Oral d'informatique

Concours d'admission en troisième année à l'ENS de Cachan

1-2 juin 2006

Préparation : 45 minutes. Interrogation : 45 minutes.

1 Problèmes de mots

Soit Σ un alphabet fini. On note Σ^* le *monoïde libre* engendré par Σ, et ε le *mot vide*. On considère un *système de réécriture* sur Σ, c'est à-dire un ensemble fini $R \subset \Sigma^* \times \Sigma^*$.

Si $(x, y) \in R$ et $u, v \in \Sigma^*$, on écrit $uxv \to_R uyv$ (*réduction en une étape*). Si $x \to_R y$ ou $y \to_R x$, on écrit $x \leftrightarrow_R y$. Enfin, on note $\to_R^*$ la *clôture réflexive transitive* de $\to_R$, et $\leftrightarrow_R^*$ celle de $\leftrightarrow_R$.

Décrire la congruence $\leftrightarrow_R^*$ dans chacun des cas suivants :

1. $\Sigma = \{a\}$ et $R = \{(aa, \varepsilon)\}$;
2. $\Sigma = \{a, b\}$ et $R = \{(ba, ab)\}$;
3. $\Sigma = \{a, b\}$ et $R = \{(ab, \varepsilon), (ba, \varepsilon)\}$.

On s'intéresse aux problèmes suivants, pour Σ et R quelconques (finis) :

1. Etant donnés $x, y \in \Sigma^*$, a-t-on $x \to_R y$? (*réduction en une étape*)
2. Etant donnés $x, y \in \Sigma^*$, a-t-on $x \to_R^* y$? (*réduction*)
3. Etant donnés $x, y \in \Sigma^*$, a-t-on $x \leftrightarrow_R^* y$? (*congruence*)

Ces problèmes sont-ils décidables ? semi-décidables ?

Indication : on pourra coder le *problème d'arrêt sur piles vides* pour une *machine déterministe à 2 piles et à 2 symboles*. Une telle machine est donnée par un ensemble fini d'états numérotés de 0 à m (0 pour l'*état initial* et m pour l'*état final*), avec pour chaque état $i \neq m$:

– soit l'instruction d'empilement d'un symbole (à gauche ou à droite) et un *état suivant* j ;
– soit l'instruction de dépilement (à gauche ou à droite) et un *branchement* j, k (selon le symbole lu).

On dit qu'un mot x à 2 symboles est *accepté par la machine* si, en partant de l'état 0 avec le mot x dans la pile gauche et rien dans la pile droite, la machine finit par s'arrêter dans l'état m avec les 2 piles vides.

Bien entendu, si la machine exécute une instruction de dépilement alors que la pile correspondante est vide, le calcul échoue et le mot n'est pas accepté. On admettra l'existence d'une machine déterministe à 2 piles et à 2 symboles pour laquelle l'ensemble des mots acceptés n'est pas récursif.

Préparation : 45 minutes. Interrogation : 45 minutes.

2 Calcul des prédicats

Une *machine déterministe à 2 registres* est donnée par un ensemble fini d'états numérotés de 0 à m (0 pour l'*état initial* et m pour l'*état final*), avec pour chaque état $i \neq m$:

– soit une *instruction d'incrémentation* et un *état suivant* j ;
– soit une *instruction de test à 0 avec décrémentation* et un *branchement* j, k (selon le résultat du test).

Les registres contiennent des entiers positifs ou nuls. La décrémentation n'est donc pas appliquée lorsque le registre est nul.

Construire une machine à 2 registres pour chacune des opérations suivantes (sur $\mathbb{N}$) :

1. l'addition : $(p, q) \mapsto p + q$;
2. la multiplication par 2 : $p \mapsto 2p$;
3. la division euclidienne par 2 : $p \mapsto (p \div 2, p \bmod 2)$.

On pourra s'inspirer de la syntaxe des langages impératifs pour écrire les instructions.

On considère le langage du premier ordre constitué d'un symbole de fonction unaire $\mathbf{S}$, d'un symbole de constante $\mathbf{0}$, et d'un symbole de relation binaire $\mathbf{R}_i$ pour chaque état i de la machine. Si $p \in \mathbb{N}$, on écrit p comme abréviation pour le terme $\mathbf{S}^p\mathbf{0}$.

On fixe $p_0, q_0 \in \mathbb{N}$. On considère le modèle $\mathcal{M}$ constitué de l'ensemble $\mathbb{N}$ avec l'interprétation standard de $\mathbf{S}$ et $\mathbf{0}$, le prédicat $\mathbf{R}_i$ étant défini par $p\,\mathbf{R}_i\,q$ si, en partant de l'état 0 avec p_0 et q_0 dans les registres, on peut arriver dans l'état i avec p et q dans les registres.

Exprimer chaque instruction de la machine comme une formule ϕ_i de telle sorte que la formule $\phi_0 \wedge \phi_1 \wedge \cdots \wedge \phi_{m-1} \wedge p_0\,\mathbf{R}_0\,q_0 \Rightarrow p\,\mathbf{R}_i\,q$ est prouvable si et seulement si $p\,\mathbf{R}_i\,q$ dans le modèle $\mathcal{M}$.

On pourra utiliser les règles de la déduction naturelle.

Montrer que le problème d'arrêt des machines à 2 registres se réduit au problème de prouvabilité en calcul des prédicats.

Peut-on coder de même l'arrêt des machines de Turing dans le calcul des prédicats ?

La prouvabilité est-elle décidable ? semi-décidable ?

Préparation : 45 minutes. Interrogation : 45 minutes.

3 Langage de l'arithmétique

On considère le langage du premier ordre $\mathcal{L}$ constitué des deux symboles de fonctions binaires $+$ et $\times$ et du symbole de relation binaire $=$.

Notez qu'il n'y a aucun symbole de constante dans ce langage. Dans le modèle standard $\mathbb{N}$, on peut toutefois exprimer le prédicat $x = 0$ par la formule $x + x = x$, ou encore par la formule $\forall y.\ x \times y = x$.

En utilisant les connecteurs logiques et les quantificateurs, exprimer chacun des prédicats suivants dans le langage $\mathcal{L}$ (pour le modèle standard $\mathbb{N}$) :

1. $x \neq 0$;
2. $x = 1$;
3. $x = n$ (pour $n \in \mathbb{N}$ donné) ;
4. $x \leq y$;
5. $x < y$;
6. x divise y ;
7. $x \neq 0$ et z est le reste de la division de y par x ;
8. x est un nombre premier (on ne considère pas 1 comme un nombre premier) ;
9. x et y sont des nombres premiers consécutifs ;
10. x est une puissance de 2.

Quels sont ceux qui peuvent s'exprimer en n'utilisant que l'addition (ou la multiplication) ?

Remarque : il n'est pas nécessaire d'écrire explicitement la formule dans chaque cas. Il suffit d'expliquer comment on la construit.

Exprimer le prédicat $2^x = y$ dans le langage $\mathcal{L}$.

Indication : on pourra coder les suites finies d'entiers en utilisant le *lemme chinois*. Plus précisément, on admet que si $p_1, \ldots, p_n$ sont des nombres premiers distincts et si $a_1, \ldots, a_n$ sont des entiers naturels tels que $a_1 < p_1, \ldots, a_n < p_n$, alors il existe a tel que $a_1 = a \bmod p_1, \ldots, a_n = a \bmod p_n$.

Les prédicats récursifs sont-ils exprimables dans le langage $\mathcal{L}$?

Préparation : 45 minutes. Interrogation : 45 minutes.

4 Théorie de la réécriture

Soit Σ un alphabet fini. On note Σ^* le *monoïde libre* engendré par Σ, et ε le *mot vide*. On considère un *système de réécriture* sur Σ, c'est à-dire un ensemble fini $R \subset \Sigma^* \times \Sigma^*$.

Si $(x, y) \in R$ et $u, v \in \Sigma^*$, on écrit $uxv \to_R uyv$ (*réduction en une étape*). On note $\to_R^*$ la *clôture réflexive transitive* de $\to_R$. On dit qu'un mot $x \in \Sigma^*$ est *réduit* s'il n'existe aucun mot y tel que $x \to_R y$. On supposera que ε est réduit. On note $\mathcal{L}$ l'ensemble de tous les mots réduits.

On considère les exemples suivants :

1. $\Sigma = \{a\}$ et $R = \{(aa, \varepsilon)\}$;
2. $\Sigma = \{a, b\}$ et $R = \{(ab, \varepsilon), (ba, \varepsilon)\}$;
3. $\Sigma = \{a, b\}$ et $R = \{(bab, aba)\}$;
4. $\Sigma = \{a, b\}$ et $R = \{(aa, \varepsilon), (bb, \varepsilon), (bab, aba)\}$.

Décrire $\mathcal{L}$ dans chacun de ces exemples.

En général, que peut-on dire du langage $\mathcal{L}$?

On s'intéresse aux propriétés suivantes :

1. *terminaison* : il n'existe aucune chaîne infinie $x_0 \to_R x_1 \to_R x_2 \to_R \cdots \to_R x_n \to_R \cdots$;
2. *confluence locale* : si $x \to_R y$ et $x \to_R z$, alors il existe t tel que $y \to_R^* t$ et $z \to_R^* t$;
3. *confluence* : si $x \to_R^* y$ et $x \to_R^* z$, alors il existe t tel que $y \to_R^* t$ et $z \to_R^* t$;
4. *existence et unicité de la forme réduite* : pour tout x, il existe un unique y réduit tel que $x \to_R^* y$.

Montrer que la terminaison et la confluence impliquent l'existence et l'unicité de la forme réduite.

Dans ce cas, la forme réduite de x est-elle calculable ?

Montrer que la terminaison et la confluence locale impliquent la confluence.

Indication : supposer qu'il existe un x pour lequel la confluence n'est pas vérifiée et construire une chaîne infinie $x = x_0 \to_R x_1 \to_R x_2 \to_R \cdots \to_R x_n \to_R \cdots$

Etudier les propriétés de terminaison et de confluence dans les exemples ci-dessus.

Trouver un critère simple de confluence locale.

Indication : remarquer que la propriété est évidente dans le cas où les réductions $x \to_R y$ et $x \to_R z$ portent sur des sous-mots disjoints.

En déduire que si la propriété de terminaison est satisfaite, alors on peut décider si la propriété de confluence est satisfaite.

Préparation : 45 minutes. Interrogation : 45 minutes.

5 Machines à registres

Une *machine déterministe à n registres* est donnée par un ensemble fini d'états numérotés de 0 à m (0 pour l'*état initial* et m pour l'*état final*), avec pour chaque état $i \neq m$:

– soit une *instruction d'incrémentation* et un *état suivant* j ;
– soit une *instruction de test à 0 avec décrémentation* et un *branchement* j, k (selon le résultat du test).

Chaque instruction concerne l'un des n registres. Les registres contiennent des entiers positifs ou nuls. La décrémentation n'est donc pas appliquée lorsque le registre est nul.

On dit qu'une application (c'est-à-dire une fonction partout définie) $f : \mathbb{N} \to \mathbb{N}$ est calculable par cette machine si, en partant de l'état 0 avec la valeur x dans le premier registre, la machine finit par s'arrêter dans l'état m avec la valeur $f(x)$ dans ce même registre.

Montrer que les applications suivantes sont calculables par une machine à 1 registre :

$$f_1(x) = x + 1, \qquad f_2(x) = \max(x - 1, 0), \qquad f_3(x) = x \bmod 2.$$

On pourra s'inspirer de la syntaxe des langages impératifs pour écrire les instructions.

Montrer que l'application $f : \mathbb{N} \to \mathbb{N}$ est calculable par une machine à 1 registre si et seulement si l'une des deux conditions suivantes est satisfaite :

– soit il existe $p \in \mathbb{Z}$ tel que $f(x) = x + p$ pour tout x assez grand (*cas affine*) ;
– soit il existe $p > 0$ tel que $f(x + p) = f(x)$ pour tout x assez grand (*cas périodique*).

Indication : si le calcul est assez long, on doit passer deux fois par le même état.

Remarque : il n'est pas nécessaire de détailler la preuve. Il suffit de donner l'idée générale.

Montrer que les applications suivantes sont calculables par une machine à 2 registres :

$$g_1(x) = 2x, \qquad g_2(x) = x \div 2, \qquad g_3(x) = \begin{cases} x/2 & \text{si } x \text{ est pair,} \\ x & \text{si } x \text{ est impair.} \end{cases}$$

Sont-elles calculables par une machine à 1 registre ?

Montrer qu'une machine à n registres peut être simulée par une machine à 2 registres.

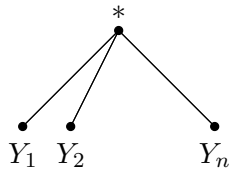
Indication : utiliser la décomposition en nombres premiers.

Préparation : 45 minutes. Interrogation : 45 minutes.

6 Coloriages

Un *graphe colorié* est un graphe G dont tous les sommets sont coloriés, et tel que 2 sommets reliés par une arête ne portent jamais la même couleur. Dans la suite, on partira de *graphes partiellement coloriés*, c'est-à-dire dont certains sommets sont déjà coloriés, et on se limitera aux coloriages qui étendent ce coloriage partiel. On n'aura droit qu'à 3 couleurs : les 2 booléens 0, 1, et une couleur auxiliaire $*$.

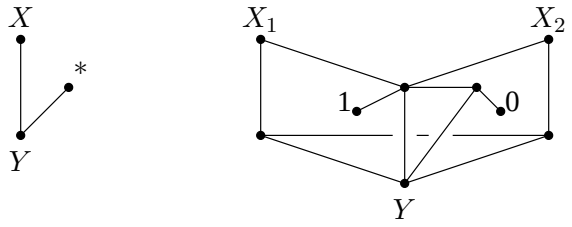
Donner le nombre de coloriages du graphe G_0 suivant, dont le sommet supérieur est déjà colorié :



Soit G un graphe partiellement colorié avec n *sommets d'entrée* $X_1, \ldots, X_n$ et 1 *sommet de sortie* Y. On dira que G *représente* l'opération booléenne $f : \mathbb{B}^n \to \mathbb{B}$ si les conditions suivantes sont réalisées :

1. Pour tout $(x_1, \ldots, x_n) \in \mathbb{B}^n$, il existe au moins un coloriage de G tel que chaque sommet d'entrée X_i porte la couleur x_i ;
2. Pour tout coloriage de G tel que chaque sommet d'entrée X_i porte la couleur $x_i \in \mathbb{B}$, le sommet de sortie Y porte nécessairement la couleur $f(x_1, \ldots, x_n)$.

Montrer que les deux graphes suivants représentent des opérations booléennes bien connues :



Montrer que pour toute formule propositionnelle A, on peut construire (en temps polynomial) un graphe partiellement colorié $G(A)$ tel que A est satisfaisable si et seulement si $G(A)$ est coloriable.

On se contentera d'expliquer le principe de la construction, et à titre d'exemple, on dessinera les graphes $G(p \wedge \neg q)$ et $G(p \wedge \neg p)$.

Montrer que le problème du coloriage des graphes partiellement coloriés se ramène au problème du coloriage des graphes.

Que peut-on en déduire sur la complexité algorithmique de ces problèmes ?