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 1 (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, 2006], [Mycka and Costa, 2004], [Graça, 2004], [Campagnolo, 2004] 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 2 (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 3 (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.

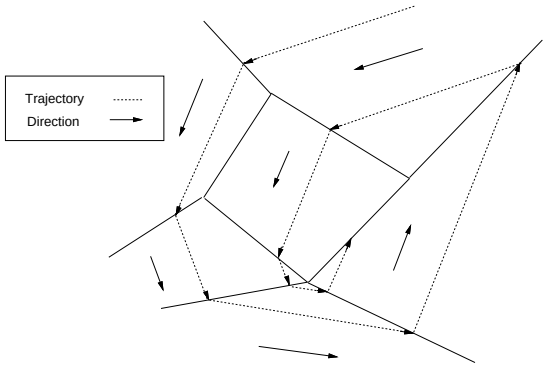

FIG. A.1 – Un système PCD en dimension 2.

Théorème 1 (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 1 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 2 (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 1 (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, 1998] 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 3 (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 3. 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 4 (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 5 (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 6 (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 1 (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 4 (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 7 (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 [Šíma and Orponen, 2003].

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 5 (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 2 (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 6 ($\mathbb{P} \subset$ 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 7 ($P/rec \subset$ 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 8 ($P \subset$ 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 8 (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 9 (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 10 ($P =$ 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 11 ($P/rec =$ 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 12 ($\mathbb{P} =$ 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 4, 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 13 (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].

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,

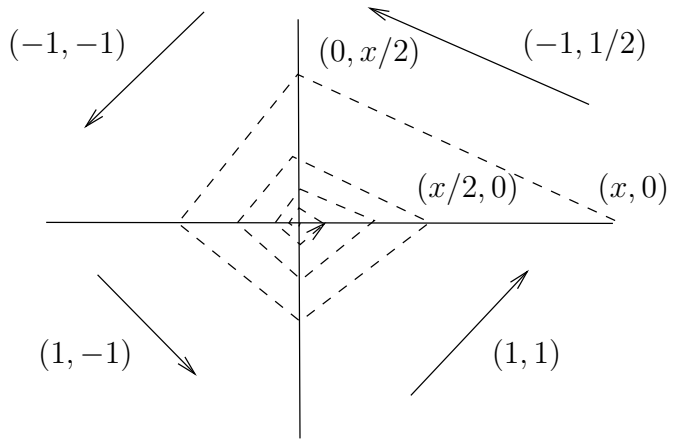

FIG. A.2 – Paradoxe de Zénon : une trajectoire de temps continu $5x$ et de temps discret ω entre le point $(x,0)$ et le point $(0,0)$.

$k = lim\ k_i$, 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 14 (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 1, 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.

Bibliographie

[Aaronson, 2005] Aaronson, S. (2005). NP-complete problems and physical reality. *ACM SIGACT News*, 36(1) :30–52.

[Antsaklis, 2000] Antsaklis, P. J., editor (2000). *Proceedings of the IEEE*, volume 88.

[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 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.

[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.

[Boker and Dershowitz, 2005] Boker, U. and Dershowitz, N. (2005). A speculative Church-Turing theorem. Preprint.

[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 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.

[Campagnolo, 2004] Campagnolo, M. (2004). Continuous time computation with restricted integration capabilities. *Theoretical Computer Science*, 317(4) :147–165.

[Church, 1936] Church, A. (1936). An unsolvable problem of elementary number theory. *American Journal of Mathematics,*, 58 :345–363. Reprinted in [Davis, 1965].

[Cleland, 2004] Cleland, C. E. (2004). The concept of computability. *Theoretical Computer Science*, 317(1–3) :209–225.

[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.

[Davis, 1965] Davis, M. (1965). *The Undecidable : Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions*. Raven Press. Reprinted by Dover Publications, Incorporated in 2004.

[Gandy, 1980] Gandy, R. (1980). Church's thesis and principles for mechanisms. *The Kleene Symposium*, pages 123–148.

[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, 2004] Graça, D. S. (2004). Some recent developments on Shannon's general purpose analog computer. *Mathematical Logic Quarterly*, 50(4–5) :473–485.

[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].

[Kleene, 1936] Kleene, S. C. (1936). General recursive functions of natural numbers. *Mathematical Annals*, 112 :727–742. Also in [Davis, 1965].

[Ko, 1991] Ko, K.-I. (1991). *Complexity Theory of Real Functions*. Progress in Theoretical Computer Science. Birkhäuser, Boston.

[Moore, 1990] Moore, C. (1990). Unpredictability and undecidability in dynamical systems. *Physical Review Letters*, 64(20) :2354–2357.

[Moore, 1996] Moore, C. (1996). Recursion theory on the reals and continuous-time computation. *Theoretical Computer Science*, 162(1) :23–44.

[Moore, 1998] Moore, C. (1998). 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)*. Springer.

[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, 2006] Mycka, J. and Costa, J. F. (2006). The $P \neq NP$ conjecture in the context of real and complex analysis. *Journal of Complexity*, 22(2) :287–303.

[Ord, 2006] Ord, T. (2006). The many forms of hypercomputation. *Applied Mathematics and Computation*, 178(1) :143–153.

[Papadimitriou, 1994] Papadimitriou, C. H. (1994). *Computational Complexity*. Addison-Wesley.

[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 initial data such that its unique solution is not computable. *Advances in Mathematics*, 39 :215–239.

[Rogers Jr., 1987] Rogers Jr., H. (1987). *Theory of Recursive Functions and Effective Computability*. MIT 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 Sontag, 1994] Siegelmann, H. T. and Sontag, E. D. (1994). Analog computation via neural networks. *Theoretical Computer Science*, 131(2) :331–360.

[Šíma and Orponen, 2003] Šíma, J. and Orponen, P. (2003). General-purpose computation with neural networks : A survey of complexity theoretic results. *Neural Computation*, 15(12) :2727–2778.

[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.

[Turing, 1936] Turing, A. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42(2) :230–265. Reprinted in [Davis, 1965].

[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.

[Weihrauch, 2000] Weihrauch, K. (2000). *Computable Analysis*. Springer.

[Yao, 2003] Yao, A. C.-C. (2003). Classical physics and the Church–Turing Thesis. *Journal of the ACM*, 50(1) :100–105.