Chapitre 6

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, 1992].

6.1 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, 1992], 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., 2006], [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, 1992]. 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, 1992] 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.

6.2 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 1 *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 2 *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 3 *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.*

Nous pouvons maintenant définir les classes de complexité principales.

[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.

Définition 4 *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 1 ([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 $\mathcal{K} = (\mathbb{R}, +, -, *, =, \leq, \{c \in \mathbb{R}\})$.*

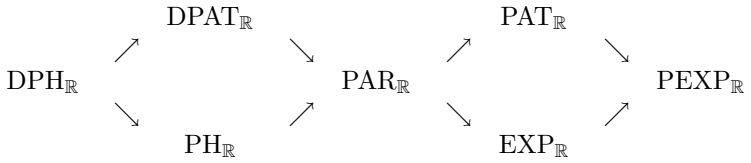
Il est aussi possible de considérer l'analogue de toutes les classes connues en complexité classique.

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.

6.3 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.

6.3.1 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 5 *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.

6.3.2 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 1 *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 $\mathcal{K}$.

6.4 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, 1992].

6.4.1 Fonctions récursives sûres

Exemple 1 *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

$$\begin{aligned}
\mathsf{Cons}(\epsilon, \overline{y}) &= \overline{y} \\
\mathsf{Cons}(a.\overline{x}, \overline{y}) &= \mathsf{cons}(a, \mathsf{Cons}(\overline{x}, \overline{y})) \\
\exp(\epsilon) &= \mathbf{1} \\
\exp(a.\overline{x}) &= \mathsf{Cons}(\exp(\overline{x}), \exp(\overline{x})).
\end{aligned}$$

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, 1992], 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, 1992], 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 6 *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* $a \in \mathbb{K}, \overline{x}, \overline{x_1}, \overline{x_2} \in \mathbb{K}^*$

$$\begin{array}{rclrclrcl}
\mathsf{hd}(;a.\overline{x}) & = & a & \mathsf{tl}(;a.\overline{x}) & = & \overline{x} & \mathsf{cons}(;a.\overline{x_1}, \overline{x_2}) & = & a.\overline{x_2} \\
\mathsf{hd}(;\epsilon) & = & \epsilon & \mathsf{tl}(;\epsilon) & = & \epsilon & \mathsf{cons}(;\epsilon, \overline{x_2}) & = & \overline{x_2}.
\end{array}$$

(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 :*

$$\begin{aligned}
\mathsf{Op}_i(;a_1.\overline{x_1}, \ldots, a_{n_i}.\overline{x_{n_i}}) &= (op_i(a_1, \ldots, a_{n_i})).\overline{x_{n_i}} \\
\mathsf{Rel}_i(;a_1.\overline{x_1}, \ldots, a_{n_i}.\overline{x_{n_i}}) &= \begin{cases} \mathbf{1} \text{ si } rel_i(a_1, \ldots, a_{n_i}) \\ \mathbf{0} \text{ sinon.} \end{cases}
\end{aligned}$$

La relation d'égalité sera notée Equal.

(iv) *Une fonction de sélection.*

$$\mathsf{Select}(;\overline{x}, \overline{y}, \overline{z}) = \begin{cases} \overline{y} & \text{si } \mathsf{hd}(\overline{x}) = \mathbf{1} \\ \overline{z} & \text{sinon.} \end{cases}$$

Définition 7 *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 1 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.

6.4.2 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, 1992] au cas des structures arbitraires.

Théorème 2 *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 $\mathcal{K}$.

6.5 Temps parallèle polynomial

6.5.1 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 8 *$FPAR_{\mathcal{K}}$ 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.

[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.

6.5.2 Fonctions récursives sûres avec substitutions

Définition 9 *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.

6.5.3 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 3 *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.

6.6 Hiérarchie polynomiale

6.6.1 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 10 *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].

6.6.2 Fonctions récursives sûres avec minimisations prédicatives

Dans l'esprit de [Bellantoni, 1994], nous introduisons maintenant la notion de minimisation prédicative.

Définition 11 *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*

$$f(\overline{x};\overline{a}) = \mathfrak{Z}\overline{b}(h(\overline{x};\overline{a},\overline{b})) = \begin{cases} \mathbf{1} & \text{s'il existe } \overline{b} \in \mathbb{K}^* \text{ tel que } h(\overline{x};\overline{a},\overline{b}) = \mathbf{0} \\ \mathbf{0} & \text{sinon.} \end{cases}$$

Nous introduisons maintenant de nouveaux ensembles de fonctions.

Définition 12 *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 13 *Supposons que* F *soit une classe de fonctions : une fonction* f *est dans* $\mathfrak{Z}F$ *si elle est définie avec une minimisation prédicative sur une fonction* h *de* F*.*

Nous définissons par induction les ensembles suivants :

- $F^0_{\mathcal{K}} = \mathrm{SR}_{\mathcal{K}}$.
- $$F^{i+1}_{\mathcal{K}} = RSR_{\mathcal{K}}(F^i_{\mathcal{K}} \bigcup \mathfrak{Z}F^i_{\mathcal{K}}),$$

  pour $i \geq 0$*.*

Nous dénotons par

$$\mathfrak{Z}\mathrm{PH}_{\mathcal{K}} = \bigcup_{i\in\mathbb{N}} F^i_{\mathcal{K}}$$

la clôture des fonctions sûres de base sur $\mathcal{K}$ *par applications de récursions sûres restreintes, minimisations prédicatives, et compositions sûres.*

6.6.3 Caractérisation de la hiérarchie polynomiale

Nous avons prouvé le résultat suivant dans les articles [Bournez et al., 2003], [Bournez et al., 2005a].

Théorème 4 *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 1 *Un problème de décision sur* $\mathcal{K}$ *appartient à* $\mathrm{PH}_{\mathcal{K}}$ *si et seulement si sa fonction caractéristique peut être définie dans* $\exists\mathrm{PH}_{\mathcal{K}}$.

6.7 Hiérarchie polynomiale digitale

6.7.1 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 14 *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*

$$\mathrm{DFPH}_{\mathcal{K}} = \bigcup_{i=0}^{\infty} \mathrm{DF}\Delta^i_{\mathcal{K}}.$$

6.7.2 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 15 *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 16 *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.*

6.7.3 Caractérisation de la hiérarchie polynomiale digitale

Nous avons prouvé le résultat suivant dans les articles [Bournez et al., 2003], [Bournez et al., 2005a].

Théorème 5 *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 2 *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* $\mathfrak{Z}_{\mathrm{D}}\mathrm{DPH}_{\mathcal{K}}$*.*

Quand on considère des structures finies, cela donne une caractérisation de la hiérarchie polynomiale classique alternative à celle de [Bellantoni, 1994].

Corollaire 3 *Un problème de décision appartient à* PH *si et seulement si sa fonction caractéristique peut être définie dans* $\mathfrak{Z}_{\mathrm{D}}\mathrm{DPH}_{\{\mathbf{0},\mathbf{1}\}}$*.*

6.8 Temps polynomial alternant

6.8.1 Définitions

Définition 17 *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 $\mathrm{PH}_{\mathcal{K}} \subseteq \mathrm{PAT}_{\mathcal{K}}$.

6.8.2 Fonctions récursives sûres avec substitutions prédicatives

Définition 18 *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*

$$f(\overline{x};\overline{a}) = \mathbf{\exists}^{[1]} c(h(\overline{x};\overline{a},c)) = \begin{cases} \mathbf{1} & \text{s'il existe } c \in \mathbb{K} \text{ tel que } h(\overline{x};\overline{a},c) = \mathbf{0} \\ \mathbf{0} & \text{sinon.} \end{cases}$$

Définition 19 *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*

$$\begin{aligned} f(\epsilon,\overline{x};\overline{u},\overline{y}) &= h(\overline{x};\overline{u},\overline{y}) \\ f(a.\overline{z},\overline{x};\overline{u},\overline{y}) &= g(\overline{z},\overline{x};\mathbf{\exists}^{[1]} c(f(\overline{z},\overline{x};c.\overline{u},\overline{y})),\overline{y}). \end{aligned}$$

Définition 20 *L'ensemble* $\mathbf{\exists}^{[1]}\mathrm{PAT}_{\mathcal{K}}$ *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.*

6.8.3 Caractérisation du temps polynomial alternant

Nous avons prouvé le résultat suivant dans les articles [Bournez et al., 2004b], [Bournez et al., 2006].

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 dans $\mathrm{FPAT}_{\mathcal{K}}$ *si et seulement si elle peut être définie comme une fonction dans* $\mathbf{\exists}^{[1]}\mathrm{PAT}_{\mathcal{K}}$.

6.9 Temps polynomial alternant digital

6.9.1 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

$$\mathrm{DFPAT}_{\mathcal{K}} = \mathrm{FP}_{\mathcal{K}}^{\mathrm{DPAT}_{\mathcal{K}}}.$$

6.9.2 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 21 *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,

$$f(\overline{x};\overline{a}) = \mathbf{\exists}_{\mathsf{D}}^{[1]} c(h(\overline{x};\overline{a},c)) = \begin{cases} \mathbf{1} & \text{s'il existe } c \in \{\mathbf{0},\mathbf{1}\} \text{ tel que } h(\overline{x};\overline{a},c) = \mathbf{0} \\ \mathbf{0} & \text{sinon.} \end{cases}$$

Définition 22 *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*

$$\begin{aligned} f(\epsilon,\overline{x};\overline{u},\overline{y}) &= h(\overline{x};\overline{u},\overline{y}) \\ f(a.\overline{z},\overline{x};\overline{u},\overline{y}) &= g(\overline{z},\overline{x};\mathbf{\exists}_{\mathsf{D}}^{[1]} c f(\overline{z},\overline{x};c.\overline{u},\overline{y}),\overline{y}). \end{aligned}$$

Définition 23 *L'ensemble* $\mathbf{\exists}_{\mathsf{D}}^{[1]}\mathrm{PAT}_{\mathcal{K}}$ *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.*

6.9.3 Caractérisation du temps polynomial alternant digital

Nous avons prouvé le résultat suivant dans les articles [Bournez et al., 2004b], [Bournez et al., 2006].

Théorème 7 *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 4 *Un ensemble* $S \subset \{0,1\}^*$ *est dans* PSPACE *si et seulement si sa fonction caractéristique peut être définie dans* $\exists^{[1]}\mathrm{DPAT}_{\{\mathbf{0},\mathbf{1}\}}$.

6.9.4 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 24 *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 25 *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., 2006].

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 $\mathrm{DFPAT}_{\mathcal{K}}$ *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.

Bibliographie

[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.

[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, 1992] Bellantoni, S. and Cook, S. (1992). A new recursion-theoretic characterization of the poly-time functions. *Computational Complexity*, 2 :97–110.

[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.

[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 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., 2006] Bournez, O., Cucker, F., de Naurois, P. J., and Marion, J.-Y. (2006). 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.

[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.

[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.

[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.

[Ebbinghaus and Flum, 1995] Ebbinghaus, H. and Flum, J. (1995). *Finite Model Theory.* Perspectives in Mathematical Logic, Omega Series. Springer–Verlag, Heidelberg.

[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.

[Goode, 1994] Goode, J. B. (1994). Accessible telephone directories. *The Journal of Symbolic Logic*, 59(1) :92–105.

[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.

[Grzegorczyk, 1957] Grzegorczyk, A. (1957). On the definitions of computable real continuous functions. *Fundamenta Mathematicae*, 44 :61–71.

[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.

[Hofmann, 1999] Hofmann, M. (1999). Type systems for polynomial-time computation. Habilitation.

[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.

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

[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.

[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.

[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.

[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.

[Poizat, 1995] Poizat, B. (1995). *Les petits cailloux*. aléas.

[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].

[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.

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