École polytechnique — X2018 — INF412 Fondements de l'informatique

PC6 notée

Sujet proposé par Samuel Mimram et Amaury Pouly et Bruno Salvy

Cet énoncé comporte quatre parties indépendantes et qui pourront être résolues dans n'importe quel ordre. Dans chaque partie, on pourra, pour répondre à une question, admettre les résultats dont on demande la démonstration aux questions *précédentes*. Il n'est pas nécessaire de traiter toutes les questions pour avoir la note maximale. Les correcteurs vous remercient d'avance d'écrire lisiblement.

1 Graphes connexes

On considère dans cette section des graphes non-orientés, c'est-à-dire des paires d'ensembles (V, E) avec $E \subseteq V \times V$ telles que si $(x, y) \in E$ alors $(y, x) \in E$. On rappelle que la longueur d'un chemin est le nombre de ses arêtes (c'est un entier naturel) et qu'un graphe est connexe lorsqu'il existe un chemin entre toute paire de sommets.

Nous allons montrer qu'il n'existe pas de théorie du premier ordre dont les modèles sont les graphes connexes. Nous raisonnons par l'absurde et supposons qu'il existe une théorie du premier ordre $\mathcal{C}$ sur la signature constituée d'un unique symbole de relation binaire E, dont les modèles sont les graphes connexes.

Question 1.1. *Décrivez*

1. *étant donné $n \in \mathbb{N}$, une théorie du premier ordre $\mathcal{T}_n$ dont les modèles sont les triplets (G, a, b) où G est un graphe connexe et a et b sont des sommets de G tels qu'il n'existe pas de chemin de longueur au plus n de a à b,*
2. *une théorie du premier ordre $\mathcal{T}$ dont les modèles (s'il en existe) sont les triplets (G, a, b) où G est un graphe connexe et a et b sont des sommets de G tels qu'il n'existe aucun chemin de a à b.*

Question 1.2. *Montrez que la théorie $\mathcal{T}_n$ admet un modèle, pour tout $n \in \mathbb{N}$.*

Question 1.3. *Montrez que la théorie $\mathcal{T}$ admet un modèle.*

Question 1.4. *Montrez qu'il ne peut pas exister de théorie $\mathcal{C}$ dont les modèles sont les graphes connexes.*

2 Une variante des machines de Turing

Les machines de Turing définies dans le cours ont une fonction de transition

$$\delta : Q \times \Gamma \to \Gamma \times \{\leftarrow, |, \rightarrow\} \times Q$$

qui, à un état et un caractère présent sur le ruban, associe un caractère à écrire sur le ruban, un déplacement de la tête de lecture et un nouvel état. Le cycle de travail est donc :

lecture → écriture → déplacement → transition vers un autre état.

On s'intéresse dans cet exercice à une variante de machines dont le cycle de travail serait

écriture → déplacement → lecture → transition vers un autre état,

avec une fonction de transition de signature

$$\delta' : Q \to \Gamma \times \{\leftarrow, |, \rightarrow\} \times (\Gamma \to Q)$$

qui écrit un caractère, déplace la tête de lecture, et change d'état en fonction du caractère qui se trouve alors sous la tête de lecture. Initialement la tête de lecture est toujours sur le premier caractère blanc (B) à gauche du mot à traiter. On appelera ces machines des machines de Turing modifiées.

Question 2.1. *En se limitant au cas où l'alphabet de travail Γ est réduit à $\{a, b, B\}$, écrire une machine de Turing modifiée qui déplace la tête de lecture jusqu'au premier b sur la droite de sa position initiale, sans modifier le mot. (Ces machines ne seront pas représentées graphiquement ; donner juste l'ensemble des états et la fonction de transition.)*

Question 2.2. *Montrer comment simuler une machine de Turing modifiée par une machine de Turing.*

Question 2.3. *Montrer comment simuler une machine de Turing (dont la tête de lecture est initialement sur le premier caractère blanc (B) à gauche du mot à traiter) par une machine de Turing modifiée. [Indication : la machine de Turing modifiée peut avoir plus d'états que la machine de Turing initiale.]*

3 Problèmes de décision

Parmi les problèmes suivants, lesquels sont décidables, lesquels sont indécidables, et lesquels sont semi-décidables ? (Argumentez votre réponse).

Question 3.1. *Déterminer si une machine de Turing M accepte au moins 2019 mots.*

Question 3.2. *Déterminer si une machine de Turing M accepte au moins 2019 mots en au plus 2020 étapes.*

Question 3.3. *Déterminer si le langage reconnu par une machine de Turing M est reconnu par une machine (éventuellement différente de M) dont le nombre d'états est un multiple de 2019.*

4 Systèmes d'étiquettes

Les systèmes d'étiquettes [1] sont des machines très simples inventées par Emil Post en 1943. Malgré leur apparente simplicité, nous allons voir que leurs comportements peuvent être très compliqués.

Un système d'étiquettes est défini par un triplet (m, Σ, P) où :

— m est un entier naturel indiquant le nombre de lettres à effacer à chaque étape ;

— Σ est l'alphabet ;

— $P : \Sigma \to \Sigma^*$ est une règle de production associant un mot à chaque lettre.

Le paramètre m va jouer un rôle crucial, ainsi un système de paramètre m est appelé un m-système. Un calcul fonctionne de la façon suivante : à partir d'un mot initial (« l'entrée »), on modifie le mot à chaque étape en utilisant la règle de production. Le système s'arrête lorsque la taille du mot devient strictement plus petite que m. Comme pour une machine de Turing, il est possible pour un système de ne jamais s'arrêter. Plus précisémment, à chaque étape :

— on regarde la première lettre $a \in \Sigma$ du mot ;

— on supprime m lettres au début du mot ;

— on ajoute $P(a)$ à la fin du mot.

Si un mot w se transforme en w' après une étape de calcul, on note $w \vdash w'$. Si w se transforme en w' après k étapes de calculs, on note $w \vdash^k w'$. S'il existe k tel que w se transforme en w' après k étapes de calculs, on note $w \vdash^* w'$. Enfin on note ε le mot vide.

1. En anglais « tag system ».

Voici un exemple de 3-système sur l'alphabet $\Sigma = \{a, b\}$ avec les règles de transformation $P(a) = aa$ et $P(b) = bbab$ et partant de l'entrée $aaaba$. Pour simplifier la lecture, on a mis en gras les 3 lettres effacées à chaque étape.

$$\mathbf{aaa}ba \vdash \mathbf{baa}a \vdash \mathbf{abb}ab \vdash \mathbf{aba}a \vdash \mathbf{aaa} \vdash aa \quad .$$

Le problème de l'arrêt pour les m-systèmes consiste à déterminer, étant donnés la description d'un m-système et un mot w, s'il existe $k \in \mathbb{N}$ tel que $w \vdash^k w'$ avec $|w'| < m$. Le but de cet exercice est de montrer que le problème de l'arrêt pour les m-systèmes est décidable pour $m = 1$ et indécidable pour $m \geq 2$.

4.1 Indécidabilité des 2-systèmes

Nous allons maintenant montrer que l'on peut simuler une machine de Turing par un 2-système. Pour cela, on considère des machines à deux compteurs a et b. Initialement, $b = 0$ et $a \in \mathbb{N}$ est l'entrée. Une telle machine comporte un nombre fini L d'instructions. Pour $i \in \{1, \ldots, L\}$, l'instruction i est de l'une des formes

1. $\mathsf{Incr}(c, j)$ qui incrémente $c \in \{a, b\}$ puis va à l'instruction $j \neq i$;
2. $\mathsf{Decr}(c, j)$ qui décrémente $c \in \{a, b\}$, puis va à l'instruction $j \neq i$; **cette instruction ne peut être utilisée que lorsque** $c > 0$;
3. $\mathsf{IsZero}(c, j, k)$ qui teste si $c \in \{a, b\}$ est nul, va à l'instruction $j \neq i$ si c'est le cas, et à l'instruction $k \neq i$ sinon ;
4. Halt qui arrête le calcul.

Pour chaque instruction $i \in \{1, \ldots, L\}$, on introduit les lettres $a_i, b_i, A_i, B_i, a'_i, b'_i, A'_i, B'_i, Z_i$ et on ajoute aussi le symbole $*$. L'alphabet est donc

$$\Sigma = \{*\} \cup \{a_i, b_i, A_i, B_i, a'_i, b'_i, A'_i, B'_i, Z_i : i = 1, \ldots, L\}.$$

Étant donné un état (a, b, i) d'une machine à compteur, un encodage valide de cet état est n'importe quel mot de la forme

$$\overbrace{\mathsf{a}_i? \cdots \mathsf{a}_i?}^{2^a \text{ fois}} A_i? \overbrace{\mathsf{b}_i? \cdots \mathsf{b}_i?}^{2^b \text{ fois}} B_i?$$

où chaque ? peut-être remplacé par une lettre quelconque. Par exemple, le mot

$$\underbrace{a_i*}_{a_i?}\,\underbrace{a_ia_i}_{a_i?}\,\underbrace{a_ia_j}_{a_i?}\,\underbrace{a_iB_j}_{a_i?}\,\underbrace{A_iA_i}_{A_i?}\,\underbrace{b_i*}_{b_i?}\,\underbrace{b_ib_j}_{b_i?}\,\underbrace{B_i*}_{B_i?}$$

est un encodage valide de $(2, 1, i)$.

Question 4.1. *Montrer que si l'instruction i est $\mathsf{Incr}(a, j)$ alors les règles $P(a_i) = a_j*a_j*$, $P(A_i) = A_j*$, $P(b_i) = b_j*$ et $P(B_i) = B_j*$ sont correctes : tous les encodages valides de l'état (a, b, i) se transforment (en plusieurs étapes) en un encodage valide de $(a + 1, b, j)$.*

Question 4.2. *Proposer une règle pour les instructions Halt qui produit un mot vide.*

Question 4.3. *Proposer une règle pour les instructions $\mathsf{Decr}(c, j)$, on rappelle que l'on suppose que $c > 0$. Quel problème se pose lorsque $c = 0$? [Indication : souvenez-vous que « ? » peut être n'importe quelle lettre dans l'encodage.]*

On suppose que l'instruction i est $\mathsf{IsZero}(b, j, k)$ et on introduit les règles

$$\begin{array}{llllll} P(a_i) = a'_i* & , & P(A_i) = A'_i* & , & P(b_i) = b'_i & , & P(B_i) = B'_i Z_i & , \\ P(a'_i) = a_j a_k & , & P(A'_i) = A_j A_k & , & P(b'_i) = b_j b_k & , & P(B'_i) = B_j* & , \\ P(Z_i) = b_k B_k* & . & & & & & & \end{array}$$

Question 4.4. *Montrer que si l'état est un encodage de (a, b, i) avec $b > 0$ alors les règles ci-dessus produisent bien un encodage de (a, b, j).*

Question 4.5. *Montrer que si l'état est en encodage de $(a, 0, i)$ alors les règles ci-dessus produisent bien un encodage de $(a, 0, k)$. [Indication : remarquer que les règles produisent un mot de longueur impaire, ce qui change l'interprétation de l'encodage.]*

Question 4.6. *Conclure.*

4.2 Décidabilité des 1-systèmes

Considérons un 1-système sur l'alphabet Σ avec une fonction de production P. Pour chaque mot $w \in \Sigma^*$ et lettre $x \in \Sigma$, on note $|w|_x$ le nombre de fois où la lettre x apparait dans le mot w. Par exemple $|aabcac|_a = 3$. Afin de simplifier les notations, on suppose que $\Sigma = \{a_1, \ldots, a_n\}$. Soit $f : \Sigma^* \to \mathbb{N}^n$ la fonction définie par $f(w) = (|w|_{a_1}, \ldots, |w|_{a_n})$. Par exemple si $\Sigma = \{a, b, c\}$, alors $f(w) = (|w|_a, |w|_b, |w|_c)$ et donc $f(aabcac) = (3, 1, 2)$.

Question 4.7. *Montrer qu'il existe une fonction linéaire $g : \mathbb{N}^n \to \mathbb{N}^n$, qui ne dépend que de P, telle que pour tout mot w, si $w \vdash^{|w|} w'$ alors $f(w') = g(f(w))$.*

On note $B = \{0, 1\}$ l'ensemble des booléens, muni des opérations usuelles (conjonction, disjonction, négation). Soit $\pi : \mathbb{N} \to B$ la projection définie par $\pi(n) = 1$ si $n > 0$ et $\pi(0) = 0$. On étend π aux vecteurs composante par composante, ainsi $\pi : \mathbb{N}^n \to B^n$ satisfait $\pi(v_1, \ldots, v_n) = (\pi(v_1), \cdots, \pi(v_n))$. Par exemple, $\pi(4, 0, 1) = (1, 0, 1)$.

Question 4.8. *Montrer qu'il existe une fonction $h : B^n \to B^n$, telle que pour tout $v \in \mathbb{N}^n$, $\pi(g(v)) = h(\pi(v))$. [Indication : étudier $\pi(n + m)$ et $\pi(nm)$ pour $n, m \in \mathbb{N}$, puis utiliser la linéarité de g.]*

Question 4.9. *Montrer que le problème de l'arrêt des 1-systèmes est décidable.*