d'équilibre mixte, dite *d'équilibre de Nash*, dans laquelle aucun joueur n'a intérêt unilatéral de s'écarter. Ce concept majeur d'équilibre pour les jeux a été un point central de la plupart de ses développements depuis lors. L'attention est maintenant essentiellement sur les jeux non coopératifs.

La théorie des jeux a commencé dans ces années à être utilisée en économie et en sciences politiques, mais c'est seulement vers les années 1970 qu'elle a réellement provoqué une révolution en économie. Elle a aussi de nombreuses applications en biologie, sociologie et psychologie. Les économistes restent parmi les plus gros utilisateurs de ses concepts.

La reconnaissance de l'intérêt pour l'économie des apports de la théorie des jeux est attestée par l'attribution de plusieurs prix Nobel : prix Nobel d'économie en 1994 à John Nash, John Harsanyi et Reinhard Selten ; celui de 2005 à Thomas Schelling et à Robert Aumann.

Une autre attestation de sa reconnaissance a été l'enrôlement de théoriciens des jeux éminents pour la conception des enchères pour l'attribution des fréquences du spectre électromagnétique aux USA dans les années 1990 : voir [McMillan, 1994].

La théorie algorithmique des jeux, qui connaît un certain essor depuis l'exposé de Christos Papadimitriou en 2001 [Papadimitriou, 2001], se distingue de cette théorie mathématique, aux constructions parfois purement existentielles, par la prise en compte des aspects algorithmiques et de complexité dans ses constructions.

Ses domaines d'applications sont variés : elle vise à comprendre la complexité de problèmes liés aux calculs des équilibres [McKelvey and McLennan, 1996], les pertes de performances provoquées par les comportements individuels en algorithmique distribuée : voir [Papadimitriou, 2001], [Anshelevich et al., 2003] et [Correa et al., 2004], les problèmes de tarification de services dans certains protocoles [Feigenbaum et al., 2001], les problèmes liés à la théorie des enchères [Briest et al., 2005], la conception de mécanismes incitatifs en algorithmique [Nisan and Ronen, 1999], etc...

2.4.2 Concepts de base de la théorie des jeux

Présentons les concepts les plus simples de la théorie des jeux. Nous nous focaliserons sur les jeux non coopératifs, à information complète, en forme extensive.

Le jeu le plus simple est un jeu à deux joueurs I et II avec un ensemble fini d'options, appelées *stratégies pures*, $Strat(I)$ et $Strat(II)$. Notons par $a_{i,j}$ (respectivement : $b_{i,j}$) le gain (ou si cela est une variable aléatoire sa moyenne) pour le joueur I (resp. II) lorsque I utilise la stratégie $i \in Strat(I)$ et II la stratégie $j \in Strat(II)$.

Les gains sont donc donnés par des matrices $n \times m$ A et B, où n et m sont les cardinalités de $Strat(I)$ et $Strat(II)$.

Exemple 1 (Jeu des prisonniers) *Le cas où A et B sont les matrices suivantes*

$$A = \begin{pmatrix} 3 & 0 \\ 5 & 1 \end{pmatrix}, B = \begin{pmatrix} 3 & 5 \\ 0 & 1 \end{pmatrix}$$

est appelé jeu des prisonniers. *On note C (pour coopération) la première stratégie, et D (pour défection) la deuxième stratégie de chaque joueur.*

La *stratégie mixte* du joueur I qui consiste à utiliser $i \in Strat(I)$ avec probabilité x_i sera noté par le vecteur $\mathbf{x} = (x_1, \ldots, x_n)^T$. On doit avoir $\sum_{i=1}^{n} x_i = 1$, i.e. $\mathbf{x} \in S_n$, où S_n est le simplexe unitaire de $\mathbb{R}^n$, engendré par les vecteurs $\mathbf{e}_i$ de la base unitaire standard de $\mathbb{R}^n$.

De façon similaire, une stratégie mixte pour II correspond à $\mathbf{y} = (y_1, \ldots, y_m)^T$ avec $\mathbf{y} \in S_m$.

Si le joueur I utilise la stratégie mixte $\mathbf{x}$, et le joueur II la stratégie mixte $\mathbf{y}$, alors le premier a le gain moyen $\mathbf{x}^T A\mathbf{y}$ et le deuxième $\mathbf{x}^T B\mathbf{y}$.

Une stratégie $\mathbf{x} \in S_n$ est dite une meilleure réponse à la stratégie $\mathbf{y} \in S_m$, noté $\mathbf{x} \in BR(\mathbf{y})$ si

$$\mathbf{z}^T A\mathbf{y} \leq \mathbf{x}^T Ay \tag{2.14}$$

pour toute stratégie $\mathbf{z} \in S_n$.

Une paire $(\mathbf{x}, \mathbf{y})$ est un *équilibre de Nash mixte* si $\mathbf{x} \in BR(\mathbf{y})$ et $\mathbf{y} \in BR(\mathbf{x})$. Le théorème de Nash de [Nash, 1950] affirme, par un argument de point fixe, qu'un tel équilibre mixte existe toujours. Cependant, il n'est pas nécessairement unique.

En d'autres termes, deux stratégies $(\mathbf{x}, \mathbf{y})$ forment un équilibre de Nash, si cet état est tel qu'aucun des deux joueurs n'a intérêt unilatéral à s'en écarter.

Un équilibre de Nash en stratégies pures, c'est-à-dire avec $\mathbf{x} \in \{0,1\}^n$, $\mathbf{y} \in \{0,1\}^m$, n'existe pas toujours.

Exemple 2 *Sur l'exemple du jeu des prisonniers, $BR(\mathbf{y}) = (0,1)$ pour tout $\mathbf{y}$, et $BR(\mathbf{x}) = (0,1)$ pour tout $\mathbf{x}$. Donc (D,D) est l'unique équilibre de Nash, et il est pur. Dans celui-ci chacun des joueurs gagne 1. Le paradoxe est que si ils avaient joués (C,C) (la coopération) ils auraient gagner 3, soit plus. L'optimum social, (C,C), diffère de l'équilibre atteint par les joueurs rationnels (D,D), car en tout autre état chacun craint que l'adversaire ne joue C.*

Il est important de comprendre que la théorie des jeux n'est pas du tout une théorie visant à parler de dynamisme, mais plutôt d'équilibres en présence de joueurs rationnels. En fait, on suppose profondément que chaque jeu n'est joué qu'une seule et unique fois, et la théorie vise à prédire ce qu'il faut jouer. Il est vrai qu'il est très déroutant d'interpréter ce que signifie une stratégie mixte, c'est-à-dire des probabilités, concrètement, si on ne joue qu'une fois : voir la discussion dans le début du livre [Osbourne and Rubinstein, 1994].

Pour introduire un certain dynamisme en théorie des jeux, il existe deux grandes approches. La première consiste à répéter les jeux. La seconde à utiliser les modèles de la théorie évolutionnaire des jeux.

2.4.3 Jeux répétés

Répéter k fois un jeu, revient à étendre l'espace des choix en $Strat(I)^k$ et $Strat(II)^k$: le joueur I (respectivement II) choisit son action $\mathbf{x}(t) \in Strat(I)$, (resp. $\mathbf{y}(t) \in Strat(II)$) au temps t pour $t = 1, 2, \cdots, k$. Cela est donc équivalent à un jeu à deux joueurs avec respectivement n^k et m^k choix pour chacun.

Conformément à [Binmore, 1999], pour éviter les confusions, nous nommerons *actions* les choix $\mathbf{x}(t), \mathbf{y}(t)$ de chacun à un instant donné, et *stratégies* les suites $X = \mathbf{x}(1), \cdots, \mathbf{x}(k)$ et $Y = \mathbf{y}(1), \cdots, \mathbf{y}(k)$, c'est-à-dire les stratégies pour le jeu global.

Si le jeu est répété un nombre infini de fois, une stratégie devient une fonction des entiers vers les actions, et le jeu reste équivalent à un jeu à deux joueurs[1].

On comptabilise en général le gain d'un joueur avec un taux d'escompte $\delta > 0$ (que l'on peut prendre égal à 1, si k est fini ; on choisit $\delta < 1$ si k est infini). Concrètement, le gain du joueur I (respectivement II) avec les stratégies X,Y est comptabilisé comme

$$Gain(X, Y) = \sum_{i=1}^{k} \delta^{i-1} gain(i), \tag{2.15}$$

où $gain(t)$ est le gain du joueur au temps t, c'est-à-dire $\mathbf{x}(t)^T A \mathbf{y}(t)$ pour le joueur I (resp. $\mathbf{x}(t)^T B \mathbf{y}(t)$ pour le joueur II).

Répéter un nombre fini et fixe de fois un jeu n'apporte rien en général aux comportements espérés. Par exemple, un simple raisonnement de récurrence en arrière montre que chacun des joueurs a toujours intérêt à jouer D aux temps $n, n-1, \cdots, 1$ dans le dilemme des prisonniers itéré, si celui-ci est répété un nombre fini k, connu par chacun, de fois : voir [Binmore, 1999].

Il est plus intéressant de répéter le jeu un nombre infini de fois. Cette fois, la stratégie de coopération systématique (c'est-à-dire $\mathbf{x}(t) = \mathbf{y}(t) = (1, 0)^T =$ "C" pour tout t) devient un équilibre de Nash pour le jeu répété : en d'autres termes, coopérer devient intéressant pour chacun.

Observons que le gain comptabilisé par l'équation 2.15 possède une autre interprétation simple : si l'on suppose que l'on répète le jeu un nombre fini, mais aléatoire, de fois le jeu, avec à chaque étape une probabilité δ que le jeu continue encore une étape de plus, et $1 - \delta$ que ce soit la dernière étape, alors la probabilité que l'on joue la $i$ème étape vaut δ^i. En moyenne, le gain de chacun des joueurs est donc donné par l'équation 2.15 : voir [Binmore, 1999].

Le problème est que la stratégie de coopération systématique n'est pas le seul équilibre de Nash. La stratégie de défection systématique reste un équilibre de Nash du jeu répété. En fait, on montre que pour tout point (x, y) de la zone grisée de la figure 2.6, il y a une paire de stratégie (X, Y) qui garantit le gain x pour le joueur I, y pour le joueur II, et tel que (X, Y) soit un équilibre de Nash pour le jeu répété : voir [Binmore, 1999]. En d'autres termes, le nombre d'équilibres de Nash du jeu répété est indénombrable.

2.4.4 Comportements

En pratique, le joueur I (respectivement II) est confronté au problème suivant à chaque instant t : étant donnée l'histoire du jeu jusque là, c'est-à-dire, $X_{t-1} = \mathbf{x}(1), \cdots, \mathbf{x}(t-1)$ et $Y_{t-1} = \mathbf{y}(1), \cdots, \mathbf{y}(t-1)$ que jouer au temps t ? c'est-à-dire comment choisir $\mathbf{x}(t) \in Strat(I)$? (resp. $\mathbf{y}(t) \in Strat(II)$?)

[1] mais dont les matrices sont infinies.

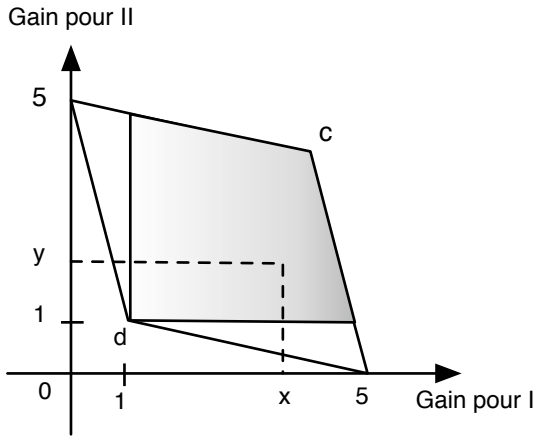


FIG. 2.6 – Pour tout point (x, y) de la zone grisée, il y a une paire X, Y de stratégies qui garantit le gain x pour le joueur I, y pour le joueur II, et tel que (X, Y) soit un équilibre de Nash pour le dilemme des prisonniers itéré. En particulier pour le point c qui correspond à la coopération systématique, et pour le point d qui correspond à la défection systématique.

Il est naturel de supposer que la réponse de chacun est donnée par certaines règles de comportement : $\mathbf{x}(t) = f(X_{t-1}, Y_{t-1})$, $\mathbf{y}(t) = g(X_{t-1}, Y_{t-1})$ où f (respectivement g) est une fonction qui détermine le comportement de I (resp. II).

La question de la meilleure règle de comportement à utiliser pour le dilemme des prisonniers itéré a fait couler beaucoup d'encre. En particulier, suite au livre [Axelrod, 1984], qui décrit le résultat de tournois de règles de comportements pour le dilemme des prisonniers itéré, et qui argumente qu'il existe un meilleur comportement, nommée $TIT - FOR - TAT$.

Le comportement $TIT - FOR - TAT$ consiste à coopérer à la première étape, puis ensuite à faire la même chose que son adversaire aux temps ultérieurs. Nous renvoyons notre lecteur à [Beaufils, 2000], [Labbani, 2003] pour des études critiques de ces résultats. De nombreux autres comportements, souvent avec des noms très imagés, ont été proposés et étudiés : voir [Axelrod, 1984], [Beaufils, 2000], [Labbani, 2003].

2.5 Dynamiques issues de jeux

2.5.1 Jeux sur un graphe

Un exemple de comportement est le comportement $PAVLOV$.

Exemple 3 (Comportement $PAVLOV$) *Le comportement $PAVLOV$ consiste dans le dilemme des prisonniers itéré, à fixer un seuil, disons 3, et à l'instant t, à rejouer l'action pure précédente si le gain est au dessus du seuil, et sinon à changer d'action.*

Concrètement, si l'on note $+$ pour C, $-$ pour D, on vérifiera facilement que cela donne les règles

$$\begin{cases} ++ & \to & ++ \\ +- & \to & -- \\ -+ & \to & -- \\ -- & \to & ++, \end{cases} \tag{2.16}$$

où le membre gauche de chaque règle dénote $\mathbf{x}(t-1)\mathbf{y}(t-1)$, et le membre droit ce qui en résulte pour $\mathbf{x}(t)\mathbf{y}(t)$.

Le comportement $PAVLOV$ est Markovien : un comportement f est *Markovien*, si $f(X_{t-1}, Y_{t-1})$ ne dépend que de $\mathbf{x}(t-1), \mathbf{y}(t-1)$.

À partir d'un ensemble de telles règles, il est facile d'établir une dynamique distribuée. Par exemple, suivons [Dyer et al., 2002].

Supposons qu'on ait un graphe connexe $G = (V, E)$, avec N sommets. Les sommets correspondent à des joueurs. Une configuration instantanée du système est donnée par un élément de $\{+, -\}^N$, c'est-à-dire par l'état + ou − de chaque sommet. Il y en a donc 2^N configurations.

À chaque instant t, on tire au hasard uniformément une arête du graphe (i, j). À ce moment-là, les joueurs i et j jouent le jeu du dilemme des prisonniers avec le comportement $PAVLOV$, c'est-à-dire les règles de l'équation 2.16.

Quel est l'état final atteint par le système ?

Le modèle sous-jacent est une énorme chaîne de Markov avec 2^N états. L'état $E^* = \{+\}^N$ est absorbant. Si le graphe G ne possède pas de sommet isolé, c'est l'unique état absorbant, et il existe une suite de transformations qui transforme n'importe quel état E en cet état E^*.

Par conséquent, quelle que soit la configuration initiale, avec probabilité 1, le système finira par être dans l'état E^* : voir [Brémaud, 2001]. Le système est *autostabilisant*.

On trouvera plusieurs résultats sur le temps de convergence vers cet état stable dans [Dyer et al., 2002], et [Fribourg et al., 2004], pour les anneaux et les graphes complets.

Ce qui est intéressant dans cet exemple est qu'il montre comment passer d'un jeu, et d'un type de comportement à une dynamique distribuée sur un graphe. Clairement, il est facile d'associer une dynamique similaire à n'importe quel comportement[2] Markovien sur un jeu symétrique.

Une question très intéressante est la suivante :

Question 5 *Quelles sont les dynamiques distribuées qui correspondent à un jeu ?*

Nous pensons la question intéressante pas uniquement pour cette façon d'associer une dynamique distribuée à un jeu, mais pour les différents moyens naturels d'associer une dynamique à un jeu.

Il existe ainsi d'autres moyens naturels d'associer une dynamique à des jeux, que ce soient des jeux à 2-joueurs, ou des jeux distribués avec des affrontements par paires comme dans la section précédente.

2.5.2 Jeux spatiaux

Tout d'abord, discutons de jeux distribués particuliers sur la grille, ne seraient-ce que parce que ceux-ci sont bien connus et très souvent cités.

Les articles [May et al., 1995] et [Nowak and May, 1992] discutent du jeu des prisonniers itéré joué sur une grille bidimensionnelle. À chaque instant, chaque joueur joue avec ses voisins immédiats au dilemme des prisonniers. Le score d'un joueur est la somme des gains de ses jeux avec ses voisins. Au début de la génération suivante, le joueur choisit l'action du voisin avec le meilleur score parmi lui-même et ses voisins.

Ces articles s'intéressent ainsi expérimentalement à l'évolution de la coopération, et aux phénomènes comme les motifs spatiaux, et le chaos qui apparaissent dans ce jeu.

Un autre jeu spatial intéressant, ne serait-ce que par ses vertus pédagogiques est le jeu du solitaire de Schelling, présenté par son auteur par exemple dans [Schelling, 1978], [Schelling, 1971a]. Nous suivrons ici la présentation de [Binmore, 1999].

Il se veut un modèle (simpliste) de la ségrégation raciale et se joue sur un échiquier. Chaque case représente une maison, noire ou blanche. Chaque pion blanc souhaite qu'au moins la moitié de ses voisins soit blanc. Chaque pion noir souhaite qu'au moins un tiers de ses voisins soit noir. À chaque génération, un pion noir ou blanc qui ne voit pas sa contrainte satisfaite se déplace au point le plus près qui le satisfait. L'ordre dans lequel les pions mécontents sont déplacés n'est pas important, et l'on peut le supposer aléatoire. Le processus est prolongé jusqu'à ce que tous les pions satisfassent leurs contraintes.

[2] Pas nécessairement pavlovien. A vrai dire le comportement $PAVLOV$ comme décrit ici n'est pas ambigu seulement sur les matrices 2×2 comme le dilemme des prisonniers.

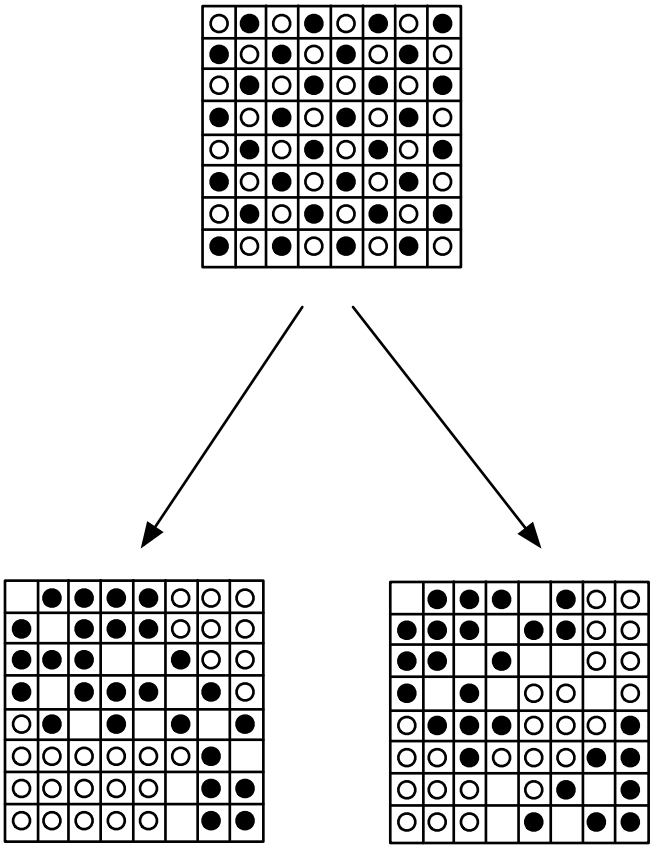


FIG. 2.7 – Le solitaire de Schelling

Si l'on débute avec des configurations comme sur la figure 2.7 en haut, par simulation, on termine avec des configurations typiquement comme celles sur la figure 2.7 en bas, c'est-à-dire avec une forte ségrégation raciale. En d'autres termes, les choix individuels de chacun, qui ne sont pas nécessairement oeuvres d'une ségrégation manifeste, mènent à une situation de forte ségrégation globale. L'équilibre social qui résulte de préférences individuelles n'est pas celui que chacun désire réellement en son sein.

Nous renvoyons aux articles [Schelling, 1971a], [Schelling, 1978], [Schelling, 1969], [Schelling, 1971b] pour l'étude de variantes sur différents paramètres, et à l'intense littérature citant ces articles pour des discussions sur les implications sociales ou philosophiques de tels phénomènes.

2.5.3 Dynamique Myope

Dans le cas général, à tout jeu à 2 joueur répété, on peut associer le comportement *myope*. Il consiste à ce qu'un joueur fasse systématiquement l'hypothèse que le joueur opposé rejouera au temps t la même chose qu'au temps $t-1$. Par conséquent, ce comportement consiste au temps t a choisir systématiquement la (ou une) meilleure réponse à l'action du joueur adverse au temps précédent :

$$f(X_{t-1}, Y_{t-1}) \in BR(\mathbf{y}(t-1)).$$

Prenons comme [Binmore, 1999], l'exemple du jeu d'une duopole de Cournot. Le modèle du duopole de Cournot est un modèle économique de la concurrence entre deux producteurs d'un même article. Dans

ce modèle, la production d'un article coûte c. On fait l'hypothèse que la demande totale est de la forme $q = q_1 + q_2 = M - p$, où p est le prix de vente, et q_1 et q_2 le nombre d'articles produits par chacune des firmes.

Le problème de la firme I (respectivement II) est de fixer q_1 (resp. q_2) de telle sorte à maximiser son bénéfice $(p-c)q_1$ (resp. $(p-c)q_2$). On montre facilement (voir [Binmore, 1999]), que la meilleure réponse à q_2 est de poser

$$q_1 = 1/2(M - c - q_2), \tag{2.17}$$

et que la meilleure réponse à q_1 est de poser

$$q_2 = 1/2(M - c - q_1), \tag{2.18}$$

de telle sorte que l'équilibre de Nash unique corresponde à l'intersection des droites définies par les équations 2.17 et 2.18.

La dynamique myope pour les deux joueurs donne alors sur ce jeu

$$\left\{ \begin{array}{lcl} q_1(t) & = & 1/2(M - c - q_2(t-1)) \\ q_2(t) & = & 1/2(M - c - q_1(t-1)). \end{array} \right.$$

Il est facile de montrer que quel que soit le point initial, une telle dynamique converge vers l'équilibre de Nash. La dynamique collective converge vers l'équilibre rationnel.

Malheureusement, comme le montre [Binmore, 1999] ce n'est pas toujours le cas : inverser par exemple les courbes de réactions sur ce jeux, mène à un système dynamique qui converge vers certains des équilibres de Nash pour certaines conditions initiales, mais qui parfois ne converge pas (oscille) pour d'autres conditions initiales.

2.5.4 Dynamique du joueur fictif

Le comportement myope peut sembler vraiment trop basique. Un comportement plus raisonnable semble le suivant : pour prédire ce que va faire le joueur adverse au temps t, utilisons la statistique de ce qu'il a fait au temps $1, 2, \cdots, t-1$: s'il a joué l'action i n_i fois, estimons qu'il jouera l'action i avec probabilité $x_i = n_i/(t-1)$ au temps t. Il s'agit de que l'on nomme la *dynamique du joueur fictif*.

Pour simplifier l'étude, suivons [Binmore, 1999], et supposons que $n = m = 2$, et que les matrices sont données par

$$A = \begin{pmatrix} 0 & 3 \\ 2 & 1 \end{pmatrix}, B = \begin{pmatrix} 2 & 0 \\ 1 & 3 \end{pmatrix}.$$

Si aux temps $1, 2, \cdots, t-1$ le joueur II a utilisé n_i fois l'action numéro i, le joueur I estimera que le joueur II jouera au temps t l'action i avec probabilité $y_i(t) = n_i/(t-1)$. Le joueur II évaluera la probabilité $x_i(t)$ que le joueur I joue l'action i de façon symétrique.

Pour étudier la dynamique, comme le montre [Binmore, 1999], il suffit de passer d'un temps discret à un temps continu.

Une analyse simple (voir [Binmore, 1999]) montre que aussi longtemps que $(x_2(t), y_2(t))$ reste dans la zone A de la partie gauche de la figure 2.8, le joueur I voudra utiliser se seconde stratégie pure, et le joueur II sa première stratégie pure.

La dynamique $(x_2(t), y_2(t))$ restera donc dans cette zone jusqu'à l'instant $t + \tau$ pour $\tau > 0$ suffisamment petit. Puisqu'on connaît le choix du joueur II entre le temps t et $t+\tau$, on peut donc évaluer $y_2(t+\tau)$ comme

$$y_2(t + \tau) = \frac{t y_2(t)}{t + \tau}. \tag{2.19}$$

Ce qui s'écrit encore

$$\frac{y_2(t + \tau) - y_2(t)}{\tau} = -y_2(t).$$

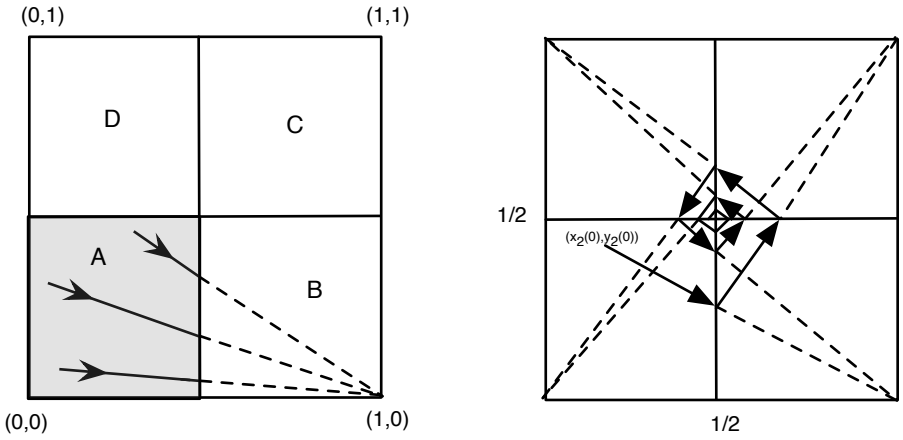


FIG. 2.8 – Convergence vers un équilibre mixte.

En faisant tendre τ vers 0, on obtient

$$y_2'(t) = \frac{y_2(t)}{t}.$$

De façon similaire, on obtient

$$x_2'(t) = \frac{1 - x_2(t)}{t}.$$

Les points qui satisfont ces deux équations se situent sur une ligne droite qui part de $(x_2(t), y_2(t))$ et qui joint le point $(1, 0)$. Une étude similaire sur les zones B,C, et D de la partie gauche de la figure 2.8 montre que la dynamique doit être celle de la partie droite de la figure 2.8. Elle converge donc vers l'équilibre de Nash mixte du jeu. Là encore, la dynamique collective converge vers l'équilibre rationnel.

Malheureusement, là encore, ce n'est pas le cas sur tous les jeux : on peut facilement considérer des jeux où les trajectoires ne convergent pas, ou avec des cycles limites [Binmore, 1999].

2.5.5 Dynamique par libration sociale

Toujours selon [Binmore, 1999], considérons le même jeu, mais supposons maintenant que nous avons une grande population de joueurs. Nous avons deux types de joueurs, garçons et filles. Les garçons seront les joueurs I, et les filles les joueurs II. Chaque joueur, quel que soit son sexe, choisit de jouer à jamais la stratégie pure 1 ou la stratégie pure 2. On suppose qu'à tout instant, on fait disparaître certains joueurs : pendant n'importe quelle durée T, la probabilité qu'un joueur quitte la population est λT.

On remplace chaque joueur qui disparaît, par un joueur dont l'action est la meilleure réponse à la situation globale statistique du sexe opposé : si dans la population courante une fraction $y_i(t)$ (respectivement $x_i(t)$) d'individus fille (resp. garçon) utilise l'action i, alors le garçon (resp. fille) injecté jouera l'action qui est la meilleure réponse à la stratégie mixte $(y_1(t), y_2(t))$ (resp. $(x_1(t), x_2(t))$). C'est-à-dire l'action qui lui procurera le meilleur bénéfice s'il rencontre aléatoirement un joueur du sexe opposé dans un jeu deux à deux.

On montre (voir [Binmore, 1999]) que l'on doit avoir lorsqu'on est dans la zone A

$$x_2'(t) = -\lambda x_2(t) + \lambda,$$

et

$$y_2'(t) = -\lambda y_2(t).$$

La dynamique est différente de celle de la section précédente, cependant les points qui satisfont ces deux équations se situent toujours sur une ligne droite qui part de $(x_2(t), y_2(t))$ et qui joint le point $(1, 0)$.

Une étude similaire sur les zones B,C, et D de la partie gauche de la figure 2.8 montre que les trajectoires sont identiques à celles de la section précédente (mais pas la vitesse de déplacement sur ces trajectoires) : voir [Binmore, 1999].

2.6 Théorie évolutionnaire des jeux

La théorie évolutionnaire des jeux est une autre façon d'associer des dynamiques à des jeux.

2.6.1 Motivation biologique

La théorie des jeux évolutionnaire est née du livre de Maynard Smith [Maynard-Smith, 1981]. Ce livre présente quelques applications de la théorie des jeux à la biologie.

Pour illustrer comment, à un jeu, on peut associer une dynamique biologique reprenons l'exemple fictif d'une population d'individus de [Binmore, 1999]. Binmore choisit d'appeler ces individus des *dodos*.

La journée d'un dodo dure une fraction τ d'une année. Il y a n types de dodos : les dodos qui jouent l'action 1, les dodos qui jouent l'action 2, ..., les dodos qui jouent l'action n.

On s'intéresse à la proportion $x_i(t)$ de dodos qui jouent l'action i. On a bien entendu $\sum_{i=1}^{n} x_i(t) = 1$.

À la fin de chaque journée les dodos se combattent deux à deux. L'issue des combats influe sur leur fécondité. On lit sur une matrice A $n \times n$ à l'entrée $a_{i,j}$ la fécondité du dodo s'il est du type i et si lors du combat, il a été confronté avec un individu du type j : son nombre espéré d'oisillons dodo au jour suivant est $\tau a_{i,j}$.

Combien un dodo de type i peut-il espérer avoir d'oisillons dodo au jour suivant ? La réponse est

$$\sum_{j=1}^{n} x_j(t)\tau a_{i,j} = (A\mathbf{x})_i \tau.$$

En effet, puisque les appariements pour les combats le soir entre dodos sont choisis uniformément parmi la population, en moyenne sa fécondité est donnée par l'expression précédente.

Le nombre de dodos de type i le lendemain matin sera donc

$$N x_i(t)(1 + (A\mathbf{x})_i \tau).$$

La mortalité n'étant pas fonction du type du dodo, le lendemain, la fraction de dodos de type i sera donnée par

$$x_i(t+\tau) = \frac{N x_i(t)(1 + (A\mathbf{x})_i \tau)}{N x_1(t)(1 + (A\mathbf{x})_1 \tau) + \cdots + N x_n(t)(1 + (A\mathbf{x})_n \tau)}.$$

Soit

$$x_i(t+\tau) = \frac{x_i(t)(1 + (A\mathbf{x})_i \tau)}{1 + \mathbf{x}^T A\mathbf{x}\tau}.$$

où $\mathbf{x}^T A\mathbf{x}\tau$ s'interprète comme le nombre moyen de naissances par dodos en un jour.

Cela peut encore s'écrire

$$\frac{x_i(t+\tau) - x_i(t)}{\tau} = x_i(t)\frac{(A\mathbf{x})_i - \mathbf{x}^T A\mathbf{x}}{1 + \mathbf{x}^T A\mathbf{x}\tau}.$$

En passant à la limite quand τ tend vers 0, on obtient

$$x_i' = x_i((A\mathbf{x})_i - \mathbf{x}^t A\mathbf{x}).$$

C'est ce qu'on appelle *une équation de réplication*. Une telle équation modélise que les populations d'individus dont le gain (fitness) donnée par la matrice A est supérieure au gain (fitness) moyen ont tendance à se reproduire. Les autres à disparaître.

2.6.2 Objectifs de cette théorie

Le modèle parlant de dodos est relativement ad hoc, mais de nombreuses situations et modèles mènent au même type de dynamiques : nous renvoyons à [Maynard-Smith, 1981] pour cela.

La théorie évolutionnaire des jeux vise à étudier le comportement de telles dynamiques en fonction de la matrice A.

Elle possède ses propres notions d'équilibres, motivées par la stabilité des systèmes dynamiques sous-jacents, comme la notion d'équilibre évolutionnairement stable. Un équilibre évolutionnairement stable est un équilibre de Nash particulier.

Elle permet de relier les notions d'équilibre pour le jeu donné par A, aux notions de stabilité pour le système dynamique correspondant.

À vrai dire, elle ne considère pas seulement des dynamiques de réplication, mais aussi d'autres dynamiques comme les dynamiques d'imitation, de meilleure réponse, avec à chaque fois dans ces dynamiques l'idée que les individus ayant un plus grand gain (fitness) dans la matrice de jeu se reproduisent plus vite que les autres.

Nous renvoyons par exemple à [Maynard-Smith, 1981], [Weibull, 1995], [Hofbauer and Sigmund, 2003] pour une présentation de ses principaux résultats.

2.7 Algorithmique distribuée

2.7.1 Modèles et machines classiques

Nous en venons à notre motivation reliée à l'algorithmique distribuée.

Il existe plusieurs modèles en algorithmique distribuée, suivant le type de machines physiques considéré, et les contraintes associées : par exemple les types de communications utilisées, communications par messages, par variables partagées, etc..., l'existence ou non de mémoire partagée, les priorités entre lecture/écriture en cas de conflit, etc...

Cependant, de façon abstraite, ces modèles consistent à dire que l'ensemble des agents, correspond à un ensemble d'automates[3] communiquants particuliers [Lynch, 1997], [Tel, 1994].

En général, une hypothèse très forte dans tous ces modèles est que l'on suppose une topologie fixée. Il y a un graphe sous-jacent tel que les communications entre agents ne peuvent se faire qu'entre voisins dans le graphe. Effectivement, en pratique, les communications ne se font bien qu'entre voisins physiquement connectés, ou alors, comme dans les réseaux ad hoc, le graphe est dynamique, mais il varie peu localement.

Dans l'évaluation de la complexité des solutions, un paramètre important est le nombre de processeurs impliqués. Effectivement, en pratique, le nombre de processeurs physiquement existants est limité, et acheter de nouveaux processeurs a un coût.

Nous renvoyons à [Lynch, 1997], [Tel, 1994] pour une présentation de l'algorithmique distribuée classique, et de ses méthodes et à [Balcàzar et al., 1990] pour une présentation des modèles vus de l'angle de la complexité.

2.7.2 Vers une nouvelle algorithmique distribuée

Cependant, nous pensons qu'il est nécessaire de remettre en cause certaines de ces hypothèses.

Premièrement, la taille de certains réseaux est maintenant telle, qu'il semble nécessaire de ne plus parler d'individus, mais de statistiques. En effet, par exemple, lorsqu'on décrit l'état d'un réseau comme Internet, il n'est pas pertinent de décrire l'état d'un individu donné, mais plutôt de la proportion d'individus dans tel ou tel état. Autre exemple : lorsque l'on décrit comme [Adar and Huberman, 2000] le comportement d'éventuel *Free Riding* des agents dans une application pair à pair de téléchargement de fichiers, le seul moyen pertinent de le faire est de parler de proportions d'agents qui ont tel comportement plutôt qu'un autre.

Nous pensons ainsi qu'il est temps de raisonner en termes de statistiques, ou de proportions, c'est-à-dire avec des modèles comme ceux qu'utilisent les biologistes, les chimistes, pour décrire leurs systèmes avec un grand nombre d'agents (individus, molécules, etc...).

[3] Ou très rarement de machines de Turing

D'autre part, plusieurs développements envisagés autour de l'algorithmique distribuée remettent fortement en cause les hypothèses classiques : voir la discussion dans [Angluin et al., 2004] pour les réseaux de capteurs. En effet, on considère que le coût actuel de production de capteurs est maintenant tellement faible, que le nombre de capteurs impliqués n'est pas si important. D'autre part, dans certaines applications, les capteurs peuvent être conduits à ne pas contrôler quels sont leurs voisins : par exemple, lors d'un largage par avions de capteurs [Alberganti, 2006], on ne peut pas prédire à priori qui sera voisin de qui, ou lorsque la RATP équipe ses clients de capteurs [Alberganti, 2006], on ne peut pas prédire quel utilisateur viendra rencontrer quelle borne de lecture.

Par le même raisonnement que celui de l'auteur de [Kephart and White, 1991] lorsqu'il parle de virus informatiques dont on ne contrôle pas à priori les contacts avec des applications saines, ou par ceux fait par les chimistes, les biologistes, et les physiciens, pour établir tous les modèles précédents, cela revient à considérer statistiquement qu'il n'y a pas de topologie, ou au mieux, une statistique sur la topologie.

2.7.3 De la description vers la programmation

Les modèles des sections précédentes permettent de décrire des populations en interactions. Cela permet, bien entendu, d'étudier la dynamique d'un système comme Internet, ou la dynamique des utilisateurs de telle ou telle application.

Cependant, nous pensons qu'un enjeu est de passer de ces modèles en tant qu'outil pour décrire des populations, à leur utilisation pour la programmation.

Par exemple, comme les biologistes utilisent les modèles de diffusion d'épidémies pour anticiper la vague de propagation et déterminer la zone à traiter la plus économique [Murray, 2003], il est envisageable de lutter contre la diffusion de virus par le traitement anticipé local de certaines zones, ou par la diffusion d'un contre-virus utilisant la même faille que le virus pour se répandre.

Question 6 *Peut-on utiliser les modèles dynamiques précédents pour "programmer" des systèmes en concurrence, et non pas seulement pour les décrire ?*

Nous allons expliciter cette question par des exemples plus loin.

D'autre part, si cela est possible, cela légitime la question suivante.

Question 7 *Quelle est la puissance des modèles de la bioinformatique, de la biologie des populations, de la virologie, ou de la théorie évolutionnaire des jeux, des sections précédentes ?*

Nous avons à chaque fois affaire à des modèles de systèmes en concurrence. Pour beaucoup d'entre eux, il n'est pas raisonnable de supposer que l'on programme tous les individus, mais seulement une fraction d'entre eux. Les modèles de la théorie des jeux, et de la théorie évolutionnaire des jeux prennent alors tout leur intérêt, puisque l'on cherche à programmer un comportement global, à l'aide de comportements locaux, en présence de concurrence.

C'est pour ces raisons que nous sommes fortement impliqués (nous en sommes le coordinateur) dans l'action de recherche amont SOGEA autour de la théorie des jeux et des aspects dynamiques.

2.7.4 Protocoles de populations de Angluin et al

Ce type de questions mène de nouveau à comprendre la puissance, c'est-à-dire ce qu'on peut calculer, ou ne pas calculer, avec chacun des modèles.

Pour présenter un exemple de type de modèles qui va dans le sens de notre discussion, et les problèmes associés, nous allons présenter le modèle récent de *protocoles de populations* [Angluin et al., 2004] issu de l'algorithmique pour les réseaux de capteurs.

Dans ce modèle, un protocole consiste à se donner un ensemble fini d'états internes $Q = \{1, 2, \ldots, k\}$, et une règle de transition $\delta : Q \times Q \to Q \times Q$. Pour $\delta(p, q) = (p', q')$, écrivons $\delta_1(p, q) = p'$, $\delta_2(p, q) = q'$.

Une configuration d'un système à un instant donné est donnée par les états internes de chacun des n individus.

On suppose les individus complètement identiques et indiscernables. Il en suit que l'état d'un système peut aussi se décrire par le nombre n_i d'individus dans l'état i, pour $1 \leq i \leq k$, plutôt que par l'état de chacun.

À chaque top discret, un unique individu i se voit mis en relation avec un autre individu j : a l'issue de de cette rencontre l'individu i est dans l'état $\delta_1(q_i, q_j)$, et l'individu j est dans l'état $\delta_2(q_i, q_j)$.

On suppose que l'on ne contrôle pas les interactions, et qu'il y a une certaine équité : si de la configuration C on peut aller vers la configuration C' en une étape (noté $C \to C'$) alors dans toute dérivation $C_0 C_1 \cdots$, avec $C_i \to C_{i+1}$ pour tout i, si C apparaît infiniment souvent, alors C' aussi.

On veut considérer certains protocoles de populations comme des reconnaisseurs de prédicats $\psi : \mathbb{N}^m \to \{0, 1\}$.

Pour cela, fixons sous-ensemble $Q^+ \subset Q$, et on dit qu'un uple $(n_1, \ldots, n_m) \in \mathbb{N}^m$, pour $m \leq k$, est accepté (respectivement : refusé) par le protocole, si partant de toute configuration avec n_i individus dans l'état i, ultimement tous les individus seront dans un état interne qui appartient à Q^+ (respectivement : son complémentaire), et cela sera vrai à tout instant ultérieur.

On dit que le protocole reconnaît $\psi : \mathbb{N}^m \to \{0, 1\}$ si pour tout uple $(n_1, \ldots, n_m)$, celui-ci est accepté lorsque $\psi(n_1, \ldots, n_m) = 1$, et celui ci est refusé lorsque $\psi(n_1, \ldots, n_m) = 0$.

On a le résultat très joli suivant (rappelons que les ensembles définissables en arithmétique de Presburger coïncident avec les ensembles semi-linéaires sur les entiers).

Théorème 9 ([Angluin et al., 2006b])
- *Tout prédicat $\psi : \mathbb{N}^m \to \{0, 1\}$ définissable en arithmétique de Presburger est calculé par un protocole de population.*
- *Réciproquement tout prédicat $\psi : \mathbb{N}^m \to \{0, 1\}$ calculé est définissable en arithmétique de Presburger*

Par exemple, puisque cela est définissable en arithmétique de Presburger, il existe un protocole pour décider s'il y a plus que 5% d'agents dans l'état 1.

Ce théorème montre, si besoin est, que ces modèles différent grandement des modèles classiques, comme les automates cellulaires. En effet, les automates cellulaires restent très proches des machines de Turing : il est facile de simuler une machine de Turing par un automate cellulaire. Par contre, le lien entre machines de Turing et ce type de modèles est loin d'être direct.

Nous renvoyons notre lecteur à [Angluin et al., 2005a], [Angluin et al., 2005b], [Angluin et al., 2006a] pour de nombreux résultats sur ce modèle, et sur quelques variantes.

2.7.5 Protocoles de populations

Le modèle précédent est motivé par les réseaux de capteurs. Mais, même s'il permet de décider des problèmes sur des proportions, comme "décider s'il y a plus que 5% d'agents dans l'état 1", le modèle n'a pas à faire avec des proportions, mais des nombres d'individus dans chaque état.

Si le nombre d'individus est grand, il est naturel de ne plus parler de nombres d'individus mais de proportions, ou de statistiques.

Par exemple, on peut considérer le protocole suivant : on a une population de n agents. Chaque agent est soit dans l'état $+$, soit dans l'état $-$. Une configuration correspond donc à un point de $S = \{+, -\}^n$.

On suppose le temps discret. À chaque top discret, les agents sont appariés deux à deux, selon les règles suivantes :

$$\begin{array}{lcl} ++ & \to & 1/2+, 1/2- \\ +- & \to & + \\ -+ & \to & + \\ -- & \to & 1/2+, 1/2- \end{array}$$

Il faut interpréter la deuxième règle de la façon suivante : si un individu est du type $+$ et qu'il est apparié avec un individu du type $-$, il devient du type $+$. Il faut interpréter la première règle de la façon suivante : si un individu est du type $+$ et qu'il est apparié avec un individu du type $+$, il devient du type $+$ avec probabilité $1/2$, et $-$ avec la probabilité $1/2$. De même pour les deux autres règles.

On suppose que les appariements sont choisis au hasard uniformément.

Expérimentalement, la proportion de + dans la population tend vers $\sqrt{2}/2$, lorsque le nombre d'individus grandit.

Formellement, on pouvait s'y attendre, puisque si p désigne la proportion de +, avec probabilité p un individu rencontre un +, et $1-p$ un −. Maintenant, la première et quatrième règle détruisent en moyenne $1/2+$ chacune, alors que les secondes et troisièmes règles créent un + chacune. En faisant le bilan de +, on trouve qu'en moyenne on crée

$$1/2p^2 + 2p(1-p) + 1/2(1-p)^2 = 1/2 + p - p^2$$

or à l'équilibre, il y a conservation, et donc cela doit valoir p, d'où

$$p^2 = 1/2.$$

$$p = \sqrt{2}/2.$$

Question 8 *Le système précédent semble converger vers $\sqrt{2}/2$. Quels sont les nombres calculables par de tels protocoles ?*

Bien entendu, en utilisant des appariements deux à deux, des probabilités rationnelles, et un nombre fini d'états internes.

Nous conjecturons qu'il s'agit de tous les nombres algébriques de $[0,1]$, mais à cet instant précis, nous n'avons déjà pas la preuve formelle de la convergence du système précédent.

Question 9 *Peut-on caractériser comme dans le théorème précédent les prédicats calculables ?*

On dit qu'un prédicat est calculable, s'il capture la relation entre la proportion initiale et la proportion à la limite.

2.7.6 Pourquoi s'intéresser à de tels protocoles

Nous avons déjà évoqué plusieurs raisons pour lesquelles considérer ce type de systèmes nous semble intéressant.

Pour compléter, et valider l'intérêt de tous les modèles de cette section pour l'algorithmique distribuée, voici un joli exercice.

Question 10 *Parmi toutes les équations différentielles décrivant une dynamique dans ce chapitre, quelles sont celles qui correspondent bien à une dynamique d'un algorithme distribué ?*

On prendra algorithme distribué au sens d'un protocole de population, ou d'une variante simple à expliciter. On ne considérera pas nécessairement que le temps doit être discret, mais aussi le cas où on le prend continu. Par exemple, le système précédent de règles calculant $\sqrt{2}/2$ correspond la description par des règles d'une (énorme) chaîne de Markov à temps discret. Mais, en utilisant des appariements qui se produisent selon un processus de poisson, en temps continu, on calcule aussi $\sqrt{2}/2$. Probablement, que cela est plus naturel pour résoudre cet exercice.

Les gens de l'algorithmique distribuée nous reprocherons que l'hypothèse d'abstraction de la topologie dans les modèles que nous considérons est trop forte.

Nous prenons ces quelques lignes pour insister. Du moment, que le système global est tel que les proportions locales de configurations dans tel ou tel état ne dépend pas du point où l'on se place, c'est-à-dire qu'il n'y a pas de phénomènes spatiaux, au sens des modèles de populations précédents, même si topologie il y a, elle peut souvent être abstraite en termes statistiques.

Remarquons que s'il y a des phénomènes spatiaux, il est aussi possible d'utiliser les modèles de la dynamique des populations, avec ses méthodes, mais il s'agira d'équations aux dérivées partielles.

Chapitre 3

Théorie des calculs pour les systèmes à temps continu

Ce chapitre correspond à une traduction (avec une légère adaptation) d'un chapitre d'un livre, signé en collaboration avec Manuel Campagnolo, qui paraîtra chez Springer Verlag "New Computational Paradigms". Il s'agit d'un survol, issu de nombreuses discussions et d'échanges de points de vue avec Manuel, sur la théorie des calculs pour les systèmes à temps continu.

Nous présentons les théories des calculs des systèmes à temps continu. Ces théories permettent à la fois de comprendre la difficulté des questions relatives aux systèmes dynamiques à temps continu, et à la fois de comprendre la puissance des modèles analogiques à temps continu.

Nous résumons les résultats obtenus à ce jour, avec de nombreux pointeurs sur la littérature.

3.1 Introduction

Comme nous l'avons vu dans les chapitres 1 et 2, les systèmes dynamiques à temps continu apparaissent immédiatement dès que l'on tente de modéliser des systèmes qui évoluent sur un espace continu avec un temps continu. Ils peuvent même apparaître comme la description naturelle de systèmes à espace ou à temps discret. C'est une approche courante dans des domaines comme la biologie, la physique ou la chimie, où une grande population d'agents (comme des molécules ou des individus) est abstraite en des quantités réelles, qui peuvent être des proportions ou des données thermodynamiques [Hirsch et al., 2003], [Murray, 2002].

Il y a plusieurs approches qui ont mené à des théories pour les calculs à temps continu. Nous allons explorer en profondeur deux approches. La première, que nous nommerons *inspirée par les machines analogiques à temps continu*, possède ses racines dans les modèles de machines analogiques naturelles ou artificielles. La seconde, que nous nommerons *inspirée par les théories des systèmes à temps continu*, est d'un spectre plus large. Elle est issue de la recherche sur les systèmes à temps continu d'un point de vue de la calculabilité ou de la complexité. Les systèmes hybrides, les automates temporisés, ou les résultats de l'analyse récursive à propos de la calculabilité des solutions d'équations différentielles en sont par exemple des sources d'inspiration.

Un large domaine de problèmes reliés aux théories des calculs à temps continu est couvert par ces deux approches, dans des domaines comme la vérification (voir par exemple [Asarin et al., 1995]), la théorie du contrôle (voir par exemple [Branicky, 1995b]), la conception de circuits intégrés (voir par exemple [Mills et al., 2005]), les réseaux de neurones (voir par exemple [Orponen, 1997]) ou la théorie des calculs sur les réels (voir par exemple [Moore, 1996]).

À ses débuts, la théorie des calculs à temps continu se focalisait principalement sur les machines analogiques. Déterminer quels systèmes peuvent réellement êtres considérés comme des modèles de calculs est une question passionnante et intrigante, reliée la question philosophique de ce qu'est ou ce que n'est pas une machine programmable, ce qui en dehors du cadre de ce chapitre.

Néanmoins, une bonne part des toutes premières machines construites sont considérées comme des machines analogiques programmables. Cela inclut le célèbre Differential Analyser, construit pour la première fois sous la supervision de Vannevar Bush au MIT en 1931 [Bush, 1931], mais aussi le Finance Phalograph de Bill Phillips, et en dans un certain sens le planimètre d'Hermann en 1814, la Pascaline de Pascal en 1642, et même le mécanisme d'Anticythère de -87 avant JC : voir [Coward, 2006]. Les modèles de calculs à temps continu incluent aussi les réseaux de neurones formels, et plus généralement les systèmes qui peuvent êtres construits en utilisant de l'électronique analogique. Puisque les modèles continus apparaissent dès que l'on cherche à modéliser des grandes populations, nous pouvons spéculer qu'ils joueront un rôle prédominant comme modèles des systèmes massivement parallèles comme l'Internet [Papadimitriou, 2001] dans l'avenir : voir aussi le chapitre 2.

Le premier vrai modèle d'une machine à temps continu universelle est dû à Shannon [Shannon, 1941], qui l'a introduite comme un modèle du Differential Analyzer. Les années 50 et 60 ont donné lieu à la naissance d'une littérature importante sur la programmation de telles machines[1]. Il y a aussi une littérature relativement importante sur comment utiliser ces machines analogiques pour la résolution de problèmes discrets ou continus : voir par exemple [Vergis et al., 1986] et ses références. Cependant, la plupart de cette littérature est maintenant relativement peu pertinente, étant donnés les développements actuels de notre compréhension de la complexité et de la calculabilité.

La recherche sur les réseaux de neurones artificiels, bien qu'à ce jour essentiellement tournée vers les modèles à temps discret, a entraîné un changement de perspective, en raison des nombreuses relations établies avec les concepts modernes de la théorie de la calculabilité ou de la complexité [Orponen, 1997], [Orponen, 1994]. Une autre ligne de travaux a été motivée par les systèmes hybrides, en relation avec des questions sur la difficulté de leur vérification ou de leur théorie du contrôle : voir par exemple [Branicky, 1995b], [Asarin et al., 1995].

Les années récentes ont aussi vu un intérêt pour d'autres alternatives aux machines digitales classiques. Cela inclut les modèles à espace continu et à temps discret comme les réseaux de neurones artificiels [Orponen, 1997], les modèles optiques [Woods and Naughton, 2005], les machines basées sur des signaux [Durand-Lose, 2005] et le modèle de Blum Shub et Smale [Blum et al., 1998]. Plus généralement, cela inclue les modèles non classiques et plus ou moins réalistes ou futuristes comme les modèles d'automates cellulaires exotiques [Grigorieff and Margenstern, 2004], les calculs moléculaires ou naturels [Head, 1987], [Adleman, 1994], [Lipton, 1994], [Păun, 2002], les calculs par trous noirs [Hogarth, 1992], les calculs quantiques [Deutsch, 1985], [Gruska, 1997], [Shor, 1994], [Kieu, 2004].

Dans un sens, les modèles à temps discret sont relativement bien connus et compris grâce à la thèse de Church, qui affirme que tous les modèles raisonnables suffisamment puissant sont équivalents. Pour les systèmes à temps continu, la situation est loin d'être aussi claire, et il n'y a pas eu autant d'efforts pour tenter d'unifier les concepts, bien que certains résultats récents établissent l'équivalence entre des modèles apparemment distincts [Graça and Costa, 2003], [Graça, 2004], [Graça et al., 2005], et [Bournez et al., 2006a], et laissent envisager qu'une théorie unifiée des calculs à temps continu est possible dans le futur.

Ce texte peut être considéré comme une mise à jour du survol de Orponen en 1997 [Orponen, 1997]. Comme l'écrit Orponen à la fin de son introduction, nous pouvons toujours dire que l'effet de l'imprécision ou du bruit sur les calculs analogiques est loin d'être compris, et qu'une théorie de la complexité robuste pour les modèles à temps continu a toujours à être réellement inventée. Cependant, nous le verrons, de nombreux progrès ont été obtenus dans ces directions dans la dernière décade.

Ce chapitre est organisé comme suit. Dans la section 3.2, nous rappelons les modèles à temps continu principaux. Dans les sections 3.3 et 3.4, nous discutons respectivement de la calculabilité et de la complexité des calculs à temps continu. Dans ces sections, nous nous focalisons principalement sur les systèmes à temps continu. Dans la section 3.5, nous nous intéressons aux effets des imprécisions et du bruit sur les calculs analogiques. Finalement, la section 3.6, est une conclusion avec des pistes et des directions pour des recherches futures dans le domaine des calculs à temps continu.

[1] Voir par exemple le très instructif *Analog Computer Museum* de Doug Coward sur Internet [Coward, 2006], et toute sa bibliographie. Observons que cette littérature révèle un art relativement oublié de la programmation des machines à temps continu, qui n'a rien à envier à la programmation actuelle

3.2 Modèles à temps continu

Avec à l'esprit une perspective historique, nous soulignons dans cette section plusieurs classes majeures de modèles à temps continu qui ont mené à des intérêts pour ce domaine de recherche. Ces modèles illustrent aussi des concepts comme les dynamiques continues, ou les entrées et les sorties.

3.2.1 Modèles inspirés par des machines analogiques

Le General Purpose Analog Computer de Shannon

La machine à temps continu probablement la plus connue est le *Differential Analyzer*, construit au MIT pour la première fois sous la supervision de Vannevar Bush [Bush, 1931] en 1931. L'idée d'assembler des périphériques réalisant des intégrations pour résoudre des équations différentielles peut être attribuée à Lord Kelvin en 1876 [Thomson, 1876]. Des machines mécaniques[2], et plus tard des Differential Analyzer électroniques ont été intensivement utilisés pour résoudre différentes équations différentielles provenant de problèmes d'ingénierie : voir par exemple [Bowles, 1996], ou plus généralement [Williams, 1996] pour des comptes-rendus historiques. A partir des années 1960, et l'invention du transistor, les analyseurs différentiels ont été progressivement écartés en faveur de la technologie digitale.

La première étude théorique des capacités de calcul des machines universelles à temps continu est due à Shannon. Shannon a proposé en 1941 [Shannon, 1941] le *General Purpose Analog Computer (GPAC)* comme un modèle théorique du Differential Analyzer de Vannevar Bush. Le modèle raffiné ensuite dans la série d'articles [Pour-El, 1974], [Lipshitz and Rubel, 1987], [Graça and Costa, 2003], [Graça, 2004], consiste simplement en des familles de circuits construits à partir des unités de base de la figure 3.1. En général, tous les types d'interconnexions ne sont pas autorisés, puisque cela conduirait à des comportements indésirables : par exemple, à des sorties non uniques. Pour plus de détails et de discussions, se référer à [Graça and Costa, 2003] et [Graça, 2002].

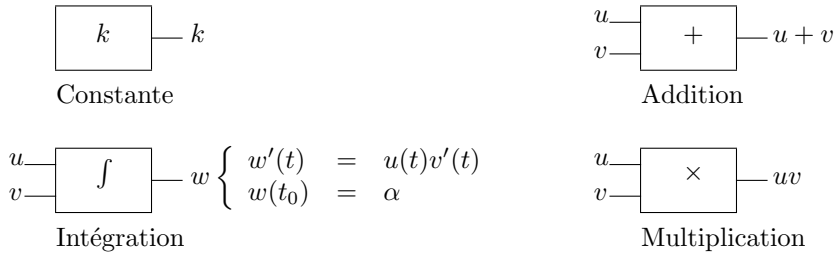


FIG. 3.1 – Les éléments de base d'un GPAC.

Shannon, dans son article original, mentionne déjà que les GPAC sait générer les polynômes, la fonction exponentielle, les fonctions trigonométriques, et leurs inverses. Plus généralement, Shannon prétend dans [Shannon, 1941] qu'une fonction est générée par un GPAC si et seulement si elle est différentiellement algébrique, c'est-à-dire si elle satisfait une équation différentielle algébrique de la forme

$$p\left(t, y, y', ..., y^{(n)}\right) = 0,$$

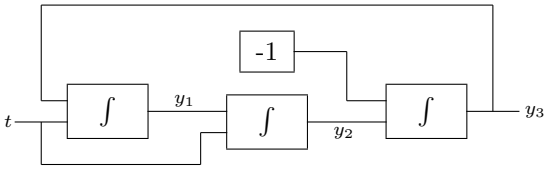
où p est un polynôme non nul en toutes ses variables. Par conséquent, si l'on note que la fonction Gamma $\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt$ ou la fonction Zeta de Riemann $\zeta(x) = \sum_{k=0}^\infty \frac{1}{k^x}$ n'est pas différentiellement algébrique [Rubel, 1989], il en suit que la fonction Gamma et la fonction Zeta sont des exemples de fonctions qui ne peuvent pas êtres générées par un GPAC.

[2]Et aussi en MECANO : voir [Bowles, 1996].

Cependant, la preuve de Shannon qui relie les fonctions générées par un GPAC avec les fonctions différentiellement algébriques est incomplète (comme montré et partiellement corrigé par [Pour-El, 1974], [Lipshitz and Rubel, 1987]). Pour la classe de GPAC plus robuste considérée dans [Graça and Costa, 2003], la propriété plus forte suivante est vérifiée : une fonction scalaire $f : \mathbb{R} \to \mathbb{R}$ est générée par un GPAC si et seulement si elle est composante d'une solution d'un système $y' = p(t, y)$ où p est un vecteur de polynômes (elle est projection d'une solution d'un problème de Cauchy polynomial selon la terminologie du chapitre 1). Une fonction $f : \mathbb{R} \to \mathbb{R}^k$ est dite générée par un GPAC si et seulement si ses composantes le sont.

La fonction Γ est en fait bien calculable par un GPAC, si une notion de calcul inspirée par celle qui est présente en analyse récursive est considérée [Graça, 2004]. Les fonctions GPAC calculables dans ce sens correspondent précisément aux fonctions calculables sur les réels : voir [Bournez et al., 2006a] ou le chapitre 4.

D'autres extensions du modèle original de GPAC de Shannon ont été considérées : en particulier Rubel a considéré dans [Rubel, 1993] le Extended Analog Computer (EAC), ou des opérations pour résoudre des problèmes aux bornes, ou pour prendre certaines limites infinies, ont été ajoutées. Nous renvoyons à [Mills, 1995] et à [Mills et al., 2005] pour la description d'implémentations électroniques du EAC de Rubel.



$$\begin{cases} y_1' = y_3 & \& \quad y_1(0) = 1 \\ y_2' = y_1 & \& \quad y_2(0) = 0 \\ y_3' = -y_1 & \& \quad y_3(0) = 0 \end{cases}$$

FIG. 3.2 – Génération de cos et sin par un GPAC : la version circuit est à gauche, et la version équation différentielle à droite. On a $y_1 = \cos$, $y_2 = \sin$, $y_3 = -\sin$.

Modèles de réseaux de Hopfield

Un autre modèle à temps continu bien connu est le modèle de réseaux de neurones artificiels proposé par John Hopfield en 1984 dans [Hopfield, 1984], qui peut s'implémenter avec des composants électroniques simples [Hopfield, 1984], ou par des périphériques optiques [Stoll and Lee, 1988].

Un réseau symétrique de Hopfield est constitué d'un nombre fini, disons n d'unités de calcul simple, ou *neurones*. L'architecture du réseau est donnée par un réseau (non orienté) dont les noeuds sont les neurones, et dont les arêtes sont étiquetées par des poids, les *poids synaptiques*. On peut supposer le graphe complet en remplaçant l'absence d'une connexion par une arête de poids nul.

L'état de chaque neurone i au temps t est donné par une valeur réelle $u_i(t)$. En partant d'un certain état initial $\mathbf{u}_0 \in \mathbb{R}^n$, la dynamique globale du réseau est définie par un système d'équations différentielles

$$C_i u_i'(t) = \sum_j W_{i,j} V_j - u_i/R_i + I_i,$$

où $V_i = \sigma(u_i)$, σ est une fonction saturante telle que $\sigma(u) = \alpha \tan u + \beta$, $W_{i,j} = W_{j,i}$ est le poids de l'arête entre i et j, et C_i, I_i, R_i sont des constantes [Hopfield, 1984].

Hopfield a montré dans [Hopfield, 1984], par un argument de fonction de Lyapunov, que de tels systèmes sont globalement asymptotiquement stables, i.e. de n'importe quel état initial, le système converge vers un état stable. En effet, si l'on considère par exemple la fonction énergie [Hopfield, 1984]

$$E = -\frac{1}{2} \sum_i \sum_j W_{i,j} V_i V_j + \sum_i \frac{1}{R_i} \int_0^{V_i} \sigma^{-1}(V) dV + \sum_i I_i V_i,$$

La fonction E est bornée et sa dérivée est négative. Par conséquent, l'évolution de l'état global du système est un déplacement dans l'espace des états à la recherche d'un minimum (qui peut être local) de E.

Ce comportement de convergence a été exploité par Hopfield pour de nombreuses applications, telles que des mémoires associatives, ou pour résoudre certains problèmes de décision comme le problème du voyageur de commerce [Hopfield, 1984], [Hopfield and Tank, 1985].

De tels réseaux sont en fait appelés réseaux de Hopfield en raison de leur analogie avec la version discrète du modèle introduite par le même auteur plus tôt. Cependant, la convergence de la dynamique continue peut aussi être vue comme une conséquence des résultats antérieurs de [Cohen and Grossberg, 1983] qui s'appliquent à des classes plus générales de systèmes dynamiques.

Une borne inférieure exponentielle sur le temps de convergence des réseaux de Hopfield à temps continu en fonction de leur dimension a été obtenue dans [Sima and Orponen, 2003]. De tels réseaux symétriques à temps continus peuvent simuler tout réseau de neurone récurrent à temps discret à états binaires, comme cela a été prouvé dans [Orponen and Sima, 2000], [Síma and Orponen, 2003a].

Réseaux de neurones à décharges

Selon [Maass, 1997b], si l'on classifie les modèles de réseaux de neurones artificiels en fonction de leur fonction d'activation et leur dynamique, on peut considérer qu'il y a trois générations. La première génération, basée sur les réseaux à seuil de McCulloch et Pitts, inclut les perceptrons multicouches, les réseaux de Hopfield symétriques à temps discret, et les machines de Boltzmann (voir par exemple [Abdi, 1994] pour une introduction à tous ces modèles). Dans cette génération, les sorties sont digitales. La seconde génération n'utilise pas des fonctions à seuil pour calculer les sorties, mais des fonctions d'activation continues. Elle inclut les réseaux de neurones sigmoïdaux récurrents, ou les réseaux de fonctions à bases radiales, ou encore les réseaux de Hopfield à temps continu. Maintenant, les entrées et les sorties deviennent analogiques. La troisième génération, qui augmente encore le niveau de réalisme biologique [Maass and Bishop, 1998], est basée sur des neurones à décharges, et encode ses variables dans des différences temporelles entre pulsations. Cette troisième génération peut exhiber des dynamiques à temps continu.

Parmi les modèles mathématiques des neurones à décharges, les propriétés calculatoires du modèle qui suit ont été relativement bien étudiées. Un réseau de neurones à décharges est donné par un graphe orienté fini. À chaque noeud v (neurone) du graphe est associée une fonction à seuil $\theta_v : \mathbb{R}^+ \to \mathbb{R} \cup \{\infty\}$, et à chaque arête (u, v) (synapse) est associée une fonction de réponse $\epsilon_{u,v} : \mathbb{R}^+ \to \mathbb{R}$ et une fonction de poids $w_{u,v}$.

Pour un neurone v qui n'est pas un neurone d'entrée, on définit son ensemble F_v de temps de décharges récursivement. Le premier élément de F_v est $\inf\{t | P_v(t) \geq \theta_v(0)\}$, et pour tout $s \in F_v$, le prochain plus grand élément de F_v est $\inf\{t | t > s \text{ and } P_v(t) \geq \theta_v(t-s)\}$ où

$$P_v(t) = 0 + \sum_u \sum_{s \in F_u, s<t} w_{u,v}(s)\epsilon_{u,v}(t-s).$$

Le 0 ci-dessus, ici pour garantir que P_v est bien défini même si $F_u = \emptyset$ pour tout u avec $w_{u,v} \neq 0$, peut être remplacé par une fonction de biais. Des hypothèses inspirées par la biologie mènent à des restrictions sur les fonctions de réponses et de biais autorisées : voir [Maass, 1997b], [Maass, 1999], [Natschläger and Maass, 2002], ou [Maass, 2002], [Maass, 2003], pour des discussions sur le modèle.

L'étude de la puissance de calculs de tels réseaux a été initiée par [Maass, 1996a] selon plusieurs variantes. Des extensions bruitées du modèle ont été considérées dans [Maass, 1996b], [Maass, 1997a], ou encore dans l'article [Maass and Natschläger, 2000]. Un aperçu des résultats de complexité obtenus peut être trouvé dans le survol [Síma and Orponen, 2003b]. Des restrictions motivées par l'implémentation matérielle des réseaux ont aussi été étudiées dans [Maass and Ruf, 1999].

Fonctions $\mathbb{R}$-récursives

Moore a proposé dans [Moore, 1996] une théorie des fonctions récursives, définie en analogie avec la théorie de la récursivité classique.

Une algèbre de fonctions

$$[B_1, B_2, ...; O_1, O_2, ...]$$

est le plus petit ensemble qui contient des fonctions de base $\{B_1, B_2, ...\}$ et qui est clos par certaines opérations $\{O_1, O_2, ...\}$, qui prennent une ou plusieurs fonctions dans la classe et en crée une nouvelle. Comme nous le verrons, ce modèle à temps continu a en particulier la capacité de résoudre des équations différentielles, de façon similaire à un intégrateur idéal dans un GPAC. En fait, la théorie des fonctions réelles récursives peut être vue comme une extension de la théorie de Shannon pour le GPAC.

Bien que des algèbres de fonctions aient été définies dans le contexte de la théorie de la récursivité sur les entiers, et aient été largement utilisées pour caractériser des classes de complexité ou de calculabilité [Clote, 1998], elles permettent aussi de définir des classes de fonctions récursives à valeurs réelles. Si les fonctions de base possèdent une certaine propriété (par exemple la continuité ou la différentiabilité) qui est préservée par les opérateurs, alors toute fonction de la classe possède cette propriété sur leur domaine de définition.

En théorie de la récursivité sur les réels, des opérations comme la composition, l'intégration, la recherche de zéros sur les réels, ou le passage à la limite de fonctions réelles ont été considérées. L'intégration est une opération qui, étant données les fonctions f et g d'arités appropriées, définit la solution du problème de Cauchy $h(x, 0) = f(x)$ et $\partial_y h(x, y) = g(x, y, h)$ sur son domaine d'existence.

Il est naturel de se demander si les fonctions dans l'algèbre sont calculables au sens de la calculabilité classique ou de l'analyse récursive. Comme attendu, la recherche de zéro ou les passages à la limite donnent naissance à des fonctions non-calculables, qui appartiennent de façon prouvable aux niveaux des hiérarchies arithmétiques et analytiques [Moore, 1996], [Mycka and Costa, 2004]. La complexité structurelle et calculatoire de classes de fonctions réelles récursives définies sans limites ou recherche de zéro, est étudiée dans la thèse [Campagnolo, 2001], et dans les articles [Campagnolo et al., 2000b], [Campagnolo et al., 2002], [Campagnolo, 2002] et [Campagnolo, 2004a], où des liens avec les fonctions récursives primitives sont établis. En particulier, si on se restreint aux intégrations linéaires (i.e. telles que g soit linéaire en h), l'algèbre de fonctions résultante donne une caractérisation des fonctions élémentaires de Kalmár [Campagnolo et al., 2002]. Graça et Costa [Graça and Costa, 2003] ont montré que les fonctions récursives primitives correspondent aux solutions des équations différentielles $y' = p(y, t)$, où p est un vecteur de polynômes, c'est-à-dire aux fonctions générées par un GPAC (voir aussi [Moore, 1996]).

La théorie de la récursivité sur les réels permet aussi de caractériser syntaxiquement les classes de fonctions calculables en analyse récursive, dès que des restrictions sur opérations de passages à la limite ou d'intégrations sont introduites [Bournez and Hainry, 2007]. De façon similaire, les fonctions élémentairement calculables sur les réels en analyse récursive correspondent précisément à la classe de fonctions réelles récursives définie à partir d'un certain ensemble de fonctions de base, et close par compositions, intégrations linéaires, et un schéma limite restreint [Bournez and Hainry, 2005]. Des résultats similaires ont été obtenus récemment, en utilisant une technique qui transfère des résultats de complexité classique sur les entiers directement en des résultats sur les fonctions sur les réels [Campagnolo and Ojakian, 2006].

Une discussion générale des motivations derrière la théorie de la $\mathbb{R}$-récursivité peut être trouvée dans [Mycka and Costa, 2005].

3.2.2 Modèles inspirés par les théories de systèmes à temps continu

Systèmes hybrides

Un nombre croissant de systèmes démontrent une certaine combinaison de comportements continus et discrets. L'investigation de ces systèmes a donné lieu à des nouveaux résultats en rapport avec la théorie des calculs à temps continu.

De nombreux modèles existent (voir par exemple la série de colloques *Hybrid Systems Computation and Controls* ou la tentative de comparaison des modèles de [Branicky, 1995a]), mais on peut dire que les systèmes hybrides sont essentiellement modélisés soit comme des équations différentielles avec des membres droits discontinus, ou par des équations différentielles avec des variables continues et discrètes, ou par des automates hybrides. Un automate hybride est un automate d'états finis étendu avec des variables. Son

évolution est donnée par des transitions discrètes gardées entre les états (places) de l'automate, qui peuvent remettre à zéro certaines variables, et par la dynamique d'évolution de chacune des variables dans les places. Les propriétés typiques des systèmes hybrides qui sont considérées dans la littérature sont les propriétés d'atteignabilité, de stabilité ou de contrôlabilité.

En relation avec l'approche par des équations différentielles, Branicky a prouvé dans [Branicky, 1995b] que tout modèle de système hybride qui peut implémenter une horloge et les équations différentielles génériques peut simuler les machines de Turing. Asarin, Maler et Pnueli ont montré dans [Asarin et al., 1995] que les équations différentielles constantes par morceaux peuvent simuler les machines de Turing dans $\mathbb{R}^3$, alors que le problème de l'atteignabilité pour ces systèmes est décidable en dimension $d \leq 2$ [Asarin et al., 1995]. Les équations différentielles constantes par morceaux, comme beaucoup de modèles de systèmes hybrides, font montre de ce que l'on appelle le phénomène de Zénon : un nombre non-fini de transitions discrètes peut avoir lieu en temps fini. Cela a été utilisé dans [Asarin and Maler, 1998] pour prouver que les ensembles arithmétiques peuvent êtres reconnus en temps fini par ces systèmes. Leur puissance de calculs exacte a été caractérisée en termes de leurs dimensions dans [Bournez, 1999a] et dans [Bournez, 1999b]. Les arguments de [Asarin et al., 1995] basés sur le théorème de Jordan pour prouver la décidabilité des équations différentielles constantes par morceaux planaires ont été généralisés pour prouver la décidabilité des équations polynomiales planaires [Ceraens and Viksna, 1996] et des systèmes d'inclusions différentielles planaires [Asarin et al., 2001].

Il y a une littérature importante à propos de l'approche de modélisation par automates hybrides visant à déterminer la frontière entre décidabilité et non-décidabilité pour les propriétés d'atteignabilité, en fonction des types de dynamiques, des gardes et des remises à zéros autorisées. Les propriétés d'atteignabilité sont connues pour êtres décidables pour les automates temporisés [Alur and Dill, 1990]. Cela a été généralisé par la suite aux automates multitaux [Alur et al., 1995], à des classes spécifiques d'automates temporisés à mise à jour [Bouyer et al., 2000a], [Bouyer et al., 2000b], ou encore aux automates rectangulaires initialisés dans [Henzinger et al., 1998], [Puri and Varaiya, 1994], à chaque fois en utilisant un argument basé sur une bissimulation finie, similaire dans l'esprit à celui de [Alur and Dill, 1990]. Il y a une multitude de résultats d'indécidabilité, la plupart se basant sur la simulation de machines à deux compteur de Minsky. Par exemple, le problème de l'atteignabilité est semi-décidable mais non-décidable pour les automates hybrides linéaires [Alur et al., 1995], [Nicollin et al., 1993]. Le même problème est connu pour être indécidable pour les automates rectangulaires avec au moins 5 horloges et une variable à deux taux [Henzinger et al., 1998], pour les automates temporisés avec deux horloges biaisées [Alur et al., 1995]. Pour des discussions, voir aussi [Asarin and Schneider, 2002]. Se référer à [Blondel and Tsitsiklis, 1999], [Collins and van Schuppen, 2004] ou à [Blondel and Tsitsiklis, 2000] pour d'autres propriétés que l'atteignabilité (par exemple la stabilité ou l'observabilité).

Théorie des automates

Il y a eu plusieurs tentatives d'étendre la théorie des automates classiques discrète au temps continu : cela est parfois appelé le programme général de Trakhtenbrot [Trakhtenbrot, 1995].

La première approche est reliée aux automates temporisés qui peuvent être vus comme des reconnaisseurs de langages [Alur and Dill, 1994]. De nombreux problèmes de décision spécifiques ont été considérés pour les automates temporisés : voir [Alur and Madhusudan, 2004], [Tripakis, 2003], [Finkel, 2005]. Les langages temporisés réguliers sont connus pour être clos par intersection, union, renommage mais non par complémentation. Le problème de l'appartenance, et le problème du vide sont décidables, alors que les problèmes de l'inclusion et de l'universalité sont indécidables. Leur clôture par mélange a été étudiée dans [Finkel, 2006]. Plusieurs variantes du théorème de Kleene ont été établies dans [Asarin et al., 1997], ou encore dans [Asarin, 1998], [Bouyer and Petit, 1999], [Asarin et al., 2002], [Bouyer and Petit, 2002], [Asarin and Dima, 2002]. Il y a eu des tentatives d'établir des lemmes de la pompe [Beauquier, 1998]. Un survol, avec discussions et problèmes ouverts reliés à cette approche peut être trouvé dans [Asarin, 2004].

Une théorie des automates indépendante et alternative a été développée dans [Rabinovich, 2003]. Il y est argumenté que les fonctions rétrospectives à mémoire finie, qui agissent sur des signaux, correspondent aux fonctions réalisables par des systèmes à états finis. Des propriétés de clôture, ou des théorèmes de

représentations sont étudiés par Rabinovich dans cet article.

Finalement, une autre approche indépendante est considérée dans l'article [Ruohonen, 2004], où une hiérarchie dans l'esprit de celle de Chomsky est établie pour des familles d'ensembles de fonctions continues par morceaux. Des équations différentielles, associées à des structures de mémoire spécifiques, sont utilisées pour reconnaître des ensembles de fonctions. Ruohonen montre que la hiérarchie résultante n'est pas triviale, et établi des propriétés de clôture et d'inclusions entre classes.

Autres modèles de calculs

En addition aux deux grandes approches précédentes, il y a un certain nombre d'autres modèles de calculs qui ont mené à des développements de la théorie des systèmes à temps continus.

La question de savoir si la théorie de la relativité générale d'Einstein admet des équations d'espace-temps qui permet à un observateur de voir une éternité en un temps fini a été répondue par l'affirmative dans [Hogarth, 1992]. La question de savoir si cela pourrait être utilisé théoriquement pour résoudre des taches hypercalculatoires a été explorée dans [Hogarth, 2001], [Hogarth, 1996], [Hogarth, 1994]. Les propriétés physiques impliquées dans de tels calculs sont discutées dans les articles [Earman and Norton, 1993] [Etesi and Németi, 2002].

Certains modèles inspirés par des machines ne sont pas clairement discrets ou analogiques. Par exemple, la puissance des mécanismes planaires constitués de barres rigides contraintes à évoluer dans un plan et jointes à leurs extrémités par des rivets a attiré beaucoup d'attention en Angleterre et en France dans la fin du 19ième siècle, et dans les années 1940 en Russie. Un théorème attribué[3] à Kempke [Kempe, 1876] dit qu'ils sont capables de calculer toutes les fonctions algébriques : voir par exemple [Artobolevskii, 1964] ou [Svoboda, 1948].

3.3 Équations différentielles et propriétés

La plupart des modèles à temps continu décrits plus haut ont une dynamique continue décrite par des équations différentielles. Dans le GPAC de Shannon et dans les réseaux de Hopfield, l'entrée d'un calcul correspond à la condition initiale, alors que la sortie est respectivement, l'évolution temporelle ou l'état d'équilibre atteint du système. D'autres modèles sont des reconnaisseurs de langages. L'entrée est à nouveau une condition initiale, ou un contrôle initial, et la sortie est déterminée par une région acceptante dans l'espace des états du système.

Tous ces systèmes tombent par conséquent dans le cadre des systèmes dynamiques à temps continu. Plus généralement, tout modèle de calcul peut être vu comme un système dynamique.

Cette section vise à rappeler quelques résultats fondamentaux à propos des systèmes dynamiques et des équations différentielles, et nous discuterons comment les différents modèles peuvent êtres comparés dans ce cadre général.

3.3.1 Systèmes dynamiques et équations différentielles

Considérons que nous travaillons dans $\mathbb{R}^n$. Soit $f : E \to \mathbb{R}^n$, où $E \subset \mathbb{R}^n$ est ouvert. Une équation différentielle (ordinaire) est donnée par $y' = f(y)$ et une solution est une fonction différentiable $y : I \subset \mathbb{R} \to E$ qui satisfait l'équation.

Pour tout $x \in E$, le théorème fondamental d'existence et d'unicité (voir par exemple [Hirsch et al., 2003]) pour les équations différentielles affirme que si f est Lipschitz sur E, i.e. s'il existe K tel que $||f(y_1) - f(y_2)|| < k||y_1 - y_2||$ pour tout $y_1, y_2 \in E$, alors la solution du problème de Cauchy

$$y' = f(y), \qquad y(t_0) = x \tag{3.1}$$

[3]Le théorème est très souvent attribué à Kempke [Artobolevskii, 1964], [Svoboda, 1948], même si apparemment il n'a jamais prouvé cela exactement.

existe et est unique sur un certain intervalle d'existence maximal $I \subset \mathbb{R}$. Dans la terminologie des systèmes dynamiques, $y(t)$ est nommée une trajectoire, $\mathbb{R}^n$ l'espace des phases, et la fonction $\phi(t, x)$, qui donne la position $y(t)$ de la solution au temps t avec la condition initiale x, est appelée le flot. Nous appelons orbite le graphe de y dans $\mathbb{R}^n$.

En particulier, si f est continûment différentiable sur E, alors les hypothèses du théorème d'existence unicité sont satisfaites [Hirsch et al., 2003]. La plupart de la théorie mathématique a été développée dans ce cas, mais elle peut s'étendre à conditions plus faibles. En particulier, si f est supposée seulement continue, alors l'unicité est perdue, mais l'existence est garantie : voir par exemple [Coddington and Levinson, 1972]. Si f est autorisée à être discontinue, alors la notion de solution nécessite d'être raffinée. L'approche la plus célèbre est celle de Fillipov [Filippov, 1988]. Certains modèles de systèmes hybrides utilisent des notions ad hoc (et distinctes) de solutions. Par exemple, une solution d'une équation différentielle constante par morceaux dans [Asarin et al., 1995] est une fonction continue dont la dérivée à droite satisfait l'équation.

De façon très générale, un système dynamique peut se définir comme l'action d'un sous-groupe $\mathcal{T}$ sur un espace X, i.e. par une fonction (un flot) $\phi : \mathcal{T} \times X \to X$ qui satisfait les deux équations suivantes

$$\phi(0, x) = x \tag{3.2}$$

$$\phi(t, \phi(s, x)) = \phi(t + s, x). \tag{3.3}$$

Il est bien connu que les sous-groupes de $\mathcal{T}$ de $\mathbb{R}$ sont soit denses dans $\mathbb{R}$ ou isomorphes aux entiers. Dans le premier cas, le temps est dit continu, dans le second discret.

Puisque les flots obtenus par des problèmes de Cauchy de la forme (3.1) satisfont ces équations, ils correspondent à des systèmes dynamiques à espace et temps continus particuliers. Pas tous les systèmes dynamiques à espace et temps continus peuvent être mis sous la forme d'une équation différentielle, mais les problèmes de Cauchy de la forme (3.1) sont suffisamment généraux pour couvrir une très large classe de tels systèmes. En particulier, si ϕ est continûment différentiable, alors $y' = f(y)$, avec $f(y) = \frac{d}{dt}\phi(t, y)\big|_{t=0}$, décrit la dynamique du système.

Pour les systèmes à temps discret, nous pouvons supposer sans perte de généralités que $\mathcal{T}$ correspond aux entiers. L'analogue du problème de Cauchy (3.1) pour les systèmes à temps discret est une équation de récurrence du type

$$y_{t+1} = f(y_t), \qquad y_0 = x. \tag{3.4}$$

Un système dynamique dont l'espace est discret et qui évolue avec un temps discret est qualifié de digital, sinon d'analogique. Une classification de plusieurs modèles de calculs en fonction de la nature de leur espace et de leur temps peut être trouvée dans la figure 3.3.1.

Etats / Temps	Discrets	Continus
Discret	Machines de Turing [Turing, 1936] Lambda calcul [Church, 1936] Fonctions récursives [Kleene, 1936] Systèmes de Post [Post, 1946] Automates cellulaires Automates à piles Automates d'états finis ⋮	Réseaux de neurones [Hopfield, 1984] à temps discret Réseaux de neurones de [Siegelmann and Sontag, 1994] Systèmes PCD de [Asarin et al., 1995] Machines de Blum Shub Smale [Blum et al., 1989] Machines optiques de [Woods and Naughton, 2005] Machines à signaux de [Durand-Lose, 2005] Reconnaisseurs dynamiques de [Moore, 1998a] ⋮
Continu	Modèle BDE [Dee and Ghil, 1984]	[Shannon, 1941] GPAC Réseaux de neurones de [Hopfield, 1984] à temps continu Systèmes hybrides de [Branicky, 1995b] Systèmes PCD de [Asarin et al., 1995] Automates temporisés [Alur and Dill, 1990] Fonctions $\mathbb{R}$-récursives [Moore, 1996] ⋮

FIG. 3.3 – Une classification de certains modèles de calculs, en fonction de leur espace et de leur temps.

3.3.2 Systèmes dissipatifs et non-dissipatifs

Un point x^* de l'espace des états est appelé un point d'équilibre si $f(x^*) = 0$. Si le système est en x^* il le restera. Il est dit stable si, pour tout voisinage U de x^*, il y a un voisinage W de x^* dans U tel que toute solution partant d'un point x de W est définie et est dans U pour tout temps $t > 0$. Le point est asymptotiquement stable si, en addition des propriétés ci-dessus, on a $\lim y(t) = x^*$ [Hirsch et al., 2003].

Des conditions locales sur la différentielle $Df(x^*)$ de f en x^* sont bien connues. Si en un point d'équilibre x^* toutes les valeurs propres de $Df(x^*)$ possèdent des parties réelles strictement négatives, alors x^* est asymptotiquement stable, et en outre les solutions dans le voisinage approchent x^* exponentiellement. Dans ce cas, x^* est nommé un point source. En outre, il est connu que dans un point stable d'équilibre x^*, aucune valeur propre de $Df(x^*)$ ne peut avoir une partie réelle strictement positive [Hirsch et al., 2003].

En pratique, le théorème de stabilité de Lyapunov s'applique de façon plus large (même si x^* n'est pas un point source). Il affirme que s'il existe une fonction continue V définie sur un voisinage de x^*, différentiable (sauf peut être en x^*) avec $V(x^*) = 0$, $V(x) > 0$ pour $x \neq x^*$, et $dV(x)/dt \leq 0$ pour $x \neq x^*$ alors x^* est stable. Si en outre on a aussi $dV(x)/dt < 0$ pour $x \neq x^*$, alors x^* est asymptotiquement stable : voir [Hirsch et al., 2003].

Si la fonction V satisfait ces conditions en tout point, alors le système est globalement asymptotiquement stable : quel que soit le point initial x, les trajectoires convergeront ultimement vers les minimaux locaux de V. Dans ce contexte, la fonction de Lyapunov V peut s'interpréter comme une énergie, et ses minimaux correspondent à des attracteurs du système dynamique. Ils constituent des sous-ensembles bornés de l'espace des phases vers lesquels toute région de conditions initiales de volume non-vide converge lorsque le temps croit.

Un système dynamique est dit dissipatif si le volume d'un ensemble dans l'espace des phases décroît sous l'action du flot. Les systèmes dissipatifs sont caractérisés par des attracteurs. Par opposition, un système dynamique est dit conservatif si ce volume est conservé. Par exemple, tous les systèmes hamiltoniens sont conservatifs en raison du théorème de Liouville [Arnold, 1989]. Les systèmes dynamiques conservatifs ne peuvent êtres globalement asymptotiquement stables [Arnold, 1989].

3.3.3 Calculabilité des solutions des équations différentielles

Nous revoyons maintenant certains résultats sur la calculabilité au sens de l'analyse récursive (voir par exemple [Weihrauch, 2000]) des solutions d'une équation différentielle.

En général, étant donnée une fonction calculable f, on peut chercher à comprendre si la solution du problème de Cauchy (3.1) est aussi calculable dans le sens de l'analyse récursive. Si on impose que la solution du problème de Cauchy est unique, alors cette solution est calculable. Formellement, si f est calculable sur $[0,1] \times [-1,1]$ et $y' = f(t,y)$, $y(0) = 0$ possède une solution unique sur $[0,b]$, $0 < b \leq 1$, alors la solution y est calculable sur $[0,b]$ [Pour-El and Richards, 1979].

Cela est vrai pour les problèmes de Cauchy n-dimensionnels, si l'unicité des solutions est toujours parmi les hypothèses [Ruohonen, 1996].

Cependant, la calculabilité des solutions est perdue dès que l'unicité des solutions est relâchée, même en dimension 1. En effet, [Pour-El and Richards, 1979] montrent qu'il existe une fonction calculable f (même calculable en temps polynomial) $f : [0,1] \times [-1,1] \to \mathbb{R}$, telle que l'équation $y' = f(t,y)$, avec $y(0) = 0$, possède des solutions non-uniques, mais aucune des solutions n'est calculable sur tout intervalle de temps fermé.

Des phénomènes similaires ont lieu pour d'autres équations naturelles : l'équation d'onde en dimension 3 (qui est une équation différentielle partielle), avec des données initiales calculables, peut avoir une solution unique qui est calculable nulle part[4] [Pour-El and Richards, 1981], [Pour-El and Zhong, 1997]. Observons que, même si f est supposée calculable et analytique, et la solution est unique, il peut se produire que l'intervalle maximal (α, β) d'existence de la solution soit non-calculable [Graça et al., 2006]. La même question est ouverte pour f polynomiale. Ces auteurs montrent, cependant que si f et f' sont calculables, alors la solution du problème de Cauchy $y' = f(y,t)$, $y(0) = x$, pour x calculable, est calculable sur son intervalle maximal d'existence. Se référer à [Pour-El and Richards, 1989], [Ko, 1983] pour plus de résultats d'indécidabilité, et aussi à [Ko, 1983], [Ko, 1991] pour des considérations de complexité, et pas seulement de calculabilité.

3.3.4 Indécidabilité statique

Comme cela est observé dans [Asarin, 1995] ou dans [Ruohonen, 1997b], il est facile, mais souvent pas très informatif, d'obtenir des résultats d'indécidabilité avec des systèmes dynamiques à temps continu.

À titre d'illustration, rappelons l'exemple suivant tiré de [Ruohonen, 1997b], qui discute le problème de la détection d'évènements (étant donnée une équation différentielle $y' = f(t,y)$, avec une valeur initiale $y(0)$, décider si une certaine condition $g_j(t, y(t), y'(t)) = 0, j = 1, \cdots, k$ se produit à un temps t dans un intervalle donné I), fixons n'importe quelle machine de Turing universelle $\mathcal{M}$. La suite $f_0, f_1, \cdots$ de rationnels définie par

$$f_n = \begin{cases} 2^{-m} & \text{si} \quad \mathcal{M} \text{ s'arrête en } m \text{ étapes sur l'entrée } n \\ 0 & \text{si} \quad \mathcal{M} \text{ ne s'arrête pas sur l'entrée } n \end{cases}$$

n'est pas une suite calculable de rationnels, mais est une suite calculable de réels, selon la nomenclature de [Pour-El and Richards, 1989]. Maintenant, la détection de l'évènement $y(t) = 0$ pour l'équation différentielle $y' = 0$, étant donnés n, et la valeur initiale $y(0) = f_n$, est indécidable parce que $f_n = 0$ est indécidable. Beaucoup des résultats d'indécidabilité fournis par l'analyse récursive sont d'une certaine façon dans cet esprit [Asarin, 1995].

3.3.5 Indécidabilité dynamique

Pour pouvoir discuter de façon plus profonde la calculabilité par des équations différentielles, nous nous focalisons maintenant sur des équations différentielles qui encodent les transitions d'une machine de Turing plutôt que le résultat de tout un calcul directement[5]. Typiquement, on part d'une fonction calculable (simple) injective qui code chaque configuration d'une machine de Turing M par un point de $\mathbb{R}^n$. Soit x le codage de la configuration initiale de M. On recherche alors une fonction $f : E \subset \mathbb{R}^{n+1} \to \mathbb{R}^n$ telle que la solution de

[4] Cependant, dans tous ces cas, les problèmes étudiés sont mal posés : soit la solution est non unique, ou elle est instable. L'ajout de conditions naturelles visant à régulariser et éviter que le problème soit mal posé mène à la calculabilité [Weihrauch and Zhong, 2002].

[5] Cela est nommé indécidabilité dynamique par [Ruohonen, 1993].

$y'(t) = f(y,t)$, avec $y(0) = x$, au temps $T \in N$, donne le codage de la configuration de M après T étapes. Nous verrons, dans le reste de cette section, que f peut être restreinte à être de basse dimension, a être lisse ou analytique, ou a être définie sur un domaine compact.

Plutôt que de formuler la propriété plus haut comme une simulation d'une machine de Turing, on peut le formuler en termes de résultat d'atteignabilité. Étant donné un problème de Cauchy donné par f et x, et une région $A \subset \mathbb{R}^n$, on s'intéresse à décider s'il existe un $t \geq 0$ tel que $y(t) \in A$, i.e., si le flot partant de x atteint A. Il est clair que si f simule une machine de Turing dans le sens précédent, alors le problème de l'atteignabilité pour cette classe de systèmes est indécidable (considérer A comme le codage des configurations d'arrêt de M). Par conséquent, l'atteignabilité est une autre façon d'étudier la calculabilité des équations différentielles, et un résultat négatif est souvent la conséquence d'un résultat de simulation de machines de Turing. De façon similaire, l'indécidabilité de la détection d'évènements est une conséquence de résultats de simulation de machines de Turing.

La calculabilité des ensembles invariants ou atteignables a été étudiée par [Collins, 2005] pour les systèmes dynamiques généraux à temps continu, et dans [Collins and Lygeros, 2005] pour les systèmes hybrides.

En général, interpréter les machines de Turing comme des systèmes dynamiques procure une signification physique à la machine de Turing qui n'est pas donnée par la vision classique de von Neumann [Campagnolo, 2001]. Cela montre aussi que nombre de caractéristiques qualitatives des systèmes dynamiques (analogiques ou non-analogiques), par exemple les questions relatives aux bassins d'attractions, aux comportements chaotiques ou même périodiques, ne sont pas calculables [Moore, 1990]. Réciproquement, cela amène dans le cadre des machines de Turing et de la calculabilité en général de nombreuses questions traditionnellement associées aux systèmes dynamiques. Cela inclut par exemple les relations entre l'universalité et le chaos [Asarin, 1995],des conditions nécessaires pour l'universalité [Delvenne et al., 2004], la compréhension de la frontière du chaos [Legenstein and Maass, 2005], la calculabilité de l'entropie [Koiran, 2001], et les relations avec la propriété de shadowing [Hoyrup, 2006]. On observera que les machines de Turing sont présentées comme des systèmes dynamiques déjà dans [Minsky, 1967].

3.3.6 Plongement de machines de Turing en temps continu

Le plongement des machines de Turing dans les systèmes dynamiques à temps continu est souvent réalisé en deux étapes. Les machines de Turing sont tout d'abord plongées dans des systèmes dynamiques à espace continu et à temps discret, et les systèmes obtenus sont ensuite plongés dans des systèmes à temps et espace continus.

La première étape peut être réalisée en basses dimensions avec des dynamiques simples : les articles [Moore, 1990], [Ruohonen, 1993], [Branicky, 1995b], et [Ruohonen, 1997b] considèrent des systèmes dynamiques généraux, l'article [Koiran et al., 1994] considère des fonctions affines définies par morceaux, l'article [Siegelmann and Sontag, 1995] des réseaux de neurones sigmoïdaux linéaires seuillés, [Koiran and Moore, 1999] des fonctions analytiques en forme close undimensionnelles, qui peuvent être robustes [Graça et al., 2005], et l'article [Kurganskyy and Potapov, 2005] des fonctions unidimensionnelles restreintes définies par morceaux.

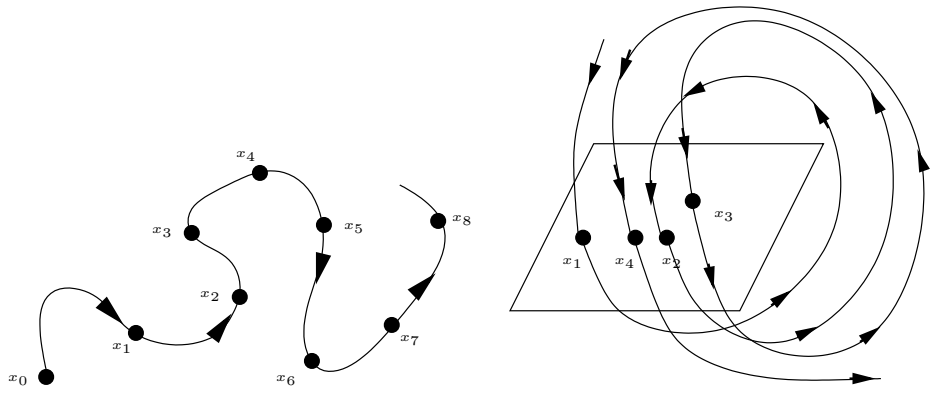


FIG. 3.4 – Discrétisation par stroboscope (à gauche) et par une section de Poincaré (à droite) d'un système dynamique à temps continu.

Pour la deuxième étape, l'astuce usuelle est de construire un système dynamique à temps et espace continu dont une discrétisation correspond au système dynamique à espace continu qui réalise le plongement.

Il existe plusieurs façons classiques de discrétiser un système à temps et espace continu : voir la figure 3.3.6. Une façon est d'utiliser un stroboscope virtuel : le flot $x_t = \phi(t, x)$, lorsque t est restreint aux entiers, définit les trajectoires d'un système dynamique à temps discret. Une autre possibilité est via l'utilisation d'une section de Poincaré : la suite x_t des intersections des trajectoires avec (par exemple) une hypersurface peut fournir le flot d'un système dynamique à temps discret : voir [Hirsch et al., 2003].

L'opération inverse, appelée suspension, est généralement réalisée en étendant et en rendant lisses les équations, et requiert généralement le recours à des équations en dimension supérieure.

Cela explique pourquoi les machines de Turing sont simulées par des systèmes dynamiques à temps continu lisses de dimension 3 dans [Moore, 1990], [Moore, 1991], [Branicky, 1995b] ou par des équations différentielles constantes par morceaux de dimension 3 dans [Asarin et al., 1995], alors que l'on sait les simuler par des fonctions affines par morceaux de dimension seulement 2 en temps discret [Koiran et al., 1994]. Il est connu que les équations différentielles constantes par morceaux de dimension 2 ne peuvent pas[6] simuler les machines de Turing arbitraires [Asarin et al., 1995], alors que la question de savoir si les fonctions affines par morceaux de dimension 1 peuvent simuler les machines de Turing arbitraires est ouverte.

D'autres simulations de machines de Turing par des dynamiques à temps continu incluent la simulation robuste avec des équations différentielles polynomiales de [Graça et al., 2005] et [Graça et al., 2006]. Ce résultat constitue une version améliorée de la simulation des machines de Turing par des fonctions réelles récursives de [Campagnolo et al., 2000b], où il est montré que des classes de fonctions lisses mais non-analytiques sont closes par itérations. Observons que, bien que la solution d'une équation différentielle polynomiale soit calculable sur son domaine maximal d'existence (voir la section 3.3.3), les résultats de simulation montrent que le problème de l'atteignabilité est indécidable pour les équations différentielles polynomiales.

En addition des machines de Turing, d'autres modèles discrets peuvent être simulés par des équations différentielles. La simulation de machines à 2 compteurs peut se réaliser en dimension 2 et même en dimension 1 au prix d'équations différentielles discontinues [Ruohonen, 1997b]. La simulation d'automates cellulaires peut être réalisée par des équations différentielles partielles à l'aide de fonctions C^∞ [Omohundro, 1984].

3.3.7 Une discussion

Une technique clé pour plonger l'évolution temporelle d'une machine de Turing dans un flot est d'utiliser des "horloges continues" comme dans [7] [Branicky, 1995b].

[6] Voir aussi les généralisations de ce résultat dans [Ceraens and Viksna, 1996] et [Asarin et al., 2001].

[7] Branicky attribue l'idée d'un calcul en deux phases à [Brockett, 1989] et [Brockett, 1991]. Une astuce similaire est présente dans [Ruohonen, 1993]. Nous ne suivons pas en fait l'article original [Branicky, 1995b] mais sa présentation dans

L'idée est de partir d'une fonction $f : \mathbb{R} \to \mathbb{R}$, qui préserve les entiers, et de construire l'équation différentielle sur $\mathbb{R}^3$

$$\begin{aligned} y_1' &= c(f(r(y_2)) - y_1)^3\theta(\sin(2\pi y_3)) \\ y_2' &= c(r(y_1) - y_2)^3\theta(-\sin(2\pi y_3)) \\ y_3' &= 1. \end{aligned}$$

Ici $r(x)$ est l'analogue d'une fonction entière : $r(x) = n$ pour $x \in [n - 1/4, n + 1/4]$ pour un entier n, et $\theta(x)$ vaut 0 pour $x \leq 0$, $\exp(-1/x)$ pour $x > 0$, et c est une constante bien choisie.

La variable y_3 sert uniquement à fournir la variable temps t. Supposons $y_1(0) = y_2(0) = x$. Pour $t \in [0, 1/2]$, $y_2' = 0$, et donc y_2 est gardé fixé à x. Maintenant, si $f(x) = x$, il est clair que y_1 sera gardé fixé à x. Si $f(x) \neq x$, alors $y_1(t)$ approchera $f(x)$ sur cet intervalle de temps, et en se référant aux calculs dans [Campagnolo, 2001], si c est choisi suffisamment grand, on peut être sûr que $|y_1(1/2) - f(x)| \leq 1/4$. Par conséquent, nous aurons $r(y_1(1/2)) = f(x)$. Maintenant, pour $t \in [1/2, 1]$, les rôles sont inversés $y_1' = 0$, et donc y_1 est gardé fixé à la valeur de $f(x)$. La variable y_2 approche $f(x)$, et nous aurons $r(y_2(1)) = f(x)$. L'équation a un comportement similaire pour tous les intervalles qui suivent de la forme $[n, n + 1/2]$, $[n + 1/2, n + 1]$, et donc à tout temps entier t, $f^{[t]}(x) = r(y_1(t))$.

En d'autres termes, la construction plus haut transforme une fonction sur $\mathbb{R}$ en une équation différentielle (en dimension supérieure) qui simule ses itérations. Cependant, pour faire cela, nous avons eu besoin de $\theta(\sin(2\pi y_3))$ qui sert d'horloge. À la base, le système dynamique à temps continu obtenu a une "senteur hybride". Même si le flot est lisse (i.e. est C^∞) par rapport au temps, l'orbite n'admet pas de tangente en tout point puisque y_1 et y_2 sont alternativement constants.

On peut chercher à dépasser cette restriction en se limitant aux simulations de machines de Turing par des flots et des fonctions analytiques. Bien qu'il soit connu que les fonctions analytiques sur des domaines non bornés peuvent simuler la fonction de transition de n'importe quelle machine de Turing [Koiran and Moore, 1999], ce n'est que très récemment qu'il a été montré que les machines de Turing peuvent êtres simulées par des flots analytiques (et même polynomiaux) sur des domaines non bornés [Graça et al., 2005]. Il serait désirable d'étendre ce type de résultats aux domaines compacts. Cependant, il est conjecturé dans [Moore, 1998b] que cela n'est pas possible, i.e. qu'aucune fonction analytique sur un domaine fini dimensionnel compact peut simuler une machine de Turing arbitraire via un codage des entrées et des sorties raisonnable.

3.3.8 Contractions du temps et de l'espace

Les machines de Turing sont simulées par les équations différentielles en temps réel : par exemple, dans les constructions évoquées plus haut, l'état $y(T)$ au temps $T \in N$ encode la configuration de la machine de Turing après T étapes. Cependant, puisque les systèmes à temps continu peuvent exhiber des phénomènes de contractions d'espace et de temps, des machines de Turing accélérantes[8] [Davies, 2001], [Copeland, 1998], et même des machines de Turing avec oracles peuvent êtres simulées en des temps arbitrairement courts.

En effet, en suivant la présentation de [Ruohonen, 1993], dénotons un système à temps continu par un triplet (F, n, A), où F défini l'équation différentielle $y' = F(y)$ sur $\mathbb{R}^n$, avec l'ensemble acceptant A : une entrée x est acceptée si et seulement si la trajectoire partant avec la condition initiale x atteint A.

Une telle machine peut être accélérée : la substitution $t = e^u - 1$ par exemple change la machine $\mathcal{M} = (F, n, A)$ en $((G, 1), n + 1, A \times \mathbb{R})$ où

$$\frac{dg}{du} = G(g(u), u) = F(g(u))e^u \text{ and } g(u) = y(e^u - 1),$$

ce qui donne une accélération du temps exponentielle. Observons que les dérivées de la solution par rapport à la nouvelle variable temporelle u sont exponentiellement plus grandes.

[Campagnolo, 2001].

[8] Des possibilités similaires de simulation des machines de Turing accélérantes en mécanique quantique sont discutées dans [Calude and Pavlov, 2002].

La substitution $t = \tan(\pi u/2)$ mène à une accélération temporelle infinie, i.e. compresse tout calcul, même infini, en un calcul sur l'intervalle de temps fini $0 \leq u < 1$. Maintenant, les dérivées vont vers l'infini lors du calcul.

Pour obtenir une contraction de l'espace, remplaçons l'état $y(t)$ de la machine $\mathcal{M} = (F, n, A)$ par $r(t) = y(t)e^{-t}$. Cela donne une machine dont l'espace est exponentiellement réduit $((H, 1), m + 1, H_1)$ où

$$\frac{dr}{dt} = H(r(t), t) = F(r(t)e^t)e^{-t} - r(t)$$

et

$$H_1 = \{(e^{-t}q, t) | q \in A \text{ and } t \geq 0\}.$$

Bien entendu, cette transformation réduit exponentiellement la distance entre les trajectoires, ce qui impose une précision supplémentaire pour les distinguer.

Des résultats de dureté pour les niveaux des hiérarchies arithmétiques et analytiques pour plusieurs problèmes de décisions pour les systèmes à temps continu sont déduits de constructions similaires dans [Ruohonen, 1993], [Ruohonen, 1994], [Moore, 1996], ou dans [Asarin and Maler, 1998]. Des résultats de complétude, tout comme une caractérisation exacte du pouvoir de reconnaissance des équations différentielles constantes par morceaux en fonction de leur dimension sont obtenus dans [Bournez, 1999a], [Bournez, 1999b]. Observons que de tels phénomènes sont des instances du phénomène dit de Zénon dans la littérature des systèmes hybrides [Alur and Dill, 1990] [Asarin and Maler, 1998].

Il peut être remarqué que les constructions précédentes fournissent des résultats d'indécidabilité seulement pour des fonctions sur des domaines infinis ou semi-ouverts, puisque les entiers positifs ou nuls, qui correspondent aux temps entiers des machines de Turing sont envoyés sur les intervalles de la forme $[0, 1)$. Une construction par l'analyse a été développée dans [Ruohonen, 1997a] pour montrer qu'une compression temporelle est possible sur un domaine compact de la forme $[0, 1]$, avec une fonction G continue, et bornée sur $[0, 1]$ mais non-différentiable. Il en suit que le problème de la détection d'évènements est indécidable même pour les fonctions continues sur un domaine compact [Ruohonen, 1997a].

L'indécidabilité est écartée, cependant, si la fonction G est suffisamment lisse sur $[0, 1]$ (disons $\mathcal{C}^1$), si à la fois G et les conditions initiales sont calculables, et si une condition d'acception suffisamment robuste est considérée. En effet, des problèmes comme la détection d'évènements deviennent décidables, puisque le système peut être simulé effectivement [Ruohonen, 1997a].

Plutôt que de plonger des machines de Turing dans des systèmes dynamiques continus, il est naturel de se demander s'il y aurait une meilleure notion de calcul et de complexité pour les systèmes dynamiques provenant de modèles de notre monde physique. Nous nous penchons sur cette question dans la section qui suit.

3.4 Vers une théorie de la complexité

Nous discutons maintenant un certain nombre de vues sur la complexité des systèmes dynamiques continus. Nous considérons des systèmes généraux et la question de la difficulté de simuler des systèmes à temps continu par un modèle digital. Nous nous focalisons alors sur les systèmes dissipatifs, où les trajectoires convergent vers des attracteurs. En particulier, nous discutons l'idée que le temps de calcul devrait être le temps naturel de l'équation différentielle. Finalement, nous présentons des résultats de complexité pour des systèmes à temps continu plus généraux qui correspondent à des classes de fonctions réelles récursives.

3.4.1 Systèmes généraux

Dans [Vergis et al., 1986], les auteurs se demandent si les machines analogiques peuvent être plus efficaces que les machines digitales. Vergis et al. postulent la validité de la thèse de Church forte pour les modèles analogiques, qui instanciée sur ces modèles, clame que le temps requis par un ordinateur digital pour simuler un machine analogique est borné par une fonction polynomiale des ressources utilisées par la machine analogique. Vergis et al. prétendent que la thèse de Church forte est prouvable pour les systèmes dynamiques à temps continu définis par une équation différentielle $y' = f(y)$ Lipchitzienne.

Les ressources utilisées par un ordinateur analogique incluent l'intervalle temporel d'opération, disons $[0, T]$, la taille du système, qui peut être mesurée par $\max_{t\in[0,T]} ||y(t)||$, tout comme une borne sur les dérivées de y. Par exemple, le poids, le temps d'opération, et les forces appliquées sont toutes des ressources utilisées par une particule décrite en mécanique Newtonienne [Vergis et al., 1986].

L'affirmation plus haute nécessite de préciser ce que l'on entend par "simulation". Dans [Vergis et al., 1986], il est considéré que le problème de Cauchy $y' = f(y)$, $y(0) = x$ est simulé si, étant donné une précision ε, on peut calculer une approximation de $y(T)$ avec une erreur au plus ε. En utilisant la méthode d'Euler, et en supposant que les erreurs d'arrondis sont majorées par σ, la borne totale sur l'erreur est donnée par

$$||y(T) - y_N^*|| \leq \frac{h}{\lambda}\left[\frac{R}{2} + \frac{\sigma}{h^2}\right](e^T\lambda - 1), \tag{3.5}$$

où y_N^* est l'approximation après N étapes, h est le pas utilisé, λ la constante de Lipchitz pour f sur $[0, T]$, et $R = \max\{||y''(t)||, t \in [0, T]\}$. De cette borne (3.5), Vergis *et al.* concluent que le nombre N d'étapes nécessaires pour simuler le système par la méthode d'Euler est polynomial en R et $\frac{1}{\varepsilon}$. Ils utilisent ce fait pour affirmer que la thèse de Church forte est valide pour les équations différentielles Lipchitziennes.

Cependant, N est exponentiel en T, et T correspond au temps d'opération de la machine analogique modélisée. Cela rend les arguments de [Vergis et al., 1986] pas totalement convaincants, comme cela est souligné par exemple explicitement dans [Orponen, 1997].

Plus récemment, Smith a discuté dans [Smith, 2006] si des phénomènes d'hypercalculs sont possibles avec le problème des n corps en mécanique. En particulier, il a montré que cette dépendance exponentielle en T peut être éliminée. Comme le fait observé Smith, toutes les méthodes classiques numériques d'ordre fixe pour résoudre les équations différentielles souffrent du même problème de dépendance exponentielle en T. Cependant, en considérant une combinaison de méthodes de Runge-Kutta avec des ordres qui varient linéairement en T, il est effectivement possible de construire une méthode qui requiert un nombre d'étapes N polynomial en T, dès que la valeur absolue de chaque composante de f, y et la valeur absolue de chacune des dérivées partielles de f par rapport à chacun de ses arguments, ayant un degré total de différentiation k, est en $(kT)^{\mathcal{O}(k)}$ [Smith, 2006]. Les implications pour la thèse de Church forte sont discutées dans [Smith, 2006] et dans [Bournez, 2006] (voir aussi l'annexe A).

La même question peut être étudiée dans le cadre de l'analyse récursive. Lorsque $f : [0, 1]\times[-1, 1] \to \mathbb{R}$ est calculable en temps polynomial, et satisfait une forme faible de la condition de Lipchitz, l'(unique) solution y sur $[0, 1]$ du problème de Cauchy $y' = f(t, y)$, $y(0) = 0$ est toujours calculable en espace polynomial [Ko, 1983]. Cependant, résoudre une équation différentielle en temps polynomial avec cette condition faible de Lipchitz est essentiellement aussi difficile que de résoudre un problème PSPACE-complet, puisqu'il existe une fonction f calculable en temps polynomial, qui satisfait cette condition, pour laquelle la solution y n'est pas calculable en temps polynomial sauf si $P = PSPACE$ [Ko, 1983], [Ko, 1991].

Les résultats de Ko ne sont pas directement comparables aux bornes polynomiales de Smith. En analyse récursive, l'entrée est le nombre de bits de précision. Si la borne sur l'erreur d'approximation de $y(t)$ est mesurée en bits, i.e. si $\varepsilon = 2^{-d}$, alors le nombre d'étapes N requis dans [Smith, 2006] est exponentiel en d.

Si f est analytique, alors la solution de $y' = f(y)$ est aussi analytique. Dans ce cas, les méthodes par pas peuvent êtres évitées. C'est l'approche suivie dans [Müller and Moiske, 1993], où il est prouvé que si f est analytique et calculable en temps polynomial alors la solution est calculable en temps polynomial.

En résumé, bien que la thèse de Church forte soit vérifiée pour les équations différentielles analytiques dans ce sens, elle n'a pas été complètement établie pour les systèmes généraux d'équations différentielles Lipchitziennes. La possibilité de calculs hyper-polynomiaux par des équations différentielles ne peut pas être écartée, au moins dans le principe à ce jour. Pour des discussions informelles sur la thèse de Church en relation avec les systèmes analogiques, voir aussi [Aaronson, 2005] et [MacLennan, 2001].

Plusieurs auteurs ont montré que certains problèmes de décision ou d'optimisation (par exemple la connectivité de graphes, la programmation linéaire) peuvent être résolus par des systèmes dynamiques spécifiques. Des exemples et des références peuvent êtres trouvés dans [Vergis et al., 1986], [Brockett, 1991], [Faybusovich, 1991], [Helmke and Moore, 1994], et [Ben-Hur et al., 2002].

3.4.2 Systèmes dissipatifs

Nous nous focalisons maintenant sur les systèmes dissipatifs, en présentant deux approches. La première concerne essentiellement les réseaux de neurones formels, comme les réseaux de Hopfield. Elle se base sur les notions de la complexité de circuits, et mène à des bornes inférieures sur la complexité de tels réseaux. La seconde concerne les phénomènes de convergence vers les attracteurs, et considère les fonctions d'énergie comme des façons de mesurer la complexité des processus à temps continu.

Lorsque l'on considère des systèmes dissipatifs, comme les réseaux de neurones de Hopfield, l'approche suivante pour définir une théorie de la complexité devient naturelle. On considère des familles $(C_n)_{n\in N}$ de systèmes à temps continu, chaque C_n pour une certaine longueur d'entrée $n \geq 0$. Étant donnée une entrée (digitale) $w \in \{0,1\}^*$, on considère l'évolution du système C_n sur (un certain codage de) l'entrée w, où n est la longueur de w. Elle va finir ultimement par converger vers un état stable, qui sera considéré comme le résultat du calcul.

Cette notion de calcul, inspirée par les circuits, est la plus courante dans la littérature qui concerne la théorie de la complexité des réseaux de neurones : voir par exemple le survol [Síma and Orponen, 2003b]. Dans ce cadre, les réseaux symétriques à temps continu de Hopfield avec une fonction d'activation linéaire seuillée peuvent simuler les réseaux de neurones récurrents binaires à temps discret arbitraires, avec un surcoût au plus linéaire [Orponen and Sima, 2000], [Síma and Orponen, 2003a].

On peut penser que cela va contre l'intuition, puisque de tels réseaux symétriques, qui sont contraints par une fonction d'énergie de Lyapunov, peuvent seulement exhiber des phénomènes de convergence, et donc ne peuvent même pas réaliser un simple bit alternant. Cependant, la convergence des systèmes à temps continu peut être exponentiellement longue en la taille du système [Sima and Orponen, 2003], et donc la simulation est réalisée en utilisant un sous-réseau qui fournit 2^n pulsations d'horloge avant de converger.

Les langages reconnus par des familles de réseaux de Hopfield à temps discret ont été prouvés dans [Orponen, 1996] comme correspondant à la classe de complexité (non uniforme) $PSPACE/poly$ pour des poids d'interconnexions arbitraires, et $P/poly$ pour des poids de tailles polynomialement bornées. Par conséquent, les familles de réseaux de Hopfield symétriques à temps continu ont au moins cette puissance. Ces bornes inférieures peuvent ne pas être optimales, puisque des bornes supérieures pour les dynamiques à temps continu ne sont pas connues [Síma and Orponen, 2003a], [Síma and Orponen, 2003b].

Intéressons nous maintenant aux systèmes dissipatifs avec une fonction de Lyapunov E.

Gori et Meer [Gori and Meer, 2002] considèrent un modèle de calcul qui a la capacité de trouver les minimisateurs (i.e. les points de minimum local ou global) de E. Pour éviter que la complexité d'un problème soit cachée dans la description de E, cette fonction doit être facile à calculer. Dans ce cadre, un problème Π est considéré comme facile s'il existe une fonction E unimodale (i. e. tous les minimisateurs de E sont des minimisateurs globaux) telle que la solution de Π puisse être obtenue facilement à partir du minimum global de E.

Plus précisément, Gori et Meer font l'investigation dans [Gori and Meer, 2002] d'un modèle où un problème Π sur les réels est considéré comme résolu s'il existe une famille $(E_n)_n : \mathbb{R}^n \times \mathbb{R}^{q(n)} \to \mathbb{R}$ de fonctions d'énergie, donnée par une famille uniforme de programmes en droite ligne (straight line programe) (q est un polynôme fixe), et une autre famille $(N_n)_n$ de programmes en droite ligne, telles que, pour toute entrée d, une solution $\Pi(d)$ du problème peut être calculée en utilisant $N_{q(n)}(w^*)$, à partir d'un minimisateur global w^* de $w \to E_n(d,w)$.

Ils introduisent deux classes U et NU, en analogie avec P et NP en complexité classique. U correspond au cas ci-dessus où, pour tout d, $w \to E_n(d,w)$ est unimodal, par opposition à NU, le cas général.

Des notions de réductions sont introduites, et il est montré qu'un problème d'optimisation naturel (trouver le minimum d'une fonction objective linéaire sur un ensemble défini par des contraintes polynomiales multivariées quadratiques) est NU-difficile. Il est montré qu'il existe des problèmes (artificiels) NU complets. Ces idées sont généralisées pour obtenir une hiérarchie polynomiale, avec des problèmes complets [Gori and Meer, 2002].

Le cadre proposé est en fait relativement abstrait, en évitant plusieurs problèmes reliés avec ce que certains pourraient attendre d'une vraie théorie de la complexité pour les calculs en temps continu. Néanmoins, il offre le grand avantage de ne pas se baser sur aucune mesure de complexité spécifique à propos des trajectoires.

Voir la discussion très intéressante dans [Gori and Meer, 2002].

Cependant, il semble nécessaire de comprendre la complexité d'approcher les minima des fonctions d'énergie, qui correspondent aux équilibres des systèmes dynamiques. Des premières étapes vers ces directions ont été proposées dans l'article [Ben-Hur et al., 2002], où l'on se restreint aux systèmes dissipatifs avec convergences exponentielles. Rappelons que si x^* est une source, alors la vitesse de convergence vers x^* est du type

$$|x(t) - x^*| \equiv e^{-\lambda t}$$

où $-\lambda$ est la plus grande partie réelle d'une valeur propre de $Df(x^*)$. Cela signifie que $\tau = 1/\lambda$ est un temps caractéristique naturel de l'attracteur : tout les $\tau \log 2$ unités de temps, un nouveau bit de l'attracteur est calculé.

Pour les systèmes considérés dans [Ben-Hur et al., 2002], chaque source possède une région d'attraction, dans laquelle les trajectoires deviennent prisonnières. On définit le temps de calcul comme $t_c = max(t_c(\epsilon), t_c(U))$, où $t_c(\epsilon)$ est le temps requis pour atteindre un ϵ voisinage d'un attracteur, et $t_C(U)$ le temps requis pour atteindre sa région d'attraction. Alors $T = \frac{t_c}{\tau}$ est une mesure de complexité sans dimension, invariante par toute contraction temporelle linéaire.

Deux algorithmes à temps continu, MAX pour calculer le maximum de n nombres, et $FLOW$ pour résoudre le problème du flot maximal ont été étudiés dans ce cadre dans [Ben-Hur et al., 2002]. MAX a été montré comme appartenant à la classe de complexité $CLOG$ (temps logarithmique continu) et $FLOW$ à CP (temps polynomial continu). Les auteurs conjecturent que CP correspond aux temps polynomial classique [Ben-Hur et al., 2002]. Les deux problèmes MAX et $FLOW$ sont des instances spécifiques d'un flot proposé dans [Faybusovich, L., 1991] pour résoudre les problèmes de programmation linéaire, étudié plus profondément dans [Ben-Hur et al., 2003] et [Ben-Hur et al., 2004a]. Des variations sur les définitions des classes de complexité, tout comme des moyens d'introduire des classes de complexité non-déterministes en relations avec les attracteurs chaotiques ont aussi été discutés dans l'article [Siegelmann and Fishman, 1998].

3.4.3 Complexité et fonctions réelles récursives

De nombreuses classes de complexité en espace et en temps ont des caractérisations récursives [Clote, 1998]. Cela suggère un lien naturel entre ces classes et les classes de fonctions réelles récursives. Par exemple, une certaine classe $\mathcal{L}$ de fonctions réelles récursives, close par composition et par intégration linéaire, donne une caractérisation des fonctions élémentaires de Kalmár sur les entiers [Campagnolo et al., 2002]. En termes de systèmes dynamiques, $\mathcal{L}$ correspond à des cascades de systèmes linéaires de profondeur finie, chaque niveau dépendant linéairement de ses propres variables et des sorties des niveaux précédents. Il est assez surprenant que de tels systèmes, par rapport aux systèmes non-linéaires généraux, ait tant de puissance de calculs. Des bornes inférieures et supérieures, allant de l'espace linéaire à l'espace élémentaire ont été décrites dans [Campagnolo, 2004a] pour les fonctions réelles récursives, en considérant des formes restreintes d'opérateurs de composition et d'intégration.

La question $P = NP$ en complexité classique a mené à des investigations en utilisant les fonctions réelles récursives. Une approche pour cela a été proposée par Costa et Mycka. Leur programme, explicitement présenté dans [Mycka and Costa, 2005], examine les questions de la complexité classique grâce aux outils de l'analyse. En particulier, ils proposent deux classes de fonctions réelles récursives telles que leur séparation implique $P \neq NP$. Voir [Costa and Mycka, 2006a], [Costa and Mycka, 2006c], [Costa and Mycka, 2006b] et [Mycka and Costa, 2006c] pour des séries de résultats dans cet esprit.

3.5 Robustesse aux bruits et aux imprécisions

Jusqu'à ce point, nous avons considéré des calculs en temps continu dans des espaces idéalisés sans aucun bruit. La plupart des résultats que nous avons discutés ont été établis en ignorant l'impact du bruit ou des imprécisions dans les systèmes à temps continu. Cela constitue une faiblesse critiquée de façon récurrente dans le domaine, déjà observée par Orponen dans son survol [Orponen, 1997]. Observons que considérer des calculs en précision non-bornée est fortement analogue à considérer des machines de Turing à rubans

infinis : la construction de machines à rubans infinis n'est pas possible, comme la construction de machines à précision infinie.

Bien qu'ils n'y aient pas eu de réelles percées vis-à-vis de ces questions en ce qui concerne spécifiquement les systèmes à temps continu, des développements intéressants concernant les effets du bruit et de l'imprécision ont été réalisés à propos des systèmes analogiques à temps discret. Aussi, dans cette section, nous élargissons notre propos pour discuter aussi de résultats relatifs à des systèmes à temps discret. Nous pensons que ces résultats peuvent être généralisés, ou tout du moins donnent des pistes pour être généralisés, aux systèmes à temps continu, bien que cela reste souvent à faire.

Nous nous focalisons dans un premier temps sur les systèmes à espace borné, à propos desquels une conjecture de tout le monde (folklore conjecture) clame que la robustesse implique la décidabilité. Nous présentons des résultats qui appuient cette conjecture, et des résultats qui semblent aller dans le sens opposé. À la fin de cette section, nous discutons les systèmes à temps continu avec un espace non-borné.

Les techniques communes pour simuler une machine de Turing par un système dynamique sur un espace borné nécessitent le codage des configurations de la machine de Turing en des nombres réels. Ces simulations sont détruites si les nombres réels, ou les fonctions impliquées ne sont pas représentées avec une précision infinie. Cela a mené à conjecture de tout le monde, populaire en particulier dans la communauté de la vérification, que l'indécidabilité n'a pas lieu pour les systèmes "réalistes", "non-précis", "bruités", "flous", "robustes". Voir par exemple [Fränzle, 1999], ou [Foy, 2004] pour différentes discussions dans le sens de cette conjecture, et [Asarin, 2006] pour des discussions des arguments formels ou informels qui mènent ou ont mené à cette conjecture.

Il n'y a pas de consensus à propos de ce qu'est un modèle de bruit réaliste. Une discussion de ce sujet nécessite de faire recours à la question de quels sont les bons modèles de notre monde physique. En l'absence d'un modèle de bruit universellement accepté, on peut cependant considérer différents modèles pour le bruit, l'imprécision, ou des conditions de régularité, et étudier les propriétés des systèmes résultants.

En particulier, il y a eu plusieurs tentatives de prouver que les systèmes analogiques bruités sont au mieux équivalents aux automates finis. Brockett montre dans [Brockett, 1989] que les systèmes dynamiques à temps continu peuvent simuler les automates finis arbitraires. En utilisant des arguments topologiques basés sur des relations d'équivalence par homotopies et les transformations de Deck associées, il montre réciproquement dans [Brockett, 1994] qu'à certains systèmes à temps continu dissipatifs on peut associer un automate.

Dans [Maass and Orponen, 1998], Maass et Orponen prouvent que la présence d'un bruit probabiliste réduit la puissance d'une large classe de modèles analogiques à temps discret sur un domaine compact aux langages réguliers. Cela étend un résultat dans [Casey, 1996], [Casey, 1998] pour le cas particulier des réseaux de neurones récurrents.

L'idée de Maass et d'Orponen est de remplacer une dynamique en temps discret parfaite de type $x_{i+1} = f(x_i, a_i)$, où a_i est le symbole d'entrée au temps i, sur un domaine compact, par une dynamique probabiliste

$$Probability(x_{i+1} \in B) = \int_{q \in B} z(f(x_i, a_i), q) d\mu,$$

où B est n'importe quel Borélien. Ici z est un noyau de densité qui reflète un bruit arbitraire, qui est supposé équicontinu par morceaux : pour tout ϵ, il existe δ tel que pour tout r, p, q, $\|p - q\| \leq \delta$ implique $|z(r, p) - z(r, q)| \leq \epsilon$.

Dénotons par $\pi_x(q)$ la distribution des états après que la chaîne x ait été lue, partant d'un état initial q. Une condition d'acception robuste est considérée : un langage L est reconnu, s'il existe $\rho > 0$ tel que $x \in L$ si et seulement si $\int_F \pi_{xu}(q) d\mu \geq 1/2 + \rho$ pour un certain $u \in \{U\}^*$, et $x \notin L$ si et seulement si $\int_F \pi_{xu}(q) d\mu \leq 1/2 - \rho$ pour tout $u \in \{U\}^*$, où U est un symbole de blanc, et F l'ensemble des états acceptants.

L'observation principale est alors que l'espace des fonctions $\pi_x(.)$ peut être partitionné en un nombre fini de classes C telles que deux fonctions $\pi_x(.)$ et $\pi_y(.)$ dans la même classe satisfont $\int_r |\pi_x(r) - \pi_y(r)| d\mu \leq \rho$, de telle sorte que deux mots x, y correspondant à la même classe satisfont $xw \in L$ si et seulement si $yw \in L$ pour tous mots w.

En fait, pour n'importe quel bruit usuel et commun, comme un bruit gaussien, qui n'est pas nul sur une partie suffisamment large de l'espace des états, de tels systèmes ne sont même pas capables de reconnaître

tous les langages réguliers [Maass and Sontag, 1999]. Ils reconnaissent précisément les langages définis de [Rabin, 1963], comme cela est démontré dans [Maass and Sontag, 1999], [Ben-Hur et al., 2004b]. Si le support du bruit est borné, alors tous les langages réguliers peuvent être reconnus [Maass and Orponen, 1998].

De façon alternative à l'approche de bruit probabiliste de Maass et Orponen, le bruit peut se modéliser par l'introduction d'un bruit non-déterministe.

À un système dynamique discret parfait (déterministe) S défini par $x_{i+1} = f(x_i)$, associons le système S_ϵ non-déterministe ϵ-perturbé dont les trajectoires sont les suites $(x_n)_n$ avec $\|x_{i+1} - f(x_i)\| \leq \epsilon$. Écrivons $Reach[S](x, y)$ (respectivement $Reach_n[S](x, y)$) s'il existe une trajectoire de S allant de x vers y (resp. en $i \leq n$ étapes).

La vérification algorithmique des propriétés de sûreté est reliée très étroitement au problème du calcul des états atteignables. Étant donné S, avec un sous-ensemble d'états initiaux S_0, écrivons $Reach[S]$ pour l'ensemble des y tels que $Reach[S](x, y)$ pour un certain $x \in S_0$. Étant donné une propriété sur les états p (c'est-à-dire une propriété p qui est soit vraie soit fausse en chaque point s de l'espace), soit $[[\neg p]]$ l'ensemble des états s en lesquels p est fausse. Alors S est sûr (c'est-à-dire p est un invariant) si et seulement si $Reach[S] \cap [[\neg p]] = \emptyset$ (voir par exemple [Alur et al., 1995] et [Nicollin et al., 1993]).

Si la classe de systèmes considérés est telle que la relation $Reach_n[S](x, y)$ est récursive[9], alors $Reach[S]$ est récursivement énumérable, puisque $Reach[S] = \bigcup_n Reach_n[S]$. Plusieurs articles ont visé à prouver que $Reach[S]$ est en fait récursif pour les classes de systèmes dynamiques robustes, pour différentes notions de robustesse. Nous revoyons maintenant ces articles.

Fränzle observe dans [Fränzle, 1999] que le calcul de $Reach[S_\epsilon]$ par la formule $Reach[S_\epsilon] = \bigcup_n Reach_n[S_\epsilon]$ termine toujours, sauf si éventuellement $Reach[S_\epsilon]$ n'a pas un diamètre fortement fini : il existe un nombre infini de points dans $Reach[S_\epsilon]$ à distance mutuelle au moins ϵ. Cela est exclu par exemple sur un domaine borné. Il en suit que si nous appelons robuste un système qui est soit non sûr, ou tel qu'une ϵ-perturbation est sûre pour un certain ϵ, alors la sûreté est décidable pour les systèmes robustes sur un domaine compact [Fränzle, 1999].

Considérons, comme dans [Puri, 1998], la relation définie par $Reach_\omega[S] = \bigcap_{\epsilon>0} Reach[S_\epsilon]$, qui correspond aux états qui restent atteignables lorsque le bruit converge vers 0. Asarin et Bouajjani montrent que pour une large classe de systèmes dynamiques à temps discret et à temps continu (les machines de Turing, les fonctions affines par morceaux, les équations différentielles constantes par morceaux), $Reach_\omega[S]$ est co-récursivement énumérable. En outre, toute relation co-récursivement énumérable est de la forme $Reach_\omega[S]$ pour un certain système S de chacune de ces classes [Asarin and Bouajjani, 2001]. Il suit de la première observation que si nous appelons robuste un système tel que $Reach[S] = Reach_\omega[S]$, alors calculer $Reach[S]$ est décidable pour les systèmes robustes [Asarin and Bouajjani, 2001].

Asarin et Collins considèrent dans [Asarin and Collins, 2005] un modèle de machine de Turing exposée à un faible bruit stochastique, dont la puissance de calcul est caractérisée comme Π_2^0. Il est intéressant de comparer ce résultat avec les résultats précédents ou un bruit faible non-déterministe mène seulement à Π_1^0, c'est-à-dire aux ensembles co-récursivement énumérables.

Nous nous tournons maintenant vers des résultats qui vont dans le sens contraire de la conjecture que robustesse implique décidabilité.

Un premier exemple est que la sûreté des systèmes reste indécidable si la relation de transition des systèmes considérés est ouverte, comme cela est prouvé dans [Henzinger and Raskin, 1999] [Asarin, 2006]. Cependant, la question pour un bruit uniforme non-déterministe borné inférieurement est une question ouverte [Asarin, 2006].

Le bruit peut aussi être introduit en perturbant les trajectoires. Gupta Henzinger et Jagadeesan proposent dans [Gupta et al., 1997] de considérer une métrique sur les trajectoires des automates temporisés, et de supposer que si un système accepte une trajectoire, il doit aussi accepter les trajectoires du voisinage. Ils montrent que cette notion de robustesse n'est pas suffisante pour éviter l'indécidabilité de la complémentation des automates temporisés. Henzinger et Raskin prouvent dans [Henzinger and Raskin, 1999] que les principaux résultats d'indécidabilité pour la vérification des systèmes hybrides restent indécidables pour les systèmes robustes dans ce sens.

[9] Récursive en x, y et en n.

Finalement, nous revoyons un résultat récent pour les systèmes dynamiques à temps continu avec un espace d'états non-borné. Graça, Campagnolo et Buescu ont prouvé récemment que les équations différentielles polynomiales peuvent simuler de façon robuste les machines de Turing en temps réel. Plus précisément, considérons que $\theta : \mathbb{N}^3 \to \mathbb{N}^3$ soit la fonction de transition d'une machine de Turing M dont les configurations sont encodées dans $\mathbb{N}^3$. Alors, il existe un $\epsilon > 0$, un polynôme p, et une condition initiale $y(0)$ tels que la solution de $y' = p(t, y)$ encode l'état de M après t étapes avec une erreur d'au plus ϵ. En outre, cela reste vrai sur le voisinage de tout entier t, même si p et si la condition initiale $y(0)$ sont perturbés. Bien entendu, ce type de simulation requiert un espace d'états non-borné.

3.6 Conclusion

Par ce survol du domaine de la théorie des calculs pour les systèmes à temps continu, nous voyons qu'elle fournit des éclairages pour des domaines divers comme la vérification, la théorie du contrôle, la conception de circuits intégrés, les machines analogiques, la théorie de la récursion, la théorie des équations différentielles, et la complexité.

Nous avons tenté de fournir une présentation synthétique et systématique de tous les modèles principaux et des résultats sur les systèmes à temps continu. Dans la dernière décade, de nombreux nouveaux résultats ont été établis, ce qui prouve que c'est un domaine de recherche actif. Nous avons présenté des développements récents de la théorie des calculs à temps continu vis-à-vis de la calculabilité, la complexité, la robustesse au bruit et aux imprécisions, et nous avons identifié plusieurs problèmes ouverts. Pour conclure, nous voudrions discuter de plusieurs directions pour des recherches futures.

Calculabilité.

Il n'est pas clair si un concept unificateur comme la thèse de Church-Turing existe pour les systèmes à temps continu. Bien qu'il ait été montré que certains modèles à temps continu possèdent des capacités hypercalculatoires, tous ces résultats se basent sur l'utilisation d'une certaine quantité infinie de ressources, comme le temps, l'espace, la précision ou l'énergie. En général, il est souvent conjecturé que les modèles à temps continu "raisonnables" ne peuvent pas calculer plus que les machines de Turing. Cela mène à la question de savoir si les calculs à temps continu par des systèmes physiquement réalistes peuvent êtres aussi puissants que les modèles digitaux. Nous avons vu que si l'on se restreint aux systèmes à temps continu qui évoluent sur un espace d'états borné, et qui sont sujets à du bruit, alors ils deviennent comparables aux automates finis. Puisque les systèmes à temps continu robustes et analytiques peuvent simuler les machines de Turing avec un espace non-borné, nous pensons que les calculs digitaux et analogiques sont également puissants du point de vue de la calculabilité. En outre, comme nous l'avons vu, plusieurs résultats récents établissent l'équivalence entre les fonctions calculables par des équations différentielles polynomiales, les fonctions calculables par GPAC, et les fonctions calculables dans le sens de l'analyse récursive. Ce type de résultats renforce l'idée qu'il pourrait y avoir un cadre unifié pour les calculs à temps continu, similaire à celui qui existe en théorie de la calculabilité classique.

Nous pensons qu'un paradigme général de calcul à temps continu idéal ne devrait faire appel qu'à des fonctions analytiques, puisque ces dernières sont souvent considérées comme les plus acceptables d'un point de vue physique. Les systèmes dynamiques continus sont des modèles naturels pour représenter de nombreux phénomènes à temps continu. Les systèmes continus classiques, comme l'équation de van der Pol, les systèmes de Lotka Volterra, les équations de Lorenz, ou tous les exemples des chapitres 1 et 2 sont décrits en utilisant des équations différentielles analytiques, et même à membre droit polynomial. Ces arguments reliés à la physique, combinés avec les propriétés de calcul des systèmes d'équations différentielles polynomiales nous mènent à suggérer que le modèles à temps continu des équations différentielles définies par des problèmes de Cauchy polynomiaux est un candidat possible pour un paradigme général des calculs à temps continu. Cette idée mérite plus d'attention.

Complexité

Nous avons vu plusieurs pistes pour construire une théorie de la complexité pour les systèmes à temps continu. À vrai dire, à ce jour, le travail est largement en cours, et il n'y a pas d'accord général entre les auteurs sur les définitions de base comme le temps de calculs, ou la taille des entrées. Les résultats établis dans la section 3.4 sont soit dérivés de concepts intrinsèques aux systèmes à temps continu considérés, ou reliés à la théorie classique de la complexité. Puisque l'analyse récursive est un cadre bien établi et bien compris pour l'étude des problèmes de complexité pour les systèmes continus, nous pensons que mieux comprendre les relations qui existent entre les différentes approches et l'analyse récursive du point de vue de la complexité est de première importance. Il y a de nombreux problèmes qui restent ouverts à propos de bornes supérieures sur la puissance des systèmes à temps continu. Par exemple, des bornes supérieures ne sont pas connues pour les réseaux de neurones de Hopfield, ou pour les systèmes d'équations différentielles Lipchitziennes, ce qui compromet la validité de la thèse de Church forte. Nous avons vu que cette thèse peut peut-être être prouvée pour les équations différentielles analytiques. Cela mène à se demander si une théorie des calculs à temps continu basée sur les équations différentielles définies par des problèmes de Cauchy polynomiaux pourrait s'étendre en une théorie de la complexité.

L'analyse récursive permet aussi d'étudier la complexité des fonctions réelles. Un des domaines de recherche les plus intriguant consiste à comprendre les liens qui existent entre les fonctions réelles récursives et la complexité classique, pour établir par exemple des traductions des problèmes ouverts de la complexité dans le cadre de l'analyse.

Robustesse

Nous avons vu que très peu de recherche a été fait à ce jour en ce qui concerne les effets de bruits ou d'imprécision sur les calculs à temps continu. À ce jour, l'essentiel des résultats concerne les systèmes à temps discret, et leur transfert au cas du temps continu mérite de l'attention. On peut aussi se demander comment la puissance des systèmes analogiques croit avec leur précision. La question a été formulée et formalisée pour les systèmes à temps discret, en particulier pour les reconnaisseurs dynamiques dans [Moore, 1998b], mais la plupart de la recherche dans cette direction reste à faire. De nombreuses questions ouvertes surgissent si l'on demande si les résultats d'indécidabilité pour les systèmes à temps continu restent vrais pour les systèmes robustes. Cela est de première importance dans le domaine de la vérification par exemple, puisque cette question est reliée de façon très proche à la question de la preuve de la terminaison des procédures automatiques de vérification. Une meilleure compréhension des hypothèses avec lesquelles du bruit mène à de la décidabilité ou de l'indécidabilité est nécessaire. Par exemple, un bruit non-déterministe pour les systèmes ouverts n'écarte pas l'indécidabilité, mais la question est ouverte pour un bruit uniforme non-déterministe borné inférieurement [Asarin, 2006].

Nouveaux modèles

D'une façon plus spéculative, comme les systèmes à temps continu arrivent naturellement lorsque l'on cherche à modéliser les grandes populations (voir le chapitre 2), une question fascinante est de comprendre si la théorie des calculs en temps continu peut réellement aider à comprendre les modèles massivement parallèles comme l'Internet. Nous pensons que la théorie des calculs pour les systèmes à temps continu peut être au coeur de la prochaine génération de la complexité et de la calculabilité.

Chapitre 4

Sur les liens entre quelques modèles

Dans ce chapitre, nous présentons quelques-un de nos résultats de comparaisons entre différents modèles de calculs à temps continu. Nous commençons par nous intéresser à la puissance du General Purpose Analog Computer de Shannon [Shannon, 1941], et aux problèmes de Cauchy polynomiaux[1]. Par la suite, nous nous intéressons à des sous-classes de fonctions $\mathbb{R}$-récursives. Les fonctions $\mathbb{R}$-récursives ont été introduites par [Moore, 1998a]. Nous les relions aussi aux fonctions calculables en analyse récursive.

Les résultats de ce chapitre ont tous été obtenus en collaborations. Les résultats sur le GPAC sont le fruit d'une collaboration avec Manuel Campagnolo, Daniel Graça et Emmanuel Hainry. Les résultats sur les fonctions $\mathbb{R}$-récursives relèvent de la thèse d'Emmanuel Hainry, que nous avons eu le plaisir d'encadrer.

4.1 Une thèse de Church-Turing pour les modèles à temps continu ?

Selon la thèse de Church-Turing[2] tous les modèles *raisonnables* et suffisamment puissants de calculs digitaux sont équivalents au modèle de la machine de Turing.

Aucun résultat similaire n'est connu lorsqu'on considère les calculs analogiques à temps continu[3]. Alors que de nombreux modèles ont été étudiés, comme le modèle de Blum Shub et Smale [Blum et al., 1989], les fonctions $\mathbb{R}$-récursives de Moore [Moore, 1998a], les réseaux de neurones [Siegelmann, 1999], ou l'analyse récursive [Pour-El and Richards, 1989], [Ko, 1991], [Weihrauch, 2000], aucun d'entre eux ne peut se prétendre "universel".

Cela est dû en partie au fait que peu de relations entre les modèles sont connues. En outre, certains des résultats affirment que les modèles ne sont pas équivalents, rendant l'espoir d'une thèse de Church-Turing pour les modèles analogiques inaccessible. Par exemple, le modèle de Blum Shub et Smale autorise les fonctions discontinues, alors que seules les fonctions continues sont calculables en analyse récursive [Weihrauch, 2000].

Cependant, nous allons montrer que cela n'est peut-être pas si utopique.

4.2 GPAC, problèmes de Cauchy polynomiaux et analyse récursive : trois paradigmes équivalents

Nous avons prouvé l'équivalence de deux modèles de calculs qui étaient jusque-là considérés comme non équivalents : l'analyse récursive et le *General Purpose Analog Computer* (GPAC) de Claude Shannon.

[1] Observons que, de façon informelle, nous montrons quelque part que les machines universelles du début du XXième siècle valent bien celles du début du XXIième : GPAC, et machines de Turing, même combat. 1 partout

[2] Voir aussi l'annexe A et ses discussions sur les variantes de cette thèse.

[3] Une thèse de Church-Turing a été proposée par Siegelmann et Sontag pour les modèles à espace continu et temps discret [Siegelmann and Sontag, 1994], [Siegelmann, 1995].

4.2.1 Point de vue historique

Le GPAC a été introduit en 1941 par Shannon [Shannon, 1941] comme un modèle mathématique d'un dispositif analogique : l'Analyseur Différentiel. Le tout premier analyseur différentiel a été construit au MIT en 1931 sous la direction de Vannevar Bush [Bush, 1931]. Les analyseurs différentiels ont été utilisés des années 1930 aux débuts des années 1960 pour résoudre des problèmes numériques, comme par exemple des problèmes balistiques. Les toutes premières versions étaient mécaniques. Plus tard, les analyseurs différentiels sont devenus électroniques. Nous renvoyons au chapitre 3 pour plus de détails.

Un GPAC peut être vu comme un circuit constitué d'éléments de base interconnectés. Ces éléments réalisent chacun un des types d'opérations de la figure 4.1, où les entrées sont des fonctions d'une variable indépendante, appelée le *temps* (dans un analyseur différentiel électronique, les entrées correspondent à des tensions). Ces éléments *boîtes noires* réalisent des additions, des multiplications, génèrent des constantes, ou intègrent.

Alors que beaucoup des fonctions sont connues pour être générables par un GPAC, une exception notable est la fonction

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt.$$

Ce fait est déjà observé par Shannon dans son article original [Shannon, 1941].

Si nous avons à l'esprit que cette fonction est connue pour être calculable dans le cadre de l'analyse récursive [Pour-El and Richards, 1989], ce résultat a souvent été interprété comme la preuve que le GPAC est un modèle trop faible, et en tout cas plus faible que l'analyse récursive.

Cependant, cela est plus dû à la notion de GPAC-calculabilité, plutôt qu'au modèle. En effet, on considère en général qu'un GPAC doit calculer une fonction en *temps réel*, une notion très restrictive de calcul. Si nous changeons la notion de calcul en une notion de *calcul convergent*, comme celle utilisée en analyse récursive, il a été montré récemment par Daniel Graça que la fonction Γ devient calculable [Graça, 2004].

Dans la suite de cette section, nous allons renforcer fortement ce résultat en montrant qu'en fait *toute fonction calculable* sur un domaine compact peut être calculée par un GPAC dans ce sens[4]. Réciproquement, nous montrons qu'avec des hypothèses raisonnables, la réciproque est aussi vraie.

En d'autres termes, nous prouvons que la puissance de calcul des GPAC coïncide avec celle de l'analyse récursive.

Observons que la notion de "calcul convergent" correspond à une classe particulière de fonctions $\mathbb{R}$-récursives [Moore, 1998a], [Mycka and Costa, 2004], [Bournez and Hainry, 2007].

Observons en outre qu'il a été montré dans [Graça et al., 2005] que les machines de Turing peuvent être simulées par des GPACs. Nous montrons quelque part que cela peut être étendu aux machines de Turing de type 2, ou avec oracles, utilisées en analyse récursive : voir [Weihrauch, 2000].

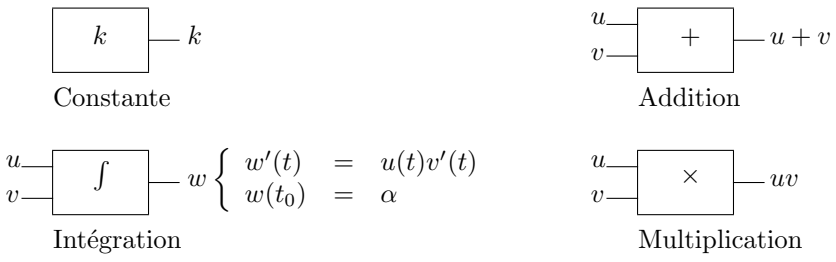


FIG. 4.1 – Les éléments de base d'un GPAC.

[4]Sauf explicitement précisé, dans ce chapitre, "fonction calculable" signifie "fonction calculable dans le sens de l'analyse récursive."

4.2.2 Le GPAC

Le GPAC a été initialement introduit par Shannon dans [Shannon, 1941], et le modèle a été raffiné dans les articles [Pour-El, 1974], [Lipshitz and Rubel, 1987], [Graça and Costa, 2003], et [Graça, 2004].

Le modèle consiste simplement en les familles de circuits que l'on peut construire avec les unités de base présentées dans la figure 4.1. Les connexions avec des rétroactions (feedback) sont autorisées, si ce n'est que certaines restrictions existent, puisque sinon cela peut mener soit à des systèmes sans évolution, ou dont la dynamique n'est pas définie (par exemple si l'on branche la sortie d'un additionneur sur son entrée), soit avec des comportements indésirables (comme la possibilité de plusieurs sorties pour une entrée fixée). Nous renvoyons pour les détails à [Graça and Costa, 2003].

Shannon, dans son article original, mentionne déjà que le GPAC génère tous les polynômes, la fonction exponentielle, les fonctions trigonométriques, et leurs inverses. Plus généralement, Shannon affirme que les fonctions générées par un GPAC sont différentiellement algébriques, c'est-à-dire satisfont les conditions de la définition suivante.

Définition 2 *La fonction unaire y est différentiellement algébrique (d.a.) sur l'intervalle I s'il existe un polynôme non nul p à coefficients réels tel que*

$$p\left(t, y, y', ..., y^{(n)}\right) = 0, \qquad \text{on } I. \tag{4.1}$$

Comme corollaire, puisque l'on sait que la fonction

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$$

n'est pas différentiellement algébrique [Hölder, 1887], nous obtenons le résultat de non-calculabilité[5] suivant, présent dans [Shannon, 1941].

Proposition 1 *La fonction Γ ne peut pas être générée par un GPAC.*

Cependant, la preuve de Shannon, qui relie les fonctions générées par un GPAC avec les fonctions différentiellement algébriques, est incomplète (cela a été montré et partiellement corrigé dans [Pour-El, 1974], [Lipshitz and Rubel, 1987]). En fait, comme montré dans [Graça and Costa, 2003], le modèle initial du GPAC de Shannon n'est pas bien (complètement) défini.

4.2.3 Problèmes de Cauchy polynomiaux

Pour la classe mieux définie de GPAC dans [Graça and Costa, 2003], la propriété suivante est vraie :

Proposition 2 *Une fonction scalaire $f : \mathbb{R} \to \mathbb{R}$ est générée par un GPAC si et seulement si elle est une projection d'une solution d'un problème de Cauchy polynomial, c'est-à-dire d'un système d'équations différentielles*

$$y' = p(y, t), \tag{4.2}$$

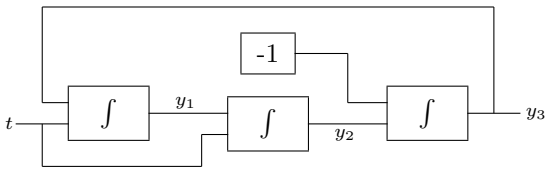
où p est un vecteur de polynômes. Une fonction $f : \mathbb{R} \to \mathbb{R}^k$ est générée par un GPAC si seulement si toutes ses projections le sont.

A partir de maintenant, nous considérerons principalement qu'un GPAC correspond à une équation différentielle du type de (4.2), c'est-à-dire un problème de Cauchy polynomial selon la terminologie du chapitre 1. Pour un exemple d'utilisation de la proposition précédente, voir la figure 4.2.

Les fonctions générables par GPAC dans ce sens sont évidemment différentiellement algébriques.

Si on se rappelle que les solutions d'une équation différentielle ordinaire analytique sont toujours analytiques (voir par exemple [Arnold, 1992]), une autre conséquence est le résultat suivant :

[5] Il est très instructif de comparer le raisonnement utilisé, par rapport aux arguments usuels pour prouver la non calculabilité d'un problème ou d'une fonction pour les machines de Turing.



$$\begin{cases} y_1' = y_3 & \& \quad y_1(0) = 1 \\ y_2' = y_1 & \& \quad y_2(0) = 0 \\ y_3' = -y_1 & \& \quad y_3(0) = 0 \end{cases}$$

FIG. 4.2 – Génération de cos et sin avec un GPAC : version circuit sur la gauche et version équation différentielle sur la droite. On a $y_1 = \cos$, $y_2 = \sin$, $y_3 = -\sin$.

Corollaire 1 *Si f est une fonction générée par un GPAC, alors elle est analytique.*

Comme nous l'avons vu dans la proposition 1, la fonction Γ n'est pas générable par un GPAC.

Cependant, il a été récemment prouvé par Daniel Graça qu'elle peut être calculée par un GPAC si nous modifions la notion de GPAC calculabilité en une notion de "calcul convergent" [Graça, 2004].

Dans les définitions originales de Shannon de GPAC, rien n'est supposé sur les constantes et sur les conditions initiales de l'équation différentielle (4.2). En particulier, il peut y avoir des réels non-calculables. Ce type de GPAC mène de façon triviale à des calculs non simulables par machines de Turing. Pour éviter cela, nous restreignons le modèle de [Graça, 2004] comme suit.

Définition 3 *Une fonction $f : [a, b] \to \mathbb{R}$ est GPAC-calculable*[6] *et si et seulement s'il existe un polynôme calculable $p : \mathbb{R}^{n+1} \to \mathbb{R}^n$ et $n-1$ valeurs calculables $\alpha_1, ..., \alpha_{n-1}$, et des indices $i, j \in \{1, ..., n\}$, tels que, pour tout $x \in [a, b]$,*

- *si $(y_1, ..., y_n)$ est la solution*[7] *du problème de Cauchy $y' = p(y, t)$ avec $y(0) = (\alpha_1, ..., \alpha_{n-1}, x)$ au temps $t_0 = 0$,*
- *alors $\lim_{t\to\infty} y_j(t) = 0$ et $|f(x) - y_i(t)| \leq y_j(t)$.*

Nous remarquerons que $\alpha_1, \ldots, \alpha_{n-1}$ sont des paramètres auxiliaires nécessaires pour calculer f.

Le résultat de [Graça, 2004] peut alors se formuler en.

Proposition 3 ([Graça, 2004]) *La fonction Γ est GPAC-calculable.*

4.2.4 Analyse récursive

Le principe sous-jacent à l'analyse récursive est de définir les fonctions calculables sur les réels en considérant des fonctionnelles sur des suites de rationnels rapidement convergentes [Weihrauch, 2000].

Soit $\nu_{\mathbb{Q}} : \mathbb{N} \to \mathbb{Q}$ la représentation suivante[8] des nombres rationnels par les entiers :

$$\nu_{\mathbb{Q}}(\langle p, r, q\rangle) \mapsto \frac{p-r}{q+1},$$

où $\langle ., ., .\rangle : \mathbb{N}^3 \to \mathbb{N}$ est une bijection calculable.

Une suite d'entiers $(x_i)_{i\in\mathbb{N}}$ *représente un réel x* si $(\nu_{\mathbb{Q}}(x_i))$ converge rapidement vers x, noté par $(x_i) \rightsquigarrow x$, dans le sens suivant :

$$\forall i, |\nu_{\mathbb{Q}}(x_i) - x| < \exp(-i).$$

Pour une suite de k-tuples $(\mathbf{x}_i)_{i\in\mathbb{N}}$, nous écrivons $(\mathbf{x}_i) \rightsquigarrow \mathbf{x}$ lorsque cela est vrai composante par composante.

[6] Notons que dans ce chapitre, la terminologie GPAC calculable réfère à cette notion particulière. L'expression "générée par un GPAC"' correspond à la notion de calcul de Shannon en temps réel.

[7] Nous supposons que chacun des $y_k(t)$ est défini pour tout $t \geq 0$.

[8] D'autres représentations naturelles des nombres rationnels peuvent être utilisées, et donnent la même classe de fonctions calculées : voir [Weihrauch, 2000].

Définition 4 (Analyse récursive [Weihrauch, 2000]) *On dit qu'une fonction $f : \mathbb{R}^k \to \mathbb{R}$ est* calculable *s'il existe une fonctionnelle récursive $\Phi : (\mathbb{N}^k)^{\mathbb{N}} \times \mathbb{N} \to \mathbb{N}$ telle que pour tout $\mathbf{x} \in \mathbb{R}^k$, pour toute suite $X = (\mathbf{x}_n) \in (\mathbb{N}^k)^{\mathbb{N}}$, nous avons $(\Phi(X, j))_j \rightsquigarrow f(\mathbf{x})$ lorsque $X \rightsquigarrow \mathbf{x}$.*

Une fonction $f : \mathbb{R}^k \to \mathbb{R}^l$, avec $l > 1$, sera dite calculable, si ses projections le sont.

4.2.5 Notre résultat principal

Dans notre article [Bournez et al., 2006a], nous prouvons que cela est vrai en fait pour toutes les fonctions calculables sur un compact.

En effet, nous montrons que si une fonction réelle f, définie sur un compact, est calculable, alors elle est GPAC-calculable. Réciproquement, nous montrons que si f, définie sur un compact, est GPAC-calculable, alors elle est calculable.

Théorème 10 (GPAC=Analyse Récursive) *Une fonction $f : [a, b] \to \mathbb{R}$ est calculable si et seulement si elle est GPAC calculable.*

Autrement dit, GPAC, problèmes de Cauchy polynomiaux, et analyse récursive sont trois paradigmes de calculs équivalents.

Corollaire 2 (Problèmes de Cauchy polynomiaux=Analyse Récursive) *Une fonction $f : [a, b] \to \mathbb{R}$ est calculable si et seulement si elle est correspond à la limite d'une fonction définissable par un problème de Cauchy polynomial dans le sens de la définition 3.*

Il peut être utile de mettre en rapport ce résultat d'une part sur nos remarques dans les chapitres 1 et 2, sur la puissance de modélisation des problèmes de Cauchy polynomiaux, et d'autre part, sur les liens avec une éventuelle thèse de Church-Turing pour les modèles analogiques.

D'un point de vue des calculs, nos résultats suggèrent que les problèmes de Cauchy polynomiaux et les GPACs sont de vraies contreparties aux machines de Turing.

Nous passons maintenant à une autre série de résultats, reliant fonctions $\mathbb{R}$-récursives et analyse récursive.

4.3 Fonctions $\mathbb{R}$-récursives et analyse récursive

4.3.1 Fonctions $\mathbb{R}$-récursives

Dans [Moore, 1996], Moore a introduit une classe de fonctions sur les réels inspirée de la définition algébrique des fonctions calculables sur les entiers : en observant que l'analogue continu d'une récursion primitive est une équation différentielle (une intégration), Moore propose de considérer la classe des fonctions $\mathbb{R}$-*récursives*, définie comme la plus petite classe de fonctions qui contient certaines fonctions de base, et qui est close par composition, intégration, et par minimisation.

Cette classe de fonctions, aussi étudiée par des articles comme [Mycka, 2003b], [Mycka and Costa, 2006a], [Mycka and Costa, 2006b],[Mycka and Costa, 2005], peut se relier aux fonctions calculées par GPAC (aux fonctions générées par GPAC, selon la terminologie de la section précédente) : voir [Moore, 1996], corrigé par [Graça and Costa, 2003].

Si l'on met de côté les objections à propos de la réalité physique de l'opérateur de minimisation considéré dans l'article initial [Moore, 1996], les définitions originales des classes de [Moore, 1996] souffrent de plusieurs problèmes techniques[9]. Au moins une partie de ces problèmes permet d'utiliser un *"compression trick"* (une autre incarnation du paradoxe de Zénon), pour simuler en un temps borné un nombre non borné de transitions discrètes, et permet de reconnaître des langages arithmétiques.

[9]Par exemple, des fonctions pas bien définies sont considérées, $\infty \times 0$ est toujours considéré comme 0, etc...Certains de ces problèmes sont discutés dans [Campagnolo et al., 2000a], [Campagnolo et al., 2002], [Campagnolo, 2001], et en fait même par son auteur dans [Moore, 1996].

Dans la série d'articles [Campagnolo et al., 2000a], [Campagnolo et al., 2002], [Campagnolo, 2001], Campagnolo, Costa et Moore proposent de considérer la classe $\mathcal{L}$ (bien définie) de fonctions qui correspond à la plus petite classe de fonctions qui ne contient certaines fonctions de base, et qui est close par composition et intégration *linéaire*.

La classe $\mathcal{L}$ est reliée aux fonctions élémentaires sur les entiers en théorie de la calculabilité classique, et aux fonctions sur les réels élémentairement calculables en analyse récursive (les fonctions élémentairement calculables en analyse récursive sont aussi discutées dans [Zhou, 1997]) : toute fonction de la classe $\mathcal{L}$ est élémentairement calculable au sens de l'analyse récursive, et réciproquement, toute fonction sur les entiers élémentaire est la restriction aux entiers d'une fonction qui appartient à la classe $\mathcal{L}$ [Campagnolo et al., 2002], [Campagnolo, 2001].

Ces résultats établissent donc des liens entre les fonctions $\mathbb{R}$-récursives, le GPAC et la calculabilité classique sur les entiers.

4.3.2 Présentation de nos résultats

Les résultats précédents ne fournissent pas une caractérisation de *toutes* les fonctions sur les réels qui sont élémentairement calculables, mais seulement de celles qui préservent les entiers.

Nous avons prouvé qu'il était possible d'aller beaucoup plus loin.

Théorème 11 *Pour les fonctions sur les réels de classe $\mathcal{C}^2$ définies sur un produit d'intervalles compacts à extrémités rationnelles , f est élémentairement calculable dans le sens de l'analyse récursive si et seulement si f appartient à la plus petite classe de fonctions qui contient certaines fonctions de base, et qui est close par composition, intégration linéaire et un schéma limite simple.*

Nous avons étendu ce résultat à une caractérisation de tous les niveaux de la hiérarchie de Grzegorczyk (observons que le théorème précédent peut se voir comme un corollaire du théorème qui suit).

Théorème 12 *Pour les fonctions sur les réels de classe $\mathcal{C}^2$ définies sur un produit d'intervalles compacts à extrémités rationnelles , f est calculable dans le sens de l'analyse récursive dans le niveau $n \geq 3$ de la hiérarchie de Grzegorczyk si et seulement si f appartient à la plus petite classe de fonctions qui contient certaines (autres) fonctions de base, et qui est close par composition, intégration linéaire et un schéma limite simple.*

D'autre part, nous avons étendu cela aux fonctions calculables au sens de l'analyse récursive, et pas seulement aux fonctions *élémentairement calculables*.

Nous avons montré qu'il était possible de définir un schéma de minimisation, assez élégant, pour que :

Théorème 13 *Pour les fonctions sur les réels de classe $\mathcal{C}^2$ définies sur un produit d'intervalles compacts à extrémités rationnelles, f est calculable dans le sens de l'analyse récursive si et seulement si f appartient à la plus petite classe de fonctions qui contient certaines fonctions de base, et qui est close par composition, intégration linéaire, minimisation, et un schéma limite simple.*

Nous présentons maintenant concrètement ces résultats.

4.3.3 Préliminaires mathématiques

Les résultats mathématiques suivants, bien connus, peuvent aider à motiver certains de nos schémas : voir par exemple [Ramis et al., 1995] pour la preuve du premier. Le second, très simple à établir, est prouvé explicitement dans [Bournez and Hainry, 2005].

Lemme 1 (Théorème des fonctions implicites) *Soit $f : \mathcal{D} \times \mathcal{I} \subset \mathbb{R}^{k+1} \to \mathbb{R}$ une fonction, où $\mathcal{D} \times \mathcal{I}$ est un produit d'intervalles fermés, de classe $\mathcal{C}^k$, for $k \geq 1$. Supposons que pour tout $\mathbf{x} \in \mathcal{D}$, l'équation $f(\mathbf{x}, y) = 0$*

possède exactement une solution y_0 et que cet y_0 appartient à l'intérieur de l'intervalle $\mathcal{I}$. Supposons, que pour tout $\mathbf{x}$,

$$\frac{\partial f}{\partial y}(\mathbf{x}, y_0) \neq 0$$

en y_0.

Alors la fonction $g : \mathbb{R}^k \to \mathbb{R}$ qui envoie $\mathbf{x}$ sur la racine correspondante y_0 est définie sur $\mathcal{D}$ et est aussi de classe $\mathcal{C}^k$.

Lemme 2 (Résultat de convergence simple) *Soit $F : \mathbb{R} \times \mathcal{V} \subset \mathbb{R}^{k+1} \to \mathbb{R}^l$ une fonction de classe $\mathcal{C}^1$, et $\beta(\mathbf{x}) : \mathcal{V} \to \mathbb{R}$, $K(\mathbf{x}) : \mathcal{V} \to \mathbb{R}$ des fonctions continues. Supposons que pour tout t et $\mathbf{x}$,*

$$\|\frac{\partial F}{\partial t}(t, \mathbf{x})\| \leq K(\mathbf{x}) \exp(-t\beta(\mathbf{x})).$$

Soit $\mathcal{D}$ le sous ensemble des $\mathbf{x} \in \mathcal{V}$ tels que $\beta(\mathbf{x}) > 0$.
Alors,
- *pour tout $\mathbf{x} \in \mathcal{D}$, $F(t, \mathbf{x})$ possède une limite $L(\mathbf{x})$ en $t = +\infty$.*
- *La fonction $L(\mathbf{x})$ est une fonction continue.*
- *En outre*

$$\|F(t, \mathbf{x}) - L(\mathbf{x})\| \leq \frac{K(\mathbf{x}) \exp(-t\beta(\mathbf{x}))}{\beta(\mathbf{x})}.$$

4.3.4 Théorie classique de la calculabilité

La théorie de la récursion classique parle de fonctions sur les entiers. La plupart des classes de cette théorie peuvent se caractériser comme la clôture d'un ensemble de fonctions de base, par un nombre fini de règles pour former de nouvelles fonctions [Clote, 1998], [Rose, 1984], [Odifreddi, 1992] : étant donnés un ensemble $\mathcal{F}$ de fonctions, et un ensemble $\mathcal{O}$ d'opérateurs sur les fonctions (un opérateur est une fonction qui envoie une ou plusieurs fonctions sur une nouvelle fonction), $[\mathcal{F}; \mathcal{O}]$ désignera la clôture de l'ensemble F par applications des opérations de $\mathcal{O}$.

Proposition 4 (Cas classique [Rose, 1984], [Odifreddi, 1992]) *Soit f une fonction de $\mathbb{N}^k$ vers $\mathbb{N}$ pour $k \in \mathbb{N}$.*

La fonction f est
- élémentaire *si et seulement si elle appartient à*

$$\mathcal{E} = [0, S, U, +, \ominus; \text{COMP}, \text{BSUM}, \text{BPROD}];$$

- *dans la classe $\mathcal{E}_n$ de la hiérarchie de Grzegorczyk ($n \geq 3$) si et seulement si elle appartient à*

$$\mathcal{E}_n = [0, S, U, +, \ominus, E_{n-1}; \text{COMP}, \text{BSUM}, \text{BPROD}];$$

- primitive récursive *si et seulement si elle appartient à*

$$\mathcal{PR} = [0, S, U; \text{COMP}, \text{REC}];$$

- récursive[10] *si et seulement si elle appartient à*

$$\mathcal{R}ec = [0, S, U; \text{COMP}, \text{REC}, \text{MU}].$$

[10]Cette classe est souvent appelée la classe des fonctions *partielles récursives* car elle contient des fonctions partielles, par oppositions aux fonctions totales récursives.

Une fonction $f : \mathbb{N}^k \to \mathbb{N}^l$ *est élémentaire (resp : primitive récursive, récursive,* $\mathcal{E}_n$*) si et seulement si ses projections le sont.*

Les fonctions de base $0, (U_i^m)_{i,m\in\mathbb{N}}, S, +, \ominus$ *et les opérateurs* BSUM, BPROD, COMP, REC, MU *sont donnés par*

1. 0 *est la constante* 0 *;*
2. $U_i^m : \mathbb{N}^m \to \mathbb{N}$, $U_i^m : (n_1, \ldots, n_m) \mapsto n_i$ *;*
3. $S : \mathbb{N} \to \mathbb{N}, S : n \mapsto n + 1$ *;*
4. $+ : \mathbb{N}^2 \to \mathbb{N}$, $+ : (n_1, n_2) \mapsto n_1 + n_2$ *;*
5. $\ominus : \mathbb{N}^2 \to \mathbb{N}$, $\ominus : (n_1, n_2) \mapsto max(0, n_1 - n_2)$ *;*
6. BSUM *: somme bornée. Étant donnée une fonction* f*,* $h = \mathrm{BSUM}(f)$ *est défini par*
$$h : (\mathbf{x}, y) \mapsto \sum_{z<y} f(\mathbf{x}, z);$$
7. BPROD *: produit borné. Étant donnée une fonction* f*,* $h = \mathrm{BPROD}(f)$ *est défini par*
$$h : (\mathbf{x}, y) \mapsto \prod_{z<y} f(\mathbf{x}, z);$$
8. COMP *: composition. Étant données les fonctions* $f_1, \ldots, f_p$ *et* g*,* $h = \mathrm{COMP}(f_1, \ldots, f_p, g)$ *est définie comme la fonction qui satisfait*
$$h(\mathbf{x}) = g(f_1(\mathbf{x}), \ldots, f_p(\mathbf{x}));$$
9. REC *: récursion primitive. Étant données les fonctions* f *et* g*,* $h = \mathrm{REC}(f, g)$ *est défini comme la fonction qui satisfait*
$$\begin{cases} h(\mathbf{x}, 0) & = & f(\mathbf{x}) \\ h(\mathbf{x}, n+1) & = & g(\mathbf{x}, n, h(\mathbf{x}, n)); \end{cases}$$
10. MU *: minimisation. Étant donnée une fonction* f*, la fonction* μf *est définie pour tout les* $\mathbf{x}$ *pour lesquels il* y *a un* y *tel que* $\forall z \leq y$*,* $f(\mathbf{x}, z)$ *est défini et* $f(\mathbf{x}, y) = 0$*. Pour un tel* $\mathbf{x}$*, la minimisation de* f *est donnée par*
$$\mu f : \mathbf{x} \mapsto \inf\{y; f(\mathbf{x}, y) = 0\}.$$
11. *Les fonctions* E_n*, impliquées dans la définition des classes* $\mathcal{E}_n$ *de la hiérarchie de Grzegorczyk , sont définies par récurrence comme suit (lorsque* f *est une fonction,* $f^{[d]}$ *dénote sa dème itérée :* $f^{[0]}(\mathbf{x}) = x$*,* $f^{[d+1]}(\mathbf{x}) = f(f^{[d]}(\mathbf{x}))$*) :*
    (a) $E_0(x, y) = x + y$,
    (b) $E_1(x) = (x + 1) \times (y + 1)$,
    (c) $E_2(x) = 2^x$ *;*
    (d) $E_{n+1}(x) = E_n^{[x]}(1)$ *pour* $n \geq 2$.

On a (voir [Rose, 1984], [Odifreddi, 1992])

$$\mathcal{E} \subseteq \mathcal{PR} \subseteq \mathcal{R}ec,$$

et les inclusions sont strictes. On a aussi

$$\mathcal{E}_3 = \mathcal{E}$$

et

$$\mathcal{PR} = \cup_i \mathcal{E}_i.$$

Si TIME(t) et SPACE(t) dénotent les classes de fonctions qui sont calculables en temps et espace t, alors pour tout $n \geq 3$,

$$\mathcal{E}_n = \text{TIME}(\mathcal{E}_n) = \text{SPACE}(\mathcal{E}_n),$$

et

$$\mathcal{E} = \text{TIME}(\mathcal{E}),$$

$$\mathcal{E} = \text{SPACE}(\mathcal{E})$$

et

$$\mathcal{PR} = \text{TIME}(\mathcal{PR}) = \text{SPACE}(\mathcal{PR}).$$

La classe $\mathcal{PR}$ correspond aux fonctions calculables par des programmes qui utilisent des boucles *For-Next*. La classe $\mathcal{E}$ correspond aux fonctions calculables bornées par une itérée de l'exponentielle. Au plus deux boucles For-Next imbriquées sont nécessaires pour une fonction de la classe $\mathcal{E}$, alors que les fonctions générales de la classe $\mathcal{PR}$ peuvent nécessiter un nombre arbitrairement grand de boucles imbriquées [Rose, 1984], [Odifreddi, 1992].

Terminons cette présentation en observant que l'opérateur de minimisation peut se renforcer en un opérateur de minimisation unique, comme suit.

Proposition 5 *Une fonction f de $\mathbb{N}^k$ vers $\mathbb{N}^l$, pour $k, l \in \mathbb{N}$, est récursive si et seulement si ses projections appartiennent à $[0, U, S; \text{COMP}, \text{REC}, \text{UMU}]$ où l'opérateur* UMU *est défini comme suit :*

1. UMU *: minimisation unique. Étant donnée une fonction f, telle que pour tout $\mathbf{x}$, il y a au plus un y avec $f(\mathbf{x}, y)$ défini et égal à 0, la minimisation unique de f, notée par $!\mu(f)(\mathbf{x})$, est définie pour tous les $\mathbf{x}$ pour lesquels il y a un (unique) y avec $f(\mathbf{x}, y) = 0$. Pour un tel $\mathbf{x}$, $!\mu(f)(\mathbf{x})$ est défini comme cet y.*

En calculabilité classique, des objets plus généraux que les fonctions sur les entiers peuvent être considérés, en particulier des fonctionnelles, c'est-à-dire des fonctions $\Phi : (\mathbb{N}^m)^{\mathbb{N}} \times \mathbb{N}^k \to \mathbb{N}^l$. Une fonctionnelle sera dite *élémentairement* (ou $\mathcal{E}_n$, *primitive récursivement*, *récursivement*) calculable lorsqu'elle appartient[11] à la classe correspondante.

En analyse récursive, une fonction f sur les réels sera dite *élémentairement* (respectivement $\mathcal{E}_n$) calculable si la fonctionnelle correspondante Φ dans la définition 4 l'est. La classe des fonctions calculables (respectivement élémentairement calculables, $\mathcal{E}_n$) sur les réels sera notée $\mathcal{R}ec(\mathbb{R})$ (resp. $\mathcal{E}(\mathbb{R})$, $\mathcal{E}_n(\mathbb{R})$).

4.3.5 Résultats de Campagnolo, Costa, Moore

En suivant les idées de [Moore, 1996], mais en observant que le schéma de minimisation de cet article est problématique, Campagnolo, Costa et Moore ont proposé dans la série [Campagnolo et al., 2000a], [Campagnolo et al., 2002], [Campagnolo, 2001] de ne pas considérer des classes de fonctions sur les réels définies en analogie avec les fonctions récursives, mais seulement en analogie avec des sous-classes. Les fonctions considérées par Campagnolo et al. sont ainsi construites en analogie avec les fonctions élémentaires ou les classes de la hiérarchie de Grzegorczyk.

En outre, ils proposent de restreindre le schéma d'intégration général de [Moore, 1996] en un schéma d'intégration *linéaire*.

Nous appelons *extension réelle d'une fonction* $f : \mathbb{N}^k \to \mathbb{N}^l$ une fonction $\tilde{f}$ de $\mathbb{R}^k$ vers $\mathbb{R}^l$ dont la restriction à $\mathbb{N}^k$ est f.

[11] Formellement, une fonction f sur les entiers peut être considérée comme une fonctionnelle $\overline{f} : (V, \vec{n}) \mapsto f(\vec{n})$. De façon similaire, un opérateur Op sur les fonctions $f_1, \ldots, f_m$ sur les entiers peut être étendu à un opérateur sur les fonctionnelles en fixant le premier argument $\overline{Op}(F_1, \ldots, F_m) : (V, \vec{n}) \mapsto Op(f_1(V, .), \ldots, f_m(V, .))(\vec{n})$.

Dans cet esprit, étant donné un ensemble $\mathcal{F}$ de fonctions de base $\mathbb{N}^k \to \mathbb{N}^l$, et un ensemble $\mathcal{O}$ d'opérateurs sur les fonctions sur les entiers, nous dénoterons toujours, de façon abusive, par $[f_1, \ldots, f_p; O_1, \ldots, O_q]$, la plus petite classe de fonctionnelles qui contient les fonctions de base $\overline{f_1}, \ldots, \overline{f_p}$, plus la fonctionnelle $Map : (V, n) \to V_n$, qui donne le $n$ème élément d'une suite V, et qui est close par les opérateurs $\overline{O_1}, \ldots, \overline{O_q}$. Par exemple, une fonctionnelle sera dite élémentaire si et seulement si elle appartient à $\mathcal{E} = [Map, \overline{0}, \overline{S}, \overline{U}, \overline{+}, \overline{\ominus}; \overline{\text{COMP}}, \overline{\text{BSUM}}, \overline{\text{BPROD}}]$.

Définition 5 ([Campagnolo, 2001], [Campagnolo et al., 2002]) *Soient $\mathcal{L}$ et $\mathcal{L}_n$ les classes de fonctions $f : \mathbb{R}^k \to \mathbb{R}^l$, pour $k, l \in \mathbb{N}$, définies par*

$$\mathcal{L} = [0, 1, -1, \pi, U, \theta_3; \mathrm{COMP}, \mathrm{LI}]$$

et

$$\mathcal{L}_n = [0, 1, -1, \pi, U, \theta_3, \overline{E}_{n-1}; \mathrm{COMP}, \mathrm{LI}]$$

où les fonctions de base 0, 1, -1, π, $(U_i^m)_{i,m\in\mathbb{N}}$, θ_3, $\overline{E}_n$ *et les schémas* COMP *and* LI *sont définis comme suit :*

1. $0, 1, -1, \pi$ *sont les fonctions constantes correspondantes ;*
2. $U_i^m : \mathbb{R}^m \to \mathbb{R}$ *sont, comme dans le cas classique, les projections,* $U_i^m : (x_1, \ldots, x_m) \mapsto x_i$ *;*
3. $\theta_3 : \mathbb{R} \to \mathbb{R}$ *est défini comme* $\theta_3 : x \mapsto x^3$ *si* $x \geq 0, 0$ *sinon.*
4. $\overline{E}_n$ *: pour* $n \geq 3$, $\overline{E}_n$ *est une extension réelle monotone fixée et quelconque de la fonction* $\exp_n$ *sur les entiers définie par induction par* $\exp_2(x) = 2^x$, $\exp_{i+1}(x) = \exp_i^{[x]}(1)$.
5. COMP *: composition. Elle est définie comme dans le cas classique : étant données f et g, $h =$ $\mathrm{COMP}(f, g)$ est la fonction qui satisfait $h(\mathbf{x}) = g(f(\mathbf{x}))$;*
6. LI *: intégration linéaire. A partir des fonctions g et h, $\mathrm{LI}(g, h)$ est la solution maximale de l'équation différentielle linéaire*

$$\frac{\partial f}{\partial y}(\mathbf{x}, y) = h(\mathbf{x}, y) f(\mathbf{x}, y)$$

avec $f(\mathbf{x}, 0) = g(\mathbf{x})$.

Dans ce schéma, si g va vers $\mathbb{R}^n$, $f = \mathrm{LI}(g, h)$ aussi et $h(\mathbf{x}, y)$ est une matrice $n \times n$ dont les éléments sont dans $\mathcal{L}$.

Les classes $\mathcal{L}$ et $\mathcal{L}_n$ contiennent les fonctions usuelles comme $+$, sin, cos, $-$, $\times$, exp, et les fonctions constantes r pour tout $r \in \mathbb{Q}$, mais seulement des fonctions totales, et de classe $\mathcal{C}^2$ [Campagnolo, 2001], [Campagnolo et al., 2002].

Une contribution majeure de [Campagnolo et al., 2002], [Campagnolo, 2001] est de relier ces classes de fonctions sur les réels aux classes de fonctions précédentes sur les entiers. Pour comparer des classes de fonctions sur les réels à des classes de fonctions sur les entiers, nous introduisons la notation suivante : étant donnée une classe $\mathcal{C}$ de fonctions de $\mathbb{R}^k$ vers $\mathbb{R}^l$, nous écrivons $\mathrm{DP}(\mathcal{C})$ (DP pour partie discrète) pour la classe des fonctions de $\mathbb{N}^k$ vers $\mathbb{N}^l$ qui possèdent une extension réelle dans $\mathcal{C}$.

Proposition 6 ([Campagnolo et al., 2002], [Campagnolo, 2001]) *On a*
- $\mathrm{DP}(\mathcal{L}) = \mathcal{E}$ *;*
- $\mathrm{DP}(\mathcal{L}_\mathrm{n}) = \mathcal{E}_\mathrm{n}$.

En fait, ces articles prouvent des inclusions plus fortes.

Proposition 7 ([Campagnolo et al., 2002], [Campagnolo, 2001]) *On a*
- $\mathcal{L} \subset \mathcal{E}(\mathbb{R})$.
- $\mathcal{L}_n \subset \mathcal{E}_n(\mathbb{R})$.

Cependant, il n'y a pas d'espoir d'avoir les inclusions inverses, puisque $\mathcal{E}(\mathbb{R})$ et $\mathcal{E}_n(\mathbb{R})$ contiennent des fonctions partielles, alors que les classes $\mathcal{L}$ et $\mathcal{L}_n$ sont des classes de fonctions totales.

4.3.6 Caractérisation des fonctions élémentairement calculables

Nous avons proposé de considérer de nouvelles classes de fonctions qui s'avèrent correspondre précisément aux classes $\mathcal{E}(\mathbb{R})$ et $\mathcal{E}_n(\mathbb{R})$.

Pour cela, nous nous restreignons aux fonctions définies sur un domaine compact. Une motivation est que les fonctions élémentairement calculables sur un domaine quelconque ne sont pas stables par composition.

Pour des raisons techniques, nous avons besoin (en fait plutôt pour la section suivante) de remplacer le schéma LI précédent, par le schéma CLI suivant.

Définition 6 (Schéma CLI **)** *A partir des fonctions g, h, et c, avec*

- *la norme de chacune des dérivées partielles de h bornée par c, sauf éventuellement sa dérivée partielle en t,*

$\mathrm{CLI}(g,h,c)$ *est solution*[12] *de l'équation différentielle linéaire*

$$\frac{\partial f}{\partial y}(\mathbf{x},y) = h(\mathbf{x},y)f(\mathbf{x},y)$$

avec $f(\mathbf{x},0) = g(\mathbf{x})$.

Dans ce schéma, si g va vers $\mathbb{R}^n$, $f = \mathrm{CLI}(g,h,c)$ va vers $\mathbb{R}^{n+1}$ et $h(\mathbf{x},y)$ est une matrice $(n+1)\times(n+1)$ dont les éléments sont dans $\mathcal{L}$.

On peut montrer que remplacer le schéma LI par le schéma CLI ne change rien à la discussion précédente.

Proposition 8

$$\begin{aligned}\mathcal{L} &= [0,1,-1,\pi,U,\theta_3;\mathrm{COMP},\mathrm{LI}]\\ &= [0,1,-1,\pi,U,\theta_3;\mathrm{COMP},\mathrm{CLI}]\end{aligned}$$

et

$$\begin{aligned}\mathcal{L}_n &= [0,1,-1,\pi,U,\theta_3,\overline{E}_{n-1};\mathrm{COMP},\mathrm{LI}]\\ &= [0,1,-1,\pi,U,\theta_3,\overline{E}_{n-1};\mathrm{COMP},\mathrm{CLI}]\end{aligned}$$

D'autre part, nous introduisons un schéma limite. L'idée de considérer un schéma limite a déjà été envisagée dans des articles comme [Mycka and Costa, 2004], [Mycka, 2003b]. Cependant, puisque nous sommes intéressés par les fonctions $\mathbb{R}$-sous-récursives, et non pas à construire une hiérarchie au-dessus des fonctions $\mathbb{R}$-récursives, notre schéma limite est plus restreint que celui de ces articles.

Les conditions que nous imposons sur notre schéma LIM_w sont inspirées par le lemme 2.

Définition 7 (Schéma LIM**)** *Soient $f : \mathbb{R}\times\mathcal{D}\subset\mathbb{R}^{k+1}\to\mathbb{R}^l$ et $K : \mathcal{D}\to\mathbb{R}$ deux fonctions, et $\beta : \mathcal{D}\to\mathbb{R}$ une fonction polynôme, avec l'hypothèse suivante : pour tout $\mathbf{x}$, $t \geq \|\mathbf{x}\|$|,*

$$\|\frac{\partial f}{\partial t}(t,\mathbf{x})\| \leq K(\mathbf{x})\exp(-t\beta(\mathbf{x})).$$

Alors, sur tout produit d'intervalles fermés $I\subset\mathbb{R}^k$ sur lequel $\beta(\mathbf{x}) > 0$,

$$F(\mathbf{x}) = \lim_{t\to+\infty} f(t,\mathbf{x})$$

existe par le lemme 2.

Si F est de classe $\mathcal{C}^2$, alors nous définissons $\mathrm{LIM}_w(f,K,\beta)$ comme cette fonction $F : I\to\mathbb{R}$.

Nous pouvons définir nos classes.

[12] En fait n'importe quelle restriction à un produit d'intervalles fermés de la solution maximale

Définition 8 (Classes $\mathcal{L}^*$, $\mathcal{L}^*_n$) *Les classes $\mathcal{L}^*$, et $\mathcal{L}^*_n$, pour $n \geq 3$, de fonctions de $\mathbb{R}^k$ vers $\mathbb{R}^l$, pour $k, l \in \mathbb{N}$, sont définies comme les classes suivantes.*

$$\mathcal{L}^* = [0, 1, -1, U, \theta_3; \mathrm{COMP}, \mathrm{CLI}, \mathrm{LIM}],$$

et

$$\mathcal{L}^*_n = [0, 1, -1, U, \theta_3, \overline{E}_{n-1}; \mathrm{COMP}, \mathrm{CLI}, \mathrm{LIM}].$$

Ces classes peuvent maintenant contenir des fonctions partielles, et étendent les classes précédentes.

Proposition 9

$$\mathcal{L} \subsetneq \mathcal{L}^*,$$

et

$$\mathcal{L}_n \subsetneq \mathcal{L}^*_n$$

pour tout $n \geq 3$.

D'autre part, un de nos résultats majeurs, prouvé dans nos publications [Bournez and Hainry, 2005] et [Bournez and Hainry, 2004a], est que ces classes caractérisent les fonctions élémentairement calculables en analyse récursive. Les théorèmes 2 et 3 précédents sont formellement.

Théorème 2 (Caractérisation de $\mathcal{E}(\mathbb{R})$) *Soit $f : \mathcal{D} \subset \mathbb{R}^k \to \mathbb{R}^l$ une fonction sur les réels de classe $\mathcal{C}^2$, avec $\mathcal{D}$ un produit d'intervalles compacts à extrémités rationnelles.*

f est dans $\mathcal{E}(\mathbb{R})$ si et seulement si $f \in \mathcal{L}^*$.

En fait cela se généralise à tous les niveaux de la hiérarchie de Grzegorczyk (observons que le théorème 2 est le cas particulier $n = 3$ du théorème 3).

Théorème 3 (Caractérisation de $\mathcal{E}_n(\mathbb{R})$) *Soit $f : \mathcal{D} \subset \mathbb{R}^k \to \mathbb{R}^l$ une fonction sur les réels de classe $\mathcal{C}^2$, avec $\mathcal{D}$ un produit d'intervalles compacts à extrémités rationnelles. Soit $n \geq 3$.*

f est dans $\mathcal{E}_n(\mathbb{R})$ si et seulement si $f \in \mathcal{L}^*_n$.

Quelques extensions de ces résultats seront trouvées dans ces articles, en particulier avec un théorème de forme normale pour les fonctions de nos classes, et quelques schémas alternatifs à notre schéma limite, comme des schémas de recherche de minimum de fonctions convexes.

4.3.7 Caractérisation des fonctions calculables

Nous avons d'autre part ultérieurement montré qu'il était possible de caractériser les fonctions calculables, et pas seulement les fonctions élémentairement calculables.

Pour cela, nous devons introduire un opérateur de minimisation qui permet de simuler les minimisations discrètes sur les entiers.

Cependant, cet opérateur doit être plus restreint que la simple idée de retourner "le plus petit zéro", puisque cette idée, source des investigations dans [Moore, 1996], a montré être la cause de nombreux problèmes, discutés dans [Campagnolo, 2001], [Campagnolo et al., 2002], [Mycka, 2003b], [Mycka, 2003a]. Les articles [Mycka, 2003b], [Mycka and Costa, 2004] présentent une alternative bien fondée, en remplaçant la minimisation par une opération de passage à la limite. Nous proposons ici de nous en tenir à un schéma de recherche de zéro, mais restreint par rapport à celui de [Moore, 1996].

Notre idée est d'utiliser le schéma alternatif UMU schéma qui est équivalent au schéma MU en calculabilité classique (voir la proposition 5), mais qui a une contrepartie qui s'avère préserver les fonctions calculables en analyse récursive.

En effet, motivés par la proposition 5, par le lemme 1 (le théorème des fonctions implicites), et par les résultats de l'analyse récursive sur la calculabilité des zéros d'une fonction (voir par exemple [Weihrauch, 2000] où les théorèmes 6.3.5 et 6.3.8 disent que la recherche d'un zéro unique est calculable), nous définissons notre opérateur de recherche de zéro UMU comme suit.

Définition 9 (Schéma UMU**)** *Étant donnée une fonction différentiable f de $\mathcal{D} \times \mathcal{I}) \subset \mathbb{R}^{k+1}$ vers $\mathbb{R}$ où $\mathcal{D} \times \mathcal{I}$ est un produit d'intervalles fermés,*
si pour tout $\mathbf{x} \in \mathcal{D}$,

- *$y \mapsto f(\mathbf{x}, y)$ est une fonction non-décroissante,*
- *avec une unique racine y_0 sur $\mathcal{I}$,*
- *et telle que y_0 soit à l'intérieur de l'intervalle $\mathcal{I}$,*
- *avec*

$$\frac{\partial f}{\partial y}(\mathbf{x}, y_0) > 0,$$

alors UMU(f) *est défini sur $\mathcal{D}$ comme suit :*

$$\mathrm{UMU}(f) : \left\{ \begin{array}{ccc} \mathcal{D} & \longrightarrow & \mathbb{R} \\ \mathbf{x} & \mapsto & y_0 \text{ tel que } f(\mathbf{x}, y_0) = 0. \end{array} \right.$$

On définit alors.

Définition 10 (Classe $\mathcal{L}+!\mu$) *Soit $\mathcal{L}+!\mu$ l'ensemble de fonctions sur les réels défini par*

$$\mathcal{L}+!\mu = [0, 1, U, \theta_3; \mathrm{COMP}, \mathrm{CLI}, \mathrm{UMU}].$$

On peut montrer que

$$\mathcal{L} \subset \mathcal{L}+!\mu,$$

et que $\mathcal{L}+!\mu$ ne contient que des fonctions de classe $\mathcal{C}^2$, définies sur des produits d'intervalles fermés.

Notre résultat principal dans [Bournez and Hainry, 2004b], constitue la formalisation du théorème 4 informel précédent en les théorèmes 4 et 5 qui suivent.

Théorème 4 *Pour les fonctions totales*

$$\mathcal{R}ec = DP(\mathcal{L}+!\mu).$$

I.e.

- *Si une fonction de $\mathcal{L}+!\mu$ étend une fonction totale sur les entiers, alors cette fonction est totale récursive.*
- *Toute fonction totale récursive sur* les entiers*, possède une extension réelle dans $\mathcal{L}+!\mu$.*

En ajoutant un opérateur limite, et en combinant avec les constructions de [Bournez and Hainry, 2004a], [Bournez and Hainry, 2005], nous avons obtenu notre résultat majeur à propos des fonctions $\mathbb{R}$-récursives.

Définition 11 (Classe $\mathcal{L}+!\mu + \mathrm{LIM}_w$)

$$\mathcal{L}^*_{!\mu} = [0, 1, U, \theta_3; \mathrm{COMP}, \mathrm{CLI}, \mathrm{UMU}, \mathrm{LIM}_w]$$

Nous avons le théorème suivant, qui constitue notre résultat majeur dans [Bournez and Hainry, 2005].

Théorème 5 *Pour les fonctions de classe $\mathcal{C}^2$ définies sur un produit d'intervalles compacts à extrémités rationnelles,*

$$\mathcal{L}^*_{!\mu} = \mathcal{R}ec(\mathbb{R}).$$

Nous avons proposé quelques extensions de ce résultat dans cet article.

En d'autres termes, nous savons relier les fonctions calculables en analyse récursive aux fonctions $\mathbb{R}$-récursives.

4.4 Discussions

En ce qui concerne les *modèles analogiques*, tous ces résultats ont plusieurs impacts. Premièrement, ils contribuent à comprendre les modèles analogiques, en particulier les relations entre les fonctions GPAC calculables, les fonctions $\mathbb{R}$-récursives, et les fonctions calculables en analyse récursive. En outre, ils prouvent qu'aucun phénomène d'hypercalcul ne peut se produire pour ces classes de fonctions. En particulier, on a affaire à des classes de fonctions qui sont robustes dans un certain sens. Voir la discussion du chapitre 3, à propos des notions de robustesse.

En ce qui concerne *l'analyse récursive*, nos résultats fournissent une caractérisation *purement algébrique* des fonctions élémentairement calculables sur les réels. Observons les bénéfices potentiels offerts par ces caractérisations par rapport aux définitions classiques en analyse récursive, qui requierent des discussions à propos de machines de Turing d'ordre supérieur (ou de type 2) (voir par exemple [Weihrauch, 2000]), ou par rapport aux caractérisations dans l'esprit de [Brattka, 2003], [Kawamura, 2005].

D'autre part, nous observerons que nous montrons que les fonctions sur les réels calculables peuvent êtres définies par des schémas *continus*, c'est-à-dire sur des fonctions continues. Il est nettement plus naturel d'utiliser des schémas continus pour définir des fonctions continues, que d'utiliser des schémas limites sur des fonctions discrètes [Brattka, 2003], [Kawamura, 2005].

4.5 Une théorie de la complexité ?

Plus généralement, dans ce chapitre, nous avons relié plusieurs modèles, à priori distincts, de calculs à temps continu.

Tout d'abord, nous avons rappelé que les fonctions générées par le GPAC correspondent aux problèmes de Cauchy polynomiaux. Ensuite, nous avons montré que si l'on prend une notion de calcul par calculs convergents, alors le GPAC et les problèmes de Cauchy polynomiaux ont exactement la puissance de l'analyse récursive : les fonctions calculables sont les mêmes. D'autre part, nous avons aussi relié cette puissance à celle de classes de fonctions $\mathbb{R}$-récursives. Nous avons ainsi obtenu la première caractérisation algébrique, par l'analyse, des fonctions calculables en analyse récursive.

L'enjeu pour comprendre une éventuelle thèse de Church-Turing pour les modèles analogiques serait d'arriver à relier ces modèles aux autres modèles évoqués dans les autres chapitres de ce document.

Mais nous croyons qu'une question très importante est d'arriver à comprendre si on peut passer de la calculabilité à la complexité. Peut-on définir une notion de complexité, valide, et universellement acceptée pour les modèles analogiques ?

Nous avons présenté les résultats existants à ce jour dans le chapitre 3, et déja discuté de nombreux problèmes ouverts. Mais les résultats de ce chapitre, laissent entrevoir des pistes.

Par exemple, l'ouvrage [Ko, 1991] présente une définition robuste de ce que sont les fonctions calculables en temps polynomial en analyse récursive. Pourrait-on caractériser les classes de complexité de l'analyse récursive algébriquement avec des schémas comme ceux évoqués ici ?

Est-il possible de relier les fonctions calculables en temps polynomial à une classe de fonctions GPAC calculables où l'erreur ε serait donnée par une fonction polynomiale en t ?

Et si de tels résultats étaient possibles, est-ce qu'ils pourraient se généraliser à d'autres classes de complexité ?

Des pistes pour cela sont données par les caractérisations algébriques en complexité classique des classes de complexité. Par exemple, la caractérisation de Bellantoni et Cook du temps polynomial en complexité classique, en termes de fonctions récursives sûres dans [Bellantoni and Cook, 1992b]. Peut-on adapter ces caractérisations pour définir des classes de fonctions $\mathbb{R}$-récursives similaires au temps polynomial ? à l'espace polynomial ?

Le chapitre suivant est précisément relatif aux caractérisations dans l'esprit de Bellantoni et Cook, mais pour une autre classes de modèles de calculs : le modèle de Blum Shub et Smale, qui lui est à espace continu mais à temps discret.

Chapitre 5

Caractérisations syntaxiques des classes de complexité dans le modèle BSS

Dans ce chapitre nous nous focalisons sur le modèle de Blum Shub et Smale [Blum et al., 1989]. Ce modèle constitue un modèle de calculs à temps discret sur les réels qui a été introduit initialement pour discuter de la complexité de problèmes sur les polynômes [Blum et al., 1989], [Blum et al., 1998]. Il a par la suite été étendu par Poizat dans [Poizat, 1995], [Goode, 1994] en un modèle de calculs sur une structure logique arbitraire.

Nous présentons plusieurs caractérisations syntaxiques des classes de complexité dans ce modèle. Il se veut un catalogue de nos résultats relatifs à la caractérisation de classes de complexité dans ce modèle. Tous les résultats présentés dans ce chapitre sont dans l'esprit de l'article [Bellantoni and Cook, 1992a].

5.1 Introduction

Le modèle BSS est une des approches pour étudier la complexité ou la calculabilité de problèmes sur les réels. La plus connue, est l'approche de l'analyse récursive, introduite par Turing [Turing, 1936], Grzegorczyk [Grzegorczyk, 1957], et Lacombe [Lacombe, 1955]. Dans celle-ci, on considère qu'un réel est représenté par une suite de nombres rationnels rapidement convergente, et l'on dit qu'une fonction est calculable s'il existe un moyen effectif de transformer toute représentation d'un réel en une représentation de l'image du réel par la fonction : voir [Weihrauch, 2000] pour livre récent présentant l'analyse récursive. On peut aussi définir dans ce modèle une notion de complexité [Ko, 1991].

Blum, Shub et Smale ont proposé en 1989 une autre approche pour parler de la complexité de tels problèmes, en introduisant un modèle de calcul dans [Blum et al., 1989], que l'on appelle parfois la machine de Turing réelle. Ce modèle, contrairement à l'analyse récursive, mesure la complexité des problèmes en termes du nombre d'opérations arithmétiques nécessaires à leur résolution indépendamment des représentations des réels.

Le modèle, défini initialement pour parler de complexité algébrique de problèmes sur le corps des réels, ou plus généralement sur un anneau, a été par la suite été étendu par Poizat dans [Poizat, 1995, Goode, 1994] en un modèle de calculs sur une structure logique arbitraire.

Suivant la structure logique considérée, on peut obtenir toute une théorie de la complexité, avec des classes de complexité comme P, NP, des notions de réductions, des problèmes complets, des grandes questions comme $P \neq NP$?, dont il est parfois possible de répondre affirmativement ou négativement.

Comme la complexité classique peut se voir comme la restriction de cette notion de complexité au cas particulier des structures booléennes, ce modèle apporte un éclairage nouveau sur les problèmes plus anciens de la complexité classique, et sur ses liens avec la logique.

En particulier, cela ouvre le champ à de nombreux travaux cherchant à comprendre les résultats de la complexité classique qui se généralisent à d'autres structures que les booléens, et les structures logiques dans lesquelles on peut répondre aux grandes questions de la complexité comme la question $P = NP$? : voir l'ouvrage [Blum et al., 1998].

Par l'intermédiaire de la thèse de Paulin Jacobé de Naurois, avec Felipe Cucker à Hong-Kong et Jean-Yves Marion, à Nancy, nous avons cherché à comprendre si il était possible de caractériser syntaxiquement les classes de complexité dans ce modèle sur une structure arbitraire.

Puisqu'il existe en complexité classique plusieurs telles caractérisations des classes de complexité, la question peut se voir comme celle de comprendre si ces résultats s'étendent à des structures logiques arbitraires.

Une autre motivation forte est la suivante : les caractérisations syntaxiques des classes de complexité définissent ces classes sans référence à une notion explicite de machine. Puisqu'il y a plusieurs modèles de calculs sur les réels, cela ajoute à la légitimité des classes de complexité considérées. En effet, puisque les relations entre les modèles de calculs sur les réels sont loin d'être toutes claires, pouvoir définir les classes de complexité sans fixer le modèle de calcul, permet de s'affranchir du problème, et de légitimer l'idée que la notion de classe de complexité obtenue est bien indépendante de toutes les variations envisageables sur les modèles.

On observera en outre, qu'on peut considérer qu'on obtient des définitions des classes de complexité qui sont bien aussi naturelles, voir plus naturelles parfois, que les définitions classiques.

En se basant sur la caractérisation de Bellantoni et Cook du temps polynomial en complexité classiuqe dans [Bellantoni and Cook, 1992a], nous avons tout d'abord obtenu une caractérisation syntaxique des fonctions calculables, ainsi que des fonctions calculables en temps polynomial en termes de fonctions récursives sûres.

Nous avons ultérieurement obtenu une caractérisation des fonctions calculables en temps parallèle polynomial en termes de fonctions récursives sûres avec substitutions. Ce résultat généralise le résultat de [Leivant and Marion, 1995] au cas des structures arbitraires.

En généralisant les résultats de [Bellantoni, 1994], nous avons en outre obtenu une caractérisation de chacun des niveaux de la hiérarchie polynomiale en termes de récursions sûres avec minimisations prédicatives, et de la hiérarchie digitale polynomiale en termes de récursions sûres avec minimisations prédicatives digitales.

Nous avons d'autre part caractérisé le temps alternant polynomial en termes de récursions sûres avec substitutions prédicatives, et le temps digital alternant polynomial en termes de récursions sûres avec substitutions prédicatives digitales.

Nous reprenons dans ce chapitre de façon synthétique l'ensemble de nos résultats. Les résultats présentés ici sont publiés dans les articles [Bournez et al., 2006b], [Bournez et al., 2005a], et dans les congrès [Bournez et al., 2003], [Bournez et al., 2004a], et [Bournez et al., 2004b].

Toutes ces caractérisations sont en droite ligne des caractérisations, dites de la complexité implicite, amorcées par les travaux de Bellantoni et Cook [Bellantoni and Cook, 1992a]. La toute première caractérisation du temps polynomial par une algèbre de fonctions sans référence explicite à un modèle de calcul est due à Cobham dans [Cobham, 1962]. Toutefois les schémas de [Bellantoni and Cook, 1992a] sont nettement plus naturels.

D'autres caractérisations indépendantes de notions de machines existent en complexité classique : voir par exemple les monographies et survols [Clote, 1998], [Ebbinghaus and Flum, 1995], [Immerman, 1999]. En particulier, il y a tout le pan entier de recherche qui concerne les caractérisations de la complexité descriptive, basées sur des relations ou méthodes globales de la théorie des modèles finis.

La complexité descriptive est née des travaux de Fagin [Fagin, 1974], qui prouvent que la classe NP peut se caractériser comme la classe des ensembles définissables en logique existentielle du second ordre. Vardi et Immerman dans l'article [Vardi, 1982], [Immerman, 1987], [Immerman, 1986] ont utilisé cette approche pour caractériser la classe P. Plusieurs autres caractérisations existent, pour des classes comme $LOGSPACE$, dans [Gurevich, 1983] ou pour la classe $PSPACE$, dans la série [Moschovakis, 1984], [Gurevich and Shelah, 1986], [Immerman, 1987], [Bonner, 1989], [Abiteboul and Vianu, 1989], [Leivant, 1990], [Abiteboul et al., 1990] ou encore dans [Abiteboul and Vianu, 1991], [Immerman, 1991]. Une présentation synthétique du domaine peut se trouver dans [Ebbinghaus and Flum, 1995], [Immerman, 1999], [Clote, 1998]. Toutes ces caractérisations sont dans le cadre classique.

Dans [Grädel and Meer, 1995], la notion de $\mathbb{R}$-structure a été introduite, et des caractérisations des classes P et NP sur le corps des réels en termes de logiques sur ces $\mathbb{R}$-structures ont été établies. Ces résultats ont par la suite été étendus dans [Cucker and Meer, 1999] à d'autres classes de complexité, et dans [Grädel and Gurevich, 1998] à d'autres structures que le corps des réels.

Nous sommes aussi coauteurs d'une extension de la notion de $\mathbb{R}$-structure aux structures arbitraires. Cependant, nous avons décidé de nous focaliser sur les approches à la Bellantoni et Cook dans ce chapitre, et donc de ne pas évoquer ces caractérisations dans ce chapitre. Nous renvoyons à [Bournez et al., 2005b] pour ces caractérisations.

Nous ajouterons à ces arguments de motivation, que pour la question de savoir pourquoi s'intéresser au modèle BSS, le chapitre 1 a montré à plusieurs reprises que les polynômes apparaissaient naturellement lorsqu'on discute de problèmes sur les réels, et même souvent lorsqu'on ne s'y attend pas. Ce modèle est clairement plus naturel que l'analyse récursive pour discuter de la complexité[1] de problèmes reliés aux polynômes.

D'autre part, dans une perspective de programmation, une façon d'interpréter tous ces résultats est de voir la calculabilité sur une structure arbitraire comme un langage de programmation avec des opérateurs extra venant d'une librairie externe. Cette observation et son potentiel pour la construction de méthodes, pour dériver automatiquement des propriétés des programmes, dans l'esprit de [Hofmann, 1999], [Jones, 2001], est aussi une des motivations de ce travail.

Tous les résultats de ce chapitre sont le fruit (d'une partie) du travail de thèse de Paulin de Naurois. Ils sont aussi obtenus en collaboration avec ses encadrants Felipe Cucker, Jean-Yves Marion et nous-mêmes.

5.2 Calculs sur une structure arbitraire

Commençons par introduire brièvement la calculabilité et la complexité sur une structure arbitraire. Pour de plus amples détails, nous renvoyons au livre [Blum et al., 1998], pour des structures en rapport avec les nombres réels ou les nombres complexes, ou au livre [Poizat, 1995] pour des considérations sur des structures plus générales.

Définition 12 *Une* structure $\mathcal{K} = (\mathbb{K}, op_1, \cdots, op_k, rel_1, \cdots, rel_l, \{c_i\}_{i\in I}, \mathbf{0}, \mathbf{1})$ *est donnée par*
- *un ensemble sous-jacent* $\mathbb{K}$,
- *un nombre fini d'opérateurs* $op_1, \cdots, op_k$ *d'arités plus grandes que* 1,
- *des constantes* $\{c_i\}_{i\in I}$,
- *et un nombre fini de relations* $rel_1, \ldots, rel_l$.

Les constantes correspondent à des opérateurs d'arités nulles. Alors que nous autorisons l'ensemble des indices I *a être infini, le nombre d'opérateurs avec une arité plus grande que* 1 *est supposé fini, c'est-à-dire, seul le nombre de symboles pour les constantes peut être infini.*

Nous ne distinguerons pas entre les opérateurs et les symboles de relations et leurs interprétations correspondantes, comme fonctions et relations sur l'ensemble sous-jacent $\mathbb{K}$. Nous supposerons que la relation d'égalité $=$ est une relation de la structure, et qu'il y a toujours au moins deux symboles de constantes, avec des interprétations distinctes (notés par $\mathbf{0}$ et $\mathbf{1}$) dans la structure.

Un exemple de structure est donné par $\mathcal{K} = (\mathbb{R}, +, -, *, =, \leq, \{c \in \mathbb{R}\})$. Cela correspond à la structure de l'article initial [Blum et al., 1989].

Un autre exemple qui correspond à la complexité et la calculabilité classique est $\mathcal{K} = (\{0,1\}, =, \mathbf{0}, \mathbf{1})$.

Nous noterons par $\mathbb{K}^* = \bigcup_{i\in\mathbb{N}} \mathbb{K}^i$ l'ensemble des mots sur l'alphabet $\mathbb{K}$. L'espace $\mathbb{K}^*$ est l'analogue de Σ^*, l'ensemble de toutes les suites finies de 0 et de 1. Il constitue l'espace des entrées des machines sur $\mathcal{K}$.

Pour des raisons techniques, nous considérerons aussi la somme directe biinfinie $\mathbb{K}_*$: les éléments de cet espace sont de la forme

$$(\ldots, x_{-2}, x_{-1}, x_0, x_1, x_2, \ldots)$$

[1] Et pas nécessairement de la calculabilité.

où $x_i \in \mathbb{K}$ pour tout $i \in \mathbb{Z}$ et $x_k = 0$ pour k suffisamment grand en valeur absolue. La *composante d'indice i* d'un tel élément désignera x_i.

L'espace $\mathbb{K}_*$ possède des opérations naturelles de décalages, décalage à gauche $\sigma_\ell : \mathbb{K}_* \to \mathbb{K}_*$ et décalage à droite $\sigma_r : \mathbb{K}_* \to \mathbb{K}_*$ où

$$\sigma_\ell(x)_i = x_{i-1} \qquad \text{and} \qquad \sigma_r(x)_i = x_{i+1}.$$

Dans ce qui suit, les mots d'éléments de $\mathbb{K}$ seront représentés par des lettres surlignées, alors que les éléments de $\mathbb{K}$ seront représentés par des lettres simples. Par exemple, $a.\overline{x}$ désigne le mot dans $\mathbb{K}^*$ dont la première lettre est a et qui se termine par le mot $\overline{x}$. Le mot vide sera noté ϵ. La longueur d'un mot $\overline{w} \in \mathbb{K}^*$ sera notée $|\overline{w}|$.

Nous définissons maintenant les machines sur $\mathcal{K}$ selon la présentation de [Blum et al., 1998].

Définition 13 *Un* machine sur $\mathcal{K}$ *consiste en un espace d'entrée $\mathcal{I} = \mathbb{K}^*$, un espace de sortie $\mathcal{O} = \mathbb{K}^*$, et un espace de registres*[2] *$\mathcal{S} = \mathbb{K}_*$, avec un graphe orienté connexe dont les noeuds, étiquetés $0, \ldots, N$, correspondent à l'ensemble des différentes* instructions *de la machine. Les noeuds sont de l'un des cinq types suivant : entrée, sortie, calcul, branchement, et décalage. Décrivons les un peu plus.*

1. *Noeud d'entrée. Il y a un seul noeud d'entrée, étiqueté par 0. Associée à ce noeud, il y a le noeud suivant $\beta(0)$, et la fonction d'entrée $g_I : \mathcal{I} \to \mathcal{S}$.*
2. *Noeud de sortie. Il y a un seul noeud de sortie qui est étiqueté par 1. Il n'a pas de noeuds successeur, puisqu'une fois qu'il est atteint le calcul s'arrête, et la fonction de sortie $g_O : \mathcal{S} \to \mathcal{O}$ place le résultat du calcul dans l'espace de sortie.*
3. *Noeud de calcul. A un noeud m de ce type sont associés un noeud suivant $\beta(m)$ et une fonction $g_m : \mathcal{S} \to \mathcal{S}$. La fonction g_m remplace la composante d'indice 1 de $\mathcal{S}$ par la valeur $op(w_1, \ldots, w_n)$ où $w_1, w_2, \ldots, w_n$ sont les composantes d'indices 1 à n de $\mathcal{S}$ et op est une opération de la structure $\mathcal{K}$ d'arité n. Les autres composantes de $\mathcal{S}$ sont inchangées. Quand l'arité n vaut 0, m est un noeud constant*[3]*.*
4. *Noeud de branchement. Il y a deux noeuds associés à un noeud m de ce type : $\beta^+(m)$ et $\beta^-(m)$. Le prochain noeud est $\beta^+(m)$ si $rel(w_1, \ldots, w_n)$ est vrai et $\beta^-(m)$ sinon. Ici $w_1, w_2, \ldots, w_n$ sont les composantes d'indices 1 à n de $\mathcal{S}$ et rel une relation de la structure $\mathcal{K}$ d'arité n.*
5. *Noeud de décalage. A un noeud m de ce type sont associés un noeud suivant $\beta(m)$ et une fonction $\sigma : \mathcal{S} \to \mathcal{S}$. La fonction σ est soit un décalage à droite, soit un décalage à gauche.*

Plusieurs conventions pour le contenu de l'espace des registres au début du calcul ont été utilisées dans la littérature [Blum et al., 1998], [Blum et al., 1989], [Poizat, 1995]. Nous nous n'intéresserons pas à ces détails, mais nous nous focaliserons sur les idées essentielles dans ce qui suit.

Autrement dit, une machine sur $\mathcal{K}$ est essentiellement une machine de Turing, qui est capable de réaliser les opérations de base $\{op_i\}$ et les tests de base $rel_1, \ldots, rel_l$ de la structure à un coût unitaire, et dont le ruban peut contenir dans ses cases des éléments arbitraires de l'ensemble sous-jacent $\mathbb{K}$ [Poizat, 1995], [Blum et al., 1998]. Observons que l'espace des registres $\mathcal{S}$ ci-dessus a la fonction d'un ruban, et que la composante d'indice 1 joue le rôle de la case en face de la tête de lecture.

Définition 14 *Pour une machine M, la fonction φ_M associant la sortie à une entrée donnée $x \in \mathbb{K}^*$ est appelée* la fonction d'entrée sortie.

Nous dirons qu'une fonction $f : \mathbb{K}^* \to \mathbb{K}^*$ est calculable *lorsqu'il y a une machine M telle que $f = \varphi_M$.*

Nous dirons qu'un ensemble $A \subseteq \mathbb{K}^*$ est décidé *par une machine M si sa fonction caractéristique $\chi_A : \mathbb{K}^* \to \{\mathbf{0}, \mathbf{1}\}$ coïncide avec φ_M.*

[2] Dans le papier original de Blum, Shub et Smale, cela est appelé l'espace des états. Nous le renommons en espace des *registres* pour éviter la confusion avec la notion d'état dans une machine de Turing.

[3] Une machine donnée utilise un nombre fini de constantes. Cependant, pour pouvoir comparer différentes machines et parler de réductions entre elles, nous avons besoin d'inclure toutes les constantes possibles dans la structure sous-jacente $\mathcal{K}$. D'où l'ensemble d'indices I potentiellement infini.

Nous pouvons maintenant définir les classes de complexité principales.

Définition 15 *Un ensemble $S \subset \mathbb{K}^*$ est dans la classe $\mathrm{P}_{\mathcal{K}}$ (respectivement une fonction $f : \mathbb{K}^* \to \mathbb{K}^*$ est dans la classe FP_K), s'il existe un polynôme p et une machine M, tels que pour tout $\overline{w} \in \mathbb{K}^*$, M s'arrête en temps $p(|\overline{w}|)$ et M accepte si et seulement si $\overline{w} \in S$ (respectivement M calcule la fonction $f(\overline{w})$).*

Cette notion de calcul correspond à la notion classique pour les structures dont l'espace sous-jacent est fini, ou sur les entiers. Nous avons donc bien affaire à un modèle qui généralise la calculabilité et la complexité classique aux structures arbitraires.

Proposition 10 ([Blum et al., 1998], [Poizat, 1995]) **(i)** *La classe $\mathrm{P}_{\mathcal{K}}$ correspond à la classe* P *classique pour $\mathcal{K} = (\{0,1\}, =, \mathbf{0}, \mathbf{1})$.*

(ii) *La classe $\mathrm{P}_{\mathcal{K}}$ correspond à la classe $\mathrm{P}_{\mathbb{R}}$ de [Blum et al., 1989] pour $\mathcal{K} = (\mathbb{R}, +, -, *, =, \leq, \{c \in \mathbb{R}\})$.*

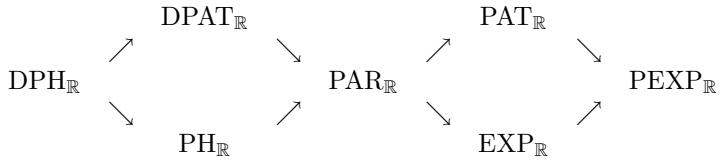
Il est aussi possible de considérer l'analogue de toutes les classes connues en complexité classique.

Pour les classes non-déterministes, une subtilité est que deux types de non-déterminisme peuvent être considérés suivant si les témoins sont autorisés a être des éléments arbitraires de la structure ou sont restreints à être dans $\{0,1\}$.

Les classes correspondantes au second cas sont souvent dites *digitales* et une lettre D est souvent accolée à leur acronyme.

Observons qu'en complexité classique, c'est-à-dire sur les structures finies, ces deux notions de non-déterminisme coïncident et donnent lieu à la même hiérarchie polynomiale, et à la même classe de temps polynomial alternant. De plus, dans le cas classique le temps polynomial alternant coïncide avec PSPACE, l'espace polynomial, et avec PAR, le temps parallèle polynomial.

Ce n'est pas nécessairement le cas sur les structures dont l'espace sous-jacent est infini. Par exemple, sur $\mathcal{K} = (\mathbb{R}, +, -, *, =, \leq, \{c \in \mathbb{R}\})$, nous avons les inclusions suivantes entre classes [Cucker, 1993].



où une flèche indique une inclusion, $\mathrm{EXP}_{\mathbb{R}}$ désigne le temps exponentiel, $\mathrm{PEXP}_{\mathbb{R}}$ le temps parallèle exponentiel, $\mathrm{PH}_{\mathbb{R}}$ est la hiérarchie polynomiale, et $\mathrm{PAT}_{\mathbb{R}}$ est le temps polynomial alternant. En outre, les deux inclusions $\mathrm{PAR}_{\mathbb{R}} \subset \mathrm{PAT}_{\mathbb{R}}$ and $\mathrm{PAR}_{\mathbb{R}} \subset \mathrm{EXP}_{\mathbb{R}}$ sont connues pour être strictes. Nous renvoyons à [Blum et al., 1998] pour une discussion complète.

Dans ce qui suit, nous allons présenter des caractérisations syntaxiques des principales classes de ce diagramme.

Le reste de ce chapitre se veut un catalogue de nos résultats.

5.3 Fonctions calculables

Commençons par caractériser les fonctions calculables. Comme dans le cas classique, les fonctions calculables sur une structure arbitraire $\mathcal{K}$ peuvent êtres caractérisées algébriquement, en termes du plus petit ensemble de fonctions qui contient certaines fonctions initiales, et qui est clos par composition, récursion primitive, et minimisation. Dans la suite de cette section, nous présentons une telle caractérisation.

5.3.1 Fonctions partielles récursives et primitive récursives

Nous considérons des fonctions $(\mathbb{K}^*)^n \to \mathbb{K}^*$, qui prennent en entrée des mots d'éléments de $\mathbb{K}$, et retournent en sortie un mot d'éléments de $\mathbb{K}$. Lorsque la sortie d'une fonction n'est pas définie, nous utilisons le symbole $\perp$.

Définition 16 *Nous appelons* fonctions de base *les quatre types suivants de fonctions :*

(i) *Les fonctions qui réalisent des manipulations élémentaires sur les mots sur* $\mathbb{K}$. *Pour tout* $a \in \mathbb{K}, \overline{x}, \overline{x_1}, \overline{x_2} \in \mathbb{K}^*$

$$\begin{array}{rclrclrcl} \mathsf{hd}(a.\overline{x}) & = & a & \mathsf{tl}(a.\overline{x}) & = & \overline{x} & \mathsf{cons}(a.\overline{x_1}, \overline{x_2}) & = & a.\overline{x_2} \\ \mathsf{hd}(\epsilon) & = & \epsilon & \mathsf{tl}(\epsilon) & = & \epsilon & \mathsf{cons}(\epsilon, \overline{x_2}) & = & \overline{x_2}. \end{array}$$

(ii) *Les projections. Pour tout* $n \in \mathbb{N}$, $i \leq n$

$$\mathsf{Pr}_i^n(\overline{x_1}, \ldots, \overline{x_i}, \ldots, \overline{x_n}) = \overline{x_i}.$$

(iii) *Les fonctions de la structure. Pour tout opérateur (en incluant les constantes considérées comme des opérateurs d'arité 0)* op_i *ou toute relation* rel_i *d'arité* n_i *nous avons les fonctions initiales suivantes :*

$$\begin{array}{rcl} \mathsf{Op}_i(a_1.\overline{x_1}, \ldots, a_{n_i}.\overline{x_{n_i}}) & = & (op_i(a_1, \ldots, a_{n_i})).\overline{x_{n_i}} \\ \mathsf{Rel}_i(a_1.\overline{x_1}, \ldots, a_{n_i}.\overline{x_{n_i}}) & = & \left\{ \begin{array}{l} \mathbf{1} \ si \ rel_i(a_1, \ldots, a_{n_i}) \\ \epsilon \ sinon. \end{array} \right. \end{array}$$

(iv) *La fonction de sélection.*

$$\mathsf{Selection}(\overline{x}, \overline{y}, \overline{z}) = \left\{ \begin{array}{ll} \overline{y} & si \ \mathsf{hd}(\overline{x}) = \mathbf{1} \\ \overline{z} & sinon. \end{array} \right.$$

L'ensemble des fonctions récursives partielles *sur* $\mathcal{K}$ *est le plus petit ensemble de fonctions* $f : (\mathbb{K}^*)^k \to \mathbb{K}^*$ *qui contient les fonctions de base et qui est clos par les opérations suivantes :*

(1) Composition. *Supposons que* $g : (\mathbb{K}^*)^n \to \mathbb{K}^*$, $h_1, \ldots, h_n : \mathbb{K}^* \to \mathbb{K}^*$ *soient des fonctions partielles. Alors la composition* $f : \mathbb{K}^* \to \mathbb{K}^*$ *est définie par*

$$f(\overline{x}) = g(h_1(\overline{x}), \ldots, h_n(\overline{x})).$$

(2) Récursion primitive. *Supposons que* $h : \mathbb{K}^* \to \mathbb{K}^*$ *et* $g : (\mathbb{K}^*)^3 \to \mathbb{K}^*$ *soient des fonctions partielles. Alors nous définissons* $f : (\mathbb{K}^*)^2 \to \mathbb{K}^*$

$$\begin{array}{rcl} f(\epsilon, \overline{x}) & = & h(\overline{x}) \\ f(a.\overline{y}, \overline{x}) & = & \left\{ \begin{array}{ll} g(\overline{y}, f(\overline{y}, \overline{x}), \overline{x}) & \text{si } f(\overline{y}, \overline{x}) \neq \perp \\ \perp & \text{sinon.} \end{array} \right. \end{array}$$

(3) Minimisation. *Supposons que* $g : (\mathbb{K}^*)^2 \to \mathbb{K}^*$ *soit donnée. La fonction* $f : \mathbb{K}^* \to \mathbb{K}^*$ *est définie par minimisation sur le premier argument de* g, *noté par* $f(\overline{y}) = \mu\overline{x}\,(g(\overline{x}, \overline{y}))$, *si :*

$$\mu\overline{x}\,(g(\overline{x}, \overline{y})) = \left\{ \begin{array}{ll} \perp & \text{si } \forall t \in \mathbb{N} \ : \mathsf{hd}(g(0^t, \overline{y})) \neq \mathbf{1} \\ \mathbf{1}^k : \ k = \min\{t \mid \mathsf{hd}(g(0^t, \overline{y})) = \mathbf{1}\} & \text{sinon.} \end{array} \right.$$

Les fonctions partielles récursives définies sans utiliser l'opération de minimisation sont dites primitive récursives.

Faisons quelques remarques à propos de ces schémas.

(i) La définition formelle de la fonction tl est en fait une définition primitive récursive sans argument de récurrence. Cependant, lorsque nous introduirons la récursion sûre, cette fonction tl aura besoin d'être donnée comme une fonction initiale pour pouvoir être appliquée à des arguments sûrs, et pas seulement à des arguments normaux. Pour des raisons de cohérence, nous la donnons ici aussi comme une fonction initiale.

(ii) Observons que les fonctions primitives récursives sont totales, alors que les fonctions partielles récursives peuvent être des fonctions partielles.

(iii) L'opération de minimisation sur le premier argument de g retourne le plus petit mot dans $\{\mathbf{1}\}^*$ qui satisfait une propriété donnée. La raison pour laquelle il ne retourne pas le plus petit mot constitué de *n'importe* quelle lettre dans $\mathbb{K}$ est pour assurer le déterminisme, et donc la calculabilité. Sur une structure sur laquelle NP n'est pas décidable, une telle minimisation non-déterministe pourrait ne pas être calculable par une machine BSS, qui est par essence déterministe.

(iv) Dans la définition de la composition, récursion primitive et de la minimisation ci-dessus, nous avons pris des arguments $\overline{x}, \overline{y} \in \mathbb{K}^*$. Cela est pour simplifier les notations. Pour être complètement formel, nous devrions autoriser des arguments dans $(\mathbb{K}^*)^p$ avec $p \geq 1$. Nous adopterons ces simplifications dans toute la suite du chapitre pour ne pas trop alourdir.

(v) Dans la définition de la récursion primitive, la variable a devant l'argument de récurrence $a.\overline{y}$ n'apparaît pas comme argument de la fonction g. La première raison pour cela est la nécessité de la consistance dans les types d'arguments : a est un élément unique dans $\mathbb{K}$ alors que tous les arguments doivent être des mots dans $\mathbb{K}^*$. La deuxième raison est que g peut toujours dépendre de la valeur du premier élément de $\overline{y}$.

(vi) Notre définition de récursion primitive et de minimisation est légèrement différente de la définition utilisée par [Blum et al., 1989]. Dans cet article, les auteurs introduisent un argument entier spécial pour chaque fonction, qui est utilisé pour contrôler les définitions par récursion et les minimisations, et considèrent que les autres arguments sont de simples éléments dans $\mathbb{K}$. Leurs fonctions sont de type $f : \mathbb{N} \times \mathbb{K}^k \to \mathbb{K}^l$. Par conséquent, ils capturent des fonctions en dimension finie. Il est connu que sur les nombres réels avec les opérateurs $+, -, *$ les fonctions en dimensions finies sont équivalentes aux fonctions en dimensions non finies [Michaux, 1989]. Mais cela n'est pas vrai sur d'autres structures, par exemple $\mathbb{Z}/2\mathbb{Z}$. Notre choix est de considérer les arguments comme des mots d'éléments de $\mathbb{K}$, et d'utiliser la longueur de ces arguments pour contrôler les récursions et minimisations. Cela nous permet de capturer les fonctions en dimension quelconque, finie ou non, sur les structures arbitraires.

Observons que sur la structure $\{\{0,1\}, =, \mathbf{0}, \mathbf{1}\}$, nos fonctions partielles récursives (respectivement primitives récursives) coïncident avec les fonctions partielles récursives (respectivement primitives récursives) classiques.

5.3.2 Caractérisation des fonctions calculables

Nous avons prouvé le résultat suivant dans les articles [Bournez et al., 2002], [Bournez et al., 2003], repris dans le journal [Bournez et al., 2005a].

Théorème 6 *Sur toute structure*

$$\mathcal{K} = (\mathbb{K}, op_1, \cdots, op_k, rel_1, \cdots, rel_l, \{c_i\}_{i \in I}, \mathbf{0}, \mathbf{1}),$$

une fonction est calculable si et seulement si elle peut être définie comme une fonction partielle récursive sur $\mathcal{K}$.

5.4 Temps polynomial

Nous allons maintenant présenter une caractérisation du temps polynomial. Pour cela, nous allons étendre à une structure arbitraire la notion de récursion sûre sur les nombres entiers définie par Bellantoni et Cook dans l'article [Bellantoni and Cook, 1992a].

5.4.1 Fonctions récursives sûres

Exemple 4 *Considérons la fonction suivante*

$$\exp(\overline{x}) = \mathbf{1}^{2^{|\overline{x}|}}$$

qui calcule en unaire l'exponentielle. Elle peut être définie facilement par récursion primitive par

$$\begin{array}{rcl} \mathsf{Cons}(\epsilon, \overline{y}) & = & \overline{y} \\ \mathsf{Cons}(a.\overline{x}, \overline{y}) & = & \mathsf{cons}(a, \mathsf{Cons}(\overline{x}, \overline{y})) \\ \exp(\epsilon) & = & \mathbf{1} \\ \exp(a.\overline{x}) & = & \mathsf{Cons}(\exp(\overline{x}), \exp(\overline{x})). \end{array}$$

D'un autre coté, observons que $\exp \notin \mathrm{FP}_{\mathcal{K}}$ *puisque la valeur calculée est exponentiellement large en son argument. Le but de cette section est d'introduire une version restreinte de la récursion, inspirée par Bellantoni et Cook [Bellantoni and Cook, 1992a], qui ne permet pas cette croissance exponentielle.*

Les fonctions récursives sûres sont définies de la même manière que les fonctions primitives récursives. En suivant les idées de [Bellantoni and Cook, 1992a], les fonctions récursives sûres ont deux types d'arguments, chacun d'entre eux ayant des propriétés et des fonctions différentes.

Le premier type d'argument, dit *normal*, est similaire aux arguments des fonctions partielles récursives et primitives récursives précédentes, puisqu'il peut être utilisé pour faire des étapes de calcul pour contrôler la récursion.

Le second type d'arguments, dit *sûr*, ne peut pas être utilisé pour contrôler les récursions. Dans une récursion, l'argument de récurrence peut être seulement de type sûr.

Nous verrons que cette distinction entre arguments sûrs et normaux garantie que les fonctions récursives sûres peuvent être calculées en temps polynomial.

Pour insister sur la distinction entre arguments normaux et sûrs, nous écrirons $f : N \times S \to R$ où N indique le domaine des arguments normaux, S celui des arguments sûrs, et R le codomaine de f. Si tous les arguments de f sont d'un type, disons sûr, nous écrirons $\emptyset$ à la place de N. Aussi, si $\overline{x}$ et $\overline{y}$ sont ces arguments, nous écrirons $f(\overline{x}; \overline{y})$ en les séparant par un " ;". Les arguments normaux sont placés à la gauche du point-virgule, et les arguments sûrs à sa droite.

Définition 17 *Nous appelons* fonctions de base *les quatre types suivant de fonctions :*

(i) *Les fonctions qui réalisent des manipulations élémentaires sur les mots sur* $\mathbb{K}$. *Pour tout* $a \in \mathbb{K}, \overline{x}, \overline{x_1}, \overline{x_2} \in \mathbb{K}^*$

$$\begin{array}{rclcrclcrcl} \mathsf{hd}(;a.\overline{x}) & = & a & & \mathsf{tl}(;a.\overline{x}) & = & \overline{x} & & \mathsf{cons}(;a.\overline{x_1}, \overline{x_2}) & = & a.\overline{x_2} \\ \mathsf{hd}(;\epsilon) & = & \epsilon & & \mathsf{tl}(;\epsilon) & = & \epsilon & & \mathsf{cons}(;\epsilon, \overline{x_2}) & = & \overline{x_2}. \end{array}$$

(ii) *Les projections. Pour tout* $n \in \mathbb{N}$, $i \leq n$,

$$\mathsf{Pr}^n_i(;\overline{x_1}, \ldots, \overline{x_i}, \ldots, \overline{x_n}) = \overline{x_i}.$$

(iii) *Les fonctions de la structure. Pour tout opérateur (en incluant les constantes considérées comme des opérateurs d'arité 0)* op_i *ou toute relation* rel_i *d'arité* n_i *nous avons les fonctions initiales suivantes :*

$$\begin{array}{rcl} \mathsf{Op}_i(;a_1.\overline{x_1}, \ldots, a_{n_i}.\overline{x_{n_i}}) & = & (op_i(a_1, \ldots, a_{n_i})).\overline{x_{n_i}} \\ \mathsf{Rel}_i(;a_1.\overline{x_1}, \ldots, a_{n_i}.\overline{x_{n_i}}) & = & \begin{cases} \mathbf{1} \text{ si } rel_i(a_1, \ldots, a_{n_i}) \\ \mathbf{0} \text{ sinon.} \end{cases} \end{array}$$

La relation d'égalité sera notée Equal.

(iv) *Une fonction de sélection.*

$$\mathsf{Select}(;\overline{x},\overline{y},\overline{z}) = \begin{cases} \overline{y} & \textit{si } \mathsf{hd}(\overline{x}) = \mathbf{1} \\ \overline{z} & \textit{sinon.} \end{cases}$$

Définition 18 *L'ensemble des* fonctions récursives sûres *sur $\mathcal{K}$, noté par $\mathrm{SR}_{\mathcal{K}}$, est le plus petit ensemble de fonctions $f : (\mathbb{K}^*)^p \times (\mathbb{K}^*)^q \to \mathbb{K}^*$ qui contient les fonctions de base et qui est clos par les opérations suivantes :*

(1) Composition sûre. *Soient $g : (\mathbb{K}^*)^m \times (\mathbb{K}^*)^n \to \mathbb{K}^*$, $h_1, \ldots, h_m : \mathbb{K}^* \times \emptyset \to \mathbb{K}^*$ et $h_{m+1}, \ldots, h_{m+n} : \mathbb{K}^* \times \mathbb{K}^* \to \mathbb{K}^*$ des fonctions récursives sûres. Leur composition sûre est la fonction $f : \mathbb{K}^* \times \mathbb{K}^* \to \mathbb{K}^*$ définie par*

$$f(\overline{x};\overline{y}) = g\left(h_1(\overline{x};),\ldots,h_m(\overline{x};);h_{m+1}(\overline{x};\overline{y}),\ldots,h_{m+n}(\overline{x};\overline{y})\right).$$

(2) Récursion sûre. *Soient $h : \mathbb{K}^* \times \mathbb{K}^* \to \mathbb{K}^*$ et $g : (\mathbb{K}^*)^2 \times (\mathbb{K}^*)^2 \to \mathbb{K}^*$. Nous définissons $f : (\mathbb{K}^*)^2 \times \mathbb{K}^* \to \mathbb{K}^*$ par récursion sûre comme suit*

$$\begin{aligned} f(\epsilon,\overline{x};\overline{y}) &= h(\overline{x};\overline{y}) \\ f(a.\overline{z},\overline{x};\overline{y}) &= g(\overline{z},\overline{x};f(\overline{z},\overline{x};\overline{y}),\overline{y}). \end{aligned}$$

Faisons quelques remarques à propos de ces définitions.

(i) En utilisant la composition sûre, il est possible de *déplacer* un argument de la position normale en la position sûre, alors que le contraire est interdit. Par exemple, supposons que $g : \mathbb{K}^* \times (\mathbb{K}^*)^2 \to \mathbb{K}^*$ soit une fonction donnée. On peut alors définir avec une composition sûre une fonction f donnée par $f(\overline{x},\overline{y};\overline{z}) = g(\overline{x};\overline{y},\overline{z})$ mais une définition comme $f(\overline{x};\overline{y},\overline{z}) = g(\overline{x},\overline{y};\overline{z})$ n'est pas valide.

(ii) Observons qu'il est impossible de transformer la définition de exp dans l'exemple 4 en un schéma de récursion sûr. $\mathsf{Cons}(x;y)$ et $\mathsf{Cons}(x,y;)$ peuvent être définis comme suit :

$$\begin{aligned} \mathsf{Cons}(\epsilon;\overline{y}) &= \overline{y} \\ \mathsf{Cons}(a.\overline{x};\overline{y}) &= \mathsf{cons}(;a,\mathsf{Cons}(\overline{x};\overline{y})) \\ \mathsf{Cons}(\epsilon,\overline{y};) &= \overline{y} \\ \mathsf{Cons}(a.\overline{x},\overline{y};) &= \mathsf{cons}(;a,\mathsf{Cons}(\overline{x},\overline{y};)). \end{aligned}$$

Cependant, exp ne peut être définie, puisque cela nécessiterait l'utilisation de l'argument de récurrence $\exp(\overline{x})$ comme un argument normal de la fonction Cons, ce qui est interdit. Cette obstruction est à la base de l'équivalence entre les fonctions récursives sûres et les calculs en temps polynomial.

5.4.2 Caractérisation du temps polynomial

Nous avons prouvé le résultat suivant dans colloques [Bournez et al., 2002], [Bournez et al., 2003], et dans le journal [Bournez et al., 2005a], qui étend [Bellantoni and Cook, 1992a] au cas des structures arbitraires.

Théorème 7 *Sur toute structure*

$$\mathcal{K} = (\mathbb{K}, \{op_i\}_{i\in I}, rel_1, \ldots, rel_l, \mathbf{0}, \mathbf{1}),$$

une fonction est calculée en temps polynomial si et seulement si elle peut être définie comme une fonction récursive sûre sur $\mathcal{K}$.

5.5 Temps parallèle polynomial

5.5.1 Définitions

On pourra trouver dans [Blum et al., 1998] la définition de machines parallèles sur une structure $\mathcal{K}$. Nous ne donnerons pas les définitions formelles ici, en renvoyant[4] à [Blum et al., 1998].

Définition 19 *$FPAR_{\mathcal{K}}$ est la classe des fonctions f calculables en temps polynomial par une machine parallèle qui utilise un nombre exponentiellement borné de processeurs et telle que $|f(\overline{x})| = |\overline{x}|^{\mathcal{O}(1)}$ pour tout $\overline{x} \in \mathbb{K}^*$.*

Nous renvoyons à [Bournez et al., 2005a] pour une définition de cette classe en termes de circuits, sans utiliser la notion de machine parallèle.

5.5.2 Fonctions récursives sûres avec substitutions

Définition 20 *L'ensemble des fonctions définies par* récursion sûre avec substitutions *sur $\mathcal{K}$ est le plus petit ensemble de fonctions $f : (\mathbb{K}^*)^p \times (\mathbb{K}^*)^q \to \mathbb{K}^*$, qui contient les fonctions sûres de base et qui est clos par composition sûre et par l'opération suivante :*

Récursion sûre avec substitutions. *Soient $h : \mathbb{K}^* \times (\mathbb{K}^*)^2 \to \mathbb{K}^*$, $g : (\mathbb{K}^*)^2 \times (\mathbb{K}^*)^{l+1} \to \mathbb{K}^*$, et $\sigma_j : \emptyset \times \mathbb{K}^* \to \mathbb{K}^*$ pour $0 < j \leq l$ des fonctions récursives sûres.*
La fonction $f : (\mathbb{K})^2 \times (\mathbb{K}^*)^2 \to \mathbb{K}^*$ est définie par récursion sûre avec substitutions comme suit :

$$\begin{aligned} f(\epsilon, \overline{x}; \overline{u}, \overline{y}), &= h(\overline{x}; \overline{u}, \overline{y}) \\ f(a.\overline{z}, \overline{x}; \overline{u}, \overline{y}) &= \begin{cases} g\left(\overline{z}, \overline{x}; f(\overline{z}, \overline{x}; \sigma_1(;\overline{u}), \overline{y}), \ldots, f(\overline{z}, \overline{x}; \sigma_l(;\overline{u}), \overline{y}), \overline{y}\right) \\ \qquad\qquad \text{si } \forall j \ f(\overline{z}, \overline{x}; \sigma_j(;\overline{u}), \overline{y}) \neq \perp \\ \perp \qquad\qquad \text{sinon.} \end{cases} \end{aligned}$$

Les fonctions σ_j sont appelées des fonctions de substitutions.

5.5.3 Caractérisation du temps parallèle polynomial

Nous avons prouvé le résultat suivant dans les articles [Bournez et al., 2003], [Bournez et al., 2005a]. Ce résultat étend [Leivant and Marion, 1995] au cas d'une structure arbitraire.

Théorème 8 *Sur toute structure*

$$\mathcal{K} = (\mathbb{K}, op_1, \cdots, op_k, rel_1, \cdots, rel_l, \{c_i\}_{i \in I}, \mathbf{0}, \mathbf{1}),$$

une fonction est dans $FPAR_{\mathcal{K}}$ si et seulement si elle peut être définie comme une fonction récursive sûre avec substitutions sur $\mathcal{K}$.

Observons que dans le cas classique, voir [Leivant and Marion, 1995], la récursion sûre avec substitutions caractérise la classe FPSPACE. Cependant, sur une structure arbitraire, dans le cas général, la notion d'espace n'a pas de sens, comme démontré par [Michaux, 1989] : sur des structures comme la structure $(\mathbb{R}, 0, 1, \leq, +, -, *,)$, tout calcul peut être fait en espace constant. Cependant, dans le cas classique, nous avons FPAR= FPSPACE.

[4] La présentation du parallélisme dans [Blum et al., 1998, Chapter 18] est pour $\mathcal{K}$ les nombres réels, mais leur définition de machine parallèle peut s'étendre assez facilement aux structures arbitraires.

5.6 Hiérarchie polynomiale

5.6.1 Définitions

Comme dans le cas classique, la hiérarchie polynomiale sur une structure $\mathcal{K}$ peut se définir de plusieurs façons, par des définitions syntaxiques, ou par des définitions sémantiques par relativisations successives du temps polynomial non-déterministe : voir [Blum et al., 1998].

Rappelons quelques classes de complexité de base : $\mathrm{P}_{\mathcal{K}}$ est la classe des problèmes sur $\mathcal{K}$ décidés en temps polynomial. $\mathrm{FP}_{\mathcal{K}}$ est la classe des fonctions sur $\mathcal{K}$ calculables en temps polynomial.

Un problème de décision A est dans $\mathrm{NP}_{\mathcal{K}}$ si et seulement s'il existe un problème de décision B dans $\mathrm{P}_{\mathcal{K}}$, et un polynôme p_B, tels que $\overline{x} \in A$ si et seulement s'il existe $\overline{y} \in \mathbb{K}^*$ avec $|\overline{y}| \leq p_B(|\overline{x}|)$ qui satisfait $(\overline{x}, \overline{y}) \in B$.

Un problème de décision A est dans $\mathrm{coNP}_{\mathcal{K}}$ si et seulement s'il existe un problème de décision B dans $\mathrm{P}_{\mathcal{K}}$, et un polynôme p_B tels que $\overline{x} \in A$ si et seulement si pour tout $\overline{y} \in \mathbb{K}^*$ avec $|\overline{y}| \leq P_B(|\overline{x}|)$, $(\overline{x}, \overline{y})$ est dans B.

Définition 21 *Soit* $\Sigma^0_{\mathcal{K}} = \mathrm{P}_{\mathcal{K}}$ *et, pour* $i \geq 1$,

$$\Sigma^i_{\mathcal{K}} = \mathrm{NP}^{\Sigma^{i-1}_{\mathcal{K}}}_{\mathcal{K}},$$

$$\Pi^i_{\mathcal{K}} = \mathrm{coNP}^{\Sigma^{i-1}_{\mathcal{K}}}_{\mathcal{K}}.$$

La hiérarchie polynomiale *sur* $\mathcal{K}$ *est*

$$\mathrm{PH}_{\mathcal{K}} = \bigcup_{i=0}^{\infty} \Sigma^i_{\mathcal{K}} = \bigcup_{i=0}^{\infty} \Pi^i_{\mathcal{K}}.$$

Une fonction est dans $\mathrm{F}\Delta^i_{\mathcal{K}}$ *si elle est calculable en temps polynomial par une machine sur* $\mathcal{K}$ *qui fait des requêtes à un oracle dans* $\Sigma^i_{\mathcal{K}}$. *C'est-à-dire*

$$\mathrm{F}\Delta^i_{\mathcal{K}} = \mathrm{FP}^{\Sigma^i_{\mathcal{K}}}_{\mathcal{K}} = \mathrm{FP}^{\Pi^i_{\mathcal{K}}}_{\mathcal{K}}.$$

La hiérarchie polynomiale fonctionnelle *sur* $\mathcal{K}$ *est*

$$\mathrm{FPH}_{\mathcal{K}} = \bigcup_{i=0}^{\infty} \mathrm{F}\Delta^i_{\mathcal{K}}.$$

Observons que des problèmes complets sont connus pour chacune des classes $\Sigma^i_{\mathcal{K}}$ et $\Pi^i_{\mathcal{K}}$ [Blum et al., 1998, Poizat, 1995].

5.6.2 Fonctions récursives sûres avec minimisations prédicatives

Dans l'esprit de [Bellantoni, 1994], nous introduisons maintenant la notion de minimisation prédicative.

Définition 22 *Étant donnée* $h : \mathbb{K}^* \times (\mathbb{K}^*)^2 \to \mathbb{K}^*$, *nous définissons* $f : \mathbb{K}^* \times \mathbb{K}^* \to \mathbb{K}$ *par* minimisation prédicative *comme suit*

$$f(\overline{x}; \overline{a}) = \boldsymbol{\Im}\overline{b}(h(\overline{x}; \overline{a}, \overline{b})) = \begin{cases} \mathbf{1} & \textit{s'il existe } \overline{b} \in \mathbb{K}^* \textit{ tel que } h(\overline{x}; \overline{a}, \overline{b}) = \mathbf{0} \\ \mathbf{0} & \textit{sinon.} \end{cases}$$

Nous introduisons maintenant de nouveaux ensembles de fonctions.

Définition 23 *Soit F une classe de fonctions. L'ensemble des* fonctions récursives sûres restreintes relativement à F *sur $\mathcal{K}$, noté par $RSR_{\mathcal{K}}(F)$, est le plus petit ensemble de fonctions qui contient les fonctions sûres de base et F, et qui est clos par l'opération de* composition sûre restreinte *suivante :*

$$f(\overline{x};\overline{y}) = g\left(h_1(\overline{x};),\ldots,h_m(\overline{x};);h_{m+1}(\overline{x};\overline{y}),\ldots,h_{m+n}(\overline{x};\overline{y})\right).$$

où les h_i appartiennent à $RSR_{\mathcal{K}}(F)$ et g à $\mathrm{SR}_{\mathcal{K}}$, et le schéma de récursion sûr restreint

$$\begin{aligned} f(\epsilon,\overline{x};\overline{y}) &= h(\overline{x};\overline{y}) \\ f(a.\overline{z},\overline{x};\overline{y}) &= g(\overline{z},\overline{x};f(\overline{z},\overline{x};\overline{y}),\overline{y}) \end{aligned}$$

où h appartient à $RSR_{\mathcal{K}}(F)$ et g à $\mathrm{SR}_{\mathcal{K}}$. Cela implique qu'aucune fonction dans $F\backslash\mathrm{SR}_{\mathcal{K}}$ ne peut être impliquée dans la définition de g.

Définition 24 *Supposons que F soit une classe de fonctions : une fonction f est dans $\exists F$ si elle est définie avec une minimisation prédicative sur une fonction h de F.*

Nous définissons par induction les ensembles suivants :

- $F^0_{\mathcal{K}} = \mathrm{SR}_{\mathcal{K}}$.
- $$F^{i+1}_{\mathcal{K}} = RSR_{\mathcal{K}}(F^i_{\mathcal{K}} \bigcup \exists F^i_{\mathcal{K}}),$$

pour $i \geq 0$.

Nous dénotons par

$$\exists\mathrm{PH}_{\mathcal{K}} = \bigcup_{i\in\mathbb{N}} F^i_{\mathcal{K}}$$

la clôture des fonctions sûres de base sur $\mathcal{K}$ par applications de récursions sûres restreintes, minimisations prédicatives, et compositions sûres.

5.6.3 Caractérisation de la hiérarchie polynomiale

Nous avons prouvé le résultat suivant dans les articles [Bournez et al., 2003], [Bournez et al., 2005a].

Théorème 9 *Sur toute structure*

$$\mathcal{K} = (\mathbb{K}, op_1, \cdots, op_k, rel_1, \cdots, rel_l, \{c_i\}_{i\in I}, \mathbf{0}, \mathbf{1}),$$

une fonction $f : (\mathbb{K}^*)^n \times \emptyset \to \mathbb{K}^*$ appartient à $\mathrm{F}\Delta^i_{\mathcal{K}}$ si et seulement si elle peut être définie comme une fonction dans $\mathrm{F}^i_{\mathcal{K}}$.

Corollaire 3 *Un problème de décision sur $\mathcal{K}$ appartient à $\mathrm{PH}_{\mathcal{K}}$ si et seulement si sa fonction caractéristique peut être définie dans $\exists\mathrm{PH}_{\mathcal{K}}$.*

5.7 Hiérarchie polynomiale digitale

5.7.1 Définitions

Dans la version digitale de la hiérarchie polynomiale, les témoins sont donnés par des choix discrets, et non par des éléments de la structure. Comme pour la hiérarchie polynomiale précédente, des problèmes complets pour chacun de ses niveaux sont connus [Blum et al., 1998].

Définition 25 *Un ensemble $A \subseteq \mathbb{K}^*$ appartient à $\mathrm{DNP}_{\mathcal{K}}$ si et seulement s'il existe un problème de décision B dans $\mathrm{P}_{\mathcal{K}}$, et un polynôme p_B, tels que pour tout $\overline{x} \in \mathbb{K}^*$,*

$$\overline{x} \in A \Leftrightarrow \exists \overline{y} \in \{\mathbf{0},\mathbf{1}\}^* \text{ tel que } |\overline{y}| \leq p_B(|\overline{x}|) \text{ et } (\overline{x},\overline{y}) \in B.$$

Soit

$$\mathrm{D}\Sigma^0_{\mathcal{K}} = \mathrm{P}_{\mathcal{K}}$$

et, pour $i \geq 1$,

$$\mathrm{D}\Sigma^i_{\mathcal{K}} = \mathrm{DNP}^{\mathrm{D}\Sigma^{i-1}_{\mathcal{K}}}_{\mathcal{K}},$$

$$\mathrm{D}\Pi^i_{\mathcal{K}} = \mathrm{coDNP}^{\mathrm{D}\Sigma^{i-1}_{\mathcal{K}}}_{\mathcal{K}}.$$

La hiérarchie polynomiale digitale *est*

$$\mathrm{DPH}_{\mathcal{K}} = \bigcup_{i=0}^{\infty} \mathrm{D}\Sigma^i_{\mathcal{K}} = \bigcup_{i=0}^{\infty} \mathrm{D}\Pi^i_{\mathcal{K}}.$$

Une fonction est dans $\mathrm{DF}\Delta^i_{\mathcal{K}}$ si elle est calculable en temps polynomial par une machine sur $\mathcal{K}$ qui fait des requêtes à un oracle dans $\mathrm{D}\Sigma^i_{\mathcal{K}}$. C'est-à-dire

$$\mathrm{DF}\Delta^i_{\mathcal{K}} = \mathrm{FP}^{\mathrm{D}\Sigma^i_{\mathcal{K}}}_{\mathcal{K}} = \mathrm{FP}^{\mathrm{D}\Pi^i_{\mathcal{K}}}_{\mathcal{K}}.$$

La hiérarchie polynomiale digitale fonctionnelle *est*

$$\mathrm{DFPH}_{\mathcal{K}} = \bigcup_{i=0}^{\infty} \mathrm{DF}\Delta^i_{\mathcal{K}}.$$

5.7.2 Fonctions récursives sûres avec minimisations prédicatives digitales

De façon similaire à la notion de minimisation prédicative dans la section précédente, nous introduisons la notion de minimisation prédicative digitale.

Définition 26 *Étant donnée $h : \mathbb{K}^* \times (\mathbb{K}^*)^2 \to \mathbb{K}^*$, nous définissons $f : \mathbb{K}^* \times \mathbb{K}^* \to \mathbb{K}$ par* minimisation prédicative digitale *comme suit*

$$f(\overline{x};\overline{a}) = \mathfrak{Z}_{\mathrm{D}}\overline{b}(h(\overline{x};\overline{a},\overline{b})) = \begin{cases} \mathbf{1} & \text{s'il existe } \overline{b} \in \{\mathbf{0},\mathbf{1}\}^* \text{ tel que } h(\overline{x};\overline{a},\overline{b}) = \mathbf{0} \\ \mathbf{0} & \text{sinon.} \end{cases}$$

Définition 27 *Soit F une classe de fonctions. Une fonction f est dans $\mathfrak{Z}_{\mathrm{D}}F$ si elle est définie avec une minimisation prédicative digitale sur une fonction h de F.*

Nous définissons par récurrence les ensembles suivants :

- $dF^0_{\mathcal{K}} = \mathrm{SR}_{\mathcal{K}}$
- $$dF^{i+1}_{\mathcal{K}} = RSR_{\mathcal{K}}(dF^i_{\mathcal{K}} \bigcup \mathfrak{Z}_{\mathrm{D}}F^i_{\mathcal{K}}),$$

  pour $i \geq 0$.

Nous dénotons par $\mathfrak{Z}_{\mathrm{D}}\mathrm{PH}_{\mathcal{K}}$ la clôture des fonctions sûres de base sur $\mathcal{K}$ par applications de récursions sûres restreintes, minimisations prédicatives digitales, et compositions sûres.

5.7.3 Caractérisation de la hiérarchie polynomiale digitale

Nous avons prouvé le résultat suivant dans les articles [Bournez et al., 2003], [Bournez et al., 2005a].

Théorème 10 *Sur toute structure*

$$\mathcal{K} = (\mathbb{K}, op_1, \cdots, op_k, rel_1, \cdots, rel_l, \{c_i\}_{i\in I}, \mathbf{0}, \mathbf{1}),$$

une fonction $f : (\mathbb{K}^*)^n \times \emptyset \to \mathbb{K}^*$ *appartient à* $\mathrm{DF}\Delta^i_{\mathcal{K}}$ *si et seulement si elle peut être définie comme une fonction de* $dF^i_{\mathcal{K}}$.

Corollaire 4 *Un problème de décision sur* $\mathcal{K}$ *appartient à la hiérarchie polynomiale digitale* $\mathrm{DPH}_{\mathcal{K}}$ *si et seulement si sa fonction caractéristique peut être définie dans* $\exists_{\mathrm{D}}\mathrm{DPH}_{\mathcal{K}}$.

Quand on considère des structures finies, cela donne une caractérisation de la hiérarchie polynomiale classique alternative à celle de [Bellantoni, 1994].

Corollaire 5 *Un problème de décision appartient à* PH *si et seulement si sa fonction caractéristique peut être définie dans* $\exists_{\mathrm{D}}\mathrm{DPH}_{\{\mathbf{0},\mathbf{1}\}}$.

5.8 Temps polynomial alternant

5.8.1 Définitions

Définition 28 *Un ensemble* $S \subseteq \mathbb{K}^*$ *appartient à* $\mathrm{PAT}_{\mathcal{K}}$, le temps polynomial alternant, *si et seulement s'il existe un polynôme* $q : \mathbb{N} \to \mathbb{N}$ *et une machine* M_S *sur* $\mathcal{K}$ *qui s'arrête en temps polynomial, tels que, pour tout* $\overline{x} \in \mathbb{K}^*$,

$$\begin{aligned}\overline{x} \in S \quad \Leftrightarrow \quad & \exists a_1 \in \mathbb{K}\, \forall b_1 \in \mathbb{K}\ \ldots\ \exists a_{q(|\overline{x}|)} \in \mathbb{K}\, \forall b_{q(|\overline{x}|)} \in \mathbb{K} \\ & M_S \text{ accepte } (\overline{x}, a_1.b_1 \ldots a_{q(|\overline{x}|)}.b_{q(|\overline{x}|)}).\end{aligned}$$

En outre, nous définissons

$$\mathrm{FPAT}_{\mathcal{K}} = \mathrm{FP}^{\mathrm{PAT}_{\mathcal{K}}}_{\mathcal{K}}.$$

Lorsque $\mathcal{K}$ est la structure $\{\{0,1\}, =, \mathbf{0}, \mathbf{1}\}$, $\mathrm{PAT}_{\mathcal{K}}$ est PSPACE.

Il est important de comprendre que le nombre d'alternances de quantificateurs n'est pas fixé, mais dépend de la longueur de l'entrée, et est polynomial en cette longueur.

On a par conséquent toujours $\mathrm{PH}_{\mathcal{K}} \subseteq \mathrm{PAT}_{\mathcal{K}}$.

5.8.2 Fonctions récursives sûres avec substitutions prédicatives

Définition 29 *Étant donnée* $h : \mathbb{K}^* \times (\mathbb{K}^*)^2 \to \mathbb{K}^*$, *nous définissons* $f : \mathbb{K}^* \times \mathbb{K}^* \to \mathbb{K}$ *par* substitution prédicative *comme suit*

$$f(\overline{x}; \overline{a}) = \exists^{[1]} c(h(\overline{x}; \overline{a}, c)) = \begin{cases} \mathbf{1} & \text{s'il existe } c \in \mathbb{K} \text{ tel que } h(\overline{x}; \overline{a}, c) = \mathbf{0} \\ \mathbf{0} & \text{sinon.} \end{cases}$$

Définition 30 *Supposons que* $h : \mathbb{K}^* \times (\mathbb{K}^*)^2 \to \mathbb{K}^*$ *et* $g : (\mathbb{K}^*)^2 \times (\mathbb{K}^*)^2 \to \mathbb{K}^*$ *soient des fonctions. La fonction* $f : (\mathbb{K}^*)^2 \times (\mathbb{K}^*)^2 \to \mathbb{K}^*$ *est définie par* récursion sûre avec substitutions prédicatives *comme suit*

$$\begin{aligned} f(\epsilon, \overline{x}; \overline{u}, \overline{y}) &= h(\overline{x}; \overline{u}, \overline{y}) \\ f(a.\overline{z}, \overline{x}; \overline{u}, \overline{y}) &= g(\overline{z}, \overline{x}; \exists^{[1]} c(f(\overline{z}, \overline{x}; c.\overline{u}, \overline{y})), \overline{y}). \end{aligned}$$

Définition 31 *L'ensemble* $\exists^{[1]}\mathrm{PAT}_{\mathcal{K}}$ *des* fonctions récursives sûres avec substitutions prédicatives *sur* $\mathcal{K}$ *est la clôture des fonctions sûres de base par applications de compositions sûres, récursions sûres, et récursions sûres avec substitutions prédicatives.*

5.8.3 Caractérisation du temps polynomial alternant

Nous avons prouvé le résultat suivant dans les articles [Bournez et al., 2004b], [Bournez et al., 2006b].

Théorème 11 *Sur toute structure*

$$\mathcal{K} = (\mathbb{K}, op_1, \cdots, op_k, rel_1, \cdots, rel_l, \{c_i\}_{i\in I}, \mathbf{0}, \mathbf{1}),$$

une fonction est dans $\mathrm{FPAT}_{\mathcal{K}}$ *si et seulement si elle peut être définie comme une fonction dans* $\exists^{[1]}\mathrm{PAT}_{\mathcal{K}}$.

5.9 Temps polynomial alternant digital

5.9.1 Définitions

La classe $\mathrm{DPAT}_{\mathcal{K}}$ est définie de façon similaire à la classe $\mathrm{PAT}_{\mathcal{K}}$ mais avec toutes les variables quantifiées appartenant à $\{\mathbf{0}, \mathbf{1}\}$. De façon similaire, nous pouvons définir

$$\mathrm{DFPAT}_{\mathcal{K}} = \mathrm{FP}_{\mathcal{K}}^{\mathrm{DPAT}_{\mathcal{K}}}.$$

5.9.2 Fonctions récursives sûres avec substitutions prédicatives digitales

De façon similaire à la notion de substitution prédicative, nous définissons la notion de substitution prédicative digitale.

Définition 32 *Étant donnée* $h : \mathbb{K}^* \times (\mathbb{K}^*)^2 \to \mathbb{K}^*$, *nous définissons* $f : \mathbb{K}^* \times \mathbb{K}^* \to \mathbb{K}$ *par* substitution prédicative,

$$f(\overline{x}; \overline{a}) = \exists^{[1]}_{\mathsf{D}} c(h(\overline{x}; \overline{a}, c)) = \begin{cases} \mathbf{1} & \text{s'il existe } c \in \{\mathbf{0}, \mathbf{1}\} \text{ tel que } h(\overline{x}; \overline{a}, c) = \mathbf{0} \\ \mathbf{0} & \text{sinon.} \end{cases}$$

Définition 33 *Supposons que* $h : \mathbb{K}^* \times (\mathbb{K}^*)^2 \to \mathbb{K}^*$ *et* $g : (\mathbb{K}^*)^2 \times (\mathbb{K}^*)^2 \to \mathbb{K}^*$ *soient des fonctions. La fonction* $f : (\mathbb{K}^*)^2 \times (\mathbb{K}^*)^2 \to \mathbb{K}^*$ *est définie par* récursion sûre avec substitutions prédicatives digitales *comme suit*

$$\begin{aligned} f(\epsilon, \overline{x}; \overline{u}, \overline{y}) &= h(\overline{x}; \overline{u}, \overline{y}) \\ f(a.\overline{z}, \overline{x}; \overline{u}, \overline{y}) &= g(\overline{z}, \overline{x}; \exists^{[1]}_{\mathsf{D}} c f(\overline{z}, \overline{x}; c.\overline{u}, \overline{y}), \overline{y}). \end{aligned}$$

Définition 34 *L'ensemble* $\exists^{[1]}_{\mathsf{D}}\mathrm{PAT}_{\mathcal{K}}$ *des* fonctions récursives sûres avec substitutions prédicatives digitales *sur* $\mathcal{K}$ *est la clôture des fonctions sûres de base par applications de compositions sûres, récursions sûres, et récursions sûres avec substitutions prédicatives digitales.*

5.9.3 Caractérisation du temps polynomial alternant digital

Nous avons prouvé le résultat suivant dans les articles [Bournez et al., 2004b], [Bournez et al., 2006b].

Théorème 12 *Sur toute structure*

$$\mathcal{K} = (\mathbb{K}, op_1, \cdots, op_k, rel_1, \cdots, rel_l, \{c_i\}_{i\in I}, \mathbf{0}, \mathbf{1}),$$

une fonction est dans $\mathrm{DFPAT}_{\mathcal{K}}$ *si et seulement si elle peut être définie comme une fonction de* $\exists^{[1]}\mathrm{DPAT}_{\mathcal{K}}$.

Lorsqu'on se restreint aux structures finies, cela donne une caractérisation alternative de la classe PSPACE.

Corollaire 6 *Un ensemble* $S \subset \{0,1\}^*$ *est dans* PSPACE *si et seulement si sa fonction caractéristique peut être définie dans* $\exists^{[1]}\mathrm{DPAT}_{\{\mathbf{0},\mathbf{1}\}}$.

5.9.4 Caractérisation alternative

Une caractérisation alternative est possible, en se basant sur la caractérisation précédente de $\mathrm{PAR}_{\mathcal{K}}$, avec de petites restrictions sur le type de fonctions impliquées dans les schéma de récursions.

Définition 35 *Nous appelons* fonction pseudo logique *toute fonction dans la clôture des opérations d'arité* 0 *(les constantes), des projections et de la fonction de sélection* Select *par applications de compositions sûres.*

Puisque aucune récursion ou fonction tl n'est impliquée dans la définition d'une fonction pseudo logique, son résultat ne dépend que de la valeur de la première lettre de ses arguments, et plus précisément en le fait que ce soient des **1** ou non.

Définition 36 *Supposons que* $h : \mathbb{K}^* \times \mathbb{K}^* \to \mathbb{K}^*$ *soit une fonction donnée,* $g : (\mathbb{K}^*)^2 \times (\mathbb{K}^*)^2 \to \mathbb{K}^*$ *une fonction pseudo logique, et* $\sigma_1, \sigma_2 : \emptyset \times \mathbb{K}^* \to \mathbb{K}^*$ *des fonctions récursives sûres. La fonction* $f : \mathbb{K}^* \times \mathbb{K}^* \to \mathbb{K}^*$ *peut être définie par* récursion sûre avec substitutions digitales *par*

$$\begin{aligned} f(\epsilon, \overline{x}; \overline{u}) &= h(\overline{x}; \overline{u}) \\ f(a.\overline{z}, \overline{x}; \overline{u}) &= g(\overline{z}, \overline{x}; f(\overline{z}, \overline{x}; \sigma_1(; \overline{u})), f(\overline{z}, \overline{x}; \sigma_2(; \overline{u}))). \end{aligned}$$

Nous définissons l'ensemble des fonctions *récursives sûres avec substitutions digitales* comme la clôture des fonctions sûres de base par applications de compositions sûres, récursions sûres et récursions sûres avec substitutions digitales.

Une caractérisation alternative est donnée par le théorème suivant présenté dans [Bournez et al., 2004b], [Bournez et al., 2006b].

Théorème 13 *Sur toute structure*

$$\mathcal{K} = (\mathbb{K}, op_1, \cdots, op_k, rel_1, \cdots, rel_l, \{c_i\}_{i \in I}, \mathbf{0}, \mathbf{1}),$$

une fonction est dans $\mathrm{DFPAT}_{\mathcal{K}}$ *si et seulement si elle peut être définie comme une fonction récursive sûre avec substitutions digitales sur* $\mathcal{K}$*.*

Lorsque $\mathcal{K}$ est une structure finie, le théorème précédent donne une caractérisation qui coïncide avec notre caractérisation de $\mathrm{PAR}_{\mathcal{K}}$, et qui capture PSPACE.

Chapitre 6

Conclusions et perspectives

Dans ce document, nous avons présenté plusieurs résultats nouveaux, nous avons résumé l'état de l'art relatif à plusieurs domaines de recherches, et présenté quelques points de vue.

Tout d'abord, dans les chapitres 1 et 2, nous avons rappelé toutes la complexité et la richesse des systèmes dynamiques continus. Nous avons pour cela présenté un certain nombre de systèmes complexes, aux dynamiques intéressantes et souvent subtiles.

Les exemples du chapitre 1 sont presque tous issus de l'excellent ouvrage [Hirsch et al., 2003]. Les exemples du chapitre 2 illustrent des modèles de systèmes avec certaines concurrences, provenant de sources diverses comme la bioinformatique, la biologie, la virologie, et la virologie informatique. Il présente aussi plusieurs modèles venant de la théorie des jeux, ou en rapport avec l'algorithmique distribuée.

Par ces exemples, nous avons cherché à montrer plusieurs faits. Tout d'abord, que tous ces systèmes correspondent à une classe particulière d'équations différentielles, que nous avons nommé les problèmes de Cauchy polynomiaux, à propos de laquelle nous revenons à plusieurs reprises dans les chapitres ultérieurs. D'autre part, que de nombreux systèmes, naturels ou artificiels, peuvent intrinsèquement êtres considérés comme des modèles de calcul. Enfin, nous avons argumenté par plusieurs exemples, que même pour des systèmes à espace et temps discret, le bon modèle d'abstraction est souvent le continu.

Plus spécifiquement, dans le chapitre 2, nous avons cherché à montrer l'intérêt et le potentiel de l'utilisation des modèles continus, pour l'algorithmique distribuée. Nous avons discuté plusieurs problèmes importants qui se posent alors.

Dans le chapitre 3, nous avons présenté un survol, que nous estimons relativement complet, de la théorie, ou des théories, des calculs pour les systèmes à temps continu. Nous avons montré que ces théories permettent à la fois de comprendre la difficulté des questions relatives aux systèmes dynamiques à temps continu, et à la fois de comprendre la puissance des modèles analogiques à temps continu. Forts de ces deux perspectives, les résultats obtenus et présentés sont motivés par des domaines aussi variés que la vérification, la théorie du contrôle, la conception de circuits intégrés , les réseaux de neurones, l'analyse numérique, ou la théorie de la récursion sur les réels. Nous avons discuté à la fois les aspects calculabilité, complexité, robustesse aux bruits et aux imprécisions, en laissant une grande part dans le texte à des discussions de problèmes ouverts et de perspectives.

Les chapitres 4, 5, et l'annexe A présentent des panoramas synthétiques de certains de nos résultats.

Dans le chapitre 4, nous avons montré qu'il était possible de relier les fonctions solutions de problèmes de Cauchy polynomiaux aux fonctions GPAC calculables, ce qui était connu, mais aussi aux fonctions calculables en analyse récursive : les fonctions calculables par GPAC, dans un certain sens très naturel, correspondent exactement aux fonctions calculables au sens de l'analyse récursive. Nous avons par ailleurs, de façon orthogonale, montré qu'il était possible de caractériser les fonctions calculables (et élémentairement calculables) au sens de l'analyse récursive comme une classe de fonctions $\mathbb{R}$-récursives. À notre connaissance, c'est la première fois qu'une caractérisation algébrique des fonctions calculables au sens de l'analyse récursive aussi naturelle est obtenue.

Dans le chapitre 5, nous avons présenté un catalogue de l'ensemble de nos caractérisations des classes

de complexité dans le modèle de Blum Shub et Smale. Toutes les caractérisations sont dans l'esprit de la caractérisation par Bellantoni et Cook du temps polynomial dans [Bellantoni and Cook, 1992b]. Nous présentons ainsi une caractérisation de presque toutes les classes considérées dans le livre [Blum et al., 1998] sur le modèle de Blum Shub et Smale (à vrai dire, il ne manque essentiellement qu'une caractérisation de NC).

Dans l'annexe A, nous discutons de la question de l'éventuelle surpuissance des systèmes continus par rapport aux modèles classiques, comme les machines de Turing. Nous présentons plusieurs mauvaises compréhensions fréquentes de ce que dit la thèse de Church, et nous caractérisons la puissance de plusieurs classes de modèles à espace continu. Cette annexe contient plusieurs résultats originaux, et peut se voir comme une remise à jour des résultats qui existaient lorsque nous avions débuté notre thèse.

Perspectives

Chacun des chapitres est volontairement écrit avec une partie importante laissée aux perspectives : en particulier, le chapitre 3 possède toute une discussion des perspectives et travaux futurs qui nous semblent intéressants en rapport avec la théorie des calculs des systèmes à temps continu.

Mais à vrai dire, les chapitres 1, et 2 contiennent aussi de nombreuses questions explicitement citées comme telles. Chacune de ces questions fondamentales peut mener à des travaux pour plusieurs années. Le chapitre 4 offre lui aussi une partie conclusion et perspectives avec des pistes de travaux futurs.

Devant ce foisonnement de questions, nous nous permettons ici de souligner celles qui nous paraissent essentielles à l'heure où nous écrivons ces lignes.

Comprendre s'il existe un concept unificateur pour les modèles à temps continu comme la thèse de Church.

La situation est loin d'être aussi claire que pour les modèles discrets. Bien qu'il ait été montré que certains modèles à temps continu possèdent des capacités hypercalculatoires, tous ces résultats se basent sur l'utilisation d'une certaine quantité infinie de ressources, comme le temps, l'espace, la précision ou l'énergie. En général, il est souvent conjecturé que les modèles à temps continu "raisonnables" ne peuvent pas calculer plus que les machines de Turing.

Puisque les systèmes à temps continu robustes et analytiques peuvent simuler les machines de Turing avec un espace non-borné, nous pensons que les calculs digitaux et analogiques sont également puissants du point de vue de la calculabilité. En outre, comme nous l'avons vu, plusieurs résultats récents établissent l'équivalence entre les fonctions calculables par des équations différentielles polynomiales, les fonctions calculables par GPAC, et les fonctions calculables dans le sens de l'analyse récursive. Ce type de résultats renforce l'idée qu'il pourrait y avoir un cadre unifié pour les calculs à temps continu, similaire à celui qui existe en théorie de la calculabilité classique.

Nous avons argumenté tout au long de ce document de la puissance de modélisation, et des propriétés remarquables des fonctions solutions de problèmes de Cauchy polynomiaux. Nous pensons que cette classe de fonctions est un réel candidat pour fournir un concept unificateur, et des bases solides, à une théorie unifiée des calculs pour les systèmes continus.

Comprendre s'il existe une théorie de la complexité bien fondé, élégante et robuste pour les modèles à temps continu.

Nous avons vu dans le chapitre 3, que plusieurs pistes ont été considérées dans la littérature pour construire des théories de la complexité pour les systèmes continus. À vrai dire, à ce jour, il n'y a pas d'accord général entre les auteurs sur les définitions de base comme le temps de calculs, ou la taille des entrées. Les résultats établis à ce jour sont dérivés de concepts intrinsèques aux systèmes à temps continu considérés.

Puisque l'analyse récursive est un cadre bien établi et bien compris pour l'étude des problèmes de complexité pour les systèmes continus, nous pensons que mieux comprendre les relations qui existent entre les différentes approches et l'analyse récursive est de première importance.

En relation avec le chapitre 4, nous pensons qu'une façon d'attaquer le problème est de tenter de caractériser les fonctions calculables en temps polynomial en analyse récursive comme une classe de fonctions $\mathbb{R}$-récursives. Peut-on étendre des caractérisations à la Bellantoni et Cook [Bellantoni and Cook, 1992b] à l'analyse récursive ? Nous avons montré que les fonctions calculables par GPAC correspondaient aux fonctions calculables en analyse récursive. La simulation utilisée semble relier fortement le temps de calcul du GPAC et de la machine de Turing utilisée. Peut-on établir une telle équivalence au niveau de la complexité, et pas seulement de la calculabilité ?

Si la classe des fonctions solutions de problèmes de Cauchy polynomiaux permet, comme nous le conjecturons plus haut, de caractériser de façon élégante la notion de calcul "raisonnable" en temps continu, ne peut-on formuler simplement une théorie de la complexité basée sur ces fonctions ?

Mieux comprendre les modèles, et leurs propriétés.

Nous avons vu dans le chapitre 3 que très peu de travaux de recherches ont été faits à ce jour en ce qui concerne les effets du bruit ou des imprécisions sur les calculs à temps continu. À ce jour, l'essentiel des résultats concerne les systèmes à temps discret.

Nous avons évoqué plusieurs façons de modéliser le bruit ou l'incertitude : par exemple, en utilisant des modèles de bruit probabiliste, ou des modèles de bruit non-déterministe. Nous avons montré que chacune de ces façons mène à des résultats réellement différents. Mieux comprendre tout cela nous semble fondamental.

Par exemple, de nombreuses questions ouvertes surgissent si l'on demande si les résultats d'indécidabilité pour les systèmes à temps continu restent vrais pour les systèmes robustes. Cela est de première importance dans le domaine de la vérification par exemple, puisque cette question est reliée de façon très forte à la terminaison des procédures automatiques de vérification. Une meilleure compréhension des hypothèses avec lesquelles du bruit mène à de la décidabilité ou de l'indécidabilité nous semble nécessaire.

Nous pensons en outre qu'à ce jour, la question a uniquement été adressée avec un point de vue de la calculabilité, et pas de la complexité. Lorsque les problèmes sont prouvés décidables, quelle est l'influence de ces notions de bruit sur la complexité des problèmes ? Comment croit-elle avec la précision, ou le bruit ?

Nous pensons ces questions au coeur de problèmes fondamentaux reliés à des questions très profondes sur les modèles mathématiques actuels de notre monde physique : comment modéliser le bruit ? Qu'est-ce qu'un modèle robuste ? Comment modéliser l'incertitude ? Est-il pertinant, et quel est le sens, de chercher à faire des preuves formelles, donc d'une certaine façon certaines, à propos de phénomènes ou de systèmes incertains ?

Utiliser les modèles continus en algorithmique distribuée.

Nous avons volontairement insisté dans le chapitre 2 sur le fait que les systèmes continus apparaissent naturellement lorsqu'on cherche à discuter de systèmes de grandes tailles.

À ce jour, l'algorithmique distribuée n'est pas traitée avec ces outils. Une question fascinante est de comprendre si les théories des calculs des systèmes continus peuvent aider à comprendre, à programmer, et à maîtriser les modèles massivement parallèles. Étant données la taille des réseaux actuels, et la remise en cause de plusieurs hypothèses fortes de l'algorithmique distribuée classique dans certaines applications basées sur des modèles comme les réseaux de capteurs, il nous semble urgent de s'intéresser à cette question.

Nous avons présenté dans le chapitre 2 plusieurs modèles allant dans ce sens. Par exemple, nous avons montré que le modèle des protocoles de populations possède des propriétés calculatoires originales en termes de relations définissables en arithmétique de Presburger. Nous avons vu que ces protocoles peuvent être généralisés pour parler de grandes populations. La puissance exacte de modèles dans cet esprit reste très largement à comprendre.

Plus globalement, pour déterminer si les systèmes continus peuvent contribuer de façon profonde à l'algorithmique distribuée, plusieurs pistes semblent prioritaires.

Tout d'abord, il convient de bien et mieux comprendre les hypothèses qui permettent de passer, dans des domaines comme la physique, la biologique, la virologie, d'un système discret à une abstraction continue, pour comprendre de façon fine quand cela peut être possible en algorithmique distribuée. Passer au continu

sur des algorithmes distribués n'est pas sans poser plusieurs difficultés spécifiques. Par exemple, jusqu'à quel point peut-on ignorer les informations topologiques, comme cela est fait dans beaucoup de modèles de populations ?

Étant donné que dans un système distribué de grande taille, on ne peut espérer programmer, ou contrôler qu'une partie des agents, il nous semble que les modèles basés sur la théorie des jeux constituent de vraies pistes pour cela. Ces modèles sont intrinsèquement continus, et la littérature très récente foisonne d'exemples de systèmes qui se modélisent naturellement par ces outils.

Cependant, à ce jour, ces théories ont été développées dans un cadre mathématique assez abstrait, et parfois qui se mélange mal aux modèles et concepts classiques en algorithmique distribuée.

Par exemple, en algorithmique distribuée, l'évaluation de la complexité des algorithmes est souvent réalisée au pire cas. Le pire cas correspond à une certaine notion d'adversaire, mais qui est relativement différente de celle utilisée en théorie des jeux. Arriver à mélanger les notions d'adversaire de la théorie des jeux et de l'algorithmique classique est un vrai challenge.

En outre, à ce jour, il n'y a quasiment aucuns travaux sur les aspects dynamiques. La théorie des jeux permet de discuter les notions d'équilibre rationnel, mais pas le dynamisme. Des modèles comme ceux qui sont évoqués dans le chapitre 2, basés sur les jeux répétés ou la théorie évolutionnaire des jeux sont conçus pour modéliser le dynamisme de situations de concurrence. Mais presque aucune application de ces théories à l'algorithmique distribuée n'a été faite à ce jour. Réussir à modéliser le dynamisme, pour l'exploiter, des situations de concurrence dans des algorithmiques distribués nous semble un autre challenge de première importance.

Nous pensons que l'ensemble des résultats dans ce document peut apporter de façon significative à la compréhension de l'algorithmique, et de la complexité des modèles actuels et futurs. Cela est le sens de plusieurs de nos travaux scientifiques en cours autour de l'utilisation de la théorie algorithmique des jeux pour l'algorithmique distribuée.

Nous croyons en particulier fermement que le spectre futur des applications de la théorie des calculs des systèmes à temps continu est loin d'être limité à ses applications à ce jour, en particulier à la liste d'applications citée dans le chapitre 3.

Bibliographie

[Aaronson, 2005] Aaronson, S. (2005). NP-complete problems and physical reality. *ACM SIGACT News*, 36(1) :30–52.

[Abdi, 1994] Abdi, H. (1994). A neural network primer. *Journal of Biological Systems*, 2 :247–281.

[Abiteboul et al., 1990] Abiteboul, S., Simon, E., and Vianu, V. (1990). Non-deterministic languages to express deterministic transformations. In ACM, editor, *PODS '90. Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems : April 2–4, 1990, Nashville, Tennessee*, volume 51(1) of *Journal of Computer and Systems Sciences*, pages 218–229, New York, NY 10036, USA. ACM Press.

[Abiteboul and Vianu, 1989] Abiteboul, S. and Vianu, V. (1989). Fixpoint extensions of first-order logic and Datalog-like languages. In *Proceedings 4th Annual IEEE Symposium on Logic in Computer Science, LICS'89, Pacific Grove, CA, USA, 5–9 June 1989*, pages 71–79. IEEE Computer Society Press, Los Alamitos, CA.

[Abiteboul and Vianu, 1991] Abiteboul, S. and Vianu, V. (1991). Datalog extensions for database queries and updates. *Journal of Computer and System Sciences*, 43(1) :62–124.

[Adar and Huberman, 2000] Adar, E. and Huberman, B. A. (2000). Free riding on gnutella. *First Monday*, 5(10). *http ://firstmonday.org/issues/issue5_10/adar/index.html*.

[Adleman, 1994] Adleman, L. M. (1994). Molecular computation of solutions to combinatorial problems. *Science*, 266 :1021–1024.

[Alberganti, 2006] Alberganti, M. (2006). Mille milliards de mouchards. *Le Monde*. Edition du 2 juin 2006.

[Alur et al., 2001] Alur, R., Belta, C., and Ivancic, F. (2001). Hybrid modeling and simulation of biomolecular networks. In Benedetto, M. D. D. and Sangiovanni-Vincentelli, A. L., editors, *Hybrid Systems : Computation and Control, 4th International Workshop, HSCC 2001, Rome, Italy, March 28-30, 2001, Proceedings*, volume 2034 of *Lecture Notes in Computer Science*, pages 19–32. Springer.

[Alur et al., 1995] Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T. A., Ho, P. H., Nicollin, X., Olivero, A., Sifakis, J., and Yovine, S. (1995). The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1) :3–34.

[Alur and Dill, 1990] Alur, R. and Dill, D. L. (1990). 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.

[Alur and Dill, 1994] Alur, R. and Dill, D. L. (1994). A theory of timed automata. *Theoretical Computer Science*, 126(2) :183–235. Fundamental Study.

[Alur and Madhusudan, 2004] Alur, R. and Madhusudan, P. (2004). Decision problems for timed automata : A survey. In Bernardo, M. and Corradini, F., editors, *Formal Methods for the Design of Real-Time Systems, International School on Formal Methods for the Design of Computer, Communication and Software Systems, SFM-RT 2004, Bertinoro, Italy, September 13-18, 2004, Revised Lectures*, volume 3185 of *Lecture Notes in Computer Science*, pages 1–24. Springer.

[Angluin et al., 2005a] Angluin, D., Aspnes, J., Chan, M., Fischer, M. J., Jiang, H., and Peralta, R. (2005a). Stably computable properties of network graphs. In Prasanna, V. K., Iyengar, S., Spirakis, P., and Welsh,

M., editors, *Distributed Computing in Sensor Systems : First IEEE International Conference, DCOSS 2005, Marina del Rey, CA, USE, June/July, 2005, Proceedings*, volume 3560 of *Lecture Notes in Computer Science*, pages 63–74. Springer-Verlag.

[Angluin et al., 2004] Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M. J., and Peralta, R. (2004). Computation in networks of passively mobile finite-state sensors. In *Twenty-Third ACM Symposium on Principles of Distributed Computing*, pages 290–299. ACM Press.

[Angluin et al., 2006a] Angluin, D., Aspnes, J., and Eisenstat, D. (2006a). Fast computation by population protocols with a leader. In *20th International Symposium on Distributed Computing (DISC'2006)*, Lecture Notes in Computer Science. Springer. To appear.

[Angluin et al., 2006b] Angluin, D., Aspnes, J., and Eisenstat, D. (2006b). Stably computable predicates are semilinear. In *PODC '06 : Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, pages 292–299, New York, NY, USA. ACM Press.

[Angluin et al., 2005b] Angluin, D., Aspnes, J., Fischer, M. J., and Jiang, H. (2005b). Self-stabilizing population protocols. In *Ninth International Conference on Principles of Distributed Systems (OPODIS'2005)*, Lecture Notes in Computer Science, pages 79–90. Springer. To appear.

[Anshelevich et al., 2003] Anshelevich, E., Dasgupta, A., Tardos, E., and Wexler, T. (2003). Near-optimal network design with selfish agents. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 511–520. ACM Press.

[Antsaklis, 2000] Antsaklis, P. J., editor (2000). *Proceedings of the IEEE*, volume 88.

[Arnold, 1989] Arnold, V. I. (1989). *Mathematical methods of classical mechanics*, volume 60 of *Graduate texts in Mathematics*. Springer, second edition.

[Arnold, 1992] Arnold, V. I. (1992). *Ordinary differential equations*. Springer-Verlag, Berlin.

[Artobolevskii, 1964] Artobolevskii, I. (1964). *Mechanisms for the generation of plane curves*. Macmillan, New York. Translated by R.D. Wills and W. Johnson.

[Asarin, 2004] Asarin (2004). Challenges in timed languages : From applied theory to basic theory. *Bulletin of the European Association for Theoretical Computer Science*, 83 :106–120.

[Asarin and Collins, 2005] Asarin and Collins (2005). Noisy turing machines. In *ICALP : Annual International Colloquium on Automata, Languages and Programming*.

[Asarin, 1995] Asarin, E. (1995). Chaos and undecidabilty (draft). Avalaible in *http ://www.liafa.jussieu.fr/ asarin/*.

[Asarin, 1998] Asarin, E. (1998). Equations on timed languages. In Henzinger, T. A. and Sastry, S., editors, *Hybrid Systems : Computation and Control, First International Workshop, HSCC'98, Berkeley, California, USA, April 13-15, 1998, Proceedings*, volume 1386 of *Lecture Notes in Computer Science*, pages 1–12. Springer.

[Asarin, 2006] Asarin, E. (2006). Noise and decidability. Continuous Dynamics and Computability Colloquium. Video and sound available trough "Diffusion des savoirs de l'Ecole Normale Supérieure", on *http ://www.diffusion.ens.fr/en/index.php ?res=conf&idconf=1226*.

[Asarin and Bouajjani, 2001] Asarin, E. and Bouajjani, A. (2001). Perturbed turing machines and hybrid systems. In *Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science (LICS-01)*, pages 269–278, Los Alamitos, CA. IEEE Computer Society Press.

[Asarin et al., 1997] Asarin, E., Caspi, P., and Maler, O. (1997). A Kleene theorem for timed automata. In *Proceedings, 12th Annual IEEE Symposium on Logic in Computer Science*, pages 160–171, Warsaw, Poland. IEEE Computer Society Press.

[Asarin et al., 2002] Asarin, E., Caspi, P., and Maler, O. (2002). Timed regular expressions. *Journal of the ACM*, 49(2) :172–206.

[Asarin and Dima, 2002] Asarin, E. and Dima, C. (2002). Balanced timed regular expressions. *Electronic Notes in Theoretical Computer Science*, 68(5).

[Asarin and Maler, 1998] Asarin, E. and Maler, O. (1998). Achilles and the tortoise climbing up the arithmetical hierarchy. *Journal of Computer and System Sciences*, 57(3) :389–398.

[Asarin et al., 1995] Asarin, E., Maler, O., and Pnueli, A. (1995). Reachability analysis of dynamical systems having piecewise-constant derivatives. *Theoretical Computer Science*, 138(1) :35–65.

[Asarin and Schneider, 2002] Asarin, E. and Schneider, G. (2002). Widening the boundary between decidable and undecidable hybrid systems. In Brim, L., Jancar, P., Kretínský, M., and Kucera, A., editors, *CONCUR 2002 - Concurrency Theory, 13th International Conference, Brno, Czech Republic, August 20-23, 2002, Proceedings*, volume 2421 of *Lecture Notes in Computer Science*, pages 193–208. Springer.

[Asarin et al., 2001] Asarin, E., Schneider, G., and Yovine, S. (2001). On the decidability of the reachability problem for planar differential inclusions. In Benedetto, M. D. D. and Sangiovanni-Vincentelli, A. L., editors, *Hybrid Systems : Computation and Control, 4th International Workshop, HSCC 2001, Rome, Italy, March 28-30, 2001, Proceedings*, volume 2034 of *Lecture Notes in Computer Science*, pages 89–104. Springer.

[Axelrod, 1984] Axelrod, R. M. (1984). *The Evolution of Cooperation*. Basic Books.

[Balcàzar et al., 1990] Balcàzar, J., Dìaz, J., and Gabarrò, J. (1990). *Structural Complexity II*, volume 22 of *EATCS Monographs on Theoretical Computer Science*. Springer.

[Balcázar et al., 1988] Balcázar, J. L., Diáz, J., and Gabarró, J. (1988). *Structural Complexity I*, volume 11 of *EATCS Monographs on Theoretical Computer Science*. Springer.

[Balcázar et al., 1993] Balcázar, J. L., Gavaldà, R., Siegelmann, H. T., and Sontag, E. D. (1993). Some structural complexity aspects of neural computation. In *8th IEEE Conference on Structure in Complexity Theory*, pages 253–256. IEEE Computer Society Press.

[Batt et al., 2006] Batt, G., Casey, R., de Jong, H., Geiselmann, J., Gouzé, J.-L., Page, M., Ropers, D., Sari, T., and Schneider, D. (2006). Qualitative analysis of the dynamics of genetic regulatory networks using piecewise-linear models. In A. Maass, S. Martinez, E. P., editor, *Mathematical and Computational Methods in Biology*. Hermann.

[Batt et al., 2005] Batt, G., Ropers, D., de Jong, H., Geiselmann, J., Mateescu, R., Page, M., and Schneider, D. (2005). Validation of qualitative models of genetic regulatory networks by model checking : analysis of the nutritional stress response in escherichia coli. *Bioinformatics*, 25 :19–28.

[Beaufils, 2000] Beaufils, B. (2000). *Modèles et simulations informatiques des problèmes de coopération entre agents*. PhD thesis, Université de Lille I.

[Beauquier, 1998] Beauquier, D. (1998). Pumping lemmas for timed automata. In Nivat, M., editor, *Foundations of Software Science and Computation Structure, First International Conference, FoSSaCS'98, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS'98, Lisbon, Portugal, March 28 - April 4, 1998, Proceedings*, volume 1378 of *Lecture Notes in Computer Science*, pages 81–94. Springer.

[Bellantoni, 1994] Bellantoni, S. (1994). Predicative recursion and the polytime hierarchy. In Clote, P. and Remmel, J., editors, *Feasible Mathematics II*, *Perspectives in Computer Science*. Birkhäuser.

[Bellantoni and Cook, 1992a] Bellantoni, S. and Cook, S. (1992a). A new recursion-theoretic characterization of the poly-time functions. *Computational Complexity*, 2 :97–110.

[Bellantoni and Cook, 1992b] Bellantoni, S. and Cook, S. (1992b). A new recursion-theoretic characterization of the poly-time functions. *Computational Complexity*, 2(2) :97–110.

[Ben-Hur et al., 2004a] Ben-Hur, Feinberg, Fishman, and Siegelmann (2004a). Random matrix theory for the analysis of the performance of an analog computer : a scaling theory. *Physics Letters A*, pages 204–209.

[Ben-Hur et al., 2003] Ben-Hur, A., Feinberg, J., Fishman, S., and Siegelmann, H. T. (2003). Probabilistic analysis of a differential equation for linear programming. *Journal of Complexity*, 19(4) :474–510.

[Ben-Hur et al., 2004b] Ben-Hur, A., Roitershtein, A., and Siegelmann, H. T. (2004b). On probabilistic analog automata. *Theoretical Computer Science*, 320(2–3) :449–464.

[Ben-Hur et al., 2002] Ben-Hur, A., Siegelmann, H. T., and Fishman, S. (2002). A theory of complexity for continuous time systems. *Journal of Complexity*, 18(1) :51–86.

[Binmore, 1999] Binmore, K. (1999). *Jeux et Théorie des jeux*. DeBoeck Universié, Paris-Bruxelles. Traduit du livre "Fun and Games : a text on game theory" par Francis Bismans et Eulalia Damaso.

[Blondel and Tsitsiklis, 1999] Blondel, V. D. and Tsitsiklis, J. N. (1999). Complexity of stability and controllability of elementary hybrid systems. *Automatica*, 35(3) :479–489.

[Blondel and Tsitsiklis, 2000] Blondel, V. D. and Tsitsiklis, J. N. (2000). A survey of computational complexity results in systems and control. *Automatica*, 36(9) :1249–1274.

[Blum et al., 1998] Blum, L., Cucker, F., Shub, M., and Smale, S. (1998). *Complexity and Real Computation*. Springer-Verlag.

[Blum et al., 1989] Blum, L., Shub, M., and Smale, S. (1989). On a theory of computation and complexity over the real numbers ; NP completeness, recursive functions and universal machines. *Bulletin of the American Mathematical Society*, 21(1) :1–46.

[Boker and Dershowitz, 2005] Boker, U. and Dershowitz, N. (2005). A speculative Church-Turing theorem. Preprint.

[Bonner, 1989] Bonner, A. J. (1989). Hypothetical Datalog : Negation and linear recursion. In *Proceedings of the Eighth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 286–300, Philadelphia, Pennsylvania. ACM Press.

[Bournez, 1999a] Bournez, O. (1999a). Achilles and the Tortoise climbing up the hyper-arithmetical hierarchy. *Theoretical Computer Science*, 210(1) :21–71.

[Bournez, 1999b] Bournez, O. (1999b). *Complexité Algorithmique des Systèmes Dynamiques Continus et Hybrides*. PhD thesis, Ecole Normale Supérieure de Lyon.

[Bournez, 2006] Bournez, O. (2006). How much can analog and hybrid systems be proved (super-)turing. *Applied Mathematics and Computation*, 178(1) :58–71.

[Bournez et al., 2006a] Bournez, O., Campagnolo, M. L., Graça, D. S., and Hainry, E. (2006a). The general purpose analog computer and computable analysis are two equivalent paradigms of analog computation. In Cai, J., Cooper, S. B., and Li, A., editors, *Theory and Applications of Models of Computation, Third International Conference, TAMC 2006, Beijing, China, May 15-20, 2006, Proceedings*, volume 3959 of *Lecture Notes in Computer Science*, pages 631–643. Springer.

[Bournez and Cosnard, 1996] Bournez, O. and Cosnard, M. (1996). On the computational power of dynamical systems and hybrid systems. *Theoretical Computer Science*, 168(2) :417–459.

[Bournez et al., 2003] Bournez, O., Cucker, F., de Naurois, P. J., and Marion, J.-Y. (2003). Computability over an arbitrary structure. sequential and parallel polynomial time. In Gordon, A. D., editor, *Foundations of Software Science and Computational Structures, 6th International Conference (FOSSACS'2003)*, volume 2620 of *Lecture Notes in Computer Science*, pages 185–199, Warsaw. Springer.

[Bournez et al., 2004a] Bournez, O., Cucker, F., de Naurois, P. J., and Marion, J.-Y. (2004a). Tailoring recursion to characterize non-deterministic complexity classes over arbitrary structures. In *In 2nd APPSEM II Workshop (APPSEM'04)*.

[Bournez et al., 2004b] Bournez, O., Cucker, F., de Naurois, P. J., and Marion, J.-Y. (2004b). Tailoring recursion to characterize non-deterministic complexity classes over arbitrary structures. In *3rd IFIP International Conference on Theoretical Computer Science - TCS'2004*, Toulouse, France. Kluwer Academic Press.

[Bournez et al., 2005a] Bournez, O., Cucker, F., de Naurois, P. J., and Marion, J.-Y. (2005a). Implicit complexity over an arbitrary structure : Sequential and parallel polynomial time. *Journal of Logic and Computation*, 15(1) :41–58.

[Bournez et al., 2005b] Bournez, O., Cucker, F., de Naurois, P. J., and Marion, J.-Y. (2005b). Logical characterizations of $p_{\mathcal{K}}$ and $np_{\mathcal{K}}$ over an arbitrary structure k. In *3rd APPSEM II Workshop (APPSEM'05), Frauenchiemsee, Germany, 2005. Also accepted for presentation at CIE 2005 : New Computational Paradigms.*

[Bournez et al., 2006b] Bournez, O., Cucker, F., de Naurois, P. J., and Marion, J.-Y. (2006b). Implicit complexity over an arbitrary structure : Quantifier alternations. *Information and Computation*, 202(2) :210–230.

[Bournez et al., 2002] Bournez, O., de Naurois, P., and Marion, J.-Y. (2002). Safe recursion and calculus over an arbitrary structure. In *Implicit Computational Complexity - ICC'02*, Copenhagen, Denmark.

[Bournez and Hainry, 2004a] Bournez, O. and Hainry, E. (2004a). An analog characterization of elementarily computable functions over the real numbers. In Diaz, J., Karhumäki, J., Lepisto, A., and Sannella, D. T., editors, *31th International Colloquium on Automata Languages and Programming (ICALP'04)*, volume 3142 of *Lecture Notes in Computer Science*, pages 269–280, Turku, Finland. Springer.

[Bournez and Hainry, 2004b] Bournez, O. and Hainry, E. (2004b). Real recursive functions and real extentions of recursive functions. In Margenstern, M., editor, *Machines, Computations and Universality (MCU'2004)*, volume 3354 of *Lecture Notes in Computer Science*, Saint-Petersburg, Russia.

[Bournez and Hainry, 2005] Bournez, O. and Hainry, E. (2005). Elementarily computable functions over the real numbers and $\mathbb{R}$-sub-recursive functions. *Theoretical Computer Science*, 348(2–3) :130–147.

[Bournez and Hainry, 2007] Bournez, O. and Hainry, E. (2007). Recursive analysis characterized as a class of real recursive functions. *Fundamenta Informaticae*. To appear.

[Bouyer et al., 2000a] Bouyer, P., Dufourd, C., Fleury, E., and Petit, A. (2000a). Are timed automata updatable ? In Emerson, E. A. and Sistla, A. P., editors, *Computer Aided Verification, 12th International Conference, CAV 2000, Chicago, IL, USA, July 15-19, 2000, Proceedings*, volume 1855 of *Lecture Notes in Computer Science*, pages 464–479. Springer.

[Bouyer et al., 2000b] Bouyer, P., Dufourd, C., Fleury, E., and Petit, A. (2000b). Expressiveness of updatable timed automata. In Nielsen, M. and Rovan, B., editors, *Mathematical Foundations of Computer Science 2000, 25th International Symposium, MFCS 2000, Bratislava, Slovakia, August 28 - September 1, 2000, Proceedings*, volume 1893 of *Lecture Notes in Computer Science*, pages 232–242. Springer.

[Bouyer and Petit, 1999] Bouyer, P. and Petit, A. (1999). Decomposition and composition of timed automata. In Wiedermann, J., van Emde Boas, P., and Nielsen, M., editors, *Automata, Languages and Programming, 26th International Colloquium, ICALP'99, Prague, Czech Republic, July 11-15, 1999, Proceedings*, volume 1644 of *Lecture Notes in Computer Science*, pages 210–219. Springer.

[Bouyer and Petit, 2002] Bouyer, P. and Petit, A. (2002). A Kleene/Büchi-like theorem for clock languages. *Journal of Automata, Languages and Combinatorics*, 7(2) :167–186.

[Bower and Bolouri, 2000] Bower, J. and Bolouri, H. (2000). *Computational Modeling of Genetic and Biochemical Networks*. The MIT Press.

[Bowles, 1996] Bowles, M. D. (1996). U.S. technological enthusiasm and British technological skepticism in the age of the analog brain. *IEEE Annals of the History of Computing*, 18(4) :5–15.

[Branicky, 1995a] Branicky, M. S. (1995a). *Studies in Hybrid Systems : Modeling, Analysis, and Control*. PhD thesis, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA, USA.

[Branicky, 1995b] Branicky, M. S. (1995b). Universal computation and other capabilities of hybrid and continuous dynamical systems. *Theoretical Computer Science*, 138(1) :67–100.

[Brattka, 2003] Brattka, V. (2003). Computability over topological structures. In Cooper, S. B. and Goncharov, S. S., editors, *Computability and Models*, pages 93–136. Kluwer Academic Publishers, New York.

[Brémaud, 2001] Brémaud, P. (2001). *Markov Chains, Gibbs Fields, Monte Carlo Simulation, and Queues*. Springer-Verlag, New York.

[Briest et al., 2005] Briest, P., Krysta, P., and Vöcking, B. (2005). Approximation techniques for utilitarian mechanism design. In Gabow, H. N. and Fagin, R., editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC), Baltimore, MD, USA, May 22-24, 2005*, pages 39–48. ACM Press.

[Brockett, 1989] Brockett, R. W. (1989). Smooth dynamical systems which realize arithmetical and logical operations. In Nijmeijer, H. and Schumacher, J. M., editors, *Three Decades of Mathematical Systems Theory*, volume 135 of *Lecture Notes in Computer Science*, pages 19–30. Springer.

[Brockett, 1991] Brockett, R. W. (1991). Dynamical systems that sort lists, diagonalize matrices, and solve linear programming problems. *Linear Algebra and its Applications*, 146 :79–91.

[Brockett, 1994] Brockett, R. W. (1994). Dynamical systems and their associated automata. In U. Helmke, R. M. and Saurer, J., editors, *Systems and Networks : Mathematical Theory and Applications*, volume 77, pages 49–69. Akademi-Verlag, Berlin.

[Bush, 1931] Bush, V. (1931). The differential analyser. *Journal of the Franklin Institute*, 212(4) :447–488.

[Calude and Pavlov, 2002] Calude, C. S. and Pavlov, B. (2002). Coins, quantum measurements, and turning's barrier. *Quantum Information Processing*, 1(1-2) :107–127.

[Campagnolo, 2002] Campagnolo, M. (2002). The complexity of real recursive functions. In Calude, C., Dinneen, M., and Peper, F., editors, *Unconventional Models of Computation, UMC'02*, number 2509 in LNCS, pages 1–14. Springer-Verlag.

[Campagnolo, 2004a] Campagnolo, M. (2004a). Continuous time computation with restricted integration capabilities. *Theoretical Computer Science*, 317(4) :147–165.

[Campagnolo, 2004b] Campagnolo, M. (2004b). Continuous time computation with restricted integration capabilities. *Theoretical Computer Science*, 317 :147–165.

[Campagnolo et al., 2000a] Campagnolo, M., Moore, C., and Costa, J. F. (2000a). An analog characterization of the subrecursive functions. In Kornerup, P., editor, *Proc. 4th Conference on Real Numbers and Computers*, pages 91–109. Odense University Press.

[Campagnolo et al., 2000b] Campagnolo, M., Moore, C., and Costa, J. F. (2000b). Iteration, inequalities, and differentiability in analog computers. *Journal of Complexity*, 16(4) :642–660.

[Campagnolo et al., 2002] Campagnolo, M., Moore, C., and Costa, J. F. (2002). An analog characterization of the Grzegorczyk hierarchy. *Journal of Complexity*, 18(4) :977–1000.

[Campagnolo, 2001] Campagnolo, M. L. (2001). *Computational complexity of real valued recursive functions and analog circuits*. PhD thesis, IST, Universidade Técnica de Lisboa.

[Campagnolo and Ojakian, 2006] Campagnolo, M. L. and Ojakian, K. (2006). The methods of approximation and lifting in real computation. *Electronic Notes in Theoretical Computer Science*. to appear.

[Casey, 1996] Casey, M. (1996). The dynamics of discrete-time computation, with application to recurrent neural networks and finite state machine extraction. *Neural Computation*, 8 :1135–1178.

[Casey, 1998] Casey, M. (1998). Correction to proof that recurrent neural networks can robustly recognize only regular languages. *Neural Computation*, 10 :1067–1069.

[Ceraens and Viksna, 1996] Ceraens, K. and Viksna, J. (1996). Deciding reachability for planar multi-polynomial systems. In *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*, page 389.

[Chen et al., 2003] Chen, Z., Gao, L., and Kwiat, K. (2003). Modeling the spread of active worms. In *Prooceedings of 22nd Annual IEEE Conference on Computer Communications (INFOCOM'2003)*. IEEE Computer Society Press.

[Church, 1936] Church, A. (1936). An unsolvable problem of elementary number theory. *American Journal of Mathematics,*, 58 :345–363. Also in [Davis, 1965].

[Cleland, 2004] Cleland, C. E. (2004). The concept of computability. *Theoretical Computer Science*, 317(1–3) :209–225.

[Clote, 1998] Clote, P. (1998). Computational models and function algebras. In Griffor, E. R., editor, *Handbook of Computability Theory*, pages 589–681. North-Holland, Amsterdam.

[Cobham, 1962] Cobham, A. (1962). The intrinsic computational difficulty of functions. In Bar-Hillel, Y., editor, *Proceedings of the International Conference on Logic, Methodology, and Philosophy of Science*, pages 24–30. North-Holland, Amsterdam.

[Coddington and Levinson, 1972] Coddington, E. A. and Levinson, N. (1972). *Theory of Ordinary Differentiel Equations*. McGraw-Hill.

[Cohen and Grossberg, 1983] Cohen, M. A. and Grossberg, S. (1983). Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(5) :815–826.

[Collins, 2005] Collins, P. (2005). Continuity and computability on reachable sets. *Theoretical Computer Science*, 341 :162–195.

[Collins and Lygeros, 2005] Collins, P. and Lygeros, J. (2005). Computability of finite-time reachable sets for hybrid systems. In *Proceedings of the 44th IEEE Conference on Decision and Control and the European Control Conference*. IEEE Computer Society Press.

[Collins and van Schuppen, 2004] Collins, P. and van Schuppen, J. H. (2004). Observability of piecewise-affine hybrid systems. In Alur, R. and Pappas, G. J., editors, *Hybrid Systems : Computation and Control, 7th International Workshop, HSCC 2004, Philadelphia, PA, USA, March 25-27, 2004, Proceedings*, volume 2993 of *Lecture Notes in Computer Science*, pages 265–279. Springer.

[Copeland, 1998] Copeland (1998). Even Turing machines can compute uncomputable functions. In on Unconventional Models of Computation, I. C., UMC, originally C. S. Calude, L., Casti, J., and Dinneen, M. J., editors, *Unconventional Models of Computation, Springer, 1998*, volume 1. Springer Verlag.

[Copeland, 2002] Copeland, B. J. (Fall 2002). The Church-Turing thesis. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Stanford University. Available online at : *http ://plato.stanford.edu/entries/church-turing/*.

[Copeland and Sylvan, 1999] Copeland, B. J. and Sylvan, R. (1999). Beyond the universal Turing machine. *Australasian Journal of Philosophy*, 77 :46–66.

[Correa et al., 2004] Correa, J. R., Schulz, A. S., and Moses, N. E. S. (2004). Selfish routing in capacitated networks. *Mathematical Operational Research*, 29(4) :961–976.

[Costa and Mycka, 2006a] Costa, J. F. and Mycka, J. (2006a). An analytic condition for $P \neq NP$. submitted.

[Costa and Mycka, 2006b] Costa, J. F. and Mycka, J. (2006b). The conjecture $P \neq NP$ given by some analytic condition. In Bekmann, A., Berger, U., Löwe, B., and Tucker, J., editors, *Logical Approaches to Computational Barriers, Second conference on Computability in Europe, CiE 2006*, pages 47–57, Swansea, UK. Report CSR 7-26, Report Series, University of Wales Swansea Press, 2006.

[Costa and Mycka, 2006c] Costa, J. F. and Mycka, J. (2006c). A new conceptual framework for analog computation. submitted.

[Coullet et al., 2004] Coullet, P., Monticelli, M., and Treiner, J. (2004). L'algorithme de Newton-Hooke. *Bulletin de l'Union des physiciens*, 861 :193–206.

[Coward, 2006] Coward, D. (2006). Doug coward's analog computer musueum. *http ://dcoward.best.vwh.net/analog/*.

[Cucker, 1993] Cucker, F. (1993). On the complexity of quantifier elimination : the structural approach. *The Computer Journal*, 36 :400–408.

[Cucker and Meer, 1999] Cucker, F. and Meer, K. (1999). Logics which capture complexity classes over the reals. *Journal of Symbolic Logic*, 64(1) :363–390.

[Davies, 2001] Davies, E. B. (2001). Building infinite machines. *The British Journal for the Philosophy of Science*, 52 :671–682.

[Davis, 1965] Davis, M. (1965). *The undecidable*. Raven Press.

[de Jong, 2002] de Jong, H. (2002). Modeling and simulation of genetic regulatory systems : A literature review. *Journal of Computational Biology*, 9(1) :67–103.

[Dee and Ghil, 1984] Dee, D. and Ghil, M. (1984). Boolean difference equations, I : Formulation and dynamic behavior. *SIAM Journal on Applied Mathematics*, 44(1) :111–126.

[Delvenne et al., 2004] Delvenne, Kurka, and Blondel (2004). Computational universality in symbolic dynamical systems. In *MCU : International Conference on Machines, Computations, and Universality*, volume 3354 of *Lecture Notes in Computer Science*, pages 104–115. Springer.

[Deutsch, 1985] Deutsch, D. (1985). Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society (London), Serie A*, 400 :97–117.

[Dupuis and Berland, 2004] Dupuis, G. and Berland, N. (2004). Belousov-zhabotinsky reaction. In Scott, A., editor, *Encyclopedia of Nonlinear Science*, volume 3600. Routledge, Taylor and Francis group.

[Durand-Lose, 2005] Durand-Lose, J. (2005). Abstract geometrical computation : Turing-computing ability and undecidability. In Cooper, S. B., Löwe, B., and Torenvliet, L., editors, *New Computational Paradigms, First Conference on Computability in Europe, CiE 2005, Amsterdam, The Netherlands, June 8-12, 2005, Proceedings*, volume 3526 of *Lecture Notes in Computer Science*, pages 106–116. Springer.

[Dyer et al., 2002] Dyer, M. E., Goldberg, L. A., Greenhill, C. S., Istrate, G., and Jerrum, M. (2002). Convergence of the iterated prisoner's dilemma game. *Combinatorics, Probability & Computing*, 11(2).

[Earman and Norton, 1993] Earman, J. and Norton, J. D. (1993). Forever is a day : Supertasks in Pitowksy and Malament-Hogarth spacetimes. *Philosophy of Science*, 60(1) :22–42.

[Ebbinghaus and Flum, 1995] Ebbinghaus, H. and Flum, J. (1995). *Finite Model Theory*. Perspectives in Mathematical Logic, Omega Series. Springer–Verlag, Heidelberg.

[Etesi and Németi, 2002] Etesi, G. and Németi, I. (2002). Non-turing computations via malament-hogarth space-times. *International Journal Theoretical Physics*, 41 :341–370.

[Fagin, 1974] Fagin, R. (1974). Generalized first-order spectra and polynomial-time recognizable sets. In Karp, R. M., editor, *Complexity in Computer Computations*, pages 43–73. American Mathematics Society, Providence R.I.

[Faybusovich, 1991] Faybusovich, L. (1991). Dynamical systems which solve optimization problems with linear constraints. *IMA Journal of Mathematical Control and Information*, 8 :135–149.

[Faybusovich, L., 1991] Faybusovich, L. (1991). Hamiltonian structure of dynamical systems which solve linear programming problems. *Physics*, D53 :217–232.

[Feigenbaum et al., 2001] Feigenbaum, J., Papadimitriou, C. H., and Shenker, S. (2001). Sharing the cost of multicast transmissions. *Journal of Computer and System Sciences*, 63(1) :21–41.

[Field and Burger, 1985] Field, R. and Burger, M., editors (1985). *Oscillations and Traveling Waves in Chemical Systems*. Wiley, New York.

[Filippov, 1988] Filippov, A. (1988). *Differential equations with discontinuous right-hand sides.* Kluwer Academic Publishers.

[Finkel, 2005] Finkel, O. (2005). On decision problems for timed automata. *Bulletin of the EATCS*, 87 :185–190.

[Finkel, 2006] Finkel, O. (2006). On the shuffle of regular timed languages. *Bulletin of the European Association for Theoretical Computer Science*, 88 :182–184. Technical Contributions.

[Foy, 2004] Foy, J. (2004). A dynamical system which must be stable whose stability cannot be proved. *Theoretical Computer Science*, 328(3) :355–361.

[Fribourg et al., 2004] Fribourg, L., Messika, S., and Picaronny, C. (2004). Coupling and self-stabilization. In Guerraoui, R., editor, *Distributed Computing, 18th International Conference, DISC 2004, Amsterdam, The Netherlands, October 4-7, 2004, Proceedings*, volume 3274 of *Lecture Notes in Computer Science*, pages 201–215. Springer.

[Fränzle, 1999] Fränzle, M. (1999). Analysis of hybrid systems : An ounce of realism can save an infinity of states. In Flum, J. and Rodríguez-Artalejo, M., editors, *Computer Science Logic (CSL'99)*, volume 1683 of *Lecture Notes in Computer Science*, pages 126–140. Springer Verlag.

[Gandy, 1980] Gandy, R. (1980). Church's thesis and principles for mechanisms. *The Kleene Symposium*, pages 123–148.

[Ghosh and Tomlin, 2001] Ghosh, R. and Tomlin, C. (2001). Lateral inhibition through delta-notch signaling : A piecewise affine hybrid model. In Benedetto, M. D. D. and Sangiovanni-Vincentelli, A. L., editors, *Hybrid Systems : Computation and Control, 4th International Workshop, HSCC 2001, Rome, Italy, March 28-30, 2001, Proceedings*, volume 2034 of *Lecture Notes in Computer Science*, pages 232–246. Springer.

[Gibson and Mjolsness, 2000] Gibson, M. and Mjolsness, E. (2000). *Computational Modeling of Genetic and Biochemical Networks [Bower and Bolouri, 2000]*, chapter Modeling the activity of single genes. The MIT Press.

[Gillespie, 1977] Gillespie, D. T. (1977). Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, 81(25) :2340–2361.

[Goode, 1994] Goode, J. B. (1994). Accessible telephone directories. *The Journal of Symbolic Logic*, 59(1) :92–105.

[Goodwin, 1963] Goodwin, B. (1963). *Temporal Organization in Cells*. Academic Press.

[Goodwin, 1965] Goodwin, B. (1965). Oscillatory behavior in enzymatic control processes. In Weber, G., editor, *Advances in Enzyme Regulation*, pages 425–438. Pergamon Press, Oxford.

[Gori and Meer, 2002] Gori, M. and Meer, K. (2002). A step towards a complexity theory for analog systems. *Mathematical Logic Quarterly*, 48(Suppl. 1) :45–58.

[Graça et al., 2005] Graça, D., Campagnolo, M., and Buescu, J. (2005). Robust simulations of Turing machines with analytic maps and flows. In Cooper, B., Loewe, B., and Torenvliet, L., editors, *Proceedings of CiE'05, New Computational Paradigms*, volume 3526 of *Lecture Notes in Computer Science*, pages 169–179. Springer-Verlag.

[Graça et al., 2006] Graça, D. S., Zhong, N., and Buescu, J. (2006). Computability, noncomputability and undecidability of maximal intervals of IVPs. *Transactions of the American Mathematical Society*.

[Grädel and Gurevich, 1998] Grädel, E. and Gurevich, Y. (1998). Metafinite model theory. *Information and Computation*, 140(1) :26–81.

[Grädel and Meer, 1995] Grädel, E. and Meer, K. (1995). Descriptive complexity theory over the real numbers. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on the Theory of Computing*, pages 315–324, Las Vegas, Nevada. ACM Press.

[Graça, 2002] Graça, D. (2002). The general purpose analog computer and recursive functions over the reals. Master's thesis, IST, Universidade Técnica de Lisboa.

[Graça, 2006] Graça, D. (2006). *Thèse de Daniel Graça, titre à fixer*. PhD thesis, IST. en rédaction.

[Graça, 2004] Graça, D. S. (2004). Some recent developments on Shannon's general purpose analog computer. *Mathematical Logic Quarterly*, 50(4–5) :473–485.

[Graça et al., 2006] Graça, D. S., Campagnolo, M. L., , and Buescu, J. (2006). Computability with polynomial differential equations. Technical report, CLC, Department of Mathematics, Instituto Superior Técnico, 1049-001 Lisboa, Portugal.

[Graça and Costa, 2003] Graça, D. S. and Costa, J. F. (2003). Analog computers and recursive functions over the reals. *Journal of Complexity*, 19(5) :644–664.

[Grigorieff and Margenstern, 2004] Grigorieff, S. and Margenstern, M. (2004). Register cellular automata in the hyperbolic plane. *Fundamenta Informaticae*, 1(61) :19–27.

[Gruska, 1997] Gruska, J. (1997). *Foundations of computing*. International Thomson Publishing.

[Grzegorczyk, 1957] Grzegorczyk, A. (1957). On the definitions of computable real continuous functions. *Fundamenta Mathematicae*, 44 :61–71.

[Gupta et al., 1997] Gupta, V., Henzinger, T. A., and Jagadeesan, R. (1997). Robust timed automata. In Maler, O. O., editor, *Hybrid and real-time systems : international workshop, HART '97, Grenoble, France, March 26–28, 1997*, volume 1201 of *Lecture Notes in Computer Science*, pages 331–345, New York, NY, USA. Springer-Verlag Inc.

[Gurevich, 1983] Gurevich, Y. (1983). Algebras of feasible functions. In *Twenty Fourth Symposium on Foundations of Computer Science*, pages 210–214. IEEE Computer Society Press.

[Gurevich and Shelah, 1986] Gurevich, Y. and Shelah, S. (1986). Fixed-point extensions of first order logic. *Annals of Pure and Applied Logic*, 32 :265–280.

[Gödel, 1931] Gödel, K. (1931). Über formal unentscheidbare Satze der Principia Mathematica und verwandter Systeme I. *Monatschefte fur Mathematik und Physik*, 38 :173–198. English translation in [Davis, 1965].

[Head, 1987] Head, T. (1987). Formal language theory and DNA : An analysis of the generative capacity of specific recombinant behaviors. *Bulletin of Mathematical Biology*, 49 :737–759.

[Helmke and Moore, 1994] Helmke, U. and Moore, J. (1994). *Optimization and Dynamical Systems*. Communications and Control Engineering Series. Springer Verlag, London.

[Henkel, 2001] Henkel, M. (2001). Sur la solution de sundman du probleme des trois corps. *Philosophia Scientiae*, 5 :161.

[Henzinger and Raskin, 1999] Henzinger, T. and Raskin, J.-F. (1999). Robust undecidability of timed and hybrid systems. In Vaandrager, F. and van Schuppen, J. H., editors, *Hybrid systems : computation and control ; Second International Workshop, HSCC'99, Berg en Dal, The Netherlands, March 29–31, 1999 ; proceedings*, volume 1569 of *Lecture Notes in Computer Science*, New York, NY, USA. Springer-Verlag Inc.

[Henzinger et al., 1998] Henzinger, T. A., Kopke, P. W., Puri, A., and Varaiya, P. (1998). What's decidable about hybrid automata ? *Journal of Computer and System Sciences*, 57(1) :94–124.

[Hethcote, 2000] Hethcote, H. W. (2000). The mathematics of infectious diseases. *SIAM Review*, 42(4) :599–653.

[Hirsch et al., 2003] Hirsch, M. W., Smale, S., and Devaney, R. (2003). *Differential Equations, Dynamical Systems, and an Introduction to Chaos*. Elsevier Academic Press.

[Hofbauer and Sigmund, 2003] Hofbauer, J. and Sigmund, K. (2003). Evolutionary game dynamics. *Bulletin of the American Mathematical Society*, 4 :479–519.

[Hofmann, 1999] Hofmann, M. (1999). Type systems for polynomial-time computation. Habilitation.

[Hogarth, 1994] Hogarth, M. (1994). Non-turing computers and non-turing computability. In *Proceedings of the Philosophy of Science Association (PSA'94)*, volume 1, pages 126–138.

[Hogarth, 1996] Hogarth, M. (1996). *Predictability, Computability and Spacetime*. PhD thesis, Sidney Sussex College, Cambridge.

[Hogarth, 2001] Hogarth, M. (2001). Deciding arithmetic in malament-hogarth spacetimes. Available for download on www.hypercomputation.net.

[Hogarth, 1992] Hogarth, M. L. (1992). Does general relativity allow an observer to view an eternity in a finite time ? *Foundations of Physics Letters*, 5 :173–181.

[Hopfield, 1984] Hopfield, J. J. (1984). Neural networks with graded responses have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences of the United States of America*, 81 :3088–3092.

[Hopfield and Tank, 1985] Hopfield, J. J. and Tank, D. W. (1985). 'Neural' computation of decisions in optimization problems. *Biological Cybernetics*, 52 :141–152.

[Hoyrup, 2006] Hoyrup, M. (2006). Dynamical systems : stability and simulability. Technical report, Département d'Informatique, ENS Paris.

[Hölder, 1887] Hölder, O. (1887). Über die eigenschaft der gamma funktion keiner algebraische differentialgleichung zu genügen. *Math. Ann.*, 28 :1–13.

[Immerman, 1986] Immerman, N. (1986). Relational queries computable in polynomial time. *Information and Control*, 68(1–3) :86–104.

[Immerman, 1987] Immerman, N. (1987). Languages that capture complexity classes. *SIAM Journal of Computing*, 16(4) :760–778.

[Immerman, 1991] Immerman, N. (1991). $DSPACE[n^k] = VAR[k+1]$. In Balcázar, José ; Borodin, Alan ; Gasarch, Bill ; Immerman, Neil ; Papadimitriou, Christos ; Ruzzo, Walter ; Vitányi, Paul ; Wilson, C., editor, *Proceedings of the 6th Annual Conference on Structure in Complexity Theory (SCTC '91)*, pages 334–340, Chicago, IL, USA. IEEE Computer Society Press.

[Immerman, 1999] Immerman, N. (1999). *Descriptive Complexity*. Springer.

[Jones, 2001] Jones, N. D. (2001). The expressive power of higher-order types or, life without CONS. *Journal of Functionnal Programming*, 11(1) :5–94.

[Kalmár, 1943] Kalmár, L. (1943). Egyszerü példa eldönthetetlen aritmetikai problémára. *Mate és fizikai lapok*, 50 :1–23.

[Kawamura, 2005] Kawamura, A. (2005). Type-2 computability and Moore's recursive functions. In Brattka, V., Staiger, L., and Weihrauch, K., editors, *Proceedings of the 6th Workshop on Computability and Complexity in Analysis*, volume 120 of *Electronic Notes in Theoretical Computer Science*, pages 83–95, Amsterdam. Elsevier. 6th International Workshop, CCA 2004, Wittenberg, Germany, August 16–20, 2004.

[Kempe, 1876] Kempe, A. (1876). On a general method of describing plane curves of the n–th degree by linkwork. *Proceedings of the London Mathematical Society*, 7 :213–216.

[Kephart and White, 1991] Kephart, J. O. and White, S. R. (1991). Directed-graph epidemiological models of computer viruses. In *Proceedings of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy (SSP '91)*, pages 343–361, Washington - Brussels - Tokyo. IEEE Computer Society Press.

[Kieu, 2004] Kieu, T. D. (2004). Hypercomputation with quantum adiabatic processes. *Theoretical Computer Science*, 317(1-3) :93–104.

[Kitano, 2001] Kitano, H., editor (2001). *Foundations of system biology*. MIT Press.

[Kleene, 1936] Kleene, S. C. (1936). General recursive functions of natural numbers. *Mathematical Annals*, 112 :727–742. Also in [Davis, 1965].

[Ko, 1983] Ko, K.-I. (1983). On the computational complexity of ordinary differential equations. *Information and Control*, 58(1-3) :157–194.

[Ko, 1991] Ko, K.-I. (1991). *Complexity Theory of Real Functions*. Progress in Theoretical Computer Science. Birkhäuser, Boston.

[Koiran, 2001] Koiran, P. (2001). The topological entropy of iterated piecewise affine maps is uncomputable. *Discrete Mathematics & Theoretical Computer Science*, 4(2) :351–356.

[Koiran et al., 1994] Koiran, P., Cosnard, M., and Garzon, M. (1994). Computability with low-dimensional dynamical systems. *Theoretical Computer Science*, 132(1-2) :113–128.

[Koiran and Moore, 1999] Koiran, P. and Moore, C. (1999). Closed-form analytic maps in one and two dimensions can simulate universal Turing machines. *Theoretical Computer Science*, 210(1) :217–223.

[Krivine et al., 2006] Krivine, H., Lesne, A., and Treiner, J. (2006). Discrete-time and continuous-time modelig : some bridges and gaps. *Mathematical Structures in Computer Science*. In print.

[Kurganskyy and Potapov, 2005] Kurganskyy, O. and Potapov, I. (2005). Computation in one-dimensional piecewise maps and planar pseudo-billiard systems. In Calude, C., Dinneen, M. J., Paun, G., Pérez-Jiménez, M. J., and Rozenberg, G., editors, *Unconventional Computation, 4th International Conference, UC 2005, Sevilla, Spain, October 3-7, 2005, Proceedings*, volume 3699 of *Lecture Notes in Computer Science*, pages 169–175. Springer.

[Labbani, 2003] Labbani, O. (2003). Comparaison des théories des jeux pour l'étude du comportement d'agents. Master's thesis, Université de Lille I.

[Lacombe, 1955] Lacombe, D. (1955). Extension de la notion de fonction récursive aux fonctions d'une ou plusieurs variables réelles iii. *Comptes Rendus de l'Académie des Sciences Paris*, 241 :151–153.

[Legenstein and Maass, 2005] Legenstein, R. and Maass, W. (2005). What makes a dynamical system computationally powerful ? In Haykin, S., Principe, J. C., Sejnowski, T., and McWhirter, J., editors, *New Directions in Statistical Signal Processing : From Systems to Brain*. MIT Press. to appear.

[Leivant, 1990] Leivant, D. (1990). Inductive definitions over finite structures. *Information and Computation*, 89(2) :95–108.

[Leivant and Marion, 1995] Leivant, D. and Marion, J.-Y. (1995). Ramified recurrence and computational complexity II : substitution and poly-space. In Pacholski, L. and Tiuryn, J., editors, *Computer Science Logic, 8th Workshop, CSL '94*, volume 933 of *Lecture Notes in Computer Science*, pages 486–500, Kazimierz,Poland. Springer.

[Lincoln and Tiwari, 2004] Lincoln, P. and Tiwari, A. (2004). Symbolic systems biology : Hybrid modeling and analysis of biological networks. In Alur, R. and Pappas, G. J., editors, *Hybrid Systems : Computation and Control, 7th International Workshop, HSCC 2004, Philadelphia, PA, USA, March 25-27, 2004, Proceedings*, volume 2993 of *Lecture Notes in Computer Science*, pages 660–672. Springer.

[Lipshitz and Rubel, 1987] Lipshitz, L. and Rubel, L. A. (1987). A differentially algebraic replacement theorem, and analog computability. *Proceedings of the American Mathematical Society*, 99(2) :367–372.

[Lipton, 1994] Lipton, R. J. (1994). DNA solution of hard computational problems. *Science*, pages 542–545.

[Lorenz, 1963] Lorenz, E. N. (1963). Deterministic nonperiodic flow. *Journal of the Atmospheric Science*, 20 :130–141.

[Lotka, 1920] Lotka, A. (1920). Analytical note on certain rhytmic relations in organic systems. *Proceedings of the National Academy of Science, USA*, 6 :410–415.

[Lynch, 1997] Lynch, N. (1997). *Distributed Algorithms*. Morgan Kaufmann Publishers, Inc.

[Maass, 1996a] Maass, W. (1996a). Lower bounds for the computational power of networks of spiking neurons. *Neural Computation*, 8(1) :1–40.

[Maass, 1996b] Maass, W. (1996b). On the computational power of noisy spiking neurons. In Touretzky, D., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems*, volume 8, pages 211–217. MIT Press (Cambridge).

[Maass, 1997a] Maass, W. (1997a). A model for fast analog computations with noisy spiking neurons. In Bower, J., editor, *Computational Neuroscience : Trends in research*, pages 123–127.

[Maass, 1997b] Maass, W. (1997b). Networks of spiking neurons : the third generation of neural network models. *Neural Networks*, 10 :1659–1671.

[Maass, 1999] Maass, W. (1999). Computing with spiking neurons. In Maass, W. and Bishop, C. M., editors, *Pulsed Neural Networks*, pages 55–85. MIT Press (Cambridge).

[Maass, 2002] Maass, W. (2002). Computing with spikes. *Special Issue on Foundations of Information Processing of TELEMATIK*, 8(1) :32–36.

[Maass, 2003] Maass, W. (2003). Computation with spiking neurons. In Arbib, M. A., editor, *The Handbook of Brain Theory and Neural Networks*, pages 1080–1083. MIT Press (Cambridge), 2nd edition.

[Maass and Bishop, 1998] Maass, W. and Bishop, C. (1998). *Pulsed Neural Networks*. Cambridge MA. MIT Press.

[Maass and Natschläger, 2000] Maass, W. and Natschläger, T. (2000). A model for fast analog computation based on unreliable synapses. *Neural Computation*, 12(7) :1679–1704.

[Maass and Orponen, 1998] Maass, W. and Orponen, P. (1998). On the effect of analog noise in discrete-time analog computations. *Neural Computation*, 10(5) :1071–1095.

[Maass and Ruf, 1999] Maass, W. and Ruf, B. (1999). On computation with pulses. *Information and Computation*, 148(2) :202–218.

[Maass and Sontag, 1999] Maass, W. and Sontag, E. (1999). Analog neural nets with gaussian or other common noise distributions cannot recognize arbitrary regular languages. *Neural Computation*, 11(3) :771–782.

[MacLennan, 2001] MacLennan, B. J. (2001). Can differential equation compute. Technical report, Computer Science Department, University of Tennesse, Knoxville.

[May, 1976] May, R. M. (1976). Simple mathematical models with very complicated dynamics. *Nature*, 261 :459–467.

[May et al., 1995] May, R. M., Bohoeffer, S., and Nowak, M. A. (1995). Spatial games and evolution of cooperation. In Morán, F., Moreno, A., Merelo, J. J., and Chacón, P., editors, *Proceedings of the Third European Conference on Artificial Life : Advances in Artificial Life*, volume 929 of *Lecture Notes in Artificial Intelligence*, pages 749–759, Berlin. Springer Verlag.

[Maynard-Smith, 1981] Maynard-Smith, J. (1981). *Evolution and the Theory of Games*. Cambridge University Press, Cambridge.

[McKelvey and McLennan, 1996] McKelvey, R. and McLennan, A. (1996). Computation of equilibria in finite games. In *Handbook of Computational Economics*. Elsevier.

[McMillan, 1994] McMillan, J. (1994). Selling spectrum rights. *Journal of Economic Perspectives*, pages 145–162.

[Michaux, 1989] Michaux, C. (1989). Une remarque à propos des machines sur $\mathbb{R}$ introduites par Blum, Shub et Smale. *Comptes Rendus de l'Académie des Sciences de Paris*, 309 :435–437.

[Mills, 1995] Mills, J. (1995). Programmable VLSI extended analog computer for cyclotron beam control. Technical Report 441, Indiana University Computer Science.

[Mills et al., 2005] Mills, J. W., Himebaugh, B., Allred, A., Bulwinkle, D., Deckard, N., Gopalakrishnan, N., Miller, J., Miller, T., Nagai, K., Nakamura, J., Ololoweye, B., Vlas, R., Whitener, P., Ye, M., , and Zhang, C. (2005). Extended analog computers : A unifying paradigm for VLSI, plastic and colloidal computing systems. In *Workshop on Unique Chips and Systems (UCAS-1). Held in conjunction with IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS05)*, Austin, Texas.

[Minsky, 1967] Minsky, M. L. (1967). *Computation : Finite and infinite machines*. Prentice-Hall, Englewood Cliffs.

[Moore, 1990] Moore, C. (1990). Unpredictability and undecidability in dynamical systems. *Physical Review Letters*, 64(20) :2354–2357.

[Moore, 1991] Moore, C. (1991). Generalized shifts : unpredictability and undecidability in dynamical systems. *Nonlinearity*, 4(3) :199–230.

[Moore, 1996] Moore, C. (1996). Recursion theory on the reals and continuous-time computation. *Theoretical Computer Science*, 162(1) :23–44.

[Moore, 1998a] Moore, C. (1998a). Dynamical recognizers : real-time language recognition by analog computers. *Theoretical Computer Science*, 201(1–2) :99–136.

[Moore, 1998b] Moore, C. (1998b). Finite-dimensional analog computers : Flows, maps, and recurrent neural networks. In Calude, C. S., Casti, J. L., and Dinneen, M. J., editors, *Unconventional Models of Computation (UMC'98)*, Lecture Notes in Computer Science. Springer.

[Moschovakis, 1984] Moschovakis, Y. N. (1984). Abstract recursion as a foundation for the theory of algorithms. In *Computation and Proof Theory*, volume 1104 of *Lecture Notes in Mathematics*, pages 289–364, Berlin. Springer-Verlag.

[Murray, 2002] Murray, J. D. (2002). Mathematical biology. i : An introduction. In *Biomathematics*, volume 17, pages xiv + 767. Springer Verlag, third edition.

[Murray, 2003] Murray, J. D. (2003). Mathematical biology. ii : Spatial models and biomedical applications. In *Biomathematics*, volume 18, pages xiv + 767. Springer Verlag, third edition.

[Mycka, 2003a] Mycka, J. (2003a). Infinite limits and r-recursive functions. *Acta Cybernetica*, 16 :83–91.

[Mycka, 2003b] Mycka, J. (2003b). μ-recursion and infinite limits. *Theoretical Computer Science*, 302 :123–133.

[Mycka and Costa, 2004] Mycka, J. and Costa, J. F. (2004). Real recursive functions and their hierarchy. *Journal of Complexity*, 20(6) :835–857.

[Mycka and Costa, 2005] Mycka, J. and Costa, J. F. (2005). What lies beyond the mountains, computational systems beyond the Turing limit. *European Association for Theoretical Computer Science Bulletin*, 85 :181–189.

[Mycka and Costa, 2006a] Mycka, J. and Costa, J. F. (2006a). Analog computation and beyond. Submitted.

[Mycka and Costa, 2006b] Mycka, J. and Costa, J. F. (2006b). The $P \neq NP$ conjecture. Submitted.

[Mycka and Costa, 2006c] Mycka, J. and Costa, J. F. (2006c). The $P \neq NP$ conjecture in the context of real and complex analysis. *Journal of Complexity*, 22(2) :287–303.

[Müller and Moiske, 1993] Müller, N. and Moiske, B. (1993). Solving initial value problems in polynomial time. In *Proc. 22 JAIIO - PANEL '93, Part 2*, pages 283–293.

[Nash, 1950] Nash, J. F. (1950). Equilibrium points in n-person games. *Proc. of the National Academy of Sciences*, 36 :48–49.

[Natschläger and Maass, 2002] Natschläger, T. and Maass, W. (2002). Spiking neurons and the induction of finite state machines. *Theoretical Computer Science : Special Issue on Natural Computing*, 287(1) :251–265.

[Nicollin et al., 1993] Nicollin, X., Olivero, A., Sifakis, J., and Yovine, S. (1993). An approach to the description and analysis of hybrid systems. In Grossman, R. L., Nerode, A., Ravn, A. P., and Rischel, H., editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 149–178. Springer-Verlag.

[Nisan and Ronen, 1999] Nisan, N. and Ronen, A. (1999). Algorithmic mechanism design (extended abstract). In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 129–140. ACM Press.

[Nowak and May, 1992] Nowak, M. A. and May, R. M. (1992). Evolutionary games and spatial chaos. *Nature*, 359(6398) :826–829.

[Odifreddi, 1992] Odifreddi, P. (1992). *Classical Recursion Theory*, volume 125 of *Studies in Logic and the foundations of mathematics*. North-Holland.

[Omohundro, 1984] Omohundro, S. (1984). Modelling cellular automata with partial differential equations. *Physica D*, 10D(1–2) :128–134.

[Ord, 2006] Ord, T. (2006). The many forms of hypercomputation. *Applied Mathematics and Computation*, 178(1) :143–153.

[Orponen, 1994] Orponen, P. (1994). Computational complexity of neural networks : a survey. *Nordic Journal of Computing*, 1(1) :94–110.

[Orponen, 1996] Orponen, P. (1996). The computational power of discrete hopfield nets with hidden units. *Neural Computation*, 8(2) :403–415.

[Orponen, 1997] Orponen, P. (1997). *Algorithms, Languages and Complexity*, chapter A survey of continuous-time computational theory, pages 209–224. Kluwer Academic Publishers.

[Orponen and Sima, 2000] Orponen, P. and Sima (2000). A continuous-time hopfield net simulation of discrete neural networks. In *Proceedings of the 2nd International ICSC Symposium on Neural Computations (NC'2000)*, pages 36–42, Berlin, Germany. Wetaskiwin (Canada) : ICSC Academic Press.

[Osbourne and Rubinstein, 1994] Osbourne and Rubinstein (1994). *A Course in Game Theory*. MIT Press.

[Papadimitriou, 2001] Papadimitriou, C. (2001). Algorithms, games, and the Internet. In ACM, editor, *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing : Hersonissos, Crete, Greece, July 6–8, 2001*, pages 749–753, New York, NY, USA. ACM Press.

[Papadimitriou, 1994] Papadimitriou, C. H. (1994). *Computational Complexity*. Addison-Wesley.

[Păun, 2002] Păun, G. (2002). *Membrane Computing. An Introduction*. Springer-Verlag, Berlin.

[Poizat, 1995] Poizat, B. (1995). *Les petits cailloux*. aléas.

[Post, 1946] Post, E. (1946). A variant of a recursively unsolvable problem. *Bulletin of the American Math. Soc.*, 52 :264–268.

[Pour-El and Zhong, 1997] Pour-El, M. and Zhong, N. (1997). The wave equation with computable initial data whose unique solution is nowhere computable. *Mathematical Logic Quarterly*, 43(4) :499–509.

[Pour-El, 1974] Pour-El, M. B. (1974). 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.

[Pour-El and Richards, 1979] Pour-El, M. B. and Richards, J. I. (1979). A computable ordinary differential equation which possesses no computable solution. *Annals of Mathematical Logic*, 17 :61–90.

[Pour-El and Richards, 1981] Pour-El, M. B. and Richards, J. I. (1981). The wave equation with computable inital data such that its unique solution is not computable. *Advances in Mathematics*, 39 :215–239.

[Pour-El and Richards, 1989] Pour-El, M. B. and Richards, J. I. (1989). *Computability in Analysis and Physics*. Springer-Verlag.

[Puri, 1998] Puri, A. (1998). Dynamical properties of timed automata. In Ravn, A. P. and Rischel, H., editors, *Formal Techniques in Real-Time and Fault-Tolerant Systems, 5th International Symposium, FTRTFT'98, Lyngby, Denmark, September 14-18, 1998, Proceedings*, volume 1486 of *Lecture Notes in Computer Science*, pages 210–227. Springer.

[Puri and Varaiya, 1994] Puri, A. and Varaiya, P. (1994). Decidability of hybrid systems with rectangular differential equations. In *Proceedings of the 6th workshop on Computer-aided verification*, number 818 in Lecture Notes in Computer Science, pages 95–104. Springer-Verlag.

[Rabin, 1963] Rabin, M. O. (1963). Probabilistic automata. *Information and Control*, 6(3) :230–245.

[Rabinovich, 2003] Rabinovich, A. (2003). Automata over continuous time. *Theoretical Computer Science*, 300(1–3) :331–363.

[Ramis et al., 1995] Ramis, E., Deschamp, C., and Odoux, J. (1995). *Cours de Mathématiques Spéciales, Tome 3, Topologie et éléments d'analyse*. Masson.

[Rogers Jr., 1987] Rogers Jr., H. (1987). *Theory of Recursive Functions and Effective Computability*. MIT Press.

[Rose, 1984] Rose, H. (1984). *Subrecursion*. Oxford university press.

[Rubel, 1989] Rubel, L. A. (1989). A survey of transcendentally transcendental functions. *American Mathematical Monthly*, 96(9) :777–788.

[Rubel, 1993] Rubel, L. A. (1993). The extended analog computer. *Advances in Applied Mathematics*, 14 :39–50.

[Ruohonen, 1993] Ruohonen, K. (1993). Undecidability of event detection for ODEs. *Journal of Information Processing and Cybernetics*, 29 :101–113.

[Ruohonen, 1994] Ruohonen, K. (1994). Event detection for ODEs and nonrecursive hierarchies. In Karhumäki, J. and Maurer, H., editors, *Proceedings of the Colloquium in Honor of Arto Salomaa. Results and Trends in Theoretical Computer Science (Graz, Austria, June 10-11, 1994)*, volume 812 of *Lecture Notes in Computer Science*, pages 358–371. Springer-Verlag, Berlin-Heidelberg-New York-London-Paris-Tokyo-Hong Kong-Barcelona-Budapest.

[Ruohonen, 1996] Ruohonen, K. (1996). An effective Cauchy-Peano existence theorem for unique solutions. *International Journal of Foundations of Computer Science*, 7(2) :151–160.

[Ruohonen, 1997a] Ruohonen, K. (1997a). Decidability and complexity of event detection problems for odes. *Complexity*, 2(6) :41–53.

[Ruohonen, 1997b] Ruohonen, K. (1997b). Undecidable event detection problems for ODEs of dimension one and two. *Theoretical Informatics and Applications*, 31(1) :67–79.

[Ruohonen, 2004] Ruohonen, K. (2004). Chomskian hierarchies of families of sets of piecewise continuous functions. *Theory of Computing Systems*, 37(5) :609–638.

[Schelling, 1969] Schelling, T. C. (1969). Models of segregation. *American Economic Review, Papers and Proceeings*, 59 :488–493.

[Schelling, 1971a] Schelling, T. C. (1971a). Dynamic models of segregation. *Journal of Mathematical Sociology*, 1 :143–186.

[Schelling, 1971b] Schelling, T. C. (1971b). On the ecology of micromotives. *The public interest*, 25 :61–98.

[Schelling, 1978] Schelling, T. C. (1978). *Micromotives and Macrobehavior*. Norton, NewYork.

[Serazzi and Zanero, 2003] Serazzi, G. and Zanero, S. (2003). Computer virus propagation models. In Calzarossa, M. and Gelenbe, E., editors, *MASCOTS Tutorials*, volume 2965 of *Lecture Notes in Computer Science*, pages 26–50. Springer.

[Shannon, 1941] Shannon, C. E. (1941). Mathematical theory of the differential analyser. *Journal of Mathematics and Physics MIT*, 20 :337–354.

[Shor, 1994] Shor, P. W. (1994). Algorithms for quantum computation : Discrete logarithms and factoring. In Goldwasser, S., editor, *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, Los Alamitos, CA, USA. IEEE Computer Society Press.

[Siegelmann, 1995] Siegelmann, H. T. (1995). Computation beyond the Turing limit. *Science*, 268 :545–548.

[Siegelmann, 1999] Siegelmann, H. T. (1999). *Neural Networks and Analog Computation - Beyond the Turing Limit*. Birkauser.

[Siegelmann and Fishman, 1998] Siegelmann, H. T. and Fishman, S. (1998). Analog computation with dynamical systems. *Physica D*, 120 :214–235.

[Siegelmann and Sontag, 1994] Siegelmann, H. T. and Sontag, E. D. (1994). Analog computation via neural networks. *Theoretical Computer Science*, 131(2) :331–360.

[Siegelmann and Sontag, 1995] Siegelmann, H. T. and Sontag, E. D. (1995). On the computational power of neural nets. *Journal of Computer and System Sciences*, 50(1) :132–150.

[Sima and Orponen, 2003] Sima and Orponen (2003). Exponential transients in continuous-time liapunov systems. *Theoretical Computer Science*, 306(1–3) :353–372.

[Síma and Orponen, 2003a] Síma, J. and Orponen, P. (2003a). Continuous-time symmetric hopfield nets are computationally universal. *Neural Computation*, 15(3) :693–733.

[Síma and Orponen, 2003b] Síma, J. and Orponen, P. (2003b). General-purpose computation with neural networks : A survey of complexity theoretic results. *Neural Computation*, 15(12) :2727–2778.

[Smith, 1998] Smith, W. D. (1998). Plane mechanisms and the downhill principle.

[Smith, 1999] Smith, W. D. (1999). History of "Church's theses" and a manifesto on converting physics into a rigorous algorithmic discipline. Technical report, NEC Research Institute. Avalaible on *http ://www.math.temple.edu/ wds/homepage/works.html*.

[Smith, 2003] Smith, W. D. (2003). On the uncomputability of hydrodynamics. Technical report, NEC Research Institute. Avalaible on *http ://www.math.temple.edu/ wds/homepage/works.html*.

[Smith, 2004] Smith, W. D. (2004). Church's thesis meets quantum mechanics. Technical report, NEC Research Institute. Avalaible on *http ://www.math.temple.edu/ wds/homepage/works.html*.

[Smith, 2006] Smith, W. D. (2006). Church's thesis meets the N-body problem. *Applied Mathematics and Computation*, 178(1) :154–183.

[Stadigh, 1902] Stadigh, V. E. E. (1902). Ein satz ueber funktionen die algebraische differentialgleichungen befriedigen und ueber die eigenschaft der function $\zeta(s)$ keiner solchen gleichung zu genügen. Thesis, Helsinki.

[Staniford et al., 2002] Staniford, S., Paxson, V., and Weaver, N. (2002). How to own the internet in your spare time. In Boneh, D., editor, *Proceedings of the 11th USENIX Security Symposium, San Francisco, CA, USA, August 5-9, 2002*, pages 149–167. USENIX.

[Stoll and Lee, 1988] Stoll, H. M. and Lee, L. S. (1988). A continuous-time optical neural network. In *IEEE Second International Conference on Neural Networks (2nd ICNN'88)*, volume II, pages 373–384, San Diego, CA. IEEE Society Press.

[Svoboda, 1948] Svoboda, A. (1948). *Computing mechanisms and linkages*. McGraw Hill. Dover reprint 1965.

[Tel, 1994] Tel, G. (1994). *Introduction to Distributed Algortihms*. Cambridge University Press.

[Thomson, 1876] Thomson, W. (1876). On an instrument for calculating the integral of the product of two given functions. In *Proceedings of the Royal Society of London*, volume 24, pages 266–276.

[Trakhtenbrot, 1995] Trakhtenbrot, B. (1995). Origins and metamorphoses of the trinity : Logic, nets, automata. In Kozen, D., editor, *Proceedings of the 10th Annual IEEE Symposium on Logic in Computer Science (San Diego, California, June 26-29, 1995)*, pages 506–507, Los Alamitos-Washington-Brussels-Tokyo. IEEE Computer Society, IEEE Computer Society Press.

[Tripakis, 2003] Tripakis, S. (2003). Folk theorems on the determinization and minimization of timed automata. In Larsen, K. G. and Niebert, P., editors, *Formal Modeling and Analysis of Timed Systems : First International Workshop, FORMATS 2003, Marseille, France, September 6-7, 2003. Revised Papers*, volume 2791 of *Lecture Notes in Computer Science*, pages 182–188. Springer.

[Turing, 1936] Turing, A. (1936). On computable numbers, with an application to the Ëntscheidungsproblem. *Proceedings of the London Mathematical Society*, 42(2) :230–265.

[Vardi, 1982] Vardi, M. Y. (1982). The complexity of relational query languages. In *Proceedings of the 14th ACM Symposium on Theory of Computing (STOC)*, pages 137–146. ACM Press.

[Vergis et al., 1986] Vergis, A., Steiglitz, K., and Dickinson, B. (1986). The complexity of analog computation. *Mathematics and Computers in Simulation*, 28(2) :91–113.

[Volterra, 1931] Volterra, V. (1931). *Leçons sur la théorie mathématique de la lutte pour la vie*. Gauthier-Villars, Paris.

[von Neumann and Morgenstern, 1944] von Neumann, J. and Morgenstern, O. (1944). *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, New Jersey, first edition.

[Weibull, 1995] Weibull, J. W. (1995). *Evolutionary Game Theory*. The MIT Press.

[Weihrauch, 2000] Weihrauch, K. (2000). *Computable Analysis*. Springer.

[Weihrauch and Zhong, 2002] Weihrauch, K. and Zhong, N. (2002). Is wave propagation computable or can wave computers beat the Turing machine ? *Proceedings of the London Mathematical Society*, 85(3) :312–332.

[Williams, 1996] Williams, M. R. (1996). About this issue. *IEEE Annals of the History of Computing*, 18(4).

[Woods and Naughton, 2005] Woods, D. and Naughton, T. J. (2005). An optical model of computation. *Theoretical Computer Science*, 334(1-3) :227–258.

[Yao, 2003] Yao, A. C.-C. (2003). Classical physics and the Church–Turing Thesis. *Journal of the ACM*, 50(1) :100–105.

[Zaikin and Zhabotinsky, 1970] Zaikin, A. N. and Zhabotinsky, A. M. (1970). Concentration wave propagation in two-dimensional liquid phase self oscillating system. *Nature*, 255 :535–537.

[Zhou, 1997] Zhou, Q. (1997). Subclasses of computable real valued functions. *Lecture Notes in Computer Science*, 1276 :156–165.

[Zou et al., 2002] Zou, C. C., Gong, W., and Towsley, D. F. (2002). Code red worm propagation modeling and analysis. In Atluri, V., editor, *Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, Washingtion, DC, USA, November 18-22, 2002*, pages 138–147. ACM Press.

[Zou et al., 2003] Zou, C. C., Towsley, D., and Gong, W. (2003). Email virus propagation modeling and analysis. Technical report, University of Massachusetts, Amherst. Technical report TR-CSE-03-04.

Annexe A

Un point de vue sur les hypercalculs

En 2005, nous avons été invité à exprimer notre point de vue dans un numéro spécial du journal *Applied Mathematics and Computations* sur les *hypercalculs*. On parle d'hypercalcul lorsqu'un dispositif est capable de réaliser des calculs qu'une machine de Turing ne saurait pas faire.

Nous reprenons ici ce point de vue. Ce chapitre constitue une traduction de [Bournez, 2006], légèrement adaptée. Toutes les preuves sont omises. Puisque nous estimons la discussion du chapitre 3 beaucoup plus intéressante, et à jour, que celle qui était présente dans cet article, nous omettons volontairement la dernière section de [Bournez, 2006], relative aux problèmes de robustesse.

A.1 Thèses de Church

Un résultat majeur du vingtième siècle est le théorème d'incomplétude de Gödel [Gödel, 1931], qui prouve qu'aucun système de preuve ne peut capturer notre raisonnement sur les entiers. Les arguments de [Gödel, 1931] sont basés sur une notion informelle de déduction.

Quelque temps après le résultat de Gödel, Alan Turing proposa dans l'article [Turing, 1936] son fameux modèle de machine, capable de capturer la déduction dans les systèmes formels, et en particulier la notion de déduction utilisée par Gödel dans sa preuve.

De façon simplifiée, Turing a prouvé dans [Turing, 1936] le résultat suivant : ce qui peut être calculé par un humain qui travaille mécaniquement avec un papier et un crayon en un nombre fini d'étapes (en particulier, cela couvre la déduction dans les systèmes formels) est calculable par une machine de Turing [Turing, 1936], [Copeland, 2002], [Gandy, 1980].

Très vite, on découvrit[1] que la puissance des machines de Turing pouvait être prouvée égale à celle de plusieurs autres formalismes qui avaient été introduits, dont le lambda-calcul de Alonzo Church [Church, 1936], et les fonctions récursives de Stephen Kleene [Kleene, 1936].

Ces considérations ont mené naissance à la thèse de Church-Turing :

> "Ce qui est effectivement *calculable* est calculable par une machine de Turing."

Dans cette thèse, la première notion de *calculable* fait référence à une notion donnée intuitive, alors que la seconde notion de calculable signifie "calculable par une machine de Turing" [Gandy, 1980], [Copeland, 2002], [Ord, 2006].

Comme l'argumente très justement Jack Copeland dans [Copeland, 2002], la thèse originale fait référence à une notion de calcul, entendue comme celle réalisable, au moins théoriquement, par un humain, qui travaille mécaniquement avec un papier et un crayon, et elle est souvent mal interprétée comme la thèse suivante :

> "Ce qui peut être calculé par une machine est calculable par une machine de Turing."

[1] Comme observé dans [Cleland, 2004], cela peut être considéré comme pas si surprenant, puisque chacun de ces formalismes a été explicitement conçu pour résoudre le problème *Entscheidungsproblem* d'Hilbert : décider si une formule arbitraire du calcul des prédicats est une tautologie.

Cette seconde thèse est appelée thèse M dans [Gandy, 1980], qui la distingue bien de la précédente.

Dans cette dernière, la notion de machine fait référence à une notion intuitive de machine, avec la contrainte que la machine est supposée obéir aux lois physiques[2] du monde réel [Copeland, 2002], car sinon la thèse est connue pour être fausse : voir par exemple tous les contre-exemples dans [Ord, 2006], [Copeland and Sylvan, 1999].

Une variante, proche de cette thèse, aussi discutée dans [Copeland, 2002], est la suivante :

> "Tout processus qui admet une description mathématique peut être simulé par une machine de Turing".

Encore une fois (et pour les mêmes contre-exemples en fait), si le processus n'est pas contraint de pouvoir exister dans le monde réel, la thèse est connue pour être fausse [Copeland, 2002].

Il s'avère que les trois thèses sont complètement indépendantes :

- la première concerne les calculs réalisables par un humain qui travaille mécaniquement avec un papier et un crayon, ou concerne le pouvoir de nos systèmes formels de déduction [Copeland, 2002] ;
- la seconde concerne la physique du monde qui nous entoure [Smith, 1999], [Copeland, 2002], [Yao, 2003] ;
- la troisième concerne les modèles que nous avons du monde qui nous entoure [Smith, 1999], [Copeland, 2002], [Yao, 2003].

Nous pensons que chacune de ces thèses a réellement à voir en fait avec les convictions de chacun, car aucune n'est réellement prouvable, étant donnée que chacune d'entre elles fait soit référence à des notions informelles, soit au monde physique dont nous n'avons pas de modèle[3].

Notons toutefois qu'il y a eu plusieurs tentatives de preuves dans la littérature, se basant sur des hypothèses plus basiques : voir par exemple [Gandy, 1980], [Boker and Dershowitz, 2005].

Ce qui nous semble intéressant est que si l'on prend chacune de ces thèses de façon contraposée, chacune signifie que tout système qui calcule quelque chose qui ne l'est pas par une machine de Turing doit utiliser quelque chose, que nous appellerons *ressource*, qui

- soit n'est pas calculable mécaniquement, pour la première,
- soit n'est pas calculable par une machine physique, pour la seconde,
- soit n'est pas calculable par un modèle de machine physique, pour la troisième.

Qualifions une telle ressource de *"non raisonnable"*.

Il nous semble alors important de discuter ce qui fait qu'une ressource peut être non raisonnable, indépendamment de la véracité de chacune des thèses.

A.2 Complexité d'une ressource

Dans ce chapitre, nous allons discuter de la puissance de plusieurs modèles de systèmes dynamiques à temps continu par rapport à la puissance des machines de Turing.

Nous allons considérer plusieurs variantes de systèmes, en fonction d'hypothèses faites sur leur "raisonnabilité", et nous allons chercher à comprendre systématiquement la puissance obtenue par rapport aux classes de calculabilité et de complexité classiques.

Comme argumentent José Félix Costa et Jerzy Mycka dans leur article [Mycka and Costa, 2005], ce qui manque est une notion claire et bien comprise du degré de raisonnable d'une ressource. En l'absence, d'une telle notion, nous discuterons plusieurs critères permettant ou non de garantir certains faits.

Nous voudrions ajouter que notre motivation et notre discussion à propos de la complexité des ressources impliquées est très proche d'une des motivations de José Félix Costa et de Jerzy Mycka pour étudier les calculs analogiques dans leur série d'articles, exprimée explicitement dans [Mycka and Costa, 2005].

Nous voulons aussi ajouter que nous ne prétendons pas que les modèles considérés ont la moindre réalité physique. Nous nous focalisons essentiellement sur ces modèles parce que ce sont des modèles qui ont déjà été considérés et proposés dans la littérature, à propos (d'idéalisations abstraites) de systèmes concrets de notre

[2] Sinon à ses contraintes sur les ressources.

[3] Observons que dès que l'on croit en l'existence de concepts comme les entiers, le théorème de Gödel dit précisément qu'il n'est pas possible d'avoir un modèle complet.

monde, et surtout puisque nous pensons qu'ils sont informatifs à propos des ressources "non raisonnables" dans notre monde.

La plupart des modèles que nous considérons sont clairement non réalistes, ou impliquent des éléments non-calculables, mais nous pensons que, même si nous croyons les thèses (ou un sous-ensemble des thèses) vraies, refuser de discuter de tels modèles est seulement refuser de discuter des raisons pour lesquelles on peut penser que les thèses sont vraies.

Plusieurs articles, en particulier de personnes argumentant contre les hypercalculs, ont défendu le point de vue que discuter de certains systèmes dans des théories physiques permettant des hypercalculs aide à comprendre les faiblesses des modèles physiques de notre monde [Aaronson, 2005], [Smith, 2006], [Smith, 2003], [Smith, 2004].

Notre but est en quelque sorte un point de vue parallèle, d'informaticien : discuter des modèles théoriques qui sont capables de réaliser des hypercalculs aide à comprendre les faiblesses des modèles de l'informatique théorique.

A.3 Préliminaires mathématiques

Un demi-espace ouvert (respectivement fermé) est l'ensemble des points $\mathbf{x}$, qui satisfont $\mathbf{a}.\mathbf{x} < b$ (resp. $\mathbf{a}.\mathbf{x} \leq b$), pour un certain $\mathbf{a} \in \mathbb{R}^d$, $b \in \mathbb{R}$, où . désigne le produit scalaire. Il sera dit rationnel si en outre $\mathbf{a} \in \mathbb{Q}^d$, $b \in \mathbb{Q}$.

Un polyèdre P est une combinaison booléenne (unions, intersections) de demi-espaces ouverts ou fermés. Il est dit rationnel si les demi-espaces le sont. $\|.\|$ désignera la norme *sup*.

Un concept très important dans notre travail, et récurrent, est celui de système dynamique.

Dans ce chapitre, nous prendrons la définition suivante.

Définition 37 (Système dynamique) – *Un système dynamique $\mathcal{H}$ est donné par $X \subset \mathbb{R}^d$, et une fonction $f : X \to X$.*

– *Une trajectoire de $\mathcal{H}$ partant de $\mathbf{x}_0 \in X$, est une solution de l'équation différentielle*

$$\begin{cases} \dot{\mathbf{x}} & = & f(\mathbf{x}) \\ \mathbf{x}(0) & = & \mathbf{x}_0. \end{cases}$$

Autrement dit, une fonction dérivable $\phi : \mathbb{R}^+ \to X$, avec $\phi(0) = \mathbf{x}_0$, et $\frac{d\phi}{dt}(t) = f(\phi(t))$ pour tout t.

Étant donnée une propriété des fonctions, nous dirons qu'un système dynamique possède cette propriété si la fonction correspondante f la possède. Par exemple, les systèmes dynamiques à temps continu dérivables correspondent à la classe de systèmes dynamiques $\mathcal{H} = (X, f)$, où f est dérivable.

A.4 Mesure de la complexité d'une fonction

A.4.1 Fonctions lisses

Il existe plusieurs façons de mesurer la complexité d'une fonction f. Une première façon est de parler de sa rugosité : une fonction $f : X \subset \mathbb{R}^d \to \mathbb{R}^d$ est dite de classe $\mathcal{C}^\infty$, si elle est r-fois dérivable pour tout $r \in \mathbb{N}$. Les fonctions de classes $\mathcal{C}^\infty$ contiennent les fonctions analytiques, c'est-à-dire les fonctions qui sont égales à leur développement de Taylor dans un voisinage de tout point.

A.4.2 Analyse récursive

Une autre possibilité est de parler des propriétés de la fonction dans le modèle de l'analyse récursive : voir [Weihrauch, 2000] pour une présentation récente de l'analyse récursive d'un point de vue "calculabilité" ou [Ko, 1991] pour une présentation avec un point de vue "complexité".

Suivant [Ko, 1991], soit $\nu_{\mathbb{Q}} : \mathbb{N} \to \mathbb{Q}$ la représentation suivante[4] des nombres dyadiques par les entiers :

$$\nu_{\mathbb{Q}}(\langle p, q, q\rangle) \mapsto \frac{p-q}{2^r},$$

où $\langle ., ., .\rangle : \mathbb{N}^3 \to \mathbb{N}$ est une bijection calculable en temps polynomial.

Une suite d'entiers $(x_i) \in \mathbb{N}^{\mathbb{N}}$ converge rapidement vers x, noté par

$$(x_i) \rightsquigarrow x,$$

si la propriété suivante est vérifiée pour tout i :

$$|\nu_{\mathbb{Q}}(x_i) - x| < \exp(-i).$$

Un point $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$ est dit calculable, noté par $\mathbf{x} \in \mathcal{R}ec(\mathbb{R})$, si pour tout j, il existe une suite calculable $(x_i)_{i\in\mathbb{N}}$ avec $(x_i) \rightsquigarrow x_j$. Il est dit calculable en temps polynomial, noté $\mathbf{x} \in P(\mathbb{R})$, si les suites correspondantes le sont.

Une fonction $f : X \subset \mathbb{R}^d \to \mathbb{R}$, où X est compact, est dite calculable, ce qui sera noté $f \in \mathcal{R}ec(\mathbb{R})$, s'il existe une machine de Turing M avec d-oracles telle que pour tout $\mathbf{x} = (x_1, \ldots, x_d) \in X$, pour toute suites $(x_i^j) \rightsquigarrow x_j$, M prenant comme oracle ces d suites, calcule une suite (x'_i) avec $(x'_i) \rightsquigarrow f(\mathbf{x})$.

Une fonction $f : X \subset \mathbb{R}^d \to \mathbb{R}^d$, où X est compact, est dite calculable si toutes ses projections le sont. Elle est dite calculable en temps polynomial, noté $f \in P(\mathbb{R})$, si la machine de Turing en question fonctionne en temps polynomial.

A.4.3 Classes de fonctions à la Kleene

Une autre approche pour mesurer la complexité d'une fonction est celle initiée par [Moore, 1996]. Elle consiste à discuter l'appartenance de la fonction à certaines classes de fonctions définie algébriquement à partir d'un ensemble fini de fonctions de base, et closes par des opérations simples : voir par exemple les articles [Mycka and Costa, 2006c], [Mycka and Costa, 2004], [Graça, 2004], [Campagnolo, 2004b] ou le chapitre 4.

Nous ne suivrons pas cette approche dans ce chapitre, même si elle pourrait paraître naturelle en rapport avec certains autres chapitres.

A.5 Systèmes dynamiques en tant que modèles de calculs

Dans ce chapitre, on considérera les systèmes dynamiques comme des reconnaisseurs de langages : Σ dénotera l'alphabet $\{0, 1\}$, et Σ^* dénotera les mots sur cet alphabet.

Deux codages très classiques des mots dans les réels joueront un rôle important :

– ν_X est la fonction qui envoie Σ^* sur $[0, 1]$ comme suit : le mot $w = w_1 \ldots w_n \in \{0, 1\}^*$ est envoyé sur

$$\nu_X(w) = \sum_{i=1}^{n} \frac{(2w_i + 1)}{4^i}.$$

(il s'agit d'une représentation "Cantorienne".)

– $\nu_{\mathbb{N}}$ est la fonction qui envoie Σ^* sur $\mathbb{N}$ comme suit : le mot $w = w_1 \ldots w_n \in \{0, 1\}^*$ est envoyé sur

$$\nu_{\mathbb{N}}(w) = \sum_{i=1}^{n} (2w_i + 1)4^i.$$

On peut alors définir :

[4] Plusieurs autres représentations naturelles peuvent être choisies à la place de celle-ci et donnent la même classe de fonctions calculables : voir [Weihrauch, 2000, Ko, 1991].

Définition 38 (Systèmes dynamiques comme reconaisseurs de langages) *Soit $\mathcal{H}$ un système dynamique à temps continu sur l'espace X. On considérera deux cas :*

1. *$X = [-1, 1]^d$ (cas compact),*
2. *ou $X = \mathbb{R}^d$ (cas général).*

Considérons $\nu = \nu_X$ pour le premier, et $\nu = \nu_{\mathbb{N}}$ pour le second.

Soit V_{accept} l'ensemble des $\mathbf{x} \in X$ avec $\|\mathbf{x}\| \leq 1/4$.

Soit $V_{compute}$ l'ensemble des $\mathbf{x} \in X$ avec $\|\mathbf{x}\| \geq 1/2$.

On dira que $\mathcal{H}$ calcule un langage $L \subset \Sigma^*$, sur l'alphabet $\Sigma = \{0, 1\}$, si pour tout $w \in \Sigma^*$, $w \in L$ si et seulement si la trajectoire de $\mathcal{H}$ partant de

$$(\nu(w), 0, \cdots, 0, 1)$$

atteint V_{accept}.

Pour des raisons de robustesse, on supposera que, pour tout $w \notin L$, la trajectoire correspondante reste à jamais dans $V_{compute}$.

Étant donnée une notion de temps associée aux trajectoires, on dira que L est reconnu en temps T si en outre, lorsque la trajectoire atteint V_{accept}, cela se fait à un temps borné supérieurement par T. Elle sera dite calculée en temps $f : \mathbb{N} \to \mathbb{N}$, si en outre, $T(w) \leq f(|w|)$, pour tout w, où $|w|$ désigne la longueur du mot w.

A.6 Un exemple jouet

Nous allons discuter l'exemple des systèmes à dérivée constante par morceaux, PCD, pour Piecewise Constant Derivative Systems, introduit par Eugène Asarin, Oded Maler et Amir Pnueli dans [Asarin et al., 1995], comme un modèle simple de systèmes hybrides. Il a été ensuite discuté dans plusieurs articles comme [Asarin and Bouajjani, 2001], [Asarin and Maler, 1998], [Bournez, 1999b].

Un système hybride est un système qui combine des évolutions continues avec des transitions discrètes. De tels modèles apparaissent dès qu'on essaie de modéliser des systèmes où un système discret, comme un calculateur ou un ordinateur classique, évolue dans un environnement continu : voir par exemple [Antsaklis, 2000].

D'un point de vue d'informaticien théorique, un intérêt des modèles des systèmes hybrides est qu'ils généralisent à la fois les systèmes à transitions discrètes et les systèmes dynamiques à temps continu.

Définition 39 (PCD System [Asarin and Maler, 1998]) *Un système (rationnel) à dérivée constante par morceaux (PCD) est un système dynamique à temps continu $\mathcal{H}$, défini par une équation différentielle*

$$\dot{\mathbf{x}} = f(\mathbf{x})$$

sur $X \subset \mathbb{R}^d$, où $f : X \to \mathbb{R}^d$, peut se représenter par la formule

$$f(\mathbf{x}) = \boldsymbol{c}_i \text{ for } \mathbf{x} \in P_i,\ i = 1, \ldots, n$$

où $c_i \in \mathbb{Q}^d$, et les P_i, constituent une partition de X en polyèdres rationnels.

Une trajectoire de $\mathcal{H}$ partant du point $\mathbf{x}_0 \in X$ est une solution de l'équation différentielle $\dot{\mathbf{x}} = f(\mathbf{x})$ avec la condition initiale $\mathbf{x}(0) = \mathbf{x}_0$: c'est-à-dire une fonction continue $\phi : \mathbb{R}^+ \to X$ telle que $\phi(0) = \mathbf{x}_0$, et telle que pour tout t, $f(\phi(t))$ est égal à la *dérivée à droite* de $\phi(t)$.

En d'autres termes, un système PCD consiste à partitionner l'espace en des sous-ensembles polyèdraux (régions), et à affecter une dérivée constante à tous les points qui partagent la même région.

Les trajectoires de tels systèmes sont des lignes brisées, avec les points de brisure sur les frontières des régions [Asarin et al., 1995] : voir la figure A.1.

Eugène Asarin, Oded Maler, et Amir Pnueli ont montré que les systèmes PCD pouvaient simuler les machines de Turing, dès que l'on suppose la dimension $d \geq 3$ [Asarin et al., 1995]. Réciproquement, les systèmes PCD peuvent être simulés par machines de Turing.

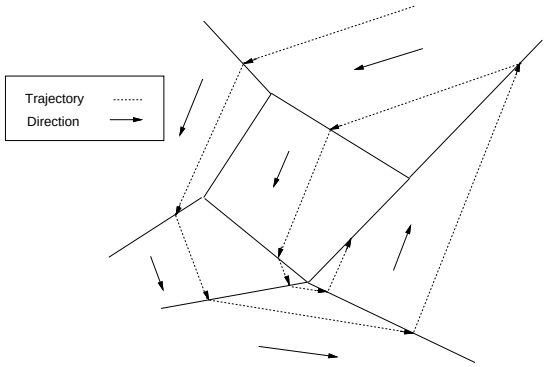


FIG. A.1 – Un système PCD en dimension 2.

Théorème 14 (Systèmes PCD = Turing [Asarin et al., 1995]) *1. Tout langage récursivement énumérable L peut être calculé par un système PCD (rationnel) $\mathcal{H}$ sur $[-1,1]^3$.*

2. Cela ne peut pas être le cas sur $[-1,1]^2$, ni $\mathbb{R}^2$, dans le cas général.

A.7 Sur la rugosité

Il peut être objecté que les systèmes à dérivée constante par morceaux utilisent des fonctions discontinues, ce qui constitue quelque chose de "non raisonnable", et que le théorème 14 ne concerne donc pas des fonctions "réalistes".

En fait, il peut être renforcé comme suit (dans [Bournez and Cosnard, 1996], une preuve alternative à [Moore, 1990] est proposée).

Théorème 15 (Systèmes lisses $\geq$ Turing [Moore, 1990]) *Tout langage récursivement énumérable L peut être calculé par un système dynamique à temps continu $\mathcal{C}^\infty$ (et $\mathcal{R}ec(\mathbb{R})$) $\mathcal{H}$ sur $[-1,1]^3$.*

Il est connu qu'il existe des équations différentielles, avec des coefficients calculables, des conditions initiales calculables, qui ne peuvent pas être résolues par aucune méthode déterministe par un ordinateur digital. Un tel exemple a été montré par Marian Pour-El et Ian Richards dans [Pour-El and Richards, 1979] : il existe une fonction calculable en temps polynomial $f : [0,1] \times [-1,1] \to \mathbb{R}$ telle que l'équation $x' = f(t,x)$ définie par f, n'aie pas de solution calculable y sur $[0,\delta]$, pour tout $\delta > 0$.

Les mêmes auteurs ont ensuite étendu dans [Pour-El and Richards, 1981] leur résultat pour montrer que l'équation d'onde (qui est une équation aux dérivées partielles), même avec des conditions initiales calculables, peut avoir une solution qui n'est pas calculable.

Cependant, si une équation différentielle sur un compact possède une unique solution, alors celle-ci est calculable : voir par exemple [Ko, 1991]. Cela est vrai dès que f est deux fois continûment différentiable.

Corollaire 7 (Systèmes calculables lisses = Turing) *Les systèmes dynamiques à temps continu $\mathcal{C}^\infty \cap \mathcal{R}ec(\mathbb{R})$ sur $[-1,1]^d$ ont précisément la puissance des machines de Turing : ils reconnaissent les langages récursivement énumérables.*

Il est conjecturé dans [Moore, 1998b] qu'aucune fonction analytique sur un espace compact ne peut simuler une machine de Turing, via un codage des entrées et sorties raisonnable. La question de savoir si nous pouvons supposer les systèmes dynamiques analytiques dans le corollaire précédent est à priori une question distincte. Cependant, si nous croyons la conjecture vraie, une réponse négative serait surprenante, puisque la plupart des résultats d'indécidabilité (si l'on met de côté ceux qui sont obtenus par pure diagonalisation) se basent sur une simulation d'une machine de Turing[5].

Si la contrainte d'espace borné est relâchée, il a été récemment obtenu par Daniel Graça, Manuel Campagnolo et Jorge Buescu que les machines de Turing peuvent être simulées par des fonctions analytiques (et en outre d'une façon robuste aux erreurs) [Graça et al., 2005].

Théorème 16 (Systèmes analytiques non-compacts $\geq$ Turing) *Tout langage récursivement énumérable L est calculé par un système dynamique à temps continu analytique (et $\mathcal{R}ec(\mathbb{R})$) sur $\mathbb{R}^7$.*

A.8 Sur la rationalité

Revenons aux systèmes PCD. Supposons que l'on relâche l'hypothèse que les vecteurs $\mathbf{c}_i$ et les polyèdres P_i doivent être rationnels dans la définition 39. Si aucune contrainte n'est mise sur les constantes réelles impliquées, il a été prouvé dans [Bournez and Cosnard, 1996] que tout langage $L \subset \Sigma^*$ pouvait être calculé par un système PCD.

Nous croyons que se restreindre au temps polynomial donne des résultats plus intéressants : le temps discret d'une trajectoire est défini comme le nombre de régions traversées par la trajectoire. Formellement :

Définition 40 (Temps discret) *A chaque trajectoire $\phi : \mathbb{R}^+ \to X$ d'un système PCD $\mathcal{H}$, on peut associer l'ensemble T_ϕ des temps $t_i \geq 0$ auxquels la direction de ϕ change : la dérivée à gauche de ϕ en t_i n'existe pas, ou est distincte de sa dérivée à droite.*

Nous dirons que le temps discret de ϕ est n, si T_ϕ contient n éléments.

Notons qu'il existe aussi une autre notion naturelle de temps pour les systèmes dynamiques à temps continu, et en particulier pour les systèmes PCD.

Définition 41 (Temps continu) *Le temps continu de la trajectoire $\phi(t)$ est la variable t.*

Par exemple, le temps discret de la trajectoire de la figure A.1 vaut 9. Si nous supposons que la norme des vecteurs vitesse est 1 dans chaque région, alors son temps continu est égal à sa longueur.

Rappelons (voir par exemple [Balcázar et al., 1988], [Papadimitriou, 1994]) qu'une famille de circuits booléens $\mathcal{C} = (C_i)_{i\in\mathbb{N}}$, avec C_i possédant i entrées et une sortie, reconnaît un langage $L \subset \Sigma^*$, si et seulement si pour tout $w \in \Sigma^*$, $w \in L$ si et seulement si $C_{|w|}$ accepte w. Nous noterons $taille(C)$ pour la taille d'un circuit.

Définition 42 (Classe $\mathbb{P}$) *Un langage $L \subset \Sigma^*$ est dans $\mathbb{P}$ si et seulement si L est reconnu par une famille de circuits de taille polynomiale : il existe un polynôme p, avec $taille(C_i) = p(i)$ pour tout i.*

[5] Ou de modèles comme les machines à deux compteurs qui simulent les machines de Turing. On peut aussi parfois utiliser des réductions à des problèmes comme le problème de correspondance de Post qui encode la simulation d'une machine de Turing non-déterministe.

La classe $\mathbb{P}$ est aussi connue sous le nom de $P/poly$, puisqu'elle correspond au temps polynomial avec un conseil polynomial [Balcázar et al., 1988].

Elle contient des langages non-récursifs (et non-récursivement énumérables) [Balcázar et al., 1988].

Elle correspond aussi aux ensembles reconnaissables en temps polynomial avec un oracle *tally* : voir [Balcázar et al., 1988].

Elle est la classe naturelle pour caractériser la puissance de plusieurs classes de systèmes dynamiques à temps et espace continus : voir [Siegelmann, 1995], [Siegelmann, 1999], [Siegelmann and Sontag, 1994].

La classe $\mathbb{P}$ correspond au temps polynomial non uniforme, puisqu'elle peut être obtenue en supprimant la seconde condition dans la caractérisation suivante du temps polynomial : voir [Papadimitriou, 1994].

Proposition 11 (P versus $\mathbb{P}$) *Un langage $L \subset \Sigma^*$ est reconnu en temps polynomial par une machine de Turing si et seulement si*

1. *il est dans $\mathbb{P}$;*
2. *la fonction qui envoie 1^n vers le codage du circuit C_n est calculable en temps polynomial.*

Les résultats suivants sont établis dans [Bournez and Cosnard, 1996].

Théorème 17 (P(PCD Systems) = $\mathbb{P}$ [Bournez and Cosnard, 1996])

- *Tout langage $L \in \mathbb{P}$ est calculé par un système PCD non rationnel $\mathcal{H}$ en temps discret polynomial sur $[-1,1]^3$.*
- *Tout langage L calculé par un système PCD non rationnel $\mathcal{H}$ en temps discret polynomial est dans $\mathbb{P}$.*

Observons que les langages reconnus en temps polynomial par les systèmes PCD rationnels correspondent précisément à la classe P. P désigne, bien entendu, le temps polynomial pour les machines de Turing.

Puisque $\mathbb{P}$ contient des ensembles non-récursivement énumérables, les systèmes PCD non rationnels sont plus puissants que les machines de Turing [Bournez and Cosnard, 1996].

Leur surpuissance vient des constantes non-calculables présentes dans leurs descriptions : étant donné un système PCD $\mathcal{H}$, nous écrivons $Constant(\mathcal{H})$ pour les constantes $\alpha_1, \ldots, \alpha_m$ en nombre fini impliquées dans la description des polyèdres P_i, les constantes $\beta_1, \ldots, \beta_m$ en nombre fini impliquées dans les coordonnées des vecteurs $\mathbf{c}_i$, et pour tous les produits $\alpha_i\beta_j$ en nombre fini.

Définition 43 (Systèmes PCD calculables) *Un système PCD sera dit un PCD à constantes calculables, noté $\mathcal{H} \in \mathcal{R}ec(\mathbb{R})$, si $Constant(\mathcal{H}) \subset \mathcal{R}ec(\mathbb{R})$.*

Nous dirons qu'un langage appartient à P/rec, s'il appartient à $\mathbb{P}$, et si la fonction qui envoie[6] 1^n vers le codage du circuit C_n est calculable (observons que nous ne disons pas calculable en temps polynomial, puisque cela donnerait une définition de P).

On a

$$P/rec = \mathbb{P} \cap \mathcal{R}ec,$$

où $\mathcal{R}ec$ désigne la classe des langages récursifs, et donc, P/rec peut être nommé la partie récursive de $\mathbb{P}$, comme cela est fait dans [Síma and Orponen, 2003b].

Puisqu'il existe des fonctions non-calculables en temps polynomial, le temps polynomial est strictement inclus dans P/rec, lui-même strictement inclus dans $\mathbb{P}$.

On peut prouver : voir [Bournez, 2006].

Théorème 18 (P(Systèmes PCD calculables)= P/rec)

- *Tout langage $L \in P/rec$ est calculé en temps discret polynomial par un système PCD non rationnel $\mathcal{H}$ à constantes calculables sur $[-1,1]^3$.*
- *Tout langage L calculé en temps discret polynomial par un système PCD non rationnel $\mathcal{H}$ à constantes calculables est dans P/rec.*

[6] ou envoie n, cela ne changerait pas la définition.

Puisque P/rec est inclus dans l'ensemble des langages récursifs, nous obtenons.

Corollaire 8 (P(PCD Systems) $\subset$ Recursive) *Tout langage calculé par un système PCD non rationnel $\mathcal{H}$ à constantes calculables en temps discret polynomial est récursif.*

Observons que les arguments de [Bournez, 2006] peuvent aussi êtres généralisés pour donner lieu à une complexité structurelle de la puissance des systèmes PCD en fonction de leurs constantes, similaire[7] à celle obtenue pour les réseaux de neurones dans [Balcázar et al., 1993].

A.9 Retour sur la rugosité

Selon les constructions de [Bournez and Cosnard, 1996], on peut construire un système lisse à partir d'un PCD. Le temps discret du système PCD correspond au temps continu du système lisse obtenu, de telle sorte que l'on arrive à prouver.

Théorème 19 ($\mathbb{P} \subset$ P(Systèmes lisses) [Bournez and Cosnard, 1996])
Tout langage $L \in \mathbb{P}$ peut être calculé par un système dynamique à temps continu $\mathcal{C}^\infty$ $\mathcal{H}$ en temps continu polynomial sur $[-1,1]^3$.

Théorème 20 ($P/rec \subset$ P(Systèmes lisses et calculables))
Tout langage $L \in P/rec$ est calculé par un système dynamique à temps continu $\mathcal{C}^\infty$ $\mathcal{H}$ de $\mathcal{R}ec(\mathbb{R})$ en temps continu polynomial sur $[-1,1]^3$.

Théorème 21 ($P \subset$ P(Systèmes lisses et calculables en temps poly.))
Tout langage $L \in P$ est calculé par un système dynamique à temps continu $\mathcal{C}^\infty$ $\mathcal{H}$ de $P(\mathbb{R})$ en temps continu polynomial sur $[-1,1]^3$.

Réciproquement, une question naturelle est de comprendre s'il est possible d'obtenir des bornes supérieures sur la puissance de calcul des systèmes dynamiques à temps continu en temps continu polynomial.

Tout système suffisamment lisse, défini sur un domaine compact, peut être simulé par une méthode numérique : étant donnés un t, et un n, on peut estimer la position de la trajectoire au temps t avec la précision 2^{-n}.

Le point délicat est que les méthodes usuelles, comme la méthode d'Euler, fonctionnent en un temps qui est proportionnel en une exponentiel en t.

Cela est aussi vrai pour la plupart des méthodes numériques d'ordre fixe : voir par exemple la discussion dans [Smith, 2006].

On peut penser que cela est une limitation intrinsèque des méthodes numériques, et donc que potentiellement, les systèmes dynamiques à temps continu peuvent faire des choses plus rapidement que les machines de Turing : voir par exemple les raisons pour lesquelles Anastasios Vergis, Kenneth Steiglitz, et Bradley Dickinson dans [Vergis et al., 1986] évitent de prendre la valeur du temps comme une ressource naturelle dans leur discussion.

Cependant, Warren Smith a récemment démontré qu'il était possible de prouver que le temps peut être considéré comme une ressource raisonnable, avec des hypothèses additionnelles.

Définition 44 (Variation Poly. Limitée (PLV) [Smith, 2006])
Un système dynamique à temps continu (X, f) est dit être à variations polynomialement limitées s'il est de classe $\mathcal{C}^\infty$, et sur tout intervalle de temps $0 \leq t \leq T$, la valeur absolue de chaque composante de f, de chaque composante de $\phi^{(k)}$ pour une trajectoire ϕ, tout comme la valeur absolue de chaque dérivée partielle de f par rapport à chacun de ses arguments, ayant un degré de différentiation total k, est bornée de façon similaire, par des bornes du type $(kT)^{O(k)}$.

[7] Toutefois différente, puisque le modèle ici n'est pas vraiment équivalent et est plus problématique. Principalement ici, la précision linéaire ne suffit pas.

En utilisant les schémas d'intégration numérique de Runge-Kutta de Butcher, avec un ordre pris comme dépendant linéairement de T, Warren Smith prouve :

Théorème 22 (PLV Implique Simulation Efficace[Smith, 2006])
Tout système dynamique de $P(\mathbb{R})$ à variations polynomialement limitées peut être simulé numériquement efficacement par une machine de Turing.

Étant donnée une condition initiale dans $P(\mathbb{R})$, si on note $\phi(t)$ la trajectoire correspondante, la valeur de $\phi(t)$ à tout temps $0 \leq t \leq T$, peut être calculée avec la précision ϵ désirée, pour tout $\epsilon > 0$, en un temps qui dépend seulement polynomialement en T, et $\min(\epsilon, 1)^{-1/\max(1,T)}$.

Avec ces résultats, il est possible de prouver : voir [Bournez, 2006].

Théorème 23 (P = P(Systèmes PLV Poly. Calculables et Lisses))
Les langages calculés par les systèmes dynamiques à temps continu $\mathcal{H}$ de $P(\mathbb{R})$ à variations polynômialement limitées en temps continu polynomial sur $[-1, 1]^d$ correspondent précisément aux langages de P.

Théorème 24 (P/rec = P(Systèmes PLV Calculables et Lisses))
Les langages calculés par les systèmes dynamiques à temps continu $\mathcal{H}$ de $\mathcal{R}ec(\mathbb{R})$ à variations polynomialement limitées en temps continu polynomial sur $[-1, 1]^d$ correspondent précisément aux langages de P/rec.

Théorème 25 ($\mathbb{P}$ = P(Systèmes PLV Lisses))
Les langages calculés par les systèmes dynamiques à temps continu $\mathcal{H}$ à variations polynomialement limitées en temps continu polynomial sur $[-1, 1]^d$ correspondent précisément aux langages de $\mathbb{P}$.

A.10 Sur le phénomène de Zénon

Retournons maintenant aux systèmes PCD. A un temps continu fini, peut correspondre un temps discret non-fini : considérons par exemple la trajectoire maximale définie par le système PCD sur la figure A.2.

Cela a déjà été observé dans [Asarin and Maler, 1998], et a été utilisé pour montrer que tout langage arithmétique peut être reconnu par un système PCD.

Avec la terminologie de la définition 40, il est facile de montrer que T_ϕ est toujours un ensemble bien ordonné. Comme tout ensemble bien ordonné, il est isomorphe à un certain ordinal. Cet ordinal est considéré comme le temps discret de la trajectoire dans le cas général [Bournez, 1999b].

Par exemple, sur la figure A.2, la trajectoire partant de $(x, 0)$ vers $(0, 0)$ a un temps discret ω, pour un temps continu de

$$\frac{5}{2}x + \frac{5}{4}x + \frac{5}{8}x + \frac{5}{16}x + \cdots = 5x.$$

Le temps discret d'une trajectoire de temps continu fini peut alors être borné en fonction de la dimension.

Théorème 26 (Temps discret vs Temps Continu [Bournez, 1999a]) *Le temps discret T_d de toute trajectoire ϕ de temps continu fini d'un système PCD sur $\mathbb{R}^d$, satisfait $T_d < \omega^{d-1}$ pour $d \geq 3$, et $T_d \leq \omega$ pour $d = 2$.*

Rappelons que la hiérarchie hyperarithmétique est une extension de la hiérarchie arithmétique aux ordinaux constructibles. Elle consiste en les classes de langages

$$\Sigma_1, \Sigma_2, \ldots, \Sigma_k, \ldots, \Sigma_\omega, \Sigma_{\omega+1}, \Sigma_{\omega+2}, \ldots, \Sigma_{\omega 2}, \Sigma_{\omega 2+1}, \ldots, \Sigma_{\omega^2}, \ldots$$

indicées par les nombres ordinaux constructibles. C'est une hiérarchie qui satisfait les inclusions strictes $\Sigma_\alpha \subset \Sigma_\beta$ dès que $\alpha < \beta$. Elle peut se relier à la hiérarchie analytique par $\Delta_1^1 = \cup_\beta \Sigma_\beta$: voir [Rogers Jr., 1987].

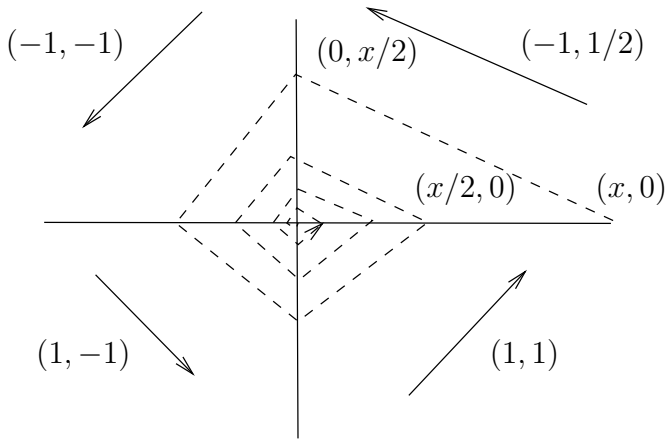


FIG. A.2 – Paradoxe de Zénon : une trajectoire de temps continu $5x$ et de temps discret ω entre le point $(x,0)$ et le point $(0,0)$.

La classe Σ_1 est définie comme la classe des ensembles récursivement énumérables. Lorsque k est un ordinal constructible et lorsque la classe Σ_k est définie, Σ_{k+1} est définie comme la classe des langages qui sont récursivement énumérables dans un ensemble de Σ_k. Lorsque k est un ordinal constructible limite, $k = lim\ k_i$, et lorsque les classes $(\Sigma_{k_i})_{i\in\mathbb{N}}$ sont définies, Σ_k est définie comme la classe des langages qui sont récursivement énumérables dans un diagonalisation fixe des classes $(\Sigma_{k_i})_i$: voir [Rogers Jr., 1987] pour les détails.

Il a été prouvé dans [Bournez, 1999a], [Bournez, 1999b] que la puissance des systèmes PCD en temps continu fini peut être caractérisée comme suit. Cela fournit une extension de [Asarin and Maler, 1998], en offrant une caractérisation complète de la puissance des systèmes PCD en fonction de leur dimension.

Théorème 27 (Systèmes PCD vs Hiérarchie Hyper-arithmétique) *La puissance des systèmes PCD rationnels en temps continu fini sur $[-1,1]^d$ ou $\mathbb{R}^d$ peut être caractérisée comme suit*

- *Pour $d = 2k+3$, ils reconnaissent précisément les langages de Σ_{ω^k}.*
- *Pour $d = 2k+4$, ils reconnaissent précisément les langages de Σ_{ω^k+1}.*

Grâce au corollaire 7, nous voyons que de tels phénomènes super Turing pour des systèmes lisses et $\mathcal{R}ec(\mathbb{R})$ sur $[-1,1]^d$ ne peuvent se produire.

Résumé

Les systèmes dynamiques continus permettent de modéliser de nombreux systèmes physiques, biologiques, ou issus de l'informatique distribuée. Nous nous intéressons à leur pouvoir de modélisation, et à leurs propriétés en tant que systèmes de calculs, et plus généralement aux propriétés calculatoires des modèles continus.

Les deux premiers chapitres ne visent pas à produire des résultats nouveaux, mais à motiver ce travail, et à le mettre en perspectives. Le chapitre 3 constitue un survol. Les chapitres 4, 5 et l'annexe A présentent un panorama de quelques-uns de nos résultats personnels en relations avec cette problématique.

Plus précisément, le chapitre 1 présente les systèmes dynamiques, avec un point de vue classique et mathématique. Il vise d'une part à souligner la richesse, et la subtilité des comportements possibles des systèmes dynamiques continus, et d'autre part à mettre en évidence que différents dispositifs sont intrinsèquement continus, et utilisables comme tels pour réaliser des calculs. En outre nous insistons sur la puissance de modélisation d'une classe de systèmes dynamiques, que nous nommons les problèmes de Cauchy polynomiaux.

Les exemples du chapitre 2, issus de la bioinformatique, des modèles de la biologie des populations, de la virologie biologique et de la virologie informatique, et de l'algorithmique distribuée, se distinguent de ceux du chapitre 1 par le fait qu'ils mettent explicitement en jeu une certaine notion de concurrence entre agents. Nous présentons la théorie des jeux, et ses modèles, en nous focalisant sur certains de ses modèles du dynamisme. Ces modèles continus deviennent naturels pour parler d'algorithmique distribuée, en particulier dès que l'on a affaire à des systèmes de grandes tailles, ou dont on ne contrôle pas les interactions. Nous pointons quelques modèles de l'algorithmique distribuée qui intègrent ces considérations, et le potentiel de l'utilisation des systèmes continus pour l'algorithmique distribuée.

Le chapitre 3 constitue un survol de la théorie des calculs pour les modèles à temps continu. La puissance des modèles de calculs à temps et espace discrets est relativement bien comprise grâce à la thèse de Church, qui postule que tous les modèles raisonnables et suffisamment puissants ont la même puissance, celle des machines de Turing. On peut aussi considérer des modèles où le temps est continu. Certaines grandes classes de modèles ont été considérées dans la littérature. Nous les reprenons dans ce chapitre, en présentant un panorama de ce qui est connu sur leurs propriétés calculatoires.

Le chapitre 4 présente un résumé de quelques-uns de nos résultats personnels à propos de la comparaison de la puissance de plusieurs modèles à temps continu, en relations avec la thèse de Emmanuel Hainry. Claude Shannon a introduit en 1941 le GPAC comme un modèle des dispositifs de calculs analogiques. Les résultats de Shannon ont longtemps été utilisés pour argumenter que ce modèle était plus faible que l'analyse récursive, et donc que les machines analogiques sont prouvablement plus faibles que les machines digitales. Avec Manuel Campagnolo, Daniel Graça, et Emmanuel Hainry, nous avons prouvé récemment que le GPAC et l'analyse récursive calculent en fait les mêmes fonctions. Ce résultat prend toute sa perspective si l'on comprend que les fonctions calculées par le GPAC correspondent aux problèmes de Cauchy polynomiaux, dont le pouvoir de modélisation est discuté dans le chapitre 1.

D'autre part, nous avons montré qu'il était possible de caractériser algébriquement les fonctions élémentairement calculables et calculables au sens de l'analyse récursive. Cela signifie d'une part qu'il est possible de les caractériser en termes d'une sous-classe des fonctions R-récursives à la Moore, ce qui étend les résultats de Campagnolo, Costa, Moore, de la calculabilité discrète à l'analyse récursive, mais aussi d'autre part, qu'il est possible de caractériser ces fonctions de façon purement continue, par l'analyse, sans référence à de la calculabilité.

Dans le chapitre 5, nous reprenons certains de nos résultats à propos de caractérisations logiques de classes de complexité dans le modèle de Blum Shub et Smale, en relations avec la thèse de Paulin Jacobé de Naurois. Le modèle de Blum Shub et Smale constitue un modèle de calcul à temps discret et à espace continu. Le modèle, défini initialement pour parler de complexité algébrique de problèmes sur le corps des réels, ou plus généralement sur un anneau, a été par la suite été étendu par Poizat en un modèle de calculs sur une structure logique arbitraire. Avec Paulin Jacobé de Naurois, Felipe Cucker et Jean-Yves Marion, nous avons caractérisé syntaxiquement les classes de complexité majeures dans ce modèle sur une structure arbitraire, à la Bellantoni et Cook 1992.

Le chapitre 6 est consacré à une conclusion, dans laquelle nous reprenons plusieurs questions et perspectives qui nous semblent intéressantes.

Dans l'annexe A, nous discutons un point de vue sur les hypercalculs. La question de l'existence de systèmes capables de réaliser des hypercalculs, c'est-à-dire d'effectuer des calculs exploitables qui ne seraient pas réalisables par aucune machine de Turing, fait encore couler de l'encre et des controverses. Nous avons été invité à exprimer notre point de vue dans un numéro spécial sur le sujet, que nous reprenons en annexe A. Nous y rappelons plusieurs mauvaises compréhensions fréquentes de la thèse de Church, et nous présentons un panorama de plusieurs classes de systèmes mathématiques, avec la caractérisation de leur puissance.