Calculabilité et complexité algébrique

Denis Monasse

4 mai 1998

1 Introduction

Les méthodes classiques de définition de la calculabilité ou de la complexité supposent à l'origine que l'on ne travaille que sur des entiers, et même plus précisément sur des entiers écrits en binaire, c'est à dire sur les symboles 0 et 1. Ceci résulte d'une vision microscopique de l'informatique ; la même méthode appliquée à la physique supposerait, par exemple, que pour étudier le mouvement d'une voiture on étudie le mouvement de chacun de ses atomes.

De même qu'en physique on connait (au moins depuis Newton) l'utilité, par exemple en mécanique, d'une vision macroscopique et continue par une modélisation mathématique en nombres réels, il est apparu dans les années 80 la nécessité de dépasser le cadre étroit de la calculabilité et de la complexité classique par une modélisation mathématique des algorithmes travaillant directement sur des structures de données abstraites comme les booléens, les entiers, les entiers modulo p, les nombres rationnels, les nombres complexes, indépendamment de leur implémentation en machine sous forme de 0 et de 1. Ceci suppose de nouvelles définitions pour un algorithme (et donc pour une machine de Turing) travaillant sur ces objets, pour la calculabilité d'un ensemble de tels objets ou pour la complexité d'un tel algorithme. Ce travail de mise en forme a été effectué en grande partie par Lenor BLUM, Mike SHUB et Steve SMALE qui justifient l'introduction et l'étude de machines calculant en nombres réels de la manière suivante :

Notre suggestion est que l'ordinateur moderne peut être idéalisé de la même manière que Newton idéalisait son univers discret. Les nombres en machine sont des nombres rationnels, en nombre fini, mais ils remplissent un ensemble borné de nombres réels de manière suffisamment dense pour que le fait de voir l'ordinateur manipuler des nombres réels est une idéalisation raisonnable, au moins dans bon nombre de contextes.... Pour une large variété de calculs scientifiques, les mathématiques continues que simulent la machine sont le moyen correct d'analyser l'opération de cette même machine. Ces raisons donnent une certaine justification à prendre comme modèle du calcul scientifique un modèle de machine qui accepte comme entrées les nombres réels.

Nous voulons donner ici une simple introduction à ce sujet en renvoyant le lecteur aux deux ouvrages cités dans la bibliographie pour les approfondissements et les démonstrations précises.

2 Quelques algorithmes travaillant sur les réels

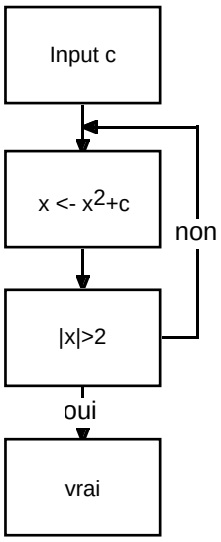
2.1 L'ensemble de Mandelbrot est-il décidable ?

On connait la définition classique de l'ensemble $\mathcal{M}$ de Mandelbrot : à chaque nombre complexe c on associe le polynôme du second degré $T_c(x) = z^2 + c$ et $\mathcal{M}$ est l'ensemble des nombres complexes c tels que la suite $(c, T_c(c), T_c(T_c(c)), T_c(T_c(T_c(c))), \ldots) = (T_c^n(c))_{n\in\mathbb{Z}}$ soit bornée. On montre facilement que la suite est bornée si et seulement si, pour tout entier $n \in \mathbb{N}$, $|T_c^n(c)| \leq 2$, ce qui conduit immédiatement à l'algorithme suivant écrit en Caml qui *teste* si un complexe (identifié à un couple de réel) est dans le complémentaire de l'ensemble $\mathcal{M}$

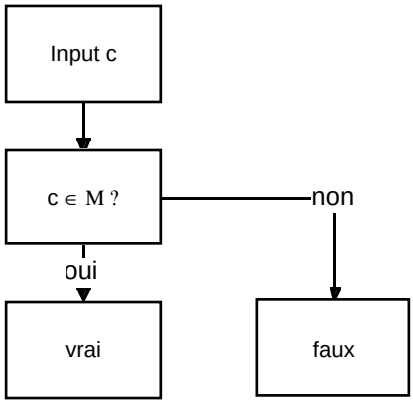
Programme 2.1 complémentaire de l'ensemble de Mandelbrot

```
let pas_dans_M (x,y) = let x1=ref x and y1=ref y in
        while !x1*. !x1+!y1*. !y1<=4 do
           let (x2,y2)=(!x1,!y1) in
               x1:=x2 *. x2-. y2 *. y2 +. x;
               y1:=2. *. x2 *. y2 +. y
        done;
        true;;
```

Ce petit programme correspond à l'organigramme suivant



Le problème naturel qui se pose est de savoir s'il existe un algorithme similaire permettant de tester réellement si un nombre complexe est dans $\mathcal{M}$ c'est à dire un algorithme qui puisse s'écrire sous forme d'organigramme de la façon suivante



On dirait alors que l'ensemble de Mandelbrot est décidable (autrement dit qu'il existe un algorithme qui teste à coup sûr, dans le cadre de la structure donnée, si un point appartient ou non à l'ensemble).

La question n'est pas si simple à formuler. La théorie classique de la calculabilité suppose que tous les ensembles considérés sont dénombrables, ce qui n'est évidemment pas le cas. Une réponse possible est de n'utiliser que des réels calculés en machine, mais le problème est qu'on ne sait plus alors tester si deux nombres réels sont égaux. Une autre réponse serait de se contenter de rechercher les complexes à partie entière et partie imaginaire rationnels (ce que l'on pourrait appeler le squelette rationnel de $\mathcal{M}$) mais on sait que, même dans le cas d'une courbe algébrique, ce squelette rationnel ne décrit en aucune façon l'ensemble étudié comme le montre l'exemple de la courbe d'équation $x^3 + y^3 = 1$ dont le squelette rationnel est vide.

Pour que la question "*l'ensemble de Mandelbrot est-il décidable ?*" prenne un sens, il nous faut donc abstraire la structure de donnée *nombre réel* sur laquelle on travaille en considérant que les nombres réels sont des objets dont l'implémentation n'a pas à être considérée, mais sur lesquels on sait faire un certain nombre d'opérations comme le produit, l'addition, la soustraction, la multiplication par 2 et la comparaison.

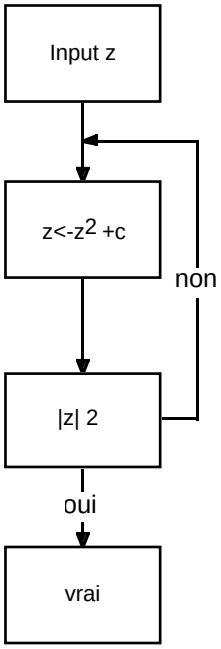
2.2 Un ensemble de Julia est-il décidable ?

Prenons cette fois encore l'application polynomiale $T_c : z \mapsto z^2 + c$; on lui associe l'ensemble de Julia $\mathcal{J}_c$ des nombres complexes z tels que la suite $(T_c^n(z))_{n\in\mathbb{Z}}$ soit bornée. Ici encore, on vérifie facilement que la suite est bornée si et seulement si $\forall n \in \mathbb{N},\ |T_c^n(z)| \leq 2$ ce qui conduit encore une fois à un programme en Caml testant si un nombre complexe z (décrit comme un couple de réels (x, y)) est dans le complémentaire d'un ensemble de Julia $\mathcal{J}_c$, où le nombre complexe c est décrit comme un couple de réels (a, b)

Programme 2.2 complémentaire de l'ensemble de Julia

```
let pas_dans_J (a,b) (x,y) = let x1=ref x and y1=ref y in
         while !x1*. !x1+!y1*. !y1<=4 do
            let (x2,y2)=(!x1,!y1) in
                x1:=x2 *. x2-. y2 *. y2 +. a;
                y1:=2. *. x2 *. y2 +. b
         done;
         true;;
```

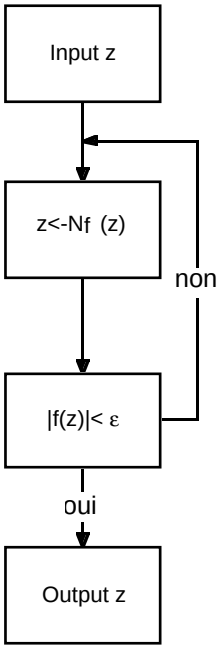
ce qui correspond à l'organigramme suivant



Ici encore, la question naturelle qui se pose est de savoir si l'ensemble $\mathcal{J}_c$ est décidable, la structure sous-jacente étant celle de l'ensemble des nombres réels muni de l'addition, de la soustraction, de la multiplication et de la relation d'ordre.

2.3 Quels sont les bons points pour la méthode de Newton ?

Considérons une fonction polynomiale $f(z)$ à coefficients complexes ; on peut associer à ce polynôme l'application de Newton $N_f : z \mapsto z - \dfrac{f(z)}{f'(z)}$. A tout nombre complexe z on peut associer la suite $(N_f^n(z))_{n\in\mathbb{N}}$ des itérées de N_f sur z. Si cette suite converge vers une limite ζ, on sait que ζ est un point fixe de N_f et donc une racine de f. D'autre part, on sait que les points fixes de N_f sont attractifs, autrement dit vérifient $|N_f'(\zeta)| < 1$. Ceci signifie que chacun d'entre eux bénéficie d'un bassin d'attraction qui est un voisinage de ζ. On peut donc associer à tout polynôme f et à tout nombre réel strictement positif une *machine de Newton* dont l'organigramme serait



Cette machine de Newton, lorsqu'elle s'arrête, fournit une valeur approchée d'une racine de f. Les problèmes qui se posent sont alors multiples :

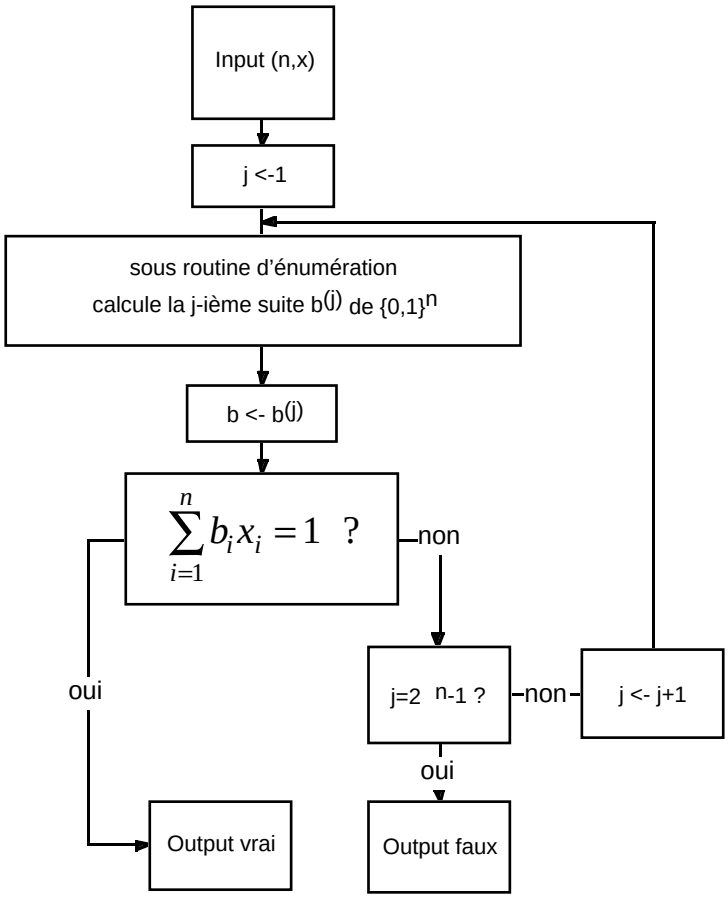
- l'ensemble des *bons points* de f (c'est à dire les points z tels que la machine de Newton s'arrête) est-il décidable ?
- peut on trouver un ensemble suffisant de bons points de f de manière à obtenir des valeurs approchées de toutes les racines complexes de f ?
- quelle est la complexité de l'algorithme obtenu en fonction du degré du polynôme et de la précision ε recherchée ?

Ici encore la structure sous-jacente naturelle est celle des nombres réels (et non des nombres complexes) comme le montre la présence du module (qui n'est pas une fonction définissable dans le cadre du corps des nombres complexes) et celle de la comparaison.

2.4 Le problème du sac à dos

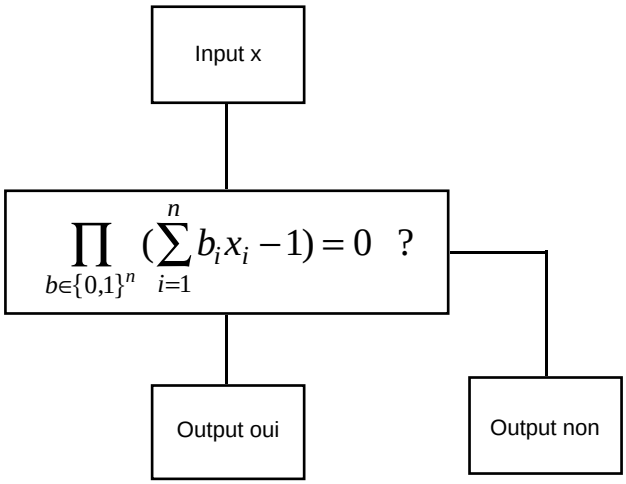
Le problème classique du sac à dos est le suivant : étant donné des entiers $x_1, \ldots, x_n$ et c, peut on trouver $S \subset \{1, \ldots, n\}$ tel que $\sum_{i \in S} x_i = c$? Si l'on traduit ce problème en nombre réels, on peut évidemment choisir $c = 1$ et le problème devient : étant donné des réels $x_1, \ldots, x_n$, peut on trouver $S \subset \{1, \ldots, n\}$ tel que $\sum_{i \in S} x_i = 1$?

Contrairement aux problèmes précédents l'ensemble des n-uples $(x_1, \ldots, x_n)$ qui répond à la question est évidemment décidable : il suffit d'effectuer une recherche exhaustive sur toutes les parties $S \subset \{1, \ldots, n\}$ et l'organigramme suivant permet de décider à coup sûr si un tel n-uple est un bon n-uple :



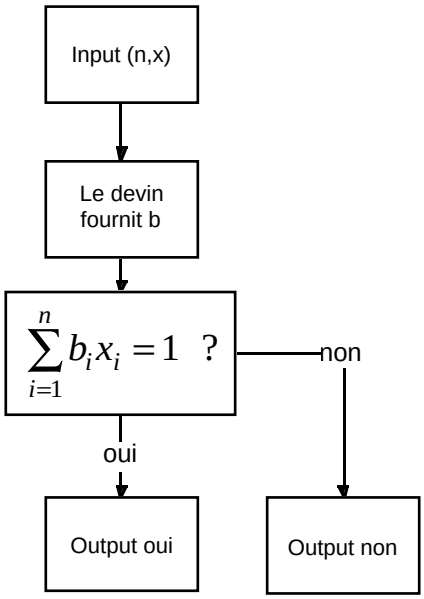
L'algorithme en question a cependant un gros défaut : le nombre de noeuds traversé par l'algorithme (en tenant compte des visites multiples) est une fonction exponentielle de n. Il a une grosse qualité : il fonctionne pour tous les n ; autrement dit son espace de travail naturel est $\bigcup_{n\in\mathbb{N}} \mathbb{R}^n$. On dit encore que l'algorithme est uniforme en n.

Supposons maintenant que nous autorisions des multiplications de 2^n-éléments dans $\mathbb{R}^n$ pour un n-fixé (cette multiplication étant par exemple cablée dans la machine, l'algorithme ne sera évidemment plus uniforme et nécessitera une machine particulière pour chaque n). Pour tester si le n-uple $(x_1, \ldots, x_n)$ est *bon*, il suffit de tester si $\prod_{b\in\{0,1\}^n} \left(\sum_{i=1}^{n} b_i x_i - 1\right) = 0$ si bien que le nombre de noeuds traversé par l'algorithme devient constant dans l'organigramme suivant :



Autrement dit, l'introduction de la non-uniformité (c'est à dire d'une machine spécifique pour un n donné) a réduit de manière dramatique la complexité de la solution. La même complexité, avec la même non-uniformité, peut être obtenue en faisant tourner 2^n processeurs en parallèle.

Une autre manière de réduire la complexité de la solution est d'introduire un *devin*. Celui-ci serait chargé de fournir à coup sûr une solution, si celle-ci existe. L'organigramme devient alors le suivant :



L'algorithme (qui est de nouveau uniforme en n) s'exécute alors dans un temps linéaire en n (le temps de calculer la somme $\sum_i b_i x_i$, l'organigramme correspondant ayant été condensé pour des raisons de simplification). Ici, c'est l'introduction du non-déterminisme (de la chance) qui a diminué de manière dramatique la complexité de la solution.

2.5 Le Nullstellensatz en tant que problème de décision

Nous considérons ici le problème de décision suivant. Soit $f_1, \ldots, f_k$ une famille finie de polynômes à n variables à coefficients dans $\mathbb{C}$. Ces polynômes ont-ils un zéro en commun (autrement dit la variété algébrique affine V qu'ils définissent dans $\mathbb{C}^n$ est-elle non vide.

L'entrée de cet algorithme peut être envisagée comme le vecteur des coefficients des f_i, vecteur de taille N si on pose

$$N = \sum_{i=1}^{k} C_{n+d_i}^{n}$$

avec $d_i = \deg f_i$. Le nombre N représente donc la *taille* de l'entrée, chaque nombre complexe étant considéré comme de taille 1.

Le Nullstellensatz de Hilbert assure que V est non vide si et seulement si l'idéal engendré par les f_i dans $\mathbb{C}[z_1, \ldots, z_n]$ n'est pas égal à $\mathbb{C}[z_1, \ldots, z_n]$ tout entier, autrement dit que

$$V = \emptyset \iff \exists g_1, \ldots, g_k, \sum_{i=1}^{k} g_i f_i = 1$$

Si l'on connaît une borne du degré des polynômes g_i, le problème de l'existence de polynômes g_i tels que $\sum_{i=1}^{k} g_i f_i = 1$ se réduit à la résolution d'un système linéaire, ce que l'on peut faire par un algorithme de pivot. Or on peut montrer que si $D = \max(3, d_1, \ldots, d_k)$, alors on peut se contenter de rechercher des polynômes g_i de degrés inférieurs à D^n (cf Kollar(1988) :Sharp effective Nullstellensatz, *J. of Amer.Math.Soc,1*,963-975).

Ceci nous fournit donc un algorithme répondant à notre problème de décision ; notre problème est donc décidable sur $\mathbb{C}$ (ou plus précisément : l'ensemble des polynômes $(f_1, \ldots, f_k)$ qui ont un zéro commun est un sous-ensemble décidable de $\mathbb{C}[z_1, \ldots, z_n]^k$). Cependant, connaissant la complexité classique en p^3 de l'algorithme du pivot pour la résolution d'un système de p équations à p inconnues, le lecteur se convaincra aisément que notre algorithme a une complexité exponentielle en la taille N des entrées. On peut formuler la conjecture qu'il n'existe aucun algorithme de complexité polynomiale en N qui réponde à notre problème de décision ; on dira encore que notre problème de décision est inaccessible sur $\mathbb{C}$, ou encore qu'il n'est pas dans la classe **P** sur $\mathbb{C}$.

Par contre, dans une hypothèse non déterministe, ce problème devient de complexité polynomiale. Supposons que nous disposions d'un devin qui fournisse à coup sûr, s'il en existe, un point $(z_1, \ldots, z_n)$ de la variété V. Il nous suffira pour répondre à notre problème de décision de tester si $f_1(z_1, \ldots, z_n) = 0, \ldots, f_k(z_1, \ldots, z_n) = 0$ ce qui se fait de façon évidente en un temps polynomial en N. On dira que notre problème est dans la classe **NP** sur $\mathbb{C}$.

2.6 Quartiques réelles

Le problème de décision du Nullstellensatz sur le corps des nombres réels se ramène immédiatement à un problème à un seul polynôme en posant $g = \sum_{i=1}^{k} f_i^2$. De plus, à moins que tous les f_i soient linéaires, on a $\deg g \geq 4$. Nous nous intéresserons donc au cas de degré minimal 4.

Donnons nous un polynôme f à n variables de degré 4 qui peut être vu comme un vecteur de $\mathbb{R}^N$ avec $N = C_{n+4}^4$. Nous cherchons à savoir s'il existe $(x_1, \ldots, x_n) \in \mathbb{R}^n$ tel que $f(x_1, \ldots, x_n)$. Tarski en 1951 puis Collins en 1975 ont fourni des algorithmes pour résoudre ce problème de décision, chacun ayant une complexité hyper-exponentielle en N. Plus récemment des algorithmes de complexité exponentielle ont été découverts.

On peut formuler la conjecture qu'il n'existe pas d'algorithme à complexité polynomiale en N qui réponde à notre problème, autrement dit que notre problème n'est pas dans la classe **P** sur $\mathbb{R}$. Bien entendu, notre problème est dans la classe **NP**, puisque l'aide d'un devin résoudra celui-ci en un temps polynomial (celui de l'évaluation).

3 Calculs dans un anneau

3.1 Introduction

Citons Bruno POIZAT :

Dans un calcul classique, où on ne manipule que les deux chiffres 0 et 1, les opérations, outre des opérations de gestion ou de déplacement propre au fonctionnement de l'algorithme, sont d'écrire 0, ou d'écrire 1, ou bien de tester si le chiffre que l'on a devant soi est un 0 ou un 1.

Nous étendons cette notion à un contexte moins réaliste : considérant une structure M arbitraire, on manipule les éléments de M comme si c'était les lettres 0 et 1, comme opérations on admet les fonctions de M et comme tests ses relations.(...) On ne se préoccupe pas de la façon dont sont effectuées ces opérations et ces tests ; on convient que, quelle qu'en soit la nature, chaque opération s'effectue en une unité de temps : le temps de calcul c'est le nombre de tours effectués dans l'exécution du programme, c'est le nombre de fois où le calculateur doit consulter sa liste d'instructions.

Pour simplifier, nous supposerons par la suite que M est un anneau (ou un corps), que les opérations de M sont les opérations de l'anneau (somme, différence, produit) ou du corps (avec en plus la division) et que les relations de comparaisons sont l'égalité sur un anneau général (comme $\mathbb{C}$), l'égalité et l'inégalité sur un

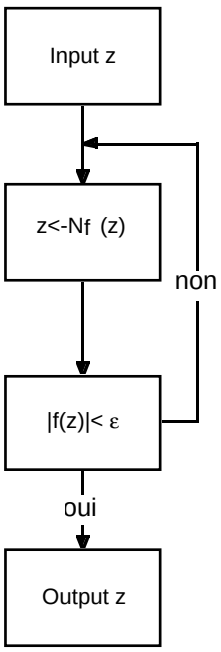
anneau ordonné (comme $\mathbb{R}$). Les booléens pourront se traduire en les éléments 0 et 1 de l'anneau ; les opérations booléennes étant classiquement définies sur ces éléments par $\neg x = 1 - x$, $x \vee y = x + y - xy$ et $x \wedge y = xy$, opérations polynomiales que l'on peut étendre à tous les éléments de l'anneau. Remarquons également que si x est un booléen (c'est à dire $x \in \{0, 1\}$) la formule conditionnelle (on dit encore le *sélecteur*)

si x alors y sinon z

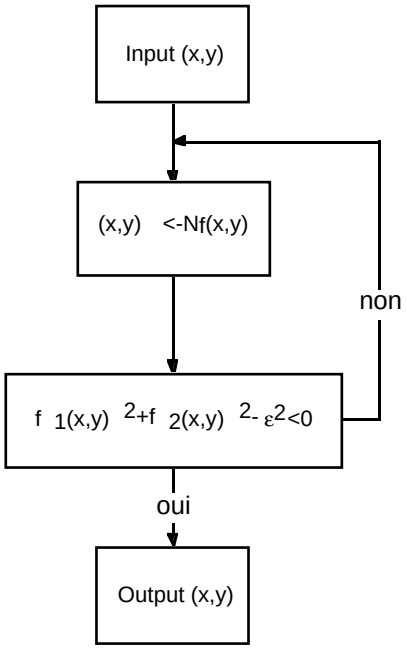
peut se traduire par une formule polynomiale $xy + (1 - x)z$, qui est encore définie si x n'est pas un booléen.

3.2 Retour à la méthode de Newton

Reprenons l'organigramme d'une machine de Newton très simple recherchant une racine approchée à ε près d'un polynôme $f(z)$ en itérant la fonction $N_f : z \mapsto z - \dfrac{f(z)}{f'(z)}$.



Cette machine est en fait une machine sur $\mathbb{R}$ (comme le montre la présence de l'inégalité) et nous avons intérêt à poser $z = x + iy$ et $f(z) = f_1(x, y) + if_2(x, y)$. Dans ce cas la machine devient :



Nous appellerons états de la machine les couples (x, y). La machine apparait alors comme un graphe qui va agir sur les états. Ce graphe contient 4 type de noeuds : un noeud d'entrée, un noeud de sortie, un noeud de calcul qui modifie l'état de la machine (remplace (x, y) par $N_f(x, y)$) et un noeud de test. Cette machine définit une application partielle (c'est à dire non définie partout) de $\mathbb{R}^2$ dans $\mathbb{R}^2$ qui à une entrée (x, y) associe le premier couple $N_f^k(x, y)$ (s'il existe) dont l'image par f ait une norme euclidienne inférieure à ε.

Soit $\Omega = \{(x, y) \in \mathbb{R}^2 \mid \exists k \in \mathbb{Z},\ \|f(N_f^k(x, y)\| \leq \varepsilon\}$ (où l'on identifié f au couple (f_1, f_2)). Alors Ω est exactement l'ensemble des entrées telles que la machine s'arrête au bout d'un temps fini : on dit que c'est l'ensemble d'arrêt de la machine. C'est aussi le domaine de définition de la fonction associée à notre machine de Newton.

Pour $T \in \mathbb{N}$, posons $\Omega_T = \{(x, y) \in \mathbb{R}^2 \mid \exists k \leq T,\ \|f(N_f^k(x, y)\| \leq \varepsilon\}$. On a clairement $\Omega = \bigcup_{T \in \mathbb{N}} \Omega_T$. Mais d'autre part, Ω_T est un sous-ensemble de $\mathbb{R}^2$ qui est réunion finie de sous-ensembles de $\mathbb{R}^2$ définis par des inégalités polynomiales ; on dit qu'un tel ensemble est semi-algébrique. On voit donc que Ω est réunion dénombrable d'ensembles semi-algébriques sur $\mathbb{R}$.

3.3 Modèle de calcul : la dimension finie

Soit R un anneau ou un corps, éventuellement ordonné. Pour simplifier on parlera de R-espace vectoriel même dans le cas où R est un anneau (plutôt que de parler de R module). On appellera machine de dimension finie sur R un quadruplet $M = (R^n, R^m, R^\ell, \mathcal{G})$ constitué d'un espace d'entrées $I_M = R^n$, d'un espace d'états $S_M = R^m$, d'un espace de sorties $O_M = R^\ell$ et d'un graphe orienté $\mathcal{G}$ ayant les propriétés suivantes :

- le graphe a 4 sortes de noeuds : entrée, calculs, tests, sortie
- le noeud d'entrée η_0 est unique, n'a pas de flèche entrante et une seule flèche sortante ; il lui est associé une application linéaire $I : I_M \to S_M$ de l'espace des entrées dans l'espace des états de la machine
- chaque noeud η de sortie possède un nombre quelconque de flèches entrantes, n'a pas de flèche sortante et il lui est associé une application linéaire $O_\eta : S_M \to O_M$ de l'espace des états de la machine dans l'espace des sorties
- chaque état de calcul η possède un nombre quelconque de flèches entrantes, une seule flèche sortante et il lui est associé une application polynomiale (resp. rationnelle dans le cas d'un corps) $g_\eta : S_M \to S_M$ de l'espace des états dans lui même
- chaque noeud de test η possède un nombre quelconque de flèches entrantes, deux flèches sortantes, l'une étiquetée par $+$ l'autre par $-$; il lui est associé une application polynomiale $h_\eta : S_M \to R$ de l'espace des états dans l'anneau de base R

On voit que chaque noeud η possède un unique successeur β_η à l'exception des noeuds de tests qui en possèdent 2 que nous noterons β_η^+ et β_η^- et des noeuds de sortie qui n'en possèdent aucun. Nous poserons par convention que $\beta_\eta = \eta$ si η est un noeud de sortie.

On peut interpréter cette machine de la manière suivante : l'espace des entrées est l'espace des paramètres de la fonction définie par l'organigramme $\mathcal{G}$, l'espace des sorties est l'espace des valeurs de cette fonction, l'espace des états décrit la mémoire (ou si l'on veut les registres) de notre machine. Le noeud d'entrée place la mémoire dans un certain état après interprétation des paramètres, les noeuds de calcul transforment la mémoire, les noeuds de tests orientent le flot d'exécution en fonction d'un test binaire, les noeuds de sortie interprètent l'état de la machine pour retourner un résultat.

On notera $\mathcal{C}_M$ l'ensemble des coefficients des g_η et des h_η (l'ensemble des constantes de la machine) et d_M le maximum des degrés de ces fonctions (où le degré d'une fraction rationnelle est le maximum des degrés de son numérateur et de son dénominateur), que l'on appellera le degré de la machine.

Soit $\mathcal{N} = \{1, \ldots, N\}$ l'ensemble des noeuds de notre machine. L'ensemble $\mathcal{N} \times S$ sera appellé l'ensemble des configurations de M. On peut associer à notre machine M un *endomorphisme de calcul* $H : \mathcal{N} \times S \to \mathcal{N} \times S$ de l'espace des configurations dans lui même de la manière suivante :

- si η est le noeud d'entrée, alors $\forall x \in S_M,\ H(\eta, x) = (\beta_\eta, x)$
- si η est un noeud de calcul, alors $\forall x \in S_M,\ H(\eta, x) = (\beta_\eta, g_\eta(x))$
- si η est un noeud de sortie, alors $\forall x \in S_M,\ H(\eta, x) = (\eta, x)$
- si η est un noeud de test et si R est un anneau ordonné, alors

$$H(\eta, x) = \begin{cases} (\beta_\eta^+, x) & \text{si } h_\eta(x) \geq 0 \\ (\beta_\eta^-, x) & \text{si } h_\eta(x) < 0 \end{cases}$$

- si η est un noeud de test et si R est un anneau non ordonné, alors

$$H(\eta, x) = \begin{cases} (\beta_\eta^+, x) & \text{si } h_\eta(x) \neq 0 \\ (\beta_\eta^-, x) & \text{si } h_\eta(x) = 0 \end{cases}$$

A chaque entrée $x \in I_M$, on peut associer l'état $I(x) \in S_M$, et donc la configuration $z_0 = (\eta_0, I(x))$ de la machine. On peut alors définir la suite z_k des configurations successives de la machine par la relation de récurrence $z_{k+1} = H(z_k)$, ce qui définit un système dynamique dans l'ensemble des configurations. Posons donc $z_k = (\eta_k, s_k)$. Deux cas peuvent se produire :

- soit il existe k tel que η_k soit un état de sortie de la machine ; dans ce cas on note T le plus petit k qui vérifie cette propriété ; pour tout $p \geq T$, on a $z_p = z_k$ et la suite des configurations successives est stationnaire ; on dira que T est le temps d'arrêt de la machine M, que $(\eta_0, \ldots, \eta_T)$ est le chemin d'arrêt pour l'entrée x et on posera $\Phi_M(x) = O(s_T)$ et $T_M(x) = T$
- soit il n'existe pas de k tel que η_k soit un état de sortie de la machine, on dira que la machine ne s'arrête pas sur l'entrée x ; $\Phi_M(x)$ ne sera pas défini et on posera $T_M(x) = +\infty$.

L'ensemble des $x \in I_M$ tels que $T_M(x) < +\infty$ sera appelé l'ensemble d'arrêt de la machine M.

On obtient ainsi une application partielle Φ_M de l'espace des entrées dans l'espace des sorties. Une application ainsi définie sera dire calculable en dimension finie sur R et on dira que la machine M calcule Φ_M. On dira qu'un sous-ensemble X de l'espace des entrées est décidable si sa fonction caractéristique est calculable en dimension finie et qu'il est semi-décidable s'il existe d'une machine de dimension finie sur R ayant comme espace d'arrivée R telle que $\Phi_M(x) = 1 \iff x \in X$ (autrement dit, pour les $x \notin X$, soit $\Phi_M(x) \neq 1$, soit $\Phi_M(x)$ n'est pas défini).

On voit facilement en faisant tourner deux machines en alternance (une *instruction* de l'une, une *instruction* de l'autre, une *instruction* de l'une, une *instruction* de l'autre,...) en s'arrêtant dès que l'une des deux s'arrête, qu'un ensemble est décidable si et seulement si lui-même et son complémentaire sont semi-décidables.

De même, on constate facilement qu'un ensemble est semi-décidable si et seulement si c'est l'ensemble d'arrêt d'une fonction calculable :

- si l'ensemble est semi décidable, on prend une fonction Φ_M adéquate, associée à une machine M, on change son noeud de sortie, on boucle sur l'ancien noeud de sortie si le résultat n'est pas 1 et on renvoie 1 sinon
- si l'ensemble est ensemble d'arrêt de la machine M, on change son noeud de sortie de manière à renvoyer 1 dès que la machine s'arrête

3.4 Normalisations

Remarquons tout d'abord, que sans nuire à la généralité, on peut supposer que nos machines n'ont qu'un seul noeud de sortie : il suffit d'ajouter une dimension aux états et des sélecteurs pour fusionner toutes les sorties en une seule.

Dans le cas où on autorise les fonctions de calcul g_η à être des fractions rationnelles, certaines de ces fonctions risquent de n'être pas définies pour certaines valeurs des états. Il suffit pour *corriger le tir* d'introduire un test préalable de non nullité du dénominateur et de boucler sur ce test si le dénominateur est nul.

On peut supposer également que pour tout noeud de test η, la fonction de test associée h_η est la fonction de R^n dans R qui à tout état s associe sa première coordonnée. Il suffit en effet de rajouter une dimension (d'indice 0 par exemple) à l'espace des états de la machine et d'ajouter avant chaque noeud de test η un noeud de calcul η' ayant comme fonction de calcul $g_{\eta'}(s_0, s_1, \ldots, s_n) = (h_\eta(s_1, \ldots, s_n), s_1, \ldots, s_n)$, les autres noeuds ne modifiant jamais la composante d'indice 0.

On dira qu'une machine est normale si elle ne possède qu'un seul noeud de sortie, si les fonctions de calcul sont toujours définies sur toutes les entrées possibles et si les tests ne portent que sur la positivité (ou la non-nullité) de la première coordonnée. Comme on vient de le voir, toute machine est équivalente M est équivalente à une machine normale M'. On voit immédiatement que l'on peut prendre $S_{M'} = R \times S_M$ (la coordonnée ajoutée peut servir à la fois pour normaliser les tests et les sorties), que l'ensemble $\mathcal{C}_M$ des constantes de la machine n'est pas modifié, que le degré est au plus augmenté de 1 (à cause du sélecteur) et que le temps d'arrêt sur une entrée x est au plus multiplié par 2 (puisqu'au lieu de faire en un seul coup théorique le calcul de h_η et le test, on a procédé en deux étapes).

3.5 Ensembles semi-algébriques

On dira qu'un sous ensemble X de R^m est un ensemble semi-algébrique élémentaire s'il est défini par un nombre fini d'égalités et d'inégalités portant sur des polynômes à m variables. On dira qu'un ensemble est semi-algébrique s'il est réunion d'un nombre fini d'ensembles semi-algébriques élémentaires.

L'un des résultats fondamentaux démontrés par Blum, Shub et Smale est le résultat suivant :

Théorème 3.1 *Pour toute machine M sur l'anneau R et pour tout $T \in \mathbb{N}$, l'ensemble Ω_T des points x de l'espace des entrées qui vérifient $T_M(x) \leq T$ (c'est à dire l'ensemble des points tels que la machine s'arrête avant l'instant T) est une réunion finie disjointe d'ensembles semi-algébriques élémentaires. L'ensemble d'arrêt Ω_M de la machine est réunion disjointe dénombrable d'ensembles semi-algébriques élémentaires et la restriction de Φ_M à chacun de ces ensembles est une application polynomiale (resp. rationnelle).*

L'idée de la démonstration est de prendre une suite finie $\gamma = (\eta_0, \ldots, \eta_k)$ de noeuds successifs de la machine (autrement dit chacun est un successeur du précédent) et de considérer l'ensemble V_γ des x de l'espace des entrées tels que $z_0 = (\eta_0, *)$ (ce qui est automatique) et $\forall p \leq k,\ H^p(z_0) = (\eta_p, *)$ (autrement dit l'ensemble des x tels que la machine exécute successivement exactement les instructions décrites par les noeuds $\eta_0, \ldots, \eta_k$). Alors on montre sans difficulté par récurrence sur k que V_γ est soit vide, soit défini par un nombre fini d'égalités et d'inégalités polynomiales et que l'application qui à x associe l'état de la machine dans la configuration $H^k(z_0)$ est polynomiale ou rationnelle suivant le cas. Soit alors Γ_T l'ensemble (évidemment fini) des suites $\gamma = (\eta_0, \ldots, \eta_T)$ de tels noeuds successifs telles que η_T soit noeud de sortie de M. L'ensemble Ω_T n'est autre que la réunion disjointes des V_γ pour $\gamma \in \Gamma_T$ et M calcule sur chacun de ces V_γ une fonction polynomiale (ou rationnelle). Il suffit ensuite d'écrire que Ω est la réunion des Ω_T.

Des considérations topologiques (portant en particulier sur la dimension de Hausdorff de la frontière) montrent alors que certains ensembles ne peuvent pas être réunion dénombrable d'ensembles semi-algébriques, ce qui conduit aux résultats suivants :

Théorème 3.2 *L'ensemble de Mandelbrot n'est pas décidable sur $\mathbb{R}$.*

Théorème 3.3 *L'ensemble des points tels que la méthode de Newton converge n'est en général pas décidable sur $\mathbb{R}$.*

Dans le même ordre d'idée, on peut montrer, par une étude un peu fine de degrés, le résultat suivant qui montre que le problème du sac à dos est intrinsèquement difficile

Théorème 3.4 *Soit M une machine sur un corps infini non ordonné qui résoud le problème du sac à dos de taille n en un temps borné. Alors il existe un chemin pour lequel la somme des degrés des polynômes associés aux noeuds parcourus est au moins égale à $2^n - 1$.*

3.6 Modèle de calcul : la dimension infinie

Le modèle de calcul précédent ne s'adapte malheureusement qu'à des entrées de taille fixée et ne permet pas de formaliser une notion de complexité d'algorithme sur un anneau, c'est à dire à une estimation du temps d'arrêt en fonction de la taille des données. La solution naturelle est d'autoriser comme espace d'entrée l'ensemble $R^\infty = \bigcup_{n\geq 1} R^n$. Comme les sorties peuvent également être de longueurs quelconques, on est conduit à utiliser ce même espace comme espace des sorties.

On dispose d'une application naturelle φ de R^∞ dans l'espace $R^{(\mathbb{N})}$ des suites d'éléments de R à support fini (c'est à dire qui n'ont qu'un nombre fini d'éléments non nuls) définie par $\varphi(x_1,\ldots,x_n) = (x_1,\ldots,x_n,0,\ldots,0,\ldots)$ ce qui pourrait nous inciter à prendre comme espace des états d'une machine cet espace $R^{(\mathbb{N})}$. Malheureusement cette application φ n'est pas injective, autrement dit on ne peut pas récupérer à partir de l'état initial de la machine la taille des données d'entrée. L'introduction d'une coordonnée supplémentaire, par exemple d'indice 0, dans laquelle on stockerait initialement la taille des données, n'est pas une solution qui résiste à l'examen car l'anneau R n'est pas forcément de caractéristique 0.

Une méthode possible est d'ajouter au vecteur de $R^{(\mathbb{N})}$ un nombre n de coordonnées d'indices négatifs dans lesquelles on stockera des 1. L'espace des états devient alors naturellement l'espace $R^{(\mathbb{Z})}$ des suites indexées par $\mathbb{Z}$ d'éléments de R à supports finis. On dispose alors de deux applications de conversion entre l'espace $R^\infty = \bigcup_{n\geq 1} R^n$ et l'espace des états de la machine,, une pour les entrées et l'autre pour les sorties.

La fonction de conversion d'entrées sera l'application $I : R^\infty \to \mathbb{R}^{(\mathbb{Z})}$ telle que $I(x_1,\ldots,x_N)$ soit la suite $(y_n)_{n\in\mathbb{Z}}$ à support fini indexée par $\mathbb{Z}$ définie par $y_n = x_n$ si $1 \leq n \leq N$, $y_n = 1$ si $-N+1 \leq n \leq 0$ et $y_n = 0$ sinon. Autrement dit

$$I(x_1,\ldots,x_N) = (\ldots,0,\ldots,0,\overbrace{1,\ldots,1}^{N},x_1,\ldots,x_N,0,\ldots,0,\ldots)$$

La fonction de conversion de sortie sera en quelque sorte réciproque de la précédente, en ne gardant comme information que le plus petit indice d'une coordonnée négative non nulle pour traduire la taille des données. Autrement dit, on posera que $O((y_n)_{n\in\mathbb{Z}}) = (x_1,\ldots,x_N)$ si $y_n = 0$ pour $n \leq -N$, $y_{-N+1} \neq 0$ et $x_n = y_n$ pour $1 \leq n \leq N$, c'est à dire (en convenant que $a \neq 0$)

$$O\Big((\ldots,0,\ldots,0,\overbrace{a,\ldots\ldots}^{N},x_1,\ldots,x_N,x_{N+1},\ldots,x_P,0,\ldots)\Big) = (x_1,\ldots,x_N)$$

Il nous reste à définir les opérations autorisées sur l'espace $R^{(\mathbb{Z})}$ des états de la machine. Comme précédemment nous aurons des noeuds de calcul et des noeuds de tests qui utiliseront des applications polynomiales (voire rationnelles) de $R^{(\mathbb{Z})}$ dans $R^{(\mathbb{Z})}$ (pour les calculs) ou dans R (pour les tests). Il faut définir ce que nous entendons par applications polynomiales de $R^{(\mathbb{Z})}$ dans R : nous conviendrons qu'il s'agit d'un polynôme de $R[x_1,\ldots,x_n,\ldots]$ (une application polynomiale n'introduit que les variables d'indice supérieur à 1). Une application polynomiale de $R^{(\mathbb{Z})}$ dans lui-même sera définie par un nombre fini $f_1,\ldots,f_m$ d'applications polynomiales de $R^{(\mathbb{Z})}$ dans R en posant $F((y_n)_{n\in\mathbb{Z}}) = (z_n)_{n\in\mathbb{Z}}$ avec $z_n = f_n((y_n)_{n\in\mathbb{Z}})$ pour $1 \leq n \leq m$ et $z_n = y_n$ si $n \leq 0$ ou $n \geq m+1$ (autrement dit, les noeuds de calcul ou de tests n'ont pas un accès immédiat aux informations *de dimension* qui sont stockées dans les coordonnées d'indice négatifs).

Pour utiliser ou modifier les informations de *de dimension* qui sont stockées dans les coordonnées d'indices négatifs, il nous faut alors introduire un cinquième type de noeud dans notre machine : les noeuds de décalage permettront d'effectuer des décalages des états soit vers la gauche, soit vers la droite. Ces noeuds auront un nombre quelconque de flèches entrantes, une seule flèche sortante, et à chacun de ces noeuds sera associée une fonction de décalage σ^+ ou σ^- définie par $\sigma^+((y_n)_{n\in\mathbb{Z}}) = ((y_{n+1})_{n\in\mathbb{Z}})$ ou $\sigma^+((y_n)_{n\in\mathbb{Z}}) = ((y_{n-1})_{n\in\mathbb{Z}})$.

Toutes les notions introduites en dimension finie se généralisent sans problème : espace des configurations, endomorphisme de calcul, degré de la machine, ensemble d'arrêt, temps d'arrêt, ensemble décidable ou semi-décidable. La différence essentielle est que la taille des données n'est plus limitée et que l'on dispose d'une machine ayant une mémoire infinie (ou un nombre infini de registres). Une telle machine (qui s'adapte automatiquement à toutes les dimensions d'entrées) sera appelée une machine uniforme.

Blum, Sub et Smale montrent que sur des données de taille fixée, le fait d'avoir une mémoire infinie n'améliore pas le temps de calcul d'une machine, en démontrant le théorème suivant

Théorème 3.5 *Soit M une machine sur R de degré d. Soit $t : \mathbb{N} \to \mathbb{N}$ une fonction telle que, pour tout $n \in \mathbb{N}$ et tout $x \in R^n$ (c'est à dire tout entrée de taille n) le temps d'arrêt $T_M(x)$ correspondant à l'exécution de M avec x comme entrée, soit majoré par $t(n)$. Alors, pour tout $n \in \mathbb{N}$, il existe une machine de dimension finie M_n de même degré d, qui calcule sur R^n la même fonction que M, avec un temps d'arrêt également majoré par $t(n)$.*

3.7 Complexité algébrique et problèmes associés

Soit M une machine uniforme et A une partie de R^∞ contenue dans l'ensemble d'arrêt de M. A chaque $x \in A$, on peut associer l'entier taille(x) (c'est le n tel que $x \in R^n$) et le temps d'arrêt $T_M(x)$ de la machine M sur l'entrée x.

On dira qu'une machine travaille en temps polynomial sur les données de A s'il existe un entier $N \in \mathbb{N}$ et une constante $K > 0$ telle que

$$\forall x \in A,\ T_M(x) \leq K(\text{taille}(x))^N$$

On dira qu'un ensemble A est (uniformément) décidable en temps polynomial s'il existe une machine uniforme M qui calcule en temps polynomial la fonction caractéristique de A (autrement dit qui teste sur des données de taille quelconque, en temps polynomial, si elles sont ou non dans l'ensemble considéré). On appelle $\mathbf{P}_R$ l'ensemble des problèmes de décision sur l'anneau R qui sont résolubles en temps polynomial.

Des problèmes naturels se posent alors :

- certains problèmes classiques de l'informatique (sac à dos, satisfaisabilité d'une formule logique, voyageur de commerce), de l'algèbre (recherche de racines de polynômes) ou de l'analyse numérique (résolution approchée de systèmes linéaires ou algébriques) sont-ils résolubles en temps polynomial sur un anneau R ?
- comment varie la complexité d'un problème de décision en fonction de l'anneau que l'on considère : un problème booléen peut par exemple être traduit sur n'importe quel anneau puisqu'il n'introduit que des 0 et des 1, et on peut se demander si le problème de la satisfaisabilité est plus facile sur $\mathbb{Z}/2\mathbb{Z}$, sur $\mathbb{Z}$, sur $\mathbb{R}$ ou sur $\mathbb{C}$
- l'introduction de la non-uniformité (on suppose que l'on a une machine par taille au lieu de la même machine pour toutes les tailles, en imposant que le nombre de noeuds ne croisse pas trop vite) permet-elle de réduire la complexité d'un problème de décision, ceci suivant l'anneau R
- l'introduction d'un devin (ou du non-déterminisme) permet-elle de réduire la complexité d'un problème de décision, ceci suivant l'anneau R (c'est le fameux problème $P =?NP$ reformulé en $P_R =?NP_R$)

Certaines réponses, souvent partielles, sont données dans les deux ouvrages cités en référence. Pour ôter cependant tout espoir inconsidéré au lecteur, Bruno Poizat signale qu'il ne connait aucune structure R pour lesquelles $P_R = NP_R$. Il exhibe par contre des structures R pour lesquelles $P_R \neq NP_R$.

4 Bibliographie

Bruno POIZAT *Les petits cailloux*, Ed. Aléas (15 quai Lassagne 69001 LYON), ISBN 2-908016-58-3
Leonor BLUM, Felipe CUCKER, Michael SHUB, Steve SMALE *Complexity and real computation*, Ed. Springer, ISBN 0-387-98281-7