Complexité

Frank Wagner

Leçon 1

Quelques exemples de calcul

Définition 1.1 Un *mot binaire*, ou encore *mot booléen*, est une suite finie $(x_0, \ldots, x_{n-1})$ dans l'*alphabet* $\{0, 1\}$; l'entier n désigne sa *longueur*, ou sa *taille*. Plus généralement, si $\mathcal{A}$ est un ensemble, on dénote par $\mathcal{A}^*$ les mots dans l'alphabet $\mathcal{A}$.

Dans ces notes, log désignera le logarithme à base deux. Donc, si on interprète un mot binaire $\bar{x}$ comme entier écrit en base deux, sa longueur est inférieur à $\log \bar{x}$. En premier temps, nous n'allons nous permettre que des opérations booléens, ce qui revient au calcul dans $\mathbb{F}_2$, le corps à deux éléments. Le *temps* d'un algorithme sera le nombre de ces opérations; on le mesurera comme fonction de la taille de la donnée.

Exemple 1.2 Trouver le successeur d'un nombre.

Premier algorithme: Collectionner $\bar{x}$ petit cailloux, en rajouter un, et compter. Temps exponentiel — et où est-ce qu'on trouve tous ces cailloux ?!
Deuxième algorithme: On commence par la droite, et soit change 0 en 1 et s'arrète, soit change 1 en 0, se déplace un pas vers la gauche et recommence; à la fin, si on travaille toujours, on rajoute 1. Temps linéaire.

Exemple 1.3 Trouver le prédecesseur d'un nombre.

Premier algorithme: Si $\bar{x}$ n'est pas zéro (et n'a pas de prédecesseur naturel), on commence à compter (c'est-à-dire itérer l'algorithme qui nous donne le successeur); quand on arrive à $\bar{x}$, l'avant-dernier était le prédecesseur. Temps exponentiel !
Deuxième algorithme: L'inverse du successeur: soit on change 1 en 0 et s'arrète, soit on change 0 en 1 et se déplace vers la gauche. Temps linéaire.

Exemple 1.4 Trouver la somme de deux nombres.

Premier algorithme: On utilise la définition récursive: $\bar{x} + 0 = \bar{x}$ et $\bar{x} + (\bar{y} + 1) = (\bar{x} + 1) + \bar{y}$. Donc on remplace $\bar{x}$ par son successeur et $\bar{y}$ par son prédecesseur; quand le dernier est zéro, on s'arrète. Temps exponentiel. C'est d'ailleurs un phénomène général: les algorithmes récursives prennent un temps exponentiel.
Deuxième algorithme: C'est celui qu'on a appris à l'école: on somme les derniers chiffres, porte 1 si nécessaire, etc. Si n borne la taille de $\bar{x}$ et $\bar{y}$, alors le temps sera borné par $2n$.

On retiendra que les algorithmes efficaces ne sont souvent pas ceux qui utilisent la définition de la fonction.

Exemple 1.5 Trouver la différence de deux nombres.

Premier algorithme: On calcule succéssivement $0 + \bar{y}$, $1 + \bar{y}$, $2 + \bar{y}, \ldots$ et compare avec $\bar{x}$; quand on a égalité, c'est qu'on a trouve $\bar{x} - \bar{y}$. Et quand on arrive à $\bar{x} + \bar{y}$, on s'arrète et conclut que $\bar{y} > \bar{x}$ (il ne faut pas oublier cette possibilité). Temps exponentiel. Ce qui est important, c'est qu ça marche en général: l'inverse d'une fonction calculable est calculable — mais pas nécessairement avec la même vitesse !
Deuxième algorithme: Encore celui de l'école, en temps linéaire.

Exemple 1.6 Trouver le produit de deux nombres.

Premier algorithme: Itérer $\bar{y}$ fois la somme $+\bar{x}$, en commençant par 0. Ça utilise la définition récursive, en temps exponentiel.
Deuxième algorithme: Celui de l'école, qui fonctionne en temps quadratique.

Exemple 1.7 Trouver $\bar{x}^{\bar{y}}$ modulo $\bar{z}$.

Si n borne la taille de $\bar{y}$, on calcule $\bar{x}$, $\bar{x}^2$, $\bar{x}^4$, …, $\bar{x}^{2^{n-1}}$ modulo $\bar{z}$ (c'est-à-dire on réduit modulo $\bar{z}$, prend le carré, réduit, etc.); ensuite on multiplie les puissances qui correspondent à un 1 dans l'expression binaire de $\bar{y}$, toujours modulo $\bar{z}$. On obtiendra un algorithme en temps polynomial.

Remarque 1.8 En itérant n fois un algorithme en temps $p(n)$, où p est un polynôme de degré d, on n'obtient pas un algorithme en temps polynomial de degré $d+1$ — sinon l'exponentiation serait polynomiale, et il nous faut déjà un temps exponentiel pour écrire le résultat ! Ce qui se passe, c'est qu'après la première itération les données sont de taille $p(n)$, donc on doit travailler un temps $p(p(n))$, etc.; à la fin on a bien n itérations de p — et si $p(x) \geq ax$ avec $a > 1$, c'est l'exponentiel.

Exemple 1.9 Multiplication rapide DIVIDE AND CONQUER.

Si $\bar{x}$ et $\bar{y}$ sont de taille inférieure à 2^{n+1}, on les écrit $\bar{x} = a_0 2^{2^n} + a_1$ et $\bar{y} = b_0 2^{2^n} + b_1$, avec a_0, a_1, b_0, b_1 de taille inférieure à 2^n. On a

$$\bar{x} \cdot \bar{y} = a_0 b_0 2^{2^{n+1}} + (a_0 b_1 + a_1 b_0) 2^{2^n} + a_1 b_1;$$

comme $a_0 b_1 + a_1 b_0 = (a_0 + a_1)(b_0 + b_1) - a_0 b_0 - a_1 b_1$, il suffit de calculer les trois produits $a_0 b_0$, $a_1 b_1$, et $(a_0 + a_1)(b_0 + b_1)$, de nombres de taille environ 2^n. Si t_n est le temps nécessaire pour un produit de deux nombres de taille au plus 2^n, on obtient donc

$$t_{n+1} \leq 3t_n + c2^{n+1}$$

(le deuxième terme tient compte des frais de gestion); en mettant $t_n = 3^n u_n$, on a $u_{n+1} \leq u_n + c(\frac{2}{3})^{n+1}$, et $u_n \leq 3c$, d'où $t_n \leq c3^{n+1}$. Pour multiplier deux nombres de taille m, on trouve d'abord n minimal tel que $2^n > m$, donc $\log m \leq n < 1 + \log m$, et on y met un temps de $t_n \leq c3^{2+\log m} = 9cm^{\log 3}$; comme $\log 3 < 2$, c'est mieux que quadratique.

Exemple 1.10 Addition rapide AVEC VOS COPAINS.

Evidemment, il nous faut déjà un temps linéaire pour écrire le résultat d'une addition; si on travaille seul, on peut guère améliorer. Mais si on travaille en groupe (avec plusieurs *processeurs*), ça ira plus vite; en parlera du *temps parallèle*. Le nombre des processeurs utilisés s'appelle aussi la *ressource*.

Donc si $\bar{x} = a_0 2^{2^n} + a_1$ et $\bar{y} = b_0 2^{2^n} + b_1$, avec $a_0, a_1, b_0, b_1 < 2^{2^n}$, on additionne simultanément $a_1 + b_1$, $a_0 + b_0$, et $a_0 + b_0 + 1$; à la fin on voit si on jette le deuxième ou le troisième résultat, c'est-à-dire si on porte 1 de la première addition ou pas. On obtient $t_{n+1} \leq t_n + c$, où $c > 0$ est une constante, et donc $t_n \leq t_0 + nc$. On voit qu'on met un temps $c \log m$ pour additionner deux nombres de taille m; le nombre p_n de processeurs satisfait $p_{n+1} = 3p_n$; la ressource est de l'ordre de $c' m^{\log 3}$.

Exemple 1.11 Equations linéaires booléennes LIN.
Donné un système d'équations booléennes

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & = & b_2 \\ \vdots & & \vdots & & & & \vdots & & \vdots \\ a_{n1}x_1 & + & a_{n2}x_2 & + & \cdots & + & a_{nn}x_n & = & b_n \end{array}$$

avec les a_{ij} et b_i dans $\mathbb{F}_2$, on se demande si ce système a une solution dans $\mathbb{F}_2^n$. L'entier n est un bon mésure de la taille du problème, dont la donnée consiste en $n^2 + n$ bits (ou bittes).

Premier algorithme: Il y a 2^n uples dans $\mathbb{F}_2^n$ qu'on essaie l'un après l'autre, en temps exponentiel.
Deuxième algorithme: Méthode de CRAMER, on calcule les déterminants

$$\sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i\sigma(i)},$$

ce qui nécessite $n!$ additions, encore temps exponentiel !
Troisième algorithme: Méthode de GAUSS (du pivot): on triangularise, et ensuite diagonalise le système; la triangularisation nécessite au plus

$$\sum_{i=n-1}^{1} i^2 = \frac{(n-1)n(2n-1)}{6}$$

opérations, la diagonalisation $\sum_{i=n-1}^{1} i = \frac{(n-1)n}{2}$; on voit bien que le temps de l'algorithme est borné par un polynôme de troisième degré.

Exemple 1.12 Equations polynomiales booléennes POL.
Cette fois, on se permet des produits de variables; comme $x^2 = x$ pour les éléments de $\mathbb{F}_2$, c'est inutile de considérer des puissances. Donc on cherche a savoir s'il existe une solution dans $\mathbb{F}_2^n$ d'un système

$$\begin{array}{ccccccccc} a_{11}\bar{x}_{11} & + & a_{12}\bar{x}_{12} & + & \cdots & + & a_{1n}\bar{x}_{1n} & = & b_1 \\ a_{21}\bar{x}_{21} & + & a_{22}\bar{x}_{22} & + & \cdots & + & a_{2n}\bar{x}_{2n} & = & b_2 \\ \vdots & & \vdots & & & & \vdots & & \vdots \\ a_{n1}\bar{x}_{n1} & + & a_{n2}\bar{x}_{n2} & + & \cdots & + & a_{nn}\bar{x}_{nn} & = & b_n, \end{array}$$

où chaque $\bar{x}_{ij}$ est un produit de variables $x_1, x_2, \ldots, x_n$.

Premier algorithme: Comme d'avant on essaye les 2^n uples de $\mathbb{F}_2^n$, en temps exponentiel.
Deuxième algorithme: Cette fois, il n'y a pas d'alternative: on ne connait aucun algorithme qui peut resoudre le problème POL en temps polynomial !

Remarque 1.13 Notons que si on arrive à deviner une solution, c'est facile de vérifier qu'on ne s'est pas trompé en temps polynomial.

Théorème 1.14 *S'il existe un algorithme qui resoud* POL *en temps polynomial de degré d, il existe un algorithme qui donne une solution pour le système en temps polynomial de degré* $\max\{d+1, 4\}$.

DÉMONSTRATION : On commence par déterminer si le système a une solution. Si non, on s'arrète; si oui, on remplace x_1 par 0 et regarde si le système obtenu a une solution. Si non, on met $x_1 = 1$; en tout cas on recommence avec un système en $n-1$ inconnus, qu'on élimine un par un. On a utilisé n fois l'algorithme de solvabilité, et n fois calculé des substitutions pour le système, dont une se fait en temps polynomiale de degré 3; le total est un temps polynomial de degré $\max\{d+1, 4\}$. ■
Soit POL3 le problème POL où en plus chaque équation ne fait intervenir que trois des variables.

Théorème 1.15 *Si* POL3 *est résoluble en temps polynomial, alors* POL *l'est aussi.*

DÉMONSTRATION : Pour chaque monome dans chaque équation on introduit des nouvelles variables et des nouvelles équations, en posant par exemple $x_1x_2 = u_1$, $u_1x_3 = u_2$, ..., $u_{n-1}x_n = v_{ij}$ pour le produit $\bar{x}_{ij} = x_1 \cdots x_n$; ensuite on remplace une équation $\sum_{j=1}^{n} a_{ij}\bar{x}_{ij} = b_i$ par un système $a_{i1}v_{i1} + a_{i2}v_{i2} = w_1$, $a_{i3}v_{i3} + w_1 = w_2$, ..., $a_{in}v_{in} + w_{n-1} = b_i$. Ça nous donne un problème POL3 en $n + n^3 + n(n-1)$ variables, qui se resoud en temps polynomial en n^3, qui est aussi polynomial en n. ■

Remarque 1.16 On a la même réduction de LIN à LIN3, mais ça nous n'aide guère pour la solution du problème.

Proposition 1.17 *Le problème* POL2 *est résoluble en temps polynomial.*

DÉMONSTRATION : On élimine les variables. Chaque équation qui contient x_1 s'écrit dans la forme $A(x_i)x_1 + B(x_i) = 0$, où $A(x_i)$ et $B(x_i)$ sont des termes en x_i (donc de la forme $ax + b$); elle a une solution si $A(x_i) = 1$ (et alors $x_1 = B(x_i)$), ou si $A(x_i) = B(x_i) = 0$, c'est-à-dire si $(1 + A(x_i))B(x_i) = 0$. Si $A'(x_j)x_1 + B'(x_j) = 0$ est une autre équation de cette forme, on doit en outre demander que $B(x_i) = x_1 = B'(x_j)$ dans le cas que $A(x_i) = A'(x_j) = 1$, ce qui donne l'équation additionelle $A(x_i)A'(x_j)(B(x_i) + B'(x_i)) = 0$. On voit qu'on peut éliminer x_1 avec des équations qui ne contiennent que deux variables. Le problème qu'on rencontre en général, à voire que chaque fois

on obtient environ n^2 équations du deuxième type, ce qui s'itère de façon exponentiel, ne se pose pas, comme il n'y a que $8n(n-1)$ équations en deux variables parmi les n; on les supprime dès qu'il y a des repetitions. ■

Exemple 1.18 Equations disjonctives SAT.
On se demande s'il y a une solution dans $\mathbb{F}_2^n$ pour un système d'équations disjonctives

$$\begin{array}{ccccccccc} a_{11}x_1 & \vee & a_{12}x_2 & \vee & \cdots & \vee & a_{1n}x_n & = & 1 \\ a_{21}x_1 & \vee & a_{22}x_2 & \vee & \cdots & \vee & a_{2n}x_n & = & 1 \\ \vdots & & \vdots & & & & \vdots & & \vdots \\ a_{n1}x_1 & \vee & a_{n2}x_2 & \vee & \cdots & \vee & a_{nn}x_n & = & 1, \end{array}$$

où chaque $a_{ij}x_j$ est soit x_j, soit $\neg x_j$, soit absent.

On convient que le système vide, où tous les x_i sont absents, n'a pas de solution. On définit de SAT3 comme SAT, mais en demandant en outre que chaque équation ne contient que trois variables.

Théorème 1.19 POL *est résoluble en temps polynomial si et seulement si* SAT *l'est, ou encore si et seulement si* SAT3 *l'est.*

DÉMONSTRATION : On réduit SAT à SAT3 de même façon que la réduction de POL à POL3. Ensuite on note que les opérations booléennes se transforment en opérations arithmétiques en posant $\neg x = 1 - x$, $x \wedge y = xy$, et $x \vee y = x + y + xy$. Réciproquement, les équation arithmétiques en trois variables se remplacent par de systèmes d'équations disjonctives; par exemple, $xy = z$ donne le système $x \vee \neg z$, $y \vee \neg z$, $\neg x \vee \neg y \vee z = 1$. ■

Exemple 1.20 Equations linéaires à coefficients entiers.
Cette fois, on demande s'il existe une solution rationelle d'un système

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & = & b_2 \\ \vdots & & \vdots & & & & \vdots & & \vdots \\ a_{n1}x_1 & + & a_{n2}x_2 & + & \cdots & + & a_{nn}x_n & = & b_n, \end{array}$$

où les a_{ij} et b_i sont des entiers de taille bornée par n, et la taille d'un $n < 0$ sera la même que la taille de $-n$; la taille d'un rationnel $\frac{p}{q}$ est le plus grand des tailles de p et de q.

Si on applique directement l'algorithme de Gauß, la taille des coefficients va exploser; en itérant, on obtiendra un large quotient. En effet, à la permière étape on remplace la $i$me équation L_i par $a_{11}L_i - a_{i1}L_1$, ce qui double la taille des a_{ij}. Mais à la deuxième étape, on remplace $a_{11}L_i - a_{i1}L_1$ par

$$(a_{11}a_{22} - a_{21}a_{12})(a_{11}L_i - a_{i1}L_1) - (a_{11}a_{i2} - a_{i1}a_{12})(a_{11}L_2 - a_{21}L_1)$$

pour $i \geq 3$, ce qui donne $a_{11}^2(a_{22}L_i - a_{i2}L_2) - a_{11}(a_{22}a_{i1} - a_{i2}a_{21})L_1$. On peut donc diviser par a_{11}, et la taille ne croit que linéairement en n; à la fin de l'itération elle reste quadratique. ■

Leçon 2

Termes et circuits booléens

Définition 2.1 Un *circuit booléen* est un graphe orienté fini non-vide sans cycles orientés étiqueté, dont les sommets s'appelleront *portes* et les arêtes *flêches*, tel que

1. les *entrées*, c'est-à-dire les portes qui ne reçoivent aucun flêche, sont étiquetées soit par une constante 0 ou 1, soit par une variable x_i;
2. une porte qui ne reçoit qu'une seule flêche est étiquetée id ou $\neg$;
3. Une porte qui reçoit deux flêches est étiquetée $\wedge$ ou $\vee$;
4. aucune porte reçoit plus que deux flêches;
5. les portes qui n'emettent pas de flêche (les *sorties*) sont numérotées.

Comme il n 'y a pas de cycles orientés, il y a au moins une entrée et une sortie (le graphe vide n'est pas un circuit).

Définition 2.2 Soit C un circuit. Un *sous-circuit* de C est un sous-graphe qui, lorsqu'il contient une porte, contient toutes les flêches que cette porte reçoit (et donc les portes qui emettent ces flêches, c'est-à-dire les *prédecesseurs* de p). Si p est une porte de C, le *circuit principal* C_p associé à p est le sous-circuit minimal contenant p; les sous-circuits *immédiats* de p sont les sous-circuits principaux des prédecesseurs de p.

Donc le circuit principal de p est le sous-graphe qui contient tous les chemins qui arrivent à p; notons qu'il a une unique sortie p. Si C est un circuit avec une unique sortie, ses sous-circuits immédiats seront ceux de sa sortie.

Définition 2.3 La *taille* $t(C)$ d'un circuit est le nombre de ses portes; sa *profondeur* $p(C)$ est la longueur maximal d'un chemin orienté (on compte les flêches, pas les portes). On notera également $e(C)$ le nombre de ses entrées, et $s(C)$ le nombre de ses sorties.

Un circuit C dont les étiquettes des entrées sont parmi $\{0, 1, x_i : i < n\}$ nous donne un algorithme pour calculer une fonction booléenne f_C de $\{0,1\}^n$ à $\{0,1\}^{s(C)}$: en effet, pour évaluer f_C à un point $(s_i : i < n)$ on remplace l'étiquette x_i par $s_i \in \{0,1\}$ pour $i < n$ et suit les flêches; aux portes on effectue l'opération indiquée par l'étiquette. Comme il n'y a pas de cycles, on finit par obtenir une valeur à chaque sortie, et donc un $s(C)$-uple, dont l'ordre est déterminée par la numérotation. Le temps de cet algorithme sera la taille $t(C)$, et le temps parallèle sa profondeur $p(C)$; quant à la ressource, elle est donne par une autre mésure, la largeur $l(C)$, qui est le nombre maximal des portes de même hauteur.

Remarque 2.4 On pourrait définir la hauteur $h(p)$ d'une porte p comme longueur maximal d'un chemin d'une entrée à la porte. Le problème, c'est qu'avec cette définition on doit faire toute opération aussitôt qu'on a obtenu ses données, même si le résultat n'est pas utilisé pour longtemps (c'est-à-dire si les hauteurs de tous les successeurs de p sont plus grandes que $h(p) + 1$). Mais comme il est possible qu'à ce moment-là on a plein de travail ailleurs, il serait peut-être plus économique d'attendre un peu. Autre problème: une fois que l'opération est faite, on doit préserver son résultat quelque part, ce qui n'est pas pris en compte. Ceci se remedie en demandant qu'aucune flêche saute d'hauteur; si nécessaire on doit ajouter des portes *id* au circuit.

Souvent on ne considère que les circuits à une seule sortie; pour calculer une fonction à valeurs dans $\{0,1\}^n$, on utilise n circuits. Evidemment, quand on a un circuit avec $s(C)$ portes, il suffit de le remplacer par ses sous-circuits principaux associés aux sorties. On dira que deux circuits sont *équivalents* s'ils calculent la même fonction.

On appellera la *valence* d'un circuit (*fan-out* pour les amateurs du franglais) le nombre maximal de successeurs d'une porte. L'*arité* (ou *fan-in*) est le nombre maximal de prédecesseurs, qui pour nos circuits booléens vaut 2 (au plus). Si on permet des grandes conjonctions $\bigwedge$ ou disjonctions $\bigvee$ comme étiquettes, on pourrait obtenir des circuits booléens avec une arité supérieure à deux; un tel circuit C n'a pas plus que $t(C)(t(C) - 1)$ flêches, et chaque flêche sauf les premiers deux qui entrent dans une porte $\bigwedge$ ou $\bigvee$ nécessitent

l'addition d'une porte additionelle $\wedge$ ou $\vee$ pour transformer C dans un circuit C' équivalent d'arité deux, qui sera donc de taille $t(C') \leq t(C)^2$, et de profondeur $p(C') \leq p(C)[1 + \log t(C)]$.

Définition 2.5 Un *terme booléen* est un circuit de valence 1 avec une seule sortie.

Définition 2.6 Un *expression boolénne* est un terme booléen sans entrée 0 ou 1, et où aucune variable n'étiquette deux (ou plus) entrées.

Nous avons permis permis qu'une variable étiquette plusieurs entrées, pour maintenir la correspondance entre nos termes booléens et les termes en logique. Tout circuit est équivalent à un circuit ou une variable n'étiquette qu'une seule entrée — mais en général, si le premier circuit est un terme booléen, le deuxième ne l'est plus !

Exercice 2.7 Soit Le *sélecteur* $S(x, y, z)$ est la fonction ternaire suivante: $S(0, y, z) = y$ et $S(1, y, z) = z$. Montrer que le sélecteur ne se calcule pas par une expression booléenne.

Lemme 2.8 *Soit C un circuit dont les entrées ne sont pas de sorties. Alors $t(C) \leq 3[t(C) - e(C)]$.*

Démonstration : Toute entrée a au moins un successeur (qui n'est pas une entrée), et une telle porte ne peut servir comme successeur que pour deux entrées au maximum. Donc $e(C) \leq 2[t(C) - e(C)]$. ■
Quelquefois on appelle *taille* cette quantité $t(C) - e(C)$. Nous choisirons $t(C)$.

Lemme 2.9 *Si C est un terme, alors $2e(C) - 1 \leq t(C)$; il y a égalité si toutes les portes qui ne sont pas des entrées reçoivent deux flêches.*

Démonstration : Par induction sur la profondeur; le cas de profondeur nulle (taille un) étant trivial. Il y a au plus deux sous-circuit immédiats C_1 et C_2, qui sont disjoints: $t(C) = t(C_1)+t(C_2)+1 \geq 2e(C_1)-1+2e(C_2)-1+1 = 2e(C) - 1$; avec égalité si chaque porte non-entrée a deux prédecesseurs. ■

Lemme 2.10 *Tout terme C est équivalent à un terme C' de taille et profondeur inférieures, qui satisfait $2e(C') - 1 \leq t(C') \leq 3e(C') - 1$.*

DÉMONSTRATION : On supprime toutes les identités, et fait monter les négations grâce aux lois de de Morgan, ce qui n'augmente pas la profondeur. A la fin on supprime les double négations et obtient un terme C' dont la taille serait $2e(C') - 1$ si on ne compte pas les portes-négations, dont il y a au plus $e(C')$. Donc $t(C') \leq 2e(c') - 1 + e(C') = 3e(C') - 1$.

On a peut-être augmenté la taille en montant les négations; dans ce cas on a $t(C) \leq t(C') \leq 3e(C') - 1 = 3e(C) - 1$, et on prend le circuit original. ■

Lemme 2.11 *Le nombre de flêches d'un circuit est compris entre* $t(C) - s(C)$ *et* $2[t(C) - e(C)]$.

DÉMONSTRATION : Toute porte qui n'est pas une sortie émet au moins une flêche; toute flêche arrive à une porte qui n'est pas une entrée, et cette porte reçoit au plus deux flêches. ■

Lemme 2.12 *Pour tout circuit* C *on a* $p(C) \leq t(C) - e(C)$, *et* $t(C) \leq s(C)[2^{p(C)+1} - 1]$. *Si* C *est un terme,* $\log(e(C)) \leq p(C)$.

DÉMONSTRATION : Un chemin maximal de longueur $p(C)$ passe par $p(C)+1$ portes, dont seule la première est une entrée; d'où la permière inégalité. Pour la deuxième on fait une induction sur la profondeur; si elle est zéro, toute entrée est sortie et $t(C) = s(C)$. Sinon, par hypothèse pour tout tout sous-circuit C_i immediat des circuits principaux assiciés aux sorties, on a $t(C_i) \leq 2^{p(C_i)+1} - 1$; avec $p(C_i) \leq p(C) - 1$ ça donne

$$t(C) \leq \sum_i t(C_i) \leq 2s(C)[2^{p(C)} - 1] + s(C) \leq s(C)[2^{p(C)+1} - 1].$$

Si C est un terme, $2e(C) - 1 \leq t(C)$, et $e(C) \leq 2^{p(C)}$. ■

Exercice 2.13 Si C est un circuit d'arité n (≥ 2), alors $t(C) \leq s(C)[n^{p(C)+1} - 1]/(n - 1)$.

Nous allons considérer une transformation de circuit comme pratiquable si elle n'affecte la taille que polynomialement. Elle sera particulièrement innocente si elle n'affecte la profondeur que linéairement: il faut qu'il y ait une constante a telle que si C est le circuit de départ et C' celui d'arrivée, alors $p(C') \leq a[p(C) + \log t(C)]$.

Proposition 2.14 HOOVER, KLAWE, PIPPENGER *Pour tout circuit* C *il y a un circuit équivalent* C' *de valence deux, tel que* $t(C') \leq 3t(C)$ *et* $p(C') \leq 2p(C) + \log s(C)$.

DÉMONSTRATION : Chaque fois qu'une porte p dans C émet plus que deux flêches, en la remplace par un arbre binaire renversé, avec p comme racine et les successeurs de p comme feuilles; cet arbre comportera en tous cas $n-2$ nouvelles portes qu'on va étiqueter par id, si n est le nombre de successeurs de p. Donc $t(C') - t(C)$ est borné par le nombre de flêches de C, donc par $2t(C)$: on a bien $t(C') \leq 3t(C)$.

Pour la profondeur, il nous faut soigneusement choisir ces arbres binaires renversés: si on choisit partout l'arbre le plus symétrique, on risque de multiplier la profondeur par $\log t(C)$. Prenons donc notre circuit C; le *poids* d'une porte p sera la longueur maximale d'un chemin de p à une sortie; quand on va modifier C ce poids va augmenter. Définissons le *niveau* d'une porte son poids calculé dans C; pour faciliter le calcul on va ajouter des portes id dans les flêches qui sautent de niveau (on va les effacer à la fin).

Lemme 2.15 *Si p est une porte de niveau $i+1$ avec n successeurs des poids $a_1, \ldots, a_n$, alors on peut insérer un arbre binaire renversé (étiqueté par id) de racine p et feuilles les successeurs de p, tel que le nouveau poids c de p satisfait $2^c \leq 2(2^{a_1} + \ldots + 2^{a_n})$.*

DÉMONSTRATION : S'il y a deux sucesseurs de même poids a, on les remplace par un prédecesseur commun de poids $a+1$, ce qui ne fait que réduire le nombre de prédecesseurs. On peut donc supposer que $a_1 < a_2 < \cdots < a_n$; on insère un peigne qui donne poids $a_n + 1$ à p:

$$\begin{array}{ccccccccccc} p & \to & id & \to & id & \to & \cdots & \to & id & \to & a_1 \\ \downarrow & & \downarrow & & \downarrow & & & & \downarrow & & \\ a_n & & a_{n-1} & & a_{n-2} & & & & a_2 & & \end{array} \quad \blacksquare$$

On ajoute d'abord ces arbres entre niveaux 0 et 1, ensuite entre 1 et 2, etc., jusqu'au niveau $p(C)$. Si P_i est l'ensemble des portes de niveau i, calculons $s_i := \sum_{p \in P_i} 2^{\text{poids}(p)}$: on a $s_0 = s(C)$; comme une porte n'a qu'au plus deux prédecesseurs, $s_{i+1} \leq 2 \cdot 2s_i$. Le poids maximal d'une entrée, c'est-à-dire la nouvelle profondeur du circuit C' ainsi obtenu, est donc borné par

$$p(C') \leq \log s_{p(C)} \leq \log(4^{p(C)} s(C)) \leq 2p(C) + \log s(C). \quad \blacksquare$$

Par contre, si on veut remplacer un circuit par un terme, a priori la taille va exploser exponentiellement. Ceci est lié au problème du temps parallèle:

> Un calcul en temps (séquentiel) polynomial, est-ce qu'il s'effectue également en temps parallèle logarithmique ?

Donc, est-ce qu'il y a une constante a et une transformation qui à un circuit C associe un circuit C' de profondeur $p(C') \leq a \log t(C)$?

Voyons d'abord que la profondeur reste bornée:

Remarque 2.16 Toute fonction booléenne en n variables s'éxprime par un terme booléen de profondeur au plus $n + \log n + 2$, et de taille exponentielle en n.

DÉMONSTRATION : Trivial si $f \equiv 0$. Sinon on exprime la fonction f sous forme disjonctive normale

$$\bigvee_{f(s_1,\ldots,s_n)=1} \bigwedge_{i=1}^{n} \neg^{s_i} x_i,$$

où $\neg^0 x_i = \neg x_i$ et $\neg^1 x_i = x_i$, et remplace les $\bigvee$ et $\bigwedge$ par des arbres binaires étiquetés par $\vee$ et $\wedge$. ■

On ne peut pas borner la taille polynomialement: il y a 2^{2^n} fonctions de $\{0,1\}^n$ dans $\{0,1\}$; par contre, dans un circuit de taille t, pour toute porte il y a au plus t^2 choix pour les prédecesseurs, et $n+6$ choix pour l'étiquette parmi $\{id, \neg, \wedge, \vee, 0, 1, x_1, \ldots, x_n\}$, ce qui fait au plus $[t^2(n+6)]^t$ circuits possibles; si t est borné par un polynôme en n, ça ne suffira pas.

On dira qu'un circuit avec une seule sortie C *accepte* un argument $\bar{x}$ s'il vaut 1 sur $\bar{x}$; sinon il *refuse* $\bar{x}$.

Lemme 2.17 *Soit $e_i = 0\ldots010\ldots0$ avec le 1 en i^{me} position. Un circuit qui accepte e_i pour tout $i < n$ et refuse $\bar{0}$ est de profondeur au moins $\log n$.*

DÉMONSTRATION : Par induction sur n; c'est évident si $n = 1$. Pour le cas général considérons un circuit de profondeur minimale qui calcule $\max\{x_i : i \leq n\}$, et regardons sa sortie; si elle est étiquetée $\neg$ on fait monter la négation en utilisant de Morgan. Soient C_1 et C_2 les deux sous-circuits immédiats; si l'étiquette est $\wedge$, soit C_1 ou C_2 refuse $\bar{0}$, et accepte tous les e_i, ce qui contredit la minimalité de la profondeur.

Soit A l'ensemble des $i \leq n$ tel que C_1 accepte e_i, et B l'ensemble des $i \leq n$ tels que C_2 accepte e_i. Alors $A \cup B = \{0, \ldots, n\}$. Le circuit C_1' où on a remplacé les étiquettes x_i pour $i \notin A$ par 0 refuse $\bar{0}$ et accepte les $0\ldots010\ldots0$, avec $|A|$ variables libres; par hypothès il est de profondeur au moins $\log(|A|)$. De même $p(C_2') \geq \log(|B|)$, et

$$p(C) \geq \log(\max\{|A|, |B|\}) + 1 \geq \log \frac{n}{2} + \log 2 = \log n. \quad ■$$

Théorème 2.18 SPIRA *Un terme booléen C est équivalent à un terme (et donc un circuit) de profondeur inférieure à $4 \log t(C)$.*

DÉMONSTRATION : Par induction sur $t(C)$, le cas $t(C) = 1$ étant trivial. Supposons donc $t(C) > 1$, et soit C_0 un sous-terme minimal de C avec $t(C_0) > t(C)/2$. Comme $t(C_0) > 1$, il a un ou deux sous-termes immédiats, qui seront de taille au plus $t(C)/2 < t(C)$; par hypothèse de récurrence on peut les remplacer par des termes de profondeur au plus $4 \log t(C)/2 = 4 \log t(C) - 4$; et C_0 est équivalent à un terme C_0' de profondeur au plus $4 \log t(C) - 3$.

Si $C_0 = C$ on a fini, sinon soient C_1 et C_2 les termens obtenus en remplaçant C_0 par une seule entrée, éttiquetée 0 et 1 respectivement. Donc les tailles de C_1 et C_2 sont bornées par $t(C)/2 + 1/2$; par le même raisonnement que pour C_0 on trouve des termes équivalents C_1' et C_2' de profondeur au plus $4 \log t(C) - 3$. Alors C sera équivalent au terme $S(C_0', C_1', C_2')$; comme le sélecteur $S(x, y, z)$ s'exprime par un terme de profondeur 3, par exemple $(\neg x \wedge y) \vee (x \wedge z)$, on obtient un terme C' de profondeur au plus $3 + (4 \log t(C) - 3) = 4 \log t(C)$. ■
Notons que la taille reste raisonnable: $t(C') < 2^{p(C')+1} \leq 2t(C)^4$.

Corollaire 2.19 *Tout circuit est équivalent à un circuit de profondeur logarithmique si et seulement si tout circuit à une sortie est équivalent à un terme de taille polynomiale.*

DÉMONSTRATION : Si un circuit C à une sortie est équivalent à un terme C'' de taille $t(C'') \leq t(C)^a$, alors C'', et donc C, est équivalent à un terme C' de profondeur au plus $p(C') \leq 4 \log t(C'') \leq 4a \log t(C)$; si C a plusieurs sorties, on considera les sous-circuits principaux de ses sorties séparamment.

Réciproquement, supposons qu'un circuit C à une sortie soit équivalent à un circuit C' de profondeur $p(C') \leq a \log t(C)$. Alors C' est équivalent à un terme C'' de même profondeur (on remplace le sous-circuit principal d'une porte qui émet plusieurs flêches par plusieurs copies de ce sous-circuit), et

$$t(C'') \leq 2^{p(C'')+1} - 1 = 2^{p(C')+1} - 1 \leq 2t(C)^a - 1. \quad \blacksquare$$

Leçon 3

Termes et circuits sur une structure quelconque

Définition 3.1 Soit $\mathfrak{M}$ une structure dans un langage $\mathcal{L}$ qui comporte au moins deux constantes 0 et 1, une fonction ternaire $S(x, y, z)$ (le sélecteur), et l'égalité; on supposera que l'égalité soit la vraie égalité, et le sélecteur satisfasse $S(0, x, y) = x$ et $S(1, x, y) = y$. Un $\mathfrak{M}$-*circuit*, ou circuit *relatif à* $\mathfrak{M}$, ou encore circuit *au sens de* $\mathfrak{M}$, est un graphe orienté sans cycles orientés, étiqueté de façon que :

1. chaque entrée est étiquetée soit par une variable, soit par une constante, soit par un élément de M; ces derniers qualifieront comme *paramètre*;
2. les autres portes sont étiquetées soit par une fonction de $\mathcal{L}$, soit par une relation de $\mathcal{L}$; les flêches qu'elles reçoivent — dont le nombre doit correspondre à l'arité — sont numérotées;
3. les sorties sont numérotées.

Un $\mathfrak{M}$-circuit en variables $x_1, \ldots, x_n$ calcule une fonction $f_C : M^n \to M^{s(C)}$ comme suivant : une porte f qui reçoit $\bar{m}$ émet $f(\bar{m})$, et une porte R qui reçoit $\bar{m}$ émet 0 si $\mathfrak{M} \not\models R(\bar{m})$, et 1 si $\mathfrak{M} \models R(\bar{m})$.

Remarque 3.2
1. Comme $\neg x = S(x, 1, 0)$, $x \wedge y = S(x, 0, y)$ et $x \vee y = S(x, y, 1)$, on dispose des connecteurs booléens.
2. Si le language d'une structure $\mathfrak{M}$ ne comporte pas les constantes booléens ou le sélecteur, on peut choisir (arbitrairement) deux constantes et les

nommer 0 et 1; quant au sélecteur, on prend n'importe quelle fonction définissable sans quanteurs (c'est-à-dire dont le graphe est définissable sans quanteurs). Par abus de language, on parlera néanmoins de $\mathfrak{M}$-circuit. Par exemple, la formule

$$(x = 0 \wedge s = y) \vee (x = 1 \wedge s = z) \vee [\neg(x = 0 \vee x = 1) \wedge s = x]$$

définit un graphe $S(x, y, z) = s$. Dans le cas d'un anneau unitaire, il est plus naturel de prendre $S(x, y, z) = (1 - x)y + xz = y + x(z - y)$ (et le 0 et 1 de l'anneau pour les constantes booléens).

3. Une $\mathcal{L}(M)$-formule sans quanteurs se traduit en un $\mathfrak{M}$-circuit de taille bornée par la longueur de la formule (en tant que mot dans l'alphabet $\mathcal{L}(M) \cup \{\neg, \wedge, \vee, (,), x_i : i \in \mathbb{N}\}$). Réciproquement un $\mathfrak{M}$-circuit correspond à une formule sans quanteurs (pourvu que le résultat d'une porte relationnelle n'est pas utilisé par une porte non-booléenne ; de tels circuits se décomposent alors en une partie haute fonctionnelle, une partie basse booléenne, et des portes relationnelles faisant la transition). Mais la longueur de cette formule risque d'exploser, car le remplacement d'une fonction caractéristique par des relations introduit une distinction par cas, ce qui risque de doubler la taille chaque fois. Par exemple, la condition $S(x, y, z) = 1$ se traduit $(x = 0 \wedge y = 1) \vee (x = 1 \wedge z = 1)$; la repetition de x cause l'explosion.

4. Si le langage ne comporte pas de relations (même pas l'égalité !), on n'a pas besoin des combinaisons booléennes, et donc ni de 0 et 1, ni du sélecteur. Dans ce cas on parlera d'un circuit *fonctionnel*.

   Si $\mathfrak{M} = \langle \mathbb{R}, 0, + \rangle$, on parlera des circuits *additifs*; si $\mathfrak{M} = \langle \mathbb{R}, 0, 1, +, -, * \rangle$, des circuits *arithmétiques*.

Exemple 3.3 $\mathfrak{M} = \langle \mathbb{R}, d \rangle$, où $d(x) = 2x$. Un terme de profondeur p est la même chose qu'un circuit de taille $p + 1$ à une seule sortie, si x étiquette son entrée, il calcule $2^p x$. Donc le théorème de Spira est faux dans ce cas. En fait, le théorème de Spira est soit trivialement faux, soit trivialement vrai pour les circuits fonctionnels dont les fonctions sont toutes *unaires*.

Exercice 3.4 Le théorème de Spira est vrai pour $\langle \mathbb{R}, + \rangle$.

Remarque 3.5 Un terme C' sur $\langle \mathbb{R}, + \rangle$ calcule une combinaison linéaire $\sum_i m_i x_i + \sum_j n_j a_j$ avec $\sum_i m_i + \sum_j n_j = e(C') = \frac{t(C')+1}{2}$. Par contre, la fonction $2^p x$ se calcule par un circuit de profondeur p et de taille $p+1$; on ne peut donc pas remplacer un circuit C par un terme de taille polynomial en $t(C)$, et le problème du temps parallèle a une réponse trivialement négative.

Exercice 3.6 Un langage fonctionnel $\mathcal{L}$ est *complet* pour une structure $\mathfrak{M}$ si pour tout $n \in \mathbb{N}$ toute fonction de M^n dans M s'exprime comme $\mathcal{L}$-terme. Montrer que le théorème de Spira est vrai pour une structure finie de language fini complet.

En effet, les problèmes sur une structure finie de langage fini complet sont équivalents aux problèmes booléens. C'est un exemple de *bi-interpretation* :

Définition 3.7 Une structure $\mathfrak{M}$ *s'interprète* (librement) dans une structure $\mathfrak{N}$ s'il y a une injection σ de M dans une puissance cartésienne de N, telle que l'image $\sigma(M)$, ainsi que les images des graphes des fonctions et des relations sont calculables par des circuits sur $\mathfrak{N}$. Deux structures son *bi-interpretables*, si l'une est interpretable dans l'autre.

L'interpretation est *avec paramètres* si les circuits en utilisent. Par exemple, si $M = \{0, 1\}$, on y a la structure booléenne $\mathbb{B}_2$ et la structure corporelle $\mathbb{F}_2$ (en tant que structure fonctionelle : sans constantes); comme $x + y = (x \wedge \neg y) \vee (\neg x \wedge y)$ et $xy = x \wedge y$, le corps $\mathbb{F}_2$ s'interprète sans paramètres dans $\mathbb{B}_2$, mais pour l'interpretation de $\mathbb{B}_2$ dans $\mathbb{F}_2$ il nous faut un paramètre 1.

Si le langage est fini, la traduction d'un circuit sur $\mathfrak{M}$ en un circuit sur $\mathfrak{N}$ ne multiplie la taille et la profondeur que par un facteur constant. Par contre, le remplacement d'un terme sur $\mathfrak{M}$ par un terme sur $\mathfrak{N}$ est plus délicat: comme les termes interpretants peuvent avoir plusieurs entrées étiquetées par la même variable, ça nous mène à un redoublement des sous-cicuits immédiats au-dessus d'une porte interpretée, et une explosion de la taille. Par exemple, $\langle \mathbb{R}, d \rangle$ est interpretable dans $\langle \mathbb{R}, + \rangle$; mais la fonction $x \mapsto 2^n x$, calculable par un terme sur la première de taille $n + 1$, se calcule par un circuit sur la deuxième de même taille, mais par un terme de taille $2^{n+1} - 1$ et de profondeur au mieux n.

Remarque 3.8 Si le théorème de Spira est vrai dans la première structure, on commence par transformer un terme C sur $\mathfrak{M}$ en un terme C' de profondeur $a \log t(C)$, qui se transforme en un terme sur $\mathfrak{N}$ de profondeur $ab \log t(C)$, et donc de taille $t(C)^{ab \log n}$, où n majore l'arité de $\mathfrak{N}$.

Leçon 4

Problèmes booléens

Dans ce chapitre, $\mathfrak{M}$ sera toujours une structure dans un langage fini.

Définition 4.1 Un *problème* sur $\mathfrak{M}$ est un sous-ensemble $\mathfrak{X}$ de M^*.
Un *circuit de décision* est un circuit à une seule sortie, dont les valeurs sont 0 et 1; un tel circuit C calcule alors une fonction caractéristique f_C.

Définition 4.2 Un problème $\mathfrak{X}$ sur $\mathfrak{M}$ est $\mathbb{P}$ au sens de $\mathfrak{M}$ s'il existe un uple $\bar{a}$ de paramètres dans M et une suite de $\mathfrak{M}$-circuits de décision $C_n(\bar{x}, \bar{a})$ à paramètres $\bar{a}$ pour $n \in \mathbb{N}$, dont la taille croît polynomialement en n, tel que $C_n(\bar{x}, \bar{a})$ accepte $\bar{m}$ si et seulement si $\bar{m} \in \mathfrak{X} \cap M^n$.
Une fonction $f : M^* \to M^*$ est $\mathbb{P}$ au sens de $\mathfrak{M}$ s'il existe un uple $\bar{a}$ de paramètres dans M et une suite de $\mathfrak{M}$-circuits $C_n(\bar{x}, \bar{a})$ à paramètres $\bar{a}$ pour $n \in \mathbb{N}$, dont la taille croît polynomialement en n, tel que $C_n(\bar{x}, \bar{a})$ calcule f pour les entrées de longueur n.

Remarque 4.3 La classe $\mathbb{P}$ est la classe des problèmes polynomiaux *non-uniformes*; elle est aussi notée P/poly. Plus généralement, la classe $P/f(n)$ est la classe des problèmes calculables en temps polynomial à l'aide d'un paramètre additionnel dans $\{0,1\}^*$, qui ne dépend que de la longueur n de l'entrée et dont la taille est bornée par $f(n)$ (dans le cas $\mathbb{P}$ c'est le circuit C_n, facilement codable par un mot binaire grâce à la finitude du langage).

Définition 4.4 Un problème $\mathfrak{X}$ sur $\mathfrak{M}$ est P au sens de $\mathfrak{M}$ s'il existe un uple $\bar{a}$ de paramètres dans M et un algorithme polynomial (standard) qui a $n \in \mathbb{N}$ associe un circuit de décision $C_n(\bar{x}, \bar{y})$, tels que $C_n(\bar{x}, \bar{a})$ accepte $\bar{m}$ si et seulement $\bar{m} \in \mathfrak{X} \cap M^n$.

Une fonction $f : M^* \to M^*$ est P au sens de $\mathfrak{M}$ s'il existe un uple $\bar{a}$ de paramètres dans M et un algorithme polynomial (standard) qui à n associe un circuit de décision $C_n(\bar{x}, \bar{y})$, tels que $C_n(\bar{x}, \bar{a})$ calcule f pour les entrées de longueur n.

Remarque 4.5 Pour définir ce que c'est un algorithme standard, on peut utiliser sa définition favorie, par exemple en utilisant les machines de Turing. Comme dans ce cours on va se concentrer sur les classes non-uniformes et les algorithmes non-standards, je m'abstiens d'en donner plus de détails.

D'ailleurs, la classe des problèmes P sur $\mathfrak{M}$ se caractérise aussi comme les problèmes décidables en temps polynomial par une machine de Turing sur $\mathfrak{M}$, c'est-à-dire une machine de Turing qui peut manipuler les éléments de M en utilisant les fonctions et relations de $\mathfrak{M}$.

On dira *standard* à la place de *sur* $\mathbb{F}_2$.

Théorème 4.6 *Si $\mathfrak{M}$ est une expansion de $\langle \mathbb{R}, <, +, x \mapsto -x \rangle$ (dans un langage fini), alors les problèmes P et $\mathbb{P}$ au sens de $\mathfrak{M}$ sont les mêmes.*

Démonstration : Evidemment un problème P est $\mathbb{P}$. Soit donc $\mathfrak{X}$ un problème $\mathbb{P}$, et $C_n(\bar{x}, \bar{a})$ un $\mathfrak{M}$-circuit avec paramètres $\bar{a}$ qui calcule $\mathfrak{X} \cap M^n$, pour $n \in \mathbb{N}$. On code C_n par un mot binaire c_n, de longueur polynomiale en n. Soit b le réel $0, c_0 5 c_1 5 c_2 5 \ldots$, et $\bar{a}' = \bar{a}^\frown b$.

Notons que la première décimale de b vaut i si $i \leq b + \cdots + b < i + 1$ (somme de dix fois b); en considérant ensuite $(b + \cdots + b) + (-i)$, on peut calculer la deuxième décimale, etc. On obtient la $n$me décimale dans un temps polynomial en n, et donc c_n en temps polynomial, ainsi que $C_n(\bar{x}, \bar{a})$, qu'on utilisera pour tester une entrée $\bar{m}$ de longueur n.

Plus précisément, l'algorithme polynomial standard construira, pour $n \in \mathbb{N}$, un circuit $C'_n(\bar{x}, \bar{y}, z)$ tel que $C'_n(\bar{x}, \bar{a}, b)$ mimique $C_n(\bar{x}, \bar{a})$; comme le nombre de décimales à extraire de b, et ainsi la taille de C_n, est bornée polynomialement en n, cela se fait par un circuit qui, pour chaque porte p de C_n calcule d'abord les décimales correspondantes de b, et ensuite les utilise pour appliquer la fonction ou relation qui correspond à p; comme le langage est fini, cela correspond à un branchement borné. ■

Définition 4.7 Un problème $\mathfrak{X}$ est *booléen* si $\mathfrak{X} \subseteq \{0, 1\}^*$.

Les problèmes booléens se posant sur n'importe quelle structure, on peut les utiliser pour comparer le pouvoir algorithmique de deux structures. C'est une question qui dépend principalement du rôle joué par les paramètres.

Théorème 4.8 *Soit $\mathfrak{M} = \langle \mathbb{R}, +, -, =, 0, 1 \rangle$. Alors les problèmes $\mathbb{P}$ standard sont les mêmes que les problèmes booléens $\mathbb{P}$ au sens de $\mathfrak{M}$; les problèmes P standard sont les mêmes que les problèmes booléens P au sens de $\mathfrak{M}$.*

DÉMONSTRATION : Evidemment un problème $\mathbb{P}$ (ou P) au sens de $\mathbb{F}_2$ l'est au sens de $\mathfrak{M}$. Pour le réciproque, soit $\mathfrak{X}$ un problème booléen $\mathbb{P}$ au sens de $\mathfrak{M}$, dont la suite de circuits de décision C_n utilise des réels $a_1, \ldots, a_m$ comme paramètres. On supposera que $a_1, \ldots, a_k$ soient linéairement indépendants sur $\mathbb{Q}$, et que $a_j = \sum_{i=1}^{k} \frac{z_{ij}}{d} a_i$, avec $z_{ij} \in \mathbb{Z}$ et $d \in \mathbb{N}^+$. On peut alors remplacer chaque circuit $C_n(\bar{x}, a_1, \ldots, a_m)$ par un circuit $C'_n(\bar{x}, a_1, \ldots, a_k)$, qui d'abord calcule dx_i pour $x_i \in \bar{x}$ et da_j pour $1 \leq j \leq m$, en ensuite reprend le calcul de C_n en remplaçant les booléens $(0, 1)$ par $(0, d)$; à la fin on ajoute un test $z = d$?. La taille de C'_n étant bornée linéairement en $t(C_n)$, on peut supposer que $n = m$ et que les paramètres $a_1, \ldots, a_m$ soient $\mathbb{Q}$-linéairement indépendants, avec $a_1 = 1$.

D'après la remarque 3.5 le circuit C_n va manipuler des réels de la forme $\sum_i k_i a_i$, où $|k_i|$ est borné par $2^{t(C_n)}$ (on remplace d'abord C_n par un terme de taille exponentielle). On peut coder $\sum_i k_i a_i$ par l'uple $(\pm_1, |k_1|, \ldots, \pm_m, |k_m|)$ de taille $m\,[t(C_n) + 1]$; les inversions se feront en temps m, les additions en temps linéaire en $t(C_n)$, et les test de $\sum_i k_i a_i = 0$ se remplaceront par des tests de $\bigwedge_i k_i = 0$, ce qui se fait par un circuit standard de taille quadratique en $t(C_n)$. Donc $\mathfrak{X}$ est $\mathbb{P}$ standard.

Quant à la classe P booléenne, tout ceci se fait de manière uniforme. ■

Théorème 4.9 KOIRAN *Soit $\mathfrak{M} = \langle \mathbb{R}, +, -, <, =, 0, 1 \rangle$. Les problèmes booléens $\mathbb{P}$ au sens de $\mathfrak{M}$ sont les mêmes que les problèmes $\mathbb{P}$ standard.*

Rappelons que ce sont aussi les problèmes booléens P au sens de $\mathfrak{M}$.

DÉMONSTRATION : Soit $\mathfrak{X}$ un problème $\mathbb{P}$ au sens de $\mathfrak{M}$, et $C_n(\bar{x}, \bar{a})$ une suite de circuits de décision pour $\mathfrak{X}$. Comme avant, on peut supposer que $\bar{a}$ est $\mathbb{Q}$-linéairement indépendant, avec $a_1 = 1$; on veut remplacer C_n par un circuit standard, qui manipule des uples $(\pm_1, |k_1|, \ldots, \pm_m, |k_m|)$ à la place de $\sum_i k_i a_i$, et sait déjà traiter les additions, soustractions, et tests $\sum_i k_i a_i = 0$?.

Lemme 4.10 *1. Soit V un espace véctoriel sur $\mathbb{R}$ et $\bar{v}$ une combinaison linéaire positive de vecteurs $\bar{u}_1, \ldots, \bar{u}_n$. Alors il y a une sous-famille libre des $\bar{u}_i$ telle que $\bar{v}$ est combinaison positive de cette sous-famille.*

2. Dans un système d'équation linéaires à coefficients réels qui a une solution positive, on peut annuler certaines variables afin que le système résultant ait une unique solution, qui est positive.

DÉMONSTRATION : Si $\bar{v} = \sum_i \lambda_i \bar{u}_i$ avec $\lambda_i \geq 0$, et $\sum_i \mu_i \bar{u}_i = 0$, choisissons j tel que $|\lambda_j/\mu_j|$ soit minimal, et posons $\lambda'_i = \lambda_i - \lambda_j(\mu_i/\mu_j)$. Alors $\lambda'_i \geq 0$, $\lambda'_j = 0$, et $\sum_i \lambda'_i \bar{u}_i = \bar{v}$. Le premier énoncé en découle par récurrence.

Pour 2., soit $A\bar{x} = \bar{b}$ notre système; les hypothèses disent que $\bar{b}$ est combinaison positive des colonnes de A. On choisit une sous-famille libre de colonnes tel que $\bar{b}$ est combinaison libre des cette sous-famille, et mets les $x_i = 0$ qui correspondent aux colonnes en dehors de cette sous-famille. ■

Lemme 4.11 *Soit E l'ensemble des vecteurs $(k_1, \ldots, k_m)$ avec $\sum_i k_i a_i > 0$ et les tailles des k_i bornées par t. Alors il y a $n_1, \ldots, n_m \in \mathbb{Z}$ de tailles bornées par $2m(t + \log m + 1) + 1$, tels que $\sum_i k_i n_i > 0$ pour tout $\bar{k} \in E$.*

DÉMONSTRATION : Comme le système $\sum_i k_i x_i > 0$ (pour $\bar{k} \in E$) a une solution, le système $\sum_i k_i(u_i - v_i) = 1 + w_{\bar{k}}$ (pour $\bar{k} \in E$) a une solution positive rationelle. On en cherche une de petite taille.

D'abord, d'après le lemme précédent, on met certains variables égale à zéro, afin que le système résultant ait une unique solution, qui est positive et rationelle. S'il restent des équations qui comportent un $w_{\bar{k}}$, on les supprime; comme une tel équation est la seule qui contient $w_{\bar{k}}$, on retient un système avec une unique solution, qui est positive et rationelle, en au plus $2m$ variables. Chaque terme de la solution s'exprime comme quotient de deux déterminants de au plus $(2m)^2$ entiers de taille bornée par t; un tel déterminant vaut au plus $(2m)!\,(2^t)^{2m}$, et sa taille est bornée par $2m(t + \log 2m)$. Si on multiplie par le déterminant qui est leur dénominateur commun, on obtient une solution n_i comme différence de deux entiers de taille bornée par $2m(t + \log m + 1)$, et la taille de n_i est inférieure à $2m(t + \log m + 1) + 1$. ■

Considérons maintenant un test $\sum_i k_i a_i > 0?$. On teste d'abord si on a égalité; sinon, on teste $\sum_i k_i n_i > 0?$ avec les n_i donnés par le lemme 4.11. Ceci n'augmente que polynomialement la taille, mais on perd l'uniformité, s'il y en avait au départ : il n'y a aucun moyen d'obténir la suite $(n_1, \ldots, n_m)$ par un algorithmé standard. ■

Ajoutons la multiplication. Un circuit de taille t peut calculer un polynôme de degré 2^t avec coefficients de taille 2^t; si en développe en monômes, il peut en avoir 2^{2^t}, ce qui ne se calcule plus en temps polynomiale standard. Donc on doit faire autre chose.

Proposition 4.12 *Le problème si un circuit arithmétique C sans paramètres calcule le polynôme nul est $\mathbb{P}$ standard.*

DÉMONSTRATION : Par induction sur la profondeur, on voit que C calcule un polynôme de degré $< 2^t$; si ce polynôme n'est pas nul, on obtient un polynôme non-nul en substituant $x_2 = x_1^{2^{t+1}}$, qui se calcule par un circuit de taille $2t(C)$; par récursion on se ramène au cas d'une seule variable.

Par induction sur la profondeur, on voit que les coefficients des monômes du polynôme calculé par C sont de taille $< 2^{2t}$. Donc $2^{2^{2t}}$ ne peut annuler ce polynôme que s'il est identiquement zéro. Comme $2^{2^{2t}}$ se calcule par un circuit de taille $2t + 1$, on se ramène au cas d'un polynôme sans variables.

C calcule donc un nombre n_C de taille $< 2^t$, qui a au plus 2^t diviseurs premiers. Il y a au plus $(t^2)^t 5^t$ circuits de taille t (choix des prédecesseurs fois choix des étiquettes); au plus $t^{2t}10^t$ nombres premiers figurent dans les nombres calculés par un circuit de taille t. Comme il y a environ $a/\log a$ nombres premiers inférieur à a; pour t assez large il y a un nombre premier p_t de taille bornée par $2t^2$ qui ne divise aucun n_C non-nul. Donc pour tester si un circuit arithmétique de taille t calcule 0, on peut calculer modulo p_t, ce qui se fait par un circuit standard de taille polynomiale en t. Comme il n'y a pas d'algorithme standard polynomial pour déterminer p_t, le problème est $\mathbb{P}$ standard, et pas P. ■

Théorème 4.13 KOIRAN *Soit $\mathfrak{M} = \langle \mathbb{R}, +, -, \cdot, 0, 1\rangle$ (ou n'importe quel corps de caractéristique nulle). Les problèmes booléens $\mathbb{P}$ au sens de $\mathfrak{M}$ sont $\mathbb{P}$ standard.*

DÉMONSTRATION : Soit $\bar{a}$ l'uple de paramètres pour une suite C_n de circits de décision pour notre problème $\mathfrak{X}$; on peut supposer que $a_1, \ldots, a_m$ forment un base de transcendance pour $\mathbb{Q}(\bar{a})$; par le théorème de l'élément primitif, $\mathbb{Q}(\bar{a})$ est engendre sur $\mathbb{Q}(a_1, \ldots, a_m)$ par un seul élément b, dont le polynôme minimal sur $\mathbb{Q}(a_1, \ldots, a_m)$ est de degré d. Chaque élément de $\mathbb{Q}(\bar{a})$ s'écrit de manière unique sous la forme $\sum_{i<d} b^i\, P_i(a_1, \ldots, a_p)/Q_i(a_1, \ldots, a_p)$, où $P_i(\bar{x}), Q_i(\bar{x}) \in \mathbb{Z}[\bar{x}]$ sont des polynômes en $\bar{x} = (x_1, \ldots, x_m)$. Donc, on remplace les opérations arithmétiques sur $\mathbb{Q}(\bar{a})$ par des opérations arithmétiques sur les $2d$-uples de polynômes dans $\mathbb{Z}[\bar{x}]$; pour les tests $z = 0$? on utilise la proposition 4.12 pour tester si chaque dénominateur est le polynôme nul.

Finalement, on fait tout le calcul modulo un nombre premier p_t judicieusement choisi, pour éviter l'explosion des tailles des coefficients. ■

Exercice 4.14 KOIRAN Montrer que dans $\langle \mathbb{R}, +, -, \cdot, <\rangle$ tout problème booléen est résoluble en temps exponentiel.

Exercice 4.15 KOIRAN Il existe un problème booléen sur $\langle \mathbb{R}, +, -, \cdot, < \rangle$ qui n'est pas $\mathbb{P}$ au sens de $\langle \mathbb{R}, +, -, \cdot, < \rangle$.

Leçon 5

$\mathbb{P}$ et $\mathbb{NP}$, P et NP

Définition 5.1 Soit $\mathfrak{M}$ une structure. Un problème $\mathfrak{X}$ est $\mathbb{NP}$ (resp. NP) au sens de $\mathfrak{M}$ s'il existe un problème $\mathfrak{Y}$ qui est $\mathbb{P}$ (resp. P) au sens de $\mathfrak{M}$, et un polynôme $p(x)$, tels que un $\bar{x} \in M^n$ est dans $\mathfrak{X}$ si et seulement si il y a $\bar{y} \in M^{p(n)}$ tel que $\bar{x}\bar{y} \in \mathfrak{Y}$.
Un problème est $\mathbb{NBP}$ (resp. NBP) au sens de $\mathfrak{M}$ s'il existe un problème $\mathfrak{Y}$ qui est $\mathbb{P}$ (resp. P) au sens de $\mathfrak{M}$, et un polynôme $p(x)$, tels que $\bar{x} \in M^n$ est dans $\mathfrak{X}$ si et seulement si il y a $\bar{y} \in \{0,1\}^{p(n)}$ tel que $\bar{x}\bar{y} \in \mathfrak{Y}$.
Un problème $\mathfrak{X}$ est *co*-$\mathbb{NP}$ au sens de $\mathfrak{M}$ si $M^* - \mathfrak{X}$ est $\mathbb{NP}$ (et de même pour NP, $\mathbb{NBP}$ et NBP).

NBP standard est NP standard, et $\mathbb{NBP}$ standard est $\mathbb{NP}$ standard.

Remarque 5.2

1. On ne confondra pas $\mathbb{P}$ et NBP : dans les deux cas on se permet un conseil booléen de longueur polynomiale, mais pour $\mathbb{P}$ ce conseil ne dépend que de la longueur de l'entrée; par contre, pour NBP il peut dépendre de l'entrée elle-même.

2. $\mathbb{P}$ n'est pas forcément inclus dans NBP : Bien qu'on soit tenté de dire que pour un probleme $\mathfrak{X} \in \mathbb{P}$ on considère le probleme s'il existe un circuit de décision $C_n(\bar{x}, \bar{a})$ qui décide si $\bar{x} \in \mathfrak{X}$, la question si un circuit de décision C soit le bon pour le problèmle $\mathfrak{X}$ n'est pas nécessairement P.

3. Un problème NBP se resoud par un algorithme exponentiel, qui consiste à énumerer toutes les suites booléennes de longueur $p(n)$ et de voir si une marche. Mais sur une structure quelconque il n'y a aucune raison pourquoi un problème NP serait résoluble en temps exponentiel.

4. Il est facile de voir que co-$\mathbb{P} = \mathbb{P}$, et co-$P = P$ au sens de toute structure $\mathfrak{M}$. Par contre, les questions co-$\mathbb{NP} = \mathbb{NP}$ et co-$NP = NP$ sont ouvertes.

5. L'intersection et la réunion de deux problèmes $\mathbb{NP}$ ou NP le sont aussi.

Définition 5.3 Un problème $\mathfrak{X}$ est $\mathbb{NP}$-complet au sens de $\mathfrak{M}$ s'il est $\mathbb{NP}$, et pour tout problème $\mathfrak{Y}$ qui est $\mathbb{NP}$ au sens de $\mathfrak{M}$ il y a une fonction $f_{\mathfrak{Y}} : M^* \to M^*$ qui est $\mathbb{P}$ au sens de $\mathfrak{M}$, telle que $\bar{y} \in \mathfrak{Y}$ si et seulement si $f_{\mathfrak{Y}}(\bar{y}) \in \mathfrak{X}$.
Un problème $\mathfrak{X}$ est NP-complet au sens de $\mathfrak{M}$ s'il est NP, et pour tout problème $\mathfrak{Y}$ qui est NP au sens de $\mathfrak{M}$ il y a une fonction $f_{\mathfrak{Y}} : M^* \to M^*$, qui est P au sens de $\mathfrak{M}$, telle que $\bar{y} \in \mathfrak{Y}$ si et seulement si $f_{\mathfrak{Y}}(\bar{y}) \in \mathfrak{X}$.

On a les mêmes définitions pour les classes $\mathbb{NBP}$ et NBP.

Le problème de la *satisfaisabilité des circuits à paramètres dans* $\mathfrak{M}$ est le suivant : donné un circuit $C(\bar{x}, \bar{y})$ (codé comme mot binaire) et un uple $\bar{a} \in M$, est-ce qu'il y a $\bar{b} \in M$ tel que $C(\bar{a}, \bar{b}) = 1$? (Donc la réponse est 0 pour NON, et 1 pour OUI.)

Théorème 5.4 *Le problème de satisfaisabilité des circuits à paramètres dans* $\mathfrak{M}$ *est* $\mathbb{NP}$*-complet et* NP*-complet au sens de* $\mathfrak{M}$.

Démonstration : Comme la taille de $\bar{b}$, s'il existe, est inférieure à celle de la donnée, le calcul de $C(\bar{a}, \bar{b})$ se fait en temps polynomial, et le problème est bien NP.

Soit maintenant $\mathfrak{X}$ un problème $\mathbb{NP}$ au sens de $\mathfrak{M}$, de la forme $\exists \bar{y}\, \bar{x}\bar{y} \in \mathfrak{Y}$ pour un problème $\mathbb{P}$ résolu par une suite $C_n(\bar{x}, \bar{y}, \bar{a})$ de circuits à paramètres $\bar{a}$ dont la taille croit polynomialement en la longueur n de $\bar{x}$. La fonction f qui à $\bar{x}$ associe le paire $(C_n, \bar{x}\bar{a})$ est $\mathbb{P}$, car C_n ne dépend que de la longueur n de $\bar{x}$, et $\bar{x} \in \mathfrak{X}$ si et seulement si il y a $\bar{b} \in M$ telle que $C_n(\bar{x}, \bar{b}, \bar{a}) = 1$.

Si $\mathfrak{X}$ est un problème NP au sens de $\mathfrak{M}$, alors la suite des circuits C_n est calculable par un algorithme standard, et f est P au sens de $\mathfrak{M}$. ■

Remarque 5.5 On montre de même que le problème de satisfaisabilité d'un circuit par un uple booléen est $\mathbb{NBP}$-complet et NBP-complet.

Le fait que le problème NP-complet est aussi $\mathbb{NP}$-complet implique en particulier que $NP = P$ donne $\mathbb{NP} = \mathbb{P}$, et $NP = NBP$ donne $\mathbb{NP} = \mathbb{NBP}$; la $\mathbb{NBP}$-completude du problème NBP-complet implique que $NBP = P$ donne $\mathbb{NBP} = \mathbb{P}$, pour toute structure $\mathfrak{M}$.

Le problème de la *satisfaction des énoncés existentiels à paramètres dans* $\mathfrak{M}$ est le suivant : donnée un énoncé existentiel $\exists \bar{y} \varphi(\bar{a}, \bar{y})$ avec $\bar{a} \in M$, on se demande s'il est vrai dans $\mathfrak{M}$.

Théorème 5.6 *Dans toute structure* $\mathfrak{M}$ *le probème de satisfaction des énoncés existentiels à paramètres dans* $\mathfrak{M}$ *est* $\mathbb{NP}$*-complet et* NP*-complet.*

Démonstration : Evidemment c'est un problème NP. Soit maintenant $C_n(\bar{x}, \bar{y})$ un circuit de décision sur $\mathfrak{M}$; on va le transformer en une formule existentielle de taille comparable comme suit : Pour toute porte p de C_n on prend une variable y_p qui prendra la valeur émis par cette porte; on considère une formule φ_p qui détermine y_p en fonction des $\{y_q : q \text{ prédecesseur de } p\}$; on prend la conjonction de toutes ces formules, et quantifie existentiellement les y_p.

La formule $\varphi_n(\bar{x}, \bar{y})$ obtenue est de longueur linéaire en la taille de C_n, et $\mathfrak{M} \models \exists \bar{y} \varphi(\bar{a}, \bar{y})$ si et seulement s'il y a $\bar{b} \in M$ telle que $C_n(\bar{a}, \bar{b}) = 1$. On a donc ramené le problème de satisfaisabilité des circuits à paramètres dans $\mathfrak{M}$ au probème de satisfaction des énoncés existentiels, de manière uniforme et linéaire, ce qui signifie qu'il es $\mathbb{NP}$-complet et NP-complet. ■

Remarque 5.7 Il n'y a pas de raison que le problème de satisfaction des énoncés existentiels booléens soit $\mathbb{NBP}$- ou NBP-complet, car les variables ajoutés ne sont pas des booléens.

Définition 5.8 Une formule est *rudimentaire* si elle est une conjonction finie de formules des types suivants:

1. $x = a$ avec $a \in M$;
2. $y = f(\bar{x})$ avec f une fonction du langage ;
3. $R(\bar{x}) \vee y = 0$ ou $\neg R(\bar{x}) \vee y = 1$, où R est une relation du langage ;
4. $x = 0 \vee x = 1$, $x = \epsilon \vee y = \epsilon' \vee z = \epsilon''$, où $\epsilon, \epsilon', \epsilon''$ sont des booléens.

Théorème 5.9 *Le problème de satisfaisabilité des formules existentiels rudimentaires à paramètres dans* $\mathfrak{M}$ *est* $\mathbb{NP}$*-complet et* NP*-complet.*

Démonstration : On transforme (en temps polynomial) une formule sans quanteurs en une formule existentielle rudimentaire de longueur comparable : d'abord on introduit des variables pour les sous-termes de la formule,

grace à 1. et 2.; ensuite des variables pour les valeurs de vérité des formules atomiques, en utilisant 3.; et finalement on doit prendre des combinaisons booléens, s'appuyant sur 4.. Par exemple, $x = \neg y$ s'écrit $(x = 0 \vee y = 0) \wedge (x = 1 \vee y = 1)$. On demande que la valeur finale soit 1, prend la conjonction de tout ceci, et quantifie existentiellement sur toutes les variables introduits. ■

Corollaire 5.10 Cook *Les problèmes* Sat *et* Sat3 *sont* $\mathbb{NP}$*-complets et* NP*-complets standard.*

Démonstration : Immédiat. ■

Corollaire 5.11 Blum, Shub, Smale *Si* $\mathfrak{M} = \langle \mathbb{R}, 0, 1, +, -, \cdot, =, \leq \rangle$*, le problème d'existence d'un zéro pour un polynôme en* n *variables à coefficients réels et de degré total* 4 *est* NP*-complet.*

Démonstration : On montre d'abord qu'une composante d'une formule rudimentaire est équivalente à une formule de la forme $\exists \bar{y} P(\bar{x}, \bar{y}) = 0$, où P est un polynôme réel. C'est clair pour 1. et 2.; pour 3. on note les équivalences suivantes:

$$\begin{array}{lll} x = y \vee z = 0 & \text{ssi} & (x - y)z = 0 \\ x \neq y \vee z = 1 & \text{ssi} & \exists v\, (z - 1)[(x - y)v - 1] = 0 \\ x \leq y \vee z = 0 & \text{ssi} & \exists v\, (x + v^2 - y)z = 0 \\ x \not\leq y \vee z = 1 & \text{ssi} & \exists v \exists w\, (x - y - v^2)^2(z - 1)^2 + (vw - 1)^2 = 0\ ; \end{array}$$

le cas 4. est analogue. On ramène tous ces polynômes à des polynômes de degré au plus 2 en ajoutant des nouvelles variables; à la fin on note que $\bigwedge_i P_i = 0$ si et seulement si $\sum_i P_i^2 = 0$. ■

Définition 5.12 Une formule est *hyperrudimentaire* si elle est une conjonction finie des formules du type $x = 0$, $x = 1$, $x = y$, $x \neq y$, $y = f(\bar{x})$, $R(\bar{x})$, $\neg R(\bar{x})$, où f est une fonction et R une relation du langage.

Théorème 5.13 $\mathfrak{M}$ *satisfait* $\mathbb{NP} = \mathbb{NBP}$ *si et seulement si le problème de satisfaisabilité des formules hyperrudimentaires à paramètres dans* $\mathfrak{M}$ *est* $\mathbb{NBP}$ *dans* $\mathfrak{M}$ *(et de même pour le cas uniforme).*

Démonstration : Comme ce problème est $\mathbb{NP}$, et même NP, l'égalité des deux classes implique qu'il est $\mathbb{NBP}$. Réciproquement, pour voir que le problème de la satisfaisabilité des formules rudimentaires est $\mathbb{NBP}$, pour

chaque disjonction dans la formule on demande à un oracle booléen quelle composante on doit tester; à la fin il nous reste une formule hyperrudimentaire.

Notons que l'oracle booléen nous donne la formule hyperrudimentaire dont on doit vérifier la satisfaisabilité par un algorithme $\mathbb{NBP}$ (ou NBP) ; il ne fait pas lui-même partie de cette formule. Autrement dit : Il ne s'agit pas de construire une formule hyperrudimentaire $\psi(\bar{x}, \bar{y})$ telle que la formule rudimentaire $\varphi(\bar{x})$ est équivalente à $\exists \bar{y}\, \psi(\bar{x}, \bar{y})$, mais de trouver par un algorithme NBP une formule hyperrudimentaire qu'on va tester. ■

Proposition 5.14 $\langle \mathbb{R}, 0, 1, +, -, = \rangle$ *satisfait* $\mathbb{NP} = \mathbb{NBP}$, $\mathbb{P} \neq \mathbb{NP}$*; de même pour le cas uniforme.*

Démonstration : Il est facile de voir que le problème de satisfaisabilité des formules hyperrudimentaires à paramètres dans $\mathbb{R}$ revient à resoudre un système d'équations linéaires à coefficients entiers et second membre réel; grâce à la méthode de Gauß ce problème est même P. Par conséquence $\mathbb{NP} = \mathbb{NBP}$ et $NP = NBP$.

Considérons le problème s'il existe, donnée une suite $x_1, \ldots, x_n$ de réels, des booléens $\epsilon_1, \ldots, \epsilon_n$ tel que $\sum_i \epsilon_i x_i = 0$. Comme $\epsilon x = S(\epsilon, 0, x)$, ceci s'exprime dans le langage de $\mathfrak{M}$. Ce problème est NP; supposons qu'il soit $\mathbb{P}$, et considérons une suite $C_n(\bar{x}, \bar{a})$ de circuits de décision, dont les tailles sont bornées par un polynôme $p(n)$. Alors C_n ne manipule que de combinaisons linéaires de $\bar{x}$ et de $\bar{a}$, et il teste au plus $p(n)$ égalités de la forme $\sum_i k_i x_i = a$, où $a \in \langle \bar{a} \rangle$. Si $\bar{x}$ est linéairement indépendant modulo $\langle \bar{a} \rangle$, les réponses à ces tests sont toutes négatives, sauf pour ceux de la forme $0 = 0$?.

Si $2^n > p(n)$, on trouve des booléens $\epsilon_1, \ldots, \epsilon_n$ différent de chaque uple $|\mathrm{sgn}(\mathrm{k}_1)|, \ldots, |\mathrm{sgn}(\mathrm{k}_\mathrm{n})|$ figurant dans les tests de C_n. Il existent alors des réels $x_1, \ldots, x_n$, tels que $\sum_i \epsilon_i x_i = 0$, et tout test non-trivial $\sum_i k_i x_i = a$? de C_n donne une réponse négative : Choisir $x_1, \ldots, x_n$ de dimension $n - 1$ sur $\mathbb{Q}$ et satisfaisant $\sum_i \epsilon_i x_i = 0$. Donc C_n donne la même réponse pour cet uple dépendant qu'il donne pour un uple indépendant, contradiction. ■

Exercice 5.15 Montrer que $\langle \mathbb{R}, 0, 1, +, -, = \rangle$ n'a pas de problème booléen $\mathbb{NP}$-complet (ou NP-complet).

Exercice 5.16 $\langle \mathbb{R}, 0, 1, +, -, =, < \rangle$ satisfait $\mathbb{NP} = \mathbb{NBP}$.

Finalement, nous allons lier des question algorithmiques à l'élimination des quanteurs.

Théorème 5.17 *Une structure $\mathfrak{M}$ satisfait $\mathbb{P} = \mathbb{NP}$ si et seulement s'il existe un uple $\bar{a} \in M$ et un polynôme $p(n)$ tels que toute formule existentielle $\exists \bar{y} \varphi(\bar{x}, \bar{y})$ de taille n soit équivalente à une formule $C(\bar{x}, \bar{a}) = 1$, où C est un circuit de taille $p(n)$ qui se trouve en temps $\mathbb{P}$.*
La structure satisfait $P = NP$ si et seulement si ce circuit C se trouve de façon uniforme.

Démonstration : Si toute formule existentielle est équivalente à un circuit de taille polynomial, alors le problème de la satisfaction des énoncés existentiels à paramètres dans $\mathfrak{M}$ à une solution polynomiale; comme il est $\mathbb{NP}$- et NP-complet, on a $\mathbb{P} = \mathbb{NP}$ et $P = NP$, réspectivement.

Réciproquement, si $\mathbb{P} = \mathbb{NP}$, le problème de la satisfaction des énoncés existentiels à paramètres dans $\mathfrak{M}$ admet une suite de circuits C'_n de décision, telle que $C'_n(\varphi, \bar{b}, \bar{a}) = 1$ si et seulement si $\exists \bar{y}\, \varphi(\bar{b}, \bar{y})$ est un énoncé de taille n vrai dans $\mathfrak{M}$; le circuit $C(\bar{x}, \bar{a}) = C'(\phi, \bar{x}, \bar{a})$ sera alors équivalent à la formule existentielle $\exists \bar{y}\, \varphi(\bar{x}, \bar{y})$.

Si $P = NP$, tout se fera de manière uniforme. ■

Remarque 5.18 Donc si $\mathbb{P} = \mathbb{NP}$, une stucture élimine les quanteurs. Notons que la *formule* équivalente à une formule existentiele n'est pas nécessairement de taille polynomiale, car la transformation d'un circuit en une formule peut augmenter la taille de façon exponentielle (sauf si $\mathbb{P} = \mathbb{NC}$).

Remarque 5.19 Si $\mathfrak{M}$ satisfait $\mathbb{NP} = \mathbb{NBP}$, alors $\mathfrak{M}$ élimine les quanteurs : Un formule existentielle $\exists \bar{y}\, \varphi(\bar{x}, \bar{y})$ est équivalent à $\exists \bar{\epsilon}\, C(\bar{x}, \bar{\epsilon}, \bar{a})$ pour un certain circuit C, où $\bar{\epsilon}$ est un uple booléen ; celui-ci est équivalent au circuit $\bigvee_{\bar{\epsilon}} C(\bar{x}, \bar{\epsilon}, \bar{a})$, et donc à une formule de taille exponentielle.