UNIVERSITÉ DU QUÉBEC À MONTRÉAL

Le dixième problème de Hilbert

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

ADRIEN LEMAÎTRE

JANVIER 2008

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

RÉSUMÉ

L'objet de ce travail est d'exposer la démonstration de l'indécidabilité du dixième problème de Hilbert fournie par Matiiassevitch dans son livre *Le dixième problème de Hilbert. Son indécidabilité* paru en 1995. Le problème consiste à fournir une méthode permettant de déterminer l'existence d'une solution en nombres entiers relatifs d'une équation polynomiale à coefficients entiers quelconque, dite *équation diophantienne.* Après un exposé des notions préliminaires, on réduit le problème aux entiers naturels. On s'attache ensuite à clarifier les mécanismes fournis par Matiiassevitch (1995) qui permettent de déterminer le caractère "diophantien" de l'exponentiation dans les entiers naturels. Ceci constitue la difficulté centrale de la démonstration. Sa résolution fut le fait de Matiiassevitch (1970). Ce fait établi, on s'attache à commenter et expliciter la nouvelle preuve fournie par Matiassevitch (1995) qui utilise des méthodes de codage s'appuyant sur l'exponentiation. On voit comment leur utilisation judicieuse permet de coder les machines de Turing à l'aide d'équations diophantiennes. Ceci mène à l'équivalence entre les ensembles semi-décidables et les ensembles codés par les équations diophantiennes. Enfin, on étudie la preuve de l'indécidabilité par l'utilisation d'une méthode diagonale portant sur les équations diophantiennes.

Mots-clés : décidabilité, dixième problème de Hilbert, équations diophantiennes, machine de Turing, Matiiassevitch.

INTRODUCTION

En 1900, lors du Congrès International des Mathématiciens de Paris, David Hilbert, un des plus grands mathématiciens de tous les temps, présenta une liste de 23 problèmes mathématiques ouverts qu'il jugeait centraux et qui allaient grandement influencer la recherche en mathématique du XXe siècle. L'énoncé du dixième problème de Hilbert est particulièrement simple :

> On donne une équation de Diophante à un nombre quelconque d'inconnues et à coefficients entiers rationnels : on demande de trouver une méthode par laquelle, au moyen d'un nombre fini d'opérations, on pourra distinguer si l'équation est résoluble en nombres entiers rationnels. (Hilbert, 1900)

Une équation de Diophante, ou équation diophantienne, est une équation polynomiale à coefficients entiers en un nombre quelconque d'inconnues à valeurs entières. Hilbert demandait un algorithme capable de décider si une telle équation diophantienne possède une solution. À l'époque, l'existence d'un tel algorithme n'aurait pu être prouvée que par sa description. En effet, il n'existait pas alors de système formalisant la notion intuitive de calculabilité. On ne disposait pas d'outils permettant de statuer sur la non-existence d'un algorithme particulier. Or, comme allait le montrer Matiiassevitch en 1970, il n'existe pas d'algorithme tel que demandé dans ce dixième problème.

La théorie de la calculabilité, développée dans les années 30 par des logiciens tels que Church, Turing ou Gödel, fournit des outils pour établir la non-existence d'algorithmes particuliers. Il fut dès lors possible de démontrer que certaines procédures ne peuvent pas être décrites par un algorithme tel que notre intuition le définit, c'est-à-dire une suite d'instructions simples. Le fameux problème de l'arrêt (Turing, 1937) est un exemple classique de problème impossible à résoudre au moyen d'un algorithme. Muni de ces outils, plusieurs chercheurs commencèrent à envisager la possibilité d'une solution négative au dixième problème.

La principale difficulté rencontrée lors de la résolution du dixième problème consistait à démontrer le caractère diophantien de l'exponentiation. Julia Robinson (1952) a établi que l'existence d'une relation diophantienne quelconque de croissance approximativement exponentielle impliquerait que l'exponentiation est diophantienne. Elle démontre également que le coefficient binomial, la factorielle et l'ensemble des nombres premiers seraient diophantiens si l'exponentiation l'était. En 1953, Davis montre que le complémentaire d'un ensemble diophantien n'est pas nécessairement diophantien et établit que tout prédicat semi-décidable est exprimable sous forme d'une expression logique utilisant un quantificateur borné et une équation diophantienne. S'appuyant sur ces travaux, Davis, Putnam et Robinson (1961) montrent que le problème de trouver une solution en nombres entiers à une équation exponentielle diophantienne (qui autorise l'exponentiation pour la construction du polynôme à résoudre) est indécidable. Pour prouver que le dixième problème de Hilbert est indécidable, il ne reste plus qu'à prouver que l'exponentiation est diophantienne. C'est l'apport de Matiiassevitch à la résolution du problème. En 1970, il publie un article démontrant le caractère diophantien de la relation $\phi_{2u} = v$, où $\phi_0, \phi_1, \ldots$ sont les nombres de Fibonacci, définis par la récurrence $\phi_0 = 0$, $\phi_1 = 1$ et $\phi_{n+2} = \phi_n + \phi_{n+1}$. Cette relation satisfaisant les critères énoncés par Julia Robinson (1952), Matiiassevitch peut alors conclure que l'exponentiation est diophantienne et qu'au regard des conclusions de Davis, Putnam et Robinson (1961), un algorithme tel que demandé par Hilbert n'existe pas.

Bien que le résultat soit négatif, plusieurs faits intéressants s'en dégagent ou ont été mis en évidence lors de sa démonstration. En particulier, l'équivalence entre les ensembles diophantiens et les ensembles semi-décidables, reconnus par les machines de Turing, offre une nouvelle caractérisation de ce qui est calculable ou pas. De plus, l'indécidabilité du dixième problème étant établie, il est possible de prouver l'indécidabilité d'autres problèmes en réduisant le dixième problème à ces problèmes.

L'objectif de ce travail est d'étudier la démonstration de l'indécidabilité du dixième problème de Hilbert fournie par Matiiassevitch. Nous nous baserons sur l'ouvrage écrit par Matiiassevitch (1995). Un exposé complet d'une preuve menant à l'indécidabilité du

dixième problème de Hilbert avait déjà été fait par Davis (1973), s'appuyant sur l'article de Davis, Putnam et Robinson (1961) et sur celui de Matiiassevitch (1970). Toutefois, l'approche de Davis (1973) est assez différente de celle de Matiiassevitch (1995) ; en particulier, Davis n'emploie pas les machines de Turing et se base largement sur *les formes de Davis*, une représentation particulière des ensembles diophantiens qui utilise le concept de quantificateur universel borné. La démonstration de Matiiassevitch (1995) semble plus accessible, se basant sur un modèle de calcul (la machine de Turing) relativement simple à manipuler et à comprendre. Toutefois, un certain nombre de détails non triviaux méritent d'être explicités. On s'attachera ainsi à clarifier cette démonstration de sorte qu'il ne reste plus de sujets d'interrogation à son endroit. La plupart des résultats abordés dans ce travail se trouvent dans le livre de Matiassevitch (1995) concernant le dixième problème de Hilbert. Matiiassevitch y expose les idées nécessaires à une preuve différente de la preuve originale tout en soulignant, par le biais de notes historiques, la contribution de Davis, Putnam et Robinson à la première preuve de l'indécidabilité du problème. Les ajouts qui ont été faits pour éclaircir l'exposé de ce livre sont signalés dans ce travail par un astérisque ($*$).

CHAPITRE I

NOTIONS PRÉLIMINAIRES

Pour étudier le dixième problème de Hilbert, nous emploierons divers outils et notions mathématiques relativement connus et documentés. L'objet central de cette étude est l'équation diophantienne, que nous définirons et dont nous exposerons diverses propriétés. En particulier, nous verrons comment le dixième problème de Hilbert peut être réduit aux équations diophantiennes à solutions positives ou nulles. On se servira ensuite des équations diophantiennes pour définir d'autres objets extrêmement importants, les ensembles diophantiens.

En premier lieu, nous allons rapidement passer en revue des notions que nous utiliserons par la suite.

1.1 Apports de la théorie des nombres

Le dixième problème de Hilbert concerne la théorie des nombres. Il n'est par conséquent pas étonnant d'utiliser, dans le cadre de sa résolution, des théorèmes eux-mêmes issus de cette branche. Nous aurons besoin, au cours de démonstrations ultérieures, d'utiliser troiÃs théorèmes importants. Ces résultats sont connus et facilement accessibles au lecteur désireux de prendre connaissance de leurs démonstrations. Celles-ci sont en particulier faites par Matiiassevitch (1995) dans l'ouvrage sur lequel nous nous basons. Nous ne donnerons donc pas ces démonstrations ici.

En sus de ces théorèmes, nous préciserons la notation du coefficient binomial que

nous utiliserons par la suite.

1.1.1 Coefficient binomial

Définition 1.1.1. *Pour tous entiers naturels n et $m \leq n$, on a $C_n^m = \frac{n!}{m!(n-m)!}$. On appelle ce nombre* coefficient binomial.

1.1.2 Lemme de Bezout

Ce résultat est relativement connu mais son appellation varie beaucoup. On le rencontre aussi sous le nom d'identité de Bezout. On peut trouver sa preuve dans des ouvrages sur la théorie des nombres (par exemple, De Koninck et Mercier, 1994, p. 6, théorème 1.12).

Lemme 1.1.1. *Soient a et b des nombres naturels. Soit $d = pgcd(a, b)$. Il existe des nombres relatifs x et y tels que $ax + by = d$.*

1.1.3 Théorème des quatre carrés de Lagrange

Ce résultat, aussi connu sous le nom de conjecture de Bachet, stipule que tout nombre naturel peut être exprimé comme somme de quatre carrés d'entiers relatifs.

Théorème 1.1.2 (Théorème des quatre carrés de Lagrange (1772))**.** *Soit n un entier naturel quelconque. Il existe alors quatre entiers relatifs a, b, c, d tels que*

$$n = a^2 + b^2 + c^2 + d^2.$$

1.1.4 Théorème des restes chinois

Ce théorème est bien connu en théorie des nombres (voir De Koninck et Mercier, 1994, section 3.4) ainsi qu'en théorie des groupes. Il permet de trouver un dividende a tel que la division de a par un certain nombre de naturels $q_1, \ldots, q_n$ fournit précisément les restes $r_1, \ldots, r_n$ déterminés au préalable.

Théorème 1.1.3 (Théorème des restes chinois). *Soient $q_1, \ldots, q_n$ des nombres entiers premiers deux à deux. Soient $r_1, \ldots, r_n$ des entiers tels que pour tout $i \in \{1, \ldots, n\}$, on a*

$$0 \leq r_i < q_i.$$

Alors, il existe un entier a unique tel que, pour tout $i \in \{1, \ldots, n\}$, le reste de la division euclidienne (voir section 1.8.2) de a par q_i est r_i et $0 \leq a < q_1.q_2 \ldots q_n$.

1.1.5 Théorème de Kummer

Il s'agit peut-être du moins connu des trois théorèmes énoncés ici. Il fournit un critère simple permettant de déterminer si un coefficient binomial C_{m+n}^n est divisible par un nombre premier p.

Théorème 1.1.4 (Théorème de Kummer (1852)). *Soit p un nombre premier quelconque. Soient m et n des nombres naturels quelconques. L'exposant de p dans la décomposition de C_{m+n}^n en produit de facteurs premiers est égal au nombre de retenues nécessaires pour additionner les nombres m et n représentés en base p.*

1.2 Équations et ensembles diophantiens

1.2.1 Équations diophantiennes

Nous allons maintenant définir les objets dont nous étudierons les propriétés dans le cadre de ce travail : les équations diophantiennes.

Définition 1.2.1. *Une équation diophantienne est une équation à n inconnues de la forme*

$$D(x_1, \ldots, x_n) = \sum c_{a_1,\ldots,a_n} x_1^{a_1} \ldots x_n^{a_n} = 0$$

où $n \geq 0$ et a_i est un entier ≥ 0 pour tout $i \in \{1, \ldots, n\}$ et $c_{a_1,\ldots,a_n}$ est un entier relatif pour tout $a_1, \ldots, a_n$. La sÃomme est finie. Le domaine des inconnues est l'ensemble des entiers relatifs.

La définition précédente est équivalente à la forme

$$D_L(x_1,\ldots,x_n) = \sum c_{L,a_1,\ldots,a_n} x_1^{a_1}\ldots x_n^{a_n} = \sum c_{R,a_1,\ldots,a_n} x_1^{a_1}\ldots x_n^{a_n} = D_R(x_1,\ldots,x_n)$$

où $n \geq 0$, $a_i \geq 0$ pour tout $i \in \{1,\ldots,n\}$ et pour tous $a_1,\ldots,a_n$, $c_{L,a_1,\ldots,a_n}$ et $c_{R,a_1,\ldots,a_n}$ sont des entiers naturels. Il est facile de passer de la première définition à la seconde : il suffit de placer les coefficients de part et d'autre de l'équation de manière à éliminer les valeurs négatives.

Définition 1.2.2. *Le* degré *d'un monôme est la somme des degrés des variables individuelles de ce monôme.*

Par exemple, le degré de x^2y^3z est 6.

Définition 1.2.3. *Le* degré *total d'un polynôme D est le degré du monôme de degré maximum parmi les monômes constituant D.*
Dénotons Deg *la fonction qui associe à tout polynôme son degré total.*

Par exemple, le degré de $x^3y^2z + x^5y^2$ est 7.

Définition 1.2.4. *Le* degré *d'une équation diophantienne $D(x_1,\ldots,x_n) = 0$ est le degré du polynôme D.*

1.2.2 Systèmes d'équations diophantiennes

Nous allons maintenant prouver que résoudre un système d'équations diophantiennes est équivalent à résoudre une équation diophantienne et que par conséquent, le dixième problème de Hilbert et le problème de l'existence d'une solution en nombres entiers d'un système d'équations diophantiennes sont équivalents. C'est ce que montre le lemme suivant.

Lemme 1.2.1. *Pour tout système d'équations diophantiennes, il existe une équation diophantienne qui possède une solution en nombres entiers si et seulement si le système en possède une.*

Démonstration. Soit un système de m équations diophantiennes

$$\begin{cases} D_1(x_1, \ldots, x_n) = 0 \\ \vdots \\ D_m(x_1, \ldots, x_n) = 0. \end{cases}$$

Résoudre ce système est équivalent à résoudre l'équation diophantienne suivante :

$$D_1^2(x_1, \ldots, x_n) + \cdots + D_m^2(x_1, \ldots, x_n) = 0.$$

□

Réciproquement, il est trivial de constater qu'une équation diophantienne constitue un système d'équations diophantiennes à une équation. Donc, résoudre une équation diophantienne revient à résoudre un système d'équations diophantiennes particulier. Le problème de l'existence d'une solution en nombres entiers d'un système d'équations diophantiennes est donc équivalent au dixième problème de Hilbert.

1.2.3 Solutions en entiers naturels

Le dixième problème de Hilbert propose les entiers relatifs comme domaine des inconnues. Davis (1953, p. 35-36) montre que ce problème peut être réduit au dixième problème de Hilbert restreint aux solutions entières naturelles.

Lemme 1.2.2. *Soit $D(x_1, \ldots, x_n) = 0$ une équation diophantienne quelconque. Il existe une équation diophantienne $E(y_1, \ldots, y_m) = 0$ telle que $D(x_1, \ldots, x_n) = 0$ possède une solution en entiers naturels si et seulement si $E(y_1, \ldots, y_m) = 0$ possède une solution en entiers relatifs.*

Démonstration. D'après le théorème des quatre carrés de Lagrange, tout entier naturel peut être écrit comme somme de quatre carrés d'entiers relatifs. Il existe donc des entiers relatifs $y_{1,1}, \ldots, y_{1,4}, \ldots, y_{n,1}, \ldots, y_{n,4}$ tels que

$$x_1 = y_{1,1}^2 + \cdots + y_{1,4}^2$$

$$\vdots$$

$$x_n = y_{n,1}^2 + \cdots + y_{n,4}^2.$$

D'après le lemme 1.2.1, il existe une équation diophantienne $E(z_1, \ldots, z_n) = 0$ admettant une solution si et seulement si le système d'équations diophantiennes

$$\begin{cases} D(x_1, \ldots, x_n) &= 0 \\ x_1 &= y_{1,1}^2 + \cdots + y_{1,4}^2 \\ &\vdots \\ x_n &= y_{n,1}^2 + \cdots + y_{n,4}^2 \end{cases}$$

admet une solution. Puisque les équations

$$\begin{aligned} x_1 &= y_{1,1}^2 + \cdots + y_{1,4}^2 \\ &\vdots \\ x_n &= y_{n,1}^2 + \cdots + y_{n,4}^2 \end{aligned}$$

admettent forcément une solution si $x_1, \ldots, x_n$ sont tous des entiers naturels et n'admettent aucune solution en entiers sinon, alors le système admet une solution si et seulement si $D(x_1, \ldots, x_m) = 0$ admet une solution en entiers naturels. Donc, l'équation diophantienne $E(z_1, \ldots, z_n) = 0$ admet une solution en nombres relatifs si et seulement si l'équation diophantienne $D(x_1, \ldots, x_m) = 0$ admet une solution en nombres naturels. □

Donc, si on peut déterminer l'existence d'une solution en entiers relatifs pour une équation diophantienne quelconque, on peut également déterminer l'existence d'une solution en entiers naturels pour une équation diophantienne quelconque. Prouver l'indécidabilité du problème restreint aux solutions naturelles suffit alors à montrer que le problème originel est indécidable. C'est ainsi que nous prouverons effectivement l'indécidabilité du dixième problème de Hilbert. Toutefois, qu'aurions-nous pu conclure au sujet de l'indécidabilité du dixième problème de Hilbert si le problème restreint aux solutions naturelles avait été décidable ? A priori rien. Nous allons toutefois montrer qu'en fait la

décidabilité du problème restreint aurait entraîné la décidabilité du problème original. En effet, si on peut établir la possibilité de déterminer l'existence d'une solution en entiers naturels pour une équation diophantienne, alors on peut établir la possibilité Ãde déterminer l'existence d'une solution en entiers relatifs pour une équation diophantienne quelconque.

Lemme 1.2.3. *Soit une équation diophantienne quelconque* $D(x_1, \ldots, x_n) = 0$ *dont les inconnues appartiennent aux entiers relatifs. Il existe une équation diophantienne* $E(y_1, \ldots, y_{2n}) = 0$ *dont les inconnues appartiennent aux entiers naturels telle qu'il existe une solution de* $D(x_1, \ldots, x_n) = 0$ *si et seulement s'il existe une solution de l'équation* $E(y_1, \ldots, y_{2n}) = 0$.

Démonstration. Soit $E(a_1, \ldots, a_n, b_1, \ldots, b_n) = D(a_1 - b_1, \ldots, a_n - b_n)$ le polynôme obtenu par substitution de l'expression $a_i - b_i$ à chaque inconnue x_i dans le polynôme $D(x_1, \ldots, x_n)$.
($\Rightarrow$) Soit $(x_1, \ldots, x_n)$ une solution de $D(x_1, \ldots, x_n) = 0$. Tout entier relatif peut être exprimé comme la différence de deux entiers naturels. Par conséquent, il existe $a_1, b_1, \ldots, a_n, b_n$ tels que

$$\begin{aligned} x_1 &= a_1 - b_1, \\ &\vdots \\ x_n &= a_n - b_n. \end{aligned}$$

On a donc

$$D(x_1, \ldots, x_n) = D(a_1 - b_1, \ldots, a_n - b_n) = E(a_1, \ldots, a_n, b_1, \ldots, b_n) = 0.$$

Il existe une solution en nombres naturels de

$$E(a_1, \ldots, a_n, b_1, \ldots, b_n) = 0.$$

($\Leftarrow$) Soit $(a_1, \ldots, a_n, b_1, \ldots, b_n)$ une solution de l'équation

$$E(a_1, \ldots, a_n, b_1, \ldots, b_n) = 0$$

Soient $x_1, \ldots, x_n$ les entiers relatifs définis par

$$\begin{aligned} x_1 &= a_1 - b_1, \\ &\vdots \\ x_n &= a_n - b_n. \end{aligned}$$

On a donc

$$E(a_1, \ldots, a_n, b_1, \ldots, b_n) = D(a_1 - b_1, \ldots, a_n - b_n) = D(x_1, \ldots, , x_n) = 0.$$

Il existe une solution en nombres relatifs de

$$D(x_1, \ldots, x_n) = 0.$$

□

Donc, si déterminer l'existence de solutions en nombres entiers naturels pour toute équation diophantienne est possible, alors il est également possible de déterminer l'existence d'une solution en nombres entiers relatifs pour toute équation diophantienne. En conclusion, prouver l'indécidabilité ou la décidabilité du problème restreint aux solutions naturelles revient à prouver respectivement l'indécidabilité ou la décidabilité du problème originel. Par conséquent, nous ne considérerons plus que le problème restreint pour la suite de la démonstration. Les inconnues appartiendront donc à l'ensemble des naturels. Ce choix est fait pour faciliter certains détails techniques dans les preuves à venir.

1.2.4 Familles d'équations diophantiennes

Nous allons considérer des ensembles particuliers d'équations diophantiennes.

Définition 1.2.5. *Soit une équation diophantienne quelconque*

$$D(a_1, \ldots, a_n, x_1, \ldots, x_m) = 0.$$

On appelle famille d'équations diophantiennes *l'ensemble des équations diophantiennes*

$$D(a_1, \ldots, a_n, x_1, \ldots, x_m) = 0$$

dans lesquelles on a substitué des valeurs entières aux inconnues $a_1, \ldots, a_n$. *On appelle ces inconnues paramètres de l'équation. On parle également d'équation diophantienne paramétrée.*

Par exemple, l'équation

$$ax^2 + bx + c = 0$$

peut être considérée comme une équation diophantienne possédant une inconnue x et trois paramètres a,b, et c. La famille d'équations diophantiennes représentée par cette équation est donc la famille des équations de degré au plus 2.

1.3 Ensembles diophantiens

Nous aurons à représenter des ensembles d'uplets de nombres entiers par des équations diophantiennes.

Définition 1.3.1. *Pour tout* $n \geq 0$, *on dit qu'un ensemble* E *de* n*-uplets entiers naturels est diophantien si et seulement s'il existe une équation diophantienne*

$$D(y_1, \ldots, y_n, x_1, \ldots, x_m) = 0$$

où $y_1, \ldots, y_n$ *sont des paramètres et* $x_1, \ldots, x_m$ *des inconnues, telle qu'un* n*-uplet* $(a_1, \ldots, a_n)$ *d'entiers appartient à* E *si et seulement s'il existe des entiers* $x_1, \ldots, x_m$ *satisfaisant l'équation*

$$D(a_1, \ldots, a_n, x_1, \ldots, x_m) = 0.$$

On appelle n *la* dimension *de l'ensemble* E. *Un ensemble dont le complémentaire est diophantien est dit* co-diophantien.

L'ensemble diophantien associé à une équation diophantienne particulière est donc l'ensemble des valeurs de paramètres pour lesquelles cette équation a une solution. Nous pouvons donc représenter un ensemble au moyen d'une équation diophantienne. Par exemple, l'équation

$$2x = a,$$

où a est un paramètre et x une inconnue, représente l'ensemble des nombres entiers pairs. L'équation

$$a = b$$

où a et b sont des paramètres, représente l'ensemble des couples de valeurs égales.

On notera qu'ici, les valeurs prises par les inconnues pour satisfaire l'équation ne nous intéressent pas ; seul le fait qu'une solution existe ou non pour les paramètres choisis est pertinent.

On remarquera finalement qu'à tout ensemble diophantien correspond une infinité d'équations diophantiennes. En effet, dès qu'une équation possède une solution, il est possible d'en dériver une infinité d'équations en ajoutant des inconnues auxquelles on affectera une valeur nulle pour se ramener à l'équation initiale. Par exemple, si l'équation

$$D(x_1, \ldots, x_n) = 0$$

admet une solution, il est évident que l'équation

$$D(x_1, \ldots, x_n) + y = 0$$

admet une solution pour $y = 0$.

1.4 Relations diophantiennes

Nous avons dans la section précédente défini les ensembles diophantiens, représentables au moyen d'équations diophantiennes. Il est donc naturel de représenter des relations au moyen de ces équations. Dans la théorie ensembliste, une relation n-aire est simplement un ensemble de n-uplets. Il est donc aisé de définir une relation diophantienne.

Définition 1.4.1. *Une relation n-aire R est diophantienne s'il existe une équation diophantienne à n paramètres et m inconnues,*

$$D(y_1, \ldots, y_n, x_1, \ldots, x_m) = 0,$$

où $y_1, \ldots, y_n$ sont des paramètres et $x_1, \ldots, x_m$ sont des inconnues, telle que $R(a_1, \ldots, a_n)$ est satisfaite si et seulement s'il existe $x_1, \ldots, x_m$ tels que

$$D(a_1, \ldots, a_n, x_1, \ldots, x_m) = 0.$$

Si on reprend les exemples de la section précédente, on constate que la relation « est pair » est diophantienne. De même, la relation d'égalité est diophantienne.

1.5 Fonctions diophantiennes

De même que les relations, les fonctions peuvent être considérées comme des ensembles d'uplets dans la théorie ensembliste.

Définition 1.5.1. *Une fonction*

$$f : \mathbb{N}^m \to \mathbb{N}^n$$

est dite diophantienne s'il existe une équation diophantienne

$$D(y_1, \ldots, y_m, z_1, \ldots, z_n, x_1, \ldots, x_p) = 0,$$

où $y_1, \ldots, y_m, z_1, \ldots, z_n$ sont des paramètres et $x_1, \ldots, x_p$ sont des inconnues, telle que pour tout $a_1, \ldots, a_m, b_1, \ldots, b_n$ on a

$$f(a_1, \ldots, a_m) = (b_1, \ldots, b_n)$$

si et seulement s'il existe $x_1, \ldots, x_p$ tels que

$$D(a_1, \ldots, a_m, b_1, \ldots, b_n, x_1, \ldots, x_p) = 0.$$

Par exemple, l'équation diophantienne

$$a = b^2$$

où a et b sont des paramètres, représente la fonction diophantienne

$$f(x) = x^2.$$

Il existe des fonctions dont le caractère diophantien est beaucoup moins évident, notamment l'exponentiation qui fera l'objet du second chapitre.

1.6 Opérations sur les ensembles diophantiens

Puisqu'il est possible de représenter des ensembles au moyen d'équations diophantiennes, il est logique de s'interroger sur la réalisation de l'union et de l'intersection d'ensembles dans cette représentation. La clôture des ensembles diophantiens sous l'union et la disjonction est exposée par Davis (1953) comme une conséquence de la clôture des prédicats polynômiaux sous ces mêmes opérations. La preuve qui est donnée ici est une adaptation de ÃÃÃces résultats. Les opérations d'union et de disjonction concerneront des ensembles de même dimension. Comme nous allons le voir ultérieurement, le complémentaire d'un ensemble diophantien n'est pas nécessairement diophantien (Davis, 1953).

1.6.1 Union

L'obtention d'une équation diophantienne représentant l'union de deux ensembles diophantiens A et B se fait en utilisant la multiplication de polynômes.

Lemme 1.6.1. *Soit A l'ensemble diophantien de dimension n représenté par l'équation*

$$D_A(y_1, \ldots, y_n, x_1, \ldots, x_m) = 0.$$

Soit B l'ensemble diophantien de dimension n représenté par l'équation

$$D_B(y_1, \ldots, y_n, z_1, \ldots, z_p) = 0.$$

Alors, l'ensemble $A \cup B$ est diophantien, de dimension n et représenté par l'équation diophantienne à n paramètres et $m + p$ inconnues

$$D_A(y_1, \ldots, y_m, x_1, \ldots, x_m) \cdot D_B(y_1, \ldots, y_n, z_1, \ldots, z_p) = 0.$$

Démonstration. Soit E l'ensemble diophantien représenté par l'équation diophantienne

$$D_A(y_1, \ldots, y_n, x_1, \ldots, x_m) \cdot D_B(y_1, \ldots, y_n, z_1, \ldots, z_p) = 0.$$

Soit $(a_1, \ldots, a_n) \in A$. Par définition de A, il existe alors $x_1, \ldots, x_m$ tels que

$$D_A(a_1, \ldots, a_n, x_1, \ldots, x_m) = 0.$$

Par conséquent, on a

$$D_A(a_1, \ldots, a_n, x_1, \ldots, x_m) \cdot D_B(a_1, \ldots, a_n, z_1, \ldots, z_p) = 0.$$

De même, si $(b_1, \ldots, b_n) \in B$, alors il existe $z_1, \ldots, z_p$ tels que

$$D_A(a_1, \ldots, a_n, x_1, \ldots, x_m) \cdot D_B(a_1, \ldots, a_n, z_1, \ldots, z_p) = 0.$$

Donc, $A \cup B \subseteq E$.
Soit $(e_1, \ldots, e_n) \in E$. Par définition de E, il existe $x_1, \ldots, x_m, z_1, \ldots, z_p$ tels que

$$D_A(e_1, \ldots, e_n, x_1, \ldots, x_m) \cdot D_B(e_1, \ldots, e_n, z_1, \ldots, z_p) = 0.$$

On a donc $D_A(e_1, \ldots, e_n, x_1, \ldots, x_m) = 0$ ou $D_B(e_1, \ldots, e_n, z_1, \ldots, z_p) = 0$. Par définition de A et B, on a alors $(e_1, \ldots, e_n) \in A$ ou $(e_1, \ldots, e_n) \in B$. Donc, $E \subseteq A \cup B$.
Par conséquent, on a $A \cup B = E$. □

1.6.2 Intersection

L'obtention d'une équation diophantienne représentant l'intersection de deux ensembles diophantiens A et B se fait en utilisant l'addition et le carré de polynômes.

Lemme 1.6.2. *Soit A l'ensemble diophantien représenté par l'équation*

$$D_A(y_1, \ldots, y_n, x_1, \ldots, x_m) = 0.$$

Soit B l'ensemble diophantien représenté par l'équation

$$D_B(y_1, \ldots, y_n, z_1, \ldots, z_p) = 0.$$

Alors, l'ensemble $A \cap B$ est diophantien et représenté par l'équation diophantienne

$$D_A^2(y_1, \ldots, y_n, x_1, \ldots, x_m) + D_B^2(y_1, \ldots, y_n, z_1, \ldots, z_p) = 0.$$

Démonstration. Soit E l'ensemble diophantien représenté par l'équation diophantienne

$$D_A^2(y_1, \ldots, y_n, x_1, \ldots, x_m) + D_B^2(y_1, \ldots, y_n, z_1, \ldots, z_p) = 0.$$

Soit $(a_1, \ldots, a_n) \in A \cap B$. Par définition de A, il existe alors $x_1, \ldots, x_m$ tels que

$$D_A(a_1, \ldots, a_n, x_1, \ldots, x_m) = 0.$$

Par définition de B, il existe alors $z_1, \ldots, z_p$ tels que

$$D_B(a_1, \ldots, a_n, z_1, \ldots, z_p) = 0.$$

Par conséquent,

$$D_A^2(a_1, \ldots, a_n, x_1, \ldots, x_m) + D_B^2(a_1, \ldots, a_n, z_1, \ldots, z_p) = 0.$$

Donc, par définition de E, $(a_1, \ldots, a_n) \in E$. On a donc $A \cap B \subseteq E$.
Soit $(e_1, \ldots, e_n) \in E$. Par définition de E, il existe $x_1, \ldots, x_m, z_1, \ldots, z_p$ tels que

$$D_A^2(e_1, \ldots, e_n, x_1, \ldots, x_m) + D_B^2(e_1, \ldots, e_n, z_1, \ldots, z_p) = 0.$$

On a donc $D_A^2(e_1, \ldots, e_n, x_1, \ldots, x_m) = 0$ et $D_B^2(e_1, \ldots, e_n, z_1, \ldots, z_p) = 0$.
Par conséquent, on a $D_A(e_1, \ldots, e_n, x_1, \ldots, x_m) = 0$ et $D_B(e_1, \ldots, e_n, z_1, \ldots, z_p) = 0$.
Par définition de A et B, on a alors $(e_1, \ldots, e_n) \in A$ et $(e_1, \ldots, e_n) \in B$. Donc, $E \subseteq A \cap B$.
Par conséquent, on a $A \cap B = E$. □

1.7 Construction d'ensembles diophantiens à partir de termes diophantiens

Pour prouver qu'un ensemble, une relation ou une fonction est diophantien, il faut exhiber l'équation diophantienne le représentant. Toutefois, il n'est pas toujours pratique de le faire. Ainsi, si les équations représentant l'ensemble des nombres pairs ou la relation de divisibilité sont simples, les équations représentant l'ensemble des nombres premiers ou l'exponentiation sont, elles, beaucoup plus volumineuses, beaucoup trop pour être lisibles et utilisables dans les démonstrations. On opte alors pour des représentations alternatives des ensembles, relations et fonctions diophantiennes.

1.7.1 Expressions logiques utilisant relations et fonctions diophantiennes

On peut construire une expression logique à partir de relations et de fonctions que l'on sait diophantiennes. Par exemple, les relations $Multiple3(a)$ et $Multiple5(a)$ signifiant respectivement que a est un multiple de 3 et a est un multiple de 5 sont représentés par les équations diophantiennes suivantes

$$Multiple3(a) \Leftrightarrow \exists x[a = 3x]$$

$$Multiple5(a) \Leftrightarrow \exists x[a = 5x].$$

On peut alors représenter la relation $Multiple15(a)$ signifiant que a est un multiple de 15 au moyen de l'équation suivante

$$Multiple15(a) \Leftrightarrow Multiple3(a) \wedge Multiple5(a).$$

Nous avons vu que l'intersection et l'union d'ensembles diophantiens sont diophantiennes. Puisque la conjonction et la disjonction de relations peuvent être respectivement vues comme une intersection et une union d'ensembles, alors une expression logique construite au moyen des opérateurs $\wedge$ et $\vee$ et de relations diophantiennes représente un ensemble diophantien. De même, une expression construite par adjonction d'un quantificateur existentiel devant une relation ou fonction diophantienne décrit un ensemble diophantien. Soit $R(a_1, \ldots, a_n)$ une relation diophantienne et $D(a_1, \ldots, a_m, x_1, \ldots, x_m) = 0$ l'équation diophantienne telle que

$$R(a_1, \ldots, a_n) \Leftrightarrow \exists x_1, \ldots, x_m D(a_1, \ldots, a_m, x_1, \ldots, x_m) = 0.$$

On a alors

$$\exists w R(a_1, \ldots, a_n) \Leftrightarrow \exists w \exists y_1, \ldots, y_m D(a_1, \ldots, a_m, y_1, \ldots, y_m) = 0$$

que w soit un des paramètres $a_1, \ldots, a_n$, une des inconnues $y_1, \ldots y_m$ ou une inconnue quelconque. L'expression $\exists w R(a_1, \ldots, a_n)$ représente donc un ensemble diophantien. Nous n'avons pas besoin d'expliciter l'équation diophantienne correspondant à

l'ensemble pour le considérer comme diophantien. Nous pouvons décrire un ensemble diophantien en construisant une expression logique employant la conjonction, la disjonction et le quantificateur existentiel.

1.7.2 Composition de fonctions diophantiennes

La notation introduite dans la dernière section permet de mettre en évidence que la composition de fonctions diophantiennes est diophantienne. En effet, l'expression

$$a = f(\mathbf{t_1}, \ldots, \mathbf{t_n})$$

où $\mathbf{t_1}, \ldots, \mathbf{t_n}$ sont des fonctions diophantiennes est équivalente à

$$\exists t_1, \ldots, t_m \left[a = f(t_1, \ldots, t_m) \wedge t_1 = \mathbf{t_1} \wedge \cdots \wedge t_m = \mathbf{t_m}\right].$$

1.8 Quelques fonctions et relations diophantiennes

Par la suite, nous nous servirons de quelques relations et fonctions dont le caractère diophantien est plus ou moins évident. Nous les décrivons brièvement ici, en nous attardant sur les moins évidentes.

1.8.1 Inégalités

Lemme 1.8.1. *Soient a, b et x des entiers naturels. On a alors*

$$a \leq b \Leftrightarrow \exists x[a + x = b].$$

Lemme 1.8.2. *Soient a, b et x des entiers naturels. On a alors*

$$a < b \Leftrightarrow \exists x[a + x + 1 = b].$$

Lemme 1.8.3. *Soient a, b et x des entiers naturels. On a alors*

$$a \neq b \Leftrightarrow a < b \vee b < a.$$

1.8.2 Division euclidienne

Un résultat standard d'arithmétique (De Koninck et Mercier, 1994, p. 5) nous apprend que pour tout couple d'entiers naturels a et $b > 0$, il existe d'uniques q et r tels que $a = bq + r$ et $r < b$. On appelle q le quotient et r le reste de la division euclidienne de a par b. Si r est nul, on dit que b divise a et on écrit $b \mid a$.
Rappelons la notation suivante : $\mathbb{N}^*$ est l'ensemble des entiers naturels dont on a exclu 0.

Lemme 1.8.4. *Soient a, $b > 0$ et q des entiers naturels. On a alors*

$$b \mid a \Leftrightarrow \exists q[bq = a]$$

Définition 1.8.1. *Soit* rem *la fonction de $\mathbb{N} \times \mathbb{N}^*$ dans $\mathbb{N}$ qui à un entier naturel a et un entier naturel non nul b associe le reste de la division euclidienne de a par b.*

Lemme 1.8.5. *Soient a, $b > 0$ et r des entiers naturels. On a alors*

$$r = rem(a, b) \Leftrightarrow r < b \wedge b \mid a - r.$$

Démonstration. ($\Rightarrow$) Soit $r = rem(a, b)$. Par définition de rem, on a $r < b$. De plus, il existe q tel que $a = bq + r$, donc $a - r = bq$ et b divise $a - r$.
($\Leftarrow$) Soit $r < b$ et $b \mid a - r$. Donc, il existe q tel que $a - r = bq$, donc $a = bq + r$. Puisque $r < b$, alors par unicité du quotient et du reste d'une division euclidienne, on a $r = rem(a, b)$. □

Définition 1.8.2. *Soit* div *la fonction de $\mathbb{N} \times \mathbb{N}^*$ dans $\mathbb{N}$ qui à un entier naturel a et un entier naturel non nul b associe le quotient de la division euclidienne de a par b.*

Lemme 1.8.6. *Pour tous naturels a, b, q, on a*

$$q = a \text{ div } b \Leftrightarrow bq + rem(a, b) = a.$$

Démonstration. ($\Rightarrow$) Soit q le quotient de la division euclidienne de a par b. Par définition, il existe $r < b$ tel que $a = qb + r$ et $r = rem(a, b)$.

($\Leftarrow$) Soit $bq + rem(a,b) = a$. Par définition, $rem(a,b) < b$ et puisqu'il existe d'uniques q et $r < b$ tels que $a = bq + r$, alors q est le quotient de la division euclidienne de a par b. □

Nous concluerons en remarquant que la relation de congruence est diophantienne (De Koninck et Mercier, 1994, chap. 3).

Lemme 1.8.7. *Pour tous entiers naturels a et b, on a*

$$a \equiv b \pmod{p} \Leftrightarrow rem(a,p) = rem(b,p).$$

1.8.3 PGCD, PPCM

Puisqu'il a été démontré que la divisibilité est diophantienne, on peut maintenant démontrer que la fonction $pgcd : \mathbb{N}^* \times \mathbb{N}^* \to \mathbb{N}^*$ qui aux entiers naturels a et b associe le plus grand commun diviseur de a et b est diophantienne.

Lemme 1.8.8. *Pour tous naturels $a > 0$, $b > 0$ et $d > 0$, on a*

$$d = pgcd(a,b) \Leftrightarrow (a > 0) \wedge (b > 0) \wedge (d \mid a) \wedge (d \mid b) \wedge \exists x \exists y [d = ax - by],$$

où x, y sont des entiers naturels.

Démonstration. ($\Rightarrow$) Soit $d = pgcd(a,b)$. Par définition $a > 0$, $b > 0$, $d \mid a$ et $d \mid b$. D'après le lemme de Bezout (lemme 1.1.2), il existe des entiers relatifs x et y tels que $ax + by = d$. Puisque $d \mid a$, il existe un entier naturel q différent de 0 tel que $a = qd$. Puisque d divise a et b, alors $d \leq a, b$. Donc, on a forcément $x \leq 0$ ou $y \leq 0$, sinon on aurait $d = ax + by > a, b$. On a de plus $x > 0$ ou $y > 0$, sinon on aurait $d = ax + by \leq 0$ alors que $d \geq 1$. On a deux possibilités :

- si $y \leq 0$, alors soit $v = -y$ et $u = x$. On a $au - bv = ax + by = d$.
- si $y > 0$, alors on a deux possibilités :

- si $x = 0$, on a $ax + by = by = d$. Puisque d divise b, alors $d \leq b$, on a donc $y = 1$. Par conséquent, $b = d$. Soit $u > 0$ un entier quelconque. Soit $v = qu - 1$. Puisque $u > 0$ et $q > 0$, alors $v \geq 0$. On a

$$\begin{aligned} ua - vb &= ua - (qu - 1)b \\ &= ua - qub + b \\ &= ua - qud + b. \end{aligned}$$

Or, on a $a = qd$, d'où

$$\begin{aligned} ua - vb &= ua - qud + b \\ &= ua - ua + b \\ &= b \\ &= d. \end{aligned}$$

- si $x < 0$, soit $u = qx^2$ et $v = -qxy - y$. Puisque $y, q > 0$ et $x < 0$, alors $u, v \geq 0$. On a

$$\begin{aligned} au - bv &= aqx^2 - b(-qxy - y) \\ &= aqx^2 + bqxy + by \\ &= qx(ax + by) + by \\ &= qxd + by. \end{aligned}$$

Puisque $qd = a$, on a

$$\begin{aligned} au - bv &= ax + by \\ &= d. \end{aligned}$$

Dans tous les cas, il existe des entiers naturels u et v tels que $au - bv = d$.

($\Leftarrow$) Soient $a > 0$, $b > 0$, $d \mid a$, $d \mid b$ et $x, y \in \mathbb{N}$ tels que $d = ax - by$. Puisque

$d \mid a$ et $d \mid b$, alors il existe $q_1, q_2 \in \mathbb{N}$ tels que $a = q_1 d$ et $b = q_2 d$. On a alors

$$\begin{aligned} d &= ax - by \\ d &= q_1 dx - q_2 dy \\ 1 &= q_1 x - q_2 y. \end{aligned}$$

D'après le lemme de Bezout (lemme 1.1.2), q_1 et q_2 sont donc premiers entre eux. Par conséquent, puisque $a = q_1 d$ et $b = q_2 d$, alors le plus grand commun diviseur de a et b est d. □

La fonction *pgcd* est donc diophantienne puisqu'elle peut être définie par construction d'une expression logique ne comprenant que des termes diophantiens. On peut maintenant utiliser une propriété bien connue (De Koninck et Mercier, 1994, p. 12, théorème 1.26) du plus grand commun diviseur et du plus petit commun multiple pour donner une définition diophantienne de la fonction $ppcm : \mathbb{N}^* \times \mathbb{N}^* \rightarrow \mathbb{N}^*$ associant à deux entiers naturels a et b leur plus grand commun multiple.

Lemme 1.8.9.

$$c = ppcm(a, b) \Leftrightarrow ab = pgcd(a, b)c$$

Les notions d'équations diophantiennes et d'ensembles diophantiens étant maintenant définies, nous pouvons nous pencher sur la résolution du dixième problème de Hilbert lui-même. Rappelons qu'on ne s'intéresse plus au problème tel qu'originellement formulé, mais au problème modifié qui ne concerne que les équations diophantiennes à solutions naturelles.

CHAPITRE II

L'EXPONENTIATION EST DIOPHANTIENNE

Prouver que l'exponentiation est diophantienne fut la dernière étape de la preuve de l'indécidabilité du dixième problème de Hilbert. En effet, dès 1961, Davis, Putnam et Robinson avaient prouvé le caractère indécidable de l'existence de solutions pour une équation diophantienne exponentielle. Ils en concluaient naturellement que si l'exponentiation elle-même est diophantienne, alors les équations diophantiennes exponentielles ne sont qu'un cas particulier des équations diophantiennes et que par conséquent, le dixième problème de Hilbert est lui-même indécidable. Matiiassevitch démontra en 1970 le caractère diophantien d'une relation à croissance exponentielle, satisfaisant les critères énoncés par Julia Robinson (1952) et par conséquent le caractère diophantien de l'exponentiation (Robinson, 1952).

La démonstration originale de Matiassevitch (1970) se basait sur la suite de Fibonacci. Ce n'est pas cette suite que Matiassevitch emploie dans son livre (1995). Il utilise une suite α, définie par $\alpha_b(0) = 0$, $\alpha_b(1) = 1$ et $\alpha_b(n+2) = b\alpha_b(n+1) - \alpha_b(n)$, qui affiche quelque similitude avec la suite γ, définie par $\gamma_b(0) = 0$, $\gamma_b(1) = 1$ et $\gamma_b(n+2) = b\gamma_b(n+1) + \gamma_b(n)$, dont la croissance est aussi exponentielle (Davis, 1971). Le comportement de la suite α_b et son caractère diophantien formeront le coeur de ce chapitre.

Pour commencer, introduisons une nouvelle fonction qui sera utile dans cette section.

2.1 La fonction *arem*

Nous aurons besoin d'identifier la plus petite valeur absolue d'un nombre congru à x modulo b. C'est cette valeur que dénote $arem(x, b)$.

Lemme 2.1.1. *Pour tous entiers naturels $x \geq 0$ et $b \geq 1$, il existe un unique entier naturel a tel que $a \equiv \pm x \pmod{b}$ et $0 \leq a \leq \frac{b}{2}$.*

Démonstration. Soient q et r le quotient et le reste de la division euclidienne de x par b. On a alors $x = qb + r$ avec $0 \leq r < b$.

On a alors deux possibilités :

- Si $r \leq \frac{b}{2}$, soit $a = r$. On a alors $x - a = qb$, d'où $a \equiv x \pmod{b}$.
- Si $r > \frac{b}{2}$, soit $a = b - r$. On a alors $x + a = (q+1)b$, d'où $a \equiv -x \pmod{b}$. Puisque $\frac{b}{2} < r < b$, on a $0 < b - r < \frac{b}{2}$.

Prouvons maintenant qu'un tel a est unique. Soient a et a' tels que $a \equiv \pm x \pmod{b}$, $0 \leq a \leq \frac{b}{2}$, $a' \equiv \pm x \pmod{b}$ et $0 \leq a' \leq \frac{b}{2}$. Ceci signifie que $(\pm a) + (\pm a') = kb$, pour un entier relatif k.

On a maintenant quatre cas à considérer, dépendamment du signe de a et du signe de a' dans l'expression $(\pm a) + (\pm a') = kb$.

Si $a + a' = kb$, on a alors

$$\begin{aligned} 0 &\leq a + a' \leq b \\ 0 &\leq kb \leq b \\ 0 &\leq k \leq 1. \end{aligned}$$

Si $k = 0$, on a $a + a' = 0$, donc $a = a' = 0$.

Si $k = 1$, on a $a + a' = b$, donc $a = a' = \frac{b}{2}$.

Si $-a - a' = kb$, on a alors

$$\begin{aligned} -b &\leq -a - a' \leq 0 \\ -b &\leq kb \leq 0 \\ -1 &\leq k \leq 0. \end{aligned}$$

Si $k = 0$, on a $-a - a' = 0$, donc $a = a' = 0$.

Si $k = -1$, on a $-a - a' = b$, donc $a = a' = \frac{b}{2}$.

Si $a - a' = kb$, on a alors

$$\begin{array}{rcccl} -\frac{b}{2} & \leq & a - a' & \leq & \frac{b}{2} \\ -\frac{b}{2} & \leq & kb & \leq & \frac{b}{2} \\ -\frac{1}{2} & \leq & k & \leq & \frac{1}{2}. \end{array}$$

Donc, $k = 0$ et on a $a - a' = 0$, donc $a = a'$.

Si $-a + a' = kb$, on a alors

$$\begin{array}{rcccl} -\frac{b}{2} & \leq & -a + a' & \leq & \frac{b}{2} \\ -\frac{b}{2} & \leq & kb & \leq & \frac{b}{2} \\ -\frac{1}{2} & \leq & k & \leq & \frac{1}{2}. \end{array}$$

Donc, $k = 0$ et on a $-a + a' = 0$, donc $a = a'$. Dans tous les cas, $a = a'$. Il y a bien unicité. □

Ce lemme établit que les contraintes $a \equiv \pm x \pmod{b}$ et $0 \leq a \leq \frac{b}{2}$ constituent une fonction de $\mathbb{N} \times \mathbb{N}^*$ dans $\mathbb{N}$. On peut alors écrire la définition suivante.

Définition 2.1.1. *La fonction* $arem : \mathbb{N} \times \mathbb{N}^* \to \mathbb{N}$ *est définie comme suit :*

$$arem(x, b) \equiv \pm x \pmod{b} \quad et \quad 0 \leq arem(x, b) \leq b/2 \ .$$

On constate que d'après cette définition, *arem* est une fonction diophantienne.

Lemme 2.1.2. *Soient* a, b *et* $c > 0$ *des entiers naturels. Si* $a \equiv \pm b \pmod{c}$, *alors* $arem(a, c) = arem(b, c)$.

Démonstration. Soit $x = arem(a, c)$.

On a : $x \equiv \pm a \pmod{c}$.

Or, $a \equiv \pm b \pmod{c}$, d'où les quatre possibilités suivantes :

- $x \equiv a \pmod{c}$ et $a \equiv b \pmod{c}$ alors $x \equiv b \pmod{c}$;
- $x \equiv -a \pmod{c}$ et $a \equiv b \pmod{c}$ alors $x \equiv -b \pmod{c}$;
- $x \equiv a \pmod{c}$ et $a \equiv -b \pmod{c}$ alors $x \equiv -b \pmod{c}$;
- $x \equiv -a \pmod{c}$ et $a \equiv -b \pmod{c}$ alors $x \equiv b \pmod{c}$.

Dans tous les cas, $x \equiv \pm b \pmod{c}$ et $2x \leq c$ puisque $x = arem(a, c)$. Par définition, on a donc $x = arem(b, c)$. □

2.2 La suite α_b

La suite α_b qui nous intéresse ici est une suite à croissance exponentielle. Julia Robinson avait établi (1952) deux critères qui, satisfaits par une relation diophantienne à croissance exponentielle, permettaient de conclure immédiatement à l'existence d'une définition diophantienne de l'exponentiation. Lorsque Matiiassevitch établit le caractère diophantien de α_b (1995), il n'utilise pas directement le résultat de Julia Robinson (1952) et prouve que cette suite peut être utilisée pour donner une définition diophantienne de l'exponentiation en s'appuyant sur la notion de limite.

Dans un premier temps, définissons à nouveau la suite α_b.

Définition 2.2.1. *Soit $b \geq 2$ et soit la fonction α_b définie comme suit :*

$$\begin{aligned} &\alpha_b(0) = 0, \\ &\alpha_b(1) = 1, \\ &\alpha_b(n+2) = b\alpha_b(n+1) - \alpha_b(n). \end{aligned}$$

La suite α_b possède un certain nombre de propriétés qui seront utiles ultérieurement et que nous allons exposer dès maintenant.

Lemme 2.2.1. *Pour tout entier naturel $n \geq 0$, on a $\alpha_2(n) = n$.*

Démonstration. On procède par récurrence sur n.

- Pour $n = 0$, on a $\alpha_2(0) = 0$. La propriété est vérifiée.

- Pour $n = 1$, on a $\alpha_2(1) = 1$. La propriété est vérifiée.
- Pour $n > 1$, on a $\alpha_2(n) = 2\alpha_2(n-1) - \alpha_2(n-2)$. Par hypothèse de récurrence, on a $\alpha_2(n-1) = n-1$ et $\alpha_2(n-2) = n-2$. Donc,

$$\alpha_2(n) = 2(n-1) - (n-2) = 2n - 2 - n + 2 = n.$$

  La propriété est vérifiée.

□

Lemme 2.2.2. *Pour tout entier naturel $n \geq 0$ et tout entier naturel $b \geq 2$, on a $\alpha_b(n) \leq (b-1)\alpha_b(n) < \alpha_b(n+1)$.*

Démonstration (*). Puisque $b \geq 2$, alors on a $\alpha_b(n) \leq (b-1)\alpha_b(n)$.
Pour prouver que $(b-1)\alpha_b(n) < \alpha_b(n+1)$, on procède par récurrence sur n.

- Pour $n = 0$, on a $(b-1)\alpha_b(n) = 0 < 1 = \alpha_b(1)$.
- Pour $n = 1$, on a $(b-1)\alpha_b(n) = b - 1 < b = \alpha_b(2)$.
- Pour $n > 1$, on a : $\alpha_b(n) = b\alpha_b(n-1) - \alpha_b(n-2)$. Par hypothèse de récurrence, $\alpha_b(n-2) < \alpha_b(n-1)$, d'où

$$\begin{aligned} -\alpha_b(n-2) &> -\alpha_b(n-1) \\ b\alpha_b(n-1) - \alpha_b(n-2) &> b\alpha_b(n-1) - \alpha_b(n-1) \\ b\alpha_b(n-1) - \alpha_b(n-2) &> (b-1)\alpha_b(n-1) \\ \alpha_b(n) &> (b-1)\alpha_b(n-1). \end{aligned}$$

Dans tous les cas, la propriété est vérifiée. □

Lemme 2.2.3. *Pour tout entier naturel $n \geq 0$ et tout entier naturel $b \geq 2$, on a $\alpha_b(n) \geq n$.*

Démonstration. On procède par récurrence sur n.

- Pour $n = 0$, on a $\alpha_b(n) = 0 = n$.
- Pour $n = 1$, on a $\alpha_b(n) = 1 = n$.

- Pour $n = k + 2$, on a, d'après le lemme 2.2.2,

$$\begin{aligned} \alpha_b(k+2) &= b\alpha_b(k+1) - \alpha_b(k) \\ &> b\alpha_b(k+1) - \alpha_b(k+1) \\ &= (b-1)\alpha_b(k+1). \end{aligned}$$

Or, par hypothèse de récurrence, on a $\alpha_b(k+1) \geq k+1$, d'où

$$\begin{aligned} \alpha_b(k+2) &> (b-1)\alpha_b(k+1) \\ &\geq (b-1)(k+1) \\ &= bk + b - k - 1 \\ &= (bk-k) + (b-1). \end{aligned}$$

On a $b \geq 2$, donc $(bk-k) \geq k$. Donc, on a

$$\alpha_b(k+2) > (bk-k) + (b-1) \geq k + (b-1).$$

Par conséquent, $\alpha_b(k+2) \geq k+b$. Puisque $b \geq 2$, alors $\alpha_b(k+2) \geq k+b \geq k+2$.

□

Le lemme suivant est intéressant puisqu'il permet d'exhiber un élément de la suite α_b sans avoir à faire explicitement appel à α_b. Il s'agit d'une propriété très utile qui permet de faire intervenir des termes de la suite α_b en utilisant uniquement des opérations que l'on sait diophantiennes.

Lemme 2.2.4. *Soit un entier naturel $b \geq 2$. Soient x et y des entiers naturels. Alors $x^2 - bxy + y^2 = 1$ et $x < y$ si et seulement s'il existe m tel que $x = \alpha_b(m)$ et $y = \alpha_b(m+1)$.*

Démonstration. ($\Leftarrow$) Soit m tel que $x = \alpha_b(m)$ et $y = \alpha_b(m+1)$. D'après le lemme 2.2.2, on a $x < y$.

On procède par récurrence sur m pour prouver que $x^2 - bxy + y^2 = 1$.

- Pour $m = 0$, on a $x = 0$ et $y = 1$ et $x^2 - bxy + y^2 = 0 - 0 + 1 = 1$.

- Pour $m = 1$, on a $x = 1$ et $y = b$ et $x^2 - bxy + y^2 = 1 - b^2 + b^2 = 1$.
- Pour $m = k + 2$, on a

$$\begin{aligned}
x^2 - bxy + y^2 &= \alpha_b^2(k+2) - b\alpha_b(k+2)\alpha_b(k+3) + \alpha_b^2(k+3) \\
&= \alpha_b^2(k+2) \\
&\quad -b\alpha_b(k+2)(b\alpha_b(k+2) - \alpha_b(k+1)) \\
&\quad +(b\alpha_b(k+2) - \alpha_b(k+1))^2 \\
&= \alpha_b^2(k+2) \\
&\quad -b^2\alpha_b^2(k+2) + b\alpha_b(k+2)\alpha_b(k+1) \\
&\quad +b^2\alpha_b^2(k+2) - 2b\alpha_b(k+2)\alpha_b(k+1) + \alpha_b^2(k+1) \\
&= \alpha_b^2(k+2) - b\alpha_b(k+2)\alpha_b(k+1) + \alpha_b^2(k+1) \\
&= (b\alpha_b(k+1) - \alpha_b(k))^2 \\
&\quad -b\alpha_b(k+1)(b\alpha_b(k+1) - \alpha_b(k)) \\
&\quad +\alpha_b^2(k+1) \\
&= b^2\alpha_b^2(k+1) - 2b\alpha_b(k+1)\alpha_b(k) + \alpha_b^2(k) \\
&\quad -b^2\alpha_b^2(k+1) + b\alpha_b(k+1)\alpha_b(k) \\
&\quad +\alpha_b^2(k+1) \\
&= \alpha_b^2(k+1) - b\alpha_b(k+1)\alpha_b(k) + \alpha_b^2(k).
\end{aligned}$$

  Par hypothèse de récurrence, $\alpha_b^2(k+1) - b\alpha_b(k+1)\alpha_b(k) + \alpha_b^2(k) = 1$. On a donc $\alpha_b^2(k+2) - b\alpha_b(k+2)\alpha_b(k+3) + \alpha_b^2(k+3) = 1$.

($\Rightarrow$) Soient x et y tels que $x < y$ et $x^2 - bxy + y^2 = 1$. Montrons par récurrence sur x qu'il existe m tel que $x = \alpha_b(m)$ et $y = \alpha_b(m+1)$.

- Pour $x = 0$, on a $x^2 - bxy + y^2 = y^2 = 1$. Donc $y = 1$ et puisque $m = 0$, on a $x = \alpha_b(m)$ et $y = \alpha_b(m+1)$, l'yopthèse est vérifiée.
- Pour $x > 0$, on a

$$\begin{aligned}
x^2 - bxy + y^2 &= 1 \\
y = bx + \frac{1}{y} - \frac{x^2}{y}&.
\end{aligned}$$

On a $\frac{x^2}{y} \geq \frac{1}{y}$ puisque $x \geq 1$. Donc $\frac{1}{y} - \frac{x^2}{y} \leq 0$.
On a alors $y = bx + \frac{1}{y} - \frac{x^2}{y} \leq bx$. Puisque $x < y$, on a $\frac{x^2}{y} < x$ et $\frac{1}{y} < 1$. On a alors $0 < \frac{x^2}{y} < x$ et $0 < \frac{1}{y} < 1$. Donc, $0 \leq \frac{x^2}{y} - \frac{1}{y} < x$. On a alors $y = bx + \frac{1}{y} - \frac{x^2}{y} > bx - x$, d'où $x > bx - y$.

Soit $y_1 = x$ et $x_1 = bx - y$.
On a

$$\begin{aligned} x_1^2 - bx_1y_1 + y_1^2 &= (bx - y)^2 - b(bx - y)x + x^2 \\ &= b^2x^2 - 2bxy + y^2 - b^2x^2 + byx + x^2 \\ &= x^2 - bxy + y^2 \\ &= 1 \end{aligned}$$

De plus, on a $x = y_1 > x_1$ puisque $x > bx - y = x_1$.
On utilise donc l'hypothèse de récurrence pour conclure qu'il existe m tel que

$$y_1 = \alpha_b(m + 1)$$

et

$$x_1 = \alpha_b(m).$$

On obtient $x = y_1 = \alpha_b(m + 1)$.
Nous avons également

$$\begin{aligned} x_1 &= bx - y \\ \alpha_b(m) &= b\alpha_b(m + 1) - y \\ y &= b\alpha_b(m + 1) - \alpha_b(m) \\ y &= \alpha_b(m + 2) \end{aligned}$$

Soit $m_1 = m + 1$. On a alors $x = \alpha_b(m_1)$ et $y = \alpha_b(m_1 + 1)$.

□

Définition 2.2.2. *On dit que deux nombres entiers a et b sont* premiers entre eux *si et seulement si leur plus grand commun diviseur est* 1.

Lemme 2.2.5. *Pour tout entier naturel $n > 0$ et tout entier naturel $b \geq 2$, $\alpha_b(n)$ et $\alpha_b(n+1)$ sont premiers entre eux.*

Démonstration. On procède par récurrence sur n.

- Pour $n = 1$, on a : $\alpha_b(n) = 1$ et $\alpha_b(n+1) = b$. La propriété est vérifiée.
- Pour $n > 1$, soit d le plus grand diviseur commun de $\alpha_b(n)$ et $\alpha_b(n-1)$. Puisque $\alpha_b(n+1)$ et $\alpha_b(n)$ sont divisibles par d et que $\alpha_b(n+1) = b\alpha_b(n) - \alpha_b(n-1)$, alors $\alpha_b(n-1)$ est aussi divisible par d. Par hypothèse de récurrence, le plus grand diviseur commun à $\alpha_b(n)$ et $\alpha_b(n-1)$ est 1. Par conséquent, on a $d = 1$, ce qui signifie que $\alpha_b(n)$ et $\alpha_b(n+1)$ sont premiers entre eux.

□

Lemme 2.2.6. *Soient b_1 et $b_2 \geq 2$ des entiers naturels. Soit un entier naturel $q \geq 1$. Pour tout entier naturel $n \geq 0$, si $b_1 \equiv b_2 \pmod{q}$, alors $\alpha_{b_1}(n) \equiv \alpha_{b_2}(n) \pmod{q}$.*

Démonstration (*). On procède par récurrence sur n.

- Pour $n = 0$, on a $\alpha_{b_1}(0) = \alpha_{b_2}(0) = 0$, donc $\alpha_{b_1}(0) \equiv \alpha_{b_2}(0) \pmod{q}$.
- Pour $n = 1$, on a $\alpha_{b_1}(1) = \alpha_{b_2}(1) = 1$, donc $\alpha_{b_1}(1) \equiv \alpha_{b_2}(1) \pmod{q}$.
- Pour $n = k + 2$, on a

$$\begin{aligned} \alpha_{b_1}(k+2) - \alpha_{b_2}(k+2) &= b_1\alpha_{b_1}(k+1) - \alpha_{b_1}(k) - b_2\alpha_{b_2}(k+1) + \alpha_{b_2}(k) \\ &= b_1\alpha_{b_1}(k+1) - b_2\alpha_{b_2}(k+1) - (\alpha_{b_1}(k) - \alpha_{b_2}(k)). \end{aligned}$$

Par hypothèse de récurrence, $\alpha_{b_1}(k) \equiv \alpha_{b_2}(k) \pmod{q}$ et $\alpha_{b_1}(k+1) \equiv \alpha_{b_2}(k+1) \pmod{q}$. Puisque $\equiv$ est une relation de congruence, on a alors :

$$\begin{aligned} \alpha_{b_1}(k+1) &\equiv \alpha_{b_2}(k+1) \pmod{q} \\ b_1\alpha_{b_1}(k+1) &\equiv \alpha_{b_2}(k+1) \pmod{q} \\ b_1\alpha_{b_1}(k+1) - \alpha_{b_1}(k) &\equiv \alpha_{b_2}(k+1) - \alpha_{b_2}(k) \pmod{q} \\ \alpha_{b_1}(k+2) &\equiv \alpha_{b_2}(k+2) \pmod{q}. \end{aligned}$$

□

Lemme 2.2.7. *Soient les entiers naturels $b \geq 4$ et $m \geq 1$. On a*

$$\alpha_b(m+1) - \alpha_b(m-1) < \alpha_b(m+2) - \alpha_b(m).$$

Démonstration. On procède par récurrence sur m.

- Pour $m = 1$, on a

$$\alpha_b(2) - \alpha_b(0) = b - 0 = b$$

et

$$\alpha_b(3) - \alpha_b(1) = b^2 - 1 - 1 = b^2 - 2.$$

Puisque $b \geq 4$, alors $b^2 - 2 > b$ et la propriété est vérifiée.

- Pour $m = 2$, on a

$$\alpha_b(3) - \alpha_b(1) = b^2 - 1 - 1 = b^2 - 2$$

et

$$\alpha_b(4) - \alpha_b(2) = b^3 - 2b - b = b^3 - 3b.$$

Puisque $b \geq 4$, alors $b^3 - 3b > b^2 - 2$ et la propriété est vérifiée.

- Pour $m = k + 3$, avec $k \geq 0$, on a

$$\begin{aligned} \alpha_b(k+5) - \alpha_b(k+3) &= b\alpha_b(k+4) - \alpha_b(k+3) - (b\alpha_b(k+2) - \alpha_b(k+1)) \\ &= b(\alpha_b(k+4) - \alpha_b(k+2)) - (\alpha_b(k+3) - \alpha_b(k+1)). \end{aligned}$$

Par hypothèse de récurrence, on a

$$\alpha_b(k+4) - \alpha_b(k+2) > \alpha_b(k+3) - \alpha_b(k+1).$$

Donc,

$$\alpha_b(k+4) - \alpha_b(k+2) - (\alpha_b(k+3) - \alpha_b(k+1)) \geq 1.$$

Puisque $b > 2$, on a alors

$$\begin{aligned} & b(\alpha_b(k+4) - \alpha_b(k+2)) \\ -(\alpha_b(k+3) - \alpha_b(k+1)) &= (b-1)(\alpha_b(k+4) - \alpha_b(k+2)) \\ &\quad +(\alpha_b(k+4) - \alpha_b(k+2)) - (\alpha_b(k+3) - \alpha_b(k+1)) \\ &\geq (b-1)(\alpha_b(k+4) - \alpha_b(k+2)) + 1 \\ &> (b-1)(\alpha_b(k+4) - \alpha_b(k+2)) \\ &> \alpha_b(k+4) - \alpha_b(k+2). \end{aligned}$$

Finalement, on a bien

$$\alpha_b(k+5) - \alpha_b(k+3) > \alpha_b(k+4) - \alpha_b(k+2).$$

Donc, pour tout $m \geq 1$, on a

$$\alpha_b(m+2) - \alpha_b(m) > \alpha_b(m+1) - \alpha_b(m-1).$$

□

Lemme 2.2.8. *Soient les entiers naturels $b \geq 4$ et $m > 1$. On a*

$$\alpha_b(m+1) - \alpha_b(m-1) > b.$$

Démonstration. On procède par récurrence sur m.

- Pour $m = 2$, on a $\alpha_b(3) - \alpha_b(1) = b^2 - 1 - 1 = b^2 - 2$. Puisque $b \geq 4$, alors $b^2 - 2 > b$ et la propriété est vérifiée.
- Pour $m > 2$, on a $m = k + 3$, avec $k \geq 0$. D'après le lemme 2.2.7, on a

  $$\alpha_b(k+4) - \alpha_b(k+2) > \alpha_b(k+3) - \alpha_b(k+1).$$

  De plus, par hypothèse de récurrence,

  $$\alpha_b(k+3) - \alpha_b(k+1) > b.$$

  La propriété est vérifiée.

La propriété est donc vraie pour tout $m > 1$. □

2.3 Représentation du premier ordre de la fonction α_b

On utilise maintenant une représentation matricielle de la suite α_b pour obtenir une récursivité du premier ordre, plus aisée à manipuler que la forme précédemment fournie.

Définition 2.3.1. *On étend la définition de la suite α_b à -1 : $\alpha_b(-1) = -1$.*
Soient les matrices de format 2×2 A_b, Ξ_b et I définies comme suit pour tout entier naturel n :

$$A_b(n) = \begin{pmatrix} \alpha_b(n+1) & -\alpha_b(n) \\ \alpha_b(n) & -\alpha_b(n-1) \end{pmatrix}$$

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

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Lemme 2.3.1. *Pour tout entier $n \geq 0$, on a $A_b(n) = \Xi_b^n$.*

Démonstration $(*)$. On procède par récurrence sur n.

- Pour $n = 0$, on a

$$A_b(0) = \begin{pmatrix} \alpha_b(1) & -\alpha_b(0) \\ \alpha_b(0) & -\alpha_b(-1) \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I = \Xi_b^0.$$

- Pour $n > 0$, on a

$$\begin{aligned} A_b(n) &= \begin{pmatrix} \alpha_b(n+1) & -\alpha_b(n) \\ \alpha_b(n) & -\alpha_b(n-1) \end{pmatrix} \\ &= \begin{pmatrix} b\alpha_b(n) - \alpha_b(n-1) & -b\alpha_b(n-1) + \alpha_b(n-2) \\ \alpha_b(n) & -\alpha_b(n-1) \end{pmatrix} \\ &= \begin{pmatrix} b & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha_b(n) & -\alpha_b(n-1) \\ \alpha_b(n-1) & -\alpha_b(n-2) \end{pmatrix} \end{aligned}$$

Par hypothèse de récurrence, $A_b(n-1) = \begin{pmatrix} \alpha_b(n) & -\alpha_b(n-1) \\ \alpha_b(n-1) & -\alpha_b(n-2) \end{pmatrix} = \Xi_b^{n-1}$.

On a donc

$$\begin{aligned} A_b(n) &= \begin{pmatrix} b & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha_b(n) & -\alpha_b(n-1) \\ \alpha_b(n-1) & -\alpha_b(n-2) \end{pmatrix} \\ &= \begin{pmatrix} b & -1 \\ 1 & 0 \end{pmatrix} \Xi_b^{n-1} \\ &= \Xi_b \Xi_b^{n-1} \\ &= \Xi_b^n. \end{aligned}$$

On a bien $A_b(n) = \Xi_b^n$.

Donc $A_b(n) = \Xi_b^n$ pour tout $n \geq 0$. □

Lemme 2.3.2. *Pour tout entier naturel $n \geq 0$, on a $det(A_b(n)) = 1$.*

Démonstration (*). On a $\det(\Xi_b) = b.0 - 1.(-1) = 1$. Le lemme 2.3.1 et que $Det(A \cdot B) = Det(A) \cdot Det(B)$ (Anton, 2005, chap. 2) nous permettent alors de conclure que

$$\det(A_b(n)) = \det(\Xi_b^n) = (\det(\Xi_b))^n = 1.$$

□

Lemme 2.3.3. *Pour tout entier naturel n, on a*

$$A_b^{-1}(n) = \begin{pmatrix} -\alpha_b(n-1) & \alpha_b(n) \\ -\alpha_b(n) & \alpha_b(n+1) \end{pmatrix}.$$

Démonstration. D'après le lemme 2.3.2, $\det(A_b(n)) = 1$. D'après un résultat classique d'algèbre linéaire (Anton, 2005, chap. 2), ceci signifie que $A_b(n)$ est inversible. Il est alors facile de vérifier que

$$A_b^{-1}(n) = \begin{pmatrix} -\alpha_b(n-1) & \alpha_b(n) \\ -\alpha_b(n) & \alpha_b(n+1) \end{pmatrix}.$$

En effet, on a

$$\begin{aligned} A_b(n)A_b^{-1}(n) &= \begin{pmatrix} \alpha_b(n+1) & -\alpha_b(n) \\ \alpha_b(n) & -\alpha_b(n-1) \end{pmatrix} \cdot \begin{pmatrix} -\alpha_b(n-1) & \alpha_b(n) \\ -\alpha_b(n) & \alpha_b(n+1) \end{pmatrix} \\ &= \begin{pmatrix} -\alpha_b(n-1) & \alpha_b(n) \\ -\alpha_b(n) & \alpha_b(n+1) \end{pmatrix} \cdot \begin{pmatrix} \alpha_b(n+1) & -\alpha_b(n) \\ \alpha_b(n) & -\alpha_b(n-1) \end{pmatrix} \\ &= \begin{pmatrix} a & 0 \\ 0 & a \end{pmatrix}, \end{aligned}$$

avec $a = \alpha_b^2(n) - \alpha_b(n+1)\alpha_b(n-1)$.
On a $a = \alpha_b^2(n) - \alpha_b(n+1)\alpha_b(n-1) = \alpha_b^2(n) - b\alpha_b(n)\alpha_b(n-1) + \alpha_b(n-1)$. D'après le lemme 2.2.4, $a = \alpha_b^2(n) - b\alpha_b(n)\alpha_b(n-1) + \alpha_b(n-1) = 1$. □

Le lemme suivant est un élément important de la démonstration qui va suivre. En effet, il nous permet d'établir la relation de divisibilité entre $\alpha_b(k)$ et m sans nécessairement connaître la valeur de m, une situation dans laquelle nous nous retrouverons plus tard.

Lemme 2.3.4. *Pour tout $b \geq 2$, tout entier naturel $k \geq 0$ et tout entier naturel $m \geq 0$, on a $\alpha_b^2(k) \mid \alpha_b(m) \Rightarrow \alpha_b(k) \mid m$.*

Démonstration. Si $k = 0$, alors la partie gauche de l'implication est fausse, ce qui implique que l'implication est vraie.
Si $k > 0$, soient $b \geq 2$ et $m \geq 0$ tels que $\alpha_b^2(k) \mid \alpha_b(m)$.

Soient q et r le quotient et le reste de la division euclidienne de m par k. On a donc $m = qk + r$ et $0 \leq r < k$. Considérons les équivalences suivantes :

$$\begin{aligned}
&\begin{pmatrix} \alpha_b(m+1) & -\alpha_b(m) \\ \alpha_b(m) & -\alpha_b(m-1) \end{pmatrix} \\
&= A_b(m) \\
&= \Xi_b^m \\
&= \Xi_b^{qk+r} \\
&= \Xi_b^r \Xi_b^{qk} \\
&= \Xi_b^r (\Xi_b^k)^q \\
&= A_b(r) A_b^q(k) \\
&= \begin{pmatrix} \alpha_b(r+1) & -\alpha_b(r) \\ \alpha_b(r) & -\alpha_b(r-1) \end{pmatrix} \cdot \begin{pmatrix} \alpha_b(k+1) & -\alpha_b(k) \\ \alpha_b(k) & -\alpha_b(k-1) \end{pmatrix}^q \\
&\equiv \begin{pmatrix} \alpha_b(r+1) & -\alpha_b(r) \\ \alpha_b(r) & -\alpha_b(r-1) \end{pmatrix} \cdot \begin{pmatrix} \alpha_b(k+1) & 0 \\ 0 & -\alpha_b(k-1) \end{pmatrix}^q \pmod{\alpha_b(k)} \\
&\equiv \begin{pmatrix} \alpha_b(r+1) & -\alpha_b(r) \\ \alpha_b(r) & -\alpha_b(r-1) \end{pmatrix} \cdot \begin{pmatrix} \alpha_b^q(k+1) & 0 \\ 0 & (-1)^q \alpha_b^q(k-1) \end{pmatrix} \pmod{\alpha_b(k)} \\
&\equiv \begin{pmatrix} \alpha_b(r+1)\alpha_b^q(k+1) & (-1)^q \alpha_b(r)\alpha_b^q(k-1) \\ \alpha_b(r)\alpha_b^q(k+1) & (-1)^{q+1} \alpha_b(r-1)\alpha_b^q(k-1) \end{pmatrix} \pmod{\alpha_b(k)}.
\end{aligned}$$

On peut alors écrire l'équivalence :

$$\alpha_b(m) \equiv \alpha_b(r)\alpha_b^q(k+1) \pmod{\alpha_b(k)}.$$

Par conséquent, on a

$$\alpha_b(k) \mid \alpha_b(m) - \alpha_b(r)\alpha_b^q(k+1).$$

On sait que $\alpha_b^2(k)$ divise $\alpha_b(m)$, donc $\alpha_b(k)$ divise également $\alpha_b(m)$. Par conséquent, $\alpha_b(k)$ divise $\alpha_b(r)(\alpha_b^q(k+1))$. D'après le lemme 2.2.5, $\alpha_b(k)$ et $\alpha_b(k+1)$ sont premiers entre eux. Donc, $\alpha_b(k)$ et $(\alpha_b(k+1))^q$ sont également premiers entre eux. Par conséquent, $\alpha_b(k)$ divise forcément $\alpha_b(r)$.

On a alors deux possibilités : soit $r = 0$ et $\alpha_b(r) = 0$, soit $r > 0$ et $\alpha_b(r) > 0$.

- Si $r > 0$, alors puisque $\alpha_b(k)$ divise $\alpha_b(r)$, on a $\alpha_b(k) \leq \alpha_b(r)$, ce qui implique, d'après le lemme 2.2.2, que $k \leq r$. Or, on sait que $r < k$. Donc, il est impossible d'avoir simultanément $r > 0$ et $\alpha_b(k) \mid \alpha_b(r)$. Ce cas ne peut se produire.
- Si $r = 0$, alors on a $m = qk$, d'où $A_b(m) = A_b^q(k)$. On a donc :

$$\begin{aligned} A_b(m) &= A_b^q(k) \\ &= (\alpha_b(k)\Xi_b - \alpha_b(k-1)I)^q \\ &= \sum_{i=0}^{q} (-1)^{q-i} C_q^i \alpha_b^i(k) \alpha_b^{q-i}(k-1) \Xi_b^i \end{aligned}$$

(rappelons qu'ici, $C_q^i = \frac{q!}{i!(q-i)!}$).

Lorsque $i \geq 2$, $\alpha_b^i(k)$ est divisible par $\alpha_b^2(k)$. Cela revient à $\alpha_b^i(k) \equiv 0 \pmod{\alpha_b^2(k)}$.

On a donc

$A_b(m)$

$$\begin{aligned} &\equiv (-1)^q C_q^0 \alpha_b^0(k) \alpha_b^q(k-1) \Xi_b^0 + (-1)^{q-1} C_q^1 \alpha_b^1(k) \alpha_b^{q-1}(k-1) \Xi_b^1 \pmod{\alpha_b^2(k)} \\ &\equiv (-1)^q \alpha_b^q(k-1) I + (-1)^{q-1} q \alpha_b(k) \alpha_b^{q-1}(k-1) \Xi_b \pmod{\alpha_b^2(k)} \\ &\equiv \begin{pmatrix} (-1)^q \alpha_b^q(k-1) & 0 \\ 0 & (-1)^q \alpha_b^q(k-1) \end{pmatrix} \\ &\quad + \begin{pmatrix} (-1)^{q-1} q \alpha_b(k) \alpha_b^{q-1}(k-1) b & -(-1)^{q-1} q \alpha_b(k) \alpha_b^{q-1}(k-1) \\ (-1)^{q-1} q \alpha_b(k) \alpha_b^{q-1}(k-1) & 0 \end{pmatrix} \pmod{\alpha_b^2(k)} \\ &\equiv \begin{pmatrix} (-1)^q \alpha_b^q(k-1) + (-1)^{q-1} q \alpha_b(k) \alpha_b^{q-1}(k-1) b & (-1)^q q \alpha_b(k) \alpha_b^{q-1}(k-1) \\ (-1)^{q-1} q \alpha_b(k) \alpha_b^{q-1}(k-1) & (-1)^q \alpha_b^q(k-1) \end{pmatrix} \\ &\qquad \pmod{\alpha_b^2(k)}. \end{aligned}$$

Puisque

$$A_b(m) = \begin{pmatrix} \alpha_b(m+1) & -\alpha_b(m) \\ \alpha_b(m) & \alpha_b(m-1) \end{pmatrix},$$

on a

$$\alpha_b(m) \equiv (-1)^{q-1} q \alpha_b(k) \alpha_b^{q-1}(k-1) \pmod{\alpha_b^2(k)}.$$

Ceci revient à dire que

$$\alpha_b^2(k) \mid \alpha_b(m) - (-1)^{q-1} q \alpha_b(k) \alpha_b^{q-1}(k-1).$$

Or, $\alpha_b(m)$ est divisible par $\alpha_b^2(k)$, ce qui implique que $(-1)^{q-1}q\alpha_b(k)\alpha_b^{q-1}(k-1)$ est également divisible par $\alpha_b^2(k)$. Il existe donc un ℓ tel que

$$\ell\alpha_b^2(k) = (-1)^{q-1}q\alpha_b(k)\alpha_b^{q-1}(k-1).$$

d'où

$$\ell\alpha_b(k) = (-1)^{q-1}q\alpha_b^{q-1}(k-1).$$

Donc, $\alpha_b(k)$ divise $(-1)^{q-1}q\alpha_b^{q-1}(k-1)$. Si $k=1$, alors $\alpha_b(k)=1$ divise évidemment m qui est alors égal à q. Si $k>1$, alors d'après le lemme 2.2.2, $\alpha_b(k) > \alpha_b(1)=1$. Or, d'après le lemme 2.2.5, $\alpha_b(k)$ et $\alpha_b(k-1)$ sont premiers entre eux ce qui implique que $\alpha_b(k)$ divise q et donc m.

□

2.4 Comportements de α_b

Matiiassevitch (1995) présente succinctement les phénomènes qui permettent d'établir dans la section suivante un système de contraintes diophantiennes qui définit exactement la fonction α_b. On se propose ici de d'éclaircir son propos en détaillant plus longuement les mécanismes mis en jeu. Les raisonnements qui suivent sont informels et leur lecture n'est pas indispensable. Néanmoins, ils permettent de comprendre l'origine des contraintes utilisées dans la section suivante.

Nous voulons prouver que l'ensemble des triplets (a,b,c) tels que $b \geq 4$ et $a = \alpha_b(c)$ est diophantien. Nous pouvons considérer que cet ensemble est l'ensemble contenant tous les éléments de la suite

$$(\alpha_b(0),b,0),\ldots,(\alpha_b(n),b,n),\ldots$$

avec $b \geq 4$. Observons tout d'abord que la suite

$$\alpha_b(0),\ldots,\alpha_b(n),\ldots$$

est périodique modulo v pour tout entier $v>0$. En effet, si $n \geq 2$, chaque terme $\alpha_b(n)$ est entièrement défini par $\alpha_b(n-1)$ et $\alpha_b(n-2)$. De plus tous les termes de la suite

modulo v sont inférieurs à v. Donc, il ne peut exister qu'une quantité finie de couples de termes consécutifs distincts. Ainsi, si la suite

$$\alpha_b(0), \ldots, \alpha_b(k)$$

contient tous les couples de termes consécutifs distincts, il existe forcément k' tel que $0 \leq k' < k$ et

$$(\alpha_b(k), \alpha_b(k+1)) \equiv (\alpha_b(k'), \alpha_b(k'+1)) \pmod{v}.$$

Soit ℓ tel que $k = k' + \ell$. Puisque $\alpha_b(n) = b\alpha_b(n-1) - \alpha_b(n-2)$, nous avons alors :

$$\alpha_b(k) \equiv \alpha_b(k'+\ell) \pmod{v},\ \alpha_b(k+1) \equiv \alpha_b(k'+1+\ell) \pmod{v},\ \ldots$$

Donc, il existe dans cette suite une série de nombres périodique de période ℓ. De plus, il est certain qu'il s'agit bel et bien d'une période de la suite α_b et non pas d'une pseudo-période qui ne se répéterait qu'à partir d'une position autre que le premier terme de la suite. En effet, deux termes consécutifs $f \equiv by - x \pmod{v}$ et $g \equiv bf - y \pmod{v}$ de la suite α_b modulo v ne peuvent être engendrés qu'à partir d'un couple unique de valeurs x et y. Si f et g étaient engendrés également à partie des entiers x' et y', alors on aurait

$$\begin{aligned} g &\equiv bf - y' \pmod{v} \\ bf - y &\equiv bf - y' \pmod{v} \\ y &\equiv y' \pmod{v}, \end{aligned}$$

d'où

$$\begin{aligned} f &\equiv by' - x' \pmod{v} \\ by - x &\equiv by' - x' \pmod{v} \\ by - x &\equiv by - x' \pmod{v} \\ x &\equiv x' \pmod{v}. \end{aligned}$$

Par conséquent, tout couple de nombres consécutifs dans la suite α_b ne peut être engendré qu'à partir d'un couple unique de nombres le précédant. Ainsi, pour toute séquence

$$\ldots, y_1, y_2, x_1, x_2 \ldots, x_\ell, x_1, x_2, \ldots$$

rencontrée dans la suite, on sait que $y_1 \equiv x_{\ell-1} \pmod{v}$ et $y_2 \equiv x_\ell \pmod{v}$. De proche en proche, on peut ainsi remonter jusqu'à l'origine de la suite et en conclure que la période commence à cette position, donc que la suite est périodique.

Nous allons maintenant choisir la valeur de v de manière à obtenir certaines formes de la suite $\alpha_b(0), \ldots, \alpha_b(n), \ldots$ modulo v.

Soit $m > 0$. Soit $v = \alpha_b(m+1) - \alpha_b(m-1)$ pour un m fixé. Nous avons

$$\alpha_b(m+1) \equiv \alpha_b(m-1) \pmod{v}.$$

Supposons que la relation

$$\alpha_b(m+k) \equiv \alpha_b(m-k) \pmod{v}$$

reste vraie pour k entre 1 et $n < m$. Alors elle reste vraie pour $k = n+1 \leq m$.
En effet,

$$\begin{aligned}
\alpha_b(m+n) &\equiv \alpha_b(m-n) \pmod{v} \\
b\alpha_b(m+n) &\equiv b\alpha_b(m-n) \pmod{v} \\
b\alpha_b(m+n) - \alpha_b(m+n-1) &\equiv b\alpha_b(m-n) - \alpha_b(m-n+1) \pmod{v} \\
\alpha_b(m+n+1) &\equiv b\alpha_b(m-n) - \alpha_b(m-n+1) \pmod{v} \\
\alpha_b(m+n+1) &\equiv b\alpha_b(m-n) - b\alpha_b(m-n) + \alpha_b(m-n-1) \pmod{v} \\
\alpha_b(m+n+1) &\equiv \alpha_b(m-n-1) \pmod{v}.
\end{aligned}$$

On a alors

$$\alpha_b(2m) \equiv \alpha_b(0) \equiv 0 \equiv -\alpha_b(0) \pmod{v}$$

et

$$\begin{aligned}
\alpha_b(2m) &\equiv \alpha_b(0) \pmod{v} \\
b\alpha_b(2m) - \alpha_b(2m-1) &\equiv b\alpha_b(0) - \alpha_b(1) \pmod{v} \\
\alpha_b(2m+1) &\equiv -\alpha_b(1) \pmod{v}.
\end{aligned}$$

Supposons que la relation

$$\alpha_b(2m+k) \equiv -\alpha_b(k) \pmod{v}$$

reste vraie pour tout $k \in \{1,\ldots,n\}$. Montrons qu'elle reste vraie pour $k = n+1$.
On a

$$\begin{aligned}\alpha_b(2m+n+1) &= b\alpha_b(2m+n) - \alpha_b(2m+n-1)\\ &\equiv -b\alpha_b(n) + \alpha_b(n-1) \pmod{v}\\ &\equiv -\alpha_b(n+1) \pmod{v}.\end{aligned}$$

Donc, la relation est vraie pour tout $k \geq 0$. De cette relation, on peut déduire très facilement le fait que la suite

$$\alpha_b(0),\ldots,\alpha_b(n),\ldots$$

modulo v possède une période de longueur $4m$. En effet, on a, pour $k \geq 0$:

$$\alpha_b(k+4m) = \alpha_b(2m+2m+k) \equiv -\alpha_b(2m+k) \equiv \alpha_b(k) \pmod{v}.$$

Nous allons maintenant considérer le comportement de la suite

$$arem(\alpha_b(0),v),\ldots,arem(\alpha_b(n),v),\ldots$$

De ce qui précède, on peut constater que pour tout $n \geq 0$ tel que $n = 2m\ell \pm k$ avec $0 \leq k < m$ on a, si ℓ est impair,

$$\alpha_b(n) \equiv -\alpha_b(k) \pmod{v},$$

et si ℓ est pair,

$$\alpha_b(n) \equiv \alpha_b(k) \pmod{v}.$$

De plus, on a

$$\begin{aligned}v &= \alpha_b(m+1) - \alpha_b(m-1)\\ &= b\alpha_b(m) - \alpha_b(m-1) - \alpha_b(m-1)\\ &= b\alpha_b(m) - 2\alpha_b(m-1),\end{aligned}$$

et d'après le lemme 2.2.2, puisque $b \geq 4$,

$$2\alpha_b(m-1) < (b-1)\alpha_b(m-1) < \alpha_b(m).$$

On a donc

$$\begin{aligned} v &\geq b\alpha_b(m) - \alpha_b(m) \\ &\geq 2\alpha_b(m). \end{aligned}$$

Par définition de $arem$, on a donc, pour tout $n \geq 0$ tel que $n = 2m\ell \pm k$ avec $0 \leq k < m$

$$arem(\alpha_b(n), v) = \alpha_b(k) = arem(\alpha_b(k), v).$$

Il est alors facile de constater que pour tout $n = 2m\ell \pm k$ avec $0 \leq k < m$ on a

$$\alpha_b(2m+n) \equiv -\alpha_b(n) \equiv \pm\alpha_b(k) \pmod{v}.$$

Donc,

$$arem(\alpha_b(2m+n), v) = \alpha_b(k) = arem(\alpha_b(n), v).$$

Ceci signifie que la suite

$$arem(\alpha_b(0), v), \ldots, arem(\alpha_b(n), v), \ldots$$

possède une période de longueur $2m$ de la forme

$$\alpha_b(0), \alpha_b(1), \ldots, \alpha_b(m-1), \alpha_b(m), \alpha_b(m-1), \ldots, \alpha_b(1).$$

Observons maintenant le comportement de la suite α_2. Comme on l'a vu précédemment, pour tout $n \geq 0$, on a $\alpha_2(n) = n$. Si on considère la suite

$$\alpha_2(0), \ldots, \alpha_2(n), \ldots$$

modulo u, avec $u > 0$, on constate sans surprise qu'elle possède une période de longueur u de forme

$$0, 1, \ldots, u-1.$$

Si on s'intéresse maintenant à la suite

$$arem(\alpha_2(0), u), \ldots, arem(\alpha_2(n), u), \ldots$$

on constate qu'elle possède une période de longueur u de forme

$$0, 1, \ldots, \frac{u}{2} - 1, \frac{u}{2}, \frac{u}{2} - 1, \ldots, 1$$

si u est pair ou de forme

$$0, 1, \ldots, \frac{u-1}{2} - 1, \frac{u-1}{2}, \frac{u-1}{2}, \frac{u-1}{2} - 1, \ldots, 1$$

si u est impair.

Il est aisé de constater l'existence d'une période de longueur u. En effet, pour tout $n \geq 0$, si $arem(n, u) = x$, alors on a $2x \leq u$ et $n \equiv \pm x \pmod{u}$. Ceci implique que

$$n + u \equiv \pm x + u \equiv \pm x \pmod{u}.$$

Par conséquent, on a $arem(n + u, u) = arem(n, u) = x$. Il est à peine plus ardu de caractériser la forme de cette période. Pour tout $n \geq 0$, il existe x et q tels que $0 \leq x < u$ et $n = uq + x$. On a alors $n \equiv x \pmod{u}$.

- Si $x \leq \frac{u}{2}$, alors on a $2x \leq u$ et $n \equiv x \pmod{u}$, d'où, par définition de $arem$, $arem(n, u) = x$.
- Si $x > \frac{u}{2}$, alors on a $2x > u$, d'où

$$\begin{aligned} 2x &> u \\ 2x - 2u &> u - 2u \\ 2(x - u) &> -u \\ 2(u - x) &< u. \end{aligned}$$

  Or,

$$x \equiv x - u \equiv -(u - x) \pmod{u},$$

  et donc, $arem(n, u) = (u - x)$.

Nous pouvons maintenant constater un fait intéressant au sujet des périodes des suites

$$arem(\alpha_b(0), v), \ldots, arem(\alpha_b(n), v), \ldots$$

et

$$arem(\alpha_2(0), u), \ldots, arem(\alpha_2(n), u), \ldots$$

En effet, elles sont respectivement de la forme

$$\alpha_b(0), \alpha_b(1), \ldots, \alpha_b(m-1), \alpha_b(m), \alpha_b(m-1), \ldots, 1$$

et

$$0, 1, \ldots, \frac{u}{2} - 1, \frac{u}{2}, \frac{u}{2} - 1 \ldots, 1$$

si u est pair ou

$$0, 1, \ldots, \frac{u-1}{2} - 1, \frac{u-1}{2}, \frac{u-1}{2}, \frac{u-1}{2} - 1 \ldots, 1$$

si u est impair. Un certain segment initial de ces périodes juxtapose 0 et $\alpha_b(0)$, 1 et $\alpha_b(1)$, etc.

Si k est la longueur de ce segment initial, on est alors en mesure de former $k+1$ triplets

$$(\alpha_b(0), b, 0), (\alpha_b(1), b, 1), \ldots, (\alpha_b(k), b, k)$$

appartenant à l'ensemble des triplets (a, b, c) satisfaisant $a = \alpha_b(c)$. En utilisant la fonction $arem$, on obtient alors la suite

$$(arem(\alpha_b(0), v), b, arem(\alpha_2(0), u)), (arem(\alpha_b(1), v), b, arem(\alpha_2(1), u)),$$

$$\ldots, (arem(\alpha_b(k), v), b, arem(\alpha_2(k), u)).$$

Le problème est de déterminer la longueur ℓ au-delà de laquelle cette correspondance ne tient plus. Il faut de plus des contraintes suffisamment souples pour pouvoir donner à ℓ une valeur quelconque dans la mesure où nous désirons pouvoir obtenir n'importe quel triplet (a, b, c) satisfaisant $a = \alpha_b(c)$. Nous ne pouvons donc pas borner les valeurs prises par u et m par une valeur constante, ce qui ne nous empêche pas de borner u par rapport à m.

Supposons donc que m soit supérieur à u. Nous sommes alors assurés que les segments initiaux de longueur ℓ inférieure ou égale à $\lfloor \frac{u}{2} \rfloor$ des suites

$$\alpha_b(0), \alpha_b(1), \alpha_b(2), \ldots$$

$$\alpha_2(0), \alpha_2(1), \alpha_2(2), \ldots$$

vont respectivement être de la forme

$$\alpha_b(0), \alpha_b(1), \ldots, \alpha_b(\ell)$$

$$0, 1, \ldots, \ell.$$

Tous les triplets $(arem(\alpha_b(k), v), b, arem(\alpha_2(k), u))$ avec $k \leq \lfloor \frac{u}{2} \rfloor$ appartiennent donc à l'ensemble des triplets (a, b, c) tels que $a = \alpha_b(c)$. En choisissant un u et un m suffisamment grands, il serait donc possible de trouver dans le segment initial de longueur ℓ inférieure ou égale à $\lfloor \frac{u}{2} \rfloor$ de la suite

$$(arem(\alpha_b(0), v), b, arem(\alpha_2(0), u)), \ldots, (arem(\alpha_b(n), v), b, arem(\alpha_2(n), u)), \ldots$$

n'importe quel triplet (a, b, c) tel que $a = \alpha_b(c)$. Il semble que nous ayons maintenant suffisamment d'informations pour décrire un système de contraintes identifiant un triplet (a, b, c) tel que $a = \alpha_b(c)$. Toutefois, nous cherchons à obtenir un système de contraintes diophantiennes. Ceci va susciter quelques difficultés supplémentaires. En effet, nous cherchons à prouver que α_b est diophantienne. Ceci exclut donc que α_b apparaisse dans le système de contraintes. Nous pouvons contourner ce problème en employant le lemme 2.2.4, qui nous permet d'affirmer que s'il existe x et y tels que

$$x^2 - bxy + y^2 = 1,$$

alors il existe $z \geq 0$ tel que $x = \alpha_b(z)$. L'inconvénient est qu'on ne connaît alors pas la valeur de z et qu'on ne possède pas de moyens de l'extraire à partir de x. Toutefois, nous venons de montrer que le segment initial de longueur $\lfloor \frac{u}{2} \rfloor$ de la suite

$$(arem(\alpha_b(0), v), b, arem(\alpha_2(0), u)), \ldots, (arem(\alpha_b(n), v), b, arem(\alpha_2(n), u)), \ldots$$

ne contient que des triplets (a, b, c) tels que $a = \alpha_b(c)$.

Si nous pouvions nous ramener à ce segment initial, connaître la valeur précise de z ne serait plus nécessaire. Il faut donc faire en sorte que la suite

$$(arem(\alpha_b(0), v), b, arem(\alpha_2(0), u)), \ldots, (arem(\alpha_b(n), v), b, arem(\alpha_2(n), u)), \ldots$$

soit périodique. Nous savons déjà que les suites

$$(arem(\alpha_b(0), v), \ldots, (arem(\alpha_b(n), v)), \ldots$$

$$(arem(\alpha_2(0), u), \ldots, (arem(\alpha_2(n), u)), \ldots$$

possèdent toutes deux des périodes, respectivement de longueur $2m$ et u. Si on impose que u divise m, alors il existe q tel que $m = qu$ et la suite

$$(arem(\alpha_b(0), v)), \ldots, (arem(\alpha_b(n), v)), \ldots$$

possède une période de longueur $2m = 2qu$. Donc, si u divise m, la suite

$$(arem(\alpha_b(0), v), b, arem(\alpha_2(0), u)), \ldots, (arem(\alpha_b(n), v), b, arem(\alpha_2(n), u)), \ldots$$

est périodique de longueur $2m$. Pour s'assurer que le triplet $(arem(\alpha_b(i), v), b, arem(\alpha_2(i), u))$ que l'on observe est bien tel que

$$arem(\alpha_b(i), v) = \alpha_b(k),$$

$$arem(\alpha_2(j), u) = k' \text{ et}$$

$$k = k',$$

il suffit donc de considérer sa position au sein d'une période.

On commence par ramener i dans l'intervalle de valeurs $[0..(2m-1)]$ en le divisant par $2m$. Soit $i = 2mq + r$, avec $r < 2m$ et $q \geq 0$. Puisque la suite possède une période de longueur $2m$, alors on a

$$arem(\alpha_b(i), v) = arem(\alpha_b(r), v) \text{ et}$$

$$arem(\alpha_2(i), u) = arem(\alpha_2(r), u).$$

On sait que si $2r \leq u$, alors $arem(\alpha_2(r), u) = r$. De plus, on sait que si $\alpha_b(r) \leq m$, alors $arem(\alpha_b(r), v) = \alpha_b(r)$. Puisque $u \mid m$, on a $u \leq m$. De plus, $r \leq \alpha_b(r)$, alors si $2\alpha_b(r) \leq u$ on a

$$(arem(\alpha_b(c), v), b, arem(\alpha_2(c), u)) = (\alpha_b(c), b, c).$$

Nous avons donc une condition suffisante pour nous assurer que le triplet considéré satisfait $a = \alpha_b(c)$.

Il reste un dernier obstacle à écarter. Nous avons remarqué précédemment que si le terme $a = \alpha_b(c)$ est choisi en utilisant l'équation $a^2 - baa' + a'^2 = 0$, on ne sait pas extraire c à partir de a. Cependant, il est nécessaire d'exprimer le terme $\alpha_2(c)$. Pour contourner cette difficulté, on utilisera la propriété de la suite α évoquée dans le lemme 2.2.6. Soit w tel que $w \equiv b \pmod{v}$ et $w \equiv 2 \pmod{u}$. Alors, d'après le lemme 2.2.6,

$$\alpha_w(n) \equiv \alpha_b(n) \pmod{v}$$

$$\alpha_w(n) \equiv \alpha_2(n) \pmod{u}.$$

On a alors, d'après le lemme 2.1.2

$$arem(\alpha_b(n), v) = arem(\alpha_w(n), v)$$

$$arem(\alpha_2(n), u) = arem(\alpha_w(n), u).$$

Par conséquent, on a

$$(arem(\alpha_w(n), v), b, arem(\alpha_w(n), u)) = (arem(\alpha_b(n), v), b, arem(\alpha_2(n), u))$$

Toutes les observations effectuées dans cette section vont permettre de formaliser le système de contraintes diophantiennes définissant l'ensemble des triplets (a, b, c) tels que $a = \alpha_b(c)$.

2.5 La fonction α_b est diophantienne

Nous allons montrer dans cette section que l'ensemble

$$\{(a, b, c) \mid b \geq 4 \wedge a = \alpha_b(c)\}$$

est diophantien. Pour ce faire, nous exhiberons un système de contraintes diophantiennes qui sera satisfait si et seulement si $a = \alpha_b(c)$. Ces contraintes sont les suivantes :

1. $b \geq 4$

2. $u^2 - but + t^2 = 1$
3. $s^2 - brs + r^2 = 1$
4. $r < s$
5. $u^2 \mid s$
6. $v = bs - 2r$
7. $v \mid w - b$
8. $u \mid w - 2$
9. $w > 2$
10. $x^2 - wxy + y^2 = 1$
11. $2a < u$
12. $a = arem(x, v)$
13. $c = arem(x, u)$

On peut, à la lumière des observations de la section précédente, comprendre le rôle de ces contraintes. La contrainte (3) alliée à la contrainte (4) permet d'obtenir deux termes r et s tels que $r = \alpha_b(m)$ et $s = \alpha_b(m+1)$. En utilisant ces valeurs dans la contrainte (6), on produit une valeur v capable de créer une période particulière telle que décrite dans la section précédente. La contrainte (2) permet d'obtenir un terme $u = \alpha_b(k)$. Le lemme 2.3.4 et la contrainte (5) permettent d'obtenir l'assurance que u divise m. La contrainte (10) permet d'obtenir un terme $x = \alpha_w(n)$. Les contraintes (7) et (8) permettront d'utiliser le lemme 2.2.6 pour prouver que $arem(x, v) = arem(\alpha_b(n), v)$ et $arem(x, u) = arem(\alpha_2(n), u)$. Enfin, la contrainte (11) est celle évoquée à la fin de la section précédente et permet de s'assurer qu'il existe j tel que $arem(x, v) = \alpha_b(j)$ et $arem(x, u) = j$.

Lemme 2.5.1 (α_b est diophantienne). *Soient a, b et c des entiers naturels. Il existe des entiers naturels u, t, r, s, v, w, x, y satisfaisant les contraintes 1 à 14 si et seulement si $a = b^c$.*

Démonstration. ($\Rightarrow$) D'après le lemme 2.2.4 et les conditions (1) et (2), il existe $k \geq 0$ tel que $u = \alpha_b(k)$. D'après le lemme 2.2.4 et les conditions (1) et (3) et (4), il existe $m \geq 0$ tel que $s = \alpha_b(m)$ et $r = \alpha_b(m-1)$. Nous pouvons donc récrire la condition (5) : $\alpha_b^2(k) \mid \alpha_b(m)$. D'après le lemme 2.3.4 on a alors $\alpha_b(k) \mid m$ et puisque $\alpha_b(k) = u$, on a $u \mid m$. On récrit maintenant la condition (6) en remplaçant s et r par $\alpha_b(m)$ et $\alpha_b(m-1)$:

$$\begin{aligned} v &= b\alpha_b(m) - 2\alpha_b(m-1) \\ &= b\alpha_b(m) - \alpha_b(m-1) - \alpha_b(m-1) \\ &= \alpha_b(m+1) - \alpha_b(m-1). \end{aligned}$$

D'après le lemme 2.2.4 et les conditions (9) et (10), il existe n tel que $x = \alpha_w(n)$.
D'après la condition (8), $u \mid w-2$, d'où $w \equiv 2 \pmod{u}$. En utilisant le lemme 2.2.6, on obtient alors $\alpha_w(n) = x \equiv \alpha_2(n) \pmod{u}$. Or, d'après le lemme 2.2.1, on a $\alpha_2(n) = n$, d'où $x \equiv n \pmod{u}$.
D'après la condition (7), $v \mid w-b$, d'où $w \equiv b \pmod{v}$. En utilisant le lemme 2.2.6, on obtient alors $\alpha_w(n) = x \equiv \alpha_b(n) \pmod{v}$.

Soient $\ell \geq 0$ et $0 \leq j \leq m$ tels que $n = 2m\ell \pm j$. On a

$$\begin{aligned} A_b(n) &= \Xi_b^n \\ &= \Xi_b^{2m\ell \pm j} \\ &= ((\Xi_b^m)^2)^\ell \Xi_b^{\pm j} \\ &= ((\Xi_b^m)^2)^\ell (\Xi_b^j)^{\pm 1} \\ &= (A_b^2(m))^\ell (A_b(j))^{\pm 1}. \end{aligned}$$

Puisque $v = \alpha_b(m+1) - \alpha_b(m-1)$, on a $\alpha_b(m+1) \equiv \alpha_b(m-1) \pmod{v}$, d'où

$$\begin{aligned} A_b(m) &= \begin{pmatrix} \alpha_b(m+1) & -\alpha_b(m) \\ \alpha_b(m) & -\alpha_b(m-1) \end{pmatrix} \\ &\equiv \begin{pmatrix} \alpha_b(m-1) & -\alpha_b(m) \\ \alpha_b(m) & -\alpha_b(m+1) \end{pmatrix} \pmod{v} \\ &\equiv -\begin{pmatrix} -\alpha_b(m-1) & \alpha_b(m) \\ -\alpha_b(m) & \alpha_b(m+1) \end{pmatrix} \pmod{v} \\ &\equiv -A_b^{-1}(m) \pmod{v}. \end{aligned}$$

On a donc $A_b(m) \equiv -A_b^{-1}(m) \pmod{v}$, d'où

$$A_b^2(m) \equiv -A_b^{-1}(m)A_b(m) \equiv -I \pmod{v}.$$

On a alors

$$\begin{aligned} A_b(n) &= (A_b^2(m))^\ell A_b^{\pm 1}(j) \\ &\equiv (-I)^\ell A_b(j)^{\pm 1} \pmod{v} \\ &\equiv \pm A_b^{\pm 1}(j) \pmod{v}. \end{aligned}$$

On a vu précédemment que

$$x = \alpha_w(n) \equiv \alpha_b(n) \pmod{v}.$$

D'après l'équation précédente, on a

$$\alpha_b(n) \equiv \pm\alpha_b(j) \pmod{v},$$

et donc

$$x \equiv \pm\alpha_b(j) \pmod{v}.$$

Nous avons alors les trois inégalités suivantes :

$$\begin{aligned} 2\alpha_b(j) &\leq 2\alpha_b(m), \\ 2\alpha_b(m) &\leq (b-2)\alpha_b(m) \text{ et} \\ (b-2)\alpha_b(m) &< b\alpha_b(m) - 2\alpha_b(m-1). \end{aligned}$$

La première est justifiée par le choix de j, par hypothèse, $j \leq m$. La seconde est justifiée par le fait que $b \geq 4$. Enfin, on a, d'après le lemme 2.2.2, $\alpha_b(m-1) < \alpha_b(m)$, d'où

$$(b-2)\alpha_b(m) = b\alpha_b(m) - 2\alpha_b(m) < b\alpha_b(m) - 2\alpha_b(m-1).$$

Puisque $b\alpha_b(m) - 2\alpha_b(m-1) = v$, nous pouvons déduire du système d'inégalités précédent que $2\alpha_b(j) < v$. Nous avons alors $2\alpha_b(j) < v$ et $x \equiv \pm\alpha_b(j) \pmod{v}$. Donc par définition, on a $\alpha_b(j) = arem(x, v)$. D'après la condition (12), on a $a = arem(x, v)$. Or, on sait que $\alpha_b(j) = arem(x, v)$. Par transitivité, on a donc $a = \alpha_b(j)$. D'après la condition (11), on a $2a < u$. Or, on sait que $\alpha_b(j) = a$. Donc, $2\alpha_b(j) < u$. D'après le lemme 2.2.3, $\alpha_b(j) \geq j$, d'où $2j \leq 2\alpha_b(j) < u$. On sait que $u \mid m$ et $n = 2\ell m \pm j$, donc $u \mid n \pm j$ et $j \equiv \pm n \pmod{u}$. On sait de plus que $2j < u$. Donc, par définition, on a $j = arem(n, u)$. D'après la condition (13), $c = arem(x, u)$, donc $c \equiv \pm x \pmod{u}$ et $2c \leq u$. Or, on sait que $x \equiv n \pmod{u}$, donc $c \equiv \pm x \equiv \pm n \pmod{u}$. Par définition, on a donc $c = arem(n, u)$. Donc, on a $j = arem(n, u) = arem(x, u) = c$. Finalement, on a $c = j$ et $a = \alpha_b(j)$, d'où $a = \alpha_b(c)$.

($\Leftarrow$) Nous allons maintenant montrer la proposition réciproque, à savoir que si $b \geq 4$ et $a = \alpha_b(c)$, alors il existe u, t, r, s, v, w, x, y tels que le système de contraintes est satisfait. Par hypothèse, la condition (1), $b \geq 4$ est satisfaite. Soient u et k tels que $u > 2a$, $u > 2$, u est impair et $u = \alpha_b(k)$ pour $k > 0$. Il existe k tel que u satisfasse ces conditions puisque la suite des $\alpha_b(i)$ est croissante (lemme 2.2.2) et que pour tout $i \geq 0$, $\alpha_b(i)$ et $\alpha_b(i+1)$ sont premiers entre eux (lemme 2.2.5), ce qui implique qu'une de ces deux valeurs est impaire. La condition (11) est satisfaite. Soit $t = \alpha_b(k+1)$. D'après le lemme 2.2.4, on a $u^2 - but + t^2 = 1$. La condition (2) est satisfaite. Soient s, r et m tels que $m > 0$, $m = uk$, $s = \alpha_b(m)$ et $r = \alpha_b(m-1)$. D'après le lemme 2.2.2, on a $r < s$. D'après le lemme 2.2.4, on a $s^2 - bsr + r^2 = 1$. Les conditions (3) et (4) sont satisfaites.

On a vu précédemment, dans la démonstration du lemme 2.3.4, que si $m = qk$, on a

$$\alpha_b(m) \equiv (-1)^{q-1} q\alpha_b(k)\alpha_b^{q-1}(k-1) \pmod{\alpha_b^2(k)}.$$

Puisqu'ici, $m = uk$ et $u = \alpha_b(k)$, on a

$$s = \alpha_b(uk) \equiv (-1)^{u-1} u \alpha_b(k) \alpha_b^{u-1}(k-1) \pmod{u^2}.$$

Donc, on a

$$u^2 \mid s - (-1)^{u-1} u^2 \alpha_b^{u-1}(k-1).$$

Puisque u^2 divise $(-1)^{u-1} u^2 \alpha_b^{u-1}(k-1)$, u^2 divise aussi s. La condition (5) est satisfaite. Soit $v = bs - 2r$. La condition (6) est satisfaite. Remarquons que v possède les propriétés suivantes : $v = \alpha_b(m+1) - \alpha_b(m-1)$ et $v > 2\alpha_b(m)$. En effet, on a

$$\begin{aligned} v &= bs - 2r \\ &= b\alpha_b(m) - 2\alpha_b(m-1) \\ &= b\alpha_b(m) - \alpha_b(m-1) - \alpha_b(m-1) \\ &= \alpha_b(m+1) - \alpha_b(m-1), \end{aligned}$$

et d'après le lemme 2.2.2,

$$bs - 2r \geq 4\alpha_b(m) - 2\alpha_b(m-1) > 4\alpha_b(m) - 2\alpha_b(m) = 2\alpha_b(m).$$

Nous voulons maintenant utiliser le théorème des restes chinois pour prouver qu'il existe w tel que $v \mid w - b$ et $u \mid w - 2$. Pour utiliser ce théorème, il faut prouver que u et v sont premiers entre eux et que $u > 2$ et $v > b$. Soit $d \geq 1$ tel que $d \mid u$ et $d \mid v$. Ici, d est forcément impair puisque u est impair. Puisque $u^2 \mid s$, alors $d \mid s$. De plus, puisque $v = bs - 2r$ et que $d \mid bs$, $d \mid v$ et $d \neq 2$, on a forcément $d \mid r$. Or, on a $r^2 - bsr + s^2 = 1$ et puisque d divise r^2, bsr et s^2, alors d divise 1. Donc, $d = 1$ et u et v sont premiers entre eux. Puisque $u > 2$ et $u = \alpha_b(k)$, alors $k \geq 1$. Donc, $m = uk \geq 1$. De plus, $b \geq 4$. En appliquant le lemme 2.2.8, on obtient $\alpha_b(m+1) - \alpha_b(m-1) > b$. Or, $\alpha_b(m+1) - \alpha_b(m-1) = v$. On a donc $v > b$. De plus, par hypothèse, $u > 2$. On peut maintenant appliquer le théorème des restes chinois pour affirmer qu'il existe w tel que $b = rem(w, v)$ et $2 = rem(w, u)$, avec $0 < w < uv$. Ceci implique que $v \mid w - b$ et $u \mid w - 2$. Les conditions (7) et (8) sont satisfaites. De plus, puisque $b = rem(w, v)$ et que $b \geq 4$, alors $w > 4 > 2$. La condition (9) est satisfaite. Soient $x = \alpha_w(c)$ et

$y = \alpha_w(c+1)$. D'après le lemme 2.2.4, on a $x^2 - wxy + y^2 = 1$. La condition (10) est satisfaite. La condition (7) étant satisfaite, on a $w \equiv b \pmod{v}$. D'après le lemme 2.2.6, on a alors $\alpha_w(c) \equiv \alpha_b(c) \pmod{v}$. Donc,

$$x = \alpha_w(c) \equiv \alpha_b(c) \equiv a \pmod{v}.$$

Puisque $m = uk$ et que $u > 2$ et $k \geq 1$, alors on a $m > k$. De plus, la condition (11) est satisfaite, donc $u = \alpha_b(k) > 2\alpha_b(c) = 2a$. D'après le lemme 2.2.2, ceci implique que $k > c$. Donc, $m > k > c$ et on a, toujours par le lemme 2.2.2, $\alpha_b(m) > \alpha_b(c)$. De plus, on a vu précédemment que $v > 2\alpha_b(m)$, donc $v > 2\alpha_b(c) = 2a$. On a $v > 2a$ et $x \equiv a \pmod{v}$, par définition on a donc $a = arem(x, v)$. La condition (12) est satisfaite. Par définition, on a $x = \alpha_w(c)$ et, d'après le lemme 2.2.1, $c = \alpha_2(c)$. De plus, on a $w \equiv 2 \pmod{(w-2)}$. D'après le lemme 2.2.6, on a donc $\alpha_w(c) \equiv \alpha_2(c) \pmod{(w-2)}$, donc $x \equiv c \pmod{(w-2)}$. Nous avons alors $w - 2 \mid x - c$. Or, puisque la condition (8) est satisfaite, on a $u \mid w - 2$ et donc $u \mid x - c$. Par conséquent, on a $x \equiv c \pmod{u}$. D'après le lemme 2.2.3, on a $c \leq \alpha_b(c)$, d'où $2c \leq 2\alpha_b(c) = 2a$. Puisque la condition (11) est satisfaite, on a $2c \leq 2a < u$. Puisque $2c < u$ et que $x \equiv c \pmod{u}$, alors par définition on a $c = arem(x, u)$. La condition (13) est satisfaite.

Nous venons d'exhiber u, v, s, t, r, w, x et y tels que le système de contraintes diophantiennes est satisfait. Donc, si on a $b \geq 4$ et $a = \alpha_b(c)$, il existe u, v, s, t, r, w, x et y tels que le système est satisfait.

Nous pouvons maintenant conclure que a, b et c sont tels que $b \geq 4$ et $a = \alpha_b(c)$ si et seulement s'il existe u, v, s, t, r, w, x et y satisfaisant le système de contraintes diophantiennes présenté. □

D'après ce lemme, l'ensemble $\{(a, b, c) \mid b \geq 4 \wedge a = \alpha_b(c)\}$ est diophantien puisqu'on peut le définir au moyen d'un ensemble de contraintes diophantiennes.

2.6 L'exponentiation est diophantienne

Le résultat de la section précédente est suffisant pour conclure que l'exponentiation est diophantienne (Robinson, 1952). Toutefois, Matiiassevitch (1995) fournit des indications sur la manière de faire cette preuve en s'appuyant spécifiquement sur la suite α_b. Pour les besoins de cette section, on posera $0^0 = 1$.

Dans un premier temps, établissons quelques inégalités utiles.

Lemme 2.6.1. *Pour tout entier naturel $b \geq 2$ et tout entier naturel $n \geq 0$, on a $(b-1)^n \leq \alpha_b(n+1)$.*

Démonstration (*). On procède par récurrence sur n.

- Pour $n = 0$, on a : $(b-1)^0 = 1 = \alpha_b(1)$. La propriété est vérifiée.
- Pour $n > 0$, on a :

$$(b-1)^n = (b-1)(b-1)^{n-1}.$$

Par hypothèse de récurrence, on a :

$$(b-1)^{n-1} \leq \alpha_b(n).$$

D'où

$$\begin{aligned} (b-1)^{n-1} &\leq \alpha_b(n) \\ (b-1)(b-1)^{n-1} &\leq (b-1)\alpha_b(n) \\ (b-1)^n &\leq b\alpha_b(n) - \alpha_b(n). \end{aligned}$$

D'après le lemme 2.2.2, on a $\alpha_b(n-1) < \alpha_b(n)$, et donc :

$$\begin{aligned} (b-1)^n &\leq b\alpha_b(n) - \alpha_b(n) \\ (b-1)^n &\leq b\alpha_b(n) - \alpha_b(n-1) \\ &= \alpha_b(n+1). \end{aligned}$$

□

Lemme 2.6.2. *Pour tous entiers naturels $b \geq 2$ et $n \geq 0$, on a $\alpha_b(n+1) \leq b^n$.*

Démonstration. On procède par récurrence sur n.

- Pour $n = 0$, on a : $\alpha_b(n+1) = 1 = b^0$. La propriété est vérifiée.
- Pour $n = 1$, on a : $\alpha_b(n+1) = b = b^1$. La propriété est vérifiée.
- Pour $n > 1$, on a : $\alpha_b(n+1) = b\alpha_b(n) - \alpha_b(n-1)$.
  Par hypothèse de récurrence, $b^{n-1} \geq \alpha_b(n)$, d'où :

$$\begin{aligned} b^{n-1} &\geq \alpha_b(n) \\ bb^{n-1} &\geq b\alpha_b(n) \\ b^n &\geq b\alpha_b(n) \\ b^n &\geq b\alpha_b(n) - \alpha_b(n-1) \\ b^n &\geq \alpha_b(n+1). \end{aligned}$$

□

Les deux propriétés suivantes, aisément démontrables, serviront dans des démonstrations ultérieures. Nous les exposons ici pour alléger ces démonstrations.

Lemme 2.6.3. *Pour tous entiers naturels $k > 0$, $a \geq 0$, $b \geq 0$ et $c \geq 0$, on a*

$$(ka+b)^c = \left(k\left(a + \frac{b}{k}\right)\right)^c = k^c\left(a + \frac{b}{k}\right)^c.$$

De ce lemme, on peut immédiatement déduire le corollaire suivant.

Corollaire 2.6.4. *Pour tous entiers naturels $k > 0$, $a \geq 0$, $b \geq 0$ et $c \geq 0$, on a*

$$(ka+b)^c \leq k^c(a+b)^c.$$

Lemme 2.6.5. *Pour tout nombre réel $a < 1$ et tout entier naturel $n \geq 0$, on a*

$$(1-a)^n \geq 1 - an.$$

Démonstration. On procède par récurrence sur n.

- Pour $n = 0$, on a : $(1-a)^0 = 1 = 1 - a \cdot 0$. La propriété est vérifiée.
- Pour $n > 0$, on a :

  $$(1-a)^n = (1-a)(1-a)^{n-1}.$$

  Par hypothèse de récurrence, on a $(1-a)^{n-1} \geq 1 - a(n-1)$ et puisque $1 - a > 0$, on a $(1-a)(1-a)^{n-1} \geq (1-a)(1-a(n-1))$. On a alors

$$\begin{aligned}
(1-a)^n &= (1-a)(1-a)^{n-1} \\
&\geq (1-a)(1-a(n-1)) \\
&= 1 - a(n-1) - a + a^2(n-1) \\
&= 1 - an + a - a + a^2 n - a^2 \\
&= 1 - an + a^2 n - a^2 \\
&= 1 - an + a^2(n-1) \\
&\geq 1 - an.
\end{aligned}$$

□

Lemme 2.6.6. *Pour tout entier naturel $c > 0$, $a > 0$ et $x > 2ac$, on a*

$$\frac{1}{1 - \frac{ac}{x}} < 1 + \frac{2ac}{x}.$$

Démonstration.

$$\begin{aligned} x &> 2ac \\ x - 2ac &> 0 \\ acx - 2a^2c^2 &> 0 \\ \frac{ac}{x} - \frac{2a^2c^2}{x^2} &> 0 \\ 1 - \frac{ac}{x} + \frac{2ac}{x} - \frac{2a^2c^2}{x^2} &> 1 \\ \left(1 - \frac{ac}{x}\right)\left(1 + \frac{2ac}{x}\right) &> 1. \end{aligned}$$

Puisque $x > 2ac$, alors $1 - \frac{ac}{x} > 1 - \frac{1}{2} > 0$. Par conséquent,

$$\left(1 - \frac{ac}{x}\right)\left(1 + \frac{2ac}{x}\right) > 1 \Rightarrow 1 + \frac{2ac}{x} > \frac{1}{1 - \frac{ac}{x}}.$$

□

La particularité de α_b est que pour des valeurs de x suffisamment grandes, on a

$$b^c = \lim_{x \to \infty} \frac{\alpha_{bx+4}(c+1)}{\alpha_x(c+1)}.$$

Les limites ne faisant pas partie du formalisme diophantien, on doit recourir à la division euclidienne pour parvenir à transcrire ce résultat au moyen d'expressions diophantiennes. Ici, on va montrer que pour un x assez grand,

$$b^c = \alpha_{bx+4}(c+1) \text{ div } \alpha_{bx}(c+1).$$

Lemme 2.6.7. *Pour tout entier naturel $b \geq 0$, $c \geq 0$ et $x > 16(c+1)(b+1)^c$, on a*

$$b^c = \alpha_{bx+4}(c+1) \text{ div } \alpha_x(c+1).$$

Démonstration. Montrons dans un premier temps que

$$\frac{\alpha_{bx+4}(c+1)}{\alpha_x(c+1)} \geq b^c.$$

D'après le lemme 2.6.1, $\alpha_{bx+4}(c+1) \geq (bx+3)^c$.

D'après le lemme 2.6.2, $\alpha_x(c+1) \leq x^c$.

Par conséquent, on a

$$\frac{\alpha_{bx+4}(c+1)}{\alpha_x(c+1)} \geq \frac{(bx+3)^c}{x^c} \geq b^c.$$

Montrons maintenant que

$$\frac{\alpha_{bx+4}(c+1)}{\alpha_x(c+1)} < b^c + 1.$$

Pour $b = 0$ et $c = 0$, on a

$$\frac{\alpha_4(1)}{\alpha_x(1)} = \frac{1}{1} = 1 = 0^0;$$

et la propriété est vérifiée.

Pour $b = 0$ et $c > 0$, on a, d'après les lemmes 2.6.2 et 2.6.1,

$$\alpha_4(c+1) \leq 4^c$$

et

$$\alpha_x(c+1) \geq (x-1)^c.$$

On a donc

$$\frac{\alpha_4(c+1)}{\alpha_x(c+1)} \leq \frac{4^c}{(x-1)^c}.$$

Puisque $x > 5$, on a

$$\frac{4^c}{(x-1)^c} < 1.$$

On peut alors conclure que

$$\frac{\alpha_4(c+1)}{\alpha_x(c+1)} < 1.$$

La propriété est vérifiée.

Si $b > 0$ et $c \geq 0$, on a, d'après les lemmes 2.6.2 et 2.6.1,

$$\alpha_{bx+4}(c+1) \leq (bx+4)^c$$

et

$$\alpha_x(c+1) \geq (x-1)^c.$$

On obtient alors

$$\frac{\alpha_{bx+4}(c+1)}{\alpha_x(c+1)} \leq \frac{(bx+4)^c}{(x-1)^c}.$$

D'après le lemme 2.6.3, on a

$$(bx+4)^c = x^c \left(b + \frac{4}{x}\right)^c$$

et

$$(x-1)^c = x^c \left(1 - \frac{1}{x}\right)^c.$$

Par conséquent,

$$\frac{(bx+4)^c}{(x-1)^c} = \frac{(b+\frac{4}{x})^c}{(1-\frac{1}{x})^c}.$$

D'après le corollaire 2.6.4, on a

$$\left(b + \frac{4}{x}\right)^c \leq b^c \left(1 + \frac{4}{x}\right)^c.$$

On obtient par conséquent l'inégalité

$$\frac{(bx+4)^c}{(x-1)^c} \leq \frac{b^c(1+\frac{4}{x})^c}{(1-\frac{1}{x})^c}.$$

On a

$$0 < 1 - \frac{16}{x^2} = \left(1 + \frac{4}{x}\right)\left(1 - \frac{4}{x}\right) \leq 1.$$

On obtient donc

$$\left(\left(1 + \frac{4}{x}\right)\left(1 - \frac{4}{x}\right)\right)^c = \left(1 + \frac{4}{x}\right)^c \left(1 - \frac{4}{x}\right)^c \leq \left(1 + \frac{4}{x}\right)\left(1 - \frac{4}{x}\right) \leq 1.$$

Par conséquent,

$$\frac{1}{\left(1+\frac{4}{x}\right)^c \left(1-\frac{4}{x}\right)^c} \geq 1$$

et

$$\frac{b^c \left(1+\frac{4}{x}\right)^c}{\left(1-\frac{1}{x}\right)^c} \leq \frac{b^c \left(1+\frac{4}{x}\right)^c}{\left(1-\frac{1}{x}\right)^c \left(1+\frac{4}{x}\right)^c \left(1-\frac{4}{x}\right)^c} = \frac{b^c}{\left(1-\frac{1}{x}\right)^c \left(1-\frac{4}{x}\right)^c}.$$

On a

$$1 - \frac{1}{x} > 1 - \frac{4}{x}.$$

De plus, puisque $x > 4$, on a

$$1 - \frac{1}{x} > 0$$

et

$$1 - \frac{4}{x} > 0.$$

On obtient donc

$$\frac{(1 - \frac{1}{x})}{(1 - \frac{4}{x})} > 1,$$

et par conséquent

$$\frac{(1 - \frac{1}{x})^c}{(1 - \frac{4}{x})^c} > 1.$$

On obtient

$$\frac{b^c}{(1 - \frac{1}{x})^c (1 - \frac{4}{x})^c} \leq \frac{b^c}{(1 - \frac{1}{x})^c (1 - \frac{4}{x})^c} \cdot \frac{(1 - \frac{1}{x})^c}{(1 - \frac{4}{x})^c} = \frac{b^c}{(1 - \frac{4}{x})^c (1 - \frac{4}{x})^c} = \frac{b^c}{(1 - \frac{4}{x})^{2c}}$$

Le lemme 2.6.5 nous permet d'écrire

$$\left(1 - \frac{4}{x}\right)^{2c} \geq 1 - \frac{8c}{x},$$

d'où

$$\frac{b^c}{(1 - \frac{4}{x})^{2c}} \leq \frac{b^c}{1 - \frac{8c}{x}}.$$

Finalement, d'après le lemme 2.6.6, si $x > 16c$, on a

$$\frac{b^c}{1 - \frac{8c}{x}} \leq b^c \left(1 + \frac{16c}{x}\right).$$

On a démontré, au terme de cette série d'inégalités, que l'inégalité

$$\frac{\alpha_{bx+4}(c+1)}{\alpha_x(c+1)} \leq b^c \left(1 + \frac{16c}{x}\right)$$

est vraie si $x > 16c$. De plus, si $x > 16cb^c$, on a

$$b^c \left(1 + \frac{16c}{x}\right) = b^c + \frac{16cb^c}{x} < b^c + 1.$$

Nous pouvons trouver une borne inférieure de x pour laquelle la propriété

$$b^c = \alpha_{bx+4}(c+1) \text{ div } \alpha_x(c+1)$$

est vraie quelles que soient les valeurs de b et c :

- lorsque $b = c = 0$, nous n'avons pas eu besoin d'émettre d'hypothèses sur la valeur de x ;
- lorsque $b = 0$ et $c > 0$, il faut au moins que $x > 5$ pour conclure que la propriété est vraie ;
- lorsque $b > 0$, il faut au moins que $x > 16cb^c$ et $x > 4$ pour conclure que la propriété est vraie.

Nous pouvons donc imposer la condition suivante sur x

$$x > 16(c+1)(b+1)^c.$$

pour nous assurer que la propriété est vraie quelles que soient les valeurs de b et c. □

Le lemme 2.6.7 fournit presque une définition diophantienne de l'exponentiation. En effet, la division euclidienne est diophantienne et pour $b \geq 4$, la fonction α_b est également diophantienne (voir section 2.5). En revanche, on ne sait pas si la contrainte

$$x > 16(c+1)(b+1)^c$$

est diophantienne dans la mesure où l'exponentiation n'est toujours pas connue pour être diophantienne à ce stade de la démonstration. Pour achever la démonstration du caractère diophantien de l'exponentiation, il suffit donc de remplacer cette contrainte par une contrainte diophantienne équivalente. D'après le lemme 2.6.2, on a $\alpha_b(c+1) \geq (b-1)^c$. Nous allons utiliser cette propriété pour définir une nouvelle borne inférieure pour x. Ceci permet d'obtenir une définition diophantienne de l'exponentiation.

Théorème 2.6.8 (Définition diophantienne de l'exponentiation). *Pour tous entiers naturels a, b et c, on a*

$$a = b^c \Leftrightarrow \exists x[x > 16(c+1)\alpha_{b+4}(c+1) \wedge a = \alpha_{bx+4}(c+1) \text{ div } \alpha_x(c+1)].$$

Démonstration $(*)$. $(\Rightarrow)$ Soit $a = b^c$, avec $b \geq 0$ et $c \geq 0$ quelconques. Soit $x > 16(b+1)\alpha_{b+4}(c+1)$. D'après le lemme 2.6.1, on a

$$\alpha_{b+4}(c+1) \geq (b+3)^c > (b+1)^c,$$

d'où

$$x > 16(c+1)\alpha_{b+4}(c+1) > 16(c+1)(b+1)^c.$$

D'après le lemme 2.6.7, on a donc

$$b^c = \alpha_{bx+4}(c+1) \text{ div } \alpha_x(c+1).$$

$(\Leftarrow)$ Soient x et a tels que $x > 16(c+1)\alpha_{b+4}(c+1)$ et $a = \alpha_{bx+4}(c+1)$ div $\alpha_x(c+1)$, avec b et c quelconques. D'après le lemme 2.6.1, on a

$$\alpha_{b+4}(c+1) \geq (b+3)^c > (b+1)^c,$$

d'où

$$x > 16(c+1)\alpha_{b+4}(c+1) > 16(c+1)(b+1)^c.$$

D'après le lemme 2.6.7, on a donc

$$a = \alpha_{bx+4}(c+1) \text{ div } \alpha_x(c+1) = b^c.$$

□

Puisque nous savons désormais que l'exponentiation est diophantienne, nous pouvons, pour la suite, manipuler des équations diophantiennes exponentielles, c'est-à-dire des équations construites non plus seulement à l'aide des opérateurs arithmétiques, mais également les équations construites en utilisant l'exponentiation. Ceci découle directement de l'observation faite dans le premier chapitre concernant le caractère diophantien de la composition de fonctions diophantiennes. Ce résultat permettrait immédiatement de conclure à l'indécidabilité du dixième problème de Hilbert (Davis, Putnam et Robinson, 1961). Toutefois, Matiiassevitch (1995) présente une démonstration qui n'est pas

celle que Davis, Putnam et Robinson ont adoptée en 1961. Matiiassevitch (1976) utilise le codage positionnel qui se base sur la représentation d'un nombre dans une certaine base pour conclure à l'indécidabilité. Le prochain chapitre expose la définition et les propriétés de ce codage.

CHAPITRE III

CODAGE

Nous allons maintenant exposer deux méthodes de codage qui nous permettront de réduire, dans les systèmes de contraintes diophantiennes ultérieurs, des uplets de longueurs variables à des uplets de longueurs fixes. Les méthodes que nous emploierons sont diophantiennes; nous pouvons donc nous en servir pour démontrer le caractère diophantien des ensembles décrits.

3.1 Numérotation de Cantor

Cette méthode classique a été utilisée par Cantor pour démontrer qu'une union dénombrable d'ensembles dénombrables est dénombrable. Il utilise pour cela le polynôme suivant, qui à un couple d'entiers (a, b) associe le nombre entier :

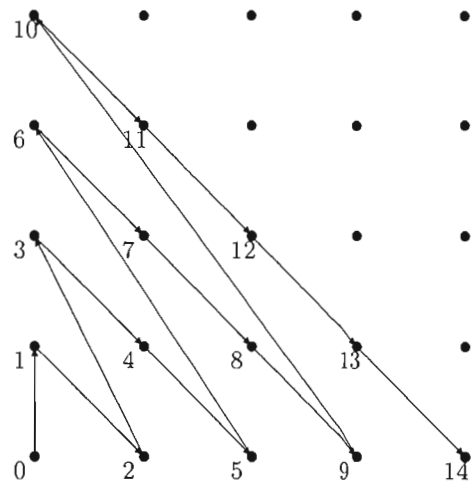
$$Cantor(a, b) = \frac{(a+b)^2 + 3a + b}{2}.$$

Lemme 3.1.1 $(*)$. *La fonction Cantor* : $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *qui à un couple d'entiers associe l'entier donné par le polynôme de Cantor est une bijection.*

Démonstration $(*)$.

L'énumération décrite par Cantor consiste à parcourir toutes les diagonales du plan discret $(\mathbb{N} \times \mathbb{N})$ joignant $(0, a)$ à $(a, 0)$, pour tout entier naturel a, dans le même sens (voir par exemple Sierpinski, 2003, p. 47-48). Il n'est pas difficile de constater que pour tout couple (a, b) d'entiers naturels, le nombre de points dans l'orthant positif du

Figure 3.1 Énumération de Cantor



plan discret situés strictement sous la diagonale passant par (a, b) est $1+2+\cdots+(a+b) = \frac{(a+b)(a+b+1)}{2}$. En ajoutant l'abscisse du couple (a, b) à $\frac{(a+b)(a+b+1)}{2}$, on obtient le nombre de couples du plan discret qui ont été « visités » par l'énumération avant d'atteindre (a, b). Ceci revient à associer à chaque couple du plan discret un nombre naturel unique. Par conséquent, la fonction

$$Cantor(a, b) = \frac{(a+b)(a+b+1)}{2} + a = \frac{(a+b)^2 + 3a + b}{2}$$

décrit une bijection entre le plan discret et les entiers naturels. □

On peut très facilement définir des relations diophantiennes permettant de récupérer à partir d'un numéro de Cantor l'un ou l'autre des éléments du couple encodé.

Définition 3.1.1. *Soient Elem*1 *et Elem*2 *les fonctions définies sur les entiers naturels par*

$$Elem1(Cantor(a, b)) = a$$

$$Elem2(Cantor(a, b)) = b.$$

Ces fonctions sont définies pour tout entier naturel puisque *Cantor* est une surjection. On peut fournir aisément une définition diophantienne de ces fonctions.

Lemme 3.1.2. *On a*

$$a = Elem1(c) \Leftrightarrow \exists y[(a+y)^2 + 3a + y = 2c] \text{ et}$$

$$b = Elem2(c) \Leftrightarrow \exists x[(x+b)^2 + 3x + b = 2c].$$

Ce lemme permet de constater que $Elem1$ et $Elem2$ sont diophantiennes. Maintenant que nous pouvons encoder et décoder des couples d'entiers naturels avec la numérotation de Cantor, nous pouvons généraliser le procédé pour des uplets de taille quelconque. Le numéro de Cantor d'un n-uplet est défini comme suit pour $n \geq 2$:

$$\begin{aligned} Cantor_2(a_1, a_2) &= Cantor(a_1, a_2) \\ Cantor_n(a_1, \ldots, a_n) &= Cantor(a_1, Cantor_{n-1}(a_2, \ldots, a_n)). \end{aligned}$$

Notons toutefois qu'il est nécessaire de connaître le nombre de composantes de l'uplet encodé pour pouvoir retrouver cet uplet à partir de son numéro de Cantor.

Lemme 3.1.3 $(*)$. *Quel que soit $n \geq 2$, $Cantor_n$ est une bijection.*

Démonstration $(*)$. On procède par récurrence sur n.

- Pour $n = 2$, on a $Cantor_2(a_1, a_2) = Cantor(a_1, a_2)$, donc $Cantor_2$ est une bijection puisque $Cantor$ en est une.
- Pour $n \geq 3$, on a $Cantor_n(a_1, \ldots, a_n) = Cantor(a_1, Cantor_{n-1}(a_2, \ldots, a_n))$. Par hypothèse de récurrence $Cantor_{n-1}$ est une bijection. De plus, $Cantor$ est une bijection. Soient $a_1, \ldots, a_n$ et $b_1, \ldots, b_n$ tels que $(a_1, \ldots, a_n) \neq (b_1, \ldots, b_n)$. On a deux possibilités :

  - Si $(a_2, \ldots, a_n) \neq (b_2, \ldots, b_n)$, on a alors

    $$Cantor_{n-1}(a_2, \ldots, b_n) \neq Cantor_{n-1}(b_2, \ldots, b_n).$$

    On a donc

    $$Cantor(a_1, Cantor_{n-1}(a_2, \ldots, a_n)) \neq Cantor(b_1, Cantor_{n-1}(b_2, \ldots, b_n)).$$

- Si $(a_2, \ldots, a_n) = (b_2, \ldots, b_n)$, alors on a nécessairement $a_1 \neq b_1$, d'où

$$Cantor(a_1, Cantor_{n-1}(a_2, \ldots, a_n)) \neq Cantor(b_1, Cantor_{n-1}(b_2, \ldots, b_n)).$$

Dans tous les cas, si $(a_1, \ldots, a_n) \neq (b_1, \ldots, b_n)$, alors

$$Cantor_n(a_1, \ldots, a_n) \neq Cantor_n(b_1, \ldots, b_n).$$

$Cantor_n$ est donc une injection.

Soit c un nombre entier quelconque, alors puisque $Cantor$ est une bijection, il existe a_1 et y tels que $c = Cantor(a_1, y)$. Par hypothèse de récurrence, $Cantor_{n-1}$ est une bijection, donc il existe $a_2, \ldots, a_n$ tels que $Cantor_{n-1}(a_2, \ldots, a_n) = y$. On a donc

$$c = Cantor(a_1, Cantor_{n-1}(a_2, \ldots, a_n)) = Cantor_n(a_1, \ldots, a_n).$$

Donc, $Cantor_n$ est une surjection. Puisque $Cantor_n$ est également une injection, alors, $Cantor_n$ est une bijection.

□

Pour être en mesure d'utiliser la numérotation de Cantor dans un système de contraintes diophantiennes, il faut parvenir à éliminer les coefficients fractionnaires du polynôme de Cantor. En effet, un polynôme diophantien ne possède que des coefficients entiers. Le lemme suivant donne un coefficient permettant l'emploi de la numérotation de Cantor dans un système diophantien..

Lemme 3.1.4 $(*)$. *Pour $n \geq 2$, le polynôme $Cantor_n(a_1, \ldots, a_n)$ peut être écrit sous la forme $\frac{P(a_1, \ldots, a_n)}{2^{2^{n-1}-1}}$, avec $P(a_1, \ldots, a_n)$ un polynôme à cofficients entiers.*

Démonstration $(*)$. On procède par récurrence sur n.

- Pour $n = 2$, on a

$$\begin{aligned} Cantor_2(a_1, a_2) &= Cantor(a_1, a_2) \\ &= \frac{(a_1 + a_2)^2 + 3a_1 + a_2}{2} \\ &= \frac{(a_1 + a_2)^2 + 3a_1 + a_2}{2^{2^{2-1}-1}}. \end{aligned}$$

La propriété est vérifiée pour le cas de base.

- Pour $n \geq 3$, on a, par définition de $Cantor_n$,

$$Cantor_n(a_1, \ldots, a_n) = Cantor(a_1, Cantor_{n-1}(a_2, \ldots, a_n)).$$

Par hypothèse de récurrence, on peut écrire

$$Cantor_{n-1}(a_2, \ldots, a_n) = \frac{P(a_2, \ldots, a_n)}{2^{2^{n-2}-1}},$$

où $P(a_2, \ldots, a_n)$ est un polynôme à coefficients entiers.

On obtient alors, par définition de $Cantor$,

$$\begin{aligned} Cantor_n(a_1, \ldots, a_n) &= Cantor(a_1, Cantor_{n-1}(a_2, \ldots, a_n)) \\ &= Cantor\left(a_1, \frac{P(a_2, \ldots, a_n)}{2^{2^{n-2}-1}}\right) \\ &= \frac{(a_1 + \frac{P(a_2,\ldots,a_n)}{2^{2^{n-2}-1}}^2) + 3a_1 + \frac{P(a_2,\ldots,a_n)}{2^{2^{n-2}-1}}}{2} \\ &= \frac{a_1^2 + 2a_1 \frac{P(a_2,\ldots,a_n)}{2^{2^{n-2}-1}} + \left(\frac{P(a_2,\ldots,a_n)}{2^{2^{n-2}-1}}\right)^2 + 3a_1 + \frac{P(a_2,\ldots,a_n)}{2^{2^{n-2}-1}}}{2} \\ &= \frac{a_1^2 + 3a_1}{2} + a_1 \frac{P(a_2, \ldots, a_n)}{2^{2^{n-2}-1}} + \frac{(\frac{P(a_2,\ldots,a_n)}{2^{2^{n-2}-1}})^2}{2} + \frac{\frac{P(a_2,\ldots,a_n)}{2^{2^{n-2}-1}}}{2} \\ &= \frac{a_1^2 + 3a_1}{2} + a_1 \frac{P(a_2, \ldots, a_n)}{2^{2^{n-2}-1}} + \frac{\frac{P^2(a_2,\ldots,a_n)}{2^{2\times 2^{n-2}-2}}}{2} + \frac{\frac{P(a_2,\ldots,a_n)}{2^{2^{n-2}-1}}}{2} \\ &= \frac{a_1^2 + 3a_1}{2} + a_1 \frac{P(a_2, \ldots, a_n)}{2^{2^{n-2}-1}} + \frac{P^2(a_2, \ldots, a_n)}{2^{2^{n-1}-1}} + \frac{P(a_2, \ldots, a_n)}{2^{2^{n-2}}} \\ &= \frac{2^{2^{n-1}-2}(a_1^2 + 3a_1) + 2^{2^{n-2}} a_1 P(a_2, \ldots, a_n) + P^2(a_2, \ldots, a_n)}{2^{2^{n-1}-1}} \\ &\quad + \frac{2^{2^{n-2}-1} P(a_2, \ldots, a_n)}{2^{2^{n-1}-1}}. \end{aligned}$$

Puisque $P(a_2, \ldots, a_n)$ est un polynôme à cofficients entiers, alors $P^2(a_2, \ldots, a_n)$ l'est aussi. D'où,

$$2^{2^{n-1}-2}(a_1^2 + 3a_1) + 2^{2^{n-2}} a_1 P(a_2, \ldots, a_n) + P^2(a_2, \ldots, a_n) + 2^{2^{n-2}-1} P(a_2, \ldots, a_n)$$

est un polynôme à coefficients entiers.

□

De ce lemme, on déduit sans difficulté le corollaire suivant.

Corollaire 3.1.5. *$2^{2^n} Cantor_n(a_1, \ldots, a_n)$ est un polynôme à coefficients entiers.*

Ceci nous permet d'utiliser une définition diophantienne pour caractériser le m-ième élément d'un uplet à n éléments encodé par $Cantor_n$.

Définition 3.1.2. *Pour tous entiers naturels n, m, x_1, ..., x_n tels que $m \leq n$, on définit la fonction $Elem_{n,m} : \mathbb{N} \to \mathbb{N}$ comme*

$$Elem_{n,m}(Cantor_n(x_1, \ldots, x_n)) = x_m.$$

D'après cette définition, il est clair qu'on a

$$a = Elem_{n,m}(c) \Leftrightarrow \exists x_1 \ldots x_{m-1} x_{m+1} \ldots x_n [Cantor_n(x_1, \ldots, x_{m-1}, a, x_{m+1}, \ldots, x_n) = c].$$

Le problème ici est que le polynôme $Cantor_n(x_1, \ldots, x_{m-1}, a, x_{m+1}, \ldots, x_n)$ est un polynôme à coefficients fractionnaires ; cette définition de $Elem_{n,m}$ n'est donc pas diophantienne. Pour résoudre ce problème, il suffit de multiplier chaque côté de l'équation par 2^{2^n}. On a alors

$$a = Elem_{n,m}(c)$$

$$\Leftrightarrow$$

$$\exists x_1 \ldots x_{m-1} x_{m+1} \ldots x_n [2^{2^n} Cantor_n(x_1, \ldots, x_{m-1}, a, x_{m+1}, \ldots, x_n) = 2^{2^n} c].$$

D'après le corollaire précédent, $2^{2^n} Cantor_n(x_1, \ldots, x_{m-1}, a, x_{m+1}, \ldots, x_n)$ est un polynôme à coefficients entiers ; donc $Elem_{n,m}$ est une fonction diophantienne.

3.2 Codage positionnel

Définition 3.2.1. *Soit un uplet quelconque* $(a_1, \ldots, a_n)$. *On dit que le triplet* (a, b, c) *est un* code positionnel *de cet uplet si et seulement si :*

- $b \geq 2$,
- $c = n$,
- *pour tout* i *compris entre 1 et* n, $b > a_i$, *et*
- $a = \sum_{i=1}^{n} a_i b^{i-1}$.

On définit la relation $Code(a, b, c)$ *si et seulement si* (a, b, c) *est un code positionnel. On appelle* a *le* numéro de code de l'uplet dans la base b. *On appelle* b *la* base d'encodage de l'uplet.

En d'autres termes, on peut dire que l'uplet codé est la représentation en base b de a. De plus, on peut constater qu'un uplet possède une infinité de codes positionnels, il suffit de choisir un b suffisamment grand. Cependant, tout triplet (a, b, c) n'est pas nécessairement un code positionnel. En effet, si tout nombre a peut être représenté de manière unique dans une base $b \geq 2$ quelconque, cela nécessite tout de même un nombre minimal de chiffres. Il faut que c soit supérieur ou égal à ce minimum.

Lemme 3.2.1 $(*)$. *Soient* $b \geq 2$ *et* n *quelconques. Si pour tout* $i \in \{1, \ldots, n\}$, *on a* $a_i < b$, *alors* $\sum_{i=1}^{n} a_i b^{i-1} < b^n$.

Démonstration $(*)$. Puisque $a_i < b$, alors $a_i \leq b - 1$. Par conséquent, on a

$$\begin{aligned}
\sum_{i=1}^{n} a_i b^{i-1} &\leq \sum_{i=1}^{n} (b-1) b^{i-1} \\
&= (b-1) \sum_{i=1}^{n} b^{i-1} \\
&= (b-1) \frac{b^n - 1}{b-1} \\
&= b^n - 1 \\
&< b^n.
\end{aligned}$$

□

Lemme 3.2.2. *Pour tous entiers naturels a, b et c, on a*

$$Code(a, b, c) \Leftrightarrow b \geq 2 \wedge a < b^c.$$

Démonstration (*). ($\Rightarrow$) Si on a $Code(a, b, c)$, alors par définition, $b \geq 2$ et $a = \sum_{i=1}^{c} a_i b^{i-1} < b^c$ (d'après le lemme 3.2.1).

($\Leftarrow$) Soient $b \geq 2$ et $a < b^c$. On procède par récurrence sur c.

- Pour $c = 0$, on a : $b^c = 1$ et puisque $a < b^c$, alors $a = 0$. Le triplet $(0, b, 0)$ est le triplet qui code l'uplet vide, on a bien $Code(a, b, c)$.
- Pour $c = 1$, on a : $b^c = b$ et $a < b$. Donc, on a bien $a = ab^0$, et le triplet $(a, b, 1)$ satisfait $Code(a, b, c)$.
- Pour $c > 1$, par division euclidienne on obtient

  $$a = a_c b^{c-1} + r$$

  avec $r < b^{c-1}$. De plus, on a $a_c < b$, car $bb^{c-1} = b^c$ et on sait que $a < b^c$. On a $r < b^{c-1}$ et $b \geq 2$, donc, par hypothèse de récurrence, on déduit que $(r, b, c-1)$ est un code positionnel et que par conséquent, $r = \sum_{i=1}^{c-1} a_i b^{i-1}$, avec $a_i < b$ pour tout $i \in \{1, \ldots, c-1\}$. On a alors

  $$\begin{aligned} a &= a_c b^{c-1} + r \\ &= a_c b^{c-1} + \sum_{i=1}^{c-1} a_i b^{i-1} \\ &= \sum_{i=1}^{c} a_i b^{i-1} \end{aligned}$$

  avec $a_i < b$ pour tout $i \in \{1, \ldots, c\}$. On a donc bien $Code(a, b, c)$.

□

Puisque l'exponentiation et la relation d'ordre sont diophantiennes, la relation *Code* est diophantienne.

Il serait maintenant commode de posséder une relation diophantienne qui permette de récupérer un élément d'un uplet à partir du code de cet uplet. La relation suivante joue ce rôle.

Définition 3.2.2. *Soit (a, b, c) un code positionnel de l'uplet $(a_1, \ldots, a_c)$. Soit Elem la fonction définie par*

$$Elem(a, b, d) = \begin{cases} a_d & \text{si } d \leq c \\ 0 & \text{sinon.} \end{cases}$$

Lemme 3.2.3. *Soit (a, b, c) un code positionnel de l'uplet $(a_1, \ldots, a_c)$. On a alors*

$$e = Elem(a, b, d) \Leftrightarrow \exists x \exists y \exists z[(d = z + 1) \wedge (a = xb^d + eb^z + y) \wedge (e < b) \wedge (y < b^z)].$$

Démonstration (*). Puisque (a, b, c) est un code positionnel de $(a_1, \ldots, a_c)$, alors

$$a = \sum_{i=1}^{c} a_i b^{i-1}$$

avec $a_i < b$ pour tout i compris entre 1 et c. Soit $a_i = 0$ pour tout $i > c$. On a alors

$$a = \sum_{i=1}^{k} a_i b^{i-1}$$

quel que soit $k \geq c$.

($\Rightarrow$) Soit $z = d - 1$. Soit $x = \sum_{i=d+1}^{c} a_i b^{i-1-d}$. Soit $y = \sum_{i=1}^{d-1} a_i b^{i-1}$. Par définition de *Elem*, on a $e = a_d < b$. On a alors

$$\begin{aligned} xb^d + eb^z + y &= \left(\sum_{i=d+1}^{c} a_i b^{i-1-d} \right) b^d + a_d b^{d-1} + \sum_{i=1}^{d-1} a_i b^{i-1} \\ &= \sum_{i=d+1}^{c} a_i b^{i-1} + a_d b^{d-1} + \sum_{i=1}^{d-1} a_i b^{i-1} \\ &= a. \end{aligned}$$

Enfin, d'après le lemme 3.2.1, on a $y = \sum_{i=1}^{d-1} a_i b^i < b^{d-1} = b^z$. Les conditions sont donc bien satisfaites et l'implication démontrée.

($\Leftarrow$) Supposons que $e = Elem(a, b, d)$. Par définition, on a

$$\exists x \exists y \exists z [d = z + 1 \wedge a = xb^d + eb^z + y \wedge e < b \wedge y < b^z].$$

Donc, $z = d - 1$ et on a :

$$\begin{aligned}
\sum_{i=1}^{c} a_i b^{i-1} &= a \\
\sum_{i=1}^{d-1} a_i b^{i-1} + \sum_{i=d+2}^{c} a_i b^{i-1} + a_{d+1} b^d + a_d b^{d-1} &= xb^d + eb^{d-1} + y \\
\sum_{i=1}^{d-1} a_i b^{i-1} + \sum_{i=1}^{c-d-1} a_{i+d+1} b^{i+d} + a_{d+1} b^d + a_d b^{d-1} &= xb^d + eb^{d-1} + y \\
\sum_{i=1}^{c-d-1} a_{i+d+1} b^{i+d} + a_{d+1} b^d + a_d b^{d-1} - xb^d - eb^{d-1} &= y - \sum_{i=1}^{d-1} a_i b^{i-1} \\
b^{d-1} \left(\sum_{i=1}^{c-d-1} a_{i+d+1} b^{i+1} + a_{d+1} b - xb - e + a_d \right) &= y - \sum_{i=1}^{d-1} a_i b^{i-1}.
\end{aligned}$$

Donc, $y - \sum_{i=1}^{d-1} a_i b^{i-1}$ est divisible par b^{d-1}. Or, $\sum_{i=1}^{d-1} a_i b^{i-1} < b^{d-1}$ puisque $a_i < b$ pour tout i et $y < b^{d-1}$. On a $-b^{d-1} < \sum_{i=1}^{d-1} a_i b^{i-1} - y < b^{d-1}$. Par conséquent, $\sum_{i=1}^{d-1} a_i b^{i-1} = y$.

On a donc :

$$\begin{aligned}
a &= xb^d + eb^{d-1} + y \\
\sum_{i=1}^{c} a_i b^{i-1} &= xb^d + eb^{d-1} + \sum_{i=1}^{d-1} a_i b^{i-1} \\
\sum_{i=1}^{d-1} a_i b^{i-1} + \sum_{i=d}^{c} a_i b^{i-1} &= xb^d + eb^{d-1} + \sum_{i=1}^{d-1} a_i b^{i-1} \\
\sum_{i=d}^{c} a_i b^{i-1} &= xb^d + eb^{d-1} \\
\sum_{i=d+1}^{c} a_i b^{i-1} + a_d b^{d-1} &= xb^d + eb^{d-1} \\
\sum_{i=1}^{c-d} a_i b^{i+d-1} + a_d b^{d-1} &= xb^d + eb^{d-1} \\
a_d b^{d-1} - eb^{d-1} &= xb^d - \sum_{i=1}^{c-d} a_i b^{i+d-1} \\
b^{d-1}(a_d - e) &= b^d \left(x - \sum_{i=1}^{c-d} a_i b^{i-1} \right) \\
a_d - e &= b \left(x - \sum_{i=1}^{c-d} a_i b^{i-1} \right).
\end{aligned}$$

Donc, $a_d - e$ est divisible par b. Or, $e < b$ et $a_d < b$. Donc, $-b < a_d - e < b$. Donc, $(a_d - e)$ est divisible par b seulement si $a_d = e$. Si $d \leq c$, alors a_d est le d-ième élément de l'uplet codé, sinon, $a_d = 0$. On a bien $e = Elem(a, b, d)$. □

L'avantage du codage positionnel est qu'il permet de définir facilement une relation diophantienne de concaténation sur des codes d'uplet à même base.

Définition 3.2.3 $(*)$. *Pour tous entiers naturels a, c, a_1, c_1, a_2, c_2 et b, on définit la relation $ConcatFaible$ comme suit :*

$$ConcatFaible(a, c, a_1, c_1, a_2, c_2, b)$$

$$\Leftrightarrow$$

$$Code(a_1, b, c_1) \wedge Code(a_2, b, c_2) \wedge a = a_2 b^{c_1} + a_1 \wedge c = c_1 + c_2.$$

Lemme 3.2.4 (*). *Soient* (a_1, b, c_1) *le code de l'uplet* $(a_{1,1}, \ldots, a_{1,c_1})$ *et* (a_2, b, c_2) *le code de l'uplet* $(a_{2,1}, \ldots, a_{2,c_2})$. *Si la relation* $ConcatFaible(a, c, a_1, c_1, a_2, c_2, b)$ *est satisfaite, alors le triplet* (a, b, c) *est un code positionnel de l'uplet* $(a_{1,1}, \ldots, a_{1,c_1}, a_{2,1}, \ldots, a_{2,c_2})$.

Démonstration (*). Par hypothèse, $a_1 = \sum_{i=1}^{c_1} a_{1,i} b^{i-1}$ et $a_2 = \sum_{i=1}^{c_2} a_{2,i} b^{i-1}$. De plus, par hypothèse, la relation $ConcatFaible(a, c, a_1, c_1, a_2, c_2, b)$ est satisfaite. Donc, on a :

$$\begin{aligned} a &= a_2 b^{c_1} + a_1 \\ &= b^{c_1} \sum_{i=1}^{c_2} a_{2,i} b^{i-1} + \sum_{i=1}^{c_1} a_{1,i} b^{i-1} \\ &= \sum_{i=1}^{c_2} a_{2,i} b^{c_1+i-1} + \sum_{i=1}^{c_1} a_{1,i} b^{i-1} \\ &= \sum_{i=c1+1}^{c_2+c1} a_{2,i} b^{i-1} + \sum_{i=1}^{c_1} a_{1,i} b^{i-1}. \end{aligned}$$

Si nous définissons $x_i = a_{1,i}$ si i est compris entre 1 et c_1 et $x_i = a_{2,i-c_1}$ si i est compris entre $c_1 + 1$ et $c_1 + c_2$, alors on obtient

$$\begin{aligned} a &= \sum_{i=c1+1}^{c_2+c1} a_{2,i} b^{i-1} + \sum_{i=1}^{c_1} a_{1,i} b^{i-1} \\ &= \sum_{i=c1+1}^{c_2+c1} x_i b^{i-1} + \sum_{i=1}^{c_1} x_i b^{i-1} \\ &= \sum_{i=1}^{c_1+c_2} x_i b^{i-1}. \end{aligned}$$

Par définition d'un code positionnel, $(a, b, c_1 + c_2)$ code l'uplet

$$(x_1, \ldots, x_{c_1}, x_{c_1+1}, \ldots, x_{c_1+c_2}).$$

Par substitution des x_i, on peut conclure que le triplet $(a, b, c_1 + c_2)$ code l'uplet

$$(a_{1,1}, \ldots, a_{1,c_1}, a_{2,1}, \ldots, a_{2,c_2}).$$

Par hypothèse, on a $c = c_1 + c_2$, ce qui permet de conclure que (a, b, c) code l'uplet précédent. □

Nous voudrions en fait obtenir une relation *Concat* qui permette de déterminer qu'un code positionnel (a, b, c) code un uplet résultant de la concaténation de deux autres uplets respectivement codés par (a_1, b_1, c_1) et (a_2, b_2, c_2). L'intérêt par rapport à la relation *ConcatFaible* est que pour *Concat*, les bases d'encodage des trois uplets pourront être différentes. Nous aborderons ce point un peu plus loin.

Le codage positionnel est utilisée par Matiiassevitch (1995) pour établir que les coefficients binomiaux, la factorielle et l'ensemble des nombres premiers sont diophantiens. Il est à noter que Julia Robinson (1952) avait établi que ces fonctions sont diophantiennes exponentielles (donc diophantiennes dans le cas où l'exponentiation est diophantienne) sans que le concept de code positionnel soit utilisé.

Lemme 3.2.5. *Pour tous entiers naturels n et i tels que $i \leq n$, on a*

$$c = C_n^i \Leftrightarrow c = Elem((2^n + 2)^n, 2^n + 1, i + 1).$$

Démonstration (*). D'après la formule du binôme de Newton, on a

$$(b+1)^n = \sum_{i=0}^{n} C_n^i b^i.$$

D'après la définition d'un codage positionnel, le triplet $((b+1)^n, b, n+1)$ est donc un code positionnel de l'uplet $(C_n^0, C_n^1, \ldots, C_n^n)$ si b est strictement supérieur à $C_n^0, C_n^1, \ldots, C_n^n$. Montrons d'abord que quels que soient n et i tels que $0 \leq i \leq n$, on a $C_n^i \leq 2^n$. Soiemt n et i quelconques. On a

$$\begin{aligned} 2^n &= (1+1)^n \\ &= \sum_{i=0}^{n} C_n^i. \end{aligned}$$

Par conséquent, on a bien $2^n \geq C_n^i$ pour tout i compris entre 0 et n. Donc, en prenant $b = 2^n + 1$, on est assuré d'avoir b strictement supérieur à $C_n^0, C_n^1, \ldots, C_n^n$. Donc, si $b = 2^n + 1$, le triplet $((b+1)^n, b, n+1) = ((2^n+2)^n, 2^n+1, n+1)$ est le code positionnel de l'uplet $(C_n^0, C_n^1, \ldots, C_n^n)$. Nous pouvons alors définir le coefficient binomial de la façon suivante

$$c = C_n^i \Leftrightarrow c = Elem((2^n + 2)^n, 2^n + 1, i + 1).$$

□

Ce lemme établit bien que le coefficient binomial est diophantien puisqu'exprimable grâce à un système de contraintes diophantiennes. On utilise immédiatement ce résultat pour établir que la factorielle est diophantienne.

Lemme 3.2.6. *Pour tous entiers naturels m et $n \geq (m+1)^{m+2}$, on a*

$$(n)^m \text{ div } C_n^m = m!.$$

La démonstration de ce lemme se base sur l'observation suivante (Matiiassevitch, 1995). Par définition du coefficient binomial, on a

$$\begin{aligned}
C_n^m &= \frac{n!}{m!(n-m)!} \\
m! &= \frac{n!}{C_n^m(n-m)!} \\
m! &= \frac{n(n-1)\dots(n-m+1)}{C_n^m} \\
m! &= \frac{n^m}{C_n^m}\left(1-\frac{1}{n}\right)\dots\left(1-\frac{m-1}{n}\right) \\
m! &= \lim_{n\to\infty}\frac{n^m}{C_n^m}\left(1-\frac{1}{n}\right)\dots\left(1-\frac{m-1}{n}\right) \\
m! &= \lim_{n\to\infty}\frac{n^m}{C_n^m}.
\end{aligned}$$

Il faut traduire le passage à la limite dans la formalisation diophantienne :

$$m! = \lim_{n\to\infty}\frac{n^m}{C_n^m}$$

signifie que pour tout m, il existe une valeur x telle que si $n \geq x$, alors $m! = n^m$ div C_n^m. Il suffit de trouver cette borne x. Au préalable, procédons à l'observation suivante.

Lemme 3.2.7 $(*)$. *Pour tout $m > 1$, $m! < (m+1)^m - 1$*

Démonstration $(*)$. On procède par récurrence sur m.

Pour $m = 2$, on a $m! = 2$ et $(m+1)^m - 1 = 3^2 - 1 = 8$. La propriété est vérifiée.

Supposons qu'elle reste vraie pour m compris entre 2 et k. Vérifions qu'elle l'est encore pour $m = k+1$. On a $(k+1)! = (k+1)k!$. Par hypothèse de récurrence, on a

$$(k+1)k! < (k+1)((k+1)^k - 1).$$

On a alors

$$\begin{aligned}(k+1)! &= (k+1)k! \\ &< (k+1)((k+1)^k - 1) \\ &= (k+1)^{k+1} - (1+k) \\ &< (k+2)^{k+1} - (1+k) \\ &\leq (k+2)^{k+1} - 1\end{aligned}$$

La propriété est donc vérifiée pour $m = k+1$ et pour tout $m > 1$. □

Nous sommes maintenant en mesure de démontrer le caractère diophantien de la factorielle.

Démonstration du lemme 3.2.6 (*). On veut montrer que $m! = n^m \text{ div } C_n^m$ pour tout $n \geq (m+1)^{m+2}$, c'est-à-dire que

$$m! = ((m+1)^{m+2} + k)^m \text{ div } C^m_{(m+1)^{m+2}+k}$$

pour tout $k \geq 0$.

- Pour $m = 0$, on a $0! = 1$ et $\frac{(1+k)^0}{C^0_{1+k}} = 1$, la proposition est vérifiée.
- Pour $m = 1$, on a $1! = 1$ et $\frac{2^3+k}{C^1_{2^3+k}} = \frac{2^3+k}{2^3+k} = 1$, la proposition est vérifiée.
- Pour $m > 1$, il s'agit de montrer que

$$m! \leq \frac{((m+1)^{(m+2)} + k)^m}{C^m_{(m+1)^{m+2}+k}} < m! + 1.$$

On a

$$\begin{aligned}
\frac{((m+1)^{(m+2)}+k)^m}{C^m_{(m+1)^{m+2}+k}} &= \frac{((m+1)^{(m+2)}+k)^m}{\left(\frac{((m+1)^{m+2}+k)!}{m!((m+1)^{m+2}+k-m)!}\right)} \\
&= \frac{((m+1)^{(m+2)}+k)^m m!((m+1)^{m+2}+k-m)!}{((m+1)^{m+2}+k)!} \\
&= \frac{m!((m+1)^{(m+2)}+k)^m}{((m+1)^{m+2}+k)\ldots((m+1)^{m+2}+k-(m-1))}.
\end{aligned}$$

Puisque $m > 1$, on a $(m+1)^{m+2}+k > (m+1)^{m+2}+k-(m-1)$. Ceci va nous permettre d'établir les inégalités souhaitées. On a d'une part

$$\begin{aligned}
\frac{((m+1)^{m+2}+k)^m}{C^m_{(m+1)^{m+2}+k}} &= \frac{((m+1)^{m+2}+k)^m}{\frac{((m+1)^{m+2}+k)!}{m!((m+1)^{m+2}+k-m)!}} \\
&= \frac{m!((m+1)^{m+2}+k)^m}{((m+1)^{m+2}+k)\ldots((m+1)^{m+2}+k-(m-1))} \\
&\geq \frac{m!((m+1)^{m+2}+k)^m}{((m+1)^{m+2}+k)\ldots((m+1)^{m+2}+k)} \\
&= \frac{m!((m+1)^{m+2}+k)^m}{((m+1)^{m+2}+k)^m} \\
&= m!\left(\frac{(m+1)^{m+2}+k}{(m+1)^{m+2}+k}\right)^m \\
&= m!.
\end{aligned}$$

On a donc

$$\frac{((m+1)^{m+2}+k)^m}{C^m_{(m+1)^{m+2}+k}} \geq m!.$$

D'autre part, on a

$$(m+1)^{m+2}+k-1 \geq (m+1)^{m+2}+k-(m-1)$$

car $m > 1$, d'où

$$\begin{aligned}
\frac{((m+1)^{m+2}+k)^m}{C^m_{(m+1)^{m+2}+k}} &= \frac{((m+1)^{m+2}+k)^m}{\frac{((m+1)^{m+2}+k)!}{m!((m+1)^{m+2}+k-m)!}} \\
&= \frac{m!((m+1)^{m+2}+k)^{m-1}}{((m+1)^{m+2}+k-1)\dots((m+1)^{m+2}+k-(m-1))} \\
&\leq \frac{m!((m+1)^{m+2}+k)^{m-1}}{((m+1)^{m+2}+k-(m-1))\dots((m+1)^{m+2}+k-(m-1))} \\
&= \frac{m!((m+1)^{m+2}+k)^{m-1}}{((m+1)^{m+2}+k-(m-1))^{m-1}} \\
&= m!\left(\frac{(m+1)^{m+2}+k}{(m+1)^{m+2}+k-(m-1)}\right)^{m-1}.
\end{aligned}$$

On veut montrer que

$$m!\left(\frac{(m+1)^{m+2}+k}{(m+1)^{m+2}+k-(m-1)}\right)^{m-1} < m!+1,$$

ce qui est équivalent à

$$\left(\frac{(m+1)^{m+2}+k}{(m+1)^{m+2}+k-(m-1)}\right)^{m-1} < 1+\frac{1}{m!}.$$

On a

$$\left(\frac{(m+1)^{m+2}+k}{(m+1)^{m+2}+k-(m-1)}\right)^{m-1} = \frac{1}{\left(1-\frac{m-1}{(m+1)^{m+2}+k}\right)^{m-1}}.$$

D'après le lemme 2.6.5, on a $(1-\alpha)^\ell > (1-\ell\alpha)$ pour tout $\ell \geq 0$ et tout α tel que $0 < \alpha < 1$. Or

$$0 < \frac{(m-1)}{(m+1)^{m+2}} < 1,$$

donc, on a

$$\left(1-\frac{(m-1)}{(m+1)^{m+2}+k}\right)^{m-1} > \left(1-\frac{(m-1)^2}{(m+1)^{m+2}+k}\right).$$

On obtient alors

$$\begin{aligned}
\left(\frac{(m+1)^{m+2}+k}{(m+1)^{m+2}+k-(m-1)}\right)^{m-1} &= \frac{1}{\left(1-\frac{(m-1)}{(m+1)^{m+2}+k}\right)^{m-1}} \\
&< \frac{1}{1-\frac{(m-1)^2}{(m+1)^{m+2}+k}}.
\end{aligned}$$

On va maintenant prouver que

$$\frac{1}{1-\frac{(m-1)^2}{(m+1)^{m+2}+k}} < 1 + \frac{1}{m!}.$$

Ceci revient à prouver que

$$\frac{1}{1-\frac{(m-1)^2}{(m+1)^{m+2}+k}} - 1 < \frac{1}{m!}.$$

On a

$$\begin{aligned}
\frac{1}{1-\frac{(m-1)^2}{(m+1)^{m+2}+k}} - 1 &= \frac{1-\left(1-\frac{(m-1)^2}{(m+1)^{m+2}+k}\right)}{1-\frac{(m-1)^2}{(m+1)^{m+2}+k}} \\
&= \frac{\frac{(m-1)^2}{(m+1)^{m+2}+k}}{1-\frac{(m-1)^2}{(m+1)^{m+2}+k}} \\
&= \frac{1}{\frac{(m+1)^{m+2}+k}{(m-1)^2} - 1}.
\end{aligned}$$

Puisque $m-1 < m+1$, alors on a

$$\frac{(m+1)^{m+2}+k}{(m-1)^2} > \frac{(m+1)^{m+2}+k}{(m+1)^2} \geq (m+1)^m.$$

Donc, on a

$$\frac{1}{\frac{(m+1)^{m+2}+k}{(m-1)^2} - 1} < \frac{1}{(m+1)^m - 1}.$$

Puisque $m \geq 2$, alors, d'après le lemme 3.2.7, on a $m! < (m+1)^m - 1$. Par conséquent, on a

$$\frac{1}{(m+1)^m - 1} < \frac{1}{m!},$$

et donc

$$\frac{1}{1-\frac{(m-1)^2}{(m+1)^{m+2}+k}} - 1 < \frac{1}{m!},$$

d'où

$$\frac{1}{1-\frac{(m-1)^2}{(m+1)^{m+2}+k}} < 1 + \frac{1}{m!}.$$

On a donc bien

$$m!\left(\frac{1}{1-\frac{(m-1)^2}{(m+1)^{m+2}+k}}\right) < m! + 1.$$

Ceci permet de conclure que

$$\frac{((m+1)^{m+2}+k)^m}{C^m_{(m+1)^{m+2}+k}} < m! + 1.$$

On a donc établi les inégalités

$$m! \leq \frac{((m+1)^{m+2}+k)^m}{C^m_{(m+1)^{m+2}+k}} < m! + 1,$$

ce qui permet d'affirmer que

$$((m+1)^{m+2}+k)^m \text{ div } C^m_{(m+1)^{m+2}+k} = m!.$$

□

Ce résultat révèle que la factorielle est diophantienne puisque pouvant être définie par une expression diophantienne. Ce résultat va maintenant nous servir à prouver que l'ensemble des entiers naturels premiers est diophantien.

Définition 3.2.4. *Pour tout entier naturel n, on définit la relation suivante : $Prime(n)$ si et seulement si n est premier.*

Lemme 3.2.8. *Pour tout entier naturel n, on a $Prime(n) \Leftrightarrow pgcd(n, (n-1)!) = 1$.*

Démonstration (*). Un nombre est premier si et seulement s'il n'est divisible que par 1 et par lui-même. Un nombre entier positif ne peut être divisé que par un nombre entier positif qui lui est inférieur ou égal. Par conséquent, un nombre entier positif n est premier si et seulement si aucun nombre entier positif inférieur ou égal à n ne divise n. D'où n est premier si et seulement si n et $(n-1)!$ ne possèdent aucun diviseur commun, c'est-à-dire si et seulement si $pgcd(n, (n-1)!) = 1$. Le prédicat $Prime(n)$ peut être défini par

$$Prime(n) \Leftrightarrow pgcd(n, (n-1)!) = 1.$$

□

Puisque le *pgcd* et la factorielle sont diophantiens et que la composition de fonctions diophantiennes est diophantienne, l'ensemble des nombres premiers est bien diophantien.

3.3 Comparer les uplets

L'objectif est maintenant de se munir d'outils diophantiens permettant de comparer des uplets en ne considérant que leurs codes.

On souhaite essentiellement pouvoir déterminer que deux codes positionnels codent le même uplet (on dit alors qu'ils sont équivalents).

Définition 3.3.1. *La relation* $Equal(a_1, b_1, c_1, a_2, b_2, c_2)$ *est satisfaite si et seulement si* (a_1, b_1, c_1) *et* (a_2, b_2, c_2) *sont des codes positionnels équivalents.*

Nous utiliserons également les relations *NotGreater* et *Small* définies comme suit

Définition 3.3.2. *Pour tous entiers naturels* a_1, b_1, a_2 *et* b_2, *on définit la relation NotGreater par*

$$NotGreater(a_1, b_1, a_2, b_2) \Leftrightarrow \forall k[Elem(a_1, b_1, k) \leq Elem(a_2, b_2, k)].$$

La relation *NotGreater* permet de vérifier que tout élément de l'uplet codé par (a_1, b_1, k) est inférieur ou égal à l'élément correspondant de l'uplet codé par (a_2, b_2, k).

Définition 3.3.3. *Pour tous entiers naturels* a_1, b_1, a_2 *et* b_2, *on définit la relation Small par*

$$Small(a, b, c, e) \Leftrightarrow Code(a, b, c) \wedge \forall k[Elem(a, b, k) \leq e].$$

La relation *Small* permet de vérifier que tout élément de l'uplet codé par (a, b, c) est inférieur ou égal à e.

Nous allons montrer que les trois relations *Equal*, *NotGreater* et *Small* sont diophantiennes. Pour ce faire, nous allons définir des relations plus fortes *PNotGreater* et *PSmall* qui sont satisfaites respectivement lorsque *NotGreater* et *Small* sont satisfaites et que la base d'encodage b est un nombre premier. En utilisant ces deux nouvelles

relations, nous prouverons que *Equal* est diophantienne puis que *NotGreater* et *Small* sont diophantiennes.

Définition 3.3.4. *Pour tous entiers naturels a_1, a_2 et b, on définit la relation PNotGreater par*

$$PNotGreater(a_1, a_2, b) \Leftrightarrow Prime(b) \wedge NotGreater(a_1, b, a_2, b).$$

Démontrons pour commencer une propriété simple de cette relation.

Lemme 3.3.1 $(*)$. *Soient (a_1, b, c_1) et (a_2, b, c_2) des codes positionnels des uplets $(a_{1,1}, \ldots, a_{1,c_1})$ et $(a_{2,1}, \ldots, a_{2,c_2})$. Si $NotGreater(a_1, b, a_2, b)$ est vérifiée, alors $a_1 \leq a_2$.*

Démonstration $(*)$. On a

$$a_1 = \sum_{i=1}^{c_1} a_{1,i} b^{i-1} = \sum_{i \geq 1} Elem(a_1, b_1, i) b^{i-1}$$

et

$$a_2 = \sum_{i=1}^{c_2} a_{2,i} b^{i-1} = \sum_{i \geq 1} Elem(a_2, b_2, i) b^{i-1}.$$

Puisque $NotGreater(a_1, b, a_2, b)$ est satisfaite, on a

$$\forall i \geq 1 [Elem(a_1, b, i) \leq Elem(a_2, b, i)].$$

D'où

$$\begin{aligned} a_2 - a_1 &= \sum_{i \geq 1} Elem(a_2, b, i) b^{i-1} - \sum_{i \geq 1} Elem(a_1, b, i) b^{i-1} \\ &= \sum_{i \geq 1} (Elem(a_2, b_2, i) - Elem(a_1, b_1, i)) b^{i-1} \\ &\geq 0. \end{aligned}$$

Donc, $a_2 \geq a_1$. □

D'après la définition de *PNotGreater*, si $PNotGreater(a_1, a_2, b)$ est satisfaite, alors $NotGreater(a_1, b, a_2, b)$ est satisfaite et donc $a_1 \leq a_2$.

Lemme 3.3.2.

$$PNotGreater(a_1, a_2, b) \Leftrightarrow Prime(b) \wedge b \nmid C_{a_2}^{a_1}.$$

Démonstration (*). Soient (a_1, b, c_1) et (a_2, b, c_2) les codes positionnels des uplets $(a_{1,1}, \ldots, a_{1,c_1})$ et $(a_{2,1}, \ldots, a_{2,c_2})$. Pour tout $i > c_1$, on définit $a_{1,i} = 0$. Pour tout $i > c_2$, on définit $a_{2,i} = 0$.

($\Rightarrow$) Par définition, $PNotGreater(a_1, a_2, b)$ implique $Prime(b)$ et $NotGreater(a_1, b, a_2, b)$. On a donc, pour tout entier naturel $i \geq 1$, $a_{1,i} \leq a_{2,i}$, d'où il existe $k_i \geq 0$ tel que $a_{2,i} = a_{1,i} + k_i$, avec $k_i \leq a_{2,i} < b$.

Soit $k = \sum_{i \geq 1} k_i b^{i-1}$. On a

$$\begin{aligned} a_1 + k &= \sum_{i \geq 1} (a_{1,i} + k_i) b^{i-1} \\ &= \sum_{i \geq 1} a_{2,i} b^{i-1} \\ &= a_2. \end{aligned}$$

Pour additionner a_1 et k en base b, on n'a pas besoin d'utiliser de retenue, puisque quel que soit $i > 0$, $a_{2,i} < b$ par définition d'un code positionnel. Donc, d'après le théorème de Kummer (voir 1.1.4), l'exposant de b dans la décomposition de $C_{a_2}^{a_1} = C_{a_1+k}^{a_1}$ en produit de facteurs premiers est 0. Ceci implique que b ne divise pas $C_{a_2}^{a_1}$. On a donc $PNotGreater(a_1, a_2, b) \Rightarrow Prime(b) \wedge b \nmid C_{a_2}^{a_1}$.

($\Leftarrow$) On a $Prime(b) \wedge b \nmid C_{a_2}^{a_1}$. On sait que b ne divise pas $C_{a_2}^{a_1}$, donc $C_{a_2}^{a_1} \neq 0$. Or, par définition du coefficient binomial, si $a_1 > a_2$, alors $C_{a_2}^{a_1} = 0$. Donc, on a $a_1 \leq a_2$. Soit k tel que $a_2 = a_1 + k$ et $k = \sum_{i \geq 1} k_i b^{i-1}$ avec quel que soit $i \geq 1$, $0 \leq k_i < b$.

On a

$$a_1 = \sum_{i \geq 1} a_{1,i} b^{i-1}$$

$$k = \sum_{i \geq 1} k_i b^{i-1}$$

d'où

$$a_2 = a_1 + k = \sum_{i \geq 1} (a_{1,i} + k_i) b^{i-1}.$$

On sait que $b \nmid C_{a_2}^{a_1}$, ce qui implique que l'exposant de b dans la décomposition en facteurs premiers de $C_{a_2}^{a_1}$ est 0 ; donc d'après le théorème de Kummer, le nombre de retenues nécessaires pour additionner a_1 et k en base b est 0, ce qui implique que quel que soit $i \geq 0$, $a_{1,i} + k_i < b$. Par unicité de la représentation d'un nombre dans une base b, on a nécessairement $a_{2,i} = a_{1,i} + k_i$ pour tout $i \geq 1$. Donc, pour tout $i \geq 1$, on a $a_{2,i} \geq a_{1,i}$. On a donc $NotGreater(a_1, b, a_2, b)$. □

Définition 3.3.5. *Pour tous entiers naturels a, b, c et e, on définit la fonction PSmall comme suit :*

$$PSmall(a, b, c, e) \Leftrightarrow Prime(b) \wedge Small(a, b, c, e).$$

Définition 3.3.6. *Pour tous entiers naturels p, q et r, on définit la fonction Repeat comme suit :*

$$Repeat(p, q, r) = p \cdot \frac{q^r - 1}{q - 1}.$$

Lemme 3.3.3. *Soient p et q des entiers naturels tels que $p < q$, alors $(Repeat(p, q, r), q, r)$ est le code positionnel du r-uplet $(p, \ldots, p)$.*

Démonstration (*).

$$\begin{aligned} Repeat(p, q, r) &= p \cdot \frac{q^r - 1}{q - 1} \\ &= p \sum_{i=0}^{r-1} q^i \\ &= p \sum_{i=1}^{r} q^{i-1} \\ &= \sum_{i=1}^{r} pq^{i-1}. \end{aligned}$$

Par définition d'un code positionnel, on constate que la propriété est vraie. □

Lemme 3.3.4. *Pour tous entiers naturels a, b, c et e, on a*

$$PSmall(a, b, c, e)$$

$$\Leftrightarrow$$

$$Prime(b) \wedge [e \geq b \vee PNotGreater(a, Repeat(e, b, c), b)].$$

Démonstration. ($\Rightarrow$) Supposons que $PSmall(a, b, c, e)$ soit vérifiée. On a alors $Prime(b)$ et $Small(a, b, c, e)$ par définition de $PSmall$. Puisqu'on a $Small(a, b, c, e)$, alors par définition de $Small$, on a $\forall k[Elem(a, b, k) \leq e]$. On veut maintenant vérifier que l'expression $[e \geq b \vee PNotGreater(a, Repeat(e, b, c), b)]$ est vraie si $PSmall(a, b, c, e)$ est vraie.

- Si $e \geq b$, alors l'expression est vraie.
- Si $e < b$, alors d'après le lemme 3.3.3, $(Repeat(e, b, c), b, c)$ est le code positionnel du c-uplet dont tous les composants valent e. Or, puisque $\forall k[Elem(a, b, k) \leq e]$, on a $\forall k[Elem(a, b, k) \leq Elem(Repeat(e, b, c), b, k)]$ car $(Repeat(e, b, c), b, c)$ est le code d'un uplet ne contenant que le nombre e. De plus, la relation $Prime(b)$ est satisfaite, ce qui permet de conclure, par définition de $PNotGreater$, que la relation $PNotGreater(a, Repeat(e, b, c), b)$ est satisfaite.

Dans tous les cas, l'expression $[e \geq b \vee PNotGreater(a, Repeat(e, b, c), b)]$ est satisfaite. Enfin, par définition de $PSmall$, $Code(a, b, c)$ est également satisfaite.

($\Leftarrow$) Supposons que les relations $Prime(b)$, $[e \geq b \vee PNotGreater(a, Repeat(e, b, c), b)]$ et $Code(a, b, c)$ soient vérifiées.

- Si $PNotGreater(a, Repeat(e, b, c), b)$ est vérifiée, par définition on a

  $$\forall k[Elem(a, b, k) \leq Elem(Repeat(e, b, c), b, k].$$

  Or $(Repeat(e, b, c), b, c)$ est le code d'un uplet ne contenant que le nombre e, donc on a $\forall k[Elem(Repeat(e, b, k), b, c) \leq k]$, puisque $Elem(Repeat(e, b, k), b, c = 0$ ou e. Par définition, on a $Small(a, b, c, e)$ et puisqu'on a également $Prime(b)$, on a, par définition, $PSmall(a, b, c, e)$.
- Si $PNotGreater(a, Repeat(e, b, c), b)$ n'est pas vérifiée, alors on a nécessairement $e \geq b$. Puisque (a, b, c) est un code positionnel, alors $\forall k[Elem(a, b, k) < b]$. On a

donc $\forall k[e \geq b > Elem(a,b,k)]$, d'où $\forall k[e > Elem(a,b,k)]$. Puisqu'en plus on a $Prime(b)$ et $Code(a,b,c)$, on obtient par définition $PSmall(a,b,c,e)$.

Dans tous les cas, l'implication est vérifiée. L'équivalence est donc établie. □

Le lemme précédent nous permet d'affirmer que *PSmall* est diophantienne puisque *PNotGreater*, *Prime* et *Code* le sont.

La proposition suivante sera utile dans la démonstration à venir.

Lemme 3.3.5 (∗). *Soient a, b et n des entiers naturels tels que $a \neq b$.*
Alors $a^n - b^n$ est divisible par $a - b$.

Démonstration (∗). Il suffit de constater que

$$(a-b)\left(\sum_{i=0}^{n} a^i b^{n-i}\right) = a^{n+1} - b^{n+1}.$$

□

Lemme 3.3.6. *Soient b_1, c_1, a_2, b_2, c_2 tels que $PSmall(a_2, b_2, c_2, b_1 - 1)$, $c_1 = c_2$ et $b_1^{c_1} + b_1 < b_2$ soient vérifiées. Alors il n'existe qu'une valeur a_1 qui satisfasse $Code(a_1, b_1, c_1)$ et $a_1 \equiv a_2 \pmod{b_2 - b_1}$. Cette valeur de a_1 est telle que (a_1, b_1, c_1) et (a_2, b_2, c_2) sont des codes positionnels du même uplet.*

Démonstration (∗). Montrons dans un premier temps que a_1 existe toujours. Puisqu'on a $PSmall(a_2, b_2, c_2, b_1 - 1)$, alors par définition, on sait que (a_2, b_2, c_2) est un code positionnel. Soit $(a_{2,1}, \ldots, a_{2,n})$ l'uplet codé par (a_2, b_2, c_2). De plus, puisqu'on a $PSmall(a_2, b_2, c_2, b_1 - 1)$, alors par définition $\forall k[a_{2,k} \leq b_1 - 1]$. Soit

$$a_1 = \sum_{i=1}^{c_2} a_{2,i} b_1^{i-1} = \sum_{i=1}^{c_1} a_{2,i} b_1^{i-1}.$$

On a bien, par définition d'un code positionnel, que (a_1, b_1, c_1) est un code positionnel. De plus, a_1 est le code positionnel de l'uplet $(a_{2,1}, \ldots, a_{2,n})$, c'est à dire l'uplet codé par

(a_2, b_2, c_2). On a alors

$$a_2 - a_1 = \sum_{i=1}^{c_1} a_{2,i}(b_2^{i-1} - b_1^{i-1}).$$

D'après le lemme 3.3.5, $b_2^n - b_1^n$ est divisible par $b_2 - b_1$ quel que soit $n \geq 0$. On a donc $a_2 - a_1$ divisible par $b_2 - b_1$. Donc, $a_1 \equiv a_2 \pmod{b_2 - b_1}$. Il existe bien une valeur de a_1 telle que $Code(a_1, b_1, c_1)$ et $a_1 \equiv a_2 \pmod{b_1 - b_2}$ soient vérifiées.

Montrons maintenant que cette solution est unique. Soit a_1 satisfaisant $Code(a_1, b_1, c_1)$ et $a_1 \equiv a_2 \pmod{b_2 - b_1}$. Soit a'_1 satisfaisant $Code(a'_1, b_1, c_1)$ et $a'_1 \equiv a_2 \pmod{b_2 - b_1}$. Par définition de $Code$, on a

$$\begin{aligned} a_1 &< b_1^{c_1} \\ a'_1 &< b_1^{c_1}. \end{aligned}$$

Or, puisque $b_1^{c_1} + b_1 < b_2$, on obtient

$$b_2 - b_1 > b_1^{c_1} > a_1$$

$$b_2 - b_1 > b_1^{c_1} > a'_1.$$

On a donc

$$-(b_2 - b_1) < a_1 - a'_1 < b_2 - b_1.$$

De plus, nous avons

$$\begin{aligned} a_1 &\equiv a_2 \pmod{b_2 - b_1} \\ \text{et } a'_1 &\equiv a_2 \pmod{b_2 - b_1}, \end{aligned}$$

ce qui signifie que $(b_2 - b_1)$ divise $a_1 - a_2$ et $a'_1 - a_2$. Il existe donc $k \geq 0$ et $k' \geq 0$ tels que

$$\begin{aligned} a_1 - a_2 &= k(b_2 - b_1) \\ \text{et } a'_1 - a_2 &= k'(b_2 - b_1), \end{aligned}$$

d'où

$$a_1 - a'_1 = (k - k')(b_2 - b_1).$$

Donc, $a_1 - a'_1$ est divisible par $b_2 - b_1$ (qui est différent de 0 puisque $b_1^{c_1} + b_1 < b_2$). Or, on a $-(b_2 - b_1) < a_1 - a'_1 < b_2 - b_1$. Donc, on a nécessairement $a_1 - a'_1 = 0$, d'où $a_1 = a'_1$. Donc, s'il existe une solution, cette solution est unique. □

Nous allons maintenant introduire une nouvelle relation Eq telle que si Eq est satisfaite, alors $Equal$ est satisfaite. L'avantage de Eq est qu'elle peut être définie dans la formalisation diophantienne. Nous redéfinirons alors $Equal$ par rapport à Eq et nous obtiendrons une formalisation diophantienne de $Equal$.

Définition 3.3.7. *Pour tous entiers naturels a_1, b_1, c_1, a_2, b_2 et c_2, on a*

$$\begin{aligned} Eq(a_1,b_1,c_1,a_2,b_2,c_2) \Leftrightarrow\ & Code(a_1,b_1,c_1) \wedge Code(a_2,b_2,c_2) \wedge c_1 = c_2 \\ & \wedge\ PSmall(a_2,b_2,c_2,b_1-1) \wedge b_1^{c_1} + b_1 < b_2 \\ & \wedge\ a_1 \equiv a_2 \pmod{b_2 - b_1}. \end{aligned}$$

D'après le lemme 3.3.6, si a_1, b_1, c_1, a_2, b_2, c_2 satisfont $Eq(a_1,b_1,c_1,a_2,b_2,c_2)$, alors (a_1,b_1,c_1) et (a_2,b_2,c_2) sont des codes positionnels d'un même uplet. On a donc

$$Eq(a_1,b_1,c_1,a_2,b_2,c_2) \Rightarrow Equal(a_1,b_1,c_1,a_2,b_2,c_2)$$

Dès lors, il est facile de fournir une définition diophantienne du prédicat $Equal$.

Lemme 3.3.7. *Pour tous entiers naturels a_1, b_1, c_1, a_2, b_2 et c_2, on a*

$$Equal(a_1,b_1,c_1,a_2,b_2,c_2) \Leftrightarrow \exists x \exists y \exists z [Eq(a_1,b_1,c_1,x,y,z) \wedge Eq(a_2,b_2,c_2,x,y,z)].$$

Démonstration (*). ($\Rightarrow$) Si $Equal(a_1,b_1,c_1,a_2,b_2,c_2)$ est satisfaite, alors (a_1,b_1,c_1) et (a_2,b_2,c_2) sont des codes positionnels du même uplet $(x_1,\ldots,x_n)$, avec $n = c_1 = c_2$. Soient x, y et z tels que soient satisfaites les conditions

$$Prime(y) \wedge y > b_1^{c_1} + b_1 \wedge y > b_2^{c_2} + b_2$$

$$x = \sum_{i=1}^{n} x_i y^{i-1}$$

$$z = n = c_1 = c_2.$$

Notons qu'il est toujours possible de trouver un y satisfaisant les conditions ci-dessus dans la mesure où il existe une infinité de nombres premiers. Par définition de $Equal$, on a $Code(a_1,b_1,c_1)$ et $Code(a_2,b_2,c_2)$. On a donc $\forall k[x_k < b_1 \wedge x_k < b_2]$ et puisque

$y > b_1$ et $y > b_2$, alors $\forall k[x_k < y]$. De plus, $z = n$. Donc, par définition d'un code positionnel, $Code(x, y, z)$ est satisfaite. Puisque y est premier, on a alors, par définition, $PSmall(x, y, z, b_1 - 1)$ et $PSmall(x, y, z, b_2 - 1)$. Enfin, on a

$$a_1 - x = \sum_{i=1}^{n} x_i {b_1}^{i-1} - \sum_{i=1}^{n} x_i y^{i-1} = \sum_{i=1}^{n} x_i (b_1^{i-1} - y^{i-1})$$

et

$$a_2 - x = \sum_{i=1}^{n} x_i {b_2}^{i-1} - \sum_{i=1}^{n} x_i y^{i-1} = \sum_{i=1}^{n} x_i (b_2^{i-1} - y^{i-1})$$

D'où, d'après le lemme 3.3.5, $a_1 - x$ et $a_2 - x$ sont respectivement divisibles par $b_1 - y$ et $b_2 - y$. On a donc

$$a_1 \equiv x \pmod{y - b_1} \text{ et}$$

$$a_2 \equiv x \pmod{y - b_2}.$$

On a donc, par définition de la relation Eq

$$Eq(a_1, b_1, c_1, x, y, z) \text{ et}$$

$$Eq(a_2, b_2, c_2, x, y, z).$$

Donc,

$$Equal(a_1, b_1, c_1, a_2, b_2, c_2) \Rightarrow \exists x \exists y \exists z [Eq(a_1, b_1, c_1, x, y, z) \wedge Eq(a_2, b_2, c_2, x, y, z)].$$

($\Leftarrow$) S'il existe x, y et z tels que $Eq(a_1, b_1, c_1, x, y, z)$ et $Eq(a_2, b_2, c_2, x, y, z)$ sont satisfaites, alors d'après le lemme 3.3.6, (a_1, b_1, c_1) et (x, y, z) sont des codes positionnels d'un même uplet et (a_2, b_2, c_2) et (x, y, z) sont des codes positionnels d'un même uplet. Donc, (a_1, b_1, c_1) et (a_2, b_2, c_2) sont des codes positionnels d'un même uplet et par définition, la relation $Equal(a_1, b_1, c_1, a_2, b_2, c_2)$ est satisfaite. □

Ce lemme nous permet d'affirmer que la relation *Equal* est diophantienne dans la mesure où la relation Eq est diophantienne. En utilisant ce fait, nous pouvons prouver que les relations *NotGreater* et *Small* sont également diophantiennes : il suffit d'utiliser *Equal* pour changer les bases d'encodage des uplets afin d'obtenir des codes positionnels dont la base est première.

Lemme 3.3.8.

$$NotGreater(a_1, b_1, a_2, b_2)$$

$$\Leftrightarrow$$

$$\exists x_1 \exists x_2 \exists y \exists z [Equal(a_1, b_1, z, x_1, y, z) \wedge Equal(a_2, b_2, z, x_2, y, z) \wedge PNotGreater(x_1, x_2, y)].$$

Démonstration (*). ($\Rightarrow$) Supposons que $NotGreater(a_1, b_1, a_2, b_2)$ soit satisfaite. Soit k tel que (a_1, b_1, k) et (a_2, b_2, k) soient des codes positionnels. Soient $(u_1, \ldots, u_k)$ et $(v_1, \ldots, v_k)$ les uplets respectivement codés par (a_1, b_1, k) et (a_2, b_2, k). Soit y un nombre premier tel que $y > b_1$ et $y > b_2$. Soit x_1 tel que (x_1, y, z) soit un code positionnel de $(u_1, \ldots, u_z)$. Soit x_2 tel que (x_2, y, z) soit un code positionnel de $(u_1, \ldots, v_z)$. On a, par définition $Equal(a_1, b_1, z, x_1, y, z)$ et $Equal(a_2, b_2, z, x_2, y, z)$. De plus, puisque $NotGreater(a_1, b_1, a_2, b_2)$ est satisfaite, on a $\forall k[u_k \leq v_k]$. Puisque y est premier, on a par définition $PNotGreater(x_1, x_2, y)$.

($\Leftarrow$) Supposons qu'il existe x_1, x_2, y et z tels que soient satisfaites

$$Equal(a_1, b_1, z, x_1, y, z)$$

$$Equal(a_2, b_2, z, x_2, y, z) \text{ et}$$

$$PNotGreater(x_1, x_2, y).$$

Alors, par définition, (a_1, b_1, z) et (x_1, y, z) sont des codes positionnels du même uplet $(u_1, \ldots, u_z)$ et (a_2, b_2, z) et (x_2, y, z) sont des codes positionnels du même uplet $(v_1, \ldots, v_z)$. De plus, puisque $PNotGreater(x_1, x_2, y)$ est satisfaite alors par définition $NotGreater(x_1, y, x_2, y)$ est satisfaite. Or puisque d'une part (x_1, y, z) et (a_1, b_1, z) codent le même uplet et que d'autre part (x_2, y, z) et (a_2, b_2, z) codent le même uplet, alors on a $NotGreater(a_1, b_1, a_2, b_2)$. □

Puisque *Equal* et *PNotGreater* sont diophantiennes, alors la relation *NotGreater* est diophantienne.

Lemme 3.3.9. $Small(a, b, c, e) \Leftrightarrow \exists x \exists y [Equal(a, b, c, x, y, c) \wedge PSmall(x, y, c, e)]$

Démonstration (*). ($\Rightarrow$) Supposons que $Small(a,b,c,e)$ soit satisfaite. Par définition, (a,b,c) est un code. Soit $(u_1,\ldots,u_c)$ l'uplet codé par (a,b,c). Soit y un nombre premier strictement supérieur à b. Soit x tel que (x,y,c) soit un code positionnel de $(u_1,\ldots,u_c)$. Par définition, on a $Equal(a,b,c,x,y,c)$. Donc puisqu'on a $Small(a,b,c,e)$, on a forcément $Small(x,y,c,e)$. De plus, puisque y est premier, on a par définition $PSmall(x,y,c,e)$.

($\Leftarrow$) Supposons que $Equal(a,b,c,x,y,c)$ et $PSmall(x,y,c,e)$ soient satisfaites. Cela implique que (a,b,c) et (x,y,c) sont des codes positionnels du même uplet $(u_1,\ldots,u_c)$. De plus, $PSmall(x,y,c,e)$ est satisfaite, ce qui implique que $\forall k[u_k \leq e]$. Puisque (a,b,c) est un code positionnel de $(u_1,\ldots,u_k)$, alors par définition, $Small(a,b,c,e)$ est satisfaite. □

Puisque *Equal* et *PSmall* sont diophantiennes, alors *Small* est diophantienne. Nous achevons ce chapitre en définissant la relation de concaténation en termes diophantiens.

Définition 3.3.8. *Soient* (a_1,b_1,c_1) *et* (a_2,b_2,c_2) *des codes positionnels des uplets* $(a_{1,1},\ldots,a_{1,c_1})$ *et* $(a_{2,1},\ldots,a_{2,c_2})$. *La relation* $Concat(a,b,c,a_1,b_1,c_1,a_2,b_2,c_2)$ *est satisfaite si et seulement si* (a,b,c) *est un code positionnel de l'uplet* $(a_{1,1},\ldots,a_{1,c_1},a_{2,1},\ldots,a_{2,c_2})$.

Maintenant, prouvons que cette définition peut être reformulée en termes diophantiens.

Lemme 3.3.10.

$$Concat(a,b,c,a_1,b_1,c_1,a_2,b_2,c_2)$$

$$\Leftrightarrow$$

$$\exists x_1 \exists x_2 [Equal(a_1,b_1,c_1,x_1,b,c_1) \wedge Equal(a_2,b_2,c_2,x_2,b,c_2)$$

$$\wedge ConcatFaible(a,c,x_1,c_1,x_2,c_2,b)].$$

Démonstration (*). ($\Rightarrow$) Si $Concat(a, b, c, a_1, b_1, c_1, a_2, b_2, c_2)$ est satisfaite, alors, par définition, (a_1, b_1, c_1) et (a_2, b_2, c_2) sont des codes positionnels d'uplets $(a_{1,1}, \ldots, a_{1,c_1})$ et $(a_{2,1}, \ldots, a_{2,c_2})$. De plus, (a, b, c) est un code positionnel d'un uplet $(a_{1,1}, \ldots, a_{1,c_1}, a_{2,1}, \ldots, a_{2,c_2})$. Par définition d'un code positionnel, on a

$$\forall i \geq 1[b > a_{1,i} \wedge b > a_{2,i}].$$

Soient

$$x_1 = \sum_{i=1}^{c_1} a_{1,i} b^{i-1} \text{ et}$$

$$x_2 = \sum_{i=1}^{c_2} a_{2,i} b^{i-1}.$$

Par définition d'un code positionnel, on a (x_1, b, c_1) et (x_2, b, c_2) des codes positionnels des uplets $(a_{1,1}, \ldots, a_{1,c_1})$ et $(a_{2,1}, \ldots, a_{2,c_2})$ respectivement. Par définition de *Equal*, les relations $Equal(a_1, b_1, c_1, x_1, b, c_1)$ et $Equal(a_2, b_2, c_2, x_2, b, c_2)$ sont satisfaites. Puisque $Concat(a, b, c, a_1, b_1, c_1, a_2, b_2, c_2)$ est satisfaite et que (a, b, c) est le code positionnel de l'uplet formé par la concaténation des uplets codés par (x_1, b, c_1) et (x_2, b, c_2), alors $ConcatFaible(a, c, x_1, c_1, x_2, c_2)$ est satisfaite par définition.

($\Leftarrow$) Soient x_1 et x_2 tels que

$$Equal(a_1, b_1, c_1, x_1, b, c_1),$$

$$Equal(a_2, b_2, c_2, x_2, b, c_2),$$

$$ConcatFaible(a, b, x_1, c_1, x_2, c_2)$$

soient satisfaites. Par définition de *Equal*, (a_1, b_1, c_1) et (x_1, b, c_1) sont des codes positionnels d'un même uplet $(a_{1,1}, \ldots, a_{1,c_1})$, et (a_2, b_2, c_2) et x_2, b, c_2 sont des codes positionnels d'un même uplet $(a_{2,1}, \ldots, a_{2,c_2})$. Par définition de $ConcatFaible$, (a, b, c) est un code positionnel de l'uplet issu de la concaténation des uplets codés par (x_1, b, c_1) et (x_2, b, c_2), donc codés par (a_1, b_1, c_1) et (a_2, b_2, c_2).

Par définition, la relation $Concat(a, b, c, a_1, b_1, c_1, a_2, b_2, c_2)$ est satisfaite. □

Pour simplifier les notations, on utilisera l'opérateur + pour désigner la concaténation d'uplets codés. Ainsi, dans l'équation $(a, b, c_1 + c_2) = (a_1, b, c_1) + (a_2, b, c_2)$, $(a, b, c_1 + c_2)$ est le code positionnel de l'uplet formé par concaténation des uplets codés par (a_1, b, c_1) et (a_2, b, c_2).

Le code positionnel que nous avons présenté dans ce chapitre est un élément central dans les démonstrations qui vont suivre. Il permet d'encoder un grand nombre d'objets mathématiques dans un triplet d'entiers naturels. Son utilisation peut être relativement triviale comme nous le verrons au chapitre 5, où il sert à encoder de simples uplets dans un format plus concis, ou plus complexe comme nous le verrons au chapitre 4, où il servira à encoder des polynômes. Dans tous les cas, la puissance de ce codage réside dans la possibilité d'agir sur le contenu de l'uplet encodé sans avoir à décoder au préalable le triplet qui constitue le code positionnel.

CHAPITRE IV

ÉQUATIONS DIOPHANTIENNES UNIVERSELLES ET ENSEMBLES NON CO-DIOPHANTIENS

Nous allons maintenant étudier l'existence d'équations diophantiennes spéciales, les équations diophantiennes universelles, capables de simuler le comportement d'autres équations diophantiennes. De l'existence de telles équations, Matiiassevitch (1995) déduit l'existence d'ensembles diophantiens dont le complémentaire n'est pas diophantien (ensembles non co-diophantiens). Il est intéressant de noter qu'historiquement, l'existence d'ensembles non co-diophantiens a été établie avant celle d'équations diophantiennes universelles. En effet, Davis (1953) prouve l'existence d'ensembles non co-diophantiens en se basant sur l'existence de problèmes indécidables. Il faut toutefois attendre 1970 et la résolution du dixième problème par Matiiassevitch pour que devienne évidente l'existence d'équations diophantiennes universelles. En établissant que les ensembles semi-décidables sont diophantiens, l'existence de machines de Turing universelles, capables de simuler toute machine de Turing, entraîne l'existence d'équations diophantiennes capables de « simuler » toute équation diophantienne. Matiiassevitch (1995) bouscule quelque peu la chronologie des découvertes pour offrir au lecteur une preuve plus simple à comprendre.

4.1 Définition

Définition 4.1.1. *Soit une équation diophantienne de la forme :*

$$U(a_1, \ldots, a_n, k_1, \ldots, k_\ell, y_1, \ldots, y_w) = 0.$$

On appelle $a_1, \ldots, a_n$ *des* paramètres d'individu. *On appelle* $k_1, \ldots, k_\ell$ *des* paramètres de code. *Cette équation est dite universelle si et seulement si quelle que soit l'équation diophantienne*

$$D(a_1, \ldots, a_n, x_1, \ldots, x_m) = 0$$

il existe $k_1^D, \ldots, k_\ell^D$ *tels que quels que soient* $a_1, \ldots, a_n$, *il existe* $y_1, \ldots, y_w$ *tels que* $U(a_1, \ldots, a_n, k_1^D, \ldots, k_\ell^D, y_1, \ldots, y_w) = 0$ *si et seulement si il existe* $x_1, \ldots, x_m$ *tels que* $D(a_1, \ldots, a_n, x_1, \ldots, x_m) = 0$.

En d'autres termes, une équation diophantienne universelle à n paramètres d'individu est capable de représenter le même ensemble diophantien qu'une quelconque équation diophantienne à n paramètres. Il suffit pour cela de trouver les paramètres de code appropriés.

4.2 Propriétés des équations diophantiennes universelles

Jusqu'ici nous avons défini ce qu'est une équation universelle diophantienne sans prouver qu'une telle équation existe. Nous verrons qu'en fait, il est suffisant de prouver qu'il existe une équation universelle à un paramètre d'individu pour garantir que pour tout nombre de paramètres d'individu, il existe une équation diophantienne universelle.

Pour commencer, constatons que toute équation universelle à ℓ paramètres de code permet de créer une équation universelle à un seul paramètre de code moyennant une augmentation du nombre de variables.

Lemme 4.2.1. *Soit* $U(a_1, \ldots, a_n, k_1, \ldots, k_\ell, y_1, \ldots, y_w) = 0$ *une équation diophantienne universelle, où* $a_1, \ldots, a_n$ *sont des paramètres d'individu,* $k_1, \ldots, k_\ell$ *sont des paramètres de code et* $y_1, \ldots, y_w$ *sont des inconnues. Alors, il existe une équation diophantienne universelle* $U'(a_1, \ldots, a_n, k, k_1, \ldots, k_\ell, y_1, \ldots, y_w) = 0$ *où* $a_1, \ldots, a_n$ *sont des paramètres d'individu,* k *est un paramètre de code et* $k_1, \ldots, k_\ell, y_1, \ldots, y_w$ *sont des inconnues.*

Démonstration. Définissons U'. Soit :

$$\begin{aligned} U'(a_1,\ldots,a_n,k,k_1,\ldots,k_\ell,y_1,\ldots,y_w) &= U^2(a_1,\ldots,a_n,k_1,\ldots,k_\ell,y_1,\ldots,y_w) \\ &\quad +(k-2^{2^\ell}Cantor_\ell(k_1,\ldots,k_\ell))^2. \end{aligned}$$

Soit $D(a_1,\ldots,a_n,x_1,\ldots,x_m)=0$ une équation diophantienne quelconque.
Puisque U est universelle, alors, par définition, il existe $k_1^D,\ldots,k_\ell^D$ tels que :

$$\forall a_1,\ldots,a_n[\exists y_1,\ldots,y_w[U(a_1,\ldots,a_n,k_1^D,\ldots,k_\ell^D,y_1,\ldots,y_w)=0].$$

$$\Leftrightarrow$$

$$\exists x_1,\ldots,x_m[D(a_1,\ldots,a_n,x_1,\ldots,x_m)=0]].$$

Soit $k^D=2^{2^\ell}Cantor(k_1^D,\ldots,k_\ell^D)$. Soient $a_1,\ldots,a_n$ quelconques. On souhaite démontrer qu'il existe des entiers naturels $k_1,\ldots,k_\ell,y_1,\ldots,y_w$ tels que

$$U'(a_1,\ldots,a_n,k,k_1,\ldots,k_\ell,y_1,\ldots,y_w)=0$$

si et seulement s'il existe des entiers naturels $x_1,\ldots,x_m$ tels que

$$D(a_1,\ldots,a_n,x_1,\ldots,x_m)=0.$$

($\Rightarrow$) S'il existe $k_1,\ldots,k_\ell,y_1,\ldots,y_w$ tels que

$$U'(a_1,\ldots,a_n,k^D,k_1,\ldots,k_\ell,y_1,\ldots,y_w)=0,$$

alors nécessairement, $k_1,\ldots,k_\ell$ sont tels que $k^D=2^{2^\ell}Cantor(k_1,\ldots,k_\ell)$. Or puisque la fonction $Cantor$ est bijective, on a : $k_1=k_1^D,\ldots,k_\ell=k_\ell^D$. D'autre part, on a forcément

$$U(a_1,\ldots,a_n,k_1,\ldots,k_\ell,y_1,\ldots,y_w)=U(a_1,\ldots,a_n,k_1^D,\ldots,k_\ell^D,y_1,\ldots,y_w)=0$$

Or, par définition de $k_1^D,\ldots,k_\ell^D$, s'il existe $y_1,\ldots,y_w$ tels que

$$U(a_1,\ldots,a_n,k_1^D,\ldots,k_\ell^D,y_1,\ldots,y_w)=0,$$

alors il existe $x_1, \ldots, x_m$ tels que

$$D(a_1, \ldots, a_n, x_1, \ldots, x_m) = 0.$$

Pour résumer, s'il existe $k_1, \ldots, k_\ell, y_1, \ldots, y_w$ tels que

$$U'(a_1, \ldots, a_n, k, k_1, \ldots, k_\ell, y_1, \ldots, y_w) = 0,$$

alors il existe $x_1, \ldots, x_m$ tels que

$$D(a_1, \ldots, a_n, x_1, \ldots, x_m) = 0.$$

Prouvons maintenant l'implication réciproque.

($\Leftarrow$) S'il existe $x_1, \ldots, x_m$ tels que

$$D(a_1, \ldots, a_n, x_1, \ldots, x_m) = 0,$$

alors, par définition de $k_1^D, \ldots, k_\ell^D$, il existe $y_1, \ldots, y_w$ tels que

$$U(a_1, \ldots, a_n, k_1^D, \ldots, k_\ell^D, y_1, \ldots, y_w) = 0.$$

Or, puisque $k^D = 2^{2^\ell} Cantor(k_1^D, \ldots, k_\ell^D)$, par définition de U', on a :

$$U'(a_1, \ldots, a_n, k^D, k_1^D, \ldots, k_\ell^D, y_1, \ldots, y_w) = 0$$

□

Nous allons maintenant montrer qu'à partir d'une équation diophantienne universelle donnée à un paramètre d'individu, un paramètre de code et M inconnues, il est possible de créer des équations diophantiennes universelles à un nombre quelconque de paramètres d'individu, un paramètre de code et M inconnues. En d'autres termes, le nombre d'inconnues n'augmente pas lorsque le nombre de paramètres d'individu augmente.

Lemme 4.2.2. *S'il existe une équation diophantienne universelle à un paramètre d'individu et un paramètre de code, alors il existe une constante M telle que pour tout n, il existe une équation diophantienne universelle à n paramètres d'individu, un paramètre de code et M inconnues.*

Démonstration. Soit $U_1(a, k, y_1, \ldots, y_w)$ une équation diophantienne universelle quelconque à un paramètre d'individu et un paramètre de code. Soit $M = w$. Définissons maintenant U_n de la manière suivante :

$$U_n(a_1, \ldots, a_n, k, y_1, \ldots, y_M) = U_1(2^{2^n} Cantor(a_1, \ldots, a_n), k, y_1, \ldots, y_M).$$

Nous allons montrer que $U_n(a_1, \ldots, a_n, k, y_1, \ldots, y_M) = 0$ est une équation diophantienne universelle à n paramètres d'individus, un paramètre de code et M inconnues. Soit

$$D(a_1, \ldots, a_n, x_1, \ldots, x_m) = 0$$

une équation diophantienne quelconque. Soit

$$D'(a, z_1, \ldots, z_n, x_1, \ldots, x_m) = D^2(z_1, \ldots, z_n, x_1, \ldots, x_m) + (a - 2^{2^n} Cantor(z_1, \ldots, z_n))^2.$$

Il s'agit là d'une équation diophantienne à un paramètre d'individu.

Puisque U_1 est universelle, il existe k^D tel que, quel que soit a

$$\exists y_1, \ldots, \exists y_M [U_1(a, k^D, y_1, \ldots, y_M) = 0]$$

$$\Leftrightarrow$$

$$\exists z_1, \ldots, \exists z_n, \exists x_1, \ldots, \exists x_m [D'(a, z_1, \ldots, z_n, x_1, \ldots, x_m) = 0].$$

Donc, quels que soient les entiers naturels $a_1, \ldots, a_n$

$$\exists y_1, \ldots, \exists y_M [U_1(2^{2^n} Cantor(a_1, \ldots, a_n), k^D, y_1, \ldots, y_M) = 0]$$

$$\Leftrightarrow$$

$$\exists z_1, \ldots, \exists z_n, \exists x_1, \ldots, \exists x_m [D'(2^{2^n} Cantor(a_1, \ldots, a_n), z_1, \ldots, z_n, x_1, \ldots, x_m) = 0].$$

S'il existe $z_1, \ldots, z_n, x_1, \ldots, x_m$ tels que
$D'(2^{2^n} Cantor(a_1, \ldots, a_n), z_1, \ldots, z_n, x_1, \ldots, x_m)$
$= D^2(z_1, \ldots, z_n, x_1, \ldots, x_m) + (2^{2^n} Cantor(a_1, \ldots, a_n) - 2^{2^n} Cantor(z_1, \ldots, z_n))^2 = 0,$
alors, on a nécessairement

$$D(z_1, \ldots, z_n, x_1, \ldots, x_m) = 0 \text{ et}$$

$$2^{2^n} Cantor(a_1, \ldots, a_n) = 2^{2^n} Cantor(z_1, \ldots, z_n).$$

Puisque *Cantor* est une bijection, on a forcément $z_1 = a_1, \ldots, z_n = a_n$. De plus, on sait qu'il existe des entiers naturels $x_1, \ldots, x_m$ tels que

$$D(a_1, \ldots, a_n, x_1, \ldots, x_m) = 0.$$

Donc, s'il existe des entiers naturels $y_1, \ldots, y_m$ tels que

$$U1(2^{2^n} Cantor(a_1, \ldots, a_n), k^D, y_1, \ldots, y_M) = 0,$$

alors il existe $x_1, \ldots, x_m$ tels que

$$D(a_1, \ldots, a_n, x_1, \ldots, x_m) = 0.$$

On a donc prouvé que pour toute équation diophantienne

$$D(a_1, \ldots, a_n, x_1, \ldots, x_m) = 0$$

il existe k tel que quels que soient les entiers naturels $a_1, \ldots, a_n$

$$\exists y_1, \ldots, \exists y_M [U_1(2^{2^n} Cantor(a_1, \ldots, a_n), k, y_1, \ldots, y_M) = 0]$$

$$\Leftrightarrow$$

$$\exists x_1, \ldots, \exists x_m [D(a_1, \ldots, a_n, x_1, \ldots, x_m) = 0].$$

Or, par définition,

$$U_1(2^{2^n} Cantor(a_1, \ldots, a_n), k, y_1, \ldots, y_M) = U_n(a_1, \ldots, a_n, k, y_1, \ldots, y_M).$$

Ceci nous permet de conclure que

$$U_n(a_1, \ldots, a_n, k, y_1, \ldots, y_M) = 0$$

est bien une équation diophantienne universelle. □

Nous avons donc montré comment, à partir d'une équation diophantienne universelle à un paramètre d'individu, un paramètre de code et M inconnues, on peut

construire une équation diophantienne universelle à n paramètres d'individu, un paramètre de code et M inconnues.

Non seulement nous savons maintenant que le nombre d'inconnues n'augmente pas nécessairement avec le nombre de paramètres d'individu, mais en plus, pour prouver l'existence d'équations diophantiennes universelles à n paramètres d'individu pour tout n, il ne reste plus qu'à montrer l'existence d'une équation diophantienne universelle à un paramètre d'individu. Nous allons nous y attacher dans les prochaines sections.

4.3 Codes d'équations

Nous allons définir dans cette section une manière de coder toutes les informations concernant une équation diophantienne dans un sextuplet (b, c_L, c_R, d, e, f). Pour commencer, nous aurons besoin du lemme suivant.

Lemme 4.3.1 $(*)$. *Soit $a_1, \ldots, a_m$ des entiers naturels quelconques. Soit $d = a_1 + \cdots + a_m$. On a alors $d! \geq a_1! \ldots a_m!$.*

Démonstration $(*)$. Pour prouver que $d! \geq a_1! \ldots a_m!$ lorsque $d = a_1 + \cdots + a_n$, il suffit de prouver que $\frac{d!}{a_1! \ldots a_m!} \geq 1$. Or, $\frac{d!}{a_1! \ldots a_m!}$ est alors le nombre de manières de partitionner d objets distincts en parties distinctes de tailles $a_1, \ldots, a_n$ (Goldstein, Schneider et Siegel, 2007, chapitre 5). Ce nombre est évidemment un entier et il existe au moins un tel arrangement. □

Soit $D(x_1, \ldots, x_m) = 0$ une équation diophantienne quelconque. Pour le type de codage qui va suivre, nous aurons besoin de polynômes à coefficients positifs ou nuls. Il faut donc éliminer les coefficients négatifs. Ceci se fait très aisément en transformant l'équation $D(x_1, \ldots, x_m) = 0$ en l'équation $C_L(x_1, \ldots, x_m) = C_R(x_1, \ldots, x_m)$ avec $C_L(x_1, \ldots, x_m) - C_R(x_1, \ldots, x_m) = D(x_1, \ldots, x_m)$. Bien entendu, il faut choisir C_L et C_R tels que leurs coefficients soient positifs ou nuls.

Soit d un nombre supérieur au degré total de D. On a donc :

$$C_L(x_1,\ldots,x_m) = \sum_{i_0+\cdots+i_m<d} c_{L,i_0,\ldots,i_m} x_0^{i_0}\ldots x_m^{i_m} \text{ et}$$
$$C_R(x_1,\ldots,x_m) = \sum_{i_0+\cdots+i_m<d} c_{R,i_0,\ldots,i_m} x_0^{i_0}\ldots x_m^{i_m}.$$

Définition 4.3.1. *Soit le polynôme*

$$C(x_0,\ldots,x_m) = \sum_{i_0+\cdots+i_m<d} c_{i_0,\ldots,i_m} x_0^{i_0}\ldots x_m^{i_m},$$

où $c_{i_0,\ldots,i_m} \in \mathbb{N}$. *On appelle* numéro du polynôme C de degré total d relativement à la base b *le nombre*

$$c = \sum_{i_0,\ldots,i_m} i_0!\ldots i_m!(d-1-i_0-\cdots-i_m)!c_{i_0,\ldots,i_m} b^{d^{m+1}-i_0d^0-\cdots-i_md^m}.$$

Observons quelques particularités de ce nombre. Tout d'abord, le lemme 4.3.1 nous permet de constater que si b est strictement supérieur à $d!\max\{c_{i_0,\ldots,i_m}\}$, alors b est strictement supérieur à $i_0!\ldots i_m!(d-1-i_0-\cdots-i_m)!c_{i_0,\ldots,i_m}$ quels que soient $i_0,\ldots,i_m$, puisque

$$i_0+\cdots+i_m+(d-1-i_0-\cdots-i_m) = d-1.$$

De plus, à tout coefficient $c_{i_o,\ldots,i_m}$ est associé un terme $b^{d^{m+1}-i_0d^0-\cdots-i_md^m}$ manifestement unique, puisque quel que soit $j \in \{0,\ldots,m\}$, $i_j < d$. Par conséquent, si $b > d!\max\{c_{i_0,\ldots,i_m}\}$, alors pour un numéro donné du polynôme C relativement à la base b, les coefficients $c_{i_0,\ldots,i_m}$ sont déterminés de façon unique par b, c et d.

Définition 4.3.2. *On appellera* code du polynôme

$$C(x_0,\ldots,x_m) = \sum_{i_0+\cdots+i_m<d} c_{i_0,\ldots,i_m} x_0^{i_0}\ldots x_m^{i_m}$$

le quintuplet (b,c,d,e,f) *si :*

- $e = m$,
- d *est strictement supérieur au degré total de* $C(x_0,\ldots,x_m)$,
- $b > d!\ max\{c_{i_0,\ldots,i_m}\}$,

- $f = 2^{d^1} + \cdots + 2^{d^e}$.

Définition 4.3.3. *On appellera le sextuplet* (b, c_L, c_R, d, e, f) code étendu de l'équation $D(x_0, \ldots, x_m) = 0$ si (b, c_L, d, e, f) et (b, c_R, d, e, f) sont des codes des polynômes $C_L(x_0, \ldots, x_m)$ et $C_R(x_0, \ldots, x_m)$.

Définition 4.3.4. *Pour tous entiers naturels d, e et f, on définit la relation Format comme suit :*

$$Format(d, e, f)$$

$$\Leftrightarrow$$

$$f = 2^{d^1} + \cdots + 2^{d^e} \wedge d > Degre(C(x_0, \ldots, x_m)) \wedge e = m.$$

Notons que nous ne pouvons pas conclure que *Format* est une relation diophantienne dans la mesure où nous ne savons pas si *Degre* est diophantienne.

4.4 Codes de solutions

On s'intéresse maintenant à une manière de coder des uplets potentiellement solutions d'équations. Ceci sera utilisé lorsqu'on voudra tester une solution potentielle d'un polynôme donné encodé selon la méthode décrite à la section précédente.

Définition 4.4.1. *Soit* $D(a, x_1, \ldots, x_e) = 0$ *une équation diophantienne quelconque. Soit* (d, e, f) *un format de cette équation. Soit* $(x_1, \ldots, x_e)$ *une solution potentielle de l'équation. Soit g tel que quel que soit* $i \in \{1, \ldots, e\}$, $g > x_i$. *Soit h tel que* $h = x_1 g^{d^1} + \cdots + x_e g^{d^e}$. *On appelle* (d, e, f, g, h) code d'une solution potentielle *de* $D(a, x_1, \ldots, x_e) = 0$ *et on dit qu'il satisfait la relation* $SCode(d, e, f, g, h)$.

Cette définition implique que $(h, g, d^e + 1)$ est un code positionnel.

Lemme 4.4.1. *Soit* (d, e, f, g, h) *un code de la solution potentielle* $(x_1, \ldots, x_e)$. *Soit* $(h_1, \ldots, h_{d^e+1})$ *l'uplet codé par* (h, g, d^e+1).*Soit* $(f_1, \ldots, f_{d^e+1})$ *l'uplet codé par* $(f, 2, d^e + 1)$. *Alors pour tout* $i \in \{1, \ldots, d^{e+1}\}$, $f_i = 1$ *si et seulement si il existe j tel que* $h_i = x_j$.

Démonstration $(*)$. Par définition, on a :

$$f = 2^{d^1} + \cdots + 2^{d^e} \text{ et}$$

$$h = x_1 g^{d^1} + \cdots + x_e g^{d^e}$$

Donc, pour tout $i \in \{1, \ldots, d^e + 1\}$, s'il existe j tel que $i = d^j$ alors $f_i = 1$ et $h_i = x_j$ et sinon $f_i = h_i = 0$. □

Ce lemme signifie que les 1 contenus dans l'uplet $(f_1, \ldots, f_{d^e+1})$ et les $x_1, \ldots, x_c$ de l'uplet $(h_1, \ldots, h_{d^e+1})$ occupent les mêmes positions dans leurs uplets respectifs.

Lemme 4.4.2 $(*)$. *Soit (a, b, c) un code positionnel de l'uplet $(a_1, \ldots, a_c)$. Soit n tel que pour tout i compris entre 1 et c, $na_i < b$. Alors, (na, b, c) est un code positionnel de l'uplet $(na_1, \ldots, na_c)$.*

Démonstration $(*)$. Par définition d'un code positionnel, on a :

$$a = \sum_{i=1}^{c} x_i b^{i-1},$$

d'où

$$na = \sum_{i=1}^{c} n x_i b^{i-1}.$$

On sait que $nx_i < b$ pour tout i compris entre 1 et c. Par définition, (na, b, c) est un code positionnel de l'uplet $(na_1, \ldots, na_c)$. □

Nous allons maintenant définir une nouvelle relation, $SCod$, exprimable en termes diophantiens. Nous montrerons par la suite que cette relation est équivalente à la relation $SCode$ sous certaines conditions. Ceci nous évitera d'avoir à prouver que $SCode$ elle-même est diophantienne.

Définition 4.4.2. *Pour tous entiers naturels d, e, f, g et h, on définit la relation $SCod$ comme suit :*

$$SCod(d, e, f, g, h)$$

$$\Leftrightarrow$$

$$\exists t[Equal(f, 2, d^e + 1, t, g, d^e + 1) \wedge NotGreater(h, g, (g-1)t, g)].$$

Lemme 4.4.3. *Pour tous entiers naturels d, e, f, g, h, on a*

$$Format(d, e, f) \Rightarrow [SCod(d, e, f, g, h) \Leftrightarrow SCode(d, e, f, g, h)].$$

Démonstration. Soit d, e, f tels que $Format(d, e, f)$ est satisfaite.
($\Rightarrow$) Si on a $SCod(d, e, f, g, h)$, alors, par définition de $SCod$, on a

$$Equal(f, 2, d^e + 1, t, g, d^e + 1),$$

donc $(t, g, d^e + 1)$ et $(f, 2, d^e + 1)$ représentent le même uplet. Donc, $t = g^{d^1} + \cdots + g^{d^e}$. Par conséquent, $(g-1)t = (g-1)g^{d^1} + \cdots + (g-1)g^{d^e}$. Donc, $((g-1)t, g, d^e + 1)$ est un code positionnel. Soit $(x_1, \ldots, x_{d^e+1})$ l'uplet encodé. Soit $(h_1, \ldots, h_{d^e+1})$ l'uplet codé par $(h, g, d^e + 1)$. La relation $NotGreater(h, g, (g-1)t, g)$ est satisfaite, donc, pour tout $i \geq 0$, $Elem(h, g, i) \leq Elem((g-1)t, g, i)$. Or, puisque pour tout $i \in \{1, \ldots, d^e + 1\}$ on a

- s'il existe j tel que $d^j = i$, alors $h_i < g$
- sinon $h_i \leq 0$, d'où $h_i = 0$.

Donc, on peut écrire

$$h = h_{d^1} g^{d^1} + \cdots + h_{d^e} g^{d^e}.$$

Par conséquent, on a bien $SCode(d, e, f, g, h)$ pour une solution potentielle $(h_{d^1}, \ldots, h_{d^e})$.
Prouvons maintenant l'implication réciproque.
($\Leftarrow$) Si (d, e, f, g, h) code la solution potentielle $(x_1, \ldots, x_e)$, alors on a

$$h = x_1 g^{d^1} + \cdots + x_m g^{d^m}$$

avec pour tout $i \in \{1, \ldots, m\}$, $x_i < g$. Soit $t = g^{d^1} + \cdots + g^{d^e}$. L'uplet $(t, g, d^e + 1)$ est le code d'un uplet ne contenant que des 0 et des 1. D'après le lemme 4.4.2,

$$((g-1)t, g, d^e + 1)$$

est le code positionnel d'un uplet ne contenant que des 0 et des $(g-1)$. De plus, les éléments non nuls de cet uplet sont situés aux mêmes emplacements que les éléments non

nuls de l'uplet codé par $(h, g, 2^e + 1)$. Par conséquent, on est assuré d'avoir la relation $NotGreater(h, g, (g-1)t, g)$. De plus, on a $f = 2^{d^1} + \cdots + 2^{d^e}$ par définition de f. Alors, puisque $t = g^{d^1} + \ldots g^{d^e}$, on sait que la relation $Equal(f, 2, d^e + 1, t, g, d^e + 1)$ est satisfaite puisque $(f, 2, d^e + 1)$ et $(t, g, d^e + 1)$ sont des codes positionnels du même uplet. Donc, il existe t tel que $Equal(f, 2, d^e + 1, t, g, d^e + 1)$ et $NotGreater(h, g, (g-1)t, g)$ sont satisfaites, donc tel que $SCod(d, e, f, g, h)$ est satisfaite. □

Nous savons maintenant coder à la fois les polynômes mais également les solutions potentielles de ces polynômes. Il est alors naturel de chercher à développer une méthode permettant, à partir de leurs codes, de calculer la valeur prise par un polynôme pour une certaine affectation de valeurs à ses inconnues.

4.5 Calcul de la valeur d'un polynôme

Ce que nous cherchons à obtenir ici est une méthode permettant de calculer la valeur d'une solution particulière d'un polynôme particulier à partir de leurs codes respectifs. On pourrait envisager de décoder le polynôme et la solution à tester et de faire effectivement le calcul. Toutefois il existe une solution plus simple. Nous allons présenter une méthode plus efficace qui permet de calculer la valeur du polynôme directement à partir des codes, sans passer par une étape de décodage.

Lemme 4.5.1. *Soit (b, c, d, e, f) le code d'un polynôme $D(x_0, \ldots, x_m)$.*
Soit (d, e, f, g, h) le code d'une solution potentielle $x = (x_1, \ldots, x_m)$ pour ce polynôme.
Si w et t sont tels que $Equal(h, g, d^e + 1, t, w, d^e + 1)$ est satisfaite, alors (d, e, f, w, t) est un code de la solution potentielle x.

Démonstration. Par hypothèse, (d, e, f, g, h) est le code d'une solution. Par définition, (d, e, f) est donc un format du polynôme D et $h = x_1 g^{d^1} + \cdots + x_e g^{d^e}$ Par définition de $Equal$, si t et w sont tels que $Equal(h, g, d^e + 1, t, w, d^e + 1)$ est satisfaite, alors $t = x_1 w^{d^1} + \cdots + x_e w^{d^e}$ et $w > x_i$ pour i $in\{1, \ldots, m\}$. Le quintuplet (d, e, f, w, t) satisfait bien la définition de code d'une solution. □

Lemme 4.5.2 (∗). *Soit (b, c, d, e, f) le code d'un polynôme $D(x_0, \ldots, x_m)$. Si w et s sont tels que $Equal(c, b, d^{e+1}+1, s, w, d^{e+1}+1)$ est satisfaite, alors (w, s, d, e, f) est un code du polynôme D.*

Démonstration (∗). Par hypothèse, (b, c, d, e, f) est le code d'un polynôme. Par définition, (d, e, f) est donc un format du polynôme D et

$$c = \sum_{i_0+\cdots+i_m<d} i_0! \ldots i_m!(d-1-i_0-\cdots-i_m)!c_{i_0,\ldots,i_m} b^{d^{m+1}-i_0-\cdots-i_m d^m}$$

Par définition de *Equal*, si s et w sont tels que $Equal(c, b, d^{e+1}+1, s, w, d^{e+1}+1)$, alors on a

$$s = \sum_{i_0+\cdots+i_m<d} i_0! \ldots i_m!(d-1-i_0-\cdots-i_m)!c_{i_0,\ldots,i_m} w^{d^{m+1}-i_0-\cdots-i_m d^m}$$

et $w > i_0! \ldots i_m!(d-1-i_0-\cdots-i_m)!c_{i_0,\ldots,i_m}$ quels que soient $i_0, \ldots, i_m$.
Le quintuplet (w, s, d, e, f) satisfait bien la définition de code du polynôme D. □

Lemme 4.5.3. *Soit (b, c, d, e, f) un code du polynôme P à un paramètre et e variables. Soit (d, e, f, g, h) un code d'une solution $S = (x_1, \ldots, x_e)$ du polynôme P. Soient w, s et t tels que $Equal(h, g, d^e+1, t, w, d^e+1)$ et $Equal(c, b, d^{e+1}, s, w, d^{e+1})$.*
La valeur de P pour une valeur a du paramètre et la solution S est donnée par

$$P(a, x_1, \ldots, x_e) = \frac{Elem((1+aw+t)^{d-1}s, w, d^{e+1}+1)}{(d-1)!}$$

pour $w > (1+a+h)^{d-1}c$.

Démonstration. Si w, s et t sont tels que $Equal(h, g, d^e+1, t, w, d^e+1)$ et $Equal(c, b, d^{e+1}, s, w, d^{e+1})$ sont satisfaites, alors, d'après les lemmes 4.5.2 et 4.5.1, (w, s, d, e, f) est un code de P et (d, e, f, w, t) est un code de S.
Par définition, on a

$$(1+aw+t)^{d-1} = (1+aw^{d^0}+x_1 w^{d^1}+\cdots+x_e w^{d^e})^{d-1}.$$

L'utilisation de la formule du binôme de Newton nous permet de déduire

$$(1+aw+t)^{d-1} = \sum_{i=0}^{d-1} \frac{(d-1)!}{i!(d-1-i)!}(aw^{d^0} + x_1 w^{d^1} + \cdots + x_e w^{d^e})^i.$$

En utilisant maintenant le multinôme de Newton, on obtient

$$\begin{aligned}
(1+aw+t)^{d-1} &= \sum_{i=0}^{d-1} \frac{(d-1)!}{i!(d-1-i)!} \sum_{i_0+\cdots+i_e=i} \frac{i!}{i_0!\ldots i_e!}(aw)^{i_0}.(x_1 w^{d_1})^{i_1}\ldots(x_e w^{d_e})^{i_e} \\
&= \sum_{i=0}^{d-1} \frac{(d-1)!}{i!(d-1-i)!} \sum_{i_0+\cdots+i_e=i} \frac{i!}{i_0!\ldots i_e!} a^{i_0}.x_1^{i_1}\ldots x_e^{i_e} w^{i_0 d^0+\cdots+i_e d^e} \\
&= \sum_{i=0}^{d-1} \sum_{i_0+\cdots+i_e=i} \frac{(d-1)!i!}{i!(d-1-i)!i_0!\ldots i_e!} a^{i_0}.x_1^{i_1}\ldots x_e^{i_e} w^{i_0 d^0+\cdots+i_e d^e} \\
&= \sum_{i=0}^{d-1} \sum_{i_0+\cdots+i_e=i} \frac{(d-1)!}{(d-1-i)!i_0!\ldots i_e!} a^{i_0}.x_1^{i_1}\ldots x_e^{i_e} w^{i_0 d^0+\cdots+i_e d^e} \\
&= \sum_{i_0+\cdots+i_e<d} \frac{(d-1)!}{(d-1-i)!i_0!\ldots i_e!} a^{i_0}.x_1^{i_1}\ldots x_e^{i_e} w^{i_0 d^0+\cdots+i_e d^e}.
\end{aligned}$$

Soit

$$s = \sum_{i_0+\cdots+i_e<d} i_0!\ldots i_e!(d-1-i_0-\cdots-i_e)!c_{i_0,\ldots,i_e} w^{d^{m+1}-i_0 d^0-\cdots-i_e d^e}.$$

Si on a $i_0,\ldots,i_e < d$, alors d'après le lemme 4.3.1, on a

$$i_0 d^0 + \cdots + i_e d^e < d^{e+1},$$

d'où

$$i_0 d^0 + \cdots + i_e d^e \leq d^{e+1} - 1.$$

On obtient alors la forme suivante

$$\begin{aligned}(1+aw+t)^{d-1}s &= \Big(\sum_{i_0+\cdots+i_e<d} \frac{(d-1)!}{(d-1-i_0-\cdots-i_e)!i_0!\ldots i_e!}a^{i_0}x^{i_1}\ldots x^{i_e}w^{i_0d^0+\ldots i_ed^e}\Big)\\ &\quad .\Big(\sum_{i_0+\cdots+i_e<d} i_0!\ldots i_e!(d-1-i_0-\cdots-i_e)!c_{i_0,\ldots,i_e}w^{d^{m+1}-i_0d^0-\cdots-i_ed^e}\Big)\\ &= \sum_{i_0+\cdots+i_e<d}\Big(\\ &\quad \sum_{j_0+\cdots+j_e<d}\Big(\frac{(d-1)!}{(d-1-i_0-\cdots-i_e)!i_0!\ldots i_e!}a^{i_0}x_1^{i_1}\ldots x_e^{i_e}j_0!\ldots j_e\\ &\quad (d-1-j_0-\cdots-j_e)!c_{j_0,\ldots,j_e}w^{i_1d^1+\cdots+i_ed^e}w^{d^{m+1}-j_0d^0-\cdots-j_ed^e}\Big)\Big)\\ &= \sum_{i=0}^{2d^{m+1}-1}C_kw^k,\end{aligned}$$

avec

$$\begin{aligned}C_k &= \sum_{d^{e+1}+(i_0-j_0)d^0-\cdots+(i_e-j_e)d^e=k}\Big(\frac{(d-1)!}{(d-1-i_0\cdots-i_e)!i_0!\ldots i_e!}a^{i_0}x^{i_1}\ldots x^{i_e}j_0!\ldots j_e!\\ &\quad (d-1-i_0-\cdots-i_e)c_{i_0,\ldots,i_e}\Big),\end{aligned}$$

avec $i_0+\cdots+i_e<d$ et $j_0+\cdots+j_e<d$. Par conséquent, pour $k=d^{e+1}$, on a

$$\begin{aligned}C_{d^{e+1}} &= \sum_{i_0+\cdots+i_e<d}\frac{(d-1)!}{i_0!\ldots i_e!(d-1-i_0-\cdots-i_e)!}i_0!\ldots i_e!(d-1-i_0-\cdots-i_e)!\\ &\quad .c_{i_0,\ldots,i_e}a^{i_0}x_1^{i_1}\ldots x_e^{i_e}\\ &= \sum_{i_0+\cdots+i_e<d}(d-1)!c_{i_0,\ldots,i_e}a^{i_0}x_1^{i_1}\ldots x_e^{i_e}\\ &= (d-1)!\sum_{i_0+\cdots+i_e<d}c_{i_o,\ldots,i_e}a^{i_0}x_1^{i_1}\ldots x_e^{i_e}.\end{aligned}$$

Par définition, on a

$$\sum_{i_0+\cdots+i_e<d}c_{i_0,\ldots,i_e}a^{i_0}x_1^{i_1}\ldots x_e^{i_e}=P(a,x_1,\ldots,x_e),$$

d'où

$$C_{d^{e+1}}=(d-1)!P(a,x_1,\ldots,x_e).$$

Donc, si on arrive à extraire $C_{d^{e+1}}$ à partir de $(1+aw+t)^{d-1}s$, on aura la valeur du polynôme P pour une constante a et des valeurs $x_1,\ldots,x_e$ des variables.

On sait que

$$(1+aw+t)^{d-1}s = \sum_{i=0}^{2d^{e+1}-1} C_k w^k;$$

donc, si on choisit w tel que pour tout $i \in \{0,\ldots,2d^{e+1}-1\}$, $w > C_i$, alors on aura par définition que $((1+aw+t)^{d-1}s, w, 2d^{e+1})$ est un code positionnel de l'uplet $(C_0, C_1, \ldots, C_{2d^{e+1}-1})$. Considérons le nombre $(1+a+h)^{d-1}c$. Ce nombre a une forme quasiment identique à celle de $(1+aw+t)^{d-1}s$, ce qui n'est pas vraiment étonnant puisque s et c sont les numéros de code d'un même uplet et que h et t sont les numéros de code d'un même uplet. On a

$$\begin{aligned}
(1+a+h)^{d-1}c &= (1+a+x_1g^1+\cdots+x_eg^e)c \\
&= \Big(\sum_{i_0+\cdots+i_e<d} \frac{(d-1)!}{(d-1-i_0-\cdots-i_e)!i_0!\ldots i_e!} a^{i_0}x_1^{i_1}\ldots x_e^{i_e} g^{i_1d^1+\cdots+i_ed^e}\Big) \\
&\cdot \Big(\sum_{i_0+\cdots+i_e<d} i_0!\ldots i_e(d-1-i_0-\cdots-i_e)!c_{i_0,\ldots,i_e} b^{d^{m+1}-i_0d^0-\cdots-i_ed^e}\Big) \\
&= \sum_{i_0+\cdots+i_e<d} \\
&\quad \sum_{j_0+\cdots+j_e<d} \Big(\frac{(d-1)!}{(d-1-i_0-\cdots-i_e)!i_0!\ldots i_e!} a^{i_0}x_1^{i_1}\ldots x_e^{i_e} j_0!\ldots j_e \\
&\quad (d-1-j_0-\cdots-j_e)!c_{j_0,\ldots,j_e} g^{i_1d^1+\cdots+i_ed^e} b^{d^{m+1}-j_0d^0-\cdots-j_ed^e}\Big).
\end{aligned}$$

Soit $w > (1+a+h)^{d-1}c$. On a naturellement

$$\forall i_0,\ldots,i_e,j_0,\ldots,j_e\, i_0+\cdots+i_e<d \wedge j_0+\cdots+j_e<d$$

$$\Rightarrow w > \frac{(d-1)!}{(d-1-i_0-\cdots-i_e)!i_0!\ldots i_e!} a^{i_0}x_1^{i_1}\ldots x_e^{i_e} j_0!\ldots j_e(d-1-j_0-\cdots-j_e)!c_{j_0,\ldots,j_e}].$$

Or, par définition, on a, pour tout k,

$$\begin{aligned}
C_k &= \sum_{i_0+\cdots+i_e<d} \\
&\quad \sum_{j_0+\cdots+j_e<d} \frac{(d-1)!}{(d-1-i_0\cdots-i_e)!i_0!\ldots i_e!} a^{i_0}x^{i_1}\ldots x^{i_e} j_0!\ldots j_e! \\
&\cdot \quad (d-1-i_0-\cdots-i_e)!c_{i_0,\ldots,i_e}
\end{aligned}$$

avec $k = d^{e-1} + (i_0-j_0)d^0 + \cdots + (i_e-j_e)d^e$. On a donc $\forall k[C_k < w]$. Par conséquent, en choisissant $w > (1+a+h)^{d-1}c$, $((1+aw+t)^{d-1}s, w, 2d^{e+1})$ est un code positionnel

de l'uplet $(C_0, C_1, \ldots, C_{2d^{e+1}-1})$.

Par définition de la fonction diophantienne *Elem*, on a

$$\begin{aligned} Elem(((1+aw+t)^{d-1}s, w, 2d^{e+1}), w, d^{e+1}+1) &= C_{d^{e+1}} \\ &= (d-1)!P(a, x_1, \ldots, x_e), \end{aligned}$$

d'où

$$P(a, x_1, \ldots, x_e) = \frac{Elem(((1+aw+t)^{d-1}s, w, 2d^{e+1}), w, d^{e+1}+1)}{(d-1)!}.$$

□

Grâce à ce lemme, nous sommes en mesure d'exprimer sous forme de fonction diophantienne la valeur d'un polynôme à un paramètre et un nombre quelconque de variables pour une certaine affectation de valeurs à ses variables et son paramètre. Il suffit pour ce faire d'utiliser des codages du polynôme et de la distribution appropriés. Cela signifie que nous pouvons désormais, à partir du code étendu d'une équation diophantienne et du code d'une solution potentielle de cette équation, vérifier si l'équation est satisfaite. De plus, la procédure de vérification sera diophantienne. En effet, le code étendu d'une équation diophantienne $C_L = C_R$ à un paramètre et e inconnues est un sextuplet (b, c_L, c_R, d, e, f) tel que (b, c_L, d, e, f) et (c, c_R, d, e, f) sont respectivement des codes des polynômes C_L et C_R. Si nous calculons les valeurs que prennent C_L et C_R pour un certain paramètre et une certaine distribution de valeurs des inconnues, il suffira alors de vérifier que l'égalité est satisfaite pour pouvoir conclure que l'équation diophantienne est satisfaite.

Nous possédons un outil nous permettant de définir des relations diophantiennes qui seront vérifiées si et seulement si une certaine équation diophantienne à un paramètre et n inconnues possède une solution. Il s'agit justement d'une des caractéristiques principales des équations diophantiennes universelles. En fait, la relation diophantienne *Solution*, définie ci-dessous, est représentée par une équation diophantienne qui est précisément universelle.

Définition 4.5.1. *Pour tous entiers a, b, c_L, c_R, d, e, f, g et h, on définit la relation Solution comme suit*

$$\begin{aligned} Solution(a,b,c_L,c_R,d,e,f,g,h) \quad \Leftrightarrow \quad & SCod(d,e,f,g,h) \\ & \wedge \exists s_L \exists s_R \exists t \exists w[w > (1+a+h)^{d-1}(c_L+c_R)] \\ & \wedge Equal(c_L,b,d^{e+1},s_L,w,d^{e+1}) \\ & \wedge Equal(c_R,b,d^{e+1},s_R,w,d^{e+1}) \\ & \wedge Equal(h,g,d^e+1,t,w,d^e+1) \\ & \wedge (Elem((1+aw+t)^{d+1}s_L,w,d^{e+1}) \\ & = Elem((1+aw+t)^{d+1}s_R,w,d^{e+1})). \end{aligned}$$

Lemme 4.5.4 $(*)$**.** *Soit (b,c_L,c_R,d,e,f) un code étendu d'une équation*

$$D(a,x_1,\ldots,x_m)=0$$

à un paramètre et m inconnues. Soient les polynômes à coefficients entiers C_L et C_R tels que

$$D(a,x_1,\ldots,x_m) = C_L(a,x_1,\ldots,x_m) - C_R(a,x_1,\ldots,x_m)$$

et dont les codes sont respectivement (b,c_L,d,e,f) et (b,c_R,d,e,f). Alors, pour tout paramètre a, on a

$$\exists g \exists h[Solution(a,b,c_L,c_R,d,e,f,g,h)]$$

$$\Leftrightarrow$$

$$\exists x_1 \ldots \exists x_m[D(a,x_1,\ldots,x_m)=0]$$

Démonstration $(*)$. Puisque (b,c_L,c_R,d,e,f) est un code d'équation, alors (d,e,f) est un format d'équation.

($\Rightarrow$) Soient g et h tels que $Solution(a,b,c_L,c_R,d,e,f,g,h)$ soit satisfaite.

Par définition, il existe donc s_L, s_R, t et w tels que les relations

$$SCod(d,e,f,g,h)$$

$$w > (1 + a + h)^{d-1}(c_L + c_R)$$

$$Equal(c_L, b, d^{e+1} + 1, s_L, w, d^{e+1} + 1)$$

$$Equal(c_R, b, d^{e+1} + 1, s_R, w, d^{e+1} + 1)$$

$$Equal(h, g, d^e + 1, t, w, d^e + 1)$$

$$Elem((1 + aw + t)^{d-1}s_L, w, d^{e+1} + 1) = Elem((1 + aw + t)^{d-1}s_R, w, d^{e+1} + 1)$$

soient satisfaites. D'après le lemme 4.4.3, puisque (d, e, f) est un format d'équation, alors

$$SCod(d, e, f, g, h) \Leftrightarrow SCode(d, e, f, g, h).$$

La relation $SCode(d, e, f, g, h)$ est satisfaite. Par définition, (d, e, f, g, h) est donc un code d'une solution potentielle $(x_1, \ldots, x_e)$. De plus, puisque $w > (1+a+h)^{d-1}(c_L + c_R)$, on a $w > (1+a+h)^{d-1}c_L$ et $w > (1+a+h)^{d-1}c_R$. Par conséquent, d'après le lemme 4.5.3, on a

$$Elem(1 + aw + t)^{d-1}s_L = (d-1)!C_L(a, x_1, \ldots, x_e) \text{ et}$$

$$Elem(1 + aw + t)^{d-1}s_R = (d-1)!C_R(a, x_1, \ldots, x_e),$$

d'où

$$\begin{aligned}
(d-1)!C_L(a, x_1, \ldots, x_e) &= (d-1)!C_R(a, x_1, \ldots, x_e) \\
C_L(a, x_1, \ldots, x_e) &= C_R(a, x_1, \ldots, x_e) \\
C_L(a, x_1, \ldots, x_e) - C_R(a, x_1, \ldots, x_e) &= 0 \\
D(a, x_1, \ldots, x_e) = 0.
\end{aligned}$$

($\Leftarrow$) Soient x_1, ..., x_e tels que

$$D(a, x_1, \ldots, x_e) = C_L(a, x_1, \ldots, x_e) - C_R(a, x_1, \ldots, x_e) = 0.$$

Soit g tel que pour tout i compris entre 1 et e, $g > x_i$ et $g > 1$. Soit $h = x_1 g^{d_1} + \cdots + x_e g^{d_e}$. Par définition, (d, e, f, g, h) est un code de la solution $(x_1, \ldots, x_e)$,

la relation $SCode(d, e, f, g, h)$ est vérifiée. Puisque (d, e, f) est un format d'équation, alors d'après le lemme 4.4.3, on a

$$SCode(d, e, f, g, h) \Leftrightarrow SCod(d, e, f, g, h).$$

Donc, la relation $SCod(d, e, f, g, h)$ est vérifiée. Soit w tel que $w > (a+1+h)^{d-1}(c_L+c_R)$ et $\forall i[1 \leq i \leq e \Rightarrow x_i < w]$. Puisque (b, c_L, d, e, f) est un code du polynôme C_L, alors $(c_L, b, d^{e+1}+1)$ est un code positionnel et

$$c_L = \sum_{i_0+\cdots+i_e<d} i_0!\ldots i_e!(d-1-i_0-\cdots-i_e)!c_{L,i_0,\ldots,i_e}b^{d^{e+1}-i_0d^{i_0}-\cdots-i_ed^{i_e}}.$$

Soit s_L défini par

$$s_L = \sum_{i_0+\cdots+i_e<d} i_0!\ldots i_e!(d-1-i_0-\cdots-i_e)!c_{L,i_0,\ldots,i_e}w^{d^{e+1}-i_0d^{i_0}-\cdots-i_ed^{i_e}}.$$

Puisque $w > c_L > i_0!\ldots i_e!(d-1-i_0-\cdots-i_e)!c_{L,i_0,\ldots,i_e}$ pour tout $i_0, \ldots, i_e$ tels que $i_0+\cdots+i_e < d$, alors par définition d'un code positionnel, les codes positionnels $(c_L, b, d^{e+1}+1)$ et $(s_L, w, d^{e+1}+1)$ codent pour le même uplet. Par définition de $Equal$,

$$Equal(c_L, b, d^{e+1}+1, s_L, w, d^{e+1}+1)$$

est satisfaite. En se basant sur les mêmes observations que précédemment, on conclut facilement que

$$Equal(c_R, b, d^{e+1}+1, s_R, w, d^{e+1}+1)$$

est satisfaite. Puisque (d, e, f, g, h) est un code de la solution $(x_1, \ldots, x_e)$, (h, g, d^e+1) un code positionnel et

$$h = x_1g^{d_1}+\cdots+x_eg^{d_e}$$

Soit $t = x_1w^{d^1}+\cdots+x_ew^{d^e}$. Puisque $w > x_i$ pour tout i compris entre 1 et e, alors (t, w, d^e+1) est un code positionnel. De plus, (h, g, d^e+1) et (t, w, d^e+1) codent pour le même uplet. Par définition de $Equal$, $Equal(h, g, d^e+1, t, w, d^e+1)$ est satisfaite. Puisque (b, c_L, d, e, f) est un code de polynôme, que (d, e, f, g, h) est un code de la solution $(x_1, \ldots, x_e)$ et que s_L, w et t sont tels que les relations

$$Equal(c_L, b, d^{e+1}+1, s_L, b, d^{d+1}+1)$$

$$Equal(h, g, d^e + 1, t, w, d^e + 1)$$

sont satisfaites, alors d'après le lemme 4.5.3,

$$(d-1)!C_L(a, x_1, \ldots, x_e) = Elem((1 + aw + t)^{d-1} s_L, w, d^{e+1} + 1).$$

En raisonnant de manière analogue, on conclut que

$$(d-1)!C_R(a, x_1, \ldots, x_e) = Elem((1 + aw + t)^{d-1} s_R, w, d^{e+1} + 1).$$

Or, on sait que $C_L(a, x_1, \ldots, x_e) = C_R(a, x_1, \ldots, x_e)$. Donc on a

$$\begin{aligned} C_L(a, x_1, \ldots, x_e) &= C_R(a, x_1, \ldots, x_e) \\ (d-1)!C_L(a, x_1, \ldots, x_e) &= (d-1)!C_R(a, x_1, \ldots, x_e) \\ Elem((1 + aw + t)^{d-1} s_L, w, d^{e+1} + 1) &= Elem((1 + aw + t)^{d-1} s_R, w, d^{e+1} + 1) \end{aligned}$$

Donc, s'il existe $x_1, \ldots, x_e$ tels que $D(a, x_1, \ldots, x_e) = 0$, alors il existe g et h tels que $SCod(d, e, f, g, h)$ est satisfaite et tels qu'il existe s_L, s_R, t et w tels que les relations

$$w > (1 + a + h)^{d-1}(c_L + c_R)$$

$$Equal(c_L, b, d^{e+1} + 1, s_L, w, d^{e+1} + 1)$$

$$Equal(c_R, b, d^{e+1} + 1, s_R, w, d^{e+1} + 1)$$

$$Equal(h, g, d^e + 1, t, w, d^e + 1)$$

$$Elem((1 + aw + t)^{d-1} s_L, w, d^{e+1} + 1) = Elem((1 + aw + t)^{d-1} s_R, w, d^{e+1} + 1).$$

sont satisfaites. Par définition de la relation *Solution*, $Solution(a, b, c_L, c_R, d, e, f, g, h)$ est alors satisfaite. Donc, s'il existe $x_1, \ldots, x_e$ tels que $D(a, x_1, \ldots, x_e) = 0$, alors il existe g et h tels que $Solution(a, b, c_L, c_R, d, e, f, g, h)$ est satisfaite. □

Lemme 4.5.5. *L'équation diophantienne*

$$U(a, b, c_L, c_R, d, e, f, g, h, y_1, \ldots, y_m)$$

représentant la relation $Solution(a, b, c_L, c_R, d, e, f, g, h)$ *est universelle si* g *et* h *sont considérées comme des inconnues.*

Démonstration (*). Soit $D(a, x_1, \ldots, x_n) = 0$ une équation diophantienne quelconque à un paramètre. Soient b^D, c_L^D, c_R^D, d^D, e^D et f^D tels que $(b^D, c_L^D, c_R^D, d^D, e^D, f^D)$ est un code étendu de cette équation. Soit a quelconque. Puisque $(b^D, c_L^D, c_R^D, d^D, e^D, f^D)$ est un code d'équation, alors d'après le lemme 4.5.4, on a

$$\exists g \exists h[Solution(a, b^D, c_L^D, c_R^D, d^D, e^D, f^D, g, h)]$$

$$\Leftrightarrow$$

$$\exists x_1 \ldots \exists x_n[D(a, x_1, \ldots, x_n) = 0].$$

Par définition d'une relation diophantienne, on a

$$Solution(a, b^D, c_L^D, c_R^D, d^D, e^D, f^D, g, h)$$

$$\Leftrightarrow$$

$$\exists y_1 \ldots \exists y_m[U(a, b^D, c_L^D, c_R^D, d^D, e^D, f^D, g, h, y_1, \ldots, y_m) = 0].$$

D'où

$$\exists g \exists h[Solution(a, b^D, c_L^D, c_R^D, d^D, e^D, f^D, g, h)]$$

$$\Leftrightarrow \exists g \exists h \exists y_1 \ldots \exists y_m[U(a, b^D, c_L^D, c_R^D, d^D, e^D, f^D, g, h, y_1, \ldots, y_m) = 0].$$

Donc,

$$\exists x_1 \ldots \exists x_n[D(a, x_1, \ldots, x_n) = 0]$$

$$\Leftrightarrow$$

$$\exists g \exists h \exists y_1 \ldots \exists y_m[U(a, b^D, c_L^D, c_R^D, d^D, e^D, f^D, g, h, y_1, \ldots, y_m) = 0].$$

Nous pouvons alors conclure que pour toute équation diophantienne

$$D(a, x_1 \ldots, x_m) = 0$$

à un paramètre et m inconnues, il existe b^D, c_L^D, c_R^D, d^D, e^D et f^D tels que quel que soit a on a

$$\exists x_1 \ldots \exists x_n[D(a, x_1, \ldots, x_n) = 0]$$

$$\Leftrightarrow$$

$$\exists g \exists h \exists y_1 \ldots \exists y_m [U(a, b^D, c_L^D, c_R^D, d^D, e^D, f^D, g, h, y_1, \ldots, y_m) = 0].$$

Par définition d'une équation diophantienne universelle, nous pouvons conclure que $U(a, b, c_L, c_R, d, e, f, g, h, y_1, \ldots, y_m) = 0$ est une équation diophantienne universelle à un paramètre d'individus (a), six paramètres de code (b, c_L, c_R, d, e, f) et $m+2$ inconnues ($g, h, y_1, \ldots, y_m$). □

Nous venons d'établir l'existence d'une équation diophantienne universelle à un paramètre d'individu et six paramètres de code. D'après le lemme 4.2.1, cela permet d'affirmer l'existence d'une équation diophantienne universelle à un paramètre d'individu et un paramètre de code. D'après le lemme 4.2.2, cela nous permet de conclure à l'existence d'équations diophantiennes universelles à un nombre quelconque de paramètres d'individus et un paramètre de code. Tout ensemble diophantien est donc représentable au moyen d'une équation diophantienne universelle à un paramètre de code et un nombre approprié de paramètres d'individu.

Nous aurons en fait besoin de deux équations universelles particulières, respectivement à aucun et un paramètre d'individu.

Définition 4.5.2. *Soit*

$$U_1(a, k, x_1, \ldots, x_M) = U^2(a, x_1, \ldots, x_M) + (k - 2^{2^6} Cantor_6(x_1, \ldots, x_6))^2.$$

Cet artifice nous permet de transformer l'équation diophantienne universelle obtenue précédemment en une nouvelle équation diophantienne universelle ne possédant qu'un seul paramètre de code. Prouver que

$$U_1(a, k, x_1, \ldots, x_M) = 0$$

est diophantienne universelle n'est pas très complexe.

Lemme 4.5.6. *L'équation $U_1(a, k, x_1, \ldots, x_M) = 0$ est diophantienne universelle.*

Démonstration. Il est évident que l'équation est diophantienne puisque nous savons que $Cantor_6$ est diophantienne.

Soit $E(a, y_1, \ldots, y_n) = 0$ une équation diophantienne quelconque à un paramètre. Puisque $U(a, x_1, \ldots, x_M) = 0$ est diophantienne universelle à six paramètres de code, alors il existe des entiers naturels $x_1, \ldots, x_6$ tels que pour tout naturel a :

$$\exists x_7, \ldots, x_M [U(a, x_1, \ldots, x_M) = 0]$$

$$\Leftrightarrow$$

$$\exists y_1, \ldots, y_n [E(a, y_1, \ldots, y_n) = 0].$$

Soit $k = 2^{2^6} Cantor_6(x_1, \ldots, x_6)$. Si pour un certain entier naturel a et certains naturels $x_1, \ldots, x_6$, il existe $x_7, \ldots, x_M$ tels que

$$U(a, x_1, \ldots, x_M) = 0,$$

alors on a

$$U^2(a, x_1, \ldots, x_M) + (k - 2^{2^6} Cantor_6(x_1, \ldots, x_6))^2 = 0.$$

Inversement, si pour un certain entier naturel a et certains entiers naturels $x_1, \ldots, x_M$, il n'existe pas de solution à l'équation

$$U(a, x_1, \ldots, x_M) = 0,$$

alors il n'existe pas non plus de solution à l'équation

$$U^2(a, x_1, \ldots, x_M) + (k - 2^{2^6} Cantor_6(x_1, \ldots, x_6))^2 = 0.$$

On peut donc affirmer que pour toute équation diophantienne à un paramètre d'individu $E(a, y_1, \ldots, y_n) = 0$, il existe k tel que pour tout entier naturel a,

$$\exists x_1, \ldots, x_M [U_1(a, k, x_1, \ldots, x_M) = 0]$$

$$\Leftrightarrow$$

$$\exists y_1, \ldots, y_n [E(a, y_1, \ldots, y_n) = 0].$$

Par définition, l'équation

$$U_1(a, k, x_1, \ldots, x_M) = 0$$

est diophantienne universelle. □

Matiiassevitch (1995) ajoute une définition supplémentaire concernant l'équation diophantienne universelle U_1. En l'état, on voit que tout entier naturel ne constitue pas un code d'équation pour U_1. En fait, tout nombre qui n'est pas divisible par 2^{2^6} ne peut pas être code d'équation pour U_1. Pour éviter de vérifier que le code d'équation que nous tentons de passer à U_1 est effectivement un code d'équation, on considère que tout entier k ne constituant pas un code d'équation représente l'équation

$$U_1(a, k, x_1, \ldots, x_M) = 0.$$

Il est aisé de constater que cette convention ne change rien au caractère diophantien universel de l'équation.

Nous pouvons désormais conclure à l'existence d'un ensemble non co-diophantien.

Définition 4.5.3. *Soit* $H_1 = \{k \mid \exists x_1, \ldots, x_M[U_1(k, k, x_1, \ldots, x_M) = 0]\}$.

Rappelons que la précaution prise précédemment garantit que tout entier est un code d'équation pour l'équation diophantienne universelle

$$U_1(a, k, x_1, \ldots, x_M) = 0.$$

Lemme 4.5.7. *Soit* $\bar{H}_1$ *le complémentaire de* H_1, *alors* $\bar{H}_1$ *n'est pas diophantien.*

Démonstration. Puisque $\bar{H}_1$ est le complémentaire de H_1, alors $\bar{H}_1$ est un ensemble d'entiers naturels. Par conséquent, si $\bar{H}_1$ est diophantien, cela implique, par définition d'une équation diophantienne universelle, qu'il existe un entier k tel que pour tout entier naturel a, l'équation

$$U_1(a, k, x_1, \ldots, x_M) = 0$$

admet une solution si et seulement si $a \in \bar{H}_1$. On a alors deux cas de figure :

- si $k \in H_1$, alors par définition de H_1, l'équation $U_1(k, k, x_1, \ldots, x_M) = 0$ admet une solution. Par définition d'une équation diophantienne universelle, on a alors $k \in \bar{H}_1$;

- si $k \in \bar{H}_1$, alors par définition d'une équation diophantienne universelle, $U_1(k, k, x_1, \ldots, x_M) = 0$ admet une solution. Ceci implique, par définition de H_1, que $k \in H_1$.

Dans les deux cas, on aboutit à une contradiction, par conséquent, $\bar{H}_1$ ne peut être diophantien. □

Davis (1953) avait établi ce résultat sans utiliser les équations diophantiennes universelles. Toutefois, sa méthode s'appuyait également sur un argument diagonal tel que celui utilisé ici.

En utilisant le résultat précédent, Matiassevitch (1995) introduit un nouvel ensemble non co-diophantien qui servira ultérieurement à achever la preuve de l'indécidabilité du dixième problème de Hilbert. Ce nouvel ensemble est intéressant puisqu'il s'agit en fait de l'ensemble des codes d'équations sans paramètre qui admettent au moins une solution. En fait, une méthode vérifiant que le code d'une équation particulière appartient ou non à cet ensemble serait une réponse au dixième problème de Hilbert. On verra dans le dernier chapitre que le fait d'être non co-diophantien est précisément ce qui interdit à une telle méthode d'exister.

Définition 4.5.4. *Soit* $U_0(k, x_1, \ldots, x_M) = U_1(0, k, x_1, \ldots, x_M)$.

Définition 4.5.5. *Soit* $H_0 = \{k \mid \exists x_1, \ldots, x_M [U_0(k, x_1, \ldots, x_M) = 0]\}$.

Par définition, H_0 est l'ensemble des codes d'équations sans paramètre (puisque ce paramètre est systématiquement nul) qui admettent au moins une solution. Comme dans le cas de H_1, sous sommes assurés que tout entier est un code d'équation grâce à la précaution prise précédemment.

Lemme 4.5.8. *Soit* $W_{p,q}(y_1, \ldots, y_M) = 0$ *l'équation obtenue par substitution de* p *et* q *par des valeurs fixées dans* $U_1(p, q, y_1, \ldots, y_M) = 0$. *Il existe un polynôme* K *à coefficients entiers et à deux inconnues* p *et* q *tel que* $K(p, q)$ *est le code de l'équation* $W_{p,q}(y_1, \ldots, y_M) = 0$.

Démonstration. Rappelons que le code d'une équation est un nombre de la forme

$$2^{2^6} Cantor_6(b, c_L, c_R, d, e, f),$$

où (d, e, f) est un format de l'équation et b, c_L et c_R des nombres satisfaisant certaines contraintes. On choisit d supérieur au degré total de l'équation

$$U_1(p, q, y_1, \ldots, y_M) = 0.$$

On aura alors nécessairement d supérieur au degré total de l'équation $W_{p,q}(y_1, \ldots, y_M)$ (le degré total ne peut pas augmenter si on supprime des inconnues). On choisit $e = M$. On a bien e égal au nombre d'inconnues de l'équation

$$W_{p,q}(y_1, \ldots, y_M) = 0.$$

On choisit $f = 2^{d^1} + \cdots + 2^{d^e}$. Le triplet (d, e, f) est bien un format d'équation tel que défini précédemment. À aucun moment les valeurs de p et q ne sont intervenues dans le choix de d, e et f.

Transformons maintenant l'équation

$$U_1(p, q, y_1, \ldots, y_M) = 0$$

en l'équation équivalente

$$C_L(p, q, y_1, \ldots, y_M) = C_R(p, q, y_1, \ldots, y_M),$$

où $C_R(p, q, y_1, \ldots, y_M)$ et $C_L(p, q, y_1, \ldots, y_M)$ sont des polynômes à coefficients positifs. On peut donc écrire

$$C_L(p, q, y_1, \ldots, y_M) = \sum_{u+v+i_1+\cdots+i_M<d} c_{L,u,v,i_1,\ldots,i_M} p^u q^v y_1^{i_1} \ldots y_M^{i_M}$$

et

$$C_R(p, q, y_1, \ldots, y_M) = \sum_{u+v+i_1+\cdots+i_M<d} c_{R,u,v,i_1,\ldots,i_M} p^u q^v y_1^{i_1} \ldots y_M^{i_M}.$$

Ceci est équivalent à

$$C_L(p, q, y_1, \ldots, y_M) = \sum_{i_1+\cdots+i_M<d} (y_1^{i_1} \ldots y_M^{i_M} (\sum_{u+v<d} c_{L,u,v,i_1,\ldots,i_M} p^u q^v))$$

et

$$C_R(p,q,y_1,\ldots,y_M) = \sum_{i_1+\cdots+i_M<d} (y_1^{i_1}\ldots y_M^{i_M}(\sum_{u+v<d} c_{R,u,v,i_1,\ldots,i_M}p^uq^v)).$$

Si p et q sont des inconnues, alors

$$(\textstyle\sum_{u+v<d} c_{L,u,v,i_1,\ldots,i_M}p^uq^v) \quad \text{et} \quad (\textstyle\sum_{u+v<d} c_{R,u,v,i_1,\ldots,i_M}p^uq^v)$$

sont des polynômes en p et q à coefficients entiers positifs ou nuls. On remarquera qu'ici, le fait que $u+v+i_0+\cdots+i_M$ soit potentiellement supérieur à d n'est pas problématique. Dans ce cas, $c_{u,v,i_1,\ldots,i_M} = 0$ et n'influence donc pas le résultat. Soit

$$b = d!(\sum_{i_1+\cdots+i_M<d} (\sum_{u+v<d} c_{L,u,v,i_1,\ldots,i_M}p^uq^v) + (\sum_{u+v<d} c_{R,u,v,i_1,\ldots,i_M}p^uq^v)) + 1.$$

Ici, b est donné par un polynôme en p et q à coefficients positifs ou nuls, puisque $d!$ est positif et les $c_{L,u,v,i_1,\ldots,i_M}$ et $c_{R,u,v,i_1,\ldots,i_M}$ sont positifs ou nuls. De plus, on a

$$b > d!\max\{c_{L,u,v,i_1,\ldots,i_M}p^uq^v\}$$

et

$$b > d!\max\{c_{R,u,v,i_1,\ldots,i_M}p^uq^v\}.$$

Par conséquent, $b^{d^{m+1}-i_1d^0-\cdots-i_Md^{M-1}}$ est un polynôme en p et q à coefficients entiers positifs ou nuls.

Soient c_L et c_R tels que

$$c_L = \sum_{i_1+\cdots+i_M<d} i_1!\ldots i_M!(d-1-i_1-\cdots-i_M)!(\sum_{u+v<d} c_{L,u,v,i_1,\ldots,i_M}p^uq^v)b^{d^{m+1}-i_1d^0-\cdots-i_Md^{M-1}}$$

et

$$c_R = \sum_{i_1+\cdots+i_M<d} i_1!\ldots i_M!(d-1-i_1-\cdots-i_M)!(\sum_{u+v<d} c_{R,u,v,i_1,\ldots,i_M}p^uq^v)b^{d^{m+1}-i_1d^0-\cdots-i_Md^{M-1}}.$$

Puisque $b^{d^{m+1}-i_1d^0-\cdots-i_Md^{M-1}}$ est un polynôme en p et q à coefficients entiers positifs ou nuls, alors c_L et c_R le sont également. Donc, b, c_L et c_R sont des polynômes en p et q à coefficients entiers positifs ou nuls. Par conséquent, compte tenu du corollaire 3.1.5,

$$2^{2^6}Cantor_6(b,c_L,c_R,d,e,f)$$

est un polynôme en p et q à coefficients entiers positifs ou nuls.
De plus, puisque (b, c_L, c_R, d, e, f) est un code étendu de l'équation

$$W_{p,q}(y_1, \ldots, y_M) = 0,$$

alors $2^{2^6} Cantor_6(b, c_L, c_R, d, e, f)$ est un code de cette même équation. Donc,

$$K(p, q) = 2^{2^6} Cantor_6(b, c_L, c_R, d, e, f)$$

est un polynôme en p et q à coefficients entiers positifs ou nuls.

□

Lemme 4.5.9. *Pour tout $a \geq 0$, on a*

$$a \in H_1 \Leftrightarrow K(a, a) \in H_0$$

Démonstration. ($\Rightarrow$) Soit $a \in H_1$. Par définition de H_1, il existe $u_1, \ldots u_M$ tels que

$$U_1(a, a, u_1, \ldots, u_M) = 0.$$

Par définition de K, $K(a, a)$ est le code de l'équation $W_{a,a}(y_1, \ldots, y_M)$. Puisque

$$W_{a,a}(y_1, \ldots, y_M) = 0$$

est obtenu par substitution de p et q dans l'équation

$$U_1(p, q, y_1, \ldots, y_M) = 0$$

et que $U_1(a, a, u_1, \ldots, u_M) = 0$, alors on a naturellement

$$W_{a,a}(u_1, \ldots, u_M) = 0.$$

Il existe donc, par définition d'une équation diophantienne universelle, $v_1, \ldots, v_M$ tels que $U_0(K(a, a), v_1, \ldots, v_M) = 0$. Par définition de H_0, on a alors $K(a, a) \in H_0$.

($\Leftarrow$) Soit $K(a, a) \in H_0$. On a alors, par définition de H_0, il existe $u_1, \ldots, u_M$ tels

que $U_0(K(a,a), u_1, \ldots, u_M) = 0$. Puisque U_0 est une équation diophantienne universelle et que $K(a,a)$ est par définition le code de l'équation $W_{a,a}(y_1, \ldots, y_M) = 0$, alors, par définition d'une équation diophantienne universelle, il existe $v_1, \ldots v_M$ tels que $W_{a,a}(v_1, \ldots, v_M) = 0$. Donc, puisque $W_{a,a}(y_1, \ldots, y_M) = 0$ est obtenu par substitution des variables p et q par la valeur a dans l'équation $U_1(a, a, y_1, \ldots, y_M) = 0$, alors on a $U_1(a, a, v_1, \ldots, v_M) = 0$. Par définition de H_1, on a donc $a \in H_1$. □

Du lemme précédent, on peut immédiatement conclure que

$$a \in \bar{H}_1 \Leftrightarrow K(a,a) \in \bar{H}_0.$$

On est maintenant en mesure de prouver que $\bar{H}_0$ n'est pas diophantien.

Lemme 4.5.10. *Soit $\bar{H}_0$ le complémentaire de H_0, alors $\bar{H}_0$ n'est pas diophantien.*

Démonstration. Supposons que $\bar{H}_0$ soit diophantien. Par définition de H_0 et d'après ce qui précède, on a donc

$$K(a,a) \in \bar{H}_0 \Leftrightarrow \exists y_1, \ldots, y_M U_0(K(a,a), y_1, \ldots, y_M) = 0.$$

On a

$$\exists y_1, \ldots, y_M U_0(K(a,a), y_1, \ldots, y_M) = 0 \Leftrightarrow \exists y_1, \ldots, y_M, x[x = K(a,a) \wedge U_0(x, y_1, \ldots, y_M) = 0].$$

Or, K est un polynôme à coefficients entiers, par conséquent, on obtient

$$a \in \bar{H}_1 \Leftrightarrow \exists y_1, \ldots, y_M, x[x = K(a,a) \wedge U_0(x, y_1, \ldots, y_M) = 0]$$

ce qui signifie que $\bar{H}_1$ est un ensemble diophantien. Or, nous avons vu au lemme 4.5.7 que ce n'est pas le cas.
Donc, $\bar{H}_0$ n'est pas un ensemble diophantien. □

Donc, l'ensemble des codes d'équations diophantiennes sans solution n'est pas diophantien. Il s'agit d'une observation importante pour la suite, puisqu'elle nous permettra de conclure à l'indécidabilité du dixième problème de Hilbert. En effet, dans

le chapitre 6, nous prouverons l'équivalence entre les ensembles diophantiens et les ensembles semi-décidables ainsi que le fait que si un ensemble est décidable, alors son complémentaire est semi-décidable. Puisque $\bar{H}_0$ n'est pas diophantien, alors il n'est pas semi-décidable et donc H_0 n'est pas décidable.

CHAPITRE V

REPRÉSENTATION DIOPHANTIENNE D'ENDOFONCTIONS

Avant de pouvoir conclure la démonstration de l'indécidabilité du dixième problème de Hilbert, nous devons définir un nouvel outil qui nous servira dans le dernier chapitre. Il s'agit de montrer que toute endofonction diophantienne définie sur un ensemble fini peut être transformée en une fonction diophantienne s'appliquant sur les éléments d'uplets. Ceci est extrêmement utile puisqu'on peut alors appliquer n'importe quelle endofonction de ce type aux éléments d'uplets. De plus, puisque la fonction étendue aux uplets manipule en fait les codes positionnels d'uplets, on n'aura pas à procéder à l'extraction des éléments des uplets. Cette méthode est décrite en partie par Matiiassevitch (1995) qui laisse au lecteur le soin de généraliser le cas particulier des fonctions à un argument.

Dans un premier temps, on étudiera le fait que la transformation peut être utilisée sur des endofonctions à un argument, puis nous généraliserons le résultat aux endofonctions à un nombre quelconque d'arguments. Cette démarche permettra d'exposer la méthode relativement simplement dans le cas d'une variable, ce qui rendra le traitement du cas de plusieurs variables plus clair.

Dans tout ce chapitre, on utilisera des bases d'encodage $b \geq 3$. On utilisera de plus la notation N_b pour représenter l'ensemble $\{0, \ldots, b-1\}$.

5.1 Fonctions à un argument

Définition 5.1.1. *Soit $F : N_b \to N_b$ une endofonction quelconque. Soit la fonction $F[b] : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ telle que si (a, b, c) est un code positionnel de l'uplet $(a_1, \ldots, a_c) \in N_b^c$, alors $(F[b](a, c), b, c)$ est un code positionnel de l'uplet $(F(a_1), \ldots, F(a_c))$.*

On va montrer que $F[b]$ est diophantienne.

Définition 5.1.2. *Soit $(a_1, \ldots, a_c) \in N_b^c$. On appelle* uplets caractéristiques de $(a_1, \ldots, a_c)$ *les uplets*

$$(h_{0,1}, \ldots, h_{0,c})$$

$$\vdots$$

$$(h_{b-1,1}, \ldots, h_{b-1,c})$$

tels que pour tout $k \in \{0, \ldots, b-1\}$ et tout $j \in \{1, \ldots, c\}$ on a

$$h_{k,j} = \begin{cases} 1 & \text{si } a_j = k \\ 0 & \text{si } a_j \neq k. \end{cases}$$

Lemme 5.1.1. *Soient*

$$(h_{0,1}, \ldots, h_{0,c}), \ldots, (h_{b-1,1}, \ldots, h_{b-1,c})$$

les uplets caractéristiques d'un uplet $(a_1, \ldots, a_c) \in N_b^c$ quelconque. Quel que soit $j \in \{1, \ldots, c\}$, il existe un unique $k \in \{0, \ldots, b-1\}$ tel que $h_{k,j} = 1$.

Démonstration. Pour tout $j \in \{1 \ldots, c\}$, on sait que $a_j \in \{0, \ldots, b-1\}$. Donc, si $k = a_j$, on a $h_{k,j} = 1$ par définition des uplets caractéristiques. De plus, il ne peut exister $k' \neq k$ tel que $h_{k',j} = 1$, puisque ceci impliquerait que $k' = a_j = k$. □

Lemme 5.1.2. *Soit (a, b, c) un code positionnel de l'uplet $(a_1, \ldots, a_c)$. Soient (h_0, b, c), $(h_1, b, c), \ldots, (h_{b-1}, b, c)$ des codes positionnels des uplets caractéristiques de $(a_1, \ldots, a_c)$. Alors*

$$\sum_{k=0}^{b-1} k h_k = a.$$

Démonstration (*).

$$\begin{aligned} \sum_{k=0}^{b-1} k h_k &= \sum_{k=0}^{b-1} \sum_{j=1}^{c} k h_{k,j} b^{j-1} \\ &= \sum_{j=1}^{c} \sum_{k=0}^{b-1} k h_{k,j} b^{j-1}. \end{aligned}$$

Or, pour tout $k \in \{0, \ldots, b-1\}$ et tout $j \in \{1, \ldots, c\}$, on a $h_{k,j} = 1$ si $a_j = k$ et $h_{k,j} = 0$ sinon. D'après le lemme 5.1.1, on sait qu'il existe un unique $k_j \in N_b$ tel que $k_j = a_j$ et $h_{k_j,j} = 1$. On a donc

$$\sum_{k=0}^{b-1} k h_{k,j} b^{j-1} = k_j b^{j-1}.$$

On peut alors conclure :

$$\begin{aligned} \sum_{k=0}^{b-1} k h_k &= \sum_{j=1}^{c} \sum_{k=0}^{b-1} k h_{k,j} b^{j-1} \\ &= \sum_{j=1}^{c} a_j b^{j-1} \\ &= a. \end{aligned}$$

□

Si on écrit les uplets caractéristiques les uns sous les autres dans une matrice, on réalise que le lemme précédent signifie que chaque colonne de cette matrice contient un unique 1. Le lemme suivant fait appel à la fonction *Repeat*, déjà définie en 3.3.6.

Lemme 5.1.3. *Soit (a, b, c) un code positionnel de l'uplet $(a_1, \ldots, a_c)$. Soient (h_0, b, c), $(h_1, b, c), \ldots, (h_{b-1}, b, c)$ des codes positionnels des uplets caractéristiques de $(a_1, \ldots, a_c)$.*

Alors

$$\sum_{k=0}^{b-1} h_k = Repeat(1, b, c) = \frac{b^c - 1}{b - 1}.$$

Démonstration (*). D'après le lemme 5.1.1, pour tout $j \in \{1, \ldots, c\}$, il existe un unique $k_j \in N_b$ tel que $h_{k_j,j} = 1$. On a donc

$$\begin{aligned}
\sum_{k=0}^{b-1} h_j &= \sum_{k=0}^{b-1} \sum_{j=1}^{c} h_{k,j} b^{j-1} \\
&= \sum_{j=1}^{c} \sum_{k=0}^{b-1} h_{k,j} b^{j-1} \\
&= \sum_{j=1}^{c} b^{j-1} \\
&= \sum_{j=0}^{c-1} b^j \\
&= \frac{b^c - 1}{b - 1}.
\end{aligned}$$

□

Définition 5.1.3. *Soient (q_1, b, c) et (q_2, b, c) des codes positionnels des uplets $(a_1, \ldots, a_c)$ et $(a'_1, \ldots, a'_c)$. La relation $Ortnorm_b(q_1, q_2, c)$ est satisfaite si et seulement si pour tout $i \in \{1, \ldots, c\}$, on a $a_i \leq 1$, $a'_i \leq 1$ et $a_i = 1$ seulement si $a'_i = 0$.*

Lemme 5.1.4 (*). *Soit (a, b, c) un code positionnel de l'uplet $(a_1, \ldots, a_c)$. Soient (h_0, b, c), (h_1, b, c), $\ldots$, (h_{b-1}, b, c) des codes positionnels des uplets caractéristiques de $(a_1, \ldots, a_c)$. Alors, pour tous k et ℓ tels que $0 \leq k < \ell < b$, la relation $Ortnorm_b(h_k, h_\ell, c)$ est satisfaite.*

Démonstration (*). Par définition des uplets caractéristiques, quel que soit $i \in \{1, \ldots, c\}$, on a $h_{k,i} \leq 1$ et $h_{\ell,i} \leq 1$. Soient k et ℓ tels que $0 \leq k < \ell < b$. Soient $(h_{k,1}, \ldots, h_{k,c})$ et $(h_{\ell,1}, \ldots, h_{\ell,c})$ les uplets codés respectivement par (h_k, b, c) et (h_ℓ, b, c). Soit $j \in \{1, \ldots, c\}$.

- Si $h_{k,j} = 1$, alors par définition des uplets caractéristiques, on a $a_j = k$. Donc, puisque $k < \ell$, on a $h_{\ell,j} = 0$.
- Si $h_{\ell,j} = 1$, alors par définition des uplets caractéristiques, on a $a_j = \ell$. Donc, puisque $k < \ell$, on a $h_{k,j} = 0$.

La relation $Ortnorm_b(h_\ell, h_k, c)$ est donc satisfaite. □

Lemme 5.1.5. *Soient q_1, q_2, b et c des entiers naturels. On a*

$$Ortnorm_b(q_1, q_2, c) \Leftrightarrow Small(q_1, b, c, 1) \wedge Small(q_2, b, c, 1) \wedge Small(q_1 + q_2, b, c, 1).$$

Démonstration (*). ($\Rightarrow$) Supposons que la relation $Ortnorm_b(q_1, q_2, c)$ soit satisfaite. Par définition, (q_1, b, c) et (q_2, b, c) sont des codes positionnels. Soient $(x_1, \ldots, x_c)$ et $(y_1, \ldots, y_c)$ les uplets respectivement codés par (q_1, b, c) et (q_2, b, c).
On a

$$q_1 = \sum_{i=1}^{c} x_i b^{i-1} \text{ et}$$

$$q_2 = \sum_{i=1}^{c} y_i b^{i-1}$$

et puisque $Ortnorm_b(q_1, q_2, c)$ est satisfaite, on sait que pour tout $i \in \{1, \ldots, c\}$, $x_i \leq 1$, $y_i \leq 1$ et $x_i + y_i \leq 1$. Par conséquent, les relations

$$Small(q_1, b, c, 1)$$

$$Small(q_2, b, c, 1)$$

sont satisfaites.
De plus, on a

$$\begin{aligned} q_1 + q_2 &= \sum_{i=1}^{c} x_i b^{i-1} + \sum_{i=1}^{c} y_i b^{i-1} \\ &= \sum_{i=1}^{c} (x_i + y_i) b^{i-1}. \end{aligned}$$

Par définition, (q_1+q_2, b, c) est un code positionnel de l'uplet $(x_1+y_1, \ldots, x_c+y_c)$, avec, quel que soit $i \in \{1, \ldots, c\}$,

$$x_i + y_i \leq 1.$$

Donc, la relation $Small(q_1 + q_2, b, c, 1)$ est satisfaite.

($\Leftarrow$) Supposons que les relations

$$Small(q_1, b, c, 1),\ Small(q_2, b, c, 1),\ Small(q_1 + q_2, b, c, 1)$$

soient satisfaites. Par définition de la relation $Small$, (q_1, b, c) et (q_2, b, c) sont des codes positionnels. Soient $(x_1, \ldots, x_c)$ et $(y_1, \ldots, y_c)$ les uplets respectivement codés par (q_1, b, c) et (q_2, b, c). On a alors

$$q_1 = \sum_{i=1}^{c} x_i b^{i-1} \text{ et}$$

$$q_2 = \sum_{i=1}^{c} y_i b^{i-1}$$

avec $x_i \leq 1$ et $y_i \leq 1$ pour tout $i \in \{1, \ldots, c\}$, d'où $x_i + y_i \leq 2$. On a alors

$$\begin{aligned} q_1 + q_2 &= \sum_{i=1}^{c} x_i b^{i-1} + \sum_{i=1}^{c} y_i b^{i-1} \\ &= \sum_{i=1}^{c} (x_i + y_i) b^{i-1}, \end{aligned}$$

et puisque $x_i + y_i \leq 2 < b$, alors par définition, $(q_1 + q_2, b, c)$ est un code positionnel de l'uplet $(x_1 + y_1, \ldots, x_c + y_c)$. Puisque $Small(q_1 + q_2, b, c, 1)$ est satisfaite, alors on a

$$x_i + y_i \leq 1$$

pour tout $i \in \{1, \ldots, c\}$.

Par conséquent, si $x_i = 1$, alors $y_i = 0$ pour tout $i \in \{1, \ldots, c\}$. De même, si $y_i = 1$, alors $x_i = 0$ pour tout $i \in \{1, \ldots, c\}$. On constate donc que la relation $Ortnorm_b(q_1, q_2, c)$ est satisfaite. □

Le lemme 5.1.5 nous permet de conclure que la relation $Ortnorm_b$ est diophantienne.

Lemme 5.1.6. *Soit (a, b, c) un code positionnel de l'uplet $(a_1, \ldots, a_c)$. Soient $h_0, \ldots, h_{b-1}$ des entiers naturels. Alors $(h_0, b, c), \ldots, (h_{b-1}, b, c)$ sont des codes positionnels des uplets caractéristiques de $(a_1, \ldots, a_c)$ si et seulement si les relations suivantes sont satisfaites*

$$\sum_{k=0}^{b-1} kh_k = a, \sum_{k=0}^{b-1} h_k = Repeat(1, b, c), \bigwedge_{0 \leq k < \ell < b} Ortnorm_b(h_k, h_\ell, c).$$

Démonstration (*). ($\Rightarrow$) Si $(h_0, b, c), \ldots, (h_{b-1}, b, c)$, sont des codes positionnels des uplets caractéristiques de $(a_1, \ldots, a_c)$, alors d'après les lemmes 5.1.2, 5.1.3 et 5.1.4, on a

$$\sum_{k=0}^{b-1} kh_k = a, \sum_{k=0}^{b-1} h_k = Repeat(1, b, c), \bigwedge_{0 \leq k < \ell < b} Ortnorm_b(h_k, h_\ell, c).$$

($\Leftarrow$) Supposons que

$$\sum_{k=0}^{b-1} kh_k = a, \sum_{k=0}^{b-1} h_k = Repeat(1, b, c), \bigwedge_{0 \leq \ell < k < b} Ortnorm_b(h_k, h_\ell, c)$$

soient satisfaites. Puisque

$$\bigwedge_{0 \leq k < \ell < b} Ortnorm_b(h_k, h_\ell, c)$$

est satisfaite, alors par définition, $(h_0, b, c), \ldots (h_{b-1}, b, c)$ sont des codes positionnels d'uplets appartenant à $\{0, 1\}^c$. Soient $(h_{0,1}, \ldots, h_{0,c})$, $\ldots$, $(h_{b-1,1}, \ldots, h_{b-1,c})$ les uplets codés respectivement par (h_0, b, c), $\ldots$, (h_{b-1}, b, c). On sait que pour tout $k \in N_b$, tout $\ell \in N_b$ et tout $j \in \{1, \ldots, c\}$, on a

$$h_{k,j} = 1 \Rightarrow h_{\ell,j} = 0.$$

On peut donc écrire, pour tout $j \in \{1, \ldots, c\}$,

$$\sum_{\ell=0}^{b-1} h_{\ell,j} \leq 1.$$

On sait de plus que

$$\begin{aligned}
\sum_{\ell=0}^{b-1} h_\ell &= Repeat(1,b,c) \\
\sum_{\ell=0}^{b-1} h_\ell &= \frac{b^c-1}{b-1} \\
\sum_{\ell=0}^{b-1} h_\ell &= \sum_{i=0}^{c-1} b^i \\
\sum_{\ell=0}^{b-1}\sum_{j=1}^{c} h_{\ell,j} b^{j-1} &= \sum_{i=1}^{c} b^{i-1} \\
\sum_{i=1}^{c}\left(\sum_{\ell=0}^{b-1} h_{\ell,i}\right) b^{i-1} &= \sum_{i=1}^{c} b^{i-1}.
\end{aligned}$$

Par définition, $(\sum_{i=0}^{c-1} b^i, b, c)$ est un code positionnel du c-uplet $(1,\ldots,1)$. De plus, d'après ce qui précède, on a

$$\sum_{i=0}^{b-1} h_{\ell,i} \leq 1 < b.$$

Donc, par unicité de la représentation d'un nombre en base b, on a pour tout $i \in \{1,\ldots,c\}$,

$$\sum_{\ell=0}^{b-1} h_{\ell,i} = 1.$$

Ceci implique que pour tout $i \in \{1,\ldots,c\}$, il existe un unique $\ell \in N_b$ tel que $h_{\ell,i} = 1$ et $h_{k,i} = 0$ pour tout $k \neq \ell$.

Soient $j_1,\ldots,j_c \in \{0,\ldots,b-1\}$ tels que $h_{j_1,1} = 1,\ldots,h_{j_c,c} = 1$.

On a

$$\begin{aligned} a &= \sum_{\ell=0}^{b-1} \ell h_\ell \\ &= \sum_{\ell=0}^{b-1} \ell \sum_{j=1}^{c} h_{\ell,j} b^{j-1} \\ &= \sum_{\ell=0}^{b-1} \sum_{j=1}^{c} \ell h_{\ell,j} b^{j-1} \\ &= \sum_{j=1}^{c} \left(\sum_{\ell=0}^{b-1} \ell h_{\ell,j} \right) b^{j-1}. \end{aligned}$$

De plus, par définition de (a, b, c), on a

$$a = \sum_{i=1}^{c} a_i b^{i-1}.$$

Par unicité de la représentation d'un nombre en base b, on a, pour tout $i \in \{1, \ldots, c\}$,

$$\sum_{\ell=0}^{b-1} \ell h_{\ell,i} = a_i.$$

On sait que $h_{\ell,i} = 1$ si $\ell = j_i$ et $h_{\ell,i} = 0$ sinon. On a donc

$$\sum_{\ell=0}^{b-1} \ell h_{\ell,i} = h_{j_i,i} \cdot j_i = a_i.$$

Donc, pour tout $\ell \in N_b$ et tout $i \in \{1, \ldots, c\}$, on a

$$h_{\ell,i} = \begin{cases} 1 & \text{si } a_i = \ell \\ 0 & \text{si } a_i \neq \ell. \end{cases}$$

Par définition, les uplets $(h_{0,1}, \ldots, h_{0,c})$, $\ldots$, $(h_{b-1,1}, \ldots, h_{b-1,c})$ codés par (h_0, b, c), $\ldots$, (h_{b-1}, b, c) sont les uplets caractéristiques de $(a_1, \ldots, a_c)$. □

Lemme 5.1.7. *Soit une fonction* $F : \{0, \ldots, b-1\} \to \{0, \ldots, b-1\}$. *Soit* (a, b, c) *un code positionnel de l'uplet* $(a_1, \ldots, a_c) \in N_b^c$. *Soient* $(h_0, b, c), \ldots, (h_{b-1}, b, c)$ *des codes positionnels des uplets caractéristiques*

$$(h_{0,1}, \ldots, h_{0,c}), \ldots, (h_{b-1,1}, \ldots, h_{b-1,c})$$

de $(a_1, \ldots, a_c)$. *Alors,* $(F(0)h_0 + \cdots + F(b-1)h_{b-1}, b, c)$ *est un code positionnel de l'uplet* $(F(a_1), \ldots, F(a_c))$.

Démonstration (*). On a

$$\begin{aligned}\sum_{k=0}^{b-1} F(k)h_k &= \sum_{k=0}^{b-1}\sum_{j=1}^{c} F(k)h_{k,j}b^{k-1} \\ &= \sum_{j=1}^{c}\sum_{k0}^{b-1} F(k)h_{k,j}b^{k-1} \\ &= \sum_{j=1}^{c}(\sum_{k=0}^{b-1} F(k)h_{\ell,j})b^{k-1}.\end{aligned}$$

D'après le lemme 5.1.1, pour tout $j \in \{1, \ldots, c\}$, il existe un unique $k_j \in \{0, \ldots, b-1\}$ tel que $h_{k_j,j} = 1$ et par définition, $a_j = k_j$.

On a alors

$$\sum_{k=0}^{b-1} F(k)h_{k,j} = F(a_j).$$

On obtient

$$\sum_{k=0}^{b-1} F(k)h_k = \sum_{j=1}^{c} F(a_j)b^{j-1}$$

et puisque $F(a_k) < b$ pour tout $j \in \{1, \ldots, c\}$, alors $(\sum_{k=0}^{b-1} F(k)h_k, b, c)$ est par définition un code positionnel de l'uplet $(F(a_1), \ldots, F(a_c))$. □

On peut maintenant conclure que l'application d'une endofonction à un uplet est diophantienne.

Lemme 5.1.8. *Soit* $b \geq 3$. *Soit la fonction* $F : N_b \to N_b$. *Soit la fonction* $F[b] : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *telle que si* (a, b, c) *est un code positionnel de l'uplet* $(a_1, \ldots, a_c)$ *à valeurs dans* N_b, *alors* $(F[b](a, c), b, c)$ *est un code positionnel de l'uplet* $(F(a_1), \ldots, F(a_c))$. *Soit* (a, b, c) *un code de l'uplet* $(a_1, \ldots, a_c)$ *à valeurs dans* N_b. *Alors, on a l'équivalence suivante*

$$F[b](a, c) = a'$$

$$\Leftrightarrow$$

$$\exists h_0 \ldots \exists h_{b-1} \left[\sum_{k=0}^{b-1} k h_k = a \wedge \sum_{k=0}^{b-1} h_k = Repeat(1, b, c) \wedge \right.$$

$$\left. \bigwedge_{0 \leq k < l < b} Ortnorm_b(h_k, h_l, c) \wedge \sum_{k=0}^{b-1} F(k) h_k = a' \right].$$

Démonstration (*). ($\Rightarrow$) Supposons que $F[b](a, c) = a'$. Soient $(h_0, b, c), \ldots, (h_{b-1}, b, c)$ des codes positionnels des uplets caractéristiques de $(a_1, \ldots, a_c)$. D'après les lemmes 5.1.2, 5.1.3 et 5.1.4, les relations

$$\sum_{k=0}^{b-1} k h_k = a, \sum_{k=0}^{b-1} h_k = Repeat(1, b, c), \bigwedge_{0 \leq k < \ell < b} Ortnorm_b(h_k, h_\ell, c)]$$

sont satisfaites. Donc, d'apès le lemme 5.1.7,

$$a' = F[b](a, c) = \sum_{k=0}^{b-1} F(k) h_k.$$

($\Leftarrow$) Supposons qu'il existe $h_0, \ldots, h_{b-1}$ tels que les relations suivantes soient satisfaites :

$$\sum_{k=0}^{b-1} k h_k = a, \qquad \sum_{k=0}^{b-1} h_k = Repeat(1, b, c),$$
$$\bigwedge_{0 \leq k < \ell < b} Ortnorm_b(h_k, h_\ell, c) \text{ et } \sum_{k=0}^{b-1} F(k) h_k = a'.$$

D'après le lemme 5.1.6, $(h_0, b, c), \ldots, (h_{b-1}, b, c)$ sont des codes positionnels des uplets caractéristiques de $(a_1, \ldots, a_c)$. Donc, d'après le lemme 5.1.7, (a', b, c) est un code positionnel de l'uplet $(F(a_1), \ldots, F(a_c))$, d'où par définition $F[b](a, c) = a'$. □

5.2 Fonctions à plusieurs arguments

Matiiassevitch (1995) prétend qu'on peut adapter les résultats obtenus précédemment pour prouver qu'une fonction à plusieurs arguments appliquée aux éléments d'uplets est également diophantienne. Toutefois, il n'en fait pas la preuve. Les résultats de cette section ne se trouvent pas dans le livre de Matiiassevitch (1995). On travaille maintenant avec une fonction quelconque $F : N_b^m \to N_b$ et on définit la fonction

$F[b] : \mathbb{N}^m \times \mathbb{N} \to \mathbb{N}$ telle que si $(a_1, b, c), \ldots, (a_m, b, c)$ sont des codes positionnels des uplets

$$(a_{1,1}, \ldots, a_{1,c}), \ldots, (a_{m,1}, \ldots, a_{m,c}),$$

alors $(F(a_1, \ldots, a_m, c), b, c)$ est un code positionnel de l'uplet

$$(F(a_{1,1}, \ldots, a_{m,1}), \ldots, F(a_{1,c}, \ldots, a_{m,c})).$$

Pour alléger la notation, on utilisera dans cette section des noms de variables en gras pour représenter des uplets et on utilisera une notation normale avec indice pour représenter les éléments d'uplets. La notation normale sans indice représentera le numéro de code positionnel de cet uplet. Par exemple, si un uplet est représenté par la variable $\mathbf{a}$, alors a_i est le i-ème élément de $\mathbf{a}$ et a est le numéro de code positionnel de $\mathbf{a}$ dans une certaine base. Définissons maintenant la notion d'uplets caractéristiques de plusieurs uplets.

Définition 5.2.1. *Soit* $b \geq 3$. *Soient* $\mathbf{a}_1, \ldots, \mathbf{a}_m \in N_b^c$. *Pour tout* $\mathbf{k} = (k_1, \ldots, k_m) \in N_b^m$ *et tout* $j \in \{1, \ldots, c\}$, *on pose*

$$h_{\mathbf{k},j} = \begin{cases} 1 & \text{si } a_{1,j} = k_1, \ldots, a_{m,j} = k_m \\ 0 & \text{sinon.} \end{cases}$$

Les uplets $\mathbf{h_k} = (h_{\mathbf{k},1}, \ldots, h_{\mathbf{k},c})$ *sont appelés* uplets caractéristiques des uplets $\mathbf{a}_1, \ldots, \mathbf{a}_m$.

Lemme 5.2.1 $(*)$. *Soient* $\mathbf{a}_1, \ldots, \mathbf{a}_m \in N_b^c$. *Soient les uplets de la forme* $\mathbf{h_k} = (h_{\mathbf{k},1}, \ldots, h_{\mathbf{k},c})$, *pour* $\mathbf{k} \in N_b^m$, *les uplets caractéristiques de* $\mathbf{a}_1, \ldots, \mathbf{a}_m$. *Quel que soit* $j \in \{1, \ldots, c\}$, *il existe un unique* $\mathbf{k} = (k_1, \ldots, k_m) \in N_b^c$ *tel que* $h_{\mathbf{k},j} = 1$.

Démonstration (*). Pour tout $j \in \{1, \ldots, c\}$, on sait que

$$a_{1,j}, \ldots, a_{m,j} \in N_b.$$

Donc, si $k_1 = a_{1,j}, \ldots, k_m = a_{m,j}$, on a $h_{(k_1,\ldots,k_m),j} = 1$ par définition des uplets caractéristiques. De plus, il ne peut exister $k'_1, \ldots, k'_m \in \{0, \ldots, b-1\}$ tels qu'il existe $q \in \{1, \ldots, m\}$ tel que $k'_q \neq k_q$ et $h_{(k'_1,\ldots,k'_m),i} = 1$, puisque ceci impliquerait que $k'_1 = a_{1,j} = k_1, \ldots, k'_m = a_{m,j} = k_m$. □

Comme pour son homologue de la section précédente, ce lemme signifie en fait que dans la représentation des uplets caractéristiques sous forme de matrice, il existe un unique 1 par colonne de cette matrice.

Lemme 5.2.2. *(*) Soient* $(a_1, b, c), \ldots, (a_m, b, c)$ *les codes positionnels des uplets* $\mathbf{a}_1, \ldots, \mathbf{a}_m \in N_b^c$ *et* $(h_\mathbf{k}, b, c)$ *le code positionnel de l'uplet caractéristique* $(h_{\mathbf{k},1}, \ldots, h_{\mathbf{k},c})$ *pour tout* $\mathbf{k} \in N_b^c$. *Pour tout* $\ell \in \{1, \ldots, m\}$, *on a*

$$\sum_{\mathbf{k} \in N_b^c} h_\mathbf{k} k_\ell = a_\ell.$$

Démonstration. Démonstration (*) Par définition, on a

$$\begin{aligned} \sum_{\mathbf{k} \in N_b^c} h_\mathbf{k} k_\ell &= \sum_{\mathbf{k} \in N_b^c} (\sum_{j=1}^{c} h_{\mathbf{k},j} b^{j-1}) k_\ell \\ &= \sum_{j=1}^{c} \sum_{\mathbf{k} \in N_b^c} h_{\mathbf{k},j} b^{j-1} k_\ell. \end{aligned}$$

D'après le lemme 5.2.1, pour tout $j \in \{1, \ldots, c\}$, il existe un unique $\mathbf{k} \in N_b^c$ tel que $h_{\mathbf{k},j} = 1$. Par définition des $h_{\mathbf{k},j}$, on a alors $k_l = a_{\ell,j}$.
On a donc

$$\begin{aligned} \sum_{j=1}^{c} \sum_{\mathbf{k} \in N_b^c} h_{\mathbf{k},j} b^{j-1} i_\ell &= \sum_{j=1}^{c} b^{j-1} a_{\ell,j} \\ &= \sum_{j=1}^{c} a_{\ell,j} b^{j-1}. \end{aligned}$$

Par définition d'un code positionnel, on a alors

$$\sum_{j=1}^{c} a_{\ell,j} b^{j-1} = a_\ell.$$

□

Lemme 5.2.3. *(*) Soient* $\mathbf{a}_1, \ldots, \mathbf{a}_m \in N_b^c$. *Soient les uplets* $(h_{\mathbf{k},1}, \ldots, h_{\mathbf{k},c})$, *avec* $\mathbf{k} \in N_b^m$ *les uplets caractéristiques de* $\mathbf{a}_1, \ldots, \mathbf{a}_m$ *et soient les triplets* $(h_\mathbf{k}, b, c)$ *leurs*

codes positionnels. On a

$$\sum_{\mathbf{k}\in N_b^m} h_{\mathbf{k}} = Repeat(1,b,c) = \frac{b^c-1}{b-1}$$

Démonstration (*). D'après le lemme 5.2.1, pour tout $j \in \{1,\ldots,c\}$, il existe un unique $\mathbf{k} \in N_b^m$ tel que $h_{\mathbf{k},j} = 1$. On a

$$\begin{aligned}
\sum_{\mathbf{k}\in N_b^m} h_{\mathbf{k}} &= \sum_{\mathbf{k}\in N_b^m}\sum_{j=1}^{c} h_{\mathbf{k},j} b^{j-1} \\
&= \sum_{j=1}^{c}\sum_{\mathbf{k}\in N_b^m} h_{\mathbf{k},j} b^{j-1} \\
&= \sum_{j=1}^{c} b^{j-1} \\
&= \frac{b^c-1}{b-1}.
\end{aligned}$$

□

Lemme 5.2.4 (*). *Soient* $\mathbf{a}_1,\ldots,\mathbf{a}_m \in N_b^c$. *Soient les uplets* $(h_{\mathbf{k},1},\ldots,h_{\mathbf{k},c})$, *avec* $\mathbf{k} \in N_b^m$, *les uplets caractéristiques de* $\mathbf{a}_1,\ldots,\mathbf{a}_m$ *et soient les triplets* $(h_{\mathbf{k}},b,c)$ *leurs codes positionnels. Alors*

$$\bigwedge_{\mathbf{k}\in N_b^m,\ \mathbf{k}'\in N_b^m,\ \mathbf{k}\neq\mathbf{k}'} Ortnorm_b(h_{\mathbf{k}},h_{\mathbf{k}'},c)$$

est satisfaite.

Démonstration. Démonstration (*) Soient $\mathbf{k} \in N_b^m$ et $\mathbf{k}' \in N_b^m$ tels que $\mathbf{k} \neq \mathbf{k}'$. Il existe donc $j \in \{1,\ldots,m\}$ tel que $k_j \neq k'_j$.
Par définition des uplets caractéristiques, quel que soit $\ell \in \{1,\ldots,c\}$, on a $0 \leq h_{\mathbf{k},\ell} \leq 1$ et $0 \leq h_{\mathbf{k}',\ell} \leq 1$.

- Si $h_{\mathbf{k},\ell} = 1$, alors, par définition des uplets caractéristiques, on a $a_{1,\ell} = k_1,\ldots,a_{m,\ell} = k_m$. Or $a_{j,\ell} = k_j \neq k'_j$. Par définition des uplets caractéristiques, on a donc $h_{\mathbf{k}',j} = 0$.

- Si $h_{\mathbf{k}',\ell} = 1$, alors, par définition des uplets caractéristiques, on a $a_{1,\ell} = k'_1, \ldots, a_{m,\ell} = k'_m$. Or $a_{j,\ell} = k'_j \neq i_k$. Par définition des uplets caractéristiques, on a donc $h_{\mathbf{k},j} = 0$.

La relation $Ortnorm_b(h_{\mathbf{k}}, h_{\mathbf{k}'}, c)$ est donc satisfaite pour tout $\mathbf{k}, \mathbf{k}' \in N_b^m$ tels que $\mathbf{k} \neq \mathbf{k}'$.

□

Lemme 5.2.5 (∗). *Soient* $\mathbf{a}_1, \ldots, \mathbf{a}_m \in N_b^c$ *et* $(a_1, b, c), \ldots, (a_m, b, c)$ *leurs codes positionnels en base* b. *Soient les* $h_{\mathbf{k}}$, *où* $\mathbf{k} \in N_b^m$. *Alors les triplets* $(h_{\mathbf{k}}, b, c)$, *où* $\mathbf{k} \in N_b^m$, *sont les codes positionnels des uplets caractéristiques de* $\mathbf{a}_1, \ldots, \mathbf{a}_m$ *si et seulement si les relations suivantes sont satisfaites :*

$$\bigwedge_{\ell=1}^{m} \sum_{\mathbf{k} \in N_b^m} h_{\mathbf{k}} k_\ell = a_\ell,$$

$$\sum_{\mathbf{k} \in N_b^m} h_{\mathbf{k}} = \frac{b^c - 1}{b - 1}$$

$$\bigwedge_{\mathbf{k} \in N_b^m,\ \mathbf{k}' \in N_b^m,\ \mathbf{k} \neq \mathbf{k}'} Ortnorm_b(h_{\mathbf{k}}, h_{\mathbf{k}'}, c).$$

Démonstration (∗). (⇒) Si les uplets de la forme $(h_{\mathbf{k}}, b, c)$, où $\mathbf{k} \in N_b^m$, sont les codes positionnels des uplets caractéristiques de $\mathbf{a}_1, \ldots, \mathbf{a}_c$, alors soient les uplets $\mathbf{h}_{\mathbf{k}}$, où $\mathbf{k} \in N_b^m$, ces uplets caractéristiques. D'après les lemmes 5.2.2, 5.2.3 et 5.2.4, les conditions suivantes sont satisfaites :

$$\bigwedge_{\ell=1}^{c} \sum_{\mathbf{k} \in N_b^m} h_{\mathbf{k}} k_\ell = a_\ell$$

$$\sum_{\mathbf{k} \in N_b^m} h_{\mathbf{k}} = \frac{b^c - 1}{b - 1}$$

$$\bigwedge_{\mathbf{k} \in N_b^m,\ \mathbf{k}' \in N_b^m,\ \mathbf{k} \neq \mathbf{k}'} Ortnorm_b(h_{\mathbf{k}}, h_{\mathbf{k}'}, c).$$

($\Leftarrow$) Supposons que les conditions

$$\bigwedge_{\ell=1}^{m} \sum_{\mathbf{k}\in N_b^m} h_{\mathbf{k}} k_\ell = a_\ell$$

$$\sum_{\mathbf{k}\in N_b^m} h_{\mathbf{k}} = \frac{b^c - 1}{b - 1}$$

$$\bigwedge_{\mathbf{k}\in N_b^m,\ \mathbf{k}'\in N_b^m,\ \mathbf{k}\neq\mathbf{k}'} Ortnorm_b(h_{\mathbf{k}}, h_{\mathbf{k}'}, c)$$

soient satisfaites pour tout $\ell \in \{1, \ldots, m\}$. Puisque

$$\bigwedge_{\mathbf{k}\in N_b^m,\ \mathbf{k}'\in N_b^m,\ \mathbf{k}\neq\mathbf{k}'} Ortnorm_b(h_{\mathbf{k}}, h_{\mathbf{k}'}, c)$$

est satisfaite, alors pour tout $\mathbf{k} \in N_b^m$, $(h_{\mathbf{k}}, b, c)$ est un code positionnel d'un uplet dont les éléments appartiennent à $\{0, 1\}$. Soit $\mathbf{h_k}$ l'uplet codé. Pour tout $\mathbf{k}, \mathbf{k}' \in N_b^m$ tels que $\mathbf{k} \neq \mathbf{k}'$, on a, pour tout $j \in \{1, \ldots, c\}$, $h_{\mathbf{k},j} = 1 \Rightarrow h_{\mathbf{k}',j} = 0$. Donc, pour tout $j \in \{1, \ldots, c\}$, on a

$$\sum_{\mathbf{k}\in N_b^m} h_{\mathbf{k},j} \leq 1.$$

Par hypothèse, on a également

$$\begin{aligned}
\sum_{\mathbf{k}\in N_b^m} h_{\mathbf{k}} &= Repeat(1, b, c) \\
\sum_{\mathbf{k}\in N_b^m} h_{\mathbf{k}} &= \frac{b^c - 1}{b - 1} \\
\sum_{\mathbf{k}\in N_b^m} h_{\mathbf{k}} &= \sum_{j=1}^{c} b^{j-1} \\
\sum_{\mathbf{k}\in N_b^m} \sum_{j=1}^{c} h_{\mathbf{k},j} b^{j-1} &= \sum_{j=1}^{c} b^{j-1} \\
\sum_{j=1}^{c} \sum_{\mathbf{k}\in N_b^m} (h_{\mathbf{k},j}) b^{j-1} &= \sum_{j=1}^{c} 1.b^{j-1}.
\end{aligned}$$

Puisque $h_{\mathbf{k},j} \leq 1 < b$, par unicité de la représentation d'un nombre en base b, on a

$$\sum_{\mathbf{k} \in N_b^m} (h_{\mathbf{k},j}) = 1$$

pour tout $j \in \{1, \ldots, c\}$. Puisque $h_{\mathbf{k},j} \in \{0, 1\}$, alors, pour tout $j \in \{1, \ldots, c\}$, il existe $\mathbf{x}_j \in N_b^m$ tel que $h_{\mathbf{x}_j,j} = 1$. Soient $\mathbf{x}_1, \ldots, \mathbf{x}_c \in N_b^m$ tels que $h_{\mathbf{x}_1,1} = 1, \ldots, h_{\mathbf{x}_c,c} = 1$. On a alors

$$h_{\mathbf{k},j} = \begin{cases} 1 & \text{si } k_1 = x_{j,1}, \ldots, k_m = x_{j,m} \\ 0 & \text{sinon.} \end{cases}$$

Par définition, pour tout $\ell \in \{1, \ldots, m\}$, on a

$$a_\ell = \sum_{i=j}^{c} a_{\ell,j} b^{j-1}.$$

Par hypothèse, on a, pour tout $\ell \in \{1, \ldots, m\}$

$$\begin{aligned} a_\ell &= \sum_{\mathbf{k} \in N_b^m} h_{\mathbf{k}} k_\ell \\ a_\ell &= \sum_{\mathbf{k} \in N_b^m} \sum_{j=1}^{c} h_{\mathbf{k},j} b^{j-1} k_\ell \\ \sum_{j=1}^{c} a_{\ell,j} b^{j-1} &= \sum_{\mathbf{k} \in N_b^m} \sum_{j=1}^{c} h_{\mathbf{k},j} b^{j-1} k_\ell \\ \sum_{j=1}^{c} a_{\ell,j} b^{j-1} &= \sum_{j=1}^{c} (\sum_{\mathbf{k} \in N_b^m} h_{\mathbf{k},j} k_\ell) b^{j-1}. \end{aligned}$$

Puisque $h_{\mathbf{k},j} \in \{0, 1\}$ et que $k_\ell \in N_b$, alors $h_{\mathbf{k},j} k_\ell \in N_b$. Par unicité de la représentation d'un nombre en base b, on a pour tout $\ell \in \{1, \ldots, m\}$ et tout $j \in \{1, \ldots, c\}$

$$a_{\ell,j} = \sum_{\mathbf{k} \in N_b^c} h_{\mathbf{k},j} k_l.$$

Or, pour tout $j \in \{1, \ldots, c\}$, il existe un unique $\mathbf{x}_j \in N_b^m$ tel que $h_{\mathbf{x}_j,j} = 1$. Pour tout $\mathbf{y} \in N_b^m$ tel que $\mathbf{x}_j \neq \mathbf{y}$, on a $h_{\mathbf{y},j} = 0$. Par conséquent, pour tout $j \in \{1, \ldots, c\}$, on a

$$a_{\ell,j} = \sum_{\mathbf{k} \in N_b^m} h_{\mathbf{k},j} k_\ell = h_{\mathbf{x}_j,j} x_{j,\ell}.$$

Donc, $a_{\ell,j} = x_{j,\ell}$. Nous pouvons alors conclure que

$$h_{\mathbf{k},\ell} = \begin{cases} 1 & \text{si } k_1 = x_{\ell,1} = a_{1,\ell}, \ldots, k_m = x_{\ell,m} = a_{m,\ell} \\ 0 & \text{sinon.} \end{cases}$$

Par définition, les uplets $(h_{\mathbf{k},1}, \ldots, h_{\mathbf{k},c})$, avec $\mathbf{k} \in N_b^m$ sont donc les uplets caractéristiques des uplets $\mathbf{a}_1, \ldots, \mathbf{a}_m$. □

Lemme 5.2.6 (∗). *Soit une fonction* $F : N_b^m \to N_b$. *Soient* $\mathbf{a}_1, \ldots, \mathbf{a}_m \in N_b^c$. *Soient* $(a_1, b, c), \ldots, (a_m, b, c)$ *des codes positionnels de ces uplets. Soient* $\mathbf{h_k} \in \{0,1\}^m$, *où* $\mathbf{k} \in N_b^m$, *les uplets caractéristiques de* $\mathbf{a}_1, \ldots, \mathbf{a}_m$. *Soient* $(h_\mathbf{k}, b, c)$, *où* $\mathbf{k} \in N_b^m$, *des codes positionnels des uplets* $\mathbf{h_k}$. *Alors,*

$$\left(\sum_{\mathbf{k} \in N_b^m} F(\mathbf{k}) h_k, b, c \right)$$

est un code positionnel de l'uplet

$$(F(a_{1,1}, \ldots, a_{m,1}), \ldots, F(a_{1,c}, \ldots, a_{m,c})).$$

Démonstration (∗). On a

$$\begin{aligned} \sum_{\mathbf{k} \in N_b^m} F(\mathbf{k}) h_\mathbf{k} &= \sum_{\mathbf{k} \in N_b^m} F(\mathbf{k}) \sum_{j=1}^{c} h_{\mathbf{k},j} b^{j-1} \\ &= \sum_{\mathbf{k} \in N_b^m} \sum_{j=1}^{c} F(\mathbf{k}) h_{\mathbf{k},j} b^{j-1} \\ &= \sum_{j=1}^{c} \sum_{\mathbf{k} \in N_b^m} F(\mathbf{k}) h_{\mathbf{k},j} b^{j-1}. \end{aligned}$$

D'après le lemme 5.2.1, pour tout $j \in \{1, \ldots, c\}$, il existe un unique $\mathbf{k} \in N_b^m$ tel que $h_{\mathbf{k},j} = 1$. Par définition des uplets caractéristiques, on a $k_1 = a_{1,j}, \ldots, k_m = a_{m,j}$. On a alors

$$\begin{aligned} \sum_{\mathbf{k} \in N_b^c} F(\mathbf{k}) h_\mathbf{k} &= \sum_{j=1}^{c} \sum_{\mathbf{k} \in N_b^m} F(\mathbf{k}) h_{\mathbf{k},j} b^{j-1} \\ &= \sum_{j=1}^{c} F(a_{1,j}, \ldots, a_{m,j}) b^{j-1}. \end{aligned}$$

Puisque $0 \leq F(a_{1,j}, \ldots, a_{m,j}) < b$, alors par définition,

$$(\sum_{j=1}^{c} F(a_{1,j}, \ldots, a_{m,j}) b^{j-1}, b, c)$$

est un code positionnel de l'uplet

$$(F(a_{1,1}, \ldots, a_{m,1}), \ldots, F(a_{1,c}, \ldots, a_{m,c})).$$

□

Lemme 5.2.7 $(*)$. *Soit $b \geq 3$. Soit la fonction $F : N_b^m \to N_b$. Soit la fonction $F[b] : \mathbb{N}^m \times N \to \mathbb{N}$ telle que si $(a_1, b, c), \ldots, (a_m, b, c)$ sont des codes positionnels des uplets $\mathbf{a}_1, \ldots, \mathbf{a}_m \in N_b^c$, alors $(F[b](a), b, c)$ est un code positionnel de l'uplet $(F(a_1), \ldots, F(a_c))$. Alors,*

$$F[b](a_1, \ldots, a_m, c) = a'$$

$$\Leftrightarrow$$

$$\exists h_{(0,\ldots,0)} \ldots \exists h_{(b-1,\ldots,b-1)} \left[\bigwedge_{\ell=1}^{m} \sum_{\mathbf{k} \in N_b^m} h_{\mathbf{k}} k_\ell = a_\ell \wedge \sum_{\mathbf{k} \in N_b^m} h_{\mathbf{k}} = Repeat(1, b, c) \wedge \right.$$

$$\left. \bigwedge_{\mathbf{k} \in N_b^m,\ \mathbf{k}' \in N_b^m,\ \mathbf{k} \neq \mathbf{k}'} Ortnorm_b(h_k, h_{k'}, c) \wedge \sum_{\mathbf{k} \in N_b^m} F(\mathbf{k}) h_{\mathbf{k}} = a' \right].$$

Démonstration $(*)$. $(\Rightarrow)$ Supposons que $F[b](a_1, \ldots, a_m, c) = a'$.
Soient $(h_{\mathbf{k}}, b, c)$ avec $\mathbf{k} \in N_b^c$ les codes positionnels des uplets caractéristiques $\mathbf{h_k}$ des uplets $\mathbf{a}_1, \ldots, \mathbf{a}_m$. D'après les lemmes 5.2.2, 5.2.3 et 5.2.4, les relations

$$\bigwedge_{\ell=1}^{m} \sum_{\mathbf{k} \in N_b^m} h_{\mathbf{k}} k_\ell = a_\ell$$

$$\sum_{\mathbf{k} \in N_b^m} h_{\mathbf{k}} = Repeat(1, b, c)$$

$$\bigwedge_{\mathbf{k} \in N_b^m,\ \mathbf{k}' \in N_b^m,\ \mathbf{k} \neq \mathbf{k}'} Ortnorm_b(h_{\mathbf{k}}, h_{\mathbf{k}'}, c)$$

sont satisfaites.

Donc, d'après le lemme 5.2.6,

$$a' = F[b](a_1, \ldots, a_m, c) = \sum_{\mathbf{k} \in N_b^m} h_{\mathbf{k}} F(\mathbf{k}) h_{\mathbf{k}}.$$

($\Leftarrow$) Supposons qu'il existe des entiers $h_{\mathbf{k}}$, où $\mathbf{k} \in N_b^m$, tels que les relations

$$\bigwedge_{\ell=1}^{m} \sum_{\mathbf{k} \in N_b^m} h_{\mathbf{k}} k_\ell = a_\ell$$

$$\sum_{\mathbf{k} \in N_b^m} h_{\mathbf{k}} = \mathit{Repeat}(1, b, c)$$

$$\bigwedge_{\mathbf{k} \in N_b^m,\ \mathbf{k}' \in N_b^m,\ \mathbf{k} \neq \mathbf{k}'} \mathit{Ortnorm}_b(h_{\mathbf{k}}, h_{\mathbf{k}'}, c)$$

$$\sum_{\mathbf{k} \in N_b^m} F(\mathbf{k}) h_{\mathbf{k}} = a'$$

soient satisfaites. D'après le lemme 5.2.5, les uplets de la forme $(h_{\mathbf{k}}, b, c)$, où $\mathbf{k} \in N_b^m$, sont des codes positionnels des uplets caractéristiques de $\mathbf{a}_1, \ldots, \mathbf{a}_m$. Donc, d'après le lemme 5.2.6, (a', b, c) est un code positionnel de l'uplet

$$(F(a_{1,1}, \ldots, a_{m,1}), \ldots, F(a_{1,c}, \ldots, a_{m,c}))$$

d'où par définition $F[b](a_1, \ldots, a_m, c) = a'$. □

Puisque toutes les conditions utilisées dans le lemme précédent sont diophantiennes, on peut conclure que la fonction $F[b]$ est elle-même diophantienne. Nous possédons donc un nouvel outil diophantien permettant d'appliquer des endofonctions aux éléments d'uplets. Nous nous servirons de cette possibilité au cours du prochain chapitre dans lequel l'encodage des machines de Turing au moyen d'équations diophantiennes réclamera justement l'application d'endofonctions aux éléments d'uplets.

CHAPITRE VI

LE DIXIÈME PROBLÈME DE HILBERT EST INDÉCIDABLE

Rappelons l'énoncé du dixième problème d'Hilbert :

> On donne une équation de Diophante à un nombre quelconque d'inconnues et à coefficients entiers rationnels : on demande de trouver une méthode par laquelle, au moyen d'un nombre fini d'opérations, on pourra distinguer si l'équation est résoluble en nombres entiers rationnels. (Hilbert, 1900)

L'approche de Matiiassevitch (1995) pour achever la preuve de l'indécidabilité du dixième problème de Hilbert est assez différente de celle employée originellement par Davis, Putnam et Robinson (1961). Ces derniers utilisaient des résultats issus de la logique et s'appuyaient notamment sur certaines représentations des ensembles diophantiens appelées formes de Davis. Matiassevitch (1976) a présenté une nouvelle approche utilisant les machines de Turing et permettant de se passer des formes de Davis. Matiiassevitch a par la suite continué à explorer des méthodes alternatives de preuve, avec notamment des résultats employant des machines à registre (Jones et Matiassevitch, 1991), autre modèle de calcul. La méthode exposée par Matiassevitch (1995) est assez simple d'approche : elle ne demande pas au lecteur d'être familiÃarisé avec le formalisme rigoureux de la logique et utilise les machines de Turing, un modèle de calcul facilement définissable et simulable.

Nous allons maintenant aborder la question de la calculabilité qui est le lien entre l'expression intuitive de la nature de la méthode demandée par Hilbert et une formalisation de ce qu'est une telle méthode. Ceci nous amènera à définir dans un premier temps la notion de calculabilité et les enjeux qui s'y rattachent. On décrira

dans un premier temps les machines de Turing. On se servira de ce formalisme pour établir l'équivalence entre les ensembles diophantiens et les ensembles semi-décidables, équivalence constituant l'objectif de ce chapitre.

6.1 La calculabilité

Intuitivement, la question de la calculabilité semble triviale : est calculable tout ce qui peut être calculé, c'est-à-dire obtenu au terme d'un nombre fini d'étapes élémentaires. Ces étapes constituent la procédure ou l'algorithme permettant d'obtenir le résultat. On rencontre rapidement les limites de la formulation intuitive de calculabilité. En effet, il n'est pas exclu de penser que certains résultats ne peuvent pas être calculés. Or, on ne peut pas, au moyen de la seule notion intuitive, prouver qu'aucune procédure n'existe pour réaliser un calcul particulieÃr. On ne peut que se borner à constater qu'on n'a pas encore explicité d'algorithme permettant sa réalisation.

Lorsque Hilbert formule son dixième problème, il n'existe pas encore d'outils permettant de démontrer l'inexistence d'un algorithme.

6.2 Les machines de Turing

Lorsqu'on parle de décrire une procédure pour exécuter une tâche particulière, on veut une méthode mécanique, une suite finie d'instructions élémentaires à exécuter pour arriver au résultat escompté. Par conséquent, une machine pourrait être créée qui accomplirait ces étapes sans aucune intervention extérieure. Introduites par Alan Turing (1937), les machines de Turing constituent un modèle de calcul pouvant être décrit comme une machine physique. Dans le cadre de l'utilisation d'une machine de Turing, on considère que le temps est discret. Une machine de Turing possède une *mémoire* constituée d'un ruban infini divisé en cellules. Par convention, on suppose que le ruban est infini à droite et que la première cellule du ruban est située à l'extrémité gauche. Chaque cellule peut soit contenir un symbole choisi dansun alphabet $A = \{\alpha_1, \ldots, \alpha_w\}$ qui est propre à la machine, soit être vide (pour faciliter l'écriture, on utilisera le symbole

Λ pour désigner le vide). La machine possède une variable appelée *état* dont la valeur appartient à un ensemble fini constituant l'ensemble des états Q possibles de la machine. Un de ces états est dit *initial* : il s'agit de l'état dans lequel se trouve la machine au début de son exécution. Certains de ces états sont dits *finaux* : lorsque la machine entre dans un état final, son fonctionnement est interrompu.

Une machine possède une tête de lecture lui permettant d'interagir avec la mémoire. La tête de lecture observe le contenu d'une cellule particulière et écrit dans cette cellule un des symboles de l'alphabet, remplaçant éventuellement le symbole qui s'y trouvait. La tête peut rester sur place ou se déplacer d'une cellule vers la droite ou vers la gauche. Chacune des opérations (lecture, écriture, déplacement) de la tête de lecture est exécutée à chaque période du temps. Le comportement de la machine est entièrement déterminé par le symbole contenu dans la case observée par la tête de lecture et l'état dans lequel se trouve la machine. Un ensemble de règles permet de préciser les actions qu'effectuera la machine pour chacune des configurations possibles. On considère, pour simplifier l'écriture, que l'ensemble des états Q est $\{q_1, \ldots, q_n\}$ avec q_1 l'état initial. Une instruction peut alors être écrite

$$q_i \alpha_j \Rightarrow \alpha_{A(i,j)} D(i,j) q_{Q(i,j)}$$

où q_i est un état non-final de la machine, α_j est le symbole de l'alphabet $A \cup \Lambda$ (on pose $\alpha_0 = \Lambda$) observé par la tête de lecture, $\alpha_{A(i,j)}$ est le symbole à inscrire dans la cellule observée par la tête de lecture, $q_{Q(i,j)}$ est le prochaine état de la machine et $D(i,j)$ est le déplacement effectué par la tête de lecture. Les valeurs que peut prendre $D(i,j)$ sont L, R et S, indiquant respectivement un déplacement à gauche, à droite et aucun déplacement.

Au début de l'exécution, un segment initial de taille fini de la mémoire d'une machine de Turing est formé de cases contenant un symbole de l'alphabet A. Ce segment constitue l'entrée passée à la machine de Turing. La machine peut adopter deux comportements en fonction de l'entrée. Soit la machine atteint un état final au bout d'un nombre fini d'étapes, soit aucun état final ne sera jamais atteint. Si la machine

entre dans un état final elle s'arrête et on dit que l'entrée est *reconnue* par la machine. Une machine de Turing est donc capable de reconnaître un langage dont les mots sont formés à partir des symboles de l'alphabet A. Si la machine s'arrête, alors le mot passé en entrée appartient au langage reconnu par la machine. Un tel langage est dit *récursivement énumérable* ou encore *semi-décidable*. On peut également considérer la machine de Turing comme un calculateur. Si on se munit d'une interprétation des mots qu'on peut former au moyen de l'alphabet, alors on peut observer un résultat sur la mémoire de la machine lorsque l'exécution s'arrête. C'est le résultat du calcul effectué par la machine de Turing à partir de l'entrée initiale. L'ensemble des valeurs pouvant être calculées de la sorte par une machine de Turing est dit *partiellement récursif*. De telles machines sont, si on considère valide la thèse de Church (voir section 6.3), capables de calculer tout ce qui est calculable. En d'autres termes, s'il n'existe pas de machine capable d'effectuer une procédure, alors il n'est pas possible de décrire une telle procédure en termes d'algorithme.

Il existe plusieurs descriptions des machines de Turing, équivalentes entre elles : machines à mémoire infinie dans les deux sens, machines multi-bandes, machines ne possédant qu'un nombre restreint de symboles d'alphabet. Le lecteur intéressé pourra se référer à Hopcroft, Motwani et Ullman (2006) pour une définition plus formelle des machines des Turing et des démonstrations d'équivalence entre les modèles.

Un résultat remarquable issu de la recherche autour du dixième problème de Hilbert est l'identité entre les ensembles semi-décidables et les ensembles diophantiens. Ceci signifie que les ensembles diophantiens constituent un modèle de calcul équivalent aux machines de Turing alors qu'a priori ces équations n'ont pas vocation à distinguer ce qui est calculable de ce qui ne l'est pas. Nous allons montrer cette équivalence, puis nous en déduirons l'indécidabilité du dixième problème de Hilbert.

6.3 La thèse de Church

La théorie de la calculabilité est une tentative de modélisation de la notion intuitive de faisabilité. En d'autres termes, elle fournit un critère permettant de décider s'il existe une procédure, un algorithme permettant de répondre à une question, de fournir le résultat d'un calcul. Il n'existe pas de preuve absolue que la notion intuitive de calculabilité et sa formalisation mathématique sont parfaitement équivalentes. La thèse de Church affirme que ce que peuvent calculer les machines de Turing coïncide exactement avec ce qui est intuitivement considéré comme calculable. Plusieurs indices fondent cette hypothèse. En premier lieu, il n'a pas encore été découvert de modèle plus puissant que les machines de Turing. En d'autres termes, il n'existe pas, pour l'instant, de modèles permettant d'effectuer des calculs que la machine de Turing est incapable de faire. De plus, aucune procédure n'a encore été décrite qui ne soit pas simulable par une machine de Turing. Ceci plaide en faveur de la thèse de Church. Aussi, arriver à démontrer qu'un calcul particulier ne peut être réalisé équivaut, si on accepte cette thèse, à prouver qu'il n'existe aucune machine de Turing capable d'effectuer ce calcul.

6.4 Convention de représentation

Les machines de Turing manipulent des symboles qui sont en eux-même dépourvus de sens. C'est à l'utilisateur de la machine de se munir d'un système d'interprétation des symboles qui lui permet de tirer parti des calculs effectués par une machine de Turing. Nous présenterons dans cette section les conventions adoptées pour représenter les objets mathématiques que nous allons fournir aux machines créées ainsi que les spécificités de ces machines. L'alphabet des machines utilisées dans ce chapitre est l'ensemble $\{\Lambda, \star, 0, 1, 2, 3\}$.

6.4.1 Symboles spéciaux

Nous commencerons par établir une convention commune à toutes les machines que nous emploierons. Cette convention permet d'éviter des mouvements invalides de

la tête de lecture en permettant de détecter l'emplacement de la première cellule de la mémoire. Par convention, on exige que le symbole contenu dans la première cellule de la mémoire d'une machine de Turing soit en tout temps le symbole $\star$. Ceci permet d'écrire des instructions qui ne provoqueront pas un déplacement vers la gauche de la tête de lecture alors qu'elle se trouve sur la première cellule. Il suffit pour cela d'interdire les mouvements vers la gauche pour toute instruction concernant le symbole $\star$ et d'imposer la réécriture de ce symbole. Il convient également d'interdire la possibilité d'écrire le symbole $\star$ ailleurs que dans la première case de la mémoire.

Hormis l'entrée passée initialement à une machine de Turing, la mémoire est composée de cases vides (c'est-à-dire contenant le symbole Λ). Si le format des instructions d'une machine de Turing permet de traiter le cas d'une case vide, il est en revanche impossible de ne rien inscrire dans cette case durant l'exécution d'une instruction. Ceci implique que toute case observée à un moment donné contiendra par la suite un symbole. De plus, il n'est pas possible qu'une case contenant un symbole redevienne vide au cours de l'exécution de la machine. Il pourrait toutefois s'avérer utile d'effacer le contenu de cases de la mémoire, par exemple pour supprimer une donnée. Pour contourner cette difficulté, on introduit un nouveau symbole, λ, qui sera considéré comme vide. Ainsi, toute instruction s'appliquant à une case vide s'appliquera également à λ. De même, lors de l'interprétation de l'état de la mémoire, on considérera identiques une case vide et une case contenant λ. On exige qu'à tout moment de l'exécution d'une machine de Turing, il n'existe aucune case vide ou contenant le symbole λ située à gauche d'une case contenant un symbole différent de λ. En d'autres termes, le segment initial de la mémoire est continu, sans « trous », toutes les cases que nous considérons vides sont situées après les cases non vides sur la bande.

6.4.2 États initial et finaux

Il est possible de tirer des conclusions non seulement de l'état de la mémoire d'une machine de Turing lors de son arrêt mais également de l'état dans lequel elle s'arrête. En effet, les machines de Turing peuvent posséder plusieurs états finaux, chacun porteur

d'un sens particulier.

Par convention, toute machine de Turing créée pour les besoins de ce chapitre aura pour état initial q_1. De plus, elle possèdera deux états finaux, q_2 et q_3, qui seront respectivement interprétés comme donnant les réponses « OUI »(ou « VRAI ») et « NON »(ou « FAUX »). Si on considère qu'une machine est en quelque sorte la formalisation d'une question et que l'entrée est le sujet de cette question, alors on peut considérer l'état final de la machine au terme de l'exécution comme la réponse à la question concernant l'entrée.

6.4.3 Nombres et uplets

On sera amené, dans le cadre de ce chapitre, à utiliser des machines de Turing pour reconnaître des uplets. Il faut donc adopter une convention de représentation des uplets dans la mémoire de ces machines.

Définition 6.4.1. *On appelle* représentation canonique *d'un nombre n dans la mémoire d'une machine de Turing un mot débutant par le symbole* 0 *suivi de n symboles* 1.

Il ne s'agit pas vraiment de la représentation unaire du nombre, à cause de la présence du 0 au début du mot. Cet ajout sert justement à représenter les uplets facilement, le 0 joue un rôle de délimiteur entre les nombres. Notons que le nombre 0 est représenté par le mot constitué d'un unique symbole 0.

Définition 6.4.2. *On appelle* représentation canonique *d'un uplet $(n_1, \ldots, n_k)$ le mot obtenu en concaténant les représentations canoniques des nombres $n_1, \ldots, n_k$ les unes après les autres par ordre croissant d'indices.*

Il ne serait pas possible de retrouver les valeurs des membres de l'uplet représenté sans le délimiteur 0 dont l'emploi trouve ici son sens.

6.4.4 Compositions de machines de Turing

Un de nos objectifs est ici de construire, à partir d'une équation diophantienne paramétrée quelconque, une machine de Turing qui atteint un état final si et seulement si l'entrée qui lui est fournie est la représentation d'un uplet de paramètres pour lesquels l'équation possède une solution. Les instructions d'une telle machine sont probablement nombreuses et leur rôle risque d'être complexe. Pour faciliter la description de cette machine, Matiiassevitch (1995), propose d'utiliser des méthodes permettant de composer des machines de Turing entre elles, c'est-à-dire de former une machine de Turing M à partir de machines préexistantes M_1 et M_2. Ainsi, on pourra décrire des machines de plus en plus complexes en composant des machines de base, dont les jeux d'instructions sont faciles à décrire. Cette démarche sera intuitivement plus compréhensible, se rapprochant de la programmation.

Nous utiliserons deux méthodes de composition différentes.
La première méthode est la suivante :

- Dans toutes les instructions de la machine M_1, on remplace l'état final q_2 par q_{v+1} où v est le nombre d'états de M_1.
- Dans toutes les instructions de la machine M_2, on remplace chaque état non final q_i par l'état q_{v+i}.
- L'ensemble des instructions de la nouvelle machine M est constitué des ensembles d'instructions modifiés tels que décrits ci-dessus.

Concrètement, cette composition fournit une machine dont le comportement est le suivant :

- La machine M se comporte dans un premier temps comme la machine M_1.
  - Si la machine M_1 n'atteint jamais un état final, alors la machine M n'atteint jamais un état final.
  - Si la machine M_1 atteint l'état q_2, alors, à cause de la modification des instructions, la machine M entre dans l'état q_{v+1}. Cet état se trouve être l'état

initial de la machine M_2. Puisque les modifications apportées à M_2 constituent juste un nouvel étiquetage des états, alors M se comporte maintenant comme M_2.

▷ Si la machine M_2 atteint un état final avec pour entrée la mémoire de M au moment où l'état de M devient q_{v+1}, alors la machine M s'arrête sur cet état final.

▷ Si la machine M_2 ne s'arrête jamais lorsque son entrée est la mémoire de M au moment où l'état de M devient q_{v+1}, alors M ne s'arrête jamais.

- Si la machine M_1 atteint l'état q_3, alors M s'arrête dans l'état final q_3.

On peut voir que l'exécution de la partie issue de M_2 est conditionnelle à une réponse « VRAI »de M_1 pour l'entrée. Si la réponse de M_1 est « FAUX », alors la réponse de M sera « FAUX ». Pour décrire une machine obtenue par cette composition, on utilisera les notations

$$M_1; M_2,$$

$$M_1 \textbf{ and } M_2,$$

ou

$$\textbf{if } M_1 \textbf{ then } M_2.$$

La seconde méthode de composition est la suivante :

- Dans toutes les instructions de la machine M_1, on remplace l'état q_2 par q_{v+1}, où v est le nombre d'états de la machine M_1.
- Dans toutes les instructions de la machine M_1, on remplace l'état q_3 par q_2.
- Dans toutes les instructions de la machine M_2, on remplace l'état q_2 par q_1.
- Dans toutes les instructions de la machine M_2, on remplace chaque état q_i par q_{v+i}, où $i \neq 2$, $i \neq 3$ et v est le nombre d'états de la machine M_1.
- L'ensemble des instructions de la machine M est constitué de l'ensemble des instructions de M_1 et M_2 modifiées.

Cette composition fournit une machine dont le comportement est le suivant :

- La machine M se comporte dans un premier temps comme la machine M_1.
  - Si la machine M_1 n'atteint jamais un état final, alors la machine M n'atteint jamais un état final.
  - Si la machine M_1 atteint l'état q_2, alors, à cause de la modification des instructions, la machine M entre dans l'état q_{v+1}. Cet état est l'état initial de la machine M_2. Puisque les modifications apportées à M_2 constituent juste un nouvel étiquetage des comportements, alors M se comporte maintenant comme M_2.
    - ▷ Si la machine M_2 s'arrête dans l'état q_2, alors la machine M revient à l'état q_1. Elle se comporte alors de nouveau comme M_1.
    - ▷ Si la machine M_2 s'arrête dans l'état q_3, alors la machine M s'arrête dans l'état q_3.
  - Si la machine M_1 atteint l'état final q_3, alors la machine M s'arrête dans l'état q_2.

Cette composition forme une boucle conditionnelle. En effet, tant que la machine M_1 répond « VRAI », alors la machine M_2 s'exécute. Si l'exécution de la machine M_2 est normale (c'est-à-dire qu'elle s'arrête dans l'état q_2), alors on relance la machine M_1, et ainsi de suite. Si la machine M_1 répond « FAUX », alors la machine M s'arrête dans l'état q_2. On peut remarquer une différence fondamentale avec la composition précédente, qui est qu'une réponse négative à la condition de la boucle n'entraîne pas une réponse négative de la machine M. Cependant, si la machine M_2 s'arrête dans l'état q_3, alors la machine M s'arrête dans l'état q_3. Pour décrire une machine obtenue par cette composition, on utilisera la notation

while M_1 **do** M_2 **od.**

Bien que la notation de ces deux compositions ressemble à celle d'un langage informatique, il est bon de remarquer qu'il existe ici quelques subtilités que ne véhiculent pas les concepts d'instructions et de boucles conditionnelles en général. Ainsi, tel

qu'évoqué précédemment, il faut noter que dans le cas d'une machine **if** M_1 **then** M_2, une réponse négative à la condition entraîne une réponse négative de la machine au complet. Au contraire, dans le cas d'une machine **while** M_1 **do** M_2 **od**, une réponse négative ne peut être obtenue que si la machine M_2 donne une réponse négative. Ainsi, si on désire créer une machine qui teste une condition et qui dans tous les cas donne une réponse positive, il faut employer une composition **while** plutôt qu'une composition **if** ... **then**, ce qui ne constitue pas le rôle usuellement associé à une boucle.

6.5 Machines utiles au test de toutes les solutions d'une équation

La démonstration du caractère semi-décidable des équations diophantiennes fait intervenir un certain nombre de machines de Turing qui sont utilisées par Matiiasse-vitch (1995) pour créer une machine capable de tester toutes les solutions possibles de l'équation considérée. Le tableau suivant présente brièvement chacune de ces machines. La partie subséquente n'est pas indispensable à la compréhension et ne fait que fournir la définition de ces machines tout en détaillant plus avant leurs caractéristiques.

Tableau 6.1 Machines de Turing utilisées

Nom	Entrée	Action
RIGHT	Quelconque	Déplace la tête de lecture d'une case vers la droite.
LEFT	Quelconque	Déplace la tête de lecture d'une case vers la gauche.
WRITE(n)	Quelconque	Écrit le symbole n dans la case observée par la tête de lecture.
READ(n)	Quelconque	S'arrête dans l'état q_2 si le symbole observé par la tête de lecture est n. S'arrête dans l'état q_3 sinon.
STOP	Quelconque	S'arrête dans l'état q_3.
NEVERSTOP	Quelconque	S'exécute indéfiniment.
READNOT(n)	Quelconque	S'arrête dans l'état q_3 si le symbole observé par la tête de lecture est n. S'arrête dans l'état q_2 sinon.
STAR	Quelconque	Positionne la tête de lecture sur la première case de la mémoire.
VACANT	Quelconque	Positionne la tête de lecture sur la première case vide de la mémoire.
JUMP	Représentation d'un uplet $(a_1, \ldots, a_m)$	Déplace la tête de lecture de sa position actuelle vers la première case à droite débutant la représentation d'un nombre.

Tableau 6.1 Machines de Turing utilisées (suite)

FIND(n)	Représentation d'un uplet $(a_1, \ldots, a_m)$	Déplace la tête de lecture vers la case débutant la représentation de a_n.
LAST	Représentation d'un uplet $(a_1, \ldots, a_m)$	Déplace la tête de lecture vers la case débutant la représentation de a_m.
NEW	Représentation d'un uplet $(a_1, \ldots, a_m)$	Transforme la représentation de $(a_1, \ldots, a_m)$ en la représentation de $(a_1, \ldots, a_m, 0)$.
INC	Représentation d'un uplet $(a_1, \ldots, a_m)$	Transforme la représentation de $(a_1, \ldots, a_m)$ en la représentation de $(a_1, \ldots, a_m + 1)$.
DEC	Représentation d'un uplet $(a_1, \ldots, a_m)$	Si $a_m > 0$, transforme la représentation de $(a_1, \ldots, a_m)$ en la représentation de $(a_1, \ldots, a_m - 1)$. Sinon ne fait rien.
DELETE	Représentation d'un uplet $(a_1, \ldots, a_m)$	Transforme la représentation de $(a_1, \ldots, a_m)$ en la représentation de $(a_1, \ldots, a_{m-1})$.
MARK(n)	Représentation d'un uplet $(a_1, \ldots, a_m)$	Remplace chaque symbole 1 rencontré à droite de la tête de lecture par le symbole n jusqu'à l'observation d'un symbole différent de 1.
THEREIS(n)	Quelconque	S'arrête dans l'état q_2 si un symbole n est présent dans la mémoire. S'arrête dans l'état q_3 sinon.
THEREWAS(n)	Quelconque	Agit comme THEREIS(n) mais remplace le symbole n, s'il est rencontré, par le symbole 1
RESTORE	Quelconque	Remplace toutes les occurrences des symboles 2 et 3 par le symbole 1.

Tableau 6.1 Machines de Turing utilisées (fin)

APPEND(k)	Représentation d'un uplet $(a_1, \ldots, a_m)$	Transforme la représentation de $(a_1, \ldots, a_m)$ en la représentation de $(a_1, \ldots, a_m + a_k)$.
COPY(k)	Représentation d'un uplet $(a_1, \ldots, a_m)$	Transforme la représentation de $(a_1, \ldots, a_m)$ en la représentation de $(a_1, \ldots, a_m, a_k)$.
ADD(i, j)	Représentation d'un uplet $(a_1, \ldots, a_m)$	Transforme la représentation de $(a_1, \ldots, a_m)$ en la représentation de $(a_1, \ldots, a_m, a_i + a_j)$.
MULT(i, j)	Représentation d'un uplet $(a_1, \ldots, a_m)$	Transforme la représentation de $(a_1, \ldots, a_m)$ en la représentation de $(a_1, \ldots, a_m, a_i a_j)$.
NOTGREATER(i, j)	Représentation d'un uplet $(a_1, \ldots, a_m)$	S'arrête dans l'état q_2 si $a_i \leq a_j$. S'arrête dans l'état q_3 sinon.
EQUAL(i, j)	Représentation d'un uplet $(a_1, \ldots, a_m)$	S'arrête dans l'état q_2 si $a_i = a_j$. S'arrête dans l'état q_3 sinon.
NOTEQUAL(i, j)	Représentation d'un uplet $(a_1, \ldots, a_m)$	S'arrête dans l'état q_2 si $a_i \neq a_j$. S'arrête dans l'état q_3 sinon.
NEXT	Représentation d'un uplet $(a_1, \ldots, a_m)$	Transforme la représentation de $(a_1, \ldots, a_m)$ en la représentation de $(a_1, \ldots, a_{m-2}, b, c)$ où (b, c) est le couple suivant (a_{m-1}, a_m) dans la numérotation de Cantor.
DECODE	Représentation d'un uplet $(a_1, \ldots, a_m)$	Transforme la représentation de $(a_1, \ldots, a_m)$ en la représentation de $(a_1, \ldots, a_m, b, c)$ où (b, c) est le couple dont le numéro de Cantor est a_m.

On va maintenant décrire plus en détail chacune de ces machines.

- LEFT et RIGHT

  Les machines RIGHT et LEFT déplacent la tête de lecture respectivement d'une case vers la droite et d'une case vers la gauche. Toutefois, si la tête de lecture observe la première cellule de la mémoire, le déplacement vers la gauche n'aura pas lieu. De même, si la tête de lecture observe une cellule vide ou contenant le symbole λ, le déplacement vers la droite ne se fera pas. La machine LEFT est décrite par les instructions

$$\begin{aligned} q_1\star &\Rightarrow \star S q_2 \\ q_1 0 &\Rightarrow 0 L q_2 \\ q_1 1 &\Rightarrow 1 L q_2 \\ q_1 2 &\Rightarrow 2 L q_2 \\ q_1 3 &\Rightarrow 3 L q_2 \\ q_1 \lambda &\Rightarrow \lambda L q_2 \\ q_1 \Lambda &\Rightarrow \lambda L q_2. \end{aligned}$$

  La machine RIGHT est décrite par les instructions

$$\begin{aligned} q_1\star &\Rightarrow \star R q_2 \\ q_1 0 &\Rightarrow 0 R q_2 \\ q_1 1 &\Rightarrow 1 R q_2 \\ q_1 2 &\Rightarrow 2 R q_2 \\ q_1 3 &\Rightarrow 3 R q_2 \\ q_1 \lambda &\Rightarrow \lambda S q_2 \\ q_1 \Lambda &\Rightarrow \lambda S q_2. \end{aligned}$$

  Quelle que soit l'entrée passée à la machine, l'exécution s'arrête forcément dans l'état q_2 après la première instruction. De plus, aucune modification de la mémoire n'est effectuée, exception faite du cas où la cellule observée est vide, auquel cas

on écrit le symbole λ dans cette cellule. Dans la mesure où aucune cellule vide et aucune cellule contenant λ n'est située à gauche d'une cellule non vide contenant un symbole différent de λ dans l'entrée, alors la mémoire obtenue en fin d'exécution ne présente aucune cellule contenant un symbole λ à gauche d'une cellule ne contenant pas λ.

- WRITE(n)

  Une machine WRITE(n) écrit le symbole $n \in \{0, 1, 2, 3, \lambda\}$ dans la cellule observée par la tête de lecture sauf si cette cellule contient le symbole $\star$, c'est-à-dire si la tête observe la première cellule de la mémoire. Le jeu d'instructions de la machine WRITE(0) est

$$\begin{aligned} q_1\star &\Rightarrow \star S q_2 \\ q_1 0 &\Rightarrow 0 S q_2 \\ q_1 1 &\Rightarrow 0 S q_2 \\ q_1 2 &\Rightarrow 0 S q_2 \\ q_1 3 &\Rightarrow 0 S q_2 \\ q_1 \lambda &\Rightarrow 0 S q_2 \\ q_1 \Lambda &\Rightarrow 0 S q_2. \end{aligned}$$

  Les machines WRITE(1), WRITE(2), WRITE(3) et WRITE(λ) possèdent des jeux d'instructions similaires.

Quelle que soit l'entrée passée à une de ces machines, l'exécution s'arrête forcément dans l'état q_2 après la première instruction. Lors de l'utilisation des machines WRITE(0), WRITE(1), WRITE(2) et $WRITE(3)$, il faut s'assurer que l'écriture ne se fait pas sur une case située à droite d'une case vide ou contenant le symbole λ. Lors de l'utilisation de la machine WRITE(λ), il faut s'assurer que l'écriture ne se fait pas sur une case située à gauche d'une case non vide et contenant un symbole différent de λ. Ces vérifications sont nécessaires pour maintenir les conventions adoptées précédemment concernant la forme des données dans la mémoire.

- READ(n)

  Une machine READ(n) indique si le symbole contenu dans la cellule observée par la tête de lecture est n, avec $n \in \{\star, 0, 1, 2, 3, \lambda\}$. La machine s'arrête donc dans l'état q_2 si le symbole observé correspond à celui reconnu par la machine et dans l'état q_3 sinon. Le comportement de READ(λ) est un peu différent dans la mesure où λ est un symbole à interpréter comme le vide. La machine READ(λ) s'arrête donc dans l'état q_2 si la cellule observée par la tête de lecture est vide ou contient λ. Le jeu d'instruction de la machine READ($\star$) est

$$
\begin{aligned}
q_1\star &\Rightarrow \star S q_2 \\
q_1 0 &\Rightarrow 0 S q_3 \\
q_1 1 &\Rightarrow 1 S q_3 \\
q_1 2 &\Rightarrow 2 S q_3 \\
q_1 3 &\Rightarrow 3 S q_3 \\
q_1\lambda &\Rightarrow \lambda S q_3 \\
q_1\Lambda &\Rightarrow \lambda S q_3.
\end{aligned}
$$

  Les machines READ(0), READ(1), READ(2) et READ(3) possèdent des jeux d'instructions similaires. Aucune modification de la mémoire n'est faite par ces machines. Dans la mesure où aucune cellule vide et aucune cellule contenant λ n'est située à gauche d'une cellule contenant un symbole différent de λ dans l'entrée, alors la mémoire obtenue en fin d'exécution ne présente aucune cellule contenant un symbole λ à gauche d'une cellule ne contenant pas λ.

Le jeu d'instruction de la machine READ(λ) est

$$\begin{aligned} q_1\star &\Rightarrow \star S q_3 \\ q_1 0 &\Rightarrow 0 S q_3 \\ q_1 1 &\Rightarrow 1 S q_3 \\ q_1 2 &\Rightarrow 2 S q_3 \\ q_1 3 &\Rightarrow 3 S q_3 \\ q_1 \lambda &\Rightarrow \lambda S q_2 \\ q_1 \Lambda &\Rightarrow \lambda S q_2. \end{aligned}$$

Aucune modification de la mémoire n'est faite par cette machine. Dans la mesure où aucune cellule vide et aucune cellule contenant λ n'est située à gauche d'une cellule contenant un symbole différent de λ dans l'entréc, alors la mémoire obtenue en fin d'exécution ne présente aucune cellule contenant un symbole λ à gauche d'une cellule ne contenant pas λ.

- STOP

  La machine STOP passe dans l'état final q_3 directement sans effectuer aucune modification sur la position de la tête ou l'état de la mémoire, excepté si la cellule observée est vide, auquel cas le symbole λ sera écrit dans cette cellule. Ses instructions sont

$$\begin{aligned} q_1\star &\Rightarrow \star S q_3 \\ q_1 0 &\Rightarrow 0 S q_3 \\ q_1 1 &\Rightarrow 1 S q_3 \\ q_1 2 &\Rightarrow 2 S q_3 \\ q_1 3 &\Rightarrow 3 S q_3 \\ q_1 \lambda &\Rightarrow \lambda S q_3 \\ q_1 \Lambda &\Rightarrow \lambda S q_3. \end{aligned}$$

Aucune modification de la mémoire n'est faite par cette machine. Dans la mesure où aucune cellule vide et aucune cellule contenant λ n'est située à gauche d'une cellule contenant un symbole différent de λ dans l'entrée, alors la mémoire obtenue en fin d'exécution ne présente aucune cellule contenant un symbole λ à gauche d'une cellule ne contenant pas λ.

- NEVERSTOP

  La machine NEVERSTOP est conçue pour ne jamais s'arrêter ; elle ne fait aucune modification de la position de la tête de lecture ou de l'état de la mémoire. Ses instructions sont

$$\begin{aligned} q_1\star &\Rightarrow \star S q_1 \\ q_1 0 &\Rightarrow 0 S q_1 \\ q_1 1 &\Rightarrow 1 S q_1 \\ q_1 2 &\Rightarrow 2 S q_1 \\ q_1 3 &\Rightarrow 3 S q_1 \\ q_1 \lambda &\Rightarrow \lambda S q_1 \\ q_1 \Lambda &\Rightarrow \lambda S q_1 . \end{aligned}$$

  Nous sommes désormais en possession de toutes les machines de Turing de base dont nous avons besoin pour la suite. À partir de maintenant, les machines que nous décrirons seront obtenues uniquement par composition.

- READNOT(n)

  Les machines READNOT(n) avec $n \in \{\star, 0, 1, 2, 3, \lambda\}$ indiquent si le symbole contenu dans la cellule observée par la tête de lecture n'est pas n. La machine s'arrête dans l'état q_2 si le symbole observé est différent du symbole attendu et dans l'état q_3 sinon. La machine est composée comme suit :

$$\text{READNOT}(n) = \textbf{while } \text{READ}(n) \textbf{ do } \text{STOP } \textbf{od}.$$

  L'utilisation de la composition **while** est un peu contre-intuitive. En effet, il n'est

pas ici question d'exécuter des instructions en boucle, mais d'effectuer une unique observation du contenu de la cellule observée. On pourrait donc penser que la composition **if** ... **then** serait plus appropriée. Cependant, une machine composée au moyen de **if** ... **then** s'arrête dans l'état q_3 si la machine servant de condition donne la réponse « FAUX ». Or ici, si le symbole observé est différent du symbole attendu, la machine READ(n) va donner la réponse « FAUX ». Si on utilisait **if** ... **then**, notre nouvelle machine répondrait alors « FAUX », ce qui n'est pas le comportement voulu. Dans le cas de la composition **while**, la réponse « FAUX »de READ(n) provoque la réponse « VRAI » de la machine composée. Si la réponse de READ(n) est « VRAI », alors on exécute les instructions de la machine STOP qui s'arrête toujours dans l'état q_3. La réponse sera alors « FAUX ». Donc, la machine READNOT(n) obtenue par la composition **while** répond « VRAI » lorsque le symbole contenu dans la cellule observée par la tête de lecture est différent du symbole attendu et « FAUX » sinon. De plus, aucune modification de la mémoire n'est faite par les machines utilisées dans la composition. Dans la mesure où aucune cellule vide et aucune cellule contenant λ n'est située à gauche d'une cellule contenant un symbole différent de λ dans l'entrée, alors la mémoire obtenue en fin d'exécution ne présente aucune cellule contenant un symbole λ à gauche d'une cellule ne contenant pas λ.

- STAR

  La machine STAR positionne la tête de lecture sur la première cellule de la mémoire.

  $$\text{STAR} \;=\; \textbf{while}\ \text{READNOT}(\star)\ \textbf{do}\ \text{LEFT}\ \textbf{od}.$$

  La mémoire n'est pas modifiée par les machines utilisées dans la composition.

- VACANT

  La machine VACANT positionne la tête de lecture sur la première cellule vide ou contenant le symbole λ de la mémoire.

  $$\text{VACANT} \;=\; \text{STAR};\ \textbf{while}\ \text{READNOT}(\lambda)\ \textbf{do}\ \text{RIGHT}\ \textbf{od}.$$

  La mémoire n'est pas modifiée par les machines utilisées dans la composition.

- JUMP

  La machine JUMP positionne la tête de lecture sur la prochaine cellule à sa droite contenant le symbole 0. Si aucune cellule à droite ne contient 0, alors la machine ne s'arrête jamais. Si la mémoire représente un uplet, la machine positionne donc la tête de lecture sur le début de la représentation la plus proche à la droite de la position initiale de la tête de lecture.

  $$\text{JUMP} \;=\; \textbf{while}\ \text{READNOT}(0)\ \textbf{do}\ \text{RIGHT}\ \textbf{od}.$$

  La mémoire n'est pas modifiée par les machines utilisées dans la composition.

- FIND(n)

  La machine FIND(n) positionne la tête de lecture sur la cellule contenant le symbole 0 débutant la représentation du n-ième élément de l'uplet représenté en mémoire. Si cet uplet contient moins de n éléments, alors la machine ne s'arrête jamais.

  $$\text{FIND}(1) = \text{STAR}\,;\ \text{JUMP}$$

  $$\text{FIND}(k+1) = \text{FIND}(k)\,;\ \text{RIGHT}\,;\ \text{JUMP}.$$

  La mémoire n'est pas modifiée par les machines utilisées dans la composition.

- LAST

  La machine LAST positionne la tête de lecture sur la cellule contenant le symbole 0 débutant la représentation du dernier élément de l'uplet représenté en mémoire.

  $$\text{LAST} = \text{VACANT}\,;\ \textbf{while}\ \text{READNOT}(0)\ \textbf{do}\ \text{LEFT}\ \textbf{od}.$$

  La mémoire n'est pas modifiée par les machines utilisées dans la composition.

- NEW

  La machine NEW écrit le symbole 0 dans la cellule à droite de la dernière cellule contenant un symbole 1. Considérant nos conventions de représentation, cela signifie que si la mémoire représentait un uplet $(a_1, \ldots, a_m)$ à l'état initial, alors elle représentera l'uplet $(a_1, \ldots, a_m, 0)$ lors de l'arrêt de la machine.

  $$\text{NEW} = \text{VACANT}\,;\ \text{WRITE}(0).$$

La machine VACANT positionne la tête de lecture sur la case immédiatement à droite de la dernière case contenant un symbole différent de λ. Par convention, il n'existe pas de case située plus à droite dans la mémoire qui contienne un symbole différent de λ. Par conséquent, écrire 0 dans cette case respecte les conventions adoptées au sujet de la forme de la mémoire.

- INC et DEC

  La machine INC modifie la mémoire en écrivant un symbole 1 dans la cellule à droite de la dernière cellule contenant un symbole différent de λ. Considérant nos conventions de représentation, cela signifie que si la mémoire représentait un uplet $(a_1, \ldots, a_n)$ à l'état initial, alors elle représentera l'uplet $(a_1, \ldots, a_n + 1)$ lors de l'arrêt de la machine.

$$\text{INC} = \text{VACANT}; \text{WRITE}(1).$$

  La machine VACANT positionne la tête de lecture sur la case immédiatement à droite de la dernière case contenant un symbole différent de λ. Par convention, il n'existe pas de case située plus à droite dans la mémoire qui contienne un symbole différent de λ. Par conséquent, écrire 1 dans cette case respecte les conventions adoptées au sujet de la forme de la mémoire.

  La machine DEC modifie la mémoire en remplaçant par λ le symbole contenu dans la dernière case de la mémoire contenant un symbole différent de λ si ce symbole est différent de 0. Considérant nos conventions de représentation, cela signifie que si la mémoire représentait un uplet $(a_1, \ldots, a_n)$ avec $a_n \neq 0$ à l'état initial, alors elle représentera l'uplet $(a_1, \ldots, a_n - 1)$ lors de l'arrêt de la machine. Dans le cas particulier où $a_n = 0$, la mémoire représentera $(a_1, \ldots, a_n)$ à la fin de l'exécution mais l'état final sera alors q_3, indiquant que la soustraction n'a pas eu lieu.

$$\text{DEC} = \text{VACANT}; \text{LEFT}; \textbf{if } \text{READ}(1) \textbf{ then } \text{WRITE}(\lambda).$$

  La machine VACANT positionne la tête de lecture sur la case immédiatement à droite de la dernière case contenant un symbole différent de λ. Par convention, il n'existe pas de case située plus à droite dans la mémoire qui contienne un symbole

différent de λ. La machine LEFT décale ensuite à gauche la tête de lecture, la positionnant sur la dernière case de la mémoire contenant un symbole différent de λ. L'écriture du symbole λ dans cette case respecte donc les conventions adoptées au sujet de la forme de la mémoire.

- DELETE

  La machine DELETE modifie la mémoire en supprimant la suite de symboles $01\ldots1$ la plus à droite de la mémoire. Considérant nos conventions de représentation, cela signifie que si la mémoire représentait un uplet $(a_1,\ldots,a_n)$ à l'état initial, alors elle représentera l'uplet $(a_1,\ldots,a_{n-1})$ lors de l'arrêt de la machine.

  DELETE = VACANT;
  while READNOT(0) **do** WRITE(λ) ; LEFT **od** ;
  WRITE(λ).

  La machine VACANT positionne la tête de lecture sur la case immédiatement à droite de la dernière case contenant un symbole différent de λ. Par convention, il n'existe pas de case située plus à droite dans la mémoire qui contienne un symbole différent de λ. Le caractère λ est par la suite écrit dans toutes les cases situées entre cette position et la plus proche case située à gauche contenant le symbole 0 (inclusivement). Les conventions adoptées au sujet de la forme de la mémoire sont donc respectées.

- MARK(n)

  Les machines MARK(n) avec $n \in \{2,3\}$ modifient la mémoire et écrivent le symbole n dans les cellules observées par la tête de lecture tant que celles-ci contiennent le symbole 1. Après chaque écriture, la tête se décale vers la droite. Ainsi, dans notre représentation, si la tête de lecture est positionnée sur la première cellule contenant le symbole 1 d'une représentation d'entier naturel, alors tous les symboles 1 servant à représenter cet entier vont être remplacés par le symbole n. Ces machines permettent de marquer un entier particulier lorsqu'on travaille avec des

représentations d'uplet ; cela permet en quelque sorte de sélectionner cet élément.

$$\text{MARK}(2) = \textbf{while RIGHT}\,;\ \text{READ}(1)\ \textbf{do WRITE}(2)\ \textbf{od}.$$

$$\text{MARK}(3) = \textbf{while RIGHT}\,;\ \text{READ}(1)\ \textbf{do WRITE}(3)\ \textbf{od}.$$

Ces machines ne modifient que le contenu de cases contenant le symbole 1. Par conséquent leur action respecte nécessairement les conventions adoptées quant à la forme de la mémoire pour peu que la mémoire sur laquelle elles travaillent respecte initialement ces conventions.

- THEREIS(n)

  Les machines THEREIS(n) avec $n \in \{2, 3\}$ entrent dans l'état final q_2 si le symbole n est présent dans une des cellules de la mémoire et dans l'état final q_3 sinon. La tête de lecture s'arrête sur la première cellule rencontrée contenant le symbole n.

$$\begin{aligned}\text{THEREIS}(2) \;=\; & \text{STAR}\,; \\ & \textbf{while READNOT}(2)\ \textbf{do} \\ & \textbf{if READNOT}(\lambda)\ \textbf{then RIGHT} \\ & \textbf{od}.\end{aligned}$$

  La mémoire n'est pas modifiée par les machines utilisées dans la composition.

- THEREWAS(n)

  Les machines THEREWAS(n) avec $n \in \{2, 3\}$ entrent dans l'état final q_2 si le symbole n est présent dans une des cellules de la mémoire et dans l'état final q_3 sinon. De plus, elles écrivent le symbole 1 dans la première cellule rencontrée contenant le symbole n.

$$\text{THEREWAS}(2) = \textbf{if THEREIS}(2)\ \textbf{then WRITE}(1).$$

  La machine THEREIS(2) positionne la tête de lecture sur la première case contenant le symbole 2. Par convention, aucune case vide ou contenant le symbole λ n'est située à gauche de cette case. Puisque la machine WRITE(1) ne fait qu'inscrire le symbole 1 dans la case mentionnée précédemment, alors les conventions concernant la forme de la mémoire sont respectées.

- RESTORE

  La machine RESTORE modifie la mémoire en écrivant le symbole 1 dans toute cellule de la mémoire contenant les symboles 2 ou 3.

$$\begin{aligned} \text{RESTORE} \;=\; & \textbf{while THEREIS(2) do THEREWAS(2) od;} \\ & \textbf{while THEREIS(3) do THEREWAS(3) od.} \end{aligned}$$

  Les machines utilisées dans cette composition respectent les conventions concernant la forme de la mémoire.

- APPEND(k)

  Les machines APPEND(k) avec $k > 0$ modifient la mémoire de sorte que si l'uplet $(a_1, \ldots, a_n)$ est représenté dans l'état initial, alors à l'arrêt de la machine, la mémoire représente $(a_1, \ldots, a_n + a_k)$. Si $k > n$, alors la machine ne s'arrête pas.

$$\begin{aligned} \text{APPEND}(k) \;=\; & \text{FIND}(k)\,;\, \text{MARK(2)}\,; \\ & \textbf{while THEREWAS(2) do INC od.} \end{aligned}$$

  Les machines utilisées dans cette composition respectent les conventions concernant la forme de la mémoire.

- COPY(k)

  Les machines COPY(k) avec $k > 0$ modifient la mémoire de sorte que si l'uplet $(a_1, \ldots, a_n)$ est représenté dans l'état initial, alors à l'arrêt de la machine, la mémoire représente l'uplet $(a_1, \ldots, a_n, a_k)$. Si $k > n$, alors la machine ne s'arrête pas.

$$\text{COPY}(k) = \text{NEW}\,;\, \text{APPEND}(k).$$

  Les machines utilisées dans cette composition respectent les conventions relatives à la forme de la mémoire.

- ADD(i, j)

  Les machines ADD(i, j) avec $i, j > 0$ modifient la mémoire de sorte que si l'uplet $(a_1, \ldots, a_n)$ est représenté dans l'état initial, alors à l'arrêt de l'exécution, la mémoire représente l'uplet $(a_1, \ldots, a_n, a_i + a_j)$. Si $i > n$ ou $j > n$, alors la machine

ne s'arrêtera pas.

$$\text{ADD}(i,j) = \text{COPY}(i)\,;\ \text{APPEND}(j).$$

Les machines utilisées dans cette compositions respectent les conventions relatives à la forme de la mémoire.

- MULT(i,j)

  Les machines MULT(i,j) avec $i,j > 0$ modifient la mémoire de sorte que si l'uplet $(a_1,\ldots,a_n)$ est représenté dans l'état initial, alors à l'arrêt de l'exécution, la mémoire représente l'uplet $(a_1,\ldots,a_n,a_ia_j)$. Si $i > n$ ou $j > n$, alors la machine ne s'arrêtera pas. Il existe deux machines différentes qui gèrent deux cas : le cas où i et j sont distincts et le cas où ils sont identiques.

$$\begin{aligned}\text{MULT}(i,j) \;=\;& \text{FIND}(i)\,;\ \text{MARK}(3)\,;\\ & \textbf{while}\ \text{THEREWAS}(3)\ \textbf{do}\ \text{APPEND}(j)\ \textbf{od}.\end{aligned}$$

$$\begin{aligned}\text{MULT}(i,i) \;=\;& \text{COPY}(i)\,;\ \text{LAST}\,;\ \text{MARK}(3)\,;\\ & \textbf{while}\ \text{THEREWAS}(3)\ \textbf{do}\\ & \ \textbf{while}\ \text{THEREWAS}(3)\ \textbf{do}\ \text{APPEND}(i)\ \textbf{od}\\ & \textbf{od}.\end{aligned}$$

  Notons, en ce qui concerne la seconde machine, que le premier **while** est présent uniquement pour supprimer une occurrence de 3 dans le cas où le i-ème élément de l'uplet représenté en mémoire serait différent de 0. En effet, on copie cet élément au moyen de la machine COPY. Il suffit alors d'additionner $a_i - 1$ fois a_i à cette copie pour obtenir a_i^2. Les machines utilisées dans cette composition respectent les conventions relatives à la forme de la mémoire.

- NOTGREATER(i,j)

  Les machines NOTGREATER(i,j) avec $i,j > 0$ comparent les élements a_i et a_j de l'uplet $(a_1,\ldots,a_n)$ représenté en mémoire dans l'état initial. Si $a_i > a_j$, alors l'exécution s'arrête dans l'état q_3, sinon elle s'arrête dans l'état q_2. Si $i > n$ ou $j > n$, alors l'exécution ne s'arrête pas.

NOTGREATER(k,l) = FIND(k) ;MARK(2) ; FIND(l) ; MARK(3) ;
while THEREIS(2) **and** THEREIS(3) do
THEREWAS(2) ; THEREWAS(3)
od ;
while THEREIS(2) **do**
RESTORE ; STOP
od ;
RESTORE.

Les machines utilisées dans cette compositions respectent les conventions relatives à la forme de la mémoire.

- EQUAL(i,j) et NOTEQUAL(i,j)

  Les machines EQUAL(i,j) avec $i,j > 0$ comparent les éléments a_i et a_j de l'uplet $(a_1,\ldots,a_n)$ représenté en mémoire à l'état initial. Si $a_i = a_j$, alors l'exécution s'arrête dans l'état q_2, sinon elle s'arrête dans l'état q_3. Si $i > n$ ou $j > n$, alors l'exécution ne s'arrête pas.

  $$\text{EQUAL}(k,l) = \text{NOTGREATER}(k,l) \textbf{ and } \text{NOTGREATER}(l,k).$$

  Les machines NOTEQUAL(i,j) avec $i,j > 0$ comparent les éléments a_i et a_j de l'uplet $(a_1,\ldots,a_n)$ représenté en mémoire à l'état initial. Si $a_i \neq a_j$, alors l'exécution s'arrête dans l'état q_2, sinon elle s'arrête dans l'état q_3. Si $i > n$ ou $j > n$, alors l'exécution ne s'arrête pas.

  $$\text{NOTEQUAL}(k,l) = \textbf{while } \text{EQUAL}(k,l) \textbf{ do STOP od}.$$

  Les machines utilisées dans ces compositions respectent les conventions relatives à la forme de la mémoire.

- NEXT

  La machine NEXT modifie la mémoire de sorte que si l'uplet $(a_1,\ldots,a_n)$ est représenté dans l'état initial, alors à l'arrêt de la machine, la mémoire représente

l'uplet $(a_1, \ldots, a_{n-2}, b, c)$, où (b, c) est le couple qui suit (a_{n-1}, a_n) dans l'énumération de Cantor des couples. L'énumération de Cantor est en quelque sorte le « trajet » parcouru sur le plan $\mathbb{N} \times \mathbb{N}$ lorsqu'on énumère les couples (a, b) de $\mathbb{N} \times \mathbb{N}$ dans l'ordre correspondant à l'ordre croissant des résultats obtenus par $Cantor(a, b)$. L'énumération de Cantor est de la forme

$$(0,0), (0,1), (1,0), (0,2), (1,1), (2,0), \ldots$$

NEXT = LAST ; WRITE(1) ; RIGHT ;
while READ(λ) **do** WRITE(1) ; LAST ; RIGHT **od** ;
WRITE(0)

Notons qu'ici encore, le **while** n'a pas vocation à créer une boucle mais agit plutôt comme une vérification ponctuelle de l'état de la mémoire. La machine LAST positionne la tête de lecture sur la case contenant le 0 initiant la représentation de a_n. Par convention, il n'existe pas de case vide ou contenant le symbole λ à gauche de cette case. Donc, après l'exécution de la machine WRITE(1), la mémoire respecte les conventions relatives à la forme de la mémoire. La machine RIGHT déplace ensuite la tête de lecture d'une case vers la droite. Cette case peut être vide, contenir le symbole λ ou contenir un autre symbole.

- Si elle contient un symbole autre que λ, alors on exécute directement la machine WRITE(0). Puisqu'aucune case à gauche de la case sur laquelle est positionnée la tête de lecture n'est vide ou ne contient le symbole λ, alors les conventions relatives à la forme de la mémoire sont respectées.
- Si elle est vide ou contient le symbole λ, alors on exécute la machine WRITE(1) qui écrit le symbole 1 dans la case. Les conventions relatives à la forme de la mémoire sont respectées. L'exécution de la machine LAST repositionne la tête de lecture sur la dernière case contenant 0 présente dans la mémoire, puis la machine RIGHT positionne la tête de lecture immédiatement à droite de cette case. Par convention sur la forme de la mémoire, cette dernière case

est nécessairement vide ou contient le symbole λ. La machine WRITE(0) y écrit le symbole 0 et pour la mémoire alors obtenue, les conventions relatives à la forme de la mémoire sont respectées.

- DECODE

  La machine DECODE modifie la mémoire de sorte que si l'uplet $(a_1, \ldots, a_n)$ est représenté dans l'état initial, alors à l'arrêt de la machine, la mémoire représente l'uplet $(a_1, \ldots, a_n, b, c)$, où (b, c) est le couple dont le numéro de Cantor est a_n.

$$\begin{aligned} \text{DECODE} \;=\; & \text{LAST}\,;\, \text{MARK(2)}\,;\, \text{NEW}\,;\, \text{NEW}\,; \\ & \textbf{while}\ \text{THEREWAS(2)}\ \textbf{do}\ \text{NEXT}\ \textbf{od} \end{aligned}$$

  Les machines utilisées dans cette composition respectent les conventions relatives à la forme de la mémoire.

Nous sommes maintenant en mesure de créer une machine capable de reconnaître les ensembles diophantiens.

6.6 Reconnaissance d'un ensemble diophantien par une machine de Turing

Considérons une équation diophantienne paramétrée quelconque à n paramètres et $m+1$ inconnues

$$D(a_1, \ldots, a_n, x_1, \ldots, x_{m+1}) = 0.$$

On veut construire une machine de Turing qui, lorsqu'on lui passe en entrée la représentation d'un uplet $(a_1, \ldots, a_n)$, s'arrête si et seulement si

$$D(a_1, \ldots, a_n, x_1, \ldots, x_{m+1}) = 0$$

admet une solution. L'idée sous-jacente à la machine dont Matiiassevitch (1995) propose la construction est triviale : il s'agit simplement calculer la valeur de $D(a_1, \ldots, a_n, x_1, \ldots, x_{m+1})$ pour tous les $(m+1)$-uplets possibles jusqu'à ce qu'on trouve la valeur 0. S'il n'est pas possible de trouver une telle valeur, c'est-à-dire si l'équation n'admet pas de solution

pour ces valeurs de paramètres, alors la machine ne s'arrêtera jamais. Puisqu'il existe une bijection entre les entiers naturels et les m-uplets pour $m > 0$, alors pour s'assurer de tester tous les m-uplets possibles, il suffit d'énumérer les entiers un par un dans l'ordre croissant en testant à chaque fois le m-uplet dont il est le nombre de Cantor. Commençons par construire une machine qui, prenant en entrée la représentation de l'uplet $(a_1, \ldots, a_n, y_0)$, vérifie que le $(m+1)$-uplet codé par y_0 est une solution de l'équation $D(a_1, \ldots, a_n, x_1, \ldots, x_{m+1})$. Rappelons que

$$Cantor_{m+1}(x_1, \ldots, x_{m+1}) = Cantor(x_1, , Cantor_m(x_2, \ldots, x_{m+1})).$$

Ceci implique que pour décoder un numéro d'uplet, c'est-à-dire pour obtenir l'uplet à partir du numéro, il est nécessaire de décoder successivement le numéro initial, puis le dernier membre de l'uplet obtenu, puis le dernier membre de l'uplet obtenu et ainsi de suite. Ainsi, si nous utilisons m fois la machine DECODE en partant de la représentation de l'uplet $(a_1, \ldots, a_n, y_0)$, on obtiendra à l'arrêt de la machine la représentation de l'uplet $(a_1, \ldots, a_n, y_0, x_1, y_1, \ldots, x_m, y_m)$ avec $x_1, \ldots, x_m, y_m$ les $m+1$ éléments de l'uplet qu'on va tester comme solution potentielle et $y_0, \ldots, y_{m-1}$ les numéros de Cantor à décoder successivement. Notons qu'ici, y_m n'est pas le code d'un couple de Cantor mais le deuxième élément du dernier couple à avoir été encodé. En conséquence, on a $x_{m+1} = y_m$.

Après l'utilisation de m machines DECODE, nous sommes donc en possession de tous les éléments nécessaires au calcul de la valeur du polynôme D. Nous avons construit précédemment des machines de Turing permettant d'effectuer des modifications sur la mémoire pouvant être interprétées comme les opérations arithmétiques d'addition et de multiplication. Il est pertinent de remarquer que le calcul de la valeur d'un polynôme à coefficients entiers naturels ne nécessite pas d'autres opérations arithmétiques. Le problème est qu'ici, rien ne garantit que le polynôme D soit à coefficients entiers naturels. Cette difficulté est contournable aisément. En effet, vérifier si

$$D(a_1, \ldots, a_n, x_1, \ldots, x_{m+1}) = 0$$

revient à vérifier si

$$C_L(a_1, \ldots, a_n, x_1, \ldots, x_{m+1}) = C_R(a_1, \ldots, a_n, x_1, \ldots, x_{m+1}),$$

où C_L et C_R sont des polynômes à coefficients entiers naturels tels que

$$C_L(a_1, \ldots, a_n, x_1, \ldots, x_{m+1}) - C_R(a_1, \ldots, a_n, x_1, \ldots, x_{m+1}) = D(a_1, \ldots, a_n, x_1, \ldots, x_{m+1}).$$

Supposons maintenant que k étapes soient nécessaires pour calculer la valeur de $C_L(a_1, \ldots, a_n, x_1, \ldots, x_{m+1})$ et ℓ étapes pour calculer $C_R(a_1, \ldots, a_n, x_1, \ldots, x_{m+1})$. On peut alors enchaîner les machines NEW, INC, ADD et MULT pour obtenir, à partir de la représentation en mémoire de l'uplet $(a_1, \ldots, a_n, y_0, x_1, y_1, \ldots, x_m, y_m)$ la représentation de l'uplet

$$(a_1, \ldots, a_n, y_0, x_1, y_1, \ldots, x_m, y_m, 1, z_1, \ldots, z_k, z_{k+1}, \ldots, z_{k+\ell}))$$

dans lequel z_1 à z_k sont les valeurs successives calculées pour obtenir la valeur de $C_L(a_1, \ldots, a_n, x_1, \ldots, x_{m+1})$ et z_{k+1} à $z_{k+\ell}$ sont les valeurs successives calculées pour obtenir la valeur de $C_R(a_1, \ldots, a_n, x_1, \ldots, x_{m+1})$. Cela signifie que z_k et $z_{k+\ell}$ sont respectivement les valeurs de $C_L(a_1, \ldots, a_n, x_1, \ldots, x_{m+1})$ et $C_R(a_1, \ldots, a_n, x_1, \ldots, x_{m+1})$. Donc, pour vérifier que l'uplet $(x_1, \ldots, x_{m+1})$ constitue une solution de l'équation

$$D(a_1, \ldots, a_n, x_1 \ldots, x_{m+1}) = 0,$$

il suffit de vérifier que $z_k = z_{k+\ell}$. Notons au passage la présence de la valeur 1 dans l'uplet. Il est nécessaire de faire apparaître cette valeur dans l'uplet pour permettre l'utilisation de constantes dans les opérations arithmétiques effectuées par la machine de Turing.

La machine M_1 sera donc constituée d'un enchaînement de m machines DECODE permettant l'extraction du $(m+1)$-uplet à tester, puis d'un enchaînement de machines NEW, INC, ADD et MULT permettant de calculer la valeur des polynômes

$$C_L(a_1, \ldots, a_n, x_1, \ldots, x_{m+1})$$

et

$$C_R(a_1, \ldots, a_n, x_1, \ldots, x_{m+1}).$$

Enfin, on compose avec une machine NOTEQUAL(k,$\ell+k$), ce qui fait passer la machine dans l'état q_2 si

$$C_L(a_1, \ldots, a_n, x_1, \ldots, x_{m+1}) \neq C_R(a_1, \ldots, a_n, x_1, \ldots, x_{m+1})$$

et dans l'état q_3 sinon. Supposons maintenant que la machine M_1 s'arrête dans l'état q_2. Cela signifie que l'uplet testé ne constitue pas une solution de l'équation. Nous désirons donc tester l'uplet suivant dans la numérotation de Cantor. Après l'exécution de la machine M_1, la mémoire contient la représentation de l'uplet

$$(a_1, \ldots, a_n, y_0, x_1, y_1, \ldots, x_m, y_m, 1, z_1, \ldots, z_k, z_{k+1}, \ldots, z_{k+\ell}).$$

Pour tester un nouvel uplet, il faudrait effacer toutes les valeurs introduites par la machine M_1, c'est à dire les $2m + \ell + k + 1$ dernières valeurs. Ceci peut être effectué en employant une machine constituée d'un enchaînement de $2m + k + \ell + 1$ machines DELETE. On obtient alors un uplet $(a_1, \ldots, a_n, y_0)$.

Puisqu'il ne nous intéresse pas de tester à nouveau le même $(m + 1)$-uplet, il faut attribuer une autre valeur à y_0. Nous avons évoqué précédemment la possibilité d'énumérer les numéros de Cantor dans l'ordre croissant. Il est possible d'obtenir la représentation de l'uplet $(a_1, \ldots, a_n, y_0 + 1)$ à partir de la représentation de l'uplet $(a_1, \ldots, a_n, y_0)$ en utilisant une machine INC. La machine M_2 sera donc constituée de l'enchaînement de $2m + k + \ell + 1$ machines DELETE et d'une machine INC. Si la machine M_1 n'entre pas dans l'état q_3, on voudrait que la machine M_2 nettoie la mémoire et incrémente le numéro de Cantor de l'uplet à tester. Nous pouvons utiliser la composition **while** M_1 **do** M_2 **od** pour construire une machine testant les numéros de Cantor dans l'ordre croissant et ne s'arrêtant que lorsqu'un numéro de Cantor code une solution de l'équation. Pour achever la construction de la machine reconnaissant l'ensemble diophantien associé à l'équation, rappelons que seule la représentation de l'uplet $(a_1, \ldots, a_n)$ doit être présente en mémoire dans l'état initial. Or la machine M_1

utilise la représentation d'un uplet $(a_1, \ldots, a_n, y_0)$ pour fonctionner correctement. Il faut donc ajouter à la représentation initiale de l'uplet $(a_1, \ldots, a_n)$ la valeur du premier numéro de Cantor, c'est-à-dire 0. La machine NEW permet d'effectuer cette opération.

Au final, la machine M = NEW; **while** M_1 **do** M_2 **od** à laquelle on passe la représentation de l'uplet $(a_1, \ldots, a_n)$ atteint un état final si et seulement s'il existe une solution de l'équation $\mathrm{D}(a_1, \ldots, a_n, x_1, \ldots, x_{m+1}) = 0$, c'est à dire si et seulement si $(a_1, \ldots, a_n)$ appartient à l'ensemble diophantien représenté par l'équation. Pour chaque équation diophantienne paramétrée, il est donc possible de construire une machine de Turing capable de reconnaître exactement les éléments de l'ensemble représenté par l'équation. Donc, les ensembles diophantiens sont semi-décidables.

6.7 Simulation diophantienne des machines de Turing : les ensembles semi-décidables sont diophantiens

L'objectif de cette section est de montrer que les ensembles semi-décidables sont diophantiens. On constatera en fait qu'à partir de toute machine de Turing, il est possible de construire une équation diophantienne paramétrée admettant une solution si et seulement si les paramètres de l'équation représentent un mot reconnu par la machine de Turing (c'est-à-dire si et seulement si la machine de Turing s'arrête lorsqu'on lui passe ce mot en entrée).

Soit une machine de Turing M. Soit $A = \{\alpha_1, \ldots, \alpha_w\}$ l'alphabet de M. Soit $\{q_1, \ldots, q_v\}$ l'ensemble des états de M.

Il est possible, en utilisant les codes positionnels, de représenter une configuration de M à chaque étape de l'exécution. En effet, à tout moment de son exécution, une machine de Turing n'utilise qu'une portion de longueur finie ℓ de sa mémoire. Cette bande de mémoire peut être représentée par un uplet dont la i-ième valeur code le symbole contenu dans la i-ième cellule de la mémoire. De plus, en décidant de représenter une cellule vide par l'entier 0, on peut donner le numéro de code positionnel d'un tel uplet sans préciser sa longueur. On peut alors considérer que l'uplet possède une infinité

de valeurs nulles après les ℓ premières valeurs.

Pour représenter l'état de la machine et la position de la tête de lecture sur la mémoire, on utilisera un uplet dont tous les éléments, excepté un, seront nuls. La position de l'élément non nul dans l'uplet représentera la position de la tête de lecture et sa valeur représentera l'état de M. Ici encore, si on fournit le numéro de code positionnel de l'uplet, on n'est pas obligé de préciser la longueur de cet uplet et on peut considérer l'uplet comme possédant une infinité de valeurs nulles après les ℓ premières valeurs. Ces deux uplets seront respectivement encodés par les numéros de codes positionnels p et t en base β, avec $\beta > 4$, $\beta > w$, $\beta > v$.

Définition 6.7.1. *Dans la discussion ci-dessus, on appelle le couple (code du contenu,code de la position et de l'état)* code de configuration *de la machine M.*

Ce code contient toutes les informations nécessaires au passage d'une configuration de M à la configuration suivante : l'état de la machine, le contenu de la mémoire et la position de la tête de lecture.

On veut maintenant avoir la possibilité de manipuler directement ces codes pour obtenir les codes correspondant à la configuration suivante. Il est nécessaire d'adapter les fonctions D, A et Q à notre codage. Rappelons que D, A et Q formalisent respectivement le déplacement de la tête de lecture, le symbole à écrire et le changement d'état qui interviennent pour chaque instruction d'une machine de Turing. Toutes les fonctions qui vont être décrites par la suite dépendent de la machine M que l'on veut simuler et de la base d'encodage β choisie.

Définition 6.7.2. *Soit $NextT : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ la fonction qui à un code de configuration (p,t) correspondant à une configuration c de M associe t', le numéro du code positionnel en base β de l'uplet représentant la mémoire de M dans la configuration suivant c. Soit $NextP : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ la fonction qui à un code de configuration (p,t) correspondant à une configuration c de M associe p', le numéro de code positionnel en base β de l'uplet représentant l'état de M et la position de la tête de lecture dans la configuration suivant c.*

Montrons que ces deux fonctions sont diophantiennes. On commence par définir des fonctions traduisant les instructions des machines de Turing. Considérons une machine de Turing M à v états $\{q_1, \ldots, q_v\}$ possédant un alphabet de w symboles $\{\alpha_1, \ldots, \alpha_w\}$ et un jeu d'instruction de la forme

$$q_i\alpha_j \Rightarrow \alpha_{A(i,j)}D(i,j)q_{Q(i,j)}.$$

Les fonctions A, Q et D ne sont pas initialement définies pour tout couple de l'ensemble $\{1, \ldots, v\} \times \{0, \ldots, w\}$. En effet, il n'existe pas d'instruction concernant les états finaux, puisque le machine s'arrête lorsqu'elle entre dans un état final. On définit totalement A, Q et D en prolongeant leur définition pour les états finaux comme suit. Si i est l'indice d'un état final, alors on a

$$A(i,j) = j \quad Q(i,j) = 0 \quad D(i,j) = S$$

Définition 6.7.3. *Soit* $A' : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *définie par*

$$A'(i,j) = \begin{cases} A(i,j) & \text{si } 0 < i \leq v, 0 \leq j \leq w \\ j & \text{sinon,} \end{cases}$$

où v *le nombre d'états dans la machine* M *et* w *le nombre de symboles dans l'alphabet* A.

La fonction A' se comporte donc comme la fonction A si les arguments fournis sont les indices des états et symboles de M. Donc, si les nombres t et p sont les numéros de code des uplets représentant respectivement la mémoire de M et la position de la tête de lecture et l'état de M au pas k, alors le prolongement $A'[\beta]$ de A', défini au chapitre 5, est tel que

$$A'[\beta](t,p,\ell) = t',$$

où ℓ est un nombre quelconque tel que $t, p < \beta^\ell$ et t' est le numéro de code positionnel de l'uplet représentant l'état de la mémoire de M au pas $k+1$. En effet, la fonction $A'[\beta]$ ne modifie que le i-ème élément de l'uplet de numéro t situé à la position i telle que $p_i > 0$. Puisque p est le numéro de code de l'uplet représentant l'état de la machine et la

position de la tête de lecture, alors il n'existe qu'un seul i tel que $p_i > 0$ et i correspond alors à la position de la tête de lecture sur la bande. En conséquence, on n'applique la fonction A qu'à l'élément de l'uplet de numéro de code t représentant la case mémoire située sous la tête de lecture de M au pas k. Ceci correspond bien à l'écriture sur la mémoire effectuée par la machine M. On peut alors définir *Next* comme suit.

Lemme 6.7.1. *Soit $\beta \geq 2$. Soient t et p des codes positionnels en base β représentant la mémoire, la position de la tête de lecture et l'état d'une machine de Turing M au pas k. On a*

$$t' = NextT(p,t) \Leftrightarrow \exists \ell \left[t' = A'[\beta](t,p,\ell) \right].$$

Démonstration (*). Soit ℓ tel que $\beta^\ell > p$ et $\beta^\ell > t$.
Soient $t_1, t_2, \ldots$ et $p_1, p_2, \ldots$ tels que

$$t = \sum_{i=1}^{\ell} t_i \beta^{i-1}$$

$$p = \sum_{i=1}^{\ell} p_i \beta^{i-1}.$$

Par définition de p et t et par unicité de la représentation d'un nombre en base β, pour tout $i \in \{1, \ldots, \ell\}$, l'uplet $(t_1, \ldots, t_\ell)$ est tel que t_i représente le contenu de la i-ème case de la mémoire de M au pas k et l'uplet $(p_1, \ldots, p_\ell)$ est tel que $p_i = 0$ si la tête de lecture de M n'observe pas la i-ème case de la mémoire au pas k et $p_i = q$ où q est létat de M au pas k si la tête de lecture observe la i-ème case de la mémoire au pas k.
($\Rightarrow$) Soit t' tel que $t' = NextT(t,p)$. Soit ℓ' tel que $\beta^{\ell'} > t'$. Par unicité de la représentation d'un nombre en base β, il existe d'uniques nombres $t'_1, \ldots, t'_{\ell'}$ tels que

$$t' = \sum_{i=1}^{\ell'} t'_i \beta^{i-1}.$$

Par définition de $NextT$, t'_i représente le contenu de la i-ème case de la mémoire de M au pas $k+1$ pour tout $i \in \{1, \ldots, \ell'\}$.
Pour tout $i \in \{1, \ldots, \ell'\}$, on a donc

- $t'_i = t_i$ si la tête de lecture n'est pas en i. Dans ce cas, on a aussi $p_i = 0$. Donc, $t'_i = A'(p_i, t_i)$
- $t'_i = A(p_i, t_i)$ si la tête de lecture est en i. Dans ce cas, on a $p_i = q$. Donc, $t'_i = A'(p_i, t_i)$.

Soit $\ell'' = \text{Max}(\ell, \ell')$. On a alors $\beta^{\ell''} > t, t', p, p'$ de ce qui précède, on peut conclure que $t' = A'[\beta](p, t, \ell'')$.

($\Leftarrow$) Soit ℓ' tel que $t' = A'[\beta](p, t, \ell')$. Par unicité de la représentation d'un nombre en base β, il existe d'uniques nombres $t'_1, \ldots, t'_{\ell'}$ tels que

$$t' = \sum_{i=1}^{\ell'} t'_i \beta^{i-1}.$$

On a alors, pour tout $i \in \{1, \ldots, \ell'\}$,

$$t'_i = A'(p_i, t_i).$$

Soit $i \in \{1, \ldots, \ell'\}$.

- Si la tête de lecture de M observe la i-ème case de la mémoire, alors $p_i > 0$. Dans ce cas, $t'_i = A(p_i, t_i)$. Donc, le i-ème élément de l'uplet $(t'_1, \ldots, t'_{\ell'})$ représente le contenu de la i-ème case de la mémoire au pas $k + 1$.
- Si la tête de lecture de M n'observe pas la i-ème case de la mémoire, alors $p_i = 0$. Dans ce cas, $t'_i = t_i$. Donc, le i-ème élément de l'uplet $(t'_1, \ldots, t'_{\ell'})$ représente le contenu de la i-ème case de la mémoire au pas $k + 1$ (puisque la tête de lecture n'observe pas cette case au pas k, son contenu est inchangé au pas $k + 1$).

Pour tout $i \in \{1, \ldots, \ell'\}$, t'_i représente donc le contenu de la $i$ème case de la mémoire au pas $k + 1$. Par définition, t' code donc la mémoire de M au pas $k + 1$. On a donc $t' = NextT(p, t)$. □

Puisque A' est une endofonction diophantienne, alors, d'après le chapitre 5, le prolongement $A'[\beta]$ de cette application aux uplets est également diophantien. Donc, le lemme précédent fournit une définition diophantienne de la fonction $NextT$.

Montrons maintenant que $NextP$ est diophantienne. On utilisera la même méthode que précédemment, mais il y a une difficulté supplémentaire. S'il est facile d'appliquer la fonction Q aux uplets codants pour changer l'état de la machine codée, il est moins aisé de trouver le moyen d'appliquer D de manière adéquate, c'est-à-dire de déplacer la tête de lecture. Soit (p,t) un code de configuration. Les uplets p et $\mathbf{t}$ dont les numéros de code positionnel sont p et t représentent respectivement la mémoire de M et l'état et la position de la tête de lecture de M au pas k. Soit j tel que $p_j > 0$. Soit (p',t') le code de configuration de M au pas $k+1$. Soient $\mathbf{p}'$ et $\mathbf{t}'$ les uplets dont les numéros de code positionnel sont respectivement p' et t'. Supposons que i soit la position de la tête de lecture. Le problème est que la tête de lecture a possiblement été déplacée en $i-1$ ou en $i+1$ au pas $k+1$. Or, si on se contente de prolonger la fonction Q aux uplets en utilisant la méthode du chapitre 5, les valeurs de p'_{i-1}, p'_i et p'_{i+1} ne peuvent être respectivement obtenues qu'à partir de t_{i-1}, p_{i-1}, t_i, p_i, t_{i+1} et p_{i+1}. On ne peut pas dans ces conditions obtenir le numéro de code positionnel de l'uplet tel que $p_j > 0$ et $j \neq i$. Il est nécessaire de trouver une autre manière de procéder.

Observons en premier lieu quelques propriétés des uplets $\mathbf{p}$ et $\mathbf{t}$ dont les numéros de code sont respectivement p et t.

Définition 6.7.4. *Soient*

$$p_R = p\beta, \quad p_L = b \text{ div } \beta$$
$$t_R = p\beta, \quad t_L = b \text{ div } \beta.$$

Lemme 6.7.2 $(*)$. *Soit ℓ tel que $p < \beta^\ell$ et $t < \beta^\ell$.*
Soient $p_1, \ldots, p_\ell, t_1, \ldots, t_\ell, p'_1, \ldots, p'_\ell, t'_1, \ldots, t'_\ell, p''_1, \ldots, p''_\ell, t''_1, \ldots, t''_\ell$ tels que

$$p = \sum_{i=1}^{\ell} p_i\beta^{i-1} \qquad t = \sum_{i=1}^{\ell} t_i\beta^{i-1}$$
$$p_R = \sum_{i=1}^{\ell} p'_i\beta^{i-1} \qquad t_R = \sum_{i=1}^{\ell} t'_i\beta^{i-1}$$
$$p_L = \sum_{i=1}^{\ell} p''_i\beta^{i-1} \qquad t_L = \sum_{i=1}^{\ell} t''_i\beta^{i-1}.$$

Alors,

- *Pour $i = 1$, on a $p'_i = 0$, $p''_i = p_{i+1}$, $t'_i = 0$ et $t''_i = t_{i+1}$;*
- *Pour $i = \ell$, on a $p''_i = 0$, $p'_i = p_{i-1}$, $t''_i = 0$ et $t'_i = t_{i-1}$;*

- *Pour* $i \in \{2, \ldots, \ell - 1\}$, *on a* $p'_i = p_{i-1}$, $p''_i = p_{i+1}$, $t'_i = t_{i-1}$ *et* $t''_i = t_{i+1}$.

Démonstration (*). On a

$$\begin{aligned} p_R &= p\beta \\ &= \beta \sum_{i=1}^{\ell} p_i \beta^{i-1} \\ &= \sum_{i=1}^{\ell} p_i \beta^i. \end{aligned}$$

Soit $p_0 = 0$, on a alors

$$p_R = \sum_{i=1}^{\ell} p_i \beta^i + p_0 = \sum_{i=0}^{\ell} p_i \beta^i.$$

Donc, $(p_R, \beta, \ell + 1)$ code l'uplet $(0, p_1, \ldots, p_\ell)$. De même, on a

$$\begin{aligned} p_L &= p \text{ div } \beta \\ &= \sum_{i=1}^{\ell} p_i \beta^{i-1} \text{ div } \beta \\ &= \left(\sum_{i=2}^{\ell} p_i \beta^{i-1} \text{ div } \beta \right) + (t_1 \text{ div } \beta). \end{aligned}$$

Puisque $p_1 < \beta$, alors p_1 div $\beta = 0$ et on a

$$\begin{aligned} p_L &= \sum_{i=2}^{\ell} p_i \beta^{i-1} \text{ div } \beta \\ &= \sum_{i=2}^{\ell} p_i \beta^{i-2} \\ &= \sum_{i=1}^{\ell-1} p_{i+1} \beta^{i-1}. \end{aligned}$$

Donc, $(p_L, \beta, \ell - 1)$ code l'uplet $(p_2, \ldots, p_\ell, 0)$. On a de plus

$$p = \sum_{i=1}^{\ell} p_i \beta^{i-1},$$

$$p_R = \sum_{i=1}^{\ell} p'_i \beta^{i-1},$$

$$p_L = \sum_{i=1}^{\ell} p''_i \beta^{i-1}.$$

Donc, par unicité de la représentation d'un nombre en base β, on a bien

- pour $i = 1$, $p'_i = 0$, $p''_i = p_{i+1}$,
- pour $i = \ell$, $p''_i = 0$, $p'_i = p_{i-1}, et$
- pour $i \in \{2, \ldots, \ell - 1\}$, $p'_i = p_{i-1}$, $p''_i = p_{i+1}$.

De même, on prouve que $(t_R, \beta, \ell + 1)$ code l'uplet $(0, t_1, \ldots, t_\ell)$ et que $(t_L, \beta, \ell - 1)$ code l'uplet $(t_2, \ldots, t_\ell, 0)$, donc que

- pour $i = 1$, $t'_i = 0$, $t''_i = t_{i+1}$,
- pour $i = \ell$, $t''_i = 0$, $t'_i = t_{i-1}$, et
- pour $i \in \{2, \ldots, \ell - 1\}$, $t'_i = t_{i-1}$, $t''_i = t_{i+1}$.

□

Multiplier p (ou t) par β revient à décaler les éléments encodés par p (ou par t) vers la droite (avec introduction d'un 0 à gauche). Diviser p (ou t) par β revient à décaler les éléments encodés par p (ou par t) vers la gauche (avec disparition de p_1 ou t_1). On est donc en mesure de décaler les informations nécessaires au calcul du prochain état de M à gauche ou à droite dans les uplets codant la configuration. Ceci va servir à simuler le déplacement de la tête de lecture de la machine. Considérons les uplets suivants, avec i tel que $p_i > 0$:

$$\begin{array}{ll} (0, \ldots, 0, p_i, 0, \ldots, 0) & (t_1, \ldots, t_{i-1}, t_i, t_{i+1}, \ldots, t_\ell, 0) \\ (0, \ldots, p_i, 0, 0, \ldots, 0) & (t_2, \ldots, t_i, t_{i+1}, t_{i+2}, \ldots, t_\ell, 0, 0) \\ (0, \ldots, 0, 0, p_i, \ldots, 0) & (0, t_1, \ldots, t_i, t_{i+1}, t_{i+2}, \ldots, t_\ell), \end{array}$$

correspondant respectivement à p et t, p_L et t_L et p_R et t_R. Il existe trois possibilités de mouvement de la tête de lecture de la machine : déplacement d'une case vers la gauche, déplacement d'une case vers la droite, pas de déplacement. Pour traiter le cas où aucun déplacement n'est effectué, on peut se servir des uplets originaux codés par p et t puisque la tête de lecture ne se déplace pas, la position de l'élement non nul de p

reste la même. Si la tête de lecture se déplace vers la droite, on peut se servir des uplets codés par p_R et t_R de facon à obtenir p' tel que $(p', \beta, \ell+1)$ code l'uplet $(p'_1, \ldots, p'_\ell + 1)$ avec p'_{i+1} calculé à partir de p_i et t_i, ce qui est exactement ce que nous cherchons. De même, si la tête de lecture se déplace vers la gauche, on peut se servir des uplets codés par p_L et t_L de facon à obtenir p' tel que $(p', \beta, \ell+1)$ code l'uplet $(p'_1, \ldots, p'_{\ell+1})$ avec p'_{i-1} calculé à partir de p_i et t_i, ce qui est exactement ce que nous cherchons.

On va donc définir une nouvelle fonction DQ simulant en même temps D et Q. Cette fonction possède six arguments qui correspondent aux six éléments que la fonction prolongée aux uplets traitera simultanément pour chaque position possible. Elle est définie comme suit.

Définition 6.7.5.

$$DQ(i_L, i, i_R, j_L, j, j_R) = \begin{cases} Q(i_L, j_L) & \text{si } i_L > 0, i = i_R = 0 \text{ et } D(i_L, j_L) = R \\ Q(i, j) & \text{si } i > 0, i_L = i_R = 0 \text{ et } D(i, j) = S \\ Q(i_R, j_R) & \text{si } i_R > 0, i = i_L = 0 \text{ et } D(i_R, j_R) = L \\ 0 & \text{sinon.} \end{cases}$$

Notons que la définition donnée ici diffère de celle utilisée par Matiiassevitch (1995), dans laquelle $D(i_L, j_L) = R$ et $D(i_R, j_R) = L$. Cette différence est probablement le fait d'une coquille qui se serait glissée dans le texte de Matiiassevitch. Cette fonction permet de décider quelle paire d'informations utiliser parmi (i_L, j_L), (i, j) ou (i_R, j_R) en fonction du déplacement de la tête de lecture.

Lorsqu'on prolonge cette fonction aux uplets en utilisant la méthode du chapitre 5, son utilité devient plus claire.

Lemme 6.7.3. *Soient t et p des codes en base β représentant la mémoire et la position de la tête de lecture et l'état q d'une machine de Turing M au pas k. On a*

$$p' = NextP(p, t) \Leftrightarrow \exists \ell \left[p' = DQ[\beta](p\beta, p, p \text{ div } \beta, t\beta, t, t \text{ div } \beta, \ell) \right].$$

Démonstration (*). Soit ℓ tel que $\beta^\ell > \beta p$ et $\beta^\ell > \beta t$. Soient $t_1, \ldots, t_\ell$ et $p_1, \ldots, p_\ell$ tels

que

$$t = \sum_{i=1}^{\ell} t_i \beta^{i-1}, \quad p = \sum_{i=1}^{\ell} p_i \beta^{i-1}.$$

Par définition de p et t et par unicité de la représentation d'un nombre en base β, pour tout $i \in \{1, \ldots, \ell\}$, l'uplet $(t_1, \ldots, t_\ell)$ est tel que t_i représente le contenu de la i-ème case de la mémoire de M au pas k et l'uplet $(p_1, \ldots, p_\ell)$ est tel que $p_i = 0$ si la tête de lecture de M n'observe pas la i-ème case de la mémoire au pas k et $p_i = q$ où q est l'état de M au pas k si la tête de lecture observe la i-ème case de la mémoire au pas k. Soient $t''_1, \ldots, t''_\ell, p'_1, \ldots, p''_\ell, t'''_1, \ldots, t'''_\ell, p'''_1, \ldots, p'''_\ell$ tels que

$$t_R = \sum_{i=1}^{\ell} t''_i \beta^{i-1}, \quad p_R = \sum_{i=1}^{\ell} p''_i \beta^{i-1},$$
$$t_L = \sum_{i=1}^{\ell} t'''_i \beta^{i-1}, \quad p_L = \sum_{i=1}^{\ell} p'''_i \beta^{i-1}.$$

($\Rightarrow$) Soit $p' = NextP(p, t)$. Soient $p'_1, \ldots, p'_\ell$ tels que

$$p' = \sum_{i=1}^{\ell} p'_i \beta^{i-1}.$$

Par unicité de la représentation d'un nombre en base β, $(p'_1, \ldots, p'_\ell)$ est une représentation de la position de la tête de lecture et de l'état de M au pas $k+1$. Supposons que la tête de lecture observe la $i$ème case de la mémoire au pas $k+1$. On a trois possibilités :

- Pour $i = 1$, on a $p'_1 = q$. De plus, au pas k, la tête de lecture ne pouvait qu'observer les cases 1 ou 2. On a deux possibilités :
  - Si la tête de lecture observait la case 1 au pas k, alors $D(p_1, t_1) = S$, $p_1 > 0$, $p_2 = 0$ et $p'_1 = Q(p_1, t_1)$. Donc,

$$p' = DQ(0, p_1, p_2, 0, t_1, t_2) = DQ(p''_1, p_1, p'''_1, t''_1, t_1, t'''_1).$$

  - Si la tête de lecture observait la case 2 au pas k, alors $D(p_2, t_2) = L$, $p_2 > 0$, $p_1 = 0$ et $p'_1 = Q(p_2, t_2)$. Donc,

$$p' = DQ(0, p_1, p_2, 0, t_1, t_2) = DQ(p''_1, p_1, p'''_1, t''_1, t_1, t'''_1).$$

- Pour $i = \ell$, on a $p'_\ell = q$. De plus, au pas k, la tête de lecture ne pouvait qu'observer la case $\ell - 1$. En effet, on a $\beta p < \beta^\ell$, donc $p < \beta^{\ell-1}$, ce qui implique que $p_\ell = 0$. Puisque la tête de lecture observait la case $\ell - 1$ au pas k, alors $D(p_{\ell-1}, t_{\ell-1}) = R$, $p_{\ell-1} > 0$, $p_\ell = 0$ et $p'_\ell = Q(p_{\ell-1}, t_{\ell-1})$. Donc,

$$p' = DQ(p_{\ell-1}, p_\ell, 0, t_{\ell-1}, t_\ell, 0) = DQ(p''_\ell, p_\ell, p'''_\ell, t''_\ell, t_\ell, t'''_\ell).$$

- Pour $i \in \{2, \ldots, \ell-1\}$, on a $p'_i = q$. De plus, au pas k, la tête de lecture ne pouvait qu'observer les cases $i - 1$, i ou $i + 1$. On a trois possibilités :
  - Si la tête de lecture observait la case $i - 1$ au pas k, alors $D(p_i, t_i) = R$, $p_{i-1} > 0$, $p_{i+1} = p_i = 0$ et $p'_i = Q(p_{i-1}, t_{i-1})$. Donc,

$$p' = DQ(p_{i-1}, p_i, p_{i+1}, t_{i-1}, t_i, t_{i+1}) = DQ(p''_i, p_i, p'''_i, t''_i, t_i, t'''_i).$$

  - Si la tête de lecture observait la case i au pas k, alors $D(p_i, t_i) = S$, $p_i > 0$, $p_{i-1} = p_{i+1} = 0$ et $p'_i = Q(p_i, t_i)$. Donc,

$$p' = DQ(p_{i-1}, p_i, p_{i+1}, t_{i-1}, t_i, t_{i+1}) = DQ(p''_i, p_i, p'''_i, t''_i, t_i, t'''_i).$$

  - Si la tête de lecture observait la case $i + 1$ au pas k, alors $D(p_{i+1}, t_{i+1}) = L$, $p_{i+1} > 0$, $p_i = p_{i-1} = 0$ et $p'_i = Q(p_{i+1}, t_{i+1})$. Donc,

$$p' = DQ(p_{i-1}, p_i, p_{i+1}, t_{i-1}, t_i, t_{i+1}) = DQ(p''_i, p_i, p'''_i, t''_i, t_i, t'''_i).$$

On constate que pour tout $i \in \{1, \ldots, \ell\}$,

$$p'_i = DQ(p''_i, p_i, p'''_i, t''_i, t_i, t'''_i).$$

Donc,

$$p' = \sum_{i=1}^{\ell} DQ(p''_i, p_i, p'''_i, t''_i, t_i, t'''_i).$$

Puisque DQ est une endofonction, on peut alors conclure que

$$p' = DQ[\beta](p\beta, p, p \text{ div } \beta, t\beta, t, t \text{ div } \beta, \ell).$$

Il existe donc ℓ tel que

$$p' = DQ[\beta](p\beta, p, p \text{ div } \beta, t\beta, t, t \text{ div } \beta, \ell).$$

($\Leftarrow$) Soit ℓ tel que

$$p' = DQ[\beta](p\beta, p, p \text{ div } \beta, t\beta, t, t \text{ div } \beta, \ell).$$

Pour tout $i \in \{1, \ldots, \ell\}$, on a alors

$$p'_i = DQ(p''_i, p_i, p'''_i, t''_i, t_i, t'''_i).$$

D'après le lemme 6.7.2, pour $i \in \{2, \ldots, \ell - 1\}$, on a $p''_i = p_{i-1}$, $p'''_i = p_{i+1}$, $t''_i = t_{i-1}$ et $t'''_i = t_{i+1}$. De plus, $p''_1 = 0$, $p''_\ell = p_{\ell-1}$, $p'''_1 = t_2$, $p'''_\ell = 0$, $t''_1 = 0$, $t''_\ell = t_{\ell-1}$, $t'''_1 = t_2$ et $t'''_\ell = 0$.

Soit i la position de la tête de lecture au pas k. On a alors $i \in \{1, \ldots, \ell - 1\}$. En effet, $p\beta < \beta^\ell$, donc $p < \beta^{\ell-1}$ ce qui implique que $p_\ell = 0$. Considérons dans un premier temps le cas où $i = 1$. Pour $i = 1$, on a $p_1 = q$ et $p_j = 0$ pour tout $j \in \{2, \ldots, \ell\}$. Donc, on a $p''_2 = q$ et $p''_j = 0$ pour tout $j \in \{1, \ldots, \ell\}$ et différent de 2. On a également $p'''_j = 0$ pour tout $i \in \{1, \ldots, \ell\}$. Ceci implique que pour tout $j \in \{3, \ldots, \ell\}$, on a

$$DQ(p''_j, p_j, p'''_j, t''_j, t_j, t'''_j) = DQ(0, 0, 0, t''_j, t_j, t'''_j) = 0.$$

- Si $D(p_1, t_1) = S$, alors

$$DQ(p''_1, p_1, p'''_1, t''_1, t_1, t'''_1) = DQ(0, q, 0, 0, t_1, t_2) = Q(p_1, t_1)$$

$$DQ(p''_2, p_2, p'''_2, t''_2, t_2, t'''_2) = DQ(q, 0, 0, t_1, t_2, t_3) = 0.$$

  Puisque $D(q, t_1) = S$, alors la tête de lecture observe la position 1 de la mémoire au pas $k + 1$. L'état de la machine au pas $k + 1$ est donné par $Q(q, t_1)$. Donc, p' représente l'état et le positionnement de la tête de lecture de M au pas $k + 1$, d'où, par définition, $p' = NextP(p, t)$.

- Si $D(p_1, t_1) = R$, alors

$$DQ(p''_1, p_1, p'''_1, t''_1, t_1, t'''_1) = DQ(0, q, 0, 0, t_1, t_2) = 0$$

$$DQ(p''_2, p_2, p'''_2, t''_2, t_2, t'''_2) = DQ(q, 0, 0, t_1, t_2, t_3) = Q(q, t_1).$$

  Puisque $D(q, t_1) = R$, alors la tête de lecture observe la position 2 de la mémoire au pas $k + 1$. L'état de la machine au pas $k + 1$ est donné par $Q(q, t_1)$. Donc, p'

représente l'état et le positionnement de la tête de lecture de M au pas $k+1$, d'où, par définition, $p' = NextP(p,t)$.

Notons qu'il est impossible que $D(p_1,t_1) = L$, puisque cela positionnerait la tête de lecture en dehors de la mémoire.

Considérons maintenant le cas où $i \in \{2,\ldots,\ell-1\}$. Pour $i \in \{2,\ldots,\ell-1\}$, on a $p_i = q$ et $p_j = 0$ pour tout $j \in \{1,\ldots,\ell\}$ et différent de i. Donc, on a $p''_{i+1} = q$ et $p''_j = 0$ pour tout $j \in \{1,\ldots,\ell\}$ et différent de $i+1$. On a également $p'''_{i-1} = q$ et $p'''_j = 0$ pour tout $j \in \{1,\ldots,\ell\}$ et différent de $i-1$. Ceci implique que pour tout $j \in \{1,\ldots,\ell\}$ et différent de $i-1$, i et $i+1$ on a

$$DQ(p''_j, p_j, p'''_j, t''_j, t_j, t'''_j) = DQ(0,0,0,t''_j,t_j,t'''_j) = 0.$$

- Si $D(p_i,t_i) = S, alors$

$$DQ(p''_{i-1}, p_{i-1}, p'''_{i-1}, t''_{i-1}, t_{i-1}, t'''_{i-1}) = DQ(0,0,q,t_{i-2},t_{i-1},t_i) = 0,$$

$$DQ(p''_i, p_i, p'''_i, t''_i, t_i, t'''_i) = DQ(0,q,0,t_{i-1},t_i,t_{i+1}) = Q(q,t_i)$$

$$DQ(p''_{i+1}, p_{i+1}, p'''_{i+1}, t''_{i+1}, t_{i+1}, t'''_{i+1}) = DQ(q,0,0,t_i,t_{i+1},t_{i+2}) = 0$$

  Puisque $D(q,t_1) = S$, alors la tête de lecture observe la position i de la mémoire au pas $k+1$. L'état de la machine au pas $k+1$ est donné par $Q(q,t_i)$. Donc, p' représente l'état et le positionnement de la tête de lecture de M au pas $k+1$, d'où, par définition, $p' = NextP(p,t)$.

- Si $D(p_i,t_i) = R$, alors

$$DQ(p''_{i-1}, p_{i-1}, p'''_{i-1}, t''_{i-1}, t_{i-1}, t'''_{i-1}) = DQ(0,0,q,t_{i-2},t_{i-1},t_i) = 0$$

$$DQ(p''_i, p_i, p'''_i, t''_i, t_i, t'''_i) = DQ(0,q,0,t_{i-1},t_i,t_{i+1}) = 0$$

$$DQ(p''_{i+1}, p_{i+1}, p'''_{i+1}, t''_{i+1}, t_{i+1}, t'''_{i+1}) = DQ(q,0,0,t_i,t_{i+1},t_{i+2}) = Q(q,t_i)$$

  Puisque $D(q,t_i) = R$, alors la tête de lecture observe la position $i+1$ de la mémoire au pas $k+1$. L'état de la machine au pas $k+1$ est donné par $Q(q,t_i)$. Donc, p' représente l'état et le positionnement de la tête de lecture de M au pas $k+1$, d'où, par définition, $p' = NextP(p,t)$.

- Si $D(p_i, t_i) = L$, alors

$$DQ(p''_{i-1}, p_{i-1}, p'''_{i-1}, t''_{i-1}, t_{i-1}, t'''_{i-1}) = DQ(0, 0, q, t_{i-2}, t_{i-1}, t_i) = Q(q, t_i)$$

$$DQ(p''_i, p_i, p'''_i, t''_i, t_i, t'''_i) = DQ(0, q, 0, t_{i-1}, t_i, t_{i+1}) = 0$$

$$DQ(p''_{i+1}, p_{i+1}, p'''_{i+1}, t''_{i+1}, t_{i+1}, t'''_{i+1}) = DQ(q, 0, 0, t_i, t_{i+1}, t_{i+2}) = 0.$$

  Puisque $D(q, t_i) = L$, alors la tête de lecture observe la position $i-1$ de la mémoire au pas $k+1$. L'état de la machine au pas $k+1$ est donné par $Q(q, t_i)$. Donc, p' représente l'état et le positionnement de la tête de lecture de M au pas $k+1$, d'où, par définition, $p' = NextP(p, t)$.

Dans tous les cas, on a $p' = DQ[\beta](p\beta, p, p \text{ div } \beta, t\beta, t, t \text{ div } \beta) = NextP(p, t)$.
Donc, on a bien l'équivalence voulue. □

Les fonctions $NextP$ et $NextT$, qui permettent d'obtenir, à partir des codes de configuration d'une machine de Turing M au pas k, les codes de configuration de M au pas $k+1$ sont donc diophantiennes.

On est maintenant en mesure de simuler un nombre fini de pas d'une machine de Turing M à partir d'une configuration initiale. Par exemple, si (p_0, t_0) est un code de configuration de M, alors le système

$$\begin{cases} p_1 = NextP(p_0, t_0), & t_1 = NextT(p_0, t_0) \\ p_2 = NextP(p_1, t_1), & t_2 = NextT(p_1, t_1) \\ \vdots & \vdots \\ p_k = NextP(p_{k-1}, t_{k-1}), & t_k = NextT(p_{k-1}, t_{k-1}) \end{cases}$$

permet d'obtenir le code de configuration de M au pas k. Mais il faut connaître k au préalable. Or, le but est ici de prouver que les ensembles semi-décidables, c'est-à-dire les ensembles reconnus par les machines de Turing, sont diophantiens. On désire donc une méthode permettant de construire à partir d'une machine de Turing M quelconque un système d'équations diophantiennes paramétrées. Ce système sera tel qu'il admettra une solution si et seulement si les paramètres qui lui sont passés codent un élément

appartenant à l'ensemble reconnu par M. La simulation de M est un moyen de parvenir à ce résultat, mais on ne peut pas prédire au préalable le nombre de pas nécessaires à M pour reconnaître un élément. De plus, dans le cas où l'élément n'est pas reconnu, M ne s'arrête jamais. Il est nécessaire de se munir de nouvelles fonctions, capables de simuler un nombre quelconque de pas de la machine.

Définition 6.7.6. *Soient* $AfterP : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *et* $AfterT : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *les fonctions définies comme suit.*

$$\begin{aligned} AfterP(0,p,t) &= p \\ AfterP(k+1,p,t) &= AfterP(k, NextP(p,t), NextT(p,t)) \end{aligned}$$

$$\begin{aligned} AfterT(0,p,t) &= t \\ AfterT(k+1,p,t) &= AfterT(k, NextT(p,t), NextT(p,t)). \end{aligned}$$

Lemme 6.7.4. *Soit* M *une machine de Turing à un pas* i *quelconque de son exécution. Soient* (p,t) *un code de configuration de* M *en base* β. *Si* $p' = AfterP(k,p,t)$ *et* $t' = AfterT(k,p,t)$, *alors* p' *code la position de la tête de lecture et l'état de* M *au pas* $i+k$ *de* M *et* t' *code la mémoire de* M *au pas* $i+k$ *de* M.

Démonstration (*). Prouvons dans un premier temps que p' code la position de la tête de lecture et l'état de M après k pas.

- Si $k = 0$, alors par définition, $p' = p$ et la propriété est vérifiée. Supposons qu'elle reste vérifiée pour $k \leq n$. Prouvons qu'elle est encore vérifiée pour $k = n+1$.
- Si $k = n+1$, alors par définition,

  $$p' = AfterP(n+1,p,t) = AfterP(n, NextP(p,t), NextT(p,t)).$$

  Soit $p'' = NextP(p,t)$ et $t'' = NextT(p,t)$. Par définition de $NextP$ et $NextT$, p'' code la position de la tête de lecture et l'état de M au pas $i+1$ et t'' code la mémoire de M au pas $i+1$. Par hypothèse d'induction, on déduit alors que p' code la position de la tête de lecture et l'état de M au pas $i+1+n = i+k$. La propriété se vérifie donc pour tout $k \geq 0$.

De la même manière, on prouve aisément que $t' = AfterT(k, p, t)$ code la mémoire de M au pas $i + k$ pour tout $k \geq 0$. □

Il reste à prouver que $AfterP$ et $AfterT$ sont diophantiennes. Une caractéristique intéressante des fonctions $NextP$ et $NextT$ est que ces fonctions sont capables de traiter plusieurs configurations en parallèle. Rappelons qu'au chapitre 3, nous avons décidé d'utiliser l'opérateur + pour représenter la fonction qui aux codes positionnels (a, b, c) et (a', b, c') représentant deux uplets $\mathbf{a}$ et $\mathbf{a}'$ associe le code positionnel $(a'', b, c + c')$ représentant l'uplet issu de la concaténation de $\mathbf{a}$ et $\mathbf{a}'$.

Lemme 6.7.5. *Soient $(p_1, t_1), \ldots, (p_k, t_k)$ des couples de configurations d'une machine de Turing M codant respectivement les configurations $c_1, \ldots, c_k$. Soient $\ell_1, \ldots, \ell_k$ des longueurs de mémoires supérieures aux longueurs de mémoire utilisées respectivement par les configurations consécutives à $c_1, \ldots, c_k$. Soit $\ell = \ell_1 + \cdots + \ell_k$. Soit $(a, \beta, \ell) = (p_1, \beta, \ell_1) + \cdots + (p_k, \beta, \ell_k)$. Soit $(b, \beta, \ell) = (t_1, \beta, \ell_1) + \cdots + (t_k, \beta, \ell_k)$. Alors on a*

$$(NextP(a, b), \beta, \ell) = (NextP(p_1, t_1), \beta, \ell_1) + \cdots + (NextP(p_k, t_k)\beta, \ell_k)$$

$$(NextT(a, b), \beta, \ell) = (NextT(p_1, t_1), \beta, \ell_1) + \cdots + (NextT(p_k, t_k), \beta, \ell_k).$$

Démonstration (*). Soient $a_1, \ldots, a_\ell, b_1, \ldots, b_\ell$ tels que

$$a = \sum_{i=1}^{\ell} a_i \beta^{i-1}$$

$$b = \sum_{i=1}^{\ell} b_i \beta^{i-1}.$$

Soient $a'_1, \ldots, a'_l, b'_1, \ldots, b'_l$ tels que

$$NextP(a, b) = \sum_{i=1}^{\ell} a'_i \beta^{i-1}$$

et

$$NextT(a, b) = \sum_{i=1}^{\ell} b'_i \beta^{i-1}.$$

Soient $p_{1,1},\ldots,p_{1,\ell_1},\ldots,p_{k,1},\ldots,p_{k,\ell_k}$ tels que

$$\begin{aligned} p_1 &= \sum_{i=1}^{\ell_1} p_{1,i}\beta^{i-1} \\ &\vdots \\ p_k &= \sum_{i=1}^{\ell_k} p_{k,i}\beta^{i-1}. \end{aligned}$$

Soient $t_{1,1},\ldots,t_{1,\ell_1},\ldots,t_{k,1},\ldots,t_{k,\ell_k}$ tels que

$$\begin{aligned} t_1 &= \sum_{i=1}^{\ell_1} t_{1,i}\beta^{i-1} \\ &\vdots \\ t_k &= \sum_{i=1}^{\ell_k} t_{k,i}\beta^{i-1}. \end{aligned}$$

Soient $p'_{1,1},\ldots,p'_{1,\ell_1},\ldots,p'_{k,1},\ldots,p'_{k,\ell_k}$ tels que

$$\begin{aligned} NextP(p_1,t_1) &= \sum_{i=1}^{\ell_1} p'_{1,i}\beta^{i-1} \\ &\vdots \\ NextP(p_k,t_k) &= \sum_{i=1}^{\ell_k} p'_{k,i}\beta^{i-1}. \end{aligned}$$

Soient $t'_{1,1},\ldots,t'_{1,\ell_1},\ldots,t'_{k,1},\ldots,t'_{k,\ell_k}$ tels que

$$\begin{aligned} NextT(p_1,t_1) &= \sum_{i=1}^{\ell_1} t'_{1,i}\beta^{i-1} \\ &\vdots \\ NextT(p_k,t_k) &= \sum_{i=1}^{\ell_k} t'_{k,i}\beta^{i-1}. \end{aligned}$$

Montrons que

$$(NextT(b),\beta,\ell) = (NextT(p_1,t_1),\beta,\ell_1) + \cdots + (NextT(p_k,t_k),\beta,\ell_k).$$

Soit $i \in \{1,\ldots,k\}$. Soit $x = \sum_{c=0}^{i-1} \ell_i$. Soit $b' = A'[\beta](a,b,\ell)$. Alors, par définition, on a $b' = NextT(a,b)$. On a, par unicité de la représentation en base β d'un nombre, pour

tout $y \in \{1, \ldots, \ell_i\}$, $p_{i,y} = a_{x+y}$ et $t_{i,y} = b_{x+y}$. On a alors

$$b'_{x+y} = A'(a_{x+y}, b_{x+y}) = A'(p_{i,y}, t_{i,y}) = p'_{i,y}.$$

L'uplet (b', β, ℓ) code l'uplet

$$\begin{aligned}(b'_1, \ldots, b'_\ell) &= (p'_{1,1}, \ldots, p'_{1,\ell_1}, \ldots, p'_{k,1}, \ldots, p'_{k,\ell_k}) \\ &= (p'_{1,1}, \ldots, p'_{1,\ell_1}) + \cdots + (p'_{k,1}, \ldots, p'_{k,\ell_k}).\end{aligned}$$

On peut donc conclure que

$$(NextT(a,b), \beta, \ell) = (NextT(p_1, t_1), \beta, \ell_1) + \ldots (NextT(p_k, t_k), \beta, \ell_k).$$

Montrons maintenant que

$$(NextP(b), \beta, l) = (NextP(p_1, t_1), \beta, \ell_1) + \cdots + (NextP(p_k, t_k), \beta, \ell_k).$$

Soit $i \in \{1, \ldots, k\}$. Soit $x = \sum_{c=0}^{i-1} \ell_i$. Soit $a' = DQ[\beta](a\beta, a, a \text{ div } \beta, b\beta, b, b \text{ div } \beta)$. Par définition, on a alors $a' = NextP(a,b)$. On a, par unicité de la représentation en base β d'un nombre, que $p_{i,y} = a_{x+y}$ et $t_{i,y} = b_{x+y}$ pour tout $y \in 1, \ldots, \ell_i$. Puisque ℓ_k est la longueur utilisée de la mémoire pour la configuration (p_k, t_k), alors $p_{k,\ell_k} = a_\ell = 0$ et $t_{k,\ell_k} = b_\ell = 0$. Ceci implique que $a < \beta^{\ell-1}$ et $b < \beta^{\ell-1}$, donc que $\beta a < \beta^\ell$ et $\beta b < \beta^\ell$. Par conséquent, $(\beta a, \beta, \ell)$ et $(\beta b, \beta, \ell)$ sont des codes positionnels.
Soient $a''_1, \ldots, a''_\ell, a'''_1, \ldots, a'''_\ell, b''_1, \ldots, b''_\ell, b'''_1, \ldots, b'''_\ell$ tels que

$$\beta a = \sum_{j=1}^{\ell} a''_j \beta^{j-1},$$

$$a \text{ div } \beta = \sum_{j=1}^{\ell} a'''_j \beta^{j-1},$$

$$\beta b = \sum_{j=1}^{\ell} b''_j \beta^{j-1}$$

et

$$b \text{ div } \beta = \sum_{j=1}^{\ell} b'''_j \beta^{j-1}.$$

D'après le lemme 6.7.2,

- pour $j \in \{2, \ldots, \ell - 1\}$, on a $a''_j = a_{j-1}$, $a'''_j = a_{j+1}$, $b''_j = b_{j-1}$ et $b'''_j = b_{j+1}$.
- pour $j = 1$, on a $a''_j = b''_j = 0$, $a'''_j = a_{j+1}$ et $b'''_j = b_{j+1}$.
- pour $j = \ell$, on a $a''_j = a_{j-1}$, $b''_j = b_{j-1}$ et $a'''_j = b'''_j = 0$

Soient $p''_{1,1}, \ldots, p''_{1,\ell_1}, \ldots, p''_{k,1}, \ldots, p''_{k,\ell_k}$, $p'''_{1,1}, \ldots, p'''_{1,\ell_1}, \ldots, p'''_{k,1}, \ldots, p'''_{k,\ell_k}$, $t''_{1,1}, \ldots, t''_{1,\ell_1}$, $t''_{2,1}, \ldots, t''_{k,1}, \ldots, t''_{k,\ell_k}$, $t'''_{1,1}, \ldots, t'''_{1,\ell_1}, \ldots, t'''_{k,1}, \ldots, t'''_{k,\ell_k}$ tels que, pour tout $j \in \{1, \ldots, k\}$,

$$\begin{aligned} \beta p_j &= \sum_{c=1}^{\ell_j} p''_{j,c} \beta^{c-1} \\ p_j \text{ div } \beta &= \sum_{c=1}^{\ell_j} p'''_{j,c} \beta^{c-1} \\ \beta t_j &= \sum_{c=1}^{\ell_j} t''_{j,c} \beta^{c-1} \\ t_j \text{ div } \beta &= \sum_{c=1}^{\ell_j} t'''_{j,c} \beta^{c-1}. \end{aligned}$$

D'après le lemme 6.7.2, pour tout $j \in \{1, \ldots, k\}$,

- pour $c \in \{2, \ldots, \ell_j - 1\}$, on a $p''_{j,c} = p_{j,c-1}$, $p'''_{j,c} = p_{j,c+1}$, $t''_{j,c} = t_{j,c-1}$ et $t'''_{j,c} = t_{j,c+1}$
- pour $c = 1$, on a $p''_{j,c} = t''_{j,c} = 0$, $p'''_{j,c} = p_{j,c+1}$ et $t'''_{j,c} = t_{j,c+1}$.
- pour $c = \ell_j$, on a $p''_{j,c} = p_{j,c-1}$, $t''_{j,c} = t_{j,c-1}$ et $p'''_{j,c} = t'''_{j,c} = 0$.

On va maintenant étudier le comportement de a_{x+y} en fonction de la valeur de y.

- Pour $y = 1$, on a $p_{i,y} = a_{x+y}$, $p'''_{i,y} = p_{i,y+1} = a_{x+y+1} = a'''_{x+y}$, $t_{i,y} = b_{i,y}$ et $t'''_{i,y} = t_{i,y+1} = b_{x+y+1} = b'''_{x+y}$. Si $i = 1$, alors $x + y = 1$ et on a $p''_{i,y} = 0 = a''_{x+y}$ et $t''_{i,y} = 0 = b''_{x+y}$. Si $i \in \{2, \ldots, k\}$, alors $a_{x+y-1} = p_{i-1,\ell_{i-1}} = 0 = p''_{i,y}$ et $b_{x+y-1} = t_{i-1,\ell_{i-1}} = 0 = t''_{i,y}$ puisque le ℓ_i-ème élément de l'uplet (p_i, β, ℓ_i) est nul par hypothèse.

  On a donc

$$DQ(a''_{x+y}, a_{x+y}, a'''_{x+y}, b''_{x+y}, b_{x+y}, b'''_{x+y}) = DQ(p''_{i,y}, p_{i,y}, p'''_{i,y}, t''_{i,y}, t_{i,y}, t'''_{i,y}).$$

- Pour $y = \ell_i$, on a $p_{i,y} = a_{x+y}$, $p''_{i,y} = p_{i,y-1} = a_{x+y-1} = a''_{x+y}$, $t_{i,y} = b_{x+y}$ et $t''_{i,y} = t_{i,y-1} = t_{x+y-1} = t''_{x+y}$. Si $i = k$, alors $x + y = \ell$ et on a $p'''_{i,y} = 0 =$

a'''_{x+y} et $t'''_{i,y} = 0 = b'''_{x+y}$. Si $i \in \{1, \ldots, k-1\}$, alors $a'''_{x+y} = a_{x+y+1} = p_{i+1,1}$ et $b'''_{x+y} = b_{x+y+1} = t_{i+1,1}$. Or, puisque p_{i+1} représente le contenu de la première case de la mémoire dans la configuration (p_i, t_i), alors $D(p_{i+1,1}, t_{i+1,1}) \neq L$, puisque la tête de lecture observant le contenu de la première case de la mémoire ne pourrait se déplacer vers la gauche. Par définition de DQ, on a alors

$$\begin{aligned} DQ(a''_{x+y}, a_{x+y}, a'''_{x+y}, b''_{x+y}, b_{x+y}, b'''_{x+y}) &= DQ(a''_{x+y}, a_{x+y}, 0, b''_{x+y}, b_{x+y}, 0) \\ &= DQ(p''_{x+y}, p_{x+y}, p'''_{x+y}, t''_{x+y}, t_{x+y}, t'''_{x+y}). \end{aligned}$$

Dans tous les cas, on constate que pour $i \in \{1 \ldots, k\}$ et $y \in \{1, \ldots, \ell_k\}$, si $x = \sum_{c=1}^{i-1} \ell_i + 1$, alors $a'_{x+y} = p'_{i,y}$. Donc, (p', β, ℓ) est un code positionnel de l'uplet

$$\begin{aligned} (a'_1, \ldots, a'_\ell) &= (p'_{1,1}, \ldots, p'_{1,\ell_1}, \ldots, p'_{k,1}, \ldots, p'_{k,\ell_k}) \\ &= (p'_{1,1}, \ldots, p'_{1,\ell_1}) + \cdots + (p'_{k,1}, \ldots, p'_{k,\ell_k}). \end{aligned}$$

Par conséquent, on a

$$(NextP(a,b), \beta, \ell) = (NextP(a_1, b_1), \beta, \ell_1) + \cdots + (NextP(a_k, b_k), \beta, \ell_k).$$

□

Le lemme précédent signifie en fait qu'il est possible d'appliquer les fonctions $NextP$ et $NextT$ sur plusieurs configurations simultanément. Il suffit pour cela de concaténer ces configurations en utilisant des longueurs pour les uplets qui satisfassent les contraintes du lemme. On utilise cette propriété pour prouver que $AfterP$ et $AfterT$ sont diophantiennes.

Constatons au préalable que selon le codage choisi, le premier élément de tout uplet codé par t ou p représentant respectivement la mémoire ou l'état et la position de la tête de lecture d'une machine de Turing est le code du symbole $\star$. De plus, si ℓ doit être tel que (p, β, ℓ) et $(NextP(p,t), \beta, \ell)$ soient des codes positionnels dans tous les cas, alors il faut que $t < \beta^{\ell-1}$, c'est-à-dire que le dernier élément de l'uplet codé par (t, β, ℓ) soit nul. Ceci permet de gérer le cas où la tête de lecture observe la position ℓ de la mémoire et se décale à droite au pas suivant.

Lemme 6.7.6. *Soit (p,t) un code de configuration d'une machine de Turing M au pas i. Soit $k \geq 0$. Soit ℓ tel que $p < \beta^{\ell-k-2}$ et $p < \beta^{\ell-k-2}$. Alors le système de contraintes diophantiennes*

$$\begin{aligned}
p' &< \beta^{\ell-2}, \\
t' &< \beta^{\ell-2}, \\
(p_L, \beta, k\ell) &= (p, \beta, \ell) + (p_M, \beta, (k-1)\ell), \\
(t_L, \beta, k\ell) &= (t, \beta, \ell) + (t_M, \beta, (k-1)\ell), \\
p_R = NextP(p_L, t_L) &\quad , \quad t_R = NextT(p_L, t_L), \\
p_R &= (p_M, \beta, (k-1)\ell) + (p', \beta, \ell), \\
t_R &= (t_M, \beta, (k-1)\ell) + (t', \beta, \ell)
\end{aligned}$$

possède une unique solution telle que

$$\begin{aligned}
p_L &= (p_0, \beta, \ell) + \cdots + (p_{k-1}, \beta, \ell), \\
t_L &= (t_0, \beta, \ell) + \cdots + (t_{k-1}, \beta, \ell), \\
p_M &= (p_1, \beta, \ell) + \cdots + (p_{k-1}, \beta, \ell), \\
t_M &= (t_1, \beta, \ell) + \cdots + (t_{k-1}, \beta, \ell), \\
p_R &= (p_1, \beta, \ell) + \cdots + (p_k, \beta, \ell), \\
t_R &= (t_1, \beta, \ell) + \cdots + (t_k, \beta, \ell)
\end{aligned}$$

avec $p_0 = p$, $t_0 = t$ et $(p_1, t_1), \ldots, (p_k, t_k)$ les codes de configuration de la machine aux pas $i+1, \ldots, i+k$, $p_k = p'$ et $t_k = t'$.

Démonstration. Montrons dans un premier temps que

$$\begin{aligned}
(p_L, \beta, k\ell) &= (p_0, \beta, \ell) + \cdots + (p_{k-1}, \beta, \ell), \\
(t_L, \beta, k\ell) &= (t_0, \beta, \ell) + \cdots + (t_{k-1}, \beta, \ell), \\
(p_M, \beta, (k-1)\ell) &= (p_1, \beta, \ell) + \cdots + (p_{k-1}, \beta, \ell), \\
(t_M, \beta, (k-1)\ell) &= (t_1, \beta, \ell) + \cdots + (t_{k-1}, \beta, \ell), \\
(p_R, \beta, k\ell) &= (p_1, \beta, \ell) + \cdots + (p_k, \beta, \ell), \\
(t_R, \beta, k\ell) &= (t_1, \beta, \ell) + \cdots + (t_k, \beta, \ell)
\end{aligned}$$

est une solution du système d'équations diophantiennes proposé. Par définition de p_M et t_M, on a

$$\begin{aligned}
(p_L, \beta, k\ell) &= (p, \beta, \ell) + (p_M, \beta, (k-1)\ell), \\
(t_L, \beta, k\ell) &= (t, \beta, \ell) + (t_M, \beta, (k-1)\ell), \\
(p_R, \beta, k\ell) &= (p_M, \beta, (k-1)\ell) + (p', \beta, \ell), \\
(t_R, \beta, k\ell) &= (t_M, \beta, (k-1)\ell) + (t', \beta, \ell).
\end{aligned}$$

Puisque $t, p < \beta^{\ell-k-2}$, alors pour tout $j \in \{0, \ldots, k\}$, on a $t_j, p_j < \beta^{\ell-k-2+j} < \beta^l - 1$. En effet, puisque $t, p < \beta^{\ell-k-2}$, alors les éléments $(\ell - k - 2 + 1)$ à ℓ des uplets codés par (p, β, ℓ) et (t, β, ℓ) sont nuls. Puisqu'à chaque pas, la machine de Turing ne peut que remplir qu'une nouvelle case de la mémoire et que (p_j, t_j) code l'état de la machine au pas $i + j$, alors les éléments $\ell - k - 2 + 1 + j$ à ℓ des uplets (p_j, β, ℓ) et (t_j, β, ℓ) sont nuls. En particulier, on a $p' = p_k < \beta^{k-2}$ et $t' = t_k < \beta^{k-2}$. Donc, pour tout $j \in \{0, \ldots, k\}$, le dernier élément de l'uplet codé par (p_j, β, ℓ) et le dernier élément de l'uplet codé par (ι_j, β, ℓ) sont nuls. D'après le lemme 6.7.5, on a alors

$$(NextP(p,t), \beta, \ell) = (p_1, \beta, \ell) + \cdots + (p_k, \beta, \ell) = (p_R, \beta, \ell),$$

$$(NextT(p,t), \beta, \ell) = (t_1, \beta, \ell) + \cdots + (t_k, \beta, \ell) = (t_R, \beta, \ell).$$

On a donc $NextP(p,t) = p_R$ et $NextT(p,t) = t_R$. Toutes les contraintes diophantiennes sont donc satisfaites.

Montrons maintenant que si le système diophantien admet une solution, alors cette solution est unique. Soient $p_L, t_L, p_M, t_M, p_R, t_R, p', t'$ tels que le système de contraintes diophantiennes soit satisfait. Soient $(a_1, \ldots, a_\ell)$ et $(b_1, \ldots, b_\ell)$ les uplets codés par (p, β, ℓ) et (t, β, ℓ). Soient $(p_{L,1}, \ldots, p_{L,k\ell})$ et $(t_{L,1}, \ldots, t_{L,k\ell})$ les uplets codés par $(p_L, \beta, k\ell)$ et $(t_L, \beta, k\ell)$. Soient $(p_{R,1}, \ldots, p_{R,k\ell})$ et $(t_{R,1}, \ldots, t_{R,k\ell})$ les uplets codés par $(p_R, \beta, k\ell)$ et $(t_R, \beta, k\ell)$. Puisque $NextP(p,t) = p_R$, alors, il existe ℓ' tel que

$$p_R = DQ[\beta](p_L\beta, p_L, p_L \text{ div } \beta, p_R\beta, p_R, p_R \text{ div } \beta, \ell').$$

Soit

$$(p_{L,1}, \ldots, p_{L,\ell'})$$

l'uplet codé par (p_L, β, ℓ'). Par définition, les uplets $(p_L\beta, \beta, \ell')$ et $(p_L \text{ div } \beta, \beta, \ell')$ sont des codes positionnels qui, d'après le lemme 6.7.2, codent respectivement les uplets $(0, p_{L,1}, \ldots, p_{L,\ell'-1})$ et $(p_{L,2}, \ldots, p_{L,\ell'}, 0)$. Soient $p_{L,0} = 0$ et $p_{L,\ell'+1} = 0$. Soit

$$(t_{L,1}, \ldots, p_{t,\ell'})$$

l'uplet codé par (t_L, β, ℓ'). Par définition, $(t_L\beta, \beta, \ell')$ et $(t_L \text{ div } \beta, \beta, \ell')$ sont des codes positionnels qui, d'après le lemme 6.7.2, codent respectivement les uplets $(0, t_{L,1}, \ldots, t_{L,\ell'-1})$ et $(t_{L,2}, \ldots, t_{L,\ell'}, 0)$. Soient $p_{L,0} = t_{L,0} = 0$ et $p_{L,\ell'+1} = t_{L,\ell'+1} = 0$. Donc, pour tout $i \in \{1, \ldots, \ell'\}$, on a

$$p_{R,i} = DQ(p_{L,i-1}, p_{L,i}, p_{L,i+1}, t_{L,i-1}, t_{L,i}, t_{L,i+1}).$$

Puisque $(p_R, \beta, k\ell)$ est un code positionnel, alors si $i > k\ell$, on a $p_{R,i} = 0$. Donc, $p_{R,i}$ est uniquement déterminé par $p_{i-1}, p_i, p_{i+1}, t_{i-1}, t_i$ et t_{i+1}. Puisque $NextT(p,t) = t_R$, alors, il existe ℓ' tel que

$$t_R = A'[\beta](p_L, p_R, \ell').$$

Soit

$$(p_{L,1}, \ldots, p_{L,\ell'})$$

l'uplet codé par (p_L, β, ℓ').

Soit

$$(t_{L,1}, \ldots, p_{t,\ell'})$$

l'uplet codé par (t_L, β, ℓ').

Donc, pour tout $i \in \{1, \ldots, \ell'\}$, on a $t_{R,i} = A'(p_{L,i}, t_{L,i})$. Puisque $(t_R, \beta, k\ell)$ est un code positionnel, alors si $i > kl$, on a $t_{R,i} = 0$. Donc, $t_{R,i}$ est uniquement déterminé par $p_{L,i}$ et $t_{R,i}$. Montrons alors que pour tout $j \in \{1, \ldots, k\}$, les $\ell j - j$ premiers termes des uplets $(p_{R,1}, \ldots, p_{R,k\ell})$ et $(t_{R,1}, \ldots, t_{R,k\ell})$ sont déterminés uniquement. On procède par récurrence sur j.

- Pour $j = 1$, par définition de p_L et t_L, on a, pour tout $i \in \{1, \ldots, \ell\}$, $p_{L,i} = a_i$ et $t_{L,i} = b_i$. Puisque $p_R = NextP(p_L, t_L)$, alors les $\ell - 1$ premiers éléments de $(p_{R,1}, \ldots, p_{R,k\ell})$ sont déterminés uniquement. Puisque $t_R = NextT(p_L, t_L)$, alors les $\ell - 1$ premiers éléments de $(t_{R,1}, \ldots, t_{R,k\ell})$ sont déterminés uniquement. La propriété est vérifiée.

- Pour $j' = j+1$, on sait, par hypothèse d'induction, que les $\ell j - j$ premiers éléments de $(p_{R,1}, \ldots, p_{R,kl})$ et $(t_{R,1}, \ldots, t_{R,k\ell})$ sont déterminés uniquement.
  On a $j \leq k-1$, d'où $\ell j \leq \ell k - \ell$, donc $\ell j - j \leq \ell k - \ell$. Par définition de p_R, on sait alors que les $\ell j - j$ premiers éléments des uplets $(t_{R,1}, \ldots, t_{R,k\ell})$ et $(p_{R,1}, \ldots, p_{R,lk})$ sont respectivement les $\ell j - j$ premiers éléments des uplets $(t_{M,1}, \ldots, t_{M,k\ell})$ et $(p_{M,1}, \ldots, p_{M,k\ell})$. Par définition de t_L et p_L, on sait que les $\ell j - j$ premiers éléments de $(t_{M,1}, \ldots, t_{M,(k-1)\ell})$ et $(p_{M,1}, \ldots, p_{M,(k-1)\ell})$ sont respectivement les éléments $\ell + 1$ à $\ell + lj - j$ des uplets $(t_{L,1}, \ldots, t_{L,k\ell})$ et $(p_{L,1}, \ldots, p_{L,k\ell})$. Puisque les ℓ premiers éléments des uplets $(t_{L,1}, \ldots, t_{L,k\ell})$ et $(p_{L,1}, \ldots, p_{L,\ell k})$ sont respectivement les ℓ éléments des uplets codés par (t, β, ℓ) et (p, β, ℓ), alors on peut dire que les $\ell + \ell j - j = \ell(j+1) - j$ premiers éléments de $(t_{L,1}, \ldots, t_{L,k\ell})$ et $(p_{L,1}, \ldots, p_{L,k\ell})$ sont uniquement déterminés. Puisque $p_R = NextP(p_L, t_L)$, alors les $\ell(j+1) - j - 1 = \ell(j+1) - (j+1) = \ell j' - j'$ premiers éléments de $(p_{R,1}, \ldots, p_{R,k\ell})$ sont uniquements déterminés. Puisque $t_R = NextT(p_L, t_L)$, alors les $\ell(j+1) - j - 1 = \ell(j+1) - (j+1) = \ell j' - j'$ premiers éléments de $t_{R,1}, \ldots, p_{R,k\ell}$

sont uniquement déterminés. La propriété est vérifiée.

Donc, les $\ell k - k$ premiers éléments des uplets $(t_{R,1}, \ldots, t_{R,k\ell})$ et $(p_{R,1}, \ldots, p_{R,k\ell})$ sont uniquement déterminés. Puisque $\ell - k - 2 \geq 0$, alors $\ell \geq k + 2 \geq k$. Donc, $\ell k - k \geq \ell k - \ell$ et par définition de t_R et p_R, si les $\ell k - k$ premiers éléments de $(t_{R,1}, \ldots, t_{R,k\ell})$ et de $(p_{R,1}, \ldots, p_{R,\ell k})$ sont uniquement déterminés alors les $k\ell - \ell$ éléments de $(t_{M,1}, \ldots, t_{M,k\ell})$ et $(p_{M,1}, \ldots, p_{M,(k-1)\ell})$ sont uniquement déterminés. Par définition de t_L et p_L, on sait alors que les $\ell + k\ell - \ell = k\ell$ éléments de $(t_{L,1}, \ldots, t_{L,k\ell})$ et de $(p_{L,1}, \ldots, p_{L,lk})$ sont uniquement déterminés.

Puisque $p_R = NextP(p_L, t_L)$, alors les $k\ell - 1$ premiers éléments de $(p_{R,1}, \ldots, p_{R,k\ell})$ sont uniquement déterminés.

Puisque $t_R = nextT(p_L, t_L)$, alors les $k\ell - 1$ premiers éléments de $(t_{R,1}, \ldots, p_{R,k\ell})$ sont uniquement déterminés.

Par hypothèse, les conditions $t' < \beta^{\ell-2}$ et $p' < \beta^{\ell-2}$ sont satisfaites. Donc, le dernier élément des uplets codés par (t', β, ℓ) et (p', β, ℓ) sont nuls. Par définition de t_R et p_R, le dernier élément de l'uplet codé par (p', β, ℓ) et le dernier élément de l'uplet codé par (t', β, ℓ) sont respectivements le derniers éléments des uplets codés par $(p_R, \beta, k\ell)$ et $(t_R, \beta, k\ell)$. Ces éléments sont donc nuls dans tous les cas. Puisque les $k\ell$ éléments de $(p_{L,1}, \ldots, p_{L,kl})$ et de $(t_{L,1}, \ldots, t_{L,k\ell})$ sont uniquement déterminés, alors par définition de p_R et t_R, les ℓ éléments des uplets codés par $(p'.\beta, \ell)$ et (t', β, ℓ) sont uniquement déterminés. Par conséquent, les kl éléments des uplets $(t_{L,1}, \ldots, t_{L,k\ell})$ et $(p_{L,1}, \ldots, p_{L,k\ell})$, les $(k-1)\ell$ éléments des uplets $(t_{M,1}, \ldots, t_{M,(k-1\ell)})$ et $(p_{M,1}, \ldots, p_{M,(k-1)\ell})$, les $k\ell$ éléments des uplets $(t_{R,1}, \ldots, t_{R,k\ell})$ et $(p_{R,1}, \ldots, p_{R,k\ell})$ et les ℓ éléments des uplet codés par (t', β, ℓ) et (p', β, ℓ) sont uniquements déterminés par p, t et ℓ. Donc, pour p, t et ℓ donnés, il n'existe qu'une seule solution possible satisfaisant le système d'équations

diophantienne. Or, on a vu que

$$\begin{aligned}
p_L &= (p_0, \beta, \ell) + \cdots + (p_{k-1}, \beta, \ell), \\
t_L &= (t_0, \beta, \ell) + \cdots + (t_{k-1}, \beta, \ell), \\
p_M &= (p_1, \beta, \ell) + \cdots + (p_{k-1}, \beta, \ell), \\
t_M &= (t_1, \beta, \ell) + \cdots + (t_{k-1}, \beta, \ell), \\
p_R &= (p_1, \beta, \ell) + \cdots + (t_k, \beta, \ell), \\
t_R &= (p_1, \beta, \ell) + \cdots + (t_k, \beta, \ell)
\end{aligned}$$

est une solution de ce système. Par conséquent, cette solution seule satisfait le système.

□

Lemme 6.7.7. *Soit (p,t) la configuration d'une machine de Turing M à un pas i quelconque de son exécution. Alors on a*

$$p' = AfterP(p,t,k)$$

$$\Leftrightarrow$$

$$\exists \ell \exists t' \exists p_L \exists t_L \exists p_M \exists t_M \exists p_R \exists t_R [p' < \beta^{\ell-2} \wedge t' < \beta^{\ell-2}$$

$$\wedge\ p_R = NextP(p_L, t_L) \wedge t_R = NextT(p_L, t_L)$$

$$\wedge\ (p_L, \beta, k\ell) = (p, \beta, \ell) + (p_M, \beta, (k-1)\ell)$$

$$\wedge\ (t_L, \beta, k\ell) = (t, \beta, \ell) + (t_M, \beta, (k-1)\ell)$$

$$\wedge\ (p_R, \beta, k\ell) = (p_M, \beta, (k-1)\ell) + (p', \beta, \ell)$$

$$\wedge\ (t_R, \beta, k\ell) = (t_M, \beta, (k-1)\ell) + (t', \beta, \ell)].$$

Démonstration (*). Soit (p,t) la configuration d'une machine de Turing M à un pas i quelconque de son exécution.

($\Rightarrow$) Par définition, p' code l'état de M et la position de la tête de lecture au pas $i+k$. Soient $(p_1, t_1), \ldots, (p_k, t_k)$ des codes de configuration de M tels que pour tout j compris

entre 1 et k, (p_j, t_j) code la configuration de M à l'état $i+j$. Donc $p' = p_k$. Soit $t' = t_k$. Soit ℓ tel que $p < \beta^{\ell-k-2}$ et $t < \beta^{\ell-k-2}$. Soit p_L tel que

$$(p_L, \beta, k\ell) = (p, \beta, \ell) + (p_1, \beta, \ell) + \cdots + (p_{k-1}, \beta, \ell).$$

Soit t_L tel que

$$(t_L, \beta, k\ell) = (t, \beta, \ell) + (t_1, \beta, \ell) + \cdots + (t_{k-1}, \beta, \ell).$$

Soit p_R tel que

$$(p_R, \beta, k\ell) = (p_1, \beta, \ell) + \cdots + (p_k, \beta, \ell).$$

Soit t_R tel que

$$(t_R, \beta, k\ell) = (t_1, \beta, \ell) + \cdots + (t_k, \beta, \ell).$$

D'après le lemme 6.7.6, les conditions

$$\begin{aligned}
p' &< \beta^{\ell-2} \\
t' &< \beta^{\ell-2} \\
p_R &= NextP(p_L, t_L) \\
t_R &= NextT(p_L, t_L) \\
(p_L, \beta, k\ell) &= (p, \beta, \ell) + (p_M, \beta, (k-1)\ell) \\
(t_L, \beta, k\ell) &= (t, \beta, \ell) + (t_M, \beta, (k-1)\ell) \\
(p_R, \beta, k\ell) &= (p_M, \beta, (k-1)\ell) + (p', \beta, \ell) \\
(t_R, \beta, k\ell) &= (t_M, \beta, (k-1)\ell) + (t', \beta, \ell)
\end{aligned}$$

sont satisfaites.

($\Leftarrow$) Supposons que $p', t', p_L, t_L, p_M, t_M, p_R, t_R$ soient tels que les conditions

$$\begin{aligned}
p' &< \beta^{\ell-2} \\
t' &< \beta^{\ell-2} \\
p_R &= NextP(p_L, t_L) \\
t_R &= NextT(p_L, t_L) \\
(p_L, \beta, k\ell) &= (p, \beta, \ell) + (p_M, \beta, (k-1)\ell) \\
(t_L, \beta, k\ell) &= (t, \beta, \ell) + (t_M, \beta, (k-1)\ell) \\
(p_R, \beta, k\ell) &= (p_M, \beta, (k-1)\ell) + (p', \beta, \ell) \\
(t_R, \beta, k\ell) &= (t_M, \beta, (k-1)\ell) + (t', \beta, \ell)
\end{aligned}$$

sont satisfaites. Alors, d'après le lemme 6.7.6, on a

$$\begin{aligned}
(p_L, \beta, k\ell) &= (p, \beta, \ell) + (p_1, \beta, \ell) + \cdots + (p_{k-1}, \beta, \ell) \\
(p_R, \beta, k\ell) &= (p_1, \beta, \ell) + \cdots + (p_k, \beta, \ell) \\
(p_M, \beta, k\ell) &= (p_1, \beta, \ell) + \cdots + (p_{k-1}, \beta, \ell) \\
(t_L, \beta, k\ell) &= (t, \beta, \ell) + (t_1, \beta, \ell) + \cdots + (t_{k-1}, \beta, \ell) \\
(t_R, \beta, k\ell) &= (t_1, \beta, \ell) + \cdots + (t_k, \beta, \ell) \\
(t_M, \beta, k\ell) &= (t_1, \beta, \ell) + \cdots + (t_{k-1}, \beta, \ell)
\end{aligned}$$

avec $(p_1, t_1), \ldots, (p_k, t_k)$ tels que pour tout $j \in \{1, \ldots, k\}$, (p_j, t_j) code la configuration de M au pas $i + j$. Puisque

$$(p_R, \beta, k\ell) = (p_M, \beta, (k-1)\ell) + (p', \beta, \ell)$$

alors (p', β, ℓ) et (t_k, β, ℓ) codent le même uplet, ce qui signifie que p' code la position de la tête de lecture et l'état de la machine M au pas $i + k$. D'après le lemme 6.7.4, on a $p' = AfterP(p, t, k)$. □

Lemme 6.7.8. *Soit (p, t) la configuration d'une machine de Turing M à un pas i quelconque de son exécution. Alors on a*

$$t' = AfterT(p, t, k)$$

$$\Leftrightarrow$$

$$\exists \ell \exists p' \exists p_L \exists t_L \exists p_M \exists t_M \exists p_R \exists t_R [p' < \beta^{\ell-2} \wedge t' < \beta^{\ell-2}$$

$$\wedge\ p_R = NextP(p_L, t_L) \wedge t_R = NextT(p_L, t_L)$$

$$\wedge\ (p_L, \beta, k\ell) = (p, \beta, \ell) + (p_M, \beta, (k-1)\ell)$$

$$\wedge\ (t_L, \beta, k\ell) = (t, \beta, \ell) + (t_M, \beta, (k-1)\ell)$$

$$\wedge\ (p_R, \beta, k\ell) = (p_M, \beta, (k-1)\ell) + (p', \beta, \ell)$$

$$\wedge\ (t_R, \beta, k\ell) = (t_M, \beta, (k-1)\ell) + (t', \beta, \ell)].$$

Démonstration (*). Soit (p, t) la configuration d'une machine de Turing M à un pas i quelconque de son exécution.

($\Rightarrow$) Par définition, t' code la mémoire de M au pas $i + k$.

Soient $(p_1, t_1), \ldots, (p_k, t_k)$ des codes de configuration de M tels que, pour tout $j \in \{1, \ldots, k\}$, (p_j, t_j) code la configuration de M à l'état $i + j$. On a donc $t' = t_k$. Soit $p' = p_k$. Soit ℓ tel que $p < \beta^{\ell-k-2}$ et $t < \beta^{\ell-k-2}$. Soit p_L tel que

$$(p_L, \beta, k\ell) = (p, \beta, \ell) + (p_1, \beta, \ell) + \cdots + (p_{k-1}, \beta, \ell).$$

Soit t_L tel que

$$(t_L, \beta, k\ell) = (t, \beta, \ell) + (t_1, \beta, \ell) + \cdots + (t_{k-1}, \beta, \ell).$$

Soit p_R tel que

$$(p_R, \beta, k\ell) = (p_1, \beta, \ell) + \cdots + (p_k, \beta, \ell).$$

Soit t_R tel que

$$(t_R, \beta, k\ell) = (t_1, \beta, \ell) + \cdots + (t_k, \beta, \ell).$$

D'après le lemme 6.7.6, les conditions

$$\begin{aligned}
p' &< \beta^{\ell-2} \\
t' &< \beta^{\ell-2} \\
p_R &= NextP(p_L, t_L) \\
t_R &= NextT(p_L, t_L) \\
(p_L, \beta, k\ell) &= (p, \beta, \ell) + (p_M, \beta, (k-1)\ell) \\
(t_L, \beta, k\ell) &= (t, \beta, \ell) + (t_M, \beta, (k-1)\ell) \\
(p_R, \beta, k\ell) &= (p_M, \beta, (k-1)\ell) + (p', \beta, \ell) \\
(t_R, \beta, k\ell) &= (t_M, \beta, (k-1)\ell) + (t', \beta, \ell)
\end{aligned}$$

sont satisfaites.

($\Leftarrow$) Supposons que $p', t', p_L, t_L, p_M, t_M, p_R, t_R$ soient tels que les conditions

$$\begin{aligned}
p' &< \beta^{\ell-2} \\
t' &< \beta^{\ell-2} \\
p_R &= NextP(p_L, t_L) \\
t_R &= NextT(p_L, t_L) \\
(p_L, \beta, k\ell) &= (p, \beta, \ell) + (p_M, \beta, (k-1)\ell) \\
(t_L, \beta, k\ell) &= (t, \beta, \ell) + (t_M, \beta, (k-1)\ell) \\
(p_R, \beta, k\ell) &= (p_M, \beta, (k-1)\ell) + (p', \beta, \ell) \\
(t_R, \beta, k\ell) &= (t_M, \beta, (k-1)\ell) + (t', \beta, \ell)
\end{aligned}$$

sont satisfaites.

Alors, d'après le lemme 6.7.6, on a

$$\begin{aligned}
(p_L, \beta, k\ell) &= (p, \beta, \ell) + (p_1, \beta, \ell) + \cdots + (p_{k-1}, \beta, \ell) \\
(p_R, \beta, k\ell) &= (p_1, \beta, \ell) + \cdots + (p_k, \beta, \ell) \\
(p_M, \beta, k\ell) &= (p_1, \beta, \ell) + \cdots + (p_{k-1}, \beta, \ell) \\
(t_L, \beta, k\ell) &= (t, \beta, \ell) + (t_1, \beta, \ell) + \cdots + (t_{k-1}, \beta, \ell) \\
(t_R, \beta, k\ell) &= (t_1, \beta, \ell) + \cdots + (t_k, \beta, \ell) \\
(t_M, \beta, k\ell) &= (t_1, \beta, \ell) + \cdots + (t_{k-1}, \beta, \ell)
\end{aligned}$$

avec $(p_1, t_1), \ldots, (p_k, t_k)$ tels que, pour tout $j \in \{1, \ldots, k\}$, (p_j, t_j) code la configuration de M au pas $i + j$.

Puisque

$$(t_R, \beta, k\ell) = (t_M, \beta, (k-1)\ell) + (t', \beta, \ell),$$

alors (t', β, ℓ) et (t_k, β, ℓ) codent le même uplet, ce qui signifie que t' code la mémoire M au pas $i + k$. D'après le lemme 6.7.4, on a $t' = AfterT(p, t, k)$. □

Les deux lemmes précédents prouvent que les fonctions $AfterP$ et $AfterT$ sont diophantiennes puisque l'exponentiation et la concaténation sont diophantiennes. Il devient alors facile de proposer un système de contraintes diophantiennes à partir du code de la configuration initiale (p, t) d'une machine M qui pourra être satisfait si et seulement si la mémoire contient la représentation d'un uplet appartenant à l'ensemble reconnu par M. Il suffit de vérifier que la machine s'arrête dans un état final, donc qu'au bout d'un nombre fini d'instructions, elle entre dans un état final.

Lemme 6.7.9. *Soit une machine de Turing M. Soient $f_1, \ldots, f_z$ les états finaux de M. Soit $n > 0$. Soit E l'ensemble des n-uplets d'entiers naturels dont la représentation canonique est reconnue par M. Soit $(e_1, \ldots, e_n)$ un n-uplet quelconque. Soit (p, t) le code en base β d'une configuration initiale de M tel que $(e_1, \ldots, e_n)$ est représenté en mémoire. On a alors*

$$(e_1, \ldots, e_n) \in E$$

$$\Leftrightarrow$$

$$\exists k \exists r [Elem(AfterP(p,t,k),\beta,r) = f_1 \ \vee \ \ldots \ \vee \ Elem(AfterP(p,t,k),\beta,r) = f_z].$$

Démonstration. ($\Rightarrow$) Si $(e_1,\ldots,e_n) \in E$, alors par définition de M, après un nombre fini k d'étapes, M s'arrête dans un état final $f_k \in \{f_1,\ldots,f_z\}$. Soit r la position de la tête de lecture après k étapes. Soit (p_k,t_k) le code de configuration de M après k étapes. D'après le lemme 6.7.4, on a $p_k = AfterP(p,t,k)$. Par définition de p_k, $Elem(p_k,\beta,r) = f_k$. Puisque $f_k \in \{f_1,\ldots,f_z\}$, alors la proposition

$$Elem(AfterP(p,t,k),\beta,r) = f_1 \vee \cdots \vee Elem(AfterP(p,t,k),\beta,r) = f_z$$

est satisfaite.

($\Leftarrow$) Soient k et r tels que

$$Elem(AfterP(p,t,k),\beta,r) = f_1 \vee \cdots \vee Elem(AfterP(p,t,k),\beta,r) = f_z$$

soit satisfaite. Soit $p_k = AfterP(p,t,k)$. D'après le lemme 6.7.4, p_k code la position de la tête de lecture et l'état de M après k étapes. Donc, après k étapes, M atteint un état final. Par définition de E, ceci signifie que $(e_1,\ldots,e_n)$ appartient à E. □

Lemme 6.7.10. *Soit un n-uplet $(a_1,\ldots,a_n)$. Soit t tel que*

$$\begin{aligned}(t,\beta,1+n+a_1+\cdots+a_n) \ &= \ (\kappa,\beta,1) \\ &\quad +(\mu,\beta,1)+(Repeat(\gamma,\beta,a_1),\beta,a_1)+\ldots \\ &\quad +(\mu,\beta,1)+(Repeat(\gamma,\beta,a_n))\end{aligned}$$

avec κ l'indice du symbole $\star$, μ l'indice du symbole 0 et γ l'indice du symbole 1 dans l'alphabet de M. Alors t code en base β la mémoire d'une machine de Turing contenant la représentation canonique de l'uplet $(a_1,\ldots,a_n)$.

Démonstration. D'après le lemme 3.3.3, pour tout $i \in \{1,\ldots,n\}$, $(Repeat(\gamma,\beta,a_i),\beta,a_i)$ est le code positionnel d'un a_i-uplet dont tous les éléments sont égaux à γ c'est-à-dire

à l'indice du symbole 1. Donc, $(\mu, \beta, 1) + (Repeat(\gamma, \beta, a_i))$ est le code positionnel d'un (a_i+1)-uplet dont le premier élément est l'indice du symbole 0 et les a_i éléments suivants sont égaux à l'indice du symbole 1. Donc $(t, \beta, 1+n+a_1+\cdots+a_n)$ est le code positionnel d'un uplet dont le premier élément est l'indice du symbole $\star$ et les éléments suivants sont constitués de la concaténation de codes positionnels $(\mu, \beta, 1) + (Repeat(\gamma, \beta, a_i))$ pour tout i compris entre 1 et n. Par définition, t code la représentation canonique de $(a_1, \ldots, a_n)$ dans la mémoire de M. □

Lemme 6.7.11. *Soit une machine de Turing M dont les états finaux sont $f_1, \ldots, f_z$. Soit E l'ensemble des n- uplets dont la représentation canonique est reconnue par M. Soit $(a_1, \ldots, a_n)$ un n-uplet quelconque. Soit κ l'indice du symbole $\star$, μ l'indice du symbole 0 et γ l'indice du symbole 1 dans l' alphabet de M. Alors, $(a_1, \ldots, a_n) \in E$ si et seulement si*

$$\exists t \exists p \exists k \exists r [$$

$$\begin{aligned}((t, \beta, 1 + n + a_1 + \cdots + a_n) \;=\; & (\kappa, \beta, 1) + (\mu, \beta, 1) \\ & + (Repeat(\gamma, \beta, a_1), \beta, a_1) + \ldots \\ & + (\mu, \beta, 1) + (Repeat(\gamma, \beta, a_n)))\end{aligned}$$

$$\wedge (p = 1)$$

$$\wedge (Elem(AfterP(p, t, k), \beta, r) = f_1 \vee \cdots \vee Elem(AfterP(p, t, k), \beta, r) = f_z)].$$

Démonstration (*). ($\Rightarrow$) Soit $(a_1, \ldots, a_n) \in E$. Soit t tel que

$$\begin{aligned}(t, \beta, 1 + n + a_1 + \cdots + a_n) \;=\; & (\kappa, \beta, 1) + (\mu, \beta, 1) \\ & + (Repeat(\gamma, \beta, a_1), \beta, a_1) + \ldots \\ & + (\mu, \beta, 1) + (Repeat(\gamma, \beta, a_n)).\end{aligned}$$

D'après le lemme 6.7.10, t code la représentation canonique de $(a_1, \ldots, a_n)$ dans la mémoire de M. Soit $p = 1$. Puisque $p = 1$ et puisque q_1 est l'état initial d'une machine de Turing, alors p code la position de la tête de M dans la configuration initiale. Donc (p, t) code la configuration initiale de M pour laquelle la mémoire contient la représentation

canonique de $(a_1, \ldots, a_n)$. Puisque $(a_1, \ldots, a_n) \in E$, alors, d'après le lemme 6.7.9, il existe k et r tels que

$$(Elem(AfterP(p,t,k),\beta,r) = f_1 \vee \cdots \vee Elem(AfterP(p,t,k),\beta,r) = f_z)$$

est vérifiée. Donc, il existe p, t, k et r tels que

$$\begin{aligned}[[((t,\beta,1+n+a_1+\cdots+a_n) &= (\kappa,\beta,1)+(\mu,\beta,1)\\ &\quad +(Repeat(\gamma,\beta,a_1),\beta,a_1)+\ldots\\ &\quad +(\mu,\beta,1)+(Repeat(\gamma,\beta,a_n)))\\ &\wedge(p=1)\\ \wedge(Elem(AfterP(p,t,k),\beta,r) &= f_1 \vee \cdots \vee Elem(AfterP(p,t,k),\beta,r) = f_z)]\end{aligned}$$

est vérifiée.

($\Leftarrow$) Soient p, t, k et r tels que

$$\begin{aligned}[[((t,\beta,1+n+a_1+\cdots+a_n) &= (\kappa,\beta,1)+(\mu,\beta,1)\\ &\quad +(Repeat(\gamma,\beta,a_1),\beta,a_1)+\ldots\\ &\quad +(\mu,\beta,1)+(Repeat(\gamma,\beta,a_n)))\\ &\wedge(p=1)\\ \wedge(Elem(AfterP(p,t,k),\beta,r) &= f_1 \vee \cdots \vee Elem(AfterP(p,t,k),\beta,r) = f_z)]\end{aligned}$$

est vérifiée.

D'après le lemme précédent, t code la représentation canonique de $(a_1, \ldots, a_n)$ dans la mémoire de M. Puisque $p = 1$, alors p code la position de la tête de lecture et l'état de M dans la configuration initiale. Donc, (p, t) code la configuration initiale de M avec la représentation canonique de $(a_1, \ldots, a_n)$ en mémoire. D'après le lemme 6.7.9, puisqu'il existe k et r tels que

$$(Elem(AfterP(p,t,k),\beta,r) = f_1 \vee \cdots \vee Elem(AfterP(p,t,k),\beta,r) = f_z)$$

est satisfaite, alors $(a_1, \ldots, a_n) \in E$. □

Ce lemme implique donc qu'à toute machine de Turing reconnaissant la représentation d'uplets on peut faire correspondre une équation diophantienne paramétrée qui n'admettra une solution que si l'uplet passé en paramètre de l'équation est tel que sa représentation est acceptée par la machine de Turing. Tout ensemble semi-décidable est donc diophantien.

On peut s'interroger sur l'affirmation implicite qui est ici faite que tout ensemble semi-décidable représenté sous forme canonique peut être reconnu par une machine de Turing. Ceci est effectivement vrai, comme le prouve l'argument suivant. Une machine de Turing possède un alphabet fini de n symboles. Cela signifie que le contenu des cases peut être vu comme la représentation en base n d'un nombre. Il existe un algorithme permettant de passer de la représentation d'un nombre x dans une base n quelconque à sa représentation dans une base n' quelconque. Donc, d'après la thèse de Church, il existe une machine de Turing M qui transforme la représentation de x en base 2 en la représentation de x en base b. On peut composer M avec toute autre machine de Turing M' comme suit :

$$N = \textbf{if } M \textbf{ then } M'.$$

Donc, quelle que soit la base de représentation dans laquelle M' accepte x, la composition fournit une machine N qui reconnaît la représentation de x en base 2. Or, la représentation d'un nombre en base 2 peut être interprétée comme la représentation canonique que nous avons définie précédemment. Donc, pour tout ensemble semi-décidable, il existe une machine de Turing reconnaissant cet ensemble dans sa représentation canonique.

Nous avons donc prouvé que tout ensemble diophantien est semi-décidable et que tout ensemble semi-décidable est diophantien. Par conséquent, les notions d'ensemble diophantien et d'ensemble semi-décidable coïncident exactement.

On pourrait dès maintenant conclure cette étude en affirmant que le dixième problème de Hilbert est indécidable. En effet, le problème de l'arrêt, qui concerne les ensembles semi-décidables, est un problème que l'on sait indécidable. Il consiste à déterminer si une machine de Turing quelconque à laquelle on passe une certaine entrée

aboutira à un état final. Or, à partir de n'importe quelle machine de Turing, nous sommes en mesure de construire une équation diophantienne paramétrée qui admet une solution si et seulement si les paramètres passées à l'équation représentent un mot reconnu par la machine. S'il existait une procédure capable de déterminer si une équation diophantienne quelconque possède une solution, alors nous serions également en mesure de savoir si une machine de Turing donnée s'arrête ou non. Puisqu'on sait que le problème de l'arrêt est indécidable (Turing, 1937), il ne peut donc pas exister une procédure telle que celle décrite par Hilbert. Bien que cet argument soit suffisant, on ira un peu plus loin et on prouvera l'indécidabilité du dixième problème de Hilbert en utilisant uniquement les résultats présentés précédemment, sans faire appel à un problème notoirement indécidable.

Finalement, on peut remarquer que l'équivalence entre les ensembles diophantiens et les ensembles semi-décidables implique l'existence de fonctions diophantiennes universelles telles que décrites dans le chapitre 4. En effet, il existe des machines de Turing dites universelles capables de simuler n'importe quelle machine de Turing (Hopcroft, Motwani et Ullman, 2006). Une équation diophantienne construite à partir d'une telle machine serait capable de simuler n'importe quelle équation diophantienne, ce qui est bien la caractéristique d'une équation diophantienne universelle.

6.8 Ensembles décidables

On a établi dans la section précédente que les ensembles diophantiens sont exactement les ensembles semi-décidables.

Définition 6.8.1. *Un ensemble E d'uplets est dit* décidable *s'il existe une machine de Turing qui, démarrant avec la tête de lecture examinant le première cellule d'une mémoire contenant la représentation canonique d'un uplet* $\mathbf{u}$*, s'arrête dans l'état final* q_2 *si* $\mathbf{u} \in E$ *et dans l'état final* q_3 *sinon.*

Si un ensemble est décidable, il existe donc une machine de Turing à laquelle on peut poser la question de l'appartenance d'un uplet à cet ensemble. Il suffit alors

d'interpréter l'état final sur lequel s'arrête la machine. L'avantage de la décidabilité par rapport à la semi-décidabilité est important. La décidabilité garantit une réponse à notre question alors que la semi-décidabilité ne fait que garantir une réponse positive. En effet, on ne peut pas, au cours de leur exécution, distinguer une machine de Turing qui ne s'arrêtera jamais d'une machine de Turing qui produira une réponse. Si on accepte la thèse de Church-Turing, ce que nous ferons ici, le fait qu'un ensemble soit décidable est équivalent au fait qu'il existe une méthode permettant d'identifier les éléments de cet ensemble. Ainsi, prouver que le dixième problème de Hilbert est indécidable est équivalent à prouver qu'il n'existe pas de machine de Turing permettant l'identification des équations diophantiennes possédant une solution.

Le lien entre décidabilité et semi-décidabilité est établi par le lemme suivant.

Lemme 6.8.1. *Un ensemble E est décidable si et seulement si E et son complémentaire sont semi-décidables.*

Il s'agit d'un résultat bien connu dont on peut trouver la preuve, par exemple, dans le chapitre 8 de Hopcroft, Motwani et Ullman (2006). Il est intéressant de noter que Matiiassevitch (1995) propose une preuve de ce lemme qui se base sur l'équivalence entre ensembles semi-décidables et ensembles diophantiens. Il utilise ainsi les équations diophantiennes pour prouver que si un ensemble et son complémentaire sont semi-décidables, c'est-à-dire diophantiens, alors cet ensemble est décidable. Une conséquence de ce lemme est que pour prouver l'indécidabilité d'un ensemble, il suffit de prouver que cet ensemble ou son complémentaire n'est pas semi-décidable.

6.9 Indécidabilité du dixième problème de Hilbert (Matiassevitch, 1995)

La preuve de l'indécidabilité du dixième problème de Hilbert peut maintenant être achevée en considérant les derniers résultats obtenus. En premier lieu, on a établi que les ensembles diophantiens et les ensembles semi-décidables sont équivalents. De plus, on sait grâce au lemme 6.8.1 qu'un ensemble est décidable si et seulement s'il

est semi-décidable et possède un complémentaire semi-décidable. On a constaté au chapitre 4 que l'ensemble des codes d'équations diophantiennes sans paramètre possédant au moins une solution est diophantien. Son complémentaire, c'est-à-dire l'ensemble des codes d'équations diophantiennes sans paramètre ne possédant aucune solution n'est, d'après le lemme 4.5.10, pas diophantien. Par conséquent, l'ensemble des codes d'équations diophantiennes sans paramètre possédant au moins une solution est indécidable. Or, nous pouvons, à partir de toute équation diophantienne sans paramètre construire le code de cette équation. Supposons que le dixième problème de Hilbert soit décidable. On aurait alors une procédure permettant de déterminer si une équation diophantienne quelconque possède ou non une solution. Par conséquent, puisque nous sommes en mesure de déterminer le code de cette équation, nous disposerions d'un moyen de vérifier l'appartenance de ce code à H_0. Ceci signifierait que H_0 est décidable, ce qui est contradictoire. Par conséquent, il n'existe pas de procédure permettant de déterminer si une équation diophantienne possède une solution en nombres entiers.

CONCLUSION

Nous avons voulu, dans ce travail, offrir une vision aussi claire, aussi dénuée d'ambiguïté que possible de la démonstration de l'indécidabilité du dixième probème de Hilbert. En effet, quoique l'auteur (Matiiassevitch, 1995) expose la démonstration de manière convaincante dans son ouvrage, il reste de nombreux détails qui ne sont pas développés et qui peuvent troubler le lecteur tant il est vrai que, parfois, ces détails ne sont pas totalement triviaux. Étudier ce résultat en profondeur est une bonne occasion de se familiariser avec divers sujets, notamment la logique, la théorie des nombres et la théorie de la calculabilité qui constituent des fondements de l'informatique théorique.

L'indécidabilité du dixième problème de Hilbert est un résultat négatif, statuant sur l'inexistence d'une méthode telle que décrite dans l'énoncé du problème. Toutefois, les recherches effectuées pour arriver à cette conclusion ont contribué à améliorer la connaissance de la nature des équations diophantiennes, en plus de produire des outils mathématiques utiles. Un des résultats saillants de cette recherche est le caractère diophantien de l'exponentiation, propriété dans laquelle résidait la plus grande difficulté de la démonstration. De même, que l'ensemble des nombres premiers puisse être encodé dans une équation est surprenant. Pourtant, ces deux faits deviennent triviaux lorsqu'on prend connaissance de l'équivalence entre ensembles diophantiens et ensembles semi-décidables. Ce résultat, en plus de fournir un nouveau modèle de calcul, permet d'affirmer que tout ensemble décidable est diophantien. Ceci signifie concrètement, si on accepte la thèse de Church, qu'il existe une équation diophantienne codant tout ensemble de valeurs calculables. De plus, la méthode décrite pour simuler toute machine de Turing dans un système de contraintes diophantiennes fournit une méthode pour construire l'équation diophantienne correspondant à la machine. Ceci ne garantit toutefois pas de trouver l'équation la plus simple qui soit.

Un résultat d'indécidabilité offre également de nouvelles perspectives pour prouver l'indécidabilité d'autres problèmes. En effet, prouver l'indécidabilité d'un problème se fait d'ordinaire en ramenant un problème indécidableà ce problème. Le problème de Hilbert constitue donc un nouveau problème permettant de prouver l'indécidabilité d'autres problèmes. Notons que le problème de Hilbert est ici résolu sans que soit effectuée une réduction, même si cela est possible (en fait, on a quand même effectué une réduction du problème original au problème à solutions naturelles).

Le problème de Hilbert n'est, malgré la preuve de son indécidabilité pour des solutions relatives, pas un sujet clos. En effet, il est légitime de s'interroger sur l'extension du résultat d'indécidabilité de problème étendu aux solution rationnelles. Il faut, à ce sujet, être prudent : l'indécidabilité du dixième problème de Hilbert n'implique pas l'indécidabilité du problème étendu aux solutions rationnelles. En effet, s'il existait une méthode permettant d'affirmer qu'une équation diophantienne possède des solutions rationnelles ou pas, cela n'impliquerait pas que cette méthode serait à même de détecter des solutions naturelles. La question de l'indécidabilité du problème étendu aux solutions rationnelles est un sujet recoupant des domaines divers. On sait par exemple que ce problème est équivalent au problème de savoir si on peut fournir des coordonnées à un matroïde dans $\mathbb{Q}$ ou encore au problème de savoir si un treillis arbitraire L est isomorphe au treillis des faces d'un certain polytope convexe de sommets à coordonnées dans $\mathbb{Q}$ (voir Bokowski et Sturmfels, 1980, p. 29).

Finalement, la véracité d'une conjecture topologique présentée par Mazur (1995) impliquerait l'impossibilité de réduire le problème original étendu aux solutions rationnelles au probéme original (Cornelissen et Zahidi, 2000), ce qui, sans conclure à la décidabilité ou non du problème étendu aux solutions rationnelles, nous priverait d'un outil usuel de preuve d'indécidabilité.

Le sujet est donc encore d'actualité et a des ramifications importantes dans des branches a priori éloignées des mathématiques.

RÉFÉRENCES

Anton, Howard A. 2005. *Elementary linear algebra (9th Ed.)*. Toronto : Wiley.

Bokowski, Jürgen, et Bernd Sturmfels. 1980. *Computational Synthetic Geometry*. New York : Springer-Verlag.

Cornelissen, Gunther, et Karim Zahidi. 2000. « Topology of Diophantine sets : remarks on Mazur's conjectures ». In *Hilbert's tenth problem : relations with arithmetic and algebraic geometry* (Ghent, 1999), p. 253-260. Coll. « Contemporary Mathematics », no. 270. Providence : Amer. Math. Soc.

Davis, Martin. 1953. « Arithmetical problems and recursively enumerable predicates ». *Journal of Symbolic Logic*, vol. 18, no. 1, p. 34-41.

Davis, Martin. 1973. « Hilbert's Tenth Problem is unsolvable ». *The American Mathematical Monthly*, vol. 80, no. 3, p. 233-269.

Davis, Martin, Hilary Putnam et Julia Robinson. 1961. « The decision problem for exponential Diophantine equations ». *Annals of Mathematics, Second Series*, vol. 74, no. 3, p. 425-436.

De Koninck, Jean-Marie et Armel Mercier. 1994. *Introduction à la théorie des nombres*. Coll. « Collection universitaire de mathématiques », Québec : Modulo.

Goldstein, Larry J., David I. Schneider et Martha J. Siegel, 2007. *Finite mathematics and its applications, 9th edition*. New York : Prentice Hall.

Hilbert, David. 1900. « Mathematische Probleme. Vortrag, gehalten auf dem internationalem Mathematiker-Kongress zu Paris 1900 ». *Nachrichten von des Königliche Gesellschaft des Wissenschaften zu Göttingen*, p. 253-297 (en allemand). Traduction française avec corrections et additions, in *Compte rendu du Deuxième congrès international des mathématiciens tenu à Paris du 6 au 10 août 1900*, p. 58-94. Paris : Gauthier-Villars, 1902.

Hopcroft, John U., Rajeen Motwani et Jeffrey D. Ullman. 2006. *Introduction to automata theory, languages and computation, third edition*. New York : Addison-Wesley.

Jones, James P. et Yuri V. Matiiassevitch. 1991. « Proof of recursive unsolvability of Hibert's Tenth Problem ». *The American Mathematical Monthly*, vol. 98, no. 8, p. 689-709.

Kummer, Ernst Edward. 1852. « Über die Ergänzungssätze zu den allgemeinen Reciprocitätsgesetzen. *Journal für die Reine und Angewandte Mathematik*, 44 :93-146 (en allemand). Réimpression in André Weil, éditeur, *Ernst Edward Kummer. Collected papers,* volume 1, p. 485-538. Springer-Verlag, Berlin, 1975.

Lagrange, Joseph Louis. 1772. « Démonstration d'un théorème d'arithmétique ». *Nouveaux Mémoires de l'Académie Royale des Sciences et Belles Lettres de Berlin, année 1770* pages 123-133.

Matiiassevitch, Youri, 1970, « Diofantovost' perechislimykh mnozhestv ». *Doklady Akademii Nauk SSSR*, vol. 196, p. 770-773 (en russe). (Traduit en anglais par Yu. V. Matiiassevitch. Enumerable sets are Diophantine. *Soviet Mathematics. Doklady*, vol. 11, p. 354-358, 1970.

Matiiassevitch, Youri. 1976. « Novoe dokazatel'stvo teoremy ob èksponetsial'no diofantovom predstavlenii perechislimykh predikatov ». *Zapiski Nauchnykh Seminarov Leningradskogo Otdeleniya Matematicheskogo Instituta im. V. A. Steklova AN SSSR (LOMI)*, vol. 60, p. 75-92 (en russe). (Traduit en anglais par Yu. V. Matiiassevitch. 1980. « A new proof of the theorem on exponential diophantine representation of enumerable sets ». *Journal of Soviet Mathematics*, vol. 14, no. 5, p. 1475-1486.)

Matiiassevitch, Youri. 1995. *Le dixième problème de Hilbert.* Paris : Masson.

Mazur, Barry. 1992. « The topology of rational points ». Experiment. Math. 1, no. 1, p. 35-45.

Robinson, Julia. 1952. « Existential definability in arithmetic ». *Transactions of the American Mathematical Society*, vol. 72, no. 3, p. 437-449. Providence : Amer. Math. Soc.

Sierpinski, Waclaw. 2003. *Leçons sur les nombres transfinis.* Paris : Éditions Jacques Gabay. (Gauthier-Villars, 1928).

Turing, Alan M. 1937. « On computable numbers, with an application to the Entscheidungsproblem ». *Proceedings of the London Mathematical Society*, Série 2, vol. 42, p. 230-265. Correction, vol. 43 p. 544-546. London Mathematical Society.