Fondements de l'informatique: Examen Durée: 3h

(corrigé)

L'énoncé comporte 4 parties indépendantes, qui pourront être traitées dans un ordre quelconque. En revanche, dans chaque partie, il peut être utile, dans la réponse à une question, d'utiliser les questions précédentes! On pourra librement admettre le résultat d'une question pour passer aux questions suivantes. La difficulté des questions n'est pas une fonction linéaire ni croissante de leur numérotation.

1 Limites de la logique du premier ordre

Rappels:

- Une théorie T est un ensemble de formules.
- $\mathfrak{M}$ est un modèle de T signifie que chacune des formules de la théorie est satisfaite dans la structure $\mathfrak{M}$.
- On dit qu'une structure est *finie* si son ensemble de base est fini.

Les deux sous-sections qui suivent sont indépendantes.

1.1 La propriété d'être un ensemble fini

Soit Σ une signature du premier ordre, et $\mathcal{P}$ une propriété que chaque structure sur Σ est susceptible de vérifier ou non.

On dit que la propriété $\mathcal{P}$ est *axiomatisable* s'il existe une théorie T telle que pour toute structure $\mathfrak{M}$ sur Σ, $\mathfrak{M}$ vérifie la propriété $\mathcal{P}$ si et seulement si $\mathfrak{M}$ est un modèle de T.

L'objectif est de montrer que la propriété "être un ensemble fini" n'est pas axiomatisable.

Question 1.1. *On considère une signature Σ avec le symbole $=$ d'arité 2, que l'on suppose interprété par l'égalité dans chaque structure sur Σ.*

Montrer que pour chaque entier n, on peut écrire une formule F_n dont les modèles sont les structures dont l'ensemble de base possède au moins n éléments.

En déduire que la propriété "être un ensemble à au moins n éléments" est axiomatisable par une théorie T finie.

Solution : Considérer $F_n = \exists v_1 \exists v_2 \dots \exists v_n \bigwedge_{1 \leq i < j \leq n} \neg v_i = v_j$.

Il suffit alors de considérer $T = \{F_n\}$.

Dans cette solution, on suppose implicitement que l'égalité vérifie les axiomes de l'égalité. Il faut ajouter les axiomes de l'égalité si l'on veut le garantir. □

Question 1.2. *Montrer que la propriété "être un ensemble avec exactement n éléments" est axiomatisable par une théorie T finie.*

Solution : Il suffit de prendre $T = \{F_n \wedge \neg F_{n+1}\}$. $\square$

Question 1.3. *Montrer que la propriété "être un ensemble infini" est axiomatisable par une théorie T infinie.*

Solution : Il suffit de prendre $T = \{F_n | n \in \mathbb{N}\}$. $\square$

Question 1.4. *Prouver que la propriété "être un ensemble fini" n'est pas axiomatisable par une théorie T (on pourra, par l'absurde, considérer une certaine théorie contradictoire).*

Solution : Par l'absurde. Soit T une théorie telle que pour toute structure $\mathfrak{M}$ on ait $\mathfrak{M}$ finie si et seulement si $\mathfrak{M} \models T$. $T \cup \{F_n | n \in \mathbb{N}\}$ doit donc être une théorie contradictoire. Par le théorème de compacité, il existe un sous-ensemble fini $T' \subset T \cup \{F_n | n \in \mathbb{N}\}$ qui est lui aussi contradictoire. Puisque T' est fini, il existe $N \in \mathbb{N}$ tel que $T' \subset T \cup \{F_n | n \leq N\} = T_N$. Par conséquent T_N est contradictoire: mais cela est impossible, car T_n admet pour modèle $\{1, 2, \ldots, N\} \subset \mathbb{N}$ muni de l'égalité standard. $\square$

1.2 Validité dans les structures finies

On rappelle qu'une formule close ϕ du premier ordre est valide si elle est vraie dans toute structure.

Soit V l'ensemble des formules closes du premier ordre valides, et soit VF l'ensemble des formules closes du premier ordre valides dans toute structure finie.

Question 1.5. *Considérons une signature Σ contenant un symbole de fonction unaire f et le symbole $=$ d'arité 2. Ecrire une formule du premier ordre ψ sur la signature Σ qui exprime la condition que si f est injective alors f est surjective.*

Solution : On écrit $\forall x \forall y (f(x) = f(y) \Rightarrow x = y) \Rightarrow \forall y \exists x f(x) = y$. $\square$

Question 1.6. *En déduire que V et VF ne sont pas les mêmes ensembles.*

Solution : La formule précédente est vraie sur toutes les structures finies. Par contre, la fonction qui à x associe $2x$ sur $\mathbb{N}$ est injective sans être surjective. Elle n'est donc pas vraie dans toutes les structures. $\square$

Question 1.7. *Expliquez pourquoi V est récursivement énumérable.*

Solution : Une formule est dans V si et seulement si elle est valide. Par le théorème de complétude, il y a équivalence entre être valide et être démontrable. On a vu dans le cours que l'ensemble des formules démontrables était récursivement énumérable. $\square$

Question 1.8. *On considère l'ensemble $SATF$ des formules closes ψ qui sont finiment satisfiables: $\psi \in SATF$ signifie que dans une certaine structure finie U_n, ψ est vraie.*

Soit F une formule close. Relier l'appartenance de F et $\neg F$ à VF et à $SATF$.

Solution : Une formule F est dans VF si et seulement si $\neg F$ n'est pas dans $SATF$. $\square$

On ne considère dans la suite que des signatures finies, c'est-à-dire avec un nombre fini de symboles.

Question 1.9. *Montrer que $SATF$ est récursivement énumérable.*

Solution : Pour une signature donnée finie, et une taille donnée, il n'y a qu'un nombre fini de structures de cette taille sur cette signature (à isomorphisme près).

On considère un algorithme qui teste toutes les structures finies par tailles croissantes jusqu'à trouver un modèle de la formule ψ, et qui s'arrête alors. Si $\psi \in SATF$, l'algorithme terminera. Sinon, l'algorithme ne terminera jamais sur ψ. □

Question 1.10. *Expliquez pourquoi le problème de l'arrêt des machines de Turing se réduit au complémentaire de $SATF$.*

Solution : On a construit en cours une formule γ qui est satisfiable si et seulement si une machine de Turing M n'accepte pas un certain mot w (voir cours sur la complétude). Cela prouve le résultat, en observant que la formule γ est satisfiable sur une structure finie. □

Question 1.11. *En déduire que VF n'est pas récursivement énumérable (ce résultat porte le nom de Théorème de Trakhtenbrot).*

Solution : Si VF était récursivement énumérable, alors $SATF$ aurait son complémentaire récursivement énumérable, puisque F est dans le complémentaire de $SATF$ si et seulement si $\neg F$ est dans VF.

Or $SATF$ étant récursivement énumérable, cela impliquerait que $SATF$ serait récursif. Puisque le problème de l'arrêt des machines de Turing, se réduit au complémentaire de $SATF$, le problème de l'arrêt des machines de Turing serait récursif: absurde. □

2 Propriétés des fonctions calculables

On a vu dans le cours qu'un ensemble récursivement énumérable non-vide correspondait à un ensemble dont on pouvait énumérer les éléments. Cela peut se reformuler en: Soit $E \subset \mathbb{N}$ un ensemble récursivement énumérable non-vide. Il existe une fonction calculable $f : \mathbb{N} \to \mathbb{N}$ qui énumère E: c'est-à-dire telle que $E = \{f(x)|x \in \mathbb{N}\}$.

Question 2.1. *Supposons que $E \subset \mathbb{N}$ soit énuméré par une fonction $f : \mathbb{N} \to \mathbb{N}$ calculable croissante strictement: $E = \{f(x)|x \in \mathbb{N}\}$, avec $f(x) < f(x+1)$ pour tout x. Montrer que E est décidable.*

Solution : Voici un algorithme pour décider si un élément x est dans E: on calcule $f(y)$ pour $y = 0, 1, \ldots$ jusqu'à tomber sur $f(y) \geq x$, ce qui va forcément arriver car f est strictement croissante. Alors, si $f(y) = x$, $x \in E$, et sinon $x \notin E$. □

Question 2.2. *Montrer la réciproque: tout ensemble décidable $E \subset \mathbb{N}$ infini est énuméré par une fonction $f : \mathbb{N} \to \mathbb{N}$ calculable croissante strictement.*

Solution : Il suffit de considérer la fonction f qui correspond à l'algorithme suivant: sur l'entrée n, on met une variable m à -1 et on teste pour $k = 0, 1, \ldots$ si $k \in E$. Si c'est le cas, on incrémente m. Si $m = n$ alors on retourne k.

Parce que E est supposé infini, l'algorithme terminera bien pour tout n et retournera le $n$ième élément de E. □

Question 2.3. *Montrez que tout sous-ensemble récursivement énumérable infini de $\mathbb{N}$ contient un ensemble décidable infini.*

Solution : Soit E un tel sous-ensemble, et f une fonction totale qui l'énumère. Soit g la fonction suivante, définie par récurrence: $g(0) = f(0)$, $g(n+1) = f(\min\{x \mid f(x) > g(n)\})$. Cette fonction est bien définie car $\{x \mid f(x) > g(n)\}$ ne peut jamais être vide, vu que E est infini. g est strictement croissante, et énumère un sous-ensemble de E. On conclut par les questions précédentes. $\square$

3 Equations diophantiennes

Une fonction $f : \mathbb{N}^n \to \mathbb{N}$ est dite *polynomiale exponentielle* si elle peut s'écrire $f(x_1, \dots, x_n) = t$, où t est soit x_i, soit N, soit $t_1 * t_2$, soit $t_1 + t_2$, soit $t_1 - t_2$, soit ${t_1}^{t_2}$, où $1 \leq i \leq n$, $N \in \mathbb{N}$, et où t_1 et t_2 sont elles-même des fonctions polynomiales exponentielles de $x_1, \dots, x_n$. Une fonction polynomiale exponentielle qui peut se construire sans utiliser le cas ${t_1}^{t_2}$ correspond à une *fonction polynomiale* (un polynôme).

Par exemple,

- $f(x, y, z) = 3x+5y-71z^5$ est une fonction polynomiale, où bien entendu z^5 est $z*z*z*z*z$, et $3x$ est $x + x + x$.
- $f(p, q, r, n) = (p + 1)^{n+3} + (q + 1)^{n+3} - (r + 1)^{n+3}$ est polynomiale exponentielle.

Lorsque f est polynomiale exponentielle, une équation du type $f(x_1, x_2, \dots, x_n) = 0$ est dite *exponentielle diophantienne.* Lorsque f est polynomiale, une équation du type $f(x_1, x_2, \dots, x_n) = 0$ est dite *diophantienne.* Dans les deux cas, une *solution* de l'équation est un n-uplet d'entiers $(a_1, \dots, a_n) \in \mathbb{N}^n$ avec $f(a_1, \dots, a_n) = 0$.

Un ensemble $A \subset \mathbb{N}^n$ est dit *exponentiel diophantien* (respectivement: *diophantien*) s'il existe un entier m et une fonction polynomiale exponentielle (respectivement: polynomiale) $f : \mathbb{N}^{n+m} \to \mathbb{N}$ telle que

$$(a_1, \dots, a_n) \in A \text{ si et seulement si } \exists x_1 \in \mathbb{N}, \dots, \exists x_m \in \mathbb{N}\ f(a_1, \dots, a_n, x_1, \dots, x_m) = 0.$$

Par exemple:

- L'ensemble des entiers x tels qu'il existe y, z avec $3x + 5y - 71z^5 = 0$ est diophantien.
- L'ensemble $\{x, y, z | \exists k \in \mathbb{N} \quad x^k + y^k = z^k\}$ est exponentiel diophantien.
- Pour k fixé, l'ensemble $\{x, y, z | x^k + y^k = z^k\}$ est diophantien.

3.1 Equations diophantiennes et calcul propositionnel

Question 3.1. *Quelles sont les solutions de l'équation diophantienne* $(1 - x)^2 + (1 - y)^2 = 0$*?*

Solution : $x = 1$ et $y = 1$ sont les solutions. Rappellons que sur les réels une somme de carrés est nulle si et seulement si chacun des sommants est nul. $\square$

On dit qu'une fonction polynomiale $f(x_1, \dots, x_n)$ *représente une formule du calcul propositionnel* ϕ sur les variables propositionnelles $x_1, \dots, x_n$, si:

- pour tout $x_1, \dots, x_n \in \{0, 1\}^n$, $f(x_1, \dots, x_n)$ vaut 0 ou 1.
- les solutions de l'équation diophantienne $f(x_1, \dots, x_n) = 1$ sont exactement les $x_1, \dots, x_n \in \{0, 1\}^n$ qui rendent ϕ vraie (en interprétant 0 par faux, et 1 par vrai).

Question 3.2. *Quel est la formule propositionnelle représentée par la fonction polynomiale* $x + y - x * y$*?*

Solution : L'équation précédente représente la formule $x \vee y$. $\square$

Question 3.3. *Proposer une fonction polynomiale qui représente la formule propositionnelle* $\neg x_1$.

Solution : $1 - x_1$ par exemple. $\square$

Question 3.4. *Montrer que pour toute formule propositionnelle sur les variables* $\{x_1, \ldots, x_n\}$, *il existe une fonction polynomiale qui la représente.*

Solution : Par induction structurelle, on peut:

- pour une variable propositionnelle x, prendre la fonction x.
- pour une formule de la forme $\neg\phi$, prendre la fonction $1 - f$, où f est la fonction associée à ϕ.
- pour une formule de la forme $\phi \vee \psi$, prendre le codage $f + g - f * g$, où f et g sont les fonctions associées à ϕ et ψ.
- pour une formule de la forme $\phi \wedge \psi$, prendre la fonction $f * g$, où f et g sont les fonctions associées à ϕ et ψ.
- pour une formule de la forme $\phi \Rightarrow \psi$, prendre prendre la fonction $g + (1 - f) - g(1 - f) = 1 - f + gf$, où f et g sont les fonctions associées à ϕ et ψ.
- pour une formule de la forme $\phi \Leftrightarrow \psi$, prendre prendre la fonction $(1 - g) * (1 - f) + gf = 1 - g - f + 2 * g * f$, où f et g sont les fonctions associées à ϕ et ψ.

$\square$

3.2 Systèmes d'équations

Question 3.5. *Supposons que l'on ait un système d'équations (respectivement: exponentielle) diophantiennes* $f_1(x_1, \ldots, x_n) = 0$, $f_2(x_1, \ldots, x_n) = 0$, $\ldots$, $f_k(x_1, \ldots, x_n) = 0$. *Montrer que les solutions du système correspondent aux solutions d'une unique équation (respectivement: exponentielle) diophantienne* $f(x_1, \ldots, x_n) = 0$.

Solution : Prendre $f(x_1, \ldots, x_n) = f_1(x_1, \ldots, x_n)^2 + \cdots + f_k(x_1, \ldots, x_n)^2$. $\square$

3.3 Codage de quelques relations élémentaires

On va chercher à coder des suites de bits dans des entiers. On va utiliser l'astuce suivante: une suite $a_0, a_1, \ldots, a_n \in \{0, 1\}^{n+1}$ peut se coder par l'entier $\sum_{i=0}^{n} a_i 2^i$.

Question 3.6. *On admettra que l'ensemble des* (k, n, m) *tels que* $m = \binom{n}{k} = \frac{n!}{(n-k)!k!}$ *est un sous-ensemble de* $\mathbb{N}^3$ *qui est exponentiel diophantien: très précisément, on admettra que l'on peut construire une fonction polynomiale exponentielle* $BINOM(m, n, k, y_1, y_2, y_3)$ *telle que* $m = \binom{n}{k}$ *si et seulement s'il existe trois entiers* y_1, y_2, y_3 *avec* $BINOM(m, n, k, y_1, y_2, y_3) = 0$.

Soient $a_0, a_1, \ldots, a_n \in \{0, 1\}^{n+1}$ *et* $b_0, b_1, \ldots, b_n \in \{0, 1\}^{n+1}$ *deux suites de* $n + 1$ *bits. On considère les codages* a *et* b *de ces suites.*

On note $a \ll b$ *pour* $\forall 0 \leq i \leq n,\ a_i \leq b_i$.

On admettra que $a \ll b$ *si et seulement si* $\binom{b}{a} = \frac{b!}{(b-a)!a!}$ *est impair.*

Montrer que la relation $a \ll b$ *(vue comme un sous-ensemble de* $\mathbb{N}^2$*) est exponentielle diophantienne.*

Solution : On a m impair si et seulement si $\exists x \quad m = 2x + 1$, et donc $a \ll b$ si et seulement si $\exists x \exists y_1 \exists y_2 \exists y_3 \quad BINOM(2x + 1, b, a, y_1, y_2, y_3)$. □

3.4 Théorème de Davis-Putnam-Robinson

On s'intéresse dans cette section à prouver le théorème de Davis-Putnam-Robinson:

- Tout ensemble récursivement énumérable $A \subset \mathbb{N}^n$ est exponentiel diophantien.

Le principe de la preuve est de montrer que l'on peut exprimer les exécutions d'une machine à deux compteurs avec une équation exponentielle diophantienne.

Rappelons le principe d'une machine à deux compteurs x_1 et x_2. Initialement, $x_2 = 0$ et $x_1 \in \mathbb{N}$ est l'entrée x. Une telle machine comporte un nombre fini n d'instructions. Pour $i \in \{1, \dots, n\}$, l'instruction i est de l'une des formes

1. $\mathsf{Incr}(c, j)$ qui incrémente x_c puis va à l'instruction $j \neq i$;
2. $\mathsf{Decr}(c, j)$ qui décrémente x_c s'il n'est pas nul, puis va à l'instruction $j \neq i$;
3. $\mathsf{IsZero}(c, j, k)$ qui teste si x_c est nul, va à l'instruction $j \neq i$ si c'est le cas, et à l'instruction $k \neq i$ sinon (on supposera que $j \neq k$);
4. Halt qui arrête le calcul.

Question 3.7. *Peut-on décider si une machine à compteurs termine avec tous ses compteurs à 0?*

Solution : On peut remplacer l'instruction Halt dans tout programme de machine M à compteurs par des instructions qui décrémentent chacun des compteurs tant qu'ils ne sont pas nuls, suivi de Halt.

La machine M' obtenue (dont le programme se calcule bien à partir de celui de M) termine avec chacun de ses compteurs à 0 si et seulement si M termine.

L'arrêt des machines à 2 compteurs étant indécidable, le résultat en suit. □

Pour prouver le théorème de Davis-Putnam-Robinson, nous allons commencer par encoder l'exécution d'une machine à deux compteurs comme une matrice d'entiers.

Prenons un exemple: la machine avec le programme suivant:

1. $\mathsf{IsZero}(1, 4, 2)$
2. $\mathsf{Decr}(1, 3)$
3. $\mathsf{IsZero}(2, 1, 4)$
4. Halt

L'exécution complète de la machine sur $x_1 = 2, x_2 = 0$ peut se décrire par la matrice suivante, dont:

- les colonnes correspondent au temps t (t croissant correspond aux colonnes de la droite vers la gauche);
- et les deux premières lignes aux valeurs des compteurs x_1, x_2;
- et puisque le programme plus haut contient 4 instructions (1., 2., 3. et 4.), les 4 lignes suivantes à des 0 et des 1, avec un 1 si et seulement si l'instruction correspondante est effectuée.

7	6	5	4	3	2	1	0 =	t
0	0	0	1	1	1	2	2 =	$x_{1,t}$
0	0	0	0	0	0	0	0 =	$x_{2,t}$
0	1	0	0	1	0	0	1 =	$i_{1,t}$
0	0	0	1	0	0	1	0 =	$i_{2,t}$
0	0	1	0	0	1	0	0 =	$i_{3,t}$
1	0	0	0	0	0	0	0 =	$i_{4,t}$

Encore plus précisément: les deux premières lignes représentent la valeur des compteurs avant l'étape t, en considérant que la première étape est $t = 0$. Par exemple, x_1 possède la valeur 2 avant l'étape 0 et 1 (c'est-à-dire, $x_{1,t} = 2$ pour $t = 0$ et $t = 1$). Il possède ensuite la valeur 1 avant l'étape 2 et 3. Etc. Les lignes i indiquent quelle est l'instruction exécutée à l'étape t. Par exemple, à l'étape 0, l'instruction 1. (c'est-à-dire $\mathsf{IsZero}(1, 4, 2)$) est exécutée, et donc $i_{1,0}$ vaut 1, et dans l'étape 2, l'instruction 3. (c'est-à-dire $\mathsf{IsZero}(2, 1, 4)$) est exécutée, et donc $i_{3,2} = 1$.

Etant donnée une machine à 2 compteurs et n instructions, le but est maintenant de construire des équations exponentielles diophantiennes bien choisies qui vérifient si une matrice à $n+2$ lignes représente une exécution de la machine.

Pour cela, nous allons représenter une telle matrice par $n + 2$ entiers $x_1, x_2, i_1, i_2, \ldots, i_n$. Ces $n + 2$ entiers seront une solution des équations si et seulement si la matrice représente une exécution de la machine.

Chacun de ces $n + 2$ entiers code une ligne de la matrice. Par exemple, la ligne du compteur x_1, c'est-à-dire la ligne $(x_{1,t})_t$, va être représentée par l'entier

$$x_1 = \sum_{t=0}^{y} x_{1,t} b^t,$$

où b est un entier plus grand que tous les nombres dans la matrice, et $y = 7$ est le temps de calcul.

En faisant ainsi pour chaque ligne, la matrice précédente devient

$$\begin{array}{l}
0{*}b^7{+}0{*}b^6{+}0{*}b^5{+}1{*}b^4{+}1{*}b^3{+}1{*}b^2{+}2{*}b{+}2 = x_1 \\
0{*}b^7{+}0{*}b^6{+}0{*}b^5{+}0{*}b^4{+}0{*}b^3{+}0{*}b^2{+}0{*}b{+}0 = x_2 \\
0{*}b^7{+}1{*}b^6{+}0{*}b^5{+}0{*}b^4{+}1{*}b^3{+}0{*}b^2{+}0{*}b{+}1 = i_1 \\
0{*}b^7{+}0{*}b^6{+}0{*}b^5{+}1{*}b^4{+}0{*}b^3{+}0{*}b^2{+}1{*}b{+}0 = i_2 \\
0{*}b^7{+}0{*}b^6{+}1{*}b^5{+}0{*}b^4{+}0{*}b^3{+}1{*}b^2{+}0{*}b{+}0 = i_3 \\
1{*}b^7{+}0{*}b^6{+}0{*}b^5{+}0{*}b^4{+}0{*}b^3{+}0{*}b^2{+}0{*}b{+}0 = i_4
\end{array}$$

En faisant ainsi, toute la matrice se représente donc par les valeurs de 6 entiers, les entiers $x_1, x_2, i_1, i_2, i_3, i_4$. Dans le cas général, si l'on a 2 compteurs et n instructions, on a besoin de x_1, x_2, et i_1, i_2, $\ldots$, i_n, soit $n + 2$ entiers.

Nous allons maintenant, pour une machine donnée, produire les équations exponentielles diophantiennes sur les variables x (la donnée d'entrée), y (le nombre d'étapes), $x_1, x_2, i_1, i_2, \cdots, i_n$ et b, dont les solutions représentent l'exécution de la machine sur l'entrée x.

Tout d'abord, on choisit une base b suffisamment grande: on pose l'équation exponentielle diophantienne $b = 2^{x+y+n}$.

On a besoin d'un entier U dont la représentation en base b est une liste de 1 de longueur y: il suffit d'écrire l'équation $1 + bU = U + b^y$: en effet, le nombre $b^{y-1} + b^{y-2} + \cdots + b + 1$ satisfait cette équation et c'est le seul.

On peut alors exprimer beaucoup de faits à partir de formules construites à partir des entiers x_i et des entiers i_j, x, y, b, U et de $\ll$ (la relation $\ll$ est définie dans la question 3.6).

Question 3.8. *Soit $1 \leq l \leq n$. A l'aide de $\ll$, écrire une relation entre i_l et U qui impose que tous les coefficients de i_l doivent être des 0 ou des 1.*

Solution : il suffit d'écrire $i_l \ll U$. $\square$

On peut exprimer le fait qu'à chaque moment au plus une instruction est effectuée: il suffit d'ajouter l'équation $U = \sum_{i=1}^{n} i_l$: cette équation impose qu'il y a exactement un 1 sur chaque colonne pour les i_l (il ne peut pas y avoir de retenue, car $b > n$). En ajoutant l'équation $1 \ll i_1$, on garantit que la première instruction est celle de numéro 1, et en ajoutant $i_n = b^{y-1}$, que la dernière est celle de numéro n (on peut supposer sans perte de généralité que c'est la seule avec l'instruction Halt).

On peut aussi exprimer le fait qu'une instruction du type $\mathsf{Incr}(c, j)$, va à l'instruction j une fois terminée: il suffit d'écrire $bi_l \ll i_j$ pour chaque instruction numéro l du type $\mathsf{Incr}(c, j)$: remarquez comment on utilise le fait qu'une multiplication par b correspond à une décalage vers la gauche.

Question 3.9. *Exprimer le fait que l'instruction l du type $\mathsf{IsZero}(c, j, k)$ va à l'instruction j ou à l'instruction k une fois terminée*
(on exprimera uniquement cette contrainte sur le séquencement des instructions: c'est-à-dire que si l'on exécute l'instruction l, alors on exécute au temps suivant soit l'instruction j soit l'instruction k, sans s'intéresser à exprimer le test sous-jacent nécessaire sur le compteur x_c).

Solution : On écrit $bi_l \ll i_j + i_k$. $\square$

Question 3.10. *Expliquer pourquoi on peut supposer sans perte de généralité que l'on considère des programmes de machines à compteurs tels qu'à chaque fois que $\mathsf{Decr}(c, j)$ est exécutée, le compteur x_c vaut une valeur entière ≥ 1: l'instruction décrémente alors x_c, puis va à l'instruction $j \neq i$;*

Solution : On remplace chaque instruction $\mathsf{Decr}(c, j)$ par deux instructions:

- une instruction qui teste tout d'abord si le compteur x_c est nul, et qui si tel est le cas envoie sur l'instruction j,
- l'instruction $\mathsf{Decr}(c, j)$.

$\square$

Question 3.11. *Exprimer le fait que les instructions d'incrémentation et de décrémentation modifient les compteurs de la façon appropriée.*

Solution : On rappelle que l'état initial du compteur x_1 est x, et que les autres sont initialement nuls.

En utilisant la remarque précédente sur le fait qu'on peut supposer que les décrémentations sont des vraies décrémentations, on écrit $x_1 = x + b(x_1 + \sum_{l \in A(1)} i_l - \sum_{l \in S(1)} i_l)$ et on écrit $x_2 = b(x_j + \sum_{l \in A(2)} i_l - \sum_{l \in S(2)} i_l)$, où $A(j)$ est la liste des instructions qui incrémentent x_j, et $S(j)$ est la liste des instructions qui décrémentent x_j. $\square$

On admettra que l'on peut aussi écrire que chaque instruction l du type $\mathsf{IsZero}(c, j, k)$ va à l'instruction k si $x_c \neq 0$ par une équation exponentielle diophantienne.

Question 3.12. *En déduire le théorème de Davis-Putnam-Robinson: tout ensemble récursivement énumérable $A \subset \mathbb{N}^n$ est exponentiel diophantien.*

Solution : Soit $A \subset \mathbb{N}^n$ un ensemble récursivement énumérable. Il correspond à l'ensemble des n-uplets sur lesquels une machine à 2 compteurs termine. Par tout ce qui précède, il existe un système d'équations exponentielles diophantiennes tel que les $n$uplets sur lesquels la machine s'arrête sont les solutions du système. Par la question 3.5, cela se ramène à une unique équation exponentielle diophantienne. $\square$

3.5 Equations diophantiennes

On admettra le résultat suivant dû à Matiyasevich:

- Théorème de Matiyasevich: L'ensemble des entiers u, v, w tels que $u = v^w$ est diophantien.

Question 3.13. *En admettant le théorème de Matiyasevich, démontrer que tout ensemble récursivement énumérable est diophantien.*

Solution : Par le théorème de Davis-Putnam-Robinson, tout ensemble récursivement énumérable $A \subset \mathbb{N}^n$ est exponentiel diophantien: on peut construire $f(x, z_1, \ldots, z_n) = 0$ tel que $x \in A$ si et seulement si $\exists z_1, \ldots, \exists z_n\ f(x, z_1, \ldots, z_n) = 0$.

Par le théorème de Matiyasevich, il y a une équation diophantienne $e(u, v, w, y_1, \ldots, y_m) = 0$ telle que $u = v^w$ si et seulement si $\exists y_1, \ldots, \exists y_m\ e(x, y_1, \ldots, y_n) = 0$.

On remplace chaque occurrence dans $f(x, z_1, \ldots, z_n)$ de $t_1^{t_2}$ par une nouvelle variable u. On ajoute alors à l'équation originale $f(x, z_1, \ldots, z_n) = 0$ les équations $v = t_1, w = t_2$ et $e(u, v, w, y_1, \ldots, y_m) = 0$. Cela peut se combiner en une unique équation diophantienne par la question 3.5. □

4 NP-complétude

4.1 Partition d'ensembles

On veut prouver que le problème de décision suivant est NP-complet.
Partition d'ensemble
Données : Un ensemble S de n éléments, un ensemble P de parties de S.
Réponse : Décider si l'on peut partitionner S en deux sous-ensembles S_1 et S_2 tels qu'aucun élément de P ne soit inclus dans un des deux sous-ensembles S_1 et S_2?

Par exemple, considérons

- $S = \{1, 2, 3, 4, 5, 6\}$
- $P = \{\{1, 4\}, \{1, 3, 6\}, \{2, 5\}, \{3, 5, 6\}, \{3, 4\}\}$

Sur cette instance, la réponse doit être positive, car on peut considérer la partition suivante de S: $S_1 = \{1, 2, 3\}$ et $S_2 = \{4, 5, 6\}$. Avec cette partition, aucun élément de P est inclus dans S_1 ou dans S_2: en effet, $\{1, 4\} \not\subseteq S_1$, $\{1, 4\} \not\subseteq S_2$, $\{1, 3, 6\} \not\subseteq S_1$, ..., $\{3, 4\} \not\subseteq S_2$.

Question 4.1. *Montrer que le problème* Partition d'ensemble *est dans NP.*

Solution : La donnée de S_1 est un certificat vérifiable en temps polynomial: le vérificateur vérifie pour chacun des éléments de P que ni S_1 ni son complémentaire n'est inclus dans cet élément. □

Question 4.2. *Montrer que le problème* Partition d'ensemble *est NP-difficile.*
Indications : On peut par exemple encoder une instance du problème 3SAT *comme un problème* Partition d'ensemble*: si U est l'ensemble des variables de l'instance de* 3SAT *et leurs négations, on construira l'instance correspondante du problème* Partition d'ensemble *en prenant $S = U \cup \{FAUX\}$. A vous de trouver P, et de montrer l'équivalence entre les deux problèmes.*

Solution : Le problème 3SAT peut se formuler de la façon suivante: 3-SAT
Données : un ensemble U de variables $\{u_1, u_2, \ldots, u_\ell\}$ et une formule propositionnelle sous forme normale conjonctive ϕ ayant des clauses de 3 littéraux

Réponse : Existe-t-il une fonction $t : U \to \{0,1\}$ telle que t satisfait ϕ ?

Lorsque u est un litéral, nous écrivons $\bar{u}$ pour sa négation: $\overline{\neg x}$ vaut x. $\bar{x}$ vaut $\neg x$.
Nous allons contruire à partir d'une instance 3-SAT une instance de Partition d'ensemble.

- $S = U \cup \overline{U} \cup \{FAUX\}$, où $\overline{U} = \{\overline{u} | u \in U\}$.
- pour chaque variable $u \in U$, P contient l'ensemble $\{u, \bar{u}\}$;
- pour chaque clause $c = (x, y, z)$ de ϕ , P contient l'ensemble $\{x, y, z, FAUX\}$.

Cette construction se fait en temps polynomial, puisque pour chaque variable et chaque clause est associé un élément de P.

Montrons que il existe une fonction $t : U \to \{0,1\}$ telle que t satisfait ϕ si et seulement s'il existe une partition de S composée de deux éléments S_1 et S_2 ayant les bonnes propriétés.

=> Supposons que la fonction $t : U \to \{0,1\}$ soit telle que t satisfait ϕ. Considérons la partition de S_1 et S_2 :

- $u \in S_1$ et $\overline{u} \in S_2$ si $t(u) = 1$.
- $u \in S_2$ et $\overline{u} \in S_1$ si $t(u) = 0$.
- $FAUX \in S_2$.

Il est clair que l'ensemble $\{u, \bar{u}\}$ n'est ni contenu dans S_1 ni dans S_2. Pour chaque clause $c = (x, y, z)$ de ϕ , l'ensemble $\{x, y, z, FAUX\}$ a un élément dans S_2, puisque $FAUX \in S_2$ et un autre dans S_1 puisque $t(x) = 1$, ou $t(y) = 1$, ou $t(z) = 1$.

<= Supposons que S_1 et S_2 partitionnent S tels qu'aucun élément de P soit contenu dans S_1 ou dans S_2. Sans perte de généralité, nous supposons que $FAUX \in S_2$. Nous allons construire une fonction $t : U \to \{0,1\}$ telle que t satisfait ϕ.

- si $u \in S_1$ alors $t(u) = 1$.
- si $u \in S_2$ alors $t(u) = 0$.

Pour chaque clause $c = (x, y, z)$ de ϕ, l'ensemble $\{x, y, z, FAUX\}$ a un élément dans S_2 (puisque $FAUX \in S_2$) et un autre $x \in S_1$. Par construction, on a $t(x) = 1$. Donc t satisfait bien la clause c.

□

4.2 Equations diophantiennes

Question 4.3. *Formuler un ou des problèmes NP-complets à propos des équations diophantiennes, en utilisant les résultats de la section 3.1.*

Solution : On a vu que toute formule propositionnelle se traduit en une équation diophantienne. La satisfiabilité d'une formule propositionnelle étant un problème NP-difficile (= NP-dur), la satisfiabilité d'une équation diophantienne par des valeurs dans $\{0,1\}$ est donc NP-difficile. En imposant les solutions doivent rester bornées (par exemple dans $[-1,1]$), le problème est bien dans NP. □