

Fondements de l’informatique. Examen

Durée: 3h

Sujet proposé par Olivier Bournez

Version 7

(corrigé)

Les 5 parties sont indépendantes, et peuvent être traitées dans un ordre quelconque. On pourra admettre le résultat d’une question pour passer aux questions suivantes. On pourra utiliser tous les résultats et les théorèmes démontrés ou énoncés en cours ou en petite classe ou dans le polycopié sans chercher à les redémontrer, sauf si cela est explicitement demandé.

Il est possible d’avoir la note maximale sans répondre à toutes les questions. La difficulté des questions n’est pas une fonction linéaire ni croissante de leur numérotation.

*La **qualité et clarté de votre argumentation et de votre rédaction** sera une partie importante de votre évaluation.*

*Il peut être clair pour vous que certaines notions impliquent d’autres notions, mais à une question du type “le problème est-il A ? est-il B ? ” **il est demandé de répondre aux deux questions**, soit en répondant à chacune, soit en rendant explicite que vous savez que A implique B ou le contraire pour avoir la note maximale, possiblement une fois pour toute dans votre copie.*

On note $|S|$ pour le nombre d’éléments de l’ensemble (fini) S .

1 Décidabilité

On considère des machines de Turing qui travaillent sur un alphabet suffisant pour couvrir tous les symboles utilisés dans un traitement de texte.

Question 1. *Parmi les problèmes de décision suivants, lesquelles sont décidables ? Récursivement énumérable ? Dans NP , NP -complet, dans P ? Justifier votre réponse.*

- 1. Déterminer si un sujet d’examen contient le mot “machine de Turing”.*
- 2. Déterminer si une machine de Turing produit sur l’entrée correspondant au mot vide un sujet d’examen qui contient le mot “machine de Turing”.*

Solution : Etre polynomial implique être NP , qui implique être décidable, qui implique être récursivement énumérable.

Le problème 1. est dans P , car il suffit de parcourir le sujet.

Le problème 2. Est récursivement énumérable, car il suffit de simuler la machine de Turing, et de s’arrêter dès que l’événement se produit pour semi-décider le problème. Il n’est pas décidable car le langage universel L_u s’y réduit. En effet, soit M^+ la machine de Turing qui produit ce sujet d’examen. Cette machine existe car un sujet d’examen est un texte fini. Etant donnés une machine de Turing M et un mot w , on peut construire la machine M_w qui simule M sur w et si M accepte w simule M^+ . M accepte w si et seulement si M_w produit un sujet qui contient le mot “machine de Turing”, et comme on peut produire effectivement M_w à partir du code de M et de w , cela constitue une réduction du problème universel vers ce problème.

Par conséquent, le problème 2 n’est ni dans P , ni dans NP car indécidable. \square

2 Logique et décidabilité

Question 2. On considère le problème $SAT_{\text{propositionnel}}$ de décision suivant :

- **Donnée:** Une formule F du calcul propositionnel.
- **Réponse:** F est-elle satisfiable.

Ce problème est-il décidable ? Récursivement énumérable ? Dans NP , NP -complet, dans P ? Justifier votre réponse.

Solution : Le problème est NP -complet. Voir le cours. (SAT ou $3 - SAT$ en sont des cas particuliers, et le problème est clairement dans NP , car la donnée d'une valeur des variables de F constitue un certificat vérifiable en temps polynomial). La question de savoir s'il est dans P est par conséquent la question ouverte $P \stackrel{?}{=} NP$. Un problème de NP est décidable, et donc aussi récursivement énumérable. \square

Le but est d'étudier la difficulté du problème similaire $SAT_{\text{prédicat}}$ pour la logique du premier ordre¹.

On appelle *machine de Turing à demi-ruban infini*, une machine qui n'écrit jamais aucun symbole à gauche de son entrée : la partie à gauche de l'entrée reste donc constituée à jamais uniquement de symboles de blanc.

Question 3. Montrer que le problème de décision suivant est indécidable : Etant donnée M une machine de Turing à demi-ruban infini, décider si M accepte le mot vide.

Solution : On peut réduire le problème L_{univ} à cette question : étant donnée une machine de Turing M et un mot w , on considère la machine de Turing à demi-ruban infini M_w qui commence par écrire $\#w$, où $\#$ est un nouveau symbole qui n'est pas dans l'alphabet de M , et qui simule ensuite M : à chaque fois que M tente d'aller/écrire à gauche (ce qu'on peut détecter car elle lit le symbole $\#$), M_w remplace le contenu de son ruban de la forme $\# < \text{contenu} >$ par $\#B < \text{contenu} >$, puis reprend la simulation de M . En d'autres termes M_w simule M , mais décale le ruban vers la droite si besoin, de telle sorte que son propre ruban reste toujours à droite du marqueur $\#$.

On aura M accepte w si et seulement si M_w accepte le mot vide. Par ailleurs, le programme de M_w s'obtient facilement (de façon calculable) à partir du programme de M et donc c'est bien une réduction. \square

Etant donné un mot $w = a_1a_2 \dots a_n$ sur l'alphabet Σ , on notera \bar{w} pour le mot w renversé : $\bar{w} = a_na_{n-1} \dots a_2a_1$. Pour a un symbole fixé quelconque de Σ , on note a^T pour le mot de longueur T dont toutes les lettres sont le symbole a . En particulier, $a^0 = \epsilon$.

Etant donnée une machine de Turing M , on considère une signature Σ_M constituée du symbole de constante ϵ , d'un symbole de fonction a d'arité 1 pour chaque lettre a de Σ , et d'un symbole f_q d'arité 2 pour chaque état interne $q \in Q$ de M .

Une telle signature Σ_M possède un modèle \mathfrak{M}_M particulier : son ensemble de base est constitué des mots sur l'alphabet Σ (et donc les variables x s'interprètent comme des mots sur l'alphabet Σ), ϵ s'interprète comme le mot vide, $a(w)$ est défini comme le mot aw pour tout mot w , et $f_q(x, y)$ est défini comme vrai si et seulement si M sur l'entrée vide atteint une configuration où elle est dans l'état q , et où le contenu du ruban est $\bar{x}y$ avec la tête de lecture en face de la première lettre de y .

Bien entendu, Σ_M peut posséder d'autres modèles. Mais le théorème de complétude permet d'affirmer qu'une formule F du calcul des prédicats (logique du premier ordre) est prouvable si et seulement si elle est vraie dans tout modèle, c'est-à-dire quelle que soit l'interprétation des symboles de constantes, fonctions et relations.

1. Rappelons qu'on dit qu'une formule close est satisfiable si et seulement si elle possède un modèle.

Question 4. Soit M une machine de Turing à demi-ruban infini. Montrer que l'on peut construire une formule close F_M qui est prouvable (vraie dans tout modèle) si et seulement si la machine de Turing à demi-ruban infini M accepte le mot vide.

Solution : On peut écrire que la machine part avec le mot vide par $F_{q_0}(\epsilon, \epsilon)$. On peut décrire le programme de M de la façon suivante :

- Si le programme de M dit quand on est dans l'état q et qu'on lit un a , d'aller dans l'état r , et d'écrire un b et d'aller à droite :

$$\forall x \forall y (f_q(x, a(y)) \Rightarrow f_r(b(x), y))$$

- Si le programme de M dit quand on est dans l'état q et qu'on lit un a , d'aller dans l'état r , et d'écrire un b , et d'aller à gauche : on ajoute

$$\forall x \forall y (f_q(c(x), a'(y)) \Rightarrow f_r(x, c(b(y))))$$

pour chaque symbole $c \in \Sigma$.

On remarquera qu'il est important dans ce cas que la machine soit à semi-ruban infini, de telle sorte qu'il y ait bien un symbole à gauche.

- On doit couvrir aussi des variantes des formules précédentes pour cas de la lecture d'un blanc (d'une case visitée pour la première fois).
pour un déplacement à droite :

$$\forall x \forall y (f_q(x, \epsilon) \Rightarrow f_r(b(x), \epsilon)).$$

pour un déplacement à gauche :

$$\forall x \forall y (f_q(c(x), \epsilon) \Rightarrow f_r(x, c(b(\epsilon))));$$

$$\forall x \forall y (f_q(t, \epsilon, \epsilon) \Rightarrow f_r(\epsilon, b(\epsilon))).$$

On prend comme formule T_M la conjonction de toutes ces formules (qui sont bien en nombre fini), et prendre comme formule F_M la Formule $T_M \Rightarrow \exists x \exists y f_{q_a}(x, y)$ qui dit qu'un calcul doit être acceptant.

Par construction, si M accepte le mot vide, chacune des implications suivantes sera vraie dans n'importe quel modèle. Réciproquement, si la formule est vraie dans n'importe quel modèle, elle doit l'être dans le modèle particulier \mathfrak{M}_M , et donc M accepte le mot vide. \square

Question 5. Montrer qu'une formule close F du calcul des prédicats (logique du premier ordre) est satisfiable si et seulement si sa négation $\neg F$ n'est pas prouvable.

Solution : Cela découle du théorème de complétude qui démontre qu'une formule est prouvable si et seulement si elle est vraie dans tout modèle. Si $\neg F$ est prouvable, dans tout modèle, on doit avoir $\neg F$ vrai, et donc tout modèle de F serait nécessairement modèle de F et $\neg F$, ce qui est impossible. Réciproquement si $\neg F$ n'est pas prouvable, la preuve du théorème de complétude fournit un modèle qui satisfait F . \square

Question 6. On considère le problème $SAT_{\text{prédicat}}$ de décision suivant :

- **Donnée:** Une formule close F du calcul des prédicats (logique du premier ordre).
- **Réponse:** F est-elle satisfiable.

Ce problème est-il décidable ? Récursivement énumérable ? Dans NP , NP -complet, dans P ? Justifier votre réponse.

Solution : Il est indécidable. En effet, c'est la même chose que décider si la négation de F est prouvable (en inversant la réponse), et la question 4 réduit le problème indécidable de la question 3 à celui-ci.

On sait que F est satisfiable si et seulement si $\neg F$ n'est pas prouvable. Pour décider si $\neg F$ est prouvable, on peut énumérer toutes les preuves possibles jusqu'à trouver cette preuve. Donc son complémentaire est récursivement énumérable : il est récursivement énumérable de savoir si F n'est pas satisfiable. Comme on sait qu'un problème qui est récursivement énumérable et donc le complémentaire est récursivement énumérable est nécessairement décidable, et qu'on sait qu'il est indécidable, il n'est pas récursivement énumérable.

Il n'est pas dans NP, donc pas NP-complet, ni dans P car toutes ces classes sont des classes de langages décidables. \square

Question 7. La signature précédente possède des symboles d'arité 1 et 2.

Etant donné un entier k , on considère le problème $SAT_{prédicat,k}$ de décision suivant :

- **Donnée:** Une formule close F du calcul des prédicats (logique du premier ordre) sur une signature avec des symboles d'arité au plus k .
- **Réponse:** F est-elle prouvable.

Montrer que pour certains entier k que l'on précisera, le problème $SAT_{prédicat,k}$ est RE-complet : il est récursivement énumérable, et pour tout problème A récursivement énumérable, $A \leq_m SAT_{prédicat,k}$.

On admettra qu'il est décidable pour $k = 1$.

Solution : Pour tout entier k , le problème est récursivement énumérable : pour savoir si F est prouvable, on peut énumérer toutes les preuves possibles jusqu'à trouver cette preuve.

Pour $k \geq 2$, il est RE-difficile : étant donné un problème A récursivement énumérable, on sait qu'il existe une machine de Turing M qui accepte A . La question 4 produit une formule F_M sur une signature avec des symboles d'arité 1 et 2 qui est prouvable si et seulement si M accepte w . Pour $k > 2$, on peut ajouter des symboles d'arité k , sans que cela ne change rien. Cela constitue une réduction de A vers ce problème, si l'on observe qu'à chaque fois il est facile de produire F_M à partir de M .

Pour $k = 1$, il ne peut pas être RE-complet, car on sait qu'il est décidable, et qu'il y a des problèmes récursivement énumérables qui ne sont pas décidables. \square

3 NP-complétude

On s'intéresse dans cette question au problème de décision suivant : SET PACKING :

- **Donnée:** Un ensemble fini U , un ensemble \mathcal{C} composé de sous-ensembles S_1, \dots, S_t de U , et un entier z .
- **Réponse:** Existe-il z éléments de \mathcal{C} deux-à-deux disjoints ?

Rappelons qu'une *couverture de sommets* S d'un graphe G est un sous-ensemble de sommets tel que toutes les arêtes de G ont au moins une extrémité dans S .

Partant d'un graphe $G = (V, E)$, nous allons construire un ensemble \mathcal{C} composé de $|V|$ éléments de la façon suivante : pour chaque sommet $v \in V$, on définit l'ensemble S_v comme l'ensemble de toutes les arêtes qui ont v comme extrémité.

Question 8. Montrer que le graphe possède une couverture de sommets avec k sommets si et seulement si $n - k$ éléments de \mathcal{C} sont deux-à-deux disjoints.

Solution : Supposons que A soit une couverture de sommets avec k sommets. Considérons $B = V - A$ (le complémentaire de A).

Nous allons prouver par contradiction que si v et u appartiennent à B , alors S_v et S_u sont deux ensembles disjoints. En effet, supposons que S_v et S_u ne soient pas deux ensembles disjoints. Cela signifie qu'il existe une arête e qui est dans S_v et S_u . Cette arête e est nécessairement l'arête (u, v) , mais on a supposé que ni u ni v n'appartiennent à A , et donc cette arête ne serait pas couverte par A .

Les ensembles S_v et S_u sont donc disjoints deux-à-deux, et donc les $n - k$ éléments de \mathcal{C} sont deux-à-deux disjoints.

Réciproquement, supposons qu'il existe $n - k$ éléments de \mathcal{C} deux-à-deux disjoints. Notons les $S_{v_1}, \dots, S_{v_{n-k}}$. Considérons $B = \{v_1, \dots, v_{n-k}\}$ et $A = V - B$ (son complémentaire). Nous allons prouver que A est une couverture sommet de G . En effet, pour chaque arête $e = (u, v)$, il n'est pas possible que ni u ni v soient dans B . En effet, par construction, on aurait $e \in S_u$ et $e \in S_v$. Ceci contredirait que S_u, \dots, S_v soient des ensembles disjoints.

Puisqu'il est impossible que ni u ni v soient dans B , au moins l'un des deux est dans A . Autrement dit, A est une couverture de sommets de G , □

Question 9. Montrer que le problème SET PACKING est NP-complet.

On utilisera la NP-complétude du problème COUVERTURE DE SOMMETS (VC)

— **Donnée:** Un graphe non-orienté $G = (V, E)$ et un entier k .

— **Réponse:** G admet-il une couverture de sommets S avec au plus k sommets ?

Solution : Le problème SET PACKING est dans NP car la donnée des z éléments de \mathcal{C} constitue un certificat vérifiable facilement en temps polynomial.

En effet, on peut vérifier en temps polynomial si deux éléments J et I de \mathcal{C} sont deux-à-deux disjoints : un algorithme naïf (comparer deux-à-deux les éléments) peut se faire en $\mathcal{O}(|J| \cdot |I|)$ opérations. Décider si z éléments de \mathcal{C} sont deux-à-deux disjoints peut se faire en temps polynomial ($\mathcal{O}(|U|^2 z^2)$ opérations).

On peut réduire VC à SET PACKING de la façon suivante : Soit (G, k) une instance du problème VC. Soit n le nombre de sommets de G .

On construit une instance SET PACKING de la façon suivante

1. l'ensemble U de u éléments correspond à l'ensemble des arêtes : $U = E$;
2. l'ensemble \mathcal{C} est composé de sous-ensembles S_1, \dots, S_n : S_v est composé de toutes les arêtes adjacentes à v ;
3. $z = n - k$.

La transformation se fait bien en temps polynomial.

Par la question précédente, le graphe G possède une couverture de sommets S de taille k si et seulement s'il existe $n - k$ éléments de \mathcal{C} deux-à-deux disjoints.

On a donc bien une réduction de VC vers SET PACKING. □

4 Théorème de Hall : du cas fini au cas infini

On note \mathbb{N}^+ pour l'ensemble des entiers naturels non-nuls. On considère $I = \mathbb{N}^+$ ou $I = \{1, 2, \dots, n\}$ dans tout ce qui suit.

Etant donné un ensemble S et une famille $(S_i)_{i \in I}$ de sous-ensembles de S , un *système de représentants distincts* est un choix d'éléments (c'est-à-dire une famille $(x_i)_{i \in I}$) avec pour tout i , $x_i \in S_i$ tel que pour $i \neq j \in I$, on a $x_i \neq x_j$.

Par exemple, pour $S = \mathbb{N}^+$, $I = \mathbb{N}^+$, et la famille $(S_i)_{i \in \mathbb{N}^+}$, définie par $S_i = \{i, i + 1\}$, on a $S_1 = \{1, 2\}$, $S_2 = \{2, 3\}$, ..., on peut prendre comme système de représentants distincts $x_i = i$.

On admettra le résultat suivant², pour le cas où la famille (c'est-à-dire I) est finie :

Théorème [Hall 1935]. Considérons un entier $n > 0$. Une condition nécessaire est suffisante pour l'existence d'un système de représentants distincts pour une famille $(S_i)_{1 \leq k \leq n}$ est la suivante :

(H_n) Pour tout $1 \leq k \leq n$, et tout choix d'indices distincts $1 \leq i_1, \dots, i_k \leq n$, on a $|S_{i_1} \cup \dots \cup S_{i_k}| \geq k$.

Dans cette partie, on cherche à étendre ce théorème au cas d'une famille infinie.

Question 10. Une première tentative serait de penser qu'une condition nécessaire et suffisante pour l'existence d'un système de représentants distincts pour une famille $(S_i)_{i \in \mathbb{N}^+}$ est la suivante :

(H_∞) Pour tout $k \in \mathbb{N}^+$, et tout choix d'indices distincts $i_1, \dots, i_k \in \mathbb{N}^+$, on a

$$|S_{i_1} \cup \dots \cup S_{i_k}| \geq k.$$

Montrer que cela ne suffit pas, en exhibant une famille de sous-ensembles de \mathbb{N} (les entiers naturels) pour lesquels il n'existe pas de système de représentants distincts. On pourra choisir $S_1 = \mathbb{N}$, et les autres S_i finis.

Solution : Prendre par exemple $S_1 = \mathbb{N}$, et $S_2 = \{0\}$, $S_{n+2} = \{n\}$ pour tout n . Il ne peut pas y avoir de système de représentant distincts, car si on a choisi l'indice i pour S_1 , on ne peut pas le choisir pour S_{i+2} , et donc S_{i+2} n'a pas d'indice. \square

Par contre, cela fonctionne si l'on suppose que chaque S_i est **fini**. En effet, nous allons prouver le résultat suivant :

Théorème. On considère une famille $(S_i)_{i \in \mathbb{N}^+}$ d'ensembles **finis** de S .

Une condition nécessaire et suffisante pour l'existence d'un système de représentants distincts pour une famille $(S_i)_{i \in \mathbb{N}^+}$ est la suivante :

(H_∞) Pour tout $k \in \mathbb{N}^+$, et tout choix d'indices distincts $i_1, \dots, i_k \in \mathbb{N}^+$, on a

$$|S_{i_1} \cup \dots \cup S_{i_k}| \geq k.$$

Question 11. Prouver que (H_∞) est une condition nécessaire.

Solution : Si on a un système de représentants distincts pour la famille $(S_i)_{i \in \mathbb{N}^+}$, cela en fournir un pour toute sous-famille extraite de taille n et par le théorème de Hall, on doit avoir l'hypothèse H_n pour tout n . Ce qui implique la propriété. \square

Question 12. Prouver l'autre direction du théorème. Indication : On pourra chercher à définir une théorie qui est satisfiable si et seulement si la famille admet un système de représentants distincts.

Solution : On introduit la famille de formules $C_{n,x}$, avec $x \in S_n$.

On considère l'ensemble Σ constitué de toutes les formules :

$$(a) \neg(C_{n,x} \wedge C_{m,x}) \text{ pour } n \neq m \in \mathbb{N}^+, x \in S_n \cap S_m.$$

$$(b) \neg(C_{n,x} \wedge C_{n,y}) \text{ pour } n \in \mathbb{N}^+, x \neq y \in S_n \cap S_m.$$

$$(c) C_{n,x_1} \vee \dots \vee C_{n,x_k} \text{ pour } n \in \mathbb{N}^+, \text{ où } S_n = \{x_1, \dots, x_k\}.$$

2. Qui n'est pas difficile à démontrer.

Supposons qu'on arrive à satisfaire Σ . On peut alors obtenir un système de représentants distincts en choisissant x dans S_n si et seulement si $C_{n,x}$ est vrai dans l'affectation qui satisfait Σ .

En effet, par (b) et (c), chaque S_n se verra affecté d'un unique représentant. Par (a), on affectera pas le même représentant à plusieurs ensembles.

Σ est finiment satisfiable. En effet, pour cela, considérons un sous-ensemble Σ_0 de Σ , et appelons i_1, \dots, i_l les indices qui apparaissent dans Σ_0 . La famille $\{S_{i_1}, S_{i_2}, \dots, S_{i_l}\}$ doit alors vérifier l'hypothèse H_∞ . Par le théorème de Hall, il y a un système de représentants distincts pour $\{S_{i_1}, S_{i_2}, \dots, S_{i_l}\}$. Considérer $C_{i_r,x}$ vrai si et seulement si $x = x_r$. Cela satisfait Σ_0 .

Par le théorème de Compacité, puisque Σ est finiment satisfiable, Σ est satisfiable, et donc on a un système de représentants distincts. □

5 Hiérarchie arithmétique : vision logique

Dans cette partie, on considère des formules sur la signature $\mathcal{L} = (0, s, +, \times, <, =)$ de l'arithmétique.

Une formule sur cette signature est dite Σ_0 si elle appartient au plus petit ensemble contenant les formules atomiques et clos par :

- conjonction \wedge (finie), disjonction \vee (finie), négation \neg ;
- quantification universelle bornée : si φ est Σ_0 , si t est un terme, et x une variable n'apparaissant pas dans t , alors $\forall x \leq t \varphi$ est Σ_0 (ici, $\forall x \leq t \varphi$ est une abréviation pour $\forall x (x \leq t \Rightarrow \varphi)$);
- quantification existentielle bornée, que l'on définit de façon similaire.

Par convention, on considère que Π_0 et Σ_0 sont des synonymes.

Pour $n \in \mathbb{N}$, on dit d'une formule qu'elle est :

- Σ_{n+1} si elle est de la forme $\exists x \varphi$ pour φ une formule Π_n ;
- Π_{n+1} si elle est de la forme $\forall x \varphi$ pour φ une formule Σ_n .

L'intuition est que Σ (respectivement : Π) signifie que l'on commence par une quantification existentielle (respectivement universelle), et l'indice indique le nombre d'alternances de quantificateurs, sans compter les quantifications bornées. Par exemple, $\exists n \forall m \leq n \ n \leq m \times m$ est une formule Σ_1 , et $\exists n \forall j \forall m \leq n \ m \leq j \times n$ est une formule Σ_2 .

On cherche à comprendre les sous-ensembles de \mathbb{N}^p que l'on arrive à définir avec ces formules : on dit qu'une formule $\phi(x_1, \dots, x_p)$ a p -variables libres définit un sous-ensemble de \mathbb{N}^p : cet ensemble correspond à l'ensemble des p -uplets qui la satisfait. Un ensemble est alors dit Σ_n (respectivement : Π_n) définissable si ϕ est une formule Σ_n (respectivement : Π_n). On dit qu'il est Δ_n si il est à la fois Σ_n et Π_n définissable.

Pour $n = 0$, on peut se convaincre par (induction sur la forme des formules) que tout ensemble définit par une formule $\Pi_0 = \Sigma_0$ est décidable. Les sous-ensembles Δ_0 sont donc décidables.

Question 13. *Montrer que si une partie A de \mathbb{N}^p est Σ_1 alors A est récursivement énumérable.*

Solution : On note \vec{x} pour x_1, \dots, x_p .

Supposons que $A \subseteq \mathbb{N}^p$ soit Σ_1 : cela implique que déterminer si $\vec{x} \in A$ est équivalent à déterminer si $\exists u \phi$ pour une formule $\phi = \phi(\vec{x}, u)$ qui est Π_0 , et donc décidable. Soit M la machine de Turing qui, étant donné \vec{x} et u décide si $\phi(\vec{x}, u)$. A est semi-décidé par l'algorithme qui consiste, sur l'entrée \vec{x} , à tester pour chaque u si $\phi(\vec{x}, u)$, et accepte dès qu'elle trouve un tel u . □

On admettra que toute partie A de \mathbb{N}^p décidable est Σ_1 . On pourra aussi admettre qu'on peut³ construire une bijection $\pi : \mathbb{N} \rightarrow \mathbb{N}^2$ telle que si l'on note $\pi(r) = (\pi_1(r), \pi_2(r))$, les relations $u = \pi_1(r)$ et $v = \pi_2(r)$ sont Δ_0 .

Question 14. *Montrer la réciproque : une partie A de \mathbb{N}^p est Σ_1 si elle est récursivement énumérable.*

Solution : Supposons que A soit récursivement énumérable. A est reconnu par une machine de Turing M . On a $\vec{x} \in A$ si et seulement s'il existe un temps t tel que M accepte \vec{x} au temps t . Puisque " M accepte \vec{x} au temps t " est décidable, et qu'on admet que décidable implique Σ_1 , donc peut s'écrire sous la forme $\exists u \phi(\vec{x}, t, u)$ pour une certaine formule $\Pi_0 = \Sigma_0$. On a alors $\vec{x} \in A$ qui s'écrit $\exists r \phi(\vec{x}, \pi_1(r), \pi_2(u))$. \square

Question 15. *En déduire que les ensembles Δ_1 de \mathbb{N}^p sont exactement les ensembles décidables de \mathbb{N}^p .*

Solution : Les ensembles décidables correspondent aux ensembles semi-décidables dont le complémentaire est semi-décidable. Le complémentaire d'un ensemble Σ_1 correspond à un ensemble Π_1 . Cela découle donc de la question précédente, et de la définition de Δ_1 . \square

Fixons $p \geq 1$. On dit qu'un ensemble $U \subseteq \mathbb{N}^{p+1}$ est Σ_n -universel s'il est dans Σ_n et si pour tout $A \subseteq \mathbb{N}^p$ qui est dans Σ_n , il existe $i \in \mathbb{N}$ tel que $A = U_i$, où $U_i = \{(x_1, \dots, x_p) \mid (i, x_1, \dots, x_p) \in U\}$. On dit qu'un ensemble est Π_n universel, si on a la même propriété en remplaçant Σ_n par Π_n dans la phrase précédente.

On fixe un codage⁴ des machines de Turing par les entiers, de telle sorte que l'on puisse parler de la machine de Turing numéro n .

Question 16. *Montrer que $U = \{(i, x_1, \dots, x_p) \mid \text{la machine de Turing } i \text{ accepte } (x_1, \dots, x_p)\}$ est Σ_1 -universel.*

Solution : U correspond à un problème de décision récursivement énumérable, donc Σ_1 . D'autre part, soit A un ensemble Σ_1 de $\mathbb{N}_p : A$ est donc accepté par une machine de Turing M , qui possède un numéro i . L'ensemble A correspond alors à U_i . \square

Question 17. *Montrer que pour tout $n \geq 1$, et pour tout $p \geq 1$, il existe un ensemble Σ_n -universel, et un ensemble Π_n -universel dans \mathbb{N}^{p+1} .*

Solution : Pour $U \subseteq \mathbb{N}^{p+1}$, notons \bar{U} pour son complémentaire. Observons que $\bar{U}_i = \{(\vec{x}) \mid (i, \vec{x}) \notin U\}$ est le complémentaire de $U_i = \{(\vec{x}) \mid (i, \vec{x}) \in U\}$.

Notons U^1 pour le langage de la question précédente.

Ayant défini le langage U^n , définissons $U^{n+1} = \{(n, \vec{x}) \mid \exists u (n, u, \vec{x}) \in \bar{U}^n\}$.

On montre par récurrence que U^n est Σ_n -universel et son complémentaire \bar{U}^n est Π_n -universel.

Pour $n = 1$, il reste à voir que \bar{U}^1 est Π_1 -universel. Mais si A est Π_1 , son complémentaire est Σ_1 , et donc correspond à U_i^1 pour un certain entier i , et donc A correspond à \bar{U}_i^1 .

Supposons la propriété pour n . Un langage A de Σ_{n+1} s'écrit sous la forme $\exists u \phi$, avec ϕ qui définit un langage qui est Π_n . Par hypothèse de récurrence, cela correspond à \bar{U}_i^n pour un certain entier i . On a donc bien que A est U_i^{n+1} . Pour un langage A de Π_{n+1} , on peut raisonner sur son complémentaire qui est Σ_{n+1} et obtenir la conclusion.

Par ailleurs, on se persuade facilement par récurrence sur n que U^n correspond bien à une formule Σ_n et symétriquement pour son complémentaire. \square

3. C'est très facile.

4. Raisonnable, c'est-à-dire comme dans le cours, tel que l'on puisse bien retrouver chacun des ingrédients de la machine (programme, état initial, etc..) à partir de l'entier qui le code.

Question 18. *Montrer que les inclusions $\Delta_n \subseteq \Sigma_n$ et $\Delta_n \subseteq \Pi_n$ sont strictes pour $n \geq 1$.*

Solution : Il suffit de prouver que le langage universel U^n (qui est dans Σ_n) n'est pas dans Δ_n . Cela prouve la première inclusion stricte, et la seconde, puisque son complémentaire (qui est dans Π_n) ne saurait être dans Δ_n car Δ_n est close par complémentaire.

Montrons donc que le langage universel U^n (qui est dans Σ_n) n'est pas dans Δ_n : supposons par l'absurde qu'il l'était. On peut considérer $A = \{i | (i, i) \in U^n\}$. Δ_n étant close par complément, le complémentaire \bar{A} de A serait aussi dans Δ_n . Puisque $\Delta_n \subseteq \Sigma_n$ et que Σ_n est universel, il doit exister un entier i_0 tel que $\bar{A} = U^n_{i_0}$. Mais on obtient $i_0 \notin A$ si et seulement si $(i_0, i_0) \in U^n$ si et seulement si $i_0 \in A$ ce qui est impossible. \square

Question 19. *Montrer que chacun des niveaux Σ_n et Π_n pour $n \geq 1$ possède des problèmes complets. Donner un exemple de tels problèmes pour chacun des niveaux.*

Solution : Un problème U Σ_n -universel est nécessairement Σ_n -complet, . En effet, il est dans Σ_n par définition, et d'autre part, si on a un problème A qui est Σ_n , il se réduit à U par la fonction qui à \vec{x} associe (i, \vec{x}) , où i est l'indice correspondant à A .

Symétriquement pour Π_n .

On connaît donc des exemples par les questions précédentes pour chacun des niveaux. \square

Il est aussi possible de donner des caractérisations de chacun des niveaux de cette hiérarchie en utilisant des machines de Turing étendues (avec des oracles).

Fondements de l’informatique. Examen

Durée: 3h

Sujet proposé par Olivier Bournez

Version 9

(corrigé)

Les 4 parties sont indépendantes, et peuvent être traitées dans un ordre quelconque. On pourra admettre le résultat d’une question pour passer aux questions suivantes. On pourra utiliser tous les résultats et les théorèmes démontrés ou énoncés en cours ou en petite classe ou dans le polycopié sans chercher à les redémontrer.

Il est possible d’avoir la note maximale sans répondre à toutes les questions. La difficulté des questions n’est pas une fonction linéaire ni croissante de leur numérotation.

*La **qualité et clarté de votre argumentation et de votre rédaction** sera une partie importante de votre évaluation.*

1 A propos des gestes barrières

Question 1. *Parmi les questions suivantes, lesquelles sont décidables ? Récursivement énumérable ? dans P ? Justifier votre réponse.*

- 1. Déterminer si une machine de Turing¹ M est telle qu’elle accepte un langage qui contient le mot `appliquonslesgestesbarrieres`.*
- 2. Déterminer si un programme Python contient une variable dont le nom est `appliquonslesgestesbarrieres` et l’instruction “`appliquonslesgestesbarrieres=1`”*
- 3. Déterminer si un programme Python possède une variable dont le nom est `appliquonslesgestesbarrieres`, qui sera modifiée à un moment lorsqu’il sera exécuté.*

Solution : Le premier problèmes est indécidable. C’est une application directe du Théorème de Rice : il y a une machine qui reconnaît le langage réduit à ce mot, et une machine de Turing qui accepte le mot vide, et donc la propriété est bien non triviale. Il est récursivement énumérable, car il suffit de simuler la machine M sur la chaîne de caractère `appliquonslesgestesbarrieres` et d’accepter si et seulement si la simulation accepte pour le reconnaître. Il n’est pas dans P car les langages de P sont décidables.

Le second problème est décidable (et donc récursivement énumérable) et est dans P : il suffit de parcourir le programme à la recherche de ce mot.

Le troisième problèmes est indécidable : Le problème universel se réduit à ce problème. Étant donnée une instance (M, w) du problème universel, on peut construire un programme $P_{M,w}$ en *Python*, qui possède une variable dont le nom est `appliquonslesgestesbarrieres`, qui simule la machine de Turing M sur w , et si cette simulation termine avec M qui accepte, ajoute 1 à la variable `appliquonslesgestesbarrieres` : la fonction qui à M et w associe $P_{M,w}$ est bien facilement calculable.

Il est récursivement énumérable, car il suffit de simuler le programme pour déterminer si la propriété est vérifiée. Il n’est pas dans P car les langages de P sont décidables.

□

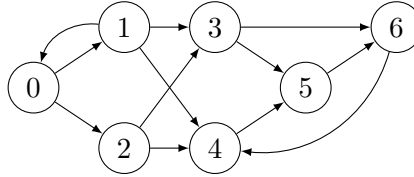
1. Qui travaille sur l’alphabet des symboles de l’alphabet Latin.

2 NP-complétude de Coupe-cycles

A propos des notations : On note $G = (V, E)$ pour un graphe dont l'ensemble des sommets est V , et $E \subseteq V \times V$ ses arêtes (dans le cas non-orienté) ou ses arcs (dans le cas orienté). On notera (u, v) pour l'arête de u vers v dans le cas non-orienté, et $(u \rightarrow v)$ pour l'arc de u vers v dans le cas orienté. Rappel : un cycle (orienté) de longueur n est une suite d'arcs du type $(u_1 \rightarrow u_2)$, $(u_2 \rightarrow u_3)$, $\dots (u_{n-1} \rightarrow u_n)$ et $(u_n \rightarrow u_1)$. Une boucle est un cycle de longueur 1, c'est-à-dire un arc $(u \rightarrow u)$.

Soit $G = (V, E)$ un graphe orienté sans boucle. Un **coupe-cycles** est un ensemble d'arcs du graphe, tel que la suppression de ces arcs rend le graphe sans cycle.

Par exemple l'ensemble des arcs $\{(1 \rightarrow 0), (6 \rightarrow 4)\}$ est un coupe-cycles du graphe orienté suivant :



On rappelle qu'une *couverture de sommets* d'un graphe $G = (V, E)$ est un ensemble $S \subseteq V$ de sommets tel que toutes les arêtes de G ont au moins une extrémité dans S .

On rappelle/admettra que le problème COUVERTURE-DE-SOMMETS suivant est NP-complet :

Données : un graphe non-orienté G sans boucle et un entier k .

Question : G contient-il une couverture de sommets avec au plus k sommets.

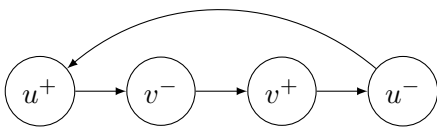
On introduit la transformation \mathcal{F} qui à un graphe $G = (V, E)$ non-orienté associé le graphe orienté $\mathcal{F}(G) = (V', E')$ défini par :

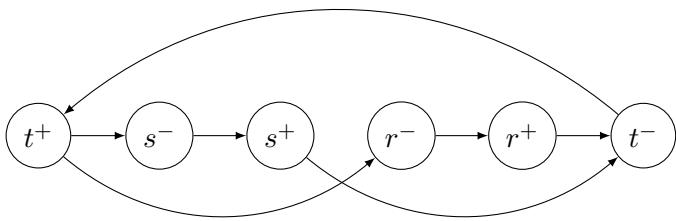
- $V' = \{u^-, u^+ : u \in V\}$;
- $E' = \{(u^- \rightarrow u^+) : u \in V\} \cup \{(u^+ \rightarrow v^-), (v^+ \rightarrow u^-) : (u, v) \in E\}$.

Question 2. — Représentez graphiquement $\mathcal{F}(G)$ et décrire un coupe-cycles de $\mathcal{F}(G)$ de la taille la plus petite possible pour $G = (V, E)$ avec $V = \{u, v\}$ et $E = \{(u, v)\}$.

- Même question pour $G = (V, E)$ avec $V = \{s, t, r\}$ et $E = \{(s, t), (t, r)\}$.

Solution :

$\mathcal{F}(G_1)$ est donné par  et admet $\{(u^- \rightarrow u^+)\}$ comme coupe-cycles

$\mathcal{F}(G_2)$ est donné par  et admet $\{(t^- \rightarrow t^+)\}$ comme coupe-cycles.

□

On note $|S|$ pour le nombre d'éléments de l'ensemble S .

Question 3. Montrer que si C est un coupe-cycles du graphe $\mathcal{F}(G)$, alors il existe un ensemble S avec $|S| \leq |C|$ qui est une couverture de sommets du graphe G .

Solution :

On considère S tel que

— $u \in S$ si $(u^- \rightarrow u^+) \in C$ ou si $(u^+ \rightarrow v^-) \in C$.

Puisque cela garantit $|S| \leq |C|$, il suffit de prouver que S est une couverture sommets de G .

Soit $e = (u, v)$ une arête de G . Observons que $u^-u^+v^-v^+u^-$ est un cycle de $\mathcal{F}(G)$. Cela signifie qu'au moins un des ces 4 arcs doit être dans C .

— Si $(u^- \rightarrow u^+) \in C$ alors cela signifie que u est dans S .

— Si $(v^- \rightarrow v^+) \in C$ alors cela signifie que v est dans S .

— Si $(v^+ \rightarrow v^-) \in C$, alors cela signifie que v est dans S .

— Si $(u^+ \rightarrow u^-) \in C$, alors cela signifie que u est dans S .

Donc pour les 4 cas, S couvre bien l'arête e . Donc S est bien une couverture de sommets. \square

Question 4. Montrer que si S est une couverture de sommets du graphe non-orienté G , alors l'ensemble $\{(u^- \rightarrow u^+) : u \in S\}$ est un coupe-cycles du graphe $\mathcal{F}(G)$.

Solution : [Sur le vocabulaire utilisé dans ce qui suit : lorsqu'on a un arc $(u \rightarrow v)$, on dit que cet arc est un arc sortant en u , et entrant en v .]

Pour tout sommet u , le seul arc sortant en u^- va vers u^+ . Les arcs entrants sont tous vers des v^- , pour $v \in V$. Par conséquent, un cycle doit donc alterner les $-$ et les $+$.

Soit C un cycle de $\mathcal{F}(G)$. Le cycle C est de longueur paire par la remarque précédente. Il ne peut pas être de longueur 2 car il n'y a pas de boucle dans $\mathcal{F}(G)$. Il est donc de la forme $u^-u^+v^-v^+ \dots u^-$ pour un certain u et un certain v , et une arête (u, v) de G . Puisque S est une couverture de sommets, soit u soit v est dans S , et donc l'un des arcs $(u^- \rightarrow u^+)$ ou $(v^- \rightarrow v^+)$ coupe le cycle et appartient à l'ensemble $\{(u^- \rightarrow u^+) : u \in S\}$. \square

Question 5. Démontrer que le problème de décision suivant est NP-complet :

Données : un graphe orienté G sans boucle et un entier k .

Question : G contient-il un coupe-cycles avec au plus k arcs ?

Solution : Le problème est dans NP, car la donnée d'un couple-cycles est un certificat vérifiable en temps polynomial : il suffit de vérifier qu'il ne possède pas plus de k arcs, et que c'est bien un coupe-cycles du graphe (ce qui peut se faire en vérifiant que le graphe obtenu en enlevant ces arcs est sans cycles). Tout cela peut bien se faire en temps polynomial.

Les questions précédentes donnent une réduction via la fonction \mathcal{F} entre le problème de couverture de sommets et ce problème. Il ne reste plus qu'à observer que \mathcal{F} se calcule clairement en temps polynomial pour conclure. \square

3 Un peu de définissabilité

Question 6. Soit T une théorie sur une signature du calcul des prédicats qui contient le symbole $=$.

Montrer que si T possède des modèles égalitaires dont l'ensemble de base est de taille arbitrairement grande, alors T possède un modèle égalitaire dont l'ensemble de base est infini.

Solution : Par hypothèse, toute partie finie de $T \cup \{F_n : n \in \mathbb{N}\}$ possède un modèle, où $F_n = \exists v_1 \exists v_2 \dots \exists v_n \bigwedge_{1 \leq i < j \leq n} \neg v_i = v_j$ exprime qu'il y a au moins n éléments.

Par le théorème de compacité, $T \cup \{F_n : n \in \mathbb{N}\}$ possède un modèle. Ce modèle a son ensemble de base infini, puisque il possède au moins n éléments pour tout n . \square

Question 7. Supposons que T est une théorie sur une signature du premier ordre avec un nombre fini de formules.

Supposons que l'on ait deux théories Σ et Λ avec la propriété que pour tout modèle \mathcal{A} de T , \mathcal{A} est un modèle de Σ si et seulement si \mathcal{A} n'est pas un modèle de Λ .

Montrer que les théories $T \cup \Sigma$ et $T \cup \Lambda$ sont finiment axiomatisables : on peut construire pour chacune une théorie avec un nombre fini de formules qui leur est équivalente.

Solution : Par hypothèse, $T \cup \Sigma \cup \Lambda$ ne possède aucun modèle. Par le théorème de compacité, il y a un sous-ensemble fini de celle-ci qui ne possède aucun modèle : donc un sous-ensemble T de T , Σ' de Σ , et Λ' de Λ tel que $T' \cup \Sigma' \cup \Lambda'$, et donc à fortiori $T \cup \Sigma' \cup \Lambda'$ ne possède aucun modèle.

Un modèle de $T \cup \Sigma'$ est donc un modèle de T et de la formule Φ qui est la conjonction de la négation de toutes les formules de Λ' . La propriété est vraie à fortiori pour les modèles de $T \cup \Sigma$. Réciproquement, un modèle de Φ et de T doit être un modèle de $T \cup \Sigma$, par la propriété de l'énoncé. Autrement dit, $T \cup \Sigma$ s'axiomatise par $T \cup \{\Phi\}$.

Symétriquement, $T \cup \Lambda$ s'axiomatise par $T \cup \{\Psi\}$, où Ψ est la conjonction de la négation de toutes les formules de Σ' . \square

4 Programmer avec des additions de vecteurs ordonnés

On considère un langage de programmation inspiré par une variation du modèle des *VECTOR ADDITION SYSTEMS* et de considérations sur un autre modèle dû à John Conway : un programme $VASS_{ord}$ est une liste V ordonnée finie de vecteurs de \mathbb{Z}^m pour un certain entier m .

On exécute le programme V sur une entrée $\vec{n} \in \mathbb{N}^m$ de la façon suivante :

1. pour le premier vecteur \vec{v} dans la liste V pour lequel $\vec{n} + \vec{v}$ est un élément de \mathbb{N}^m , on remplace \vec{n} par $\vec{n} + \vec{v}$.
2. on répète la règle 1. tant qu'on peut appliquer cette règle, c'est-à-dire, jusqu'à ce qu'il n'y ait plus de vecteur \vec{v} avec $\vec{n} + \vec{v}$ qui reste dans \mathbb{N}^m . On dit alors que le résultat du calcul est ce (dernier) vecteur \vec{n} .

(si la règle 1. s'applique pour toujours, on dit que V ne termine pas sur l'entrée \vec{n}).

Par exemple, le programme $VASS_{ord}$ donné par les vecteurs

$$V = (-1, -1, 0), (0, -1, 1)$$

appliqué à l'entrée $(2, 3, 0)$ va terminer avec $(0, 0, 1)$: en effet, il va transformer $\vec{n} = (2, 3, 0)$ en $(1, 2, 0)$ puisque $(2, 3, 0) + (-1, -1, 0) = (1, 2, 0)$, puis en $(0, 1, 0)$ puisque $(1, 2, 0) + (-1, -1, 0) = (0, 1, 0)$, puis en $(0, 0, 1)$ puisque $(0, 1, 0) + (-1, -1, 0)$ n'est pas un élément de \mathbb{N}^3 , mais que $(0, 1, 0) + (0, -1, 1) = (0, 0, 1)$.

Question 8. Que retourne ce programme sur $n = (a, b, 0)$ selon la valeur des entiers naturels a et b ?

Solution : Le premier vecteur décrémente a et b de 1 tant que cela est possible. On va donc appliquer ce vecteur jusqu'à obtenir $(a-b, 0, 0)$ si $a > b$ ou $(0, b-a, 0)$ si $b > a$, et $(0, 0, 0)$ si $a = b$. Le second vecteur va alors pouvoir s'appliquer que si $b-a > 0$, et tant que la seconde composante n'est pas nulle. Chaque application de ce vecteur décrémente de 1 la seconde composante, et augmente de 1 la troisième composante. En répétant, on va donc se retrouver avec $(0, 0, b-a)$, où cette fois plus aucun vecteur ne s'applique.

Le résultat est donc $(a-b, 0, 0)$ si $a \geq b$ et $(0, 0, b-a)$ sinon. \square

On rappelle que les machines à compteurs sont des machines qui n'ont que des instructions du type incrémenter un compteur (Inc), décrémente un compteur (Decr), tester si un compteur est nul (IsZero) et arrêt (Halt) (voir par exemple les transparents du cours 6).

On note $p_1 = 2, p_2 = 3, p_3 = 5, \dots$, et ainsi de suite, les nombres premiers.

On rappelle que $f : \mathbb{N}^m \rightarrow \mathbb{N}$ est une fonction partielle calculable s'il existe une machine de Turing M , avec m rubans, telle si l'on place initialement le codage de n_i en binaire sur le ruban i , alors si (n_1, \dots, n_m) est dans le domaine de f , alors M s'arrête avec $f(n_1, \dots, n_m)$ sur son premier ruban et tous les autres rubans vides, et M ne s'arrête pas si (n_1, \dots, n_m) n'est pas dans le domaine de f .

Question 9. Soit $f : \mathbb{N}^m \rightarrow \mathbb{N}$ une fonction partielle calculable.

Justifier pourquoi on peut construire une machine à 2 compteurs telle que si on l'exécute sur l'entier $n = 2^{n_1} 3^{n_2} 5^{n_3} 7^{n_4} \dots p_m^{n_m}$ dans son premier compteur, elle s'arrêtera avec l'entier $2^{f(n_1, n_2, \dots, n_m)}$ dans son premier compteur lorsque (n_1, \dots, n_m) est dans le domaine de f , et ne s'arrêtera pas sinon.

Solution : Le cours explique comment simuler une machine de Turing par une machine à 2-compteurs (en passant par les machines à 2 piles). La machine obtenue fait essentiellement ce travail quand elle simule la machine de Turing qui calcule f . On peut aussi passer via des machines à $2m$ piles si on préfère, pour gérer les m arguments/rubans.

Mais en réalité, ce n'est pas tout à fait cela car la machine M travaille sur des codages en binaire dans la définition plus haut, et non pas en unaire : le codage de n en binaire ne va pas correspondre à k^n dans la simulation du cours pour k un nombre premier.

Mais dans la définition plus haut de fonction partielle calculable, on pourrait remplacer "binaire" par "unaire", quitte à convertir les représentations au début et à la fin du calcul quand on s'arrête. Cette fois, la simulation du cours de cette dernière machine par une machine à compteur fait le travail voulu. \square

Question 10. Supposons que l'on change le type des instructions autorisées dans les machines à compteur. Au lieu d'avoir des instructions du type $\text{IsZero}(c, j, k)$, on autorise des instructions du type $\text{IsZero}'(c, j, k)$ dont l'effet est le suivant : cela teste si le compteur c est nul, et on va à l'instruction j si c'est le cas, et sinon on diminue de 1 le compteur de c et on va à l'instruction k .

Supposons par ailleurs que le programme n'est autorisé à ne contenir qu'une unique instruction Halt .

Justifier pourquoi cela ne change rien à la puissance du modèle obtenu des machines à compteur, en termes de ce qu'il est capable de calculer.

Solution : Il suffit de remplacer chaque instruction $\text{IsZero}(c, j, k)$ par $\text{IsZero}'(c, j, k')$ où k' est une nouvelle instruction $\text{Inc}(c, j, k)$, et le nouveau programme simulera l'ancien.

De façon duale, un programme avec cette instruction peut être simulé par les machines du cours, en remplaçant chaque instruction $\text{IsZero}'(c, j, k)$ par $\text{IsZero}(c, j, k')$ où k' est une nouvelle instruction $\text{Decr}(c, j, k)$.

On peut aussi supposer qu'il n'y a qu'une unique instruction Halt , en remplaçant chaque instruction Halt par $\text{Inc}(1, n)$ où n et $n + 1$ sont les instructions $\text{Decr}(1, n + 1)$ et Halt .

[On peut aussi simplement remplacer les adresses de saut vers Halt dans les instructions qui en ont, si vous préférez.] \square

Question 11. Soit M une machine à 2-compteurs. Expliquer comment on peut construire un programme VASS_{ord} qui simule sur $\vec{n} = (n, 0, \dots, 0)$ l'évolution de M sur l'entrée n .

[Indication : On pourra d'abord chercher à simuler sur $\vec{n} = (n, 0, 1, 0, \dots, 0)$ l'évolution de M sur l'entrée n , en travaillant dans \mathbb{N}^m avec m qui est le nombre d'instructions de M plus 2.]

Solution : L'idée est de simuler le fait que la valeur des 2 compteurs de M valent a, b et que la machine est dans l'instruction q par le fait que \vec{n} vaut $(a, b, 0, \dots, 0, 1, 0, \dots, 0)$, où le 1 est en

position q . On écrira un tel vecteur (abusivement) (a, b, \vec{e}_q) , où \vec{e}_q est le vecteur qui ne contient que des 0, sauf pour la composante q qui vaut 1.

En utilisant la question précédente, on peut supposer que les instructions sont avec l'opérateur $\text{IsZero}'(c, j, k)$.

Pour simuler $\text{Inc}(a, j)$ à l'instruction i , on introduit le vecteur $(1, 0, \vec{e}_j - \vec{e}_i)$.

Pour simuler $\text{Decr}(a, j)$ à l'instruction i , on introduit le vecteur $(-1, 0, \vec{e}_j - \vec{e}_i)$.

Pour simuler $\text{IsZero}'(a, j, \ell)$ à l'instruction i , on introduit les vecteurs $(-1, 0, \vec{e}_\ell - \vec{e}_i)$ et $(0, 0, \vec{e}_j - \vec{e}_i)$ dans cet ordre.

Pour les instructions où le premier argument vaut b , on intervertit la première et deuxième coordonnées dans ces vecteurs.

Cela résout le problème suggéré par l'indication.

Il reste à faire démarrer la simulation : donc à remplacer $\vec{n} = (n, 0, \dots, 0)$ par $\vec{n} = (n, 0, 1, 0, \dots, 0)$: il suffit d'ajouter le vecteur $(0, 0, 1, 0, \dots, 0)$ tout à la fin de la liste de vecteurs. On utilise le fait qu'il ne pourra s'appliquer que tout au début, car dans tout autre cas, un autre vecteur de ceux construits plus haut s'appliquera.

[On peut faire observer que cette façon de faire construit un programme VASS_{ord} qui simule M jusqu'à ce qu'il termine, mais lorsque M termine ne s'arrête pas lui, puisque le vecteur $(0, 0, 1, 0, \dots, 0)$ à la fin s'applique toujours : il boucle quand M s'arrête. Si l'on veut éviter cela, on peut, au lieu d'ajouter $(0, 0, 1, 0, \dots, 0)$ à la fin, plutôt ajouter un programme au début qui transforme $(n, 0, \dots, 0)$ en $(0, 0, n, 1, 0, \dots)$ quitte à travailler dans \mathbb{N}^{m+2} , et ajouter deux composantes nulles à chaque vecteur de la simulation précédente. En supposant sans perte de généralité que $n \neq 0$, une façon de faire cela est de mettre les vecteurs

$$(-1, 1, 1, -1, 0, \dots, 0), (0, -1, 0, 1, \dots, 0), (-1, 0, 1, 1, 0, \dots, 0)$$

au début du programme VASS_{ord} .]

□

John Conway a décrit un programme qui génère tous les nombres premiers : si on l'exécute sur $(2, 0, \dots, 0)$, il ne terminera pas, mais à certains instants \vec{n} sera de nouveau de la forme $(2^p, 0, \dots, 0)$, et cela se produira exactement pour p un entier premier. Et par ailleurs, le programme génère les entiers p dans ce sens dans l'ordre croissant.

Question 12. *Expliquer pourquoi il existe un tel programme. On ne demande pas de produire explicitement ce programme VASS_{ord} , mais d'expliquer comment il peut être obtenu.*

Note culturelle : John Conway a décrit explicitement un tel programme constitué de 14 vecteurs de \mathbb{Z}^{10} , qu'il a appelé les 14 vecteurs² fantastiques.

Solution : On peut construire une machine de Turing, telle que si on lui donne $n = p_i$ le i ème nombre premier, elle renvoie p_{i+1} : par exemple, de façon brutale, en tentant de diviser $n + 1$ par tous les entiers plus petit que lui, et en passant à $n + 2$ si l'on en trouve un, puis à $n + 3$ si l'on en trouve un autre, et ainsi de suite, jusqu'à trouver un nombre premier.

En utilisant la question 9, cela peut aussi se faire avec une machine à compteur. Cette dernière peut se simuler par un programme VASS_{ord} par ce qui précède. En ajoutant le vecteur $(0, 0, \vec{e}_1 - \vec{e}_{halt})$ de façon à ce que l'(unique) instruction numéro $halt$ qui contient $Halt$ fasse reboucler sur la première instruction, on obtiendra exactement ce qui est désiré. □

En réalité, il n'y a pas que les nombres premiers que l'on peut programmer en VASS_{ord} . On peut programmer en VASS_{ord} toutes les fonctions calculables (et exactement elles) :

Question 13. *Démontrer qu'il existe un programme VASS_{ord} universel : il existe une liste de vecteurs V , telle que pour toute fonction $f : \mathbb{N} \rightarrow \mathbb{N}$ partielle calculable, il existe un entier C tel que, pour tout entier n , le programme F appliqué sur le nombre $(C, 2^{2^n}, 0, \dots, 0)$ retourne le résultat $(2^{2^{f(n)}}, 0, \dots, 0)$ si et seulement si f est définie en n .*

2. En réalité, le modèle de Conway parle de fractions, et il parle des 14 fractions fantastiques.

Solution : On a démontré en cours qu'il existait une machine de Turing universelle M . Dans le cours on utilise un codage par les mots, mais quitte à voir les entiers comme leur codage en binaire, et un mot sur l'alphabet $\{0, 1\}$ comme l'écriture d'un entier, on peut aussi considérer que cette machine universelle fonctionne sur des entiers plutôt que sur des mots : on fixe un codage des machines de Turing par les entiers, et cette machine de Turing universelle M est telle que si on lui donne c et n , elle simule la machine de Turing qui correspond à c sur n [Ce paragraphe est un peu une remarque pour les plus puristes, car on a fait cela en PCs parfois implicitement].

On en déduit que la fonction f qui à 2^n et c associe 2^r où r est le résultat du calcul de la machine de Turing qui correspond à c sur l'entrée n est partiellement calculable : il suffit de vérifier que l'entrée est de la forme 2^n puis d'utiliser cette machine de Turing universelle pour simuler la machine de Turing qui correspond à c sur n , et si cette simulation termine avec le résultat r , retourner 2^r .

Le programme $VASS_{ord}$ qui lui correspond par ce qui précède (questions 9, 11) est exactement ce que l'on cherche : l'entier C correspond à 3^c , où c est le codage de la machine de Turing qui calcule f . \square

Question 14. *Peut-on énumérer (par un algorithme) les entiers C pour lesquels le programme $VASS_{ord}$ universel précédent termine pour tout n ?*

Solution : Cela n'est pas possible. En effet, par contradiction, si cela était possible, étant donné un entier n , on pourrait considérer le même entier C_n , et considérer le résultat r_n du programme $VASS_{ord}$ sur n .

La fonction qui à n associe $r_n + 1$ serait donc une fonction totale calculable. Elle devrait donc correspondre à un entier C_p pour un entier p . Mais par construction, elle ne vaut pas r_p pour l'argument p (car elle vaut $r_p + 1$). Contradiction. \square

La conjecture de Goldbach dit que tout entier pair strictement plus grand que 2 est la somme de deux entiers premiers. C'est l'un des exemples de conjecture parmi les plus célèbres pour laquelle on ne connaît aucune preuve (ni aucun contre-exemple).

On note \mathcal{V}^+ pour l'ensemble des programmes $VASS_{ord} V$, dont les coefficients des vecteurs sont soit $-1, 0$ ou 1 , et qui terminent en partant de $(1, 0, \dots, 0)$.

Question 15. *Etant donné un entier m , on note $T(m)$ pour le plus grand nombre d'étapes qu'une liste $V \in \mathcal{V}^+$ de au plus m vecteurs³ de \mathbb{Z}^m effectue avant de terminer en partant de $(1, 0, \dots, 0)$.*

Expliquer pourquoi $T(m)$ est bien définie pour toute valeur m .

Expliquer pourquoi vous savez explicitement produire un entier m_0 pour lequel déterminer la valeur $T(m_0)$ est au moins aussi difficile que de déterminer si la conjecture de Goldbach est vraie.

Solution : Quand m est fixé, il est facile de construire au moins un tel programme $V \in \mathcal{V}^+$ qui termine en partant de $(1, 0, \dots, 0)$: choisir par exemple la liste réduite au vecteur $(0, -1, 0, \dots, 0)$. Par ailleurs, il y a un nombre fini de vecteurs à coefficients dans $\{-1, 0, 1\}$. Par conséquent, $\Sigma(m)$ est donc le maximum d'un ensemble fini et non vide de valeurs, et donc est parfaitement défini pour tout entier m .

On peut construire une machine de Turing M qui termine si et seulement si la conjecture de Goldbach est fautive : construire une machine qui teste pour tous les entiers $n \geq 4$ si l'on peut trouver deux entiers inférieurs premiers tels que n est leur somme.

Elle possède n_0 états pour un certain entier n_0 . Cette machine peut se simuler par certain programme $VASS_{ord} V$ qui terminera si et seulement si la machine de Turing termine. Les simulations précédentes n'utilisent que des vecteurs avec des coefficients dans $\{-1, 0, 1\}$.

Si on note m_0 le maximum de son nombre de vecteurs (que l'on peut calculer à partir de n_0), et de leur dimension, cela signifie que connaître $T(m_0)$ est au moins aussi difficile que de résoudre la conjecture de Goldbach.

3. On peut donc aussi dire : de $\{-1, 0, 1\}^m$ au lieu de \mathbb{Z}^m .

En effet, si on le connaissait, disons que l'on saurait qu'il vaut n_0 , pour décider si V termine, il suffirait de simuler V pendant n_0 transitions, et d'accepter si et seulement si V a accepté en moins de n_0 étapes.

[Note culturelle : En fait, plus généralement, la connaissance de $T(m)$ permettrait de connaître la réponse à de nombreuses conjectures mathématiques, et pas seulement de la conjecture de Goldbach. Et cet argument n'est pas spécifique aux programmes $VASS_{ord}$ (mais est lié à ce qu'on appelle les fonctions de castor-affairé] \square

Fondements de l'informatique. Examen

Durée: 3h

Sujet proposé par Olivier Bournez

Version 4

(corrigé)

Les 4 parties sont indépendantes, et peuvent être traitées dans un ordre quelconque. On pourra admettre le résultat d'une question pour passer aux questions suivantes. On pourra utiliser tous les résultats et les théorèmes démontrés ou énoncés en cours ou en petite classe ou dans le polycopié sans chercher à les redémontrer.

Il est possible d'avoir la note maximale sans répondre à toutes les questions. La difficulté des questions n'est pas une fonction linéaire ni croissante de leur numérotation.

La qualité de votre argumentation et de votre rédaction est une partie importante de votre évaluation.

Tous les graphes considérés sont non-orientés.

1 Machines de Turing

On fixe un alphabet fini Σ . Σ^* désigne l'ensemble des mots sur l'alphabet Σ .

Question 1. *Les problèmes suivants sont-ils décidables ? Justifier.*

- *Déterminer si le langage accepté par une machine de Turing M est co-fini : on dit qu'un langage est co-fini si son complémentaire (dans Σ^*) est fini.*
- *Déterminer si une machine de Turing accepte son entrée en utilisant au plus 2^{2^n} cases du ruban, où n est la longueur du mot en entrée.*
- *On fixe une fonction calculable $g : \mathbb{N} \rightarrow \mathbb{N}$. Déterminer si une machine de Turing accepte son entrée en utilisant au plus $2^{g(n)}$ cases du ruban, où n est la longueur du mot en entrée. Même question si l'on suppose que $g : \mathbb{N} \rightarrow \mathbb{N}$ est quelconque (c'est-à-dire, possiblement non-calculable).*

Solution : Le premier problème est indécidable par une application directe du théorème de Rice : le problème est non-trivial car il y a bien des langages finis et co-finis récursivement énumérables. Par exemple, le langage $L = \{0\}$ et son complémentaire.

Le second problème est un cas particulier du troisième pour $g(n) = 2^n$ qui est calculable.

Le troisième problème est décidable, quand g est calculable car il suffit de simuler la machine en la restreignant à au plus $2^{g(n)}$ cases du ruban pour décider la propriété (si jamais la simulation réalise que l'on repasse par la même configuration, alors on sait que la machine que l'on simule ne termine pas, et donc n'accepte pas son entrée).

Si g est possiblement non-calculable, le problème peut être indécidable. Considérer par exemple la fonction $g : \mathbb{N} \rightarrow \mathbb{N}$ avec $g(n)$ qui vaut 2 ou 1 suivant si la machine de Turing numéro n termine ou non sur l'entrée vide, et une machine de Turing M qui à l'étape 1 déplace sa tête de lecture d'une case vers la droite, puis à l'étape 2 déplace sa tête de lecture d'une case vers la droite et accepte. M accepte 1^n en au plus $g(n)$ étapes si et seulement si la machine de Turing numéro n termine. Décider le problème reviendrait à savoir décider si une machine de Turing donnée s'arrête. \square

Une machine de Turing avec ruban semi-infini est une machine de Turing dont le programme est tel que la tête de lecture ne va jamais à gauche de sa position initiale.

Question 2. Montrer que le problème de savoir si une machine de Turing semi-infinie accepte un mot w est indécidable.

Solution : Le problème de l'arrêt des machines de Turing se réduit à ce problème : on transforme une machine de Turing M en une machine M' qui la simule de la façon suivante : à chaque fois que le ruban de M contient w , le ruban de M' contient $\#w$ mais avec $\#$ en face de la position initiale de la tête de lecture de M : M' commence par remplacer l'entrée w par $\#w$ en décalant w d'une case vers la droite, et en insérant un symbole $\#$ tout à gauche. A chaque fois que la tête de lecture de M souhaite aller à gauche et voudrait écrire un symbole a sur le symbole $\#$, M' décale le contenu w du ruban d'une case sur la droite et écrit a juste à droite de $\#$ pour que le ruban devienne $\#aw$. En faisant ainsi, M' simule M , et est une machine avec ruban semi-infini. M accepte w si et seulement si M' accepte w . M' s'obtient de façon calculable à partir de M . \square

2 Espaces vectoriels

Soit E un \mathbb{R} -espace vectoriel. On dit que E est *ordonnable* s'il existe une relation d'ordre total \leq sur E qui est compatible avec la structure d'espace vectoriel, c'est-à-dire :

- pour tout $x, x', y, y' \in E$, si $x \leq x'$ et $y \leq y'$ alors $x + y \leq x' + y'$;
- pour tout $x \in E$ et pour tout $\lambda \in \mathbb{R}$, si $x \geq 0$ et $\lambda \geq 0$ alors $\lambda x \geq 0$.

Question 3. Construire un ensemble \mathcal{F}_E de formules du calcul **propositionnel** sur une signature que vous préciserez tel que \mathcal{F}_E est satisfiable si et seulement si E est ordonnable.

Solution : A chaque couple $x, y \in E$, on associe une variable propositionnelle $X_{x,y}$. Soit $\mathcal{F}_E = \{R_x : x \in E\} \cup \{S_{x,y} : x \neq y \in E\} \cup \{T_{x,y,z} : x, y, z \in E\} \cup \{U_{x,y,x',y'} : x, y, x', y' \in E\} \cup \{V_{x,\lambda} : x \in E \text{ et } \lambda \in \mathbb{R}^+\}$ où

$$R_x : X_{x,x},$$

$$S_{x,y} : \neg(X_{x,y} \wedge X_{y,x})$$

$$T_{x,y,z} : X_{x,y} \wedge Y_{y,z} \Rightarrow X_{x,z}$$

$$U_{x,y,x',y'} : X_{x,x'} \wedge Y_{y,y'} \Rightarrow X_{x+y,x'+y'}$$

et

$$V_{x,\lambda} : X_{x,0} \Rightarrow X_{\lambda x,0}$$

On vérifie que E est ordonnable si et seulement si \mathcal{F}_E est satisfiable. \square

Question 4. Montrer qu'un \mathbb{R} -espace vectoriel E est ordonnable si et seulement si tous ses sous-espaces vectoriels de dimension finie¹ le sont.

Solution : Il est évident que si E est ordonnable alors tous ses sous-espaces de dimension finie le sont.

Réciproquement : Supposons que tous les sous-espaces de dimension finie de E sont ordonnables et montrons que \mathcal{F}_E est satisfiable. Soit G un sous-ensemble fini de \mathcal{F}_E . Soit $B = \{x \in E : \exists y \in E \text{ tel que } X_{x,y} \text{ ou } X_{y,x} \text{ figure dans l'une des formules de } G\}$. B est un sous-ensemble fini de E . Soit E' le sous-espace vectoriel engendré par B . E' est ordonnable, donc $\mathcal{F}_{E'}$ est satisfiable. Comme $G \subset \mathcal{F}_{E'}$, G est aussi satisfiable. Par le théorème de compacité, \mathcal{F}_E est satisfiable. \square

1. On rappelle qu'un sous-espace vectoriel de dimension finie de E est le plus petit sous-espace vectoriel engendré par (i.e. qui contient) un nombre fini d'éléments de E .

3 Chemins avec paires interdites

Etant donné un graphe $G = (V, E)$, et un ensemble S de paires de sommets $(u_1, v_1), (u_2, v_2), \dots, (u_k, v_k) \in V \times V$, on dit qu'un chemin de G évite les paires de S s'il visite au plus un sommet de chaque paire de S .

Question 5. On considère le graphe G dont les sommets sont $V = \{s, t, u, v\}$ et les arêtes sont $E = \{(s, u), (s, v), (u, t), (v, t)\}$. Quels sont les chemins entre s et t qui évitent $S = \{(u, v)\}$?

Solution : Un tel chemin visite les sommets s, u, t ou s, v, t . □

Question 6. Construire un graphe à 5 sommets $V = \{s, t, u_1, u_2, u_3\}$ et un ensemble de paires S tels que :

- les chemins entre s et t qui évitent S passent nécessairement par l'un des trois sommets u_1, u_2 , ou u_3 ;
- pour chaque sommet u_i pour $i \in \{1, 2, 3\}$, il y a au moins un chemin entre s et t qui évite S et passe par u_i .

Solution : On considère $E = \{(s, u_1), (u_1, t), (s, u_2), (u_2, t), (s, u_3), (u_3, t)\}$ et $S = \emptyset$. □

Question 7. Démontrer que le problème suivant est NP-complet :

- Données : un graphe $G = (V, E)$, deux sommets $s, t \in V$, et un ensemble S de paires $(u_1, v_1), (u_2, v_2), \dots, (u_k, v_k) \in V \times V$.
 - Question : Existe-t-il un chemin entre s et t dans le graphe G qui évite les paires de S .
- On pourra utiliser la NP-complétude du problème 3-SAT.

Solution : Le problème est dans NP, car la donnée d'un chemin constitue un certificat vérifiable en temps polynomial.

Pour montrer la complétude, on effectue une réduction à partir du problème 3-SAT que l'on sait NP-complet.

La question 5 fournit un graphe $G_1(s, u, v, t)$ qui permet de coder le fait que l'on a soit x_i soit $\neg x_i$ pour une variable propositionnelle.

La question 6 fournit un graphe $G_2(s, u_1, u_2, u_3, t)$ qui permet de coder une clause.

En concaténant chacun des graphes, on obtient la réduction.

Plus précisément, étant donnée une conjonction de clauses $C = C_1 \wedge C_2 \cdots \wedge C_\ell$ sur les variables propositionnelles x_1, x_2, \dots, x_n , avec $C_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$, où chaque $l_{i,j}$ est une variable propositionnelle ou sa négation, on construit le graphe qui contient les graphes $G_1(s, x_1, \neg x_1, t_1)$, $G_1(t_1, x_2, \neg x_2, t_2), \dots, G_1(t_{n-1}, x_n, \neg x_n, t_n)$, $G_2(t_n, l_{1,1}, l_{1,2}, l_{1,3}, t_{n+1})$, $G_2(t_{n+1}, l_{2,1}, l_{2,2}, l_{2,3}, t_{n+2})$, $\dots, G_2(t_{n+\ell-1}, l_{\ell,1}, l_{\ell,2}, l_{\ell,3}, t)$.

On prend l'ensemble S comme l'union de toutes les paires où figure une variable et sa négation : pour le dire plus formellement, S est constitué de toutes les paires $(x_i, \neg x_i)$, toutes les paires $(x_i, l_{i',j})$ où $l_{i',j} = \neg x_i$, et de toutes les paires $(\neg x_i, l_{i',j})$ où $l_{i',j} = x_i$.

Un chemin entre s et t qui évite S par construction satisfait C . Réciproquement, en chemin entre s et t qui évite S doit passer par chacun des x_i ou son complémentaire, et va indiquer comment satisfaire C selon les clauses traversées. □

4 Jouons aux dominos

Le problème de la correspondance de Post (PCP) est un problème de décision sur des *dominos* qui fut introduit par Emil Post en 1946. Il apparaît souvent dans des démonstrations d'indécidabilité : nous verrons un exemple plus bas concernant un problème sur les matrices.

Nous allons considérer plusieurs variantes de ce problème, et étudier leur difficulté.

On fixe un alphabet Σ .

On appelle "*domino*" un couple (U, V) où U et V sont des mots sur un alphabet Σ : on va représenter un tel couple graphiquement sous la forme $\begin{array}{|c|} \hline U \\ \hline V \\ \hline \end{array}$.

Etant donné une suite finie S de dominos, le problème est de savoir si l'on peut poser ces dominos l'un à la suite de l'autre (dans n'importe quel ordre) de telle sorte que le mot qui apparaît en haut soit le même que le mot qui apparaît en bas : on appelle cela une *correspondance*.

On souhaite savoir s'il existe un algorithme qui, étant donnée une suite finie S de dominos, décide si il existe une correspondance pour cette suite : on appelle cela le problème de correspondance de Post.

4.1 Sans répétition

Supposons que l'on a le droit d'utiliser chaque domino de S au plus une fois².

Par exemple, si l'on part de l'ensemble

$$S = \left\{ \begin{array}{|c|} \hline ba \\ \hline ac \\ \hline \end{array}, \begin{array}{|c|} \hline a \\ \hline ab \\ \hline \end{array}, \begin{array}{|c|} \hline ca \\ \hline a \\ \hline \end{array} \right\}$$

on peut construire la correspondance suivante :

$$\begin{array}{|c|} \hline a \\ \hline ab \\ \hline \end{array} \begin{array}{|c|} \hline ba \\ \hline ac \\ \hline \end{array} \begin{array}{|c|} \hline ca \\ \hline a \\ \hline \end{array} :$$

on a bien en haut et en bas le même mot, à savoir *abaca*.

Si cela aide de le dire très formellement : étant donné une suite de dominos³

$$S = \left\{ \begin{array}{|c|} \hline U_1 \\ \hline V_1 \\ \hline \end{array}, \begin{array}{|c|} \hline U_2 \\ \hline V_2 \\ \hline \end{array}, \dots, \begin{array}{|c|} \hline U_p \\ \hline V_p \\ \hline \end{array} \right\}, \quad (1)$$

le problème de la correspondance de Post **sans répétition** consiste à déterminer s'il existe une suite non vide d'indices i_1, i_2, \dots, i_k dans $\{1, 2, \dots, p\}$ telle que

$$U_{i_1} U_{i_2} \dots U_{i_k} = V_{i_1} V_{i_2} \dots V_{i_k}.$$

$$\left(\text{graphiquement : } \begin{array}{|c|} \hline U_{i_1} \\ \hline V_{i_1} \\ \hline \end{array} \begin{array}{|c|} \hline U_{i_2} \\ \hline V_{i_2} \\ \hline \end{array} \begin{array}{|c|} \hline U_{i_3} \\ \hline V_{i_3} \\ \hline \end{array} \dots \begin{array}{|c|} \hline U_{i_k} \\ \hline V_{i_k} \\ \hline \end{array} \right)$$

avec $i_k \neq i_l$ pour $k \neq l$.

Question 8. Montrer que le problème de la correspondance de Post sans répétition est décidable.

Solution : Etant donné S , il y a un nombre finie de possibilités de suites injectives dans $\{1, 2, \dots, p\}$. Il suffit de tester pour chacune si cela satisfait

$$\begin{array}{|c|} \hline U_{i_1} \\ \hline V_{i_1} \\ \hline \end{array} \begin{array}{|c|} \hline U_{i_2} \\ \hline V_{i_2} \\ \hline \end{array} \begin{array}{|c|} \hline U_{i_3} \\ \hline V_{i_3} \\ \hline \end{array} \dots \begin{array}{|c|} \hline U_{i_k} \\ \hline V_{i_k} \\ \hline \end{array}.$$

□

On établira dans la section 4.6 qu'il est *NP*-complet.

2. On n'interdit pas cependant qu'un même domino puisse avoir plusieurs occurrences dans S .

3. Pour les plus puristes, et en lien avec la note de bas de page 2. (page précédente) : la notation de la suite S dans l'écriture sous forme ensembliste (1) est possiblement la notation d'un multi-ensemble plutôt qu'un ensemble : par rapport à un ensemble, on autorise un même élément à apparaître deux fois ou plus dans un multi-ensemble.

4.2 Avec répétitions

Jusqu'à la section 4.6, on s'autorise à utiliser plusieurs fois un même domino de la suite S . Par exemple, si l'on part de la suite

$$S_1 = \left\{ \begin{array}{|c|} \hline b \\ \hline ca \\ \hline \end{array}, \begin{array}{|c|} \hline a \\ \hline ab \\ \hline \end{array}, \begin{array}{|c|} \hline ca \\ \hline a \\ \hline \end{array}, \begin{array}{|c|} \hline abc \\ \hline c \\ \hline \end{array} \right\}$$

on peut construire la correspondance suivante :

$$\begin{array}{|c|} \hline a \\ \hline ab \\ \hline \end{array} \begin{array}{|c|} \hline b \\ \hline ca \\ \hline \end{array} \begin{array}{|c|} \hline ca \\ \hline a \\ \hline \end{array} \begin{array}{|c|} \hline a \\ \hline ab \\ \hline \end{array} \begin{array}{|c|} \hline abc \\ \hline c \\ \hline \end{array} :$$

on a bien en haut et en bas le même mot, à savoir $abcaabc$ (et on a utilisé plusieurs fois un même domino).

Tous les ensembles S ne possèdent pas de correspondance :

Question 9. On part cette fois de la suite

$$S_2 = \left\{ \begin{array}{|c|} \hline abc \\ \hline ab \\ \hline \end{array}, \begin{array}{|c|} \hline ca \\ \hline a \\ \hline \end{array}, \begin{array}{|c|} \hline acc \\ \hline ba \\ \hline \end{array} \right\}$$

Est-il possible d'obtenir une correspondance ?

Solution : C'est impossible, car le mot en haut possède nécessairement plus de lettres strictement que le mot en bas. \square

Si cela aide de le dire formellement : étant donné un ensemble de dominos

$$S = \left\{ \begin{array}{|c|} \hline U_1 \\ \hline V_1 \\ \hline \end{array}, \begin{array}{|c|} \hline U_2 \\ \hline V_2 \\ \hline \end{array}, \dots, \begin{array}{|c|} \hline U_p \\ \hline V_p \\ \hline \end{array} \right\},$$

le problème de la correspondance de Post (avec répétitions) consiste à déterminer s'il existe une suite non vide d'indices i_1, i_2, \dots, i_k dans $\{1, 2, \dots, p\}$ telle que

$$U_{i_1} U_{i_2} \dots U_{i_k} = V_{i_1} V_{i_2} \dots V_{i_k}$$

$$\left(\text{graphiquement : } \begin{array}{|c|} \hline U_{i_1} \\ \hline V_{i_1} \\ \hline \end{array} \begin{array}{|c|} \hline U_{i_2} \\ \hline V_{i_2} \\ \hline \end{array} \begin{array}{|c|} \hline U_{i_3} \\ \hline V_{i_3} \\ \hline \end{array} \dots \begin{array}{|c|} \hline U_{i_k} \\ \hline V_{i_k} \\ \hline \end{array} \right).$$

4.3 Réduction à la variante modifiée

On s'intéresse à la variante suivante, que l'on appelle problème de correspondance de Post **modifié** : on impose que $i_1 = 1$, c'est-à-dire, on impose le premier domino.

Si cela aide de le dire très formellement : étant donné un ensemble de dominos

$$S = \left\{ \begin{array}{|c|} \hline U_1 \\ \hline V_1 \\ \hline \end{array}, \begin{array}{|c|} \hline U_2 \\ \hline V_2 \\ \hline \end{array}, \dots, \begin{array}{|c|} \hline U_p \\ \hline V_p \\ \hline \end{array} \right\},$$

le problème de la correspondance de Post **modifié** consiste à déterminer s'il existe une suite non vide d'indices i_2, i_3, \dots, i_k dans $\{1, 2, \dots, p\}$ telle que

$$U_1 U_{i_2} \dots U_{i_k} = V_1 V_{i_2} \dots V_{i_k}.$$

$$\left(\text{graphiquement : } \begin{array}{|c|} \hline U_1 \\ \hline V_1 \\ \hline \end{array} \begin{array}{|c|} \hline U_{i_2} \\ \hline V_{i_2} \\ \hline \end{array} \begin{array}{|c|} \hline U_{i_3} \\ \hline V_{i_3} \\ \hline \end{array} \dots \begin{array}{|c|} \hline U_{i_k} \\ \hline V_{i_k} \\ \hline \end{array} \right).$$

Soit $u = u_1 u_2 \dots u_n$ un mot de longueur n sur l'alphabet Σ . Soient $*$ et \diamond deux nouvelles lettres : $*$ $\notin \Sigma$, $\diamond \notin \Sigma$.

On définit $*u$, $u*$ et $*u*$ comme les mots suivants :

$$*u = *u_1 * u_2 * u_3 \dots * u_n$$

$$u* = u_1 * u_2 * u_3 \dots * u_n$$

$$*u* = *u_1 * u_2 * u_3 \dots * u_n$$

En d'autres termes, $*u$ (respectivement : $u*$, $*u*$) ajoute le symbole $*$ avant (resp. après, avant et après) chaque lettre du mot u .

Question 10. *On considère*

$$S_1^* = \left\{ \begin{array}{|c|} \hline *a \\ \hline *a * b* \\ \hline \end{array}, \begin{array}{|c|} \hline *b \\ \hline c * a* \\ \hline \end{array}, \begin{array}{|c|} \hline *a \\ \hline a * b* \\ \hline \end{array}, \begin{array}{|c|} \hline *c * a \\ \hline a* \\ \hline \end{array}, \begin{array}{|c|} \hline *a * b * c \\ \hline c* \\ \hline \end{array}, \begin{array}{|c|} \hline *\diamond \\ \hline \diamond \\ \hline \end{array} \right\}$$

(Observer comment S_1^* s'obtient à partir de S_1 en début de la section 4.2 en appliquant les opérations ci-dessus en haut et en bas sur les dominos).

Expliquer comment les correspondances⁴ de S_1^* s'obtiennent à partir de celles de S_1 et réciproquement.

Solution : Une correspondance de S_1^* débute nécessairement par le premier domino en raison de la lettre $*$: tous les dominos ont ce symbole tout à gauche en haut, et c'est le seul avec ce symbole tout à gauche en bas. De façon symétrique, une correspondance se termine par le symbole \diamond . En effet tous les dominos ont en bas comme dernier symbole $*$, et le dernier domino d'une correspondance ne peut être que $\begin{array}{|c|} \hline *\diamond \\ \hline \diamond \\ \hline \end{array}$.

Ensuite, chaque domino impose que tous les symboles en position paire (sauf le dernier \diamond) correspondent au symbole $*$. Si on efface les symboles $*$ et \diamond dans la correspondance, c'est nécessairement une correspondance pour S_1 .

Réciproquement, toute correspondance pour S_1 s'étend en une correspondance pour S_1^* en intercalant un symbole $*$ entre chaque lettre et en ajoutant le symbole \diamond à la fin. \square

Question 11. *Démontrer que le problème de Post modifié se réduit au problème de Post.*

Solution : On transforme une instance du problème de Post modifié en une instance du problème de Post comme dans la question précédente.

Concrètement pour chaque ensemble de dominos

$$S = \left\{ \begin{array}{|c|} \hline t_1 \\ \hline b_1 \\ \hline \end{array}, \begin{array}{|c|} \hline t_1 \\ \hline b_1 \\ \hline \end{array}, \begin{array}{|c|} \hline t_2 \\ \hline b_2 \\ \hline \end{array}, \begin{array}{|c|} \hline t_3 \\ \hline b_3 \\ \hline \end{array}, \dots, \begin{array}{|c|} \hline *\diamond \\ \hline \diamond \\ \hline \end{array} \right\},$$

on peut considérer

$$S^* = \left\{ \begin{array}{|c|} \hline *t_1 \\ \hline *b_1* \\ \hline \end{array}, \begin{array}{|c|} \hline *t_1 \\ \hline b_1* \\ \hline \end{array}, \begin{array}{|c|} \hline *t_2 \\ \hline b_2* \\ \hline \end{array}, \begin{array}{|c|} \hline *t_3 \\ \hline b_3* \\ \hline \end{array}, \dots, \begin{array}{|c|} \hline *\diamond \\ \hline \diamond \\ \hline \end{array} \right\}.$$

Par le raisonnement précédent, les correspondances de S sont en bijection avec celles de S^* en intercalant un symbole $*$ entre chaque lettre et en ajoutant le symbole \diamond à la fin.

Une correspondance de S^* commence nécessairement par son premier domino $\begin{array}{|c|} \hline *t_1 \\ \hline *b_1* \\ \hline \end{array}$.

Cela donne donc bien une réduction du problème de correspondance de Post modifié vers le problème de correspondance de Post (la transformation de S en S^* est bien calculable). \square

4. On parle bien des correspondances "normales", i.e. au sens des questions précédentes, pas nécessairement "modifiées" comme dans le texte qui précède.

4.4 Indécidabilité de la variante modifiée

Question 12. Démontrer que le problème de la correspondance de POST modifié est indécidable.

Etant donné une machine de Turing M semi-infinie et un mot w on cherchera à construire un ensemble S_M de dominos tel qu'une correspondance de S_M décrit un calcul de M sur l'entrée w . On pourra inclure dans S_M les dominos

#	a	#	aq_a	$q_a a$	$q_a \#\#$
$\#q_0 w \#$	a	$B \#$	q_a	q_a	#

et d'autres dominos bien choisis que l'on décrira : w est un mot, $\#, a$ une lettre, B le symbole de blanc, q_0 et q_a l'état initial et acceptant de M .

Solution : On utilise les notations du cours pour décrire la machine de Turing M . q_a désigne l'état d'acceptation. On peut supposer sans perte de généralité que la machine de Turing déplace toujours sa tête de lecture à chaque étape.

Pour tout $a, a' \in \Gamma$, et pour tout état $q, q' \in Q$ avec $q \neq q_r$, si $\delta(q, a) = (q', a', \rightarrow)$, on ajoute

qa
$a'q'$

à S_M .

Pour tout $a, a' \in \Gamma$, et pour tout état $q, q' \in Q$ avec $q \neq q_r$, si $\delta(q, a) = (q', a', \leftarrow)$, on ajoute

cqa
$q'ca'$

à S_M .

Le jeu de domino est tel que qu'une correspondance va décrire en bas (et donc aussi en haut) le diagramme espace temps du calcul de M sur w : une correspondance sera nécessairement de la forme $\#q_0 w \# C_1 \# C_2 \dots \# C_n \# \dots \# q_a \#\#$ où C_1, C_2, \dots, C_n code la suite des configurations du calcul de la machine M sur le mot w jusqu'à atteindre l'état accepteur q_a à l'étape n . Les dominos

aq_a	$q_a a$
q_a	q_a

permettent alors de "manger" les symboles en haut pour compléter et terminer avec

le domino

$q_a \#\#$
#

<il faut l'expliquer et le justifier le plus clairement possible pour obtenir la note maximale pour cette question, et cela nécessite des explications dont nous avons ici simplement donné l'idée>. \square

Par conséquent, le problème de la correspondance de Post est indécidable.

4.5 Application : Problème de matrices

Un ensemble H de matrices 3×3 à coefficients entiers est dit *mortel* si la matrice nulle peut s'obtenir comme un produit d'un nombre fini de matrices de cet ensemble (on s'autorise à utiliser plusieurs fois la même matrice de l'ensemble H dans le produit).

Le but de cette partie est de prouver qu'il est indécidable de déterminer si un ensemble H de matrices est mortel.

On considère un ensemble H de matrices 3×3 qui contient les matrices

$$S = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad T = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

ainsi que des matrices de la forme

$$W_j = \begin{bmatrix} p_j & 0 & 0 \\ 0 & r_j & 0 \\ q_j & s_j & 1 \end{bmatrix} \quad (2)$$

pour certains entiers p_j, q_j, r_j, s_j .

On note $[x, y, z]$ pour un vecteur ligne.

En observant que

$$[a, b, c]S = a[1, 0, 1]$$

$$[a, b, c]T = (a - b)[1, -1, 0]$$

et que toutes les matrices W_j sont inversibles, un raisonnement assez simple (sur les types de produits possibles avec un tel ensemble H et sur les images et noyaux de S et T) démontre le fait suivant que l'on admettra : *une condition nécessaire et suffisante pour obtenir un produit nul avec les matrices de l'ensemble H est qu'il existe un produit X des matrices W_j tel que*

$$[1, 0, 1]X = [h, h, 1]$$

pour un $h > 0$ (et dans ce cas $SXT = 0$).

Illustrons une relation entre la multiplication de matrices et la concaténation de mots par un exemple :

$$[123, 12, 1] \begin{bmatrix} 1000 & 0 & 0 \\ 0 & 100 & 0 \\ 223 & 32 & 1 \end{bmatrix} = [123223, 1232, 1]$$

Question 13. *Etant donnée une paire de mots $\langle U, V \rangle$, définir une matrice $W(U, V)$ de la forme W_j (i.e. de la forme (2)) telle si X, Y, U, V sont des mots sur l'alphabet $\{1, 2, 3\}$, alors*

$$[X, Y, 1]W(U, V) = [X.U, Y.V, 1]$$

où $.$ désigne la concaténation, et les mots sont interprétés comme des entiers de la façon évidente.

Solution : Considérer

$$W(U, V) = \begin{bmatrix} p & 0 & 0 \\ 0 & r & 0 \\ q & s & 1 \end{bmatrix}$$

où q, s sont les nombres obtenus en écrivant les mots U, V et p et r sont les entiers obtenus en considérant 1 suivi du nombre de zéros qui correspondent au nombre de lettres de U et V respectivement. \square

Question 14. *Démontrer qu'il est indécidable de savoir si un ensemble H de matrices est mortel.*

Solution : On utilise une réduction à partir du problème de la correspondance de Post que l'on a démontré dans les questions précédentes indécidable. On peut supposer sans perte de généralité que l'alphabet vérifie $\Sigma = \{2, 3\}$.

Soit $K = \left\{ \begin{bmatrix} U_1 \\ V_1 \end{bmatrix}, \begin{bmatrix} U_2 \\ V_2 \end{bmatrix}, \dots, \begin{bmatrix} U_n \\ V_n \end{bmatrix} \right\}$ un ensemble de dominos avec l'alphabet $\Sigma = \{2, 3\}$. On considère l'ensemble $H(K)$ de matrices

$$\{S, T, W(U_j, V_j), W(U_j, 1.V_j), \text{ pour } j = 1 \dots n\}.$$

$H(K)$ est mortel si et seulement si il y a un produit X de W de $H(K)$ tel que $[1, 0, 1]X = [h, h, 1]$ pour un certain h par la remarque plus haut (l'observation admise).

Un tel mot doit avoir comme première lettre 1, suivi d'un 2 et de 3 seulement, et donc un tel produit existe si et seulement si le problème de la correspondance de post admet une solution.

Le problème est dans NP car la donnée d'un produit constitue un certificat vérifiable en temps polynomial : observer que cela se vérifie bien en temps polynomial car la taille des nombres

impliqués dans le produit de matrice reste bien polynomial en la taille des nombres dans le produit.

Cela montre que le problème de la correspondance de post se réduit à ce problème : la réduction est bien calculable en temps polynomial. \square

4.6 Retour sur les versions sans répétition

On revient sur la variante du problème de la correspondance de Post **sans répétition**, considérée dans la section 4.1 (rappel : on peut bien avoir plusieurs fois un même domino dans la suite S , mais chaque domino ne peut être utilisé qu'une fois dans une correspondance).

Question 15. *Montrer que le problème de correspondance de Post sans répétition est NP-complet.*

Solution : Le problème est clairement dans NP car la donnée du choix du produit est un certificat vérifiable en temps polynomial : il suffit de vérifier que le produit est nul.

On peut démontrer le fait que le problème est NP -difficile directement (en revenant à la définition) de la façon suivante : étant donné un problème NP -complet, il est décidé par une machine de Turing M non-déterministe qui fonctionne en temps $p(n)$. On construit le jeu de dominos correspondant considéré dans la réduction dans la réponse de la question 12. Le nombre de dominos construits est une fonction de la taille de l'alphabet et du nombre d'états de la machine de Turing M . Le seul domino dont la taille peut être grande est le domino

#
q_0w

Le jeu de dominos nécessaire pour simuler $p(n)$ étapes de M est $f(p(n))$ où $f(n)$ est une fonction polynomiale qui exprime le nombre de dominos pour simuler n étapes de M .

On aura bien w accepté par M si et seulement si l'on a une instance positive au problème de correspondance de Post sans répétition.

(en fait, le jeu de dominos construit répète bien plusieurs fois certains dominos, mais on utilisera au plus un exemplaire de chacun de ces dominos). \square

Question 16. *On ne s'autorise pas les répétitions de matrices. Montrer qu'il est décidable de savoir si un ensemble H de matrices est mortel sans répétition. Montrer que le problème est NP-complet.*

Solution : Le problème est décidable, car il n'y a qu'un nombre fini de produits possibles de matrices sans répétition, et il suffit de tester pour chacun s'il est nul.

La réduction produite dans la réponse à la question 14 produit en fait aussi une réduction du problème de la correspondance de Post sans répétition à ce problème.

(la réduction est bien calculable en temps polynomial).

Par la question précédente, le problème est donc NP -complet. \square

Fondements de l'informatique. Examen

Durée: 3h

Sujet proposé par Olivier Bournez

Version 11

(corrigé)

Les 5 parties sont indépendantes, et peuvent être traitées dans un ordre quelconque. On pourra admettre le résultat d'une question pour passer aux questions suivantes. On pourra utiliser tous les résultats et les théorèmes démontrés ou énoncés en cours ou en petite classe ou dans le polycopié sans chercher à les redémontrer. Il est possible d'avoir la note maximale sans répondre à toutes les questions. La difficulté des questions n'est pas une fonction linéaire ni croissante de leur numérotation. La qualité de votre argumentation et de votre rédaction est une partie importante de votre évaluation.

1 Machines de Turing

On considère des machines de Turing qui travaillent sur l'alphabet qui contient les lettres $a, b, \dots, z, A, B, \dots, Z$ ainsi que le caractère espace.

Question 1. *Le problème suivant est-il décidable ? Déterminer si le langage accepté par une machine de Turing M est le langage réduit au mot "je suis une machine de Turing".*

Solution : C'est indécidable : il s'agit d'une application directe du théorème de Rice. □

Question 2. *Existe-t'il une machine de Turing M dont le codage est $\langle M \rangle$ qui accepte si une entrée w est le mot "je suis la machine de Turing $\langle M \rangle$ " et refuse sinon¹.*

Solution : En utilisant le théorème de récursion, on considère une machine de Turing qui commence par déterminer son propre code $\langle M \rangle$ puis teste si l'entrée est "je suis la machine de Turing $\langle M \rangle$ ". □

2 Ordres bien fondés

On rappelle qu'une relation d'ordre totale est une relation réflexive, transitive et antisymétrique, telle que tout élément est comparable à tout autre.

On dit qu'un ordre total est *bien fondé* s'il n'existe pas de suite infinie strictement décroissante. Par exemple, \leq est bien fondé sur l'ensemble des entiers \mathbb{N} , mais pas sur l'ensemble des entiers relatifs \mathbb{Z} .

Question 3. *On considère la signature avec les symboles de relation $=, \leq$ et aucun symbole de constante et de fonction.*

Démontrer qu'il n'existe aucune théorie sur cette signature dont les modèles égalitaires sont exactement les ensembles tels que l'interprétation de \leq est une relation d'ordre total bien fondé sur l'ensemble de base.

1. Bien entendu, "je suis la machine de Turing $\langle M \rangle$ " désigne la concaténation du mot "je suis la machine de Turing" et du mot correspondant à $\langle M \rangle$.

Solution : On utilise le théorème de compacité : Par l'absurde, supposons qu'il existe un tel ensemble \mathcal{F} . On considère l'ensemble de formules \mathcal{G} obtenu en prenant toutes les formules de \mathcal{F} auxquelles on ajoute les formules $c_{n+1} \leq c_n, \neg(c_n = c_{n+1})$ pour tout n , où ces symboles c_n sont de nouveaux symboles. Toute partie finie de \mathcal{G} est consistante, car il y a des relations totales bien fondées dans \mathbb{N}^* : en effet (\mathbb{N}, \leq) en est un modèle. Par le théorème de compacité, \mathcal{G} doit être consistante. On obtient une contradiction car l'interprétation des constantes c_n donne une suite infinie strictement décroissante et pourtant toutes les formules de \mathcal{F} sont satisfaites. \square

3 NP-complétude

Pour un graphe $G = (V, E)$, V désigne les sommets du graphe, et E ses arêtes. On suppose les graphes non-orientés.

On rappelle les définitions et faits suivants : un graphe est *connexe* s'il existe un chemin entre toute paire de sommets. C'est un *arbre* s'il ne possède aucun cycle. On pourra utiliser le fait suivants : un arbre avec $t + 1$ sommets possède nécessairement t arêtes. Réciproquement, un graphe connexe ayant $t + 1$ sommets et t arêtes est un arbre.

Etant donné un graphe G , on appellera *arbre de G* tout sous-graphe de G qui est un arbre. On dira qu'un tel arbre *passé par un sommet s* (de G) si ce sommet s est dans le sous-graphe.

L'objectif de cet algorithme est de prouver que le problème suivant est NP-complet.

ARBRE DE STEINER:

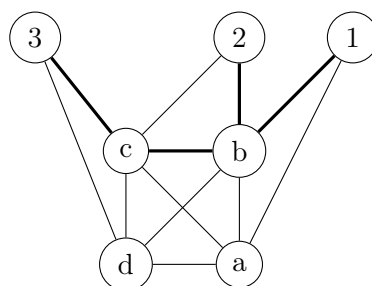
- **Donnée:** Un graphe non-orienté $G = (V, E)$, un ensemble D de sommets, un entier k .
- **Réponse:** Décider s'il existe un arbre de G qui a moins de k arêtes et qui passe par tous les sommets de D .

Soit $G = (V_G, E_G)$ un graphe. Nous allons construire un graphe $H = (V_H, E_H)$ à partir de G tel que

- $V_H = V_G \cup E_G$;
- $E_H = \{(v, u) | v, u \in V_G\} \cup \{(v, a) | v \in V_G, a \in E_G, v \text{ est extrémité de l'arête } a \text{ dans } G\}$.

Question 4. On considère le cas où $V_G = \{a, b, c, d\}$, et $E_G = \{(a, b), (b, c), (c, d)\}$. Dessiner le graphe H correspondant. Proposer un arbre de H qui possède 4 arêtes et qui passe par tous les sommets de H qui sont dans E_G .

Solution : Voici une solution. L'arbre est indiqué par ses arêtes en gras.



\square

Rappelons la définition suivante : une *couverture de sommets* S d'un graphe G est un sous-ensemble de sommets tel que toutes les arêtes de G ont au moins une de leurs extrémités dans S .

Question 5. Démontrer que si S est une couverture de sommets du graphe G , alors il existe un arbre de H qui passe par tous les sommets dans E_G et qui possède $|S| + |E_G| - 1$ arêtes.

Solution : Soit S une couverture de sommets du graphe G . Il faut construire un tel arbre.

Pour cela, nous ordonnons les sommets de S : $S = \{s_1, \dots, s_{|S|}\}$. Nous allons construire l'arbre $T = (V_T, E_T)$ de telle façon suivante :

- $V_T = S \cup E_G$;
- $\{(s_i, s_{i+1}) : 1 \leq i < s_{|S|}\} \subset E_T$;
- $(e, s_i) \in E_T$ si l'arête e de E_G a exactement une extrémité s_i dans S ou si e a deux extrémités s_j et s_i dans S avec $i < j$.

Comme tous les sommets de V_G sont reliés deux à deux dans H , il est clair que $\{(s_i, s_{i+1}) : 1 \leq i < s_{|S|}\}$ forme un chemin dans H . Comme S est une couverture de sommets, pour toute arête e , il existe au moins une extrémité dans S . Donc dans H , le sommet e est voisin d'au moins un sommet de S . T est connexe, et possède $|S| + |E_G| - 1$ arêtes avec $|S| + |E_G|$ sommets : c'est donc un arbre. \square

Question 6. Montrer que si $T = (V_T, E_T)$ est un arbre qui passe par tous les sommets dans E_G et qui possède t arêtes, alors il existe une couverture de sommets S du graphe G de cardinal² $t - |E_G| + 1$.

Solution : Considérons l'ensemble de sommets $C = V_T \setminus E_G$. Par définition de H , $C \subset V_G$. Toute arête $e = (u, v)$ de G est un sommet de H , et il est uniquement voisin des sommets u et v . Donc au moins l'un d'entre eux doit être dans C (sinon T ne serait pas connexe). Donc l'ensemble C a pour voisin toutes les arêtes de E , et il constitue une couverture de sommets de G .

Maintenant calculons son nombre d'éléments : $|C| = |V_T| - |E_G|$. Comme T est un arbre, il a $t = |V_T| - 1$ arêtes. On peut conclure que $|C| = t - |E_G| + 1$. \square

On rappelle que le problème RECOUVREMENT DE SOMMETS:

— **Donnée:** Un graphe $G = (V_G, E_G)$ non-orienté et un entier k .

— **Réponse:** Décider s'il existe une couverture S de G avec le cardinal de S qui vaut k .
est NP-complet.

Question 7. Montrer que le problème ARBRE DE STEINER est NP-complet.

Solution : Le problème ARBRE DE STEINER est dans NP car étant donnée un ensemble d'arêtes, on peut vérifier en temps polynomial

1. si cet ensemble d'arêtes forme un arbre ;
2. si l'ensemble de sommets D est couvert par l'arbre ;
3. si le nombre d'arêtes est inférieur à k .

Montrons maintenant qu'il est complet. Soit $\mathcal{I} = \langle G = (V_G, E_G), k \rangle$ une instance du problème RECOUVREMENT DE SOMMETS que l'on sait NP-complet.

Maintenant, nous allons transformer cette instance en une instance \mathcal{I}' du problème ARBRE DE STEINER $\langle H = (V_H, E_H), D, k' \rangle$ de la façon suivante :

Soit G un graphe $G = (V_G, E_G)$. Nous allons construire un graphe $H = (V_H, E_H)$ à partir de G tel que

- $V_H = V_G \cup E_G$;
- $E_H = \{(v, u) | v, u \in V_G\} \cup \{(v, a) | v \in V_G, a \in E_G, v \text{ est extrémité de l'arête } a \text{ dans } G\}$
- $D = E_G$
- $k' = k + |D| - 1$

Cette transformation peut se faire en temps polynomial : le nombre de sommets du nouveau graphe est $|V_G| + |E_G|$. Construire le graphe H nécessite au plus $\mathcal{O}((|V_G| + |E_G|)^2)$ opérations.

Les questions précédentes ont permis de prouver que le fait suivant : le graphe G admet une couverture de sommets S de taille k si et seulement si le graphe H admet un arbre ayant moins de k' arêtes passant par tous les sommets de D .

2. Le cardinal d'un ensemble est le nombre de ses éléments.

Donc le problème est NP-complet. \square

4 Le corps des réels calculables

Les définitions suivantes ne font rien que de formaliser ce à quoi on s'attend : \mathbb{Q} désigne l'ensemble des rationnels, et \mathbb{Q}_*^+ l'ensemble des rationnels strictement positifs. On suppose fixé un codage des rationnels tel que les opérations habituelles³ sont calculables sur ce codage⁴. On dira qu'une fonction $f : \mathbb{Q}_*^+ \rightarrow \mathbb{Q}$ est *calculable* si elle est calculable avec ce codage (et définie sur tout \mathbb{Q}_*^+). De même un ensemble de rationnels est dit *décidable* si l'ensemble des codages des rationnels de cet ensemble l'est. Un ensemble de rationnels est dit *récurivement énumérable* si l'ensemble des codages des rationnels de cet ensemble l'est.

Un nombre réel α est dit *calculable* si $\{q \in \mathbb{Q} | q < \alpha\}$ est décidable.

On dit qu'un réel α est *énumérable par le bas* si $\{q \in \mathbb{Q} | q < \alpha\}$ est récurivement énumérable.

On dit qu'un réel α est *énumérable par le haut* si $\{q \in \mathbb{Q} | q \geq \alpha\}$ est récurivement énumérable.

Question 8. Montrer que α est calculable si et seulement s'il est énumérable par le bas et énumérable par le haut.

Solution : Par la question précédente, un réel est calculable si et seulement si $\{q \in \mathbb{Q} | q < \alpha\}$ est décidable. On sait qu'un ensemble est décidable si et seulement s'il est récurivement énumérable et son complémentaire l'est aussi. \square

Question 9. Montrer qu'un réel α est énumérable par le bas si et seulement si c'est la borne supérieure d'une suite $u_0, u_1, \dots, u_i, \dots$ de rationnels telle que la fonction $i \mapsto u_i$ est calculable.

Solution : On sait que $\{q \in \mathbb{Q} | q < \alpha\}$ est récurivement énumérable. Il y a donc une machine de Turing qui énumère cet ensemble. α est alors la borne supérieure de cette suite.

Réciproquement, pour déterminer si $q < \alpha$, on parcourt la suite calculable de rationnels en acceptant si et seulement si l'on trouve un rationnel dans cette suite plus grand que q . \square

Un nombre réel α est dit *effectivement approximable* s'il existe une fonction calculable $a : \mathbb{Q}_*^+ \rightarrow \mathbb{Q}$ telle que pour tout rationnel $\epsilon > 0$, $a(\epsilon)$ fournit une approximation de α à ϵ -près : autrement dit : $|\alpha - a(\epsilon)| \leq \epsilon$ pour tout $\epsilon > 0$.

Question 10. Démontrer que si α est calculable alors il est effectivement approximable.

Solution : Etant donné $\epsilon > 0$, on parcourt les couples de rationnels jusqu'à trouver $p < q$ avec $p - q < \epsilon$ avec $p < \alpha$ et $\neg(q < \alpha)$. On retourne alors $a(\epsilon) = (p + q)/2$. \square

Question 11. Soit α un réel avec $\alpha \notin \mathbb{Q}$. Démontrer que si α est effectivement approximable, alors il est calculable.

Solution : Si $\alpha \notin \mathbb{Q}$, étant donné q , on teste pour $\epsilon = 1/n$ pour $n = 1, 2, \dots$ si $a(\epsilon) + \epsilon < q$, auquel cas on répond "vrai", jusqu'à ce que $a(\epsilon) - \epsilon > q$. Si ce dernier cas se produit, on répond "faux".

Nécessairement l'algorithme s'arrête car étant donné q , pour $1/n$ plus petit que $|\alpha - q|$ l'un des deux cas où l'on répond a dû se produire. \square

Question 12. En déduire que α est calculable si et seulement s'il est effectivement approximable.

3. Exemple : somme, produit, ...

4. Par exemple on peut coder le rationnel $r = p/q$ par le couple p, q où p et q sont écrits en binaire.

Solution : Il reste juste à démontrer que $\{q \in \mathbb{Q} | q < \alpha\}$ est décidable quand $\alpha \in \mathbb{Q}$. Mais ce cas est trivial. \square

Question 13. Montrer qu'un réel $\alpha \in [0, 1]$ est calculable⁵ si et seulement s'il possède une écriture⁶ en base 10 de la forme $0.u(1)u(2)\dots u(k)\dots$ avec $u : k \mapsto \{0, 1, \dots, 9\}$ calculable.

Solution : Si α est calculable, on peut calculer $u(1), u(2), \dots, u(k)$ pour k croissant, en testant si $0.u(1)u(2)\dots u(k-1)i \geq \alpha$ pour $i = 0, 1, \dots, 9$. Dès qu'un trouve $0.u(1)u(2)\dots u(k-1)i < \alpha$ on sait que $u(k)$ vaut $i - 1$, puisque $\alpha \in [0.u(1)u(2)\dots u(k-1)(i-1), 0.u(1)u(2)\dots u(k-1)i[$.

Réciproquement, étant donné $\epsilon > 0$, en déterminant k tel que $10^{-k} < \epsilon$, les k premiers chiffres de son écriture en base 10 fournissent une approximation à ϵ près de α . \square

L'ensemble des réels calculables est un sous-corps du corps des réels : en raison du fait que \mathbb{R} est un corps, il suffit de démontrer les faits suivants.

Question 14. Démontrer que la somme, le produit, la différence et le quotient de deux nombres calculables est calculable⁷.

Solution : Si $a(\epsilon)$ et $b(\epsilon)$ sont des fonctions calculables qui approximent p et q à ϵ près, alors $c(\epsilon) = a(\epsilon/2) + b(\epsilon/2)$ (respectivement : $a(\epsilon/2) - b(\epsilon/2)$) approximent $p + q$ (resp. $p - q$) à ϵ près. Elles sont bien calculables par composition de fonctions calculables.

Pour le produit

$$|pq - a(\epsilon)b(\epsilon)| \leq |p| \cdot |q - b(\epsilon)| + |b(\epsilon)||p - a(\epsilon)| \quad (1)$$

$$\leq a(1)\epsilon + b(1)\epsilon \quad (2)$$

$$= (a(1) + b(1))\epsilon \quad (3)$$

Donc $a(\frac{\epsilon}{a(1)+b(1)})b(\frac{\epsilon}{a(1)+b(1)})$ donne une approximation de pq à ϵ près.

On fait une majoration similaire pour le quotient. \square

C'est même un corps réel clos. En raison du fait que \mathbb{R} est un corps réel clos, il suffit de démontrer le fait suivant :

Question 15. Démontrer que toute racine α d'un polynôme à coefficients calculables est un réel calculable. On pourra utiliser le fait que, quitte à raisonner sur une dérivée suffisante du polynôme, on peut supposer que le polynôme change de signe en α .

Solution : Un polynôme P possède un nombre fini de racines. Pour chacune d'entre elle, on peut donc trouver un intervalle $[c, d]$ avec c et d rationnels qui l'isole : P possède une unique racine α sur $]c, d[$. P ne change pas de signe sur $[c, \alpha]$ et sur $[\alpha, d]$. Quitte à raisonner sur la dérivée de P (ou une dérivée suffisante de P) on peut supposer que $P(c)$ change de signe en α sur $[c, d]$. En remplaçant P par $-P$ si besoin, supposons $P(c) > 0$. Si α est rationnel, alors α est calculable. Si α n'est pas rationnel, alors il est aussi calculable car pour déterminer si un rationnel q satisfait $q < \alpha$, il suffit de tester si $q \leq c$ ou si ($q > c$ et $P(q) > 0$). Pour déterminer si $P(q) > 0$ ou $P(q) < 0$ (on ne peut pas avoir $P(q) = 0$), il suffit de faire pour $\epsilon = 1/n$ pour n croissant si $P(q) + \epsilon < 0$ alors répondre vrai, et si $P(q) - \epsilon > 0$ alors répondre faux. Cette boucle termine nécessairement, car $P(q) \neq 0$. \square

5. On rappelle que 1 peut aussi s'écrire $1 = 0.99999\dots$

6. Possiblement infinie.

7. Le second étant non-nul pour le quotient, évidemment.

5 Théorème d'interpolation de Craig

Dans ce qui suit, on considère deux formules φ et ψ construites sur deux signatures, Σ_1 pour φ et Σ_2 pour ψ . La signature Σ_0 est la signature dont les symboles (de constantes, relations et fonctions) sont ceux communs à Σ_1 et Σ_2 .

Une formule θ est appelée un interpolant des formules φ et ψ avec $\varphi \vdash \psi$ si l'on a $\varphi \vdash \theta$ et $\theta \vdash \psi$, et si θ est construite sur la signature commune Σ_0 .

Le théorème de Craig affirme que si l'on a deux formules φ et ψ avec $\varphi \vdash \psi$, alors il existe toujours un interpolant.

Question 16. Soit φ la formule $\forall x (x < f(x))$ sur la signature $(\emptyset, \{f\}, \{<\})$ et ψ la formule $\exists y (c < y)$ sur la signature $(\{c\}, \emptyset, \{<\})$.

Proposer un interpolant entre φ et ψ .

Solution : $\forall x \exists y (x < y)$ est un interpolant. □

Soit C un ensemble dénombrable de constantes qui n'apparaissent ni dans Σ_1 ni dans Σ_2 . Soit $\Sigma'_i = \Sigma_i \cup C$ pour $i = 0, 1, 2$.

Soit T_1 une théorie sur la signature Σ'_1 et T_2 une théorie sur la signature Σ'_2 . Une formule θ sur la signature Σ'_0 sépare T_1 de T_2 si $T_1 \vdash \theta$ et $T_2 \vdash \neg\theta$. Deux théories sont dites *inséparables* s'il n'existe aucune telle formule θ .

Question 17. Montrer que si T_1 et T_2 sont inséparables alors T_1 est cohérent et T_2 est cohérent.

Solution : Si T_1 n'était pas cohérent, cela signifierait que $T_1 \vdash F$ et $T_1 \vdash \neg F$ pour une formule F , et donc T_1 prouve une formule toujours fausse ($F \wedge \neg F$). Par conséquent, il prouve aussi n'importe quelle formule F' toujours fausse sur la signature commune. Maintenant, T_2 prouve sa négation $\neg F'$ (toujours vraie). Et donc cette formule séparerait T_1 de T_2 contrairement à l'hypothèse.

On fait un argument symétrique pour T_2 . □

Question 18. Soient les théories $T_1 = \{\varphi\}$ sur la signature Σ'_1 et $T_2 = \{\neg\psi\}$ sur la signature Σ'_2 . Montrer que si T_1 et T_2 sont séparables alors il existe toujours un interpolant des formules φ et ψ .

Solution :

Si $\theta(c_1, \dots, c_k)$ les sépare, alors $T_1 \vdash \theta(c_1, \dots, c_k)$ et $T_2 \vdash \neg\theta(c_1, \dots, c_k)$. Puisque c_1, \dots, c_k n'apparaissent pas dans φ , $\varphi \vdash \forall x_1 \dots \forall x_n \theta(x_1, \dots, x_n)$ (formule de généralisation, voir cours).

On doit avoir par ailleurs $\psi \vdash \neg\forall x_1 \dots \forall x_n \theta(x_1, \dots, x_n)$. Par conséquent, on obtient $\forall x_1 \dots \forall x_n \theta(x_1, \dots, x_n) \vdash \psi$. Donc $\forall x_1 \dots \forall x_n \theta(x_1, \dots, x_n)$ est un interpolant entre φ et ψ . □

On admettra le résultat suivant : si T_1 et T_2 sont inséparables alors $T_1 \cup T_2$ est aussi cohérent (la preuve de ce résultat peut être vue comme une généralisation de la preuve du théorème de complétude).

Question 19. En déduire le théorème de Craig.

Solution : Par contradiction. Supposons que $\varphi \vdash \psi$ et qu'il n'existe pas d'interpolant. Cela veut dire par la question précédente que $T_1 = \{\varphi\}$ et $T_2 = \{\neg\psi\}$ sont inséparables. Par le résultat admis, la théorie $T_1 \cup T_2$ est cohérente. Par le théorème de complétude, elle doit posséder un modèle, et donc on obtient une contradiction avec $\varphi \vdash \psi$. □

Fondements de l'informatique. Examen

Durée: 3h

Sujet proposé par Olivier Bournez

Version 11

(corrigé)

Les 4 parties sont indépendantes, et peuvent être traitées dans un ordre quelconque. On pourra admettre le résultat d'une question pour passer aux questions suivantes. On pourra utiliser tous les résultats et les théorèmes démontrés ou énoncés en cours ou en petite classe ou dans le polycopié sans chercher à les redémontrer. Il est possible d'avoir la note maximale sans répondre à toutes les questions. La difficulté des questions n'est pas une fonction linéaire ni croissante de leur numérotation. La qualité de votre argumentation et de votre rédaction est une partie importante de votre évaluation.

Les paragraphes qui commencent par "Commentaire :" correspondent à des discussions sur les résultats obtenus ou à venir, et peuvent être ignorés. On notera dans cet énoncé la multiplication par le symbole \cdot , et par \mathbb{N} l'ensemble des entiers.

1 Machines de Turing

Question 1. *Le problème suivant est-il décidable ? Déterminer si le langage accepté par une machine de Turing M ne contient que des mots de longueur divisible par 3.*

Solution : C'est indécidable : il s'agit d'une application directe du théorème de Rice. □

Question 2. *On fixe $c \in \mathbb{N}$. Un langage L est dit reconnaissable en espace $c \cdot n$ s'il existe une machine de Turing M qui accepte L et telle que sur tout mot w , M utilise (= écrit) au plus $c \cdot n$ cases de son ruban, où n est la longueur de w .*

Démontrer qu'un langage reconnaissable en espace $c \cdot n$ est décidable.

Solution : L est nécessairement décidable : en effet, il est décidé par une machine de Turing qui simule M en s'arrêtant dès que M essaye d'accéder à une case du ruban à distance plus que $c \cdot n$ fois de la position initiale (dans ce cas on refuse) ou dès que M passe deux fois par la même configuration (dans ce cas on refuse), et en acceptant dès que M accepte : une machine qui utilise uniquement ces $2 \cdot c \cdot n$ cases possède un nombre fini de configurations possibles ; si elle passe deux fois par la même configuration, elle le fera indéfiniment ; sinon, elle ne peut avoir qu'au plus ce nombre de configurations d'étapes avant de s'arrêter. □

2 L'idée d'une startup innovante

Un camarade souhaite créer un nouveau réseau social JeNAimePasEtreContredit. L'originalité de ce réseau social est de garantir à ses utilisateurs d'être mis en correspondance uniquement avec des utilisateurs avec lesquels ils ont déclaré être d'accord. On note $D(x, y)$ pour le fait que

x et y ont déclaré être d'accord lorsqu'ils se sont inscrits¹. On suppose que D est symétrique et réflexif : $D(x, y)$ implique $D(y, x)$ pour tout x, y , et $D(x, x)$ pour tout x .

Il vous embauche comme informaticien et vous demande de produire un algorithme qui prend en entrée une liste d'utilisateurs $L = \{x_1, x_2, \dots, x_n\}$, et la liste des couples (i, j) avec $D(x_i, x_j)$, $1 \leq i \leq n$, $1 \leq j \leq n$, ainsi qu'un entier N , et qui produit en sortie un sous-ensemble S constitué de N utilisateurs tel que $D(x, y)$ pour tout $x, y \in S$, si un tel groupe existe, et qui sinon retourne qu'il n'en existe pas.

Question 3. On suppose que $D(x, y)$ est de plus transitif : $D(x, y)$ et $D(y, z)$ impliquent $D(x, z)$ pour tout x, y, z . Pouvez-vous résoudre son problème en temps polynomial ?

Solution : Il suffit de parcourir les utilisateurs x , et pour chacun de marquer la classe d'équivalence correspondante : c'est-à-dire de marquer tous les autres y avec $D(x, y)$. Par exemple, en parcourant tous les y et en les marquant si $D(x, y)$ apparaît (puisque $D(x, y)$ est transitif, y est dans la classe d'équivalence de x si et seulement si y apparaît). Si l'on trouve une classe d'équivalence avec plus que N sommets on la retourne. Sinon, on renvoie qu'il n'y a pas de tel groupe. Tout cela se fait en temps polynomial. \square

On ne fait plus l'hypothèse dorénavant que $D(x, y)$ est transitif.

Question 4. Pouvez-vous résoudre son problème en temps polynomial ?

Solution : Ce n'est pas possible de le faire en temps polynomial, sauf si $P = NP$ car c'est le problème CLIQUE qui est NP -complet. Rappelons l'énoncé :

Donnée : Un graphe $G = (V, E)$ non-orienté et un entier N

Réponse : Décider si un graphe contient un clique de taille N .

Ici, le réseau social est le graphe tel que les sommets sont les utilisateurs et les arêtes représentent la relation binaire D . \square

Question 5. Il vous demande de produire un algorithme qui prend en entrée une liste d'utilisateurs et une liste de couples comme ci-dessus et qui produit un sous-ensemble S avec simultanément :

1. S contient au moins 10% des utilisateurs de L
2. $D(x, y)$ pour tout $x, y \in S$

si un tel groupe existe, et qui sinon retourne qu'il n'en existe pas. Pouvez-vous résoudre ce nouveau problème en temps polynomial ?

Solution : Ce n'est pas possible de le faire en temps polynomial, sauf si $P = NP$ car ce problème est aussi NP -complet. En effet, d'une part le problème est clairement dans NP : la donnée d'un sous-groupe S est un certificat vérifiable en temps polynomial.

Démontrons en effet que le problème CLIQUE se réduit à ce problème.

On considère la fonction suivante f qui envoie une instance $(G = (V, E), N)$ de CLIQUE (la question précédente) en une instance $G' = (V', E')$ de ce problème. On note n le nombre de sommets du graphe G et n' le nombre de sommets de G' . On construit f pour que $G' = f(G)$ contienne toujours le graphe G , et possède α nouveaux sommets. Le choix de α et les arêtes que l'on ajoute dépendent de si l'on a $N < \frac{n}{10}$ ou le contraire.

Pour $N < \frac{n}{10}$, on prend les α nouveaux sommets comme voisins à tous les autres sommets du graphe G , et on prend α de telle sorte que l'on aie environ $N + \alpha = \frac{n + \alpha}{10} = \frac{n'}{10}$. Très précisément, on prend $\alpha = \lceil \frac{n - 10 \cdot N}{9} \rceil$, ce qui garantit que $n - 10 \cdot N = 9\alpha - k$ pour $k \in \{0, 1, \dots, 8\}$, et donc $N + \alpha = \frac{n + \alpha + k}{10}$, soit $N + \alpha = \lceil \frac{n + \alpha}{10} \rceil = \lceil \frac{n'}{10} \rceil$.

1. On suppose que tous les utilisateurs se connaissent.

Sinon, on a $N \geq \frac{n}{10}$. On prend les α nouveaux sommets comme voisins d'aucun sommet, et on prend $\alpha = 10 \cdot N - n$, de telle sorte que $N = \frac{n+\alpha}{10} = \frac{n'}{10}$.

La fonction f ci-dessus se calcule bien en temps polynomial. Il reste à vérifier que G possède une clique avec exactement N sommets si et seulement si $G' = f(G)$ possède une clique avec au moins $\frac{n'}{10}$ sommets.

Supposons que G possède une clique S avec exactement N sommets. Pour $N < \frac{n}{10}$: S union les α nouveau sommets est aussi une clique dans G' : elle est de taille $N + \alpha$ et donc au moins $\frac{n'}{10}$. Pour $N \geq \frac{n}{10}$: S est une clique dans G' : elle est de taille $N = \frac{n'}{10}$. Donc, dans tous les cas G' possède une clique avec au moins $\frac{n'}{10}$ sommets.

Réciproquement, supposons que G' possède une clique S' avec au moins $\frac{n'}{10}$ sommets.

Pour $N < \frac{n}{10}$: puisque $N + \alpha = \lceil \frac{n'}{10} \rceil$, le nombre de sommets β dans la clique S' doit satisfaire $\beta \geq N + \alpha$. Par conséquent, la clique S' contient au moins N sommets de G , et donc il existe une clique $S = S'$ d'au moins N sommets dans G : en prenant n'importe quel sous-ensemble de taille N de S on obtient que G possède une clique de taille exactement N .

Pour $N \geq \frac{n}{10}$: comme tous les nouveaux sommets sont voisin d'aucun autre sommet, ils ne peuvent pas appartenir à une clique de plus de 1 élément.² Par conséquent, S' forme nécessairement une clique de G avec au moins $N = \frac{n'}{10}$ sommets : en prenant n'importe quel sous-ensemble de taille N de S' on obtient que G possède une clique de taille exactement N . \square

3 Théorème de Cobham

On écrit $\|x\| = \lceil \log_2(x+1) \rceil$ pour représenter la taille de l'écriture en binaire de l'entier x . Par exemple, $\|23\| = 5$ car $23 = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$ s'écrit en binaire 10111 avec 5 lettres. Pour un k -uplet d'entiers $\vec{x} = (x_1, x_2, \dots, x_k)$, on écrit $\|\vec{x}\|$ pour $\|x_1\| + \dots + \|x_k\|$.

Commentaire : On pourra librement utiliser le fait que l'on a toujours $x < 2^{\|x\|} \leq 2 \cdot x + 1$ pour tout entier x dans toute la suite.

On utilisera les fonctions suivantes de \mathbb{N} dans \mathbb{N} : $s(x) = x + 1$, $\text{double}_0(x) = 2 \cdot x$ et $\text{double}_1(x) = 2 \cdot x + 1$ ainsi que la fonction $\sharp(x, y)$ de \mathbb{N}^2 dans \mathbb{N} définie par $\sharp(x, y) = 2^{\|x\| \cdot \|y\|} - 1$.

Commentaire : La fonction \sharp vérifie la propriété $\|\sharp(x, y)\| = \|x\| \cdot \|y\|$ pour tout x et y .

Toutes les fonctions sur les entiers que l'on considère dans cette partie sont totales : c'est-à-dire, définies pour toute valeur de leurs arguments.

3.1 FP et Fonctions primitives récursives

Une fonction $f : \mathbb{N}^n \rightarrow \mathbb{N}$ est *calculable en temps polynomial* s'il existe une machine de Turing M_f qui, produit $f(x_1, \dots, x_n)$ (écrit en binaire) à partir de x_1, x_2, \dots et x_n (écrits en binaire) en un nombre d'étapes polynomial en $\|\vec{x}\|$ où $\vec{x} = (x_1, \dots, x_n)$. On note FP pour la classe des fonctions calculables en temps polynomial.

Question 6. Démontrer qu'une fonction de FP est nécessairement de taille polynomiale : c'est-à-dire qu'il existe un polynôme p tel que $\|f(\vec{x})\| \leq p(\|\vec{x}\|)$ pour tout \vec{x} .

Solution : En un temps T , une machine de Turing ne peut pas écrire plus que T cases de son ruban. Donc la taille du résultat ne peut pas être plus grande qu'un polynôme en la taille de l'entrée après un temps polynomial. \square

Question 7. Montrer que la fonction qui à x associe 2^x n'est pas calculable en temps polynomial.

2. Pour les plus puristes, on observera que la preuve dans le polycopié de la NP-complétude du problème CLIQUE montre que ce dernier reste NP-complet en supposant $N \geq 2$: par exemple en ajoutant des clauses "inutiles" à la formule 3SAT dans la preuve de $3SAT \leq STABLE$.

Solution : Cette fonction n'est pas de taille polynomiale : en effet, pour $x = 2^n - 1$, $\|x\| = n$, $\|2^x\| = \lceil \log_2(2^{2^n-1} + 1) \rceil \geq \lceil \log_2(2^{2^n-1}) \rceil = 2^n - 1 \geq 2^{n-1}$ pour n assez grand, ce qui croît plus vite que tout polynôme en n pour n assez grand. \square

Commentaire : La fonction qui à x associe 2^x est primitive récursive (cf PC1, question 1.2). Les fonctions primitives récursives et les fonctions de FP sont donc deux classes de fonctions distinctes.

Commentaire : Une idée due à Cobham pour s'approcher de FP est de remplacer la fonction $s(x) = x + 1$ qui sert de fondement dans les définitions par récurrence dans le schéma Rec des fonctions primitives récursives par les fonctions $\text{double}_0(x)$ et $\text{double}_1(x)$. On va d'abord voir que cela ne suffit pas (question 9). On va ensuite voir que se restreindre aux récurrences bornées permet de garantir que l'on obtient que des fonctions de FP (question 10). On verra ensuite que cela donne même une caractérisation de FP (théorème de Cobham).

3.2 Récurrence et taille

Question 8. On suppose que f est définie par une récurrence du type

$$\begin{cases} f(0, x_2, \dots, x_n) = g(x_2, \dots, x_n), \\ f(\text{double}_0(x_1), x_2, \dots, x_n) = h_0(f(x_1, \dots, x_n), x_1, \dots, x_n) \text{ pour } x_1 \neq 0 \\ f(\text{double}_1(x_1), x_2, \dots, x_n) = h_1(f(x_1, \dots, x_n), x_1, \dots, x_n), \end{cases}$$

pour tout entiers x_1, x_2, \dots, x_n , avec les fonctions g, h_0 , et h_1 calculables en temps polynomial.

Démontrer que si f est de taille polynomiale alors f se calcule en temps polynomial.

Solution : Sur une entrée $\vec{x} = (x_1, x_2, \dots, x_n)$, il suffit de "dérouler" la définition, ce qui fera $\|x_1\|$ itérations, chaque itération se faisant en temps polynomial en $\|x\|$. Tous les valeurs intermédiaires dont on a besoin restent bien de taille polynomiale en $\|x\|$. \square

Question 9. Donner un exemple de fonction f définie par une récurrence du type

$$\begin{cases} f(0) = 1, \\ f(\text{double}_0(x_1)) = h_0(f(x_1)) \text{ pour } x_1 \neq 0 \\ f(\text{double}_1(x_1)) = h_1(f(x_1)) \end{cases} \quad (1)$$

où h_0 et h_1 sont des fonctions calculables en temps polynomial, mais où f n'est pas de taille polynomiale.

Solution : On considère la fonction $x \mapsto 2^x$ de la question 7 : on a $2^0 = 1$, $2^{\text{double}_0(x)} = (2^x)^2$, et $2^{\text{double}_1(x)} = 2 \cdot (2^x)^2$. On pose donc $h_0(x) = x^2$, et $h_1(x) = 2 \cdot x^2$. h_0 et h_1 se calculent bien en temps polynomial. \square

3.3 Récurrence bornée

Définition. On dit qu'une fonction f est définie par *récurrence bornée* à partir des fonctions g, h_0, h_1 et m si

$$\begin{cases} f(0, x_2, \dots, x_n) = g(x_2, \dots, x_n), \\ f(\text{double}_0(x_1), x_2, \dots, x_n) = h_0(f(x_1, \dots, x_n), x_1, \dots, x_n) \text{ pour } x_1 \neq 0 \\ f(\text{double}_1(x_1), x_2, \dots, x_n) = h_1(f(x_1, \dots, x_n), x_1, \dots, x_n), \end{cases}$$

et si de plus

$$f(x_1, x_2, \dots, x_n) \leq m(x_1, \dots, x_n),$$

pour tout x_1, x_2, \dots, x_n .

Définition. Une fonction $f : \mathbb{N}^n \rightarrow \mathbb{N}$ est dans la classe de Cobham si elle est l'une des fonctions :

- la constante 0 (alors $n = 0$)
- $\text{double}_0 : x \mapsto 2 \cdot x$ et $\text{double}_1 : x \mapsto 2 \cdot x + 1$ (alors $n = 1$) ;
- $\sharp : (x, y) \mapsto 2^{\|x\| \cdot \|y\|} - 1$ (alors $n = 2$) ;
- $\text{Proj}_n^i : (x_1, \dots, x_n) \mapsto x_i$ les fonctions de projection, pour $1 \leq i \leq n$;
- $\text{Comp}_n(g, h_1, \dots, h_m) : (x_1, \dots, x_n) \mapsto g(h_1(x_1, \dots, x_n), \dots, h_m(x_1, \dots, x_n))$ la composition des fonctions dans la classe de Cobham g, h_1, \dots, h_m (pour $m \geq 0$) ;
- $\text{BoundedRecNotation}(g, h_0, h_1, m)$ une fonction f définie par *récurrence bornée* à partir des fonctions g, h_0, h_1 et m , avec g, h_0, h_1 et m dans la classe de Cobham.

Par exemple la fonction successeur $s(x) = x + 1$ est dans la classe de Cobham car elle se définit par récurrence bornée³

$$\begin{cases} s(0) = \text{double}_1(0) \\ s(\text{double}_0(x)) = \text{double}_1(x) \text{ pour } x \neq 0 \\ s(\text{double}_1(x)) = \text{double}_0(s(x)) \\ s(x) \leq \text{double}_1(x) \end{cases}$$

Deuxième exemple : la fonction taille $\|x\|$ est dans la classe de Cobham car $\|0\| = 0$, $\|\text{double}_0(x)\| = s(\|x\|)$ pour $x \neq 0$, $\|\text{double}_1(x)\| = s(\|x\|)$, $\|x\| \leq \text{Proj}_1^1(x) = x$.

Troisième exemple, la fonction $\text{concat}(y, x) = 2^{\|y\|} \cdot x + y$ qui retourne un entier dont l'écriture en binaire est la concaténation de l'écriture de x et de y en binaire : elle est dans la classe de Cobham car $\text{concat}(0, x) = x$, $\text{concat}(\text{double}_0(y), x) = \text{double}_0(\text{concat}(y, x))$, $\text{concat}(\text{double}_1(y), x) = \text{double}_1(\text{concat}(y, x))$, $\text{concat}(y, x) \leq \sharp(\|x\|, \|y\|)$.

Question 10. Démontrer que toute fonction de la classe de Cobham se calcule en temps polynomial.

Solution : Cela se fait par induction sur la définition. Les fonctions s , double_0 et double_1 sont dans FP. Les fonctions projections aussi. La fonction \sharp est dans FP : il suffit d'écrire en binaire un 1 suivi de 0 avec un nombre de 0 donné par le produit de la longueur de x et de la longueur de y moins 1. Tout cela se calcule en temps polynomial.

Les fonctions de FP sont stables par composition (composer les machines de Turing).

Supposons que f est définie par récurrence bornée à partir de g, h_0, h_1 et m dans FP. Par hypothèse d'induction, toutes ces fonctions sont dans FP. Or la taille de $f(x_1, \dots, x_n)$ est bornée par $\|m(x_1, \dots, x_n)\|$ qui est polynomiale : la fonction m étant dans FP sa taille est polynomiale par la question 6. Cela implique que f est dans FP par la question 8. \square

Question 11. Montrer que les fonctions suivantes sont dans la classe de Cobham :

1. la fonction $\text{mod}_2(x)$ qui donne le reste de la division de x par 2,
2. la fonction $\text{quo}_2(x)$ qui donne le quotient de la division de x par 2,
3. la fonction $\text{cond}(x, y)$ qui donne 0 pour $x = 0$ et y sinon.

Solution : $\text{mod}_2(0) = 0$, $\text{mod}_2(\text{double}_0(x)) = 0$, $\text{mod}_2(\text{double}_1(x)) = 1$, $\text{mod}_2(x) \leq \text{Proj}_1^1(x) = x$.
 $\text{quo}_2(0) = 0$, $\text{quo}_2(\text{double}_0(x)) = x$ pour $x \neq 0$, $\text{quo}_2(\text{double}_1(x)) = x$, $\text{quo}_2(x) \leq \text{Proj}_1^1(x) = x$.
 $\text{cond}(0, y) = 0$, $\text{cond}_0(\text{double}_0(x), y) = \text{cond}_0(x, y)$ pour $x \neq 0$, $\text{cond}_0(\text{double}_1(x), y) = y$,
 $\text{cond}_0(x, y) \leq \text{Proj}_2^2(x, y) = y$. \square

3. Une définition plus formelle serait de dire que $s(x)$ est la fonction

$$\text{BoundedRecNotation}(\text{Comp}_0(\text{double}_1, 0), \text{Comp}_2(\text{double}_1, \text{Proj}_2^2), \text{Comp}_2(\text{double}_0, \text{Proj}_2^1), \text{double}_1)$$

mais on ne demandera pas des écritures aussi formelles.

Question 12. On considère le polynôme $p(n) = c \cdot n^h$ pour deux entiers c et h . Montrer qu'il existe une fonction $T(x)$ de \mathbb{N} dans \mathbb{N} dans la classe de Cobham telle que pour tout x , $2^{p(\|x\|)} \leq T(x)$.
 Indice : on pourra chercher à construire une fonction M avec $p(\|x\|) \leq \|M(x)\|$ pour tout x .

Solution : Etant donné un entier d , on peut construire une fonction f_d qui à x associe un mot $f_d(x)$ de taille $d \cdot \|x\|$: par exemple en prenant $f_d(x)$ qui est la concaténation d fois de x (par exemple à l'aide de la fonction `concat`). On peut alors définir $M(x)$ par $f_d(x) \# f_d(x) \# \dots \# f_d(x)$ où f_d est répété h fois pour une constante entière $d \geq \sqrt[h]{c}$, et où on note $x \# y$ pour $\#(x, y)$ et utilise l'associativité de cette opération. Par construction, $\|M(x)\|$ est de taille $(d \cdot \|x\|)^h$ donc plus grande que $c \cdot \|x\|^h = p(\|x\|)$. On pose alors $T(x) = 2 \cdot M(2^{\|x\|}) + 1$ en utilisant le fait que $2^{\|M(x)\|} \leq 2 \cdot M(x) + 1$. La fonction T est bien dans la classe de Cobham, car elle s'écrit $\text{double}_1(M(\mathbf{g}(x)))$ avec $\mathbf{g}(x) = 2^{\|x\|}$ donnée par exemple par $\mathbf{g}(0) = 1$, $\mathbf{g}(\text{double}_0(x)) = \text{double}_0(\mathbf{g}(x))$ pour $x \neq 0$, $\mathbf{g}(\text{double}_1(x)) = \text{double}_0(\mathbf{g}(x))$ et $\mathbf{g}(x) \leq f_2(x)$. \square

On admettra qu'il existe une fonction dans la classe de Cobham $(x, y, z) \mapsto \langle x, y, z \rangle$ qui envoie bijectivement \mathbb{N}^3 sur \mathbb{N} ainsi que trois fonctions $\pi_1, \pi_2, \pi_3 : \mathbb{N} \rightarrow \mathbb{N}$ dans la classe de Cobham telles que $\pi_i(\langle x_1, x_2, x_3 \rangle) = x_i$ pour $i = 1, 2, 3$.

Une configuration d'une machine de Turing peut se coder par un entier. Celui-ci code un triplet : un entier qui marque le numéro de l'état ; un entier qui code la partie du ruban à gauche de la tête de lecture jusqu'au premier B ; un entier qui code la partie du ruban à droite de la tête de lecture jusqu'au premier B , ses bits de poids faible étant les plus proches de la tête de lecture. Par exemple, le ruban $(B, 1, 1, \underline{0}, 0, 0, 1, B)$ avec la machine dans l'état numéro 5 et la tête de lecture au niveau du caractère souligné peut se coder par $\langle 5, 3, 8 \rangle$ car $3 = 1 \cdot 2^1 + 1$ et $8 = 0 \cdot 2^0 + 0 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3$. Observer l'ordre des puissances de 2 dans ces écritures.

On fixe une machine de Turing M qui calcule une fonction de FP de \mathbb{N} dans \mathbb{N} .

Question 13. Exprimer par une fonction de la classe de Cobham le déplacement de la tête de lecture vers la droite, sans changer l'état de la machine et le caractère lu.

Solution : La configuration est donnée par un certain $\langle a, b, c \rangle$. Lors d'un mouvement vers la droite, c est divisé par 2, tandis que b est multiplié par 2 et le bit final de c devient son dernier bit. Ceci s'écrit

$$\text{RightMove}(x) = \langle \pi_1(x), \text{Cond}(\text{mod}_2(\pi_3(x)), \text{double}_0(\pi_2(x)), \text{double}_1(\pi_2(x))), \text{quo}_2(\pi_3(x)) \rangle.$$

\square

À ce stade on admet qu'on peut écrire une fonction $\text{Next}_M(\vec{x})$ dans la classe de Cobham qui code la fonction de transition d'une machine de Turing M donnée : si \vec{x} code une configuration, alors $\text{Next}_M(\vec{x})$ code la configuration suivante.

Question 14. On rappelle que M calcule une fonction de FP de \mathbb{N} dans \mathbb{N} . Montrer qu'il existe une fonction $\text{Run}_M(y, x)$ dans la classe de Cobham qui retourne la configuration de la machine après $\|y\|$ étapes sur l'entrée x .

Solution : On peut supposer sans perte de généralité que l'état initial q_0 de la machine M porte le numéro 0.

Le ruban de la machine M ne peut pas s'être déplacé de plus de $t = \|y\|$ cases au temps t . Par ailleurs, comme la machine M calcule une fonction de FP, elle effectue un nombre d'étapes t au plus polynomial en $\|x\|$. D'autre part, puisque $\langle \dots, \dots, \dots \rangle$ est dans la classe de Cobham, la taille de $\langle a, b, c \rangle$ reste de taille polynomiale en celle du triplet (a, b, c) . On en déduit qu'il existe un polynôme p tel que $\|\text{Run}_M(y, x)\| \leq p(\|x\| + \|y\|)$ pour tout x et y . Par conséquent,

4. B est le symbole de blanc.

$\text{Run}_M(y, x) \leq 2^{p(\|x\|+\|y\|)} = 2^{p(\|\text{concat}(x,y)\|)} \leq T(\text{concat}(x,y))$ où T est la fonction dans la classe de Cobham de la question 12 pour le polynôme p .

La fonction Run_M est bien dans la classe de Cobham car elle s'écrit :

$$\begin{cases} \text{Run}_M(0, x) = \langle 0, 0, x \rangle \\ \text{Run}_M(\text{double}_0(y), x) = \text{Next}(\text{Run}_M(y, x)) & \text{pour } y \neq 0, \\ \text{Run}_M(\text{double}_1(y), x) = \text{Next}(\text{Run}_M(y, x)), \\ \text{Run}_M(y, x) \leq T(\text{concat}(x, y)) \end{cases}$$

□

En déduire le théorème de Cobham :

Question 15. Une fonction est dans classe de Cobham si et seulement si elle appartient à FP.

Solution : Le fait qu'une fonction dans la classe de Cobham est dans FP est la question 10. Réciproquement, si une fonction f de \mathbb{N} dans \mathbb{N} est calculable en temps polynomial $c \cdot n^k$, alors f s'écrit $\pi_3(\text{Run}_M(M(x), x))$ où la fonction M est celle de la solution de la question 12, vérifiant $c \cdot \|x\|^k \leq \|M(x)\|$.

Par ailleurs, tout ce raisonnement se généralise aux fonctions de \mathbb{N}^k dans \mathbb{N} sans difficulté : considérer des machines avec k rubans par exemple (un ruban par argument). □

4 Complétude de la théorie des corps algébriquement clos & Théorème d'Ax

Dans tout cet exercice, on ne considère que des formules et théories du premier ordre (c'est-à-dire les formules considérées dans le cours et le polycopié). On suppose que les signatures sont dénombrables.

4.1 Complétude d'une théorie

Une théorie consistante T sur une signature Σ est dite *complète* si pour toute formule close ϕ sur la signature Σ , on a $T \vdash \phi$ ou $T \vdash \neg\phi$.

Question 16. Soit T une théorie complète, dont les axiomes sont récursivement énumérables. Montrer qu'elle est décidable : il y a un algorithme qui prend en entrée une formule close ϕ et qui détermine si $T \vdash \phi$.

Solution : Par la complétude de T , étant donnée ϕ soit il y a une preuve de ϕ soit il y a une preuve de $\neg\phi$ à partir de T . On peut énumérer systématiquement toutes les formules prouvables à partir des axiomes de T jusqu'à tomber soit sur la preuve de ϕ soit sur la preuve de $\neg\phi$. On répond vrai dans le premier cas, et faux dans le second. Cette machine ne boucle pas car l'un des deux cas doit se produire. □

Deux structures M et N sur une signature Σ sont *isomorphes* s'il existe une bijection entre l'ensemble de base de M et de N qui préserve l'interprétation des symboles de fonctions, de relations et des symboles de constantes : on admettra⁵ que dans ce cas, pour toute formule close ϕ sur la signature Σ , on a $M \models \phi$ si et seulement si $N \models \phi$: autrement dit, M et N satisfont exactement les mêmes formules closes sur la signature Σ , quand M et N sont isomorphes.

5. Cela se prouve par une induction sans surprise.

4.2 Corps algébriquement clos

On considère la théorie ACF des corps algébriquement clos⁶ : c'est la théorie constituée des axiomes des corps commutatifs auxquels on ajoute pour chaque $n \geq 1$ l'axiome

$$\forall x_0 \forall x_1 \cdots \forall x_{n-1} \exists x (x_0 + x_1 \cdot x + x_2 \cdot x^2 + \cdots + x_{n-1} \cdot x^{n-1} + x^n) = 0$$

comme dans le cours. On notera Σ la signature de cette théorie.

Pour un entier p , on note comme dans le cours F_p la formule $\mathbf{1} + \cdots + \mathbf{1} = 0$, où $\mathbf{1}$ est répété p fois. On note ACF_p la théorie des corps algébriquement clos de caractéristique p , pour $p \geq 0$. Formellement⁷ : pour $p \geq 1$, ACF_p est constituée des axiomes de ACF et de la formule F_p et des formules $\neg F_q$ pour tout entier $0 < q < p$. ACF_0 est constituée des axiomes de ACF auxquels on ajoute pour chaque $n \geq 1$ la formule $\neg F_n$.

On dit que deux ensembles A et B sont de même cardinal s'il existe une bijection entre A et B .

Tout ce qui suit est basé uniquement sur des arguments de logique, et nécessite uniquement les concepts et résultats suivants d'algèbre⁸ :

- (α) \mathbb{C} , le corps des complexes, est un corps algébriquement clos de caractéristique 0.
- (β) Pour chaque entier premier p , il y a un corps $\overline{K_p}$ algébriquement clos de caractéristique p .
- (γ) Si deux corps algébriquement clos sont de même caractéristique et de même cardinal alors ils sont isomorphes.

Question 17. *Montrer que ACF n'est pas une théorie complète.*

Solution : En effet, par exemple on n'a pas $ACF \vdash F_2$ ni $ACF \vdash \neg F_2$: \mathbb{C} est un corps qui vérifie ACF et qui n'est pas de caractéristique 2 et donc on ne saurait avoir $ACF \vdash F_2$ (par le théorème de correction, tout modèle de ACF devrait satisfaire F_2). De même on connaît des corps algébriquement clos de caractéristique 2 et donc on ne saurait avoir $ACF \vdash \neg F_2$. \square

4.3 ACF_0 est complète

On dira qu'un modèle est fini (respectivement : infini) si son ensemble de base l'est.

On admettra qu'une modification de la preuve du théorème de complétude permet de le renforcer en le résultat suivant : Soit A un ensemble avec un nombre infini d'éléments. Toute théorie T sur une signature dénombrable⁹ qui possède un modèle infini possède un modèle (dont l'ensemble de base est) de même cardinal que A .

Question 18. *Soit A un ensemble avec un nombre infini d'éléments. Soit T une théorie (consistante) dont tous les modèles sont infinis. Supposons que tous les modèles de T de même cardinal que A sont isomorphes. Démontrer que T est complète.*

Solution : On raisonne par l'absurde. Si T n'est pas complète, alors il existe une formule ϕ telle que $T \cup \{\phi\}$ et $T \cup \{\neg\phi\}$ admettent des modèles. Comme ces modèles sont en particulier modèles de T , ils sont infinis. D'après le résultat qui précède la question, on peut les prendre de même

6. On rappelle qu'un corps algébriquement clos est un corps commutatif où tout polynôme de degré supérieur ou égal à un admet une racine.

7. Ce n'est pas exactement comme dans le cours, car on met ici explicitement les formules $\neg F_q$ pour tout entier $0 < q < p$ (note : cela est équivalent quand p est premier, mais cette remarque n'est pas utile pour cet exercice et on prendra dans la suite la définition indiquée dans cet énoncé).

8. Pour les connaisseurs en algèbre : pour (β), il s'agit de la clôture algébrique de $\mathbb{Z}/p\mathbb{Z}$; pour (γ), cela découle du fait qu'un corps algébriquement clos est déterminé à isomorphisme près par sa caractéristique et son degré de transcendance. Mais comprendre ces concepts et ces faits n'est pas nécessaire pour ce sujet.

9. C'est-à-dire avec un nombre dénombrable de symboles de constantes, de fonctions et de relations. C'est le cas de la signature Σ .

cardinal que A . Alors par hypothèse ils sont isomorphes, mais l'isomorphisme ne peut préserver l'interprétation de ϕ , d'où la contradiction. \square

Question 19. *Montrer que tous les modèles de ACF_0 sont infinis.*

Solution : On note $\bar{0}$ l'interprétation de 0 , et \bar{p} l'interprétation de $1 + 1 + \dots + 1$ où la constante 1 est répétée p fois, pour chaque entier p . Pour tout couple d'entier $i < j$, on a nécessairement $\bar{i} \neq \bar{j}$: en effet, sinon, par les axiomes des corps, on devrait avoir $\overline{j-i} = \bar{0}$, ce qui est impossible par l'axiome $\neg F_{j-i}$. Il y a donc au moins autant d'éléments que d'entiers dans l'ensemble de base du corps. \square

Question 20. *En déduire que ACF_0 est complète.*

Solution : C'est une application directe de la question 18 : ACF_0 possède un modèle infini (\mathbb{C}) et deux modèles de ACF_0 de même cardinal sont isomorphes par le fait (γ). \square

Commentaire : On peut aussi montrer que ACF_p est complète pour $p \geq 1$.

4.4 Théorème d'Ax

On veut d'abord démontrer le résultat suivant : Les propositions suivantes sont équivalentes. Soit ϕ une formule close sur la signature Σ .

- i ϕ est vraie sur \mathbb{C}
- ii ϕ est vraie dans tous les corps algébriquement clos de caractéristique 0
- iii ϕ est vraie dans au moins un corps algébriquement clos de caractéristique 0
- iv Pour tout entier m , il y a un entier premier $p > m$ tel que ϕ est vraie dans un corps algébriquement clos de caractéristique p
- v Il y a un entier m tel que pour tout entier premier $p > m$, ϕ est vraie dans tous les corps algébriquement clos de caractéristique p

Le fait que [i] implique [iii] et que [v] implique [iv] est évident.

Question 21. *Montrer que [i], [ii] et [iii] sont équivalents.*

Solution : Ce n'est rien d'autre que la complétude de ACF_0 . \square

Question 22. *Montrer que [ii] implique [v]*

Solution : Supposons que $ACF_0 \models \phi$. Par le théorème de complétude, il y a une preuve de ϕ à partir des axiomes de ACF_0 . Cette preuve n'utilise qu'un nombre fini d'assertions $\neg F_q$, donc pour un p suffisamment grand, $ACF_p \vdash \phi$, et donc aussi $ACF_p \models \phi$. \square

Question 23. *Montrer que [iv] implique [iii]*

Solution : Supposons que $ACF_0 \not\models \phi$. Par la complétude de ACF_0 , on a $ACF_0 \models \neg\phi$. Par le théorème de complétude, il y a une preuve de $\neg\phi$ à partir des axiomes de ACF_0 . Cette preuve n'utilise qu'un nombre fini d'assertions $\neg F_q$, donc pour tout p suffisamment grand, $ACF_p \vdash \neg\phi$, et donc aussi $ACF_p \models \neg\phi$. Cela veut dire que donc [iv] devient fausse, contradiction. \square

On admettra que pour tout entier premier p , le corps algébriquement clos $\overline{K_p}$ est tel que toute fonction polynomiale injective de $(\overline{K_p})^n$ dans $(\overline{K_p})^n$ est surjective.

Question 24. *Démontrer le théorème d'Ax : Toute fonction polynomiale injective de $\mathbb{C}^n \rightarrow \mathbb{C}^n$ est surjective.*

Solution : Supposons que ce ne soit pas le cas. Notons $\vec{x} = (X_1, X_2, \dots, X_n)$. Soit $F(\vec{x})$ un contre-exemple avec $F(\vec{x}) = (F_1(\vec{x}), \dots, F_n(\vec{x}))$ où chaque polynôme $F_i \in \mathbb{C}[X_1, X_2, \dots, X_n]$ est de degré au plus d . On peut écrire une formule close $\Phi_{n,d}$ telle que, pour K un corps, $K \models \Phi_{n,d}$ si et seulement si toute fonction injective polynomiale $G : K^n \rightarrow K^n$ dont les fonctions coordonnées ont un degré au plus d est surjective. On peut quantifier sur les polynômes de degré au plus d en quantifiant sur les coefficients.

Par exemple, $\Phi_{2,2}$ est la formule

$$\begin{aligned} & \forall a_{0,0} \forall a_{0,1} \forall a_{0,2} \forall a_{1,1} \forall a_{2,0} \forall b_{0,0} \forall b_{0,1} \forall b_{0,2} \forall b_{1,0} \forall b_{1,1} \forall b_{2,0} \\ & \quad (\forall x_1 \forall y_1 \forall x_2 \forall y_2 \\ & \quad \sum a_{i,j} x_1^i y_1^j = \sum a_{i,j} x_2^i y_2^j \wedge \sum b_{i,j} x_1^i y_1^j = \sum b_{i,j} x_2^i y_2^j \Rightarrow x_1 = x_2 \wedge y_1 = y_2) \\ & \quad \Rightarrow \forall u \forall v \exists x \exists y \sum a_{i,j} x^i y^j = u \wedge \sum b_{i,j} x^i y^j = v. \end{aligned}$$

On montre la propriété iv en prenant $\Phi_{n,d}$ pour la formule ϕ . Ainsi en appliquant les questions précédentes, on en déduira que $\mathbb{C} \models \Phi_{n,d}$ ce qui mène à une contradiction.

Montrons la propriété iv : soit un entier m . On prend un entier premier $p > m$. Par la propriété (β), il y a un corps algébriquement clos $\overline{K_p}$ de caractéristique p . Par la propriété ci-dessus, il satisfait la formule $\Phi_{n,d}$. \square

Notes bibliographiques

La partie sur le théorème de Cobham est inspirée de notes de Cours de Arnaud Durand. Le théorème a été énoncé par Cobham dans [2], mais sans vrais détails sur la preuve. Se référer à [4] ou [1] pour des preuves détaillées.

L'idée de la partie sur la complétude des corps algébriquement clos est née du post dans le blog *Xor's Hammer* de Mkoconnor du 15 Août 2008. Voir <https://xorhammer.com/2008/08/15/axs-theorem/>. Elle est au final inspirée de [3].

Références

- [1] Peter Clote and Evangelos Kranakis. *Boolean functions and computation models*. Springer Science & Business Media, 2013.
- [2] A. Cobham. The intrinsic computational difficulty of functions. In Y. Bar-Hillel, editor, *Proceedings of the International Conference on Logic, Methodology, and Philosophy of Science*, pages 24–30. North-Holland, Amsterdam, 1962.
- [3] David Marker et al. Introduction to the model theory of fields. In *Model theory of Fields*, pages 1–37. Association for Symbolic Logic, 1996.
- [4] H. E. Rose. *Subrecursion, Functions and Hierarchies*. Clarendon Press, Oxford, 1984.