Corrigé de l'oral d'informatique

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

1-2 juin 2006

1 Problèmes de mots

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

1. $a^p \leftrightarrow_R^* a^q$ lorsque p et q ont la même parité ;
2. $u \leftrightarrow_R^* v$ lorsque les mots u et v ont les mêmes nombres d'occurrences de a et de b ;
3. $u \leftrightarrow_R^* v$ lorsque la différence entre ces deux nombres est la même pour les mots u et v.

Remarque : ces trois exemples correspondent à des présentations de monoïdes pour $\mathbb{Z}/2\mathbb{Z}$, $\mathbb{N} \times \mathbb{N}$, et $\mathbb{Z}$.

Ces problèmes sont-ils décidables ? semi-décidables ?

1. Le problème de la réduction en une étape est trivialement décidable.
2. Le problème de la réduction est trivialement semi-décidable, mais il n'est pas décidable.
3. Idem pour le problème de la congruence.

Pour montrer l'indécidabilité du problème de la réduction, on code le problème d'arrêt sur piles vides pour une machine déterministe à 2 piles et à 2 symboles. Etant donnée une telle machine pour laquelle ce problème est indécidable, on considère l'alphabet Σ formé des symboles $a, b, c_0, \ldots, c_m$.

On représente :

– la pile de gauche par un mot $x \in \{a, b\}^*$, le sommet de la pile étant à droite ;
– la pile de droite par un mot $y \in \{a, b\}^*$, le sommet de la pile étant à gauche ;
– une configuration de la machine par un mot de la forme xc_iy, où i représente l'état, et x, y les piles.

On introduit une règle de réécriture par instruction d'empilement, et deux par instruction de dépilement :

– $c_i \to ac_j$ ou $c_i \to bc_j$ pour l'empilement à gauche ;
– $c_i \to c_ja$ ou $c_i \to c_jb$ pour l'empilement à droite.
– $ac_i \to c_j$ et $bc_i \to c_k$ pour le dépilement à gauche ;
– $c_ia \to c_j$ et $c_ib \to c_k$ pour le dépilement à droite.

Soit R l'ensemble de ces règles. Clairement, le mot x est accepté par la machine si et seulement si on a $xc_0 \to_R^* c_m$. Ainsi, le problème de réduction est indécidable pour le système Σ, R.

De plus, comme la machine est déterministe, on montre que si u et v sont deux mots de la forme xc_iy, on a $u \leftrightarrow_R^* v$ si et seulement si il existe w, également de la forme xc_iy, tel que $u \to_R^* w$ et $v \to_R^* w$. Comme c_m est irréductible, on a donc $xc_0 \leftrightarrow_R^* c_m$ si et seulement si $xc_0 \to_R^* c_m$. Ainsi, le problème de la congruence est lui aussi indécidable.

2 Calcul des prédicats

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

Voici par exemple la machine pour la division euclidienne par 2 (le registre q doit être initialisé à 0) :

0. si $p = 0$, aller à 4, sinon décrémenter p et aller à 1 ;
1. si $p = 0$, aller à 3, sinon décrémenter p et aller à 2 ;
2. incrémenter q et aller à 0 ;
3. incrémenter p et aller à 4 ;
4. le quotient est dans le registre q et le reste dans le registre p.

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 $\mathcal{M}$.

Définition des formules ϕ_i :

– incrémentation à droite : $\phi_i = \forall x.\, \forall y.\, (x \,\mathbf{R}_i\, y \Rightarrow x \,\mathbf{R}_j\, \mathbf{S}y)$;
– décrémentation à droite : $\phi_i = \forall x.\, ((x \,\mathbf{R}_i\, \mathbf{0} \Rightarrow x \,\mathbf{R}_j\, \mathbf{0}) \wedge \forall y.\, (x \,\mathbf{R}_i\, \mathbf{S}y \Rightarrow x \,\mathbf{R}_k\, y))$;
– idem pour l'incrémentation ou la décrémentation à gauche.

Par exemple, si on considère la machine ci-dessus, on obtient les quatre formules suivantes :

$$\phi_0 = \forall y.\, ((\mathbf{0} \,\mathbf{R}_0\, y \Rightarrow \mathbf{0} \,\mathbf{R}_4\, y) \wedge \forall x.\, (\mathbf{S}x \,\mathbf{R}_0\, y \Rightarrow x \,\mathbf{R}_1\, y)),$$

$$\phi_1 = \forall y.\, ((\mathbf{0} \,\mathbf{R}_1\, y \Rightarrow \mathbf{0} \,\mathbf{R}_3\, y) \wedge \forall x.\, (\mathbf{S}x \,\mathbf{R}_1\, y \Rightarrow x \,\mathbf{R}_2\, y)),$$

$$\phi_2 = \forall x.\, \forall y.\, (x \,\mathbf{R}_2\, y \Rightarrow x \,\mathbf{R}_0\, \mathbf{S}y),$$

$$\phi_3 = \forall x.\, \forall y.\, (x \,\mathbf{R}_3\, y \Rightarrow \mathbf{S}x \,\mathbf{R}_4\, y).$$

Par construction, chacune des formules ϕ_i est satisfaite par le modèle $\mathcal{M}$, ainsi que la formule $p_0 \,\mathbf{R}_0\, q_0$. Si 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, alors elle est satisfaite par le modèle $\mathcal{M}$, donc on a $p \,\mathbf{R}_i\, q$ dans le modèle $\mathcal{M}$.

Réciproquement, si on a $p \,\mathbf{R}_i\, q$ dans le modèle $\mathcal{M}$, alors 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. Par récurrence sur la longueur de ce calcul, on peut construire une preuve de la formule $\phi_0 \wedge \phi_1 \wedge \cdots \wedge \phi_{m-1} \wedge p_0 \,\mathbf{R}_0\, q_0 \Rightarrow p \,\mathbf{R}_i\, q$.

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

De même, on montre que la formule $\exists x.\, \exists y.\, (\phi_0 \wedge \phi_1 \wedge \cdots \wedge \phi_{m-1} \wedge p_0 \,\mathbf{R}_0\, q_0 \Rightarrow x \,\mathbf{R}_m\, y)$ est prouvable si et seulement si elle est satisfaite par le modèle $\mathcal{M}$, c'est-à-dire si et seulement si la machine s'arrête.

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

On code l'arrêt d'une machine de Turing en utilisant par exemple un symbole de fonction unaire $\mathbf{S}^\alpha$ pour chaque symbole α de la machine, et un symbole de prédicat binaire $\mathbf{R}_i^\alpha$ pour chaque symbole α et pour chaque état i de la machine. Dans ce cas, les deux arguments de $\mathbf{R}_i^\alpha$ correspondent à la partie gauche et à la partie droite du ruban de la machine, le symbole lu étant α.

La prouvabilité est-elle décidable ? semi-décidable ?

La prouvabilité est semi-décidable, mais elle est indécidable, parce que l'arrêt des machine à registres (ou des machines de Turing) est indécidable.

3 Langage de l'arithmétique

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. $\neg\, x = 0$;
2. $\forall y.\ x \times y = y$;
3. $\exists y.\ (y = 1 \wedge x = y + \cdots + y)$;
4. $\exists z.\ x + z = y$;
5. $\neg\, y \leq x$;
6. $\exists z.\ x \times z = y$;
7. $z < x \wedge \exists t.\ y = x \times t + z$;
8. $\neg(x = 1) \wedge \forall y.\ \forall z.\ (x = y \times z \Rightarrow y = 1 \vee z = 1)$;
9. x est premier $\wedge$ y est premier $\wedge\ x < y \wedge \neg\, \exists z.\ (z$ est premier $\wedge\ x < z \wedge z < y)$;
10. $\forall y.\ (y$ est premier $\wedge\ y$ divise $x \Rightarrow y = 2)$.

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

On remarque que $x = 1$ peut aussi s'écrire $x \neq 0 \wedge \forall y.\ \forall z.\ (x = y + z \Rightarrow y = 0 \vee z = 0)$.

1, 2, 3, 4, 5 s'expriment en n'utilisant que l'addition, et 1, 2, 6, 8 en n'utilisant que la multiplication.

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

On écrit $p \lhd q$ si p et q sont des nombres premiers consécutifs. Si $p_0 \lhd p_1 \lhd \cdots \lhd p_n$, on associe à tout entier naturel a la suite finie $a \bmod p_0, a \bmod p_1, \ldots, a \bmod p_n$. Réciproquement, toute suite finie peut être obtenue de cette façon : il suffit d'appliquer le lemme chinois pour un p_0 assez grand.

Si on applique ce résultat à la suite arithmétique $0, 1, \ldots, n$ et à la suite géométrique $1, 2, \ldots, 2^n$ en utilisant la même suite $p_0 \lhd p_1 \lhd \cdots \lhd p_n$, on obtient a et b satisfaisant les formules suivantes :

$$a \bmod p_0 = 0, \qquad b \bmod p_0 = 1, \qquad a \bmod p_n = n, \qquad b \bmod p_n = 2^n,$$

$$\forall r.\ \forall s\ (p \leq r \wedge r \lhd s \wedge s \leq q \Rightarrow a \bmod s = (a \bmod r) + 1 \wedge b \bmod s = (b \bmod r) \times 2).$$

Soit $\phi(a, b, p, q, x, y)$ la conjonction de ces formules, où p_0, p_n, n sont remplacés par les variables p, q, x et 2^n par y. Alors le prédicat $2^x = y$ s'exprime ainsi :

$$\exists a.\ \exists b.\ \exists p.\ \exists q.\ (p \text{ est premier} \wedge q \text{ est premier} \wedge p \leq q \wedge \phi(a, b, p, q, x, y)).$$

D'après la première question, cette formule s'exprime dans le langage $\mathcal{L}$.

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

En utilisant ce principe, on exprime tous les prédicats primitifs récursifs, puis tous les prédicats récursifs.

4 Théorie de la réécriture

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

1. $\mathcal{L}$ est le langage fini $\varepsilon|a$;
2. $\mathcal{L}$ est le langage régulier $a^*|b^*$;
3. $\mathcal{L}$ est le langage régulier $(a^*bb^*aa)^*(a^*|a^*bb^*|a^*bb^*a)$;
4. $\mathcal{L}$ est le langage fini $\varepsilon|a|b|ab|ba|aba$.

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

$\mathcal{L}$ est l'ensemble de tous les mots qui n'ont pas comme sous-mot un membre gauche de règle. Comme il y a un nombre fini de règles, ce langage est toujours régulier.

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

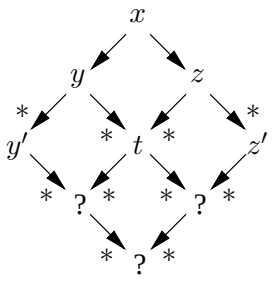
Pour tout x, il existe un y réduit tel que $x \to_R^* y$, car sinon, on pourrait construire une chaîne infinie $x = x_0 \to_R x_1 \to_R x_2 \to_R \cdots \to_R x_n \to_R \cdots$ De plus, si $x \to_R^* y$ et $x \to_R^* z$ avec y et z réduits, alors la confluence donne t tel que $y \to_R^* t$ et $z \to_R^* t$, d'où $y = t = z$.

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

Oui : il suffit d'appliquer les règles jusqu'à ce qu'on obtienne un mot réduit.

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

Soit x pour lequel la confluence n'est pas satisfaite. On a donc y', z' tels que $x \to_R^* y'$ et $x \to_R^* z'$, mais il n'existe aucun t' tel que $y' \to_R^* t'$ et $z' \to_R^* t'$. En particulier $y', z' \neq x$. On a donc y, z tels que $x \to_R y \to_R^* y'$ et $x \to_R z \to_R^* z'$. Par la confluence locale, on a t tel que $y \to_R^* t$ et $z \to_R^* t$. Alors l'un des mots y, z, t ne vérifie pas la confluence, car sinon, on pourrait construire t' :



On construit ainsi 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.

Les quatre exemples vérifient la propriété de terminaison :

– dans les deux premiers cas, la longueur d'un mot décroît à chaque étape de réduction ;
– dans les deux derniers cas, soit la longueur décroît, soit elle reste la même, et le mot devient plus petit pour l'ordre lexicographique.

Seul le troisième exemple n'est pas confluent, car on a $babab \to_R abaab$ et $babab \to_R baaba$, où les deux mots $abaab$ et $baaba$ sont réduits.

Trouver un critère simple de confluence locale.

Il suffit de tester la confluence locale dans le cas où les réductions $x \to_R y$ et $x \to_R z$ portent sur des sous-mots qui ne sont pas disjoints. Il n'y a qu'un nombre fini de configurations à tester, qui sont soit des chevauchements soit des inclusions de membres gauches (pics critiques).

Dans les exemples ci-dessus, il n'y a que des chevauchements. Par exemple aaa, bbb, $bbab$, $babb$ et $babab$ pour le dernier cas.

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

Si la propriété de terminaison est satisfaite, il suffit de tester la confluence locale pour un nombre fini de pics critiques. Etant donnés x, y, z tels que $x \to_R y$ et $x \to_R z$, on teste la confluence en calculant une forme réduite y' de y et une forme réduite z' de z, puis en vérifiant que $y' = z'$.

5 Machines à registres

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

Voici par exemple la machine pour la fonction $f_3(x) = x \bmod 2$:

0. si $x = 0$ aller à 3, sinon décrémenter x et aller à 1 ;
1. si $x = 0$ aller à 2, sinon décrémenter x et aller à 0 ;
2. incrémenter x et aller à 3 ;
3. le résultat est dans le registre x.

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*).

On construit facilement une machine à 1 registre dans le cas affine (comme pour f_1 et f_2) et aussi dans le cas périodique (comme pour f_3). Réciproquement, considérons une machine déterministe à 1 registre. Pour un $x > m$ donné, il faut au moins m étapes pour arriver à $x = 0$. Il y a deux cas :

– soit la machine s'arrête en m étapes ou moins. Alors la suite d'instructions exécutées est la même pour tout $x > m$. On est donc dans le *cas affine* ;
– soit la machine repasse au moins deux fois par le même état. Si la valeur du registre augmente ou reste la même entre ces deux passages, alors la machine ne s'arrête jamais. Sinon, elle diminue de $p > 0$ et ce p est le même pour tout $x > m$. On est alors dans le *cas périodique*.

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

Voici par exemple la machine pour la fonction g_3 (le registre y doit être initialisé à 0) :

0. si $x = 0$, aller à 3, sinon décrémenter x et aller à 1 ;
1. si $x = 0$, aller à 5, sinon décrémenter x et aller à 2 ;
2. incrémenter y et aller à 0 ;
3. si $y = 0$, aller à 9, sinon décrémenter y et aller à 4 ;
4. incrémenter x et aller à 3 ;
5. incrémenter x et aller à 6 ;
6. si $y = 0$, aller à 9, sinon décrémenter y et aller à 7 ;
7. incrémenter x et aller à 8 ;
8. incrémenter x et aller à 6 ;
9. le résultat est dans le registre x.

Sont-elles calculables par une machine à 1 registre ?

Non, car aucune de ces fonctions ne satisfait le critère établi ci-dessus.

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

On utilise la décomposition en nombres premiers pour coder n registres sur un seul. Par exemple, les valeurs x, y, z sont codées par la valeur $2^x 3^y 5^z$. Les instructions d'incrémentation et de décrémentation deviennent alors des multiplications et des divisions par 2, 3 ou 5, qui se calculent avec un second registre.

6 Coloriages

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

Il y a 2^n coloriages possibles.

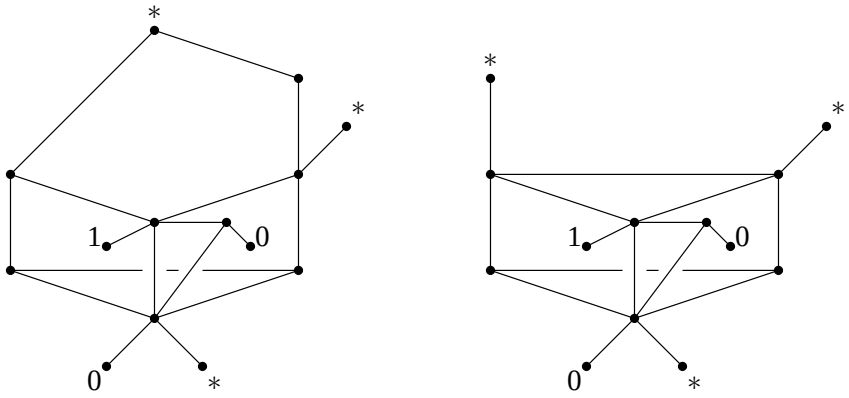
Montrer que les deux graphes suivants représentent des opérations booléennes bien connues :

On vérifie aisément que le premier graphe représente la négation, et le second représente la conjonction.

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 commence par G_0, où n est le nombre de variables propositionnelles de la formule A. On y attache un graphe représentant la fonction booléenne définie par A, construit en utilisant les graphes pour la négation et la conjonction. Finalement, on relie la sortie Y à deux sommets portant les couleurs 0 et $*$.

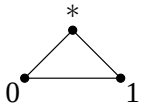
Voici par exemple les graphes $G(p \wedge \neg q)$ et $G(p \wedge \neg p)$:



Par construction, le graphe $G(A)$ est coloriable si et seulement si la formule A est satisfaisable.

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

Etant donné un graphe partiellement coloré G, on ajoute le triangle suivant :



Puis on relie chaque sommet de G portant une couleur aux deux sommets du triangle portant les autres couleurs. En supprimant toutes les couleurs, on obtient un graphe G' qui est coloriable si et seulement si le graphe partiellement coloré G est coloriable.

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

Comme le problème de la satisfaisabilité d'une formule propositionnelle est NP-complet, et comme les réductions définies ci-dessus sont polynomiales, on en déduit que le problème du coloriage des graphes est NP-complet.