Langages formels, calculabilité et complexité

Magistère MMFAI

Examen du 14 janvier 2003

durée 3 heures

Les documents sont autorisés.

Toutes les réponses devront être correctement justifiées.

Il va sans dire que la rigueur des raisonnements, la clarté des explications, mais aussi la qualité de la présentation influeront sensiblement sur la note.

1 Automates et modalités temporelles

On considère un alphabet Σ. Si $w \in \Sigma^*$ et $n \leq |w|$ on note $w[n]$ le préfixe de w de longueur n et on note $w[-n]$ le suffixe correspondant. On a donc $w = w[n]w[-n]$.

Soient $L_1,L_2 \subseteq \Sigma^*$. On suppose que $\mathcal{A}_1 = (Q_1,i_1,\delta_1,F_1)$ et $\mathcal{A}_2 = (Q_2,i_2,\delta_2,F_2)$ sont des automates déterministes qui reconnaissent les langages L_1 et L_2 respectivement.

On s'intéresse à l'ensemble des mots qui ont un préfixe dans L_2 tel que tous les préfixes suivants sont dans L_1. On appelle ce langage L_1 *depuis* L_2 et on le note $L_1 \mathsf{S}\, L_2$ (*since* en anglais). Formellement, on définit

$$L_1 \mathsf{S}\, L_2 = \{w \in \Sigma^* \mid \exists n \leq |w|,\, w[n] \in L_2 \text{ et } \forall n < m \leq |w|,\, w[m] \in L_1\}.$$

a) Donner un automate qui reconnaît le langage $L_1 \mathsf{S}\, L_2$ et prouver que l'automate reconnaît bien ce langage. Indication : on peut donner un automate déterministe ayant pour ensemble d'états $Q = Q_1 \times Q_2 \times \{0,1\}$.

On s'intéresse maintenant à l'ensemble des mots dont tous les suffixes sont dans L_1. On appelle ce langage *toujours* L_1 et on le note $\mathsf{G}\, L_1$ (*globally* en anglais). Formellement,

$$\mathsf{G}\, L_1 = \{w \in \Sigma^* \mid \forall n \leq |w|,\, w[-n] \in L_1\}.$$

b) Donner un automate qui reconnaît le langage $\mathsf{G}\, L_1$ et prouver que l'automate reconnaît bien ce langage. Indication : on peut donner un automate déterministe ayant pour ensemble d'états $Q = 2^{Q_1}$.

On s'intéresse finalement à l'ensemble des mots qui ont un suffixe dans L_2 tel que tous les suffixes précédents sont dans L_1. On appelle ce langage L_1 *jusqu'à* L_2 et on le note $L_1 \mathsf{U}\, L_2$ (*until* en anglais). Formellement, on définit

$$L_1 \mathsf{U}\, L_2 = \{w \in \Sigma^* \mid \exists n \leq |w|,\, w[-n] \in L_2 \text{ et } \forall 0 \leq m < n,\, w[-m] \in L_1\}.$$

c) Donner un automate qui reconnaît le langage $L_1 \mathsf{U} L_2$. On ne demande pas de prouver que l'automate reconnaît bien ce langage. Indication : on peut donner un automate non déterministe ayant pour ensemble d'états $Q = 2^{Q_1} \times Q_2 \times \{0,1\}$.

2 Machines de Turing et Langages algébriques

Soit $\mathcal{M} = (Q,q_0,q_f,\Sigma,\Gamma,\delta)$ une machine de Turing où $q_0 \in Q$ est l'état initial, $q_f \in Q$ est l'unique état acceptant, Σ est l'alphabet d'entrée, Γ est l'alphabet de bande et δ est la fonction de transition. Le symbole de bande *blanc* est noté $\Box$.

Une configuration de $\mathcal{M}$ est un mot de la forme uqv avec $q \in Q$, $u \in \Gamma^* \setminus \Box\Gamma^*$ et $v \in \Gamma^* \setminus \Gamma^*\Box$. La configuration initiale à partir de la donnée $w \in \Sigma^*$ est $q_0 w$. On note $C \to C'$ s'il existe une transition de $\mathcal{M}$ faisant passer de la configuration C à la configuration C'.

On suppose que $\# \notin \Gamma$ et on note $\widetilde{w}$ le miroir d'un mot w. On considère les langages $L_1 = \{C\#\widetilde{C'} \mid C \to C'\}$ et $L_2 = \{\widetilde{C}\#C' \mid C \to C'\}$.

a) Donner une grammaire qui engendre le langage $L_1 \cap \Gamma^2\Gamma^*Q\Gamma^*\Gamma^2\#\Gamma^*Q\Gamma^*$ et prouver que la grammaire convient, i.e., qu'elle engendre bien le langage ci-dessus.

b) Donner une grammaire qui engendre le langage $L_1 \cap \Gamma Q\Gamma^*\Gamma^2\#\Gamma^*Q\Gamma^*$. On ne demande pas de prouver que la grammaire convient.

c) Donner une grammaire qui engendre le langage $L_1 \cap Q\Gamma^*\Gamma^2\#\Gamma^*Q\Gamma^*$. On ne demande pas de prouver que la grammaire convient.

On peut démontrer ainsi que les langages L_1 et L_2 sont algébriques. On pourra utiliser ce fait dans la suite.

d) Montrer que les langages $K_1 = (L_1\#)^+\Gamma^*q_f\Gamma^*$ et $K_2 = q_0\Sigma^*(\#L_2)^+$ sont algébriques.

e) On suppose que tout calcul acceptant de $\mathcal{M}$ est de longueur paire. Montrer que $K_1 \cap K_2 \neq \emptyset$ si et seulement si $\mathcal{L}(\mathcal{M}) \neq \emptyset$ ($\mathcal{L}(\mathcal{M})$ est le langage accepté par $\mathcal{M}$).

f) Que peut-on en déduire sur le problème de la vacuité de l'intersection de deux langages algébriques ?

On considère les langages $M_1 = K_1 \cap q_0\Sigma^*(\#\Gamma^*Q\Gamma^*)^*$ et $M_2 = (\#L_2)^+$.

On rappelle que le quotient à droite de L par K est $LK^{-1} = \{u \mid \exists v \in K, uv \in L\}$.

g) Montrer que $q_0\mathcal{L}(\mathcal{M}) = M_1M_2^{-1} \cap q_0\Sigma^*$. Que peut-on en déduire ?

3 Décidabilité

On note PREM l'ensemble des mots sur l'alphabet $\{0,1,\#\}$ qui représentent une machine de Turing qui accepte tous les mots sur son alphabet d'entrée dont la longueur est un nombre premier. Formellement et avec les notations du cours,

$$\text{PREM} = \{c(M) \mid M \text{ accepte } w \in \Sigma_M^* \text{ ssi } |w| \text{ est premier}\}.$$

a) Prouver que le problème de l'arrêt sur bande vide (**HV**) se réduit à PREM. En déduire que PREM n'est pas le complémentaire d'un langage récursivement énumérable (PREM $\notin$ **co−RE**).

b) Montrer que PREM n'est pas récursivement énumérable (PREM $\notin$ **RE**).

4 Complexité : logique et graphes

Une formule du premier ordre sur les graphes est une formule du premier ordre utilisant uniquement deux prédicats binaires, l'égalité et $a(x,y)$ qui sera interprété comme l'existance d'une arête de x vers y. Formellement, on considère un ensemble dénombrable de variables $\mathcal{V}$ et un symbole de prédicat binaire a. $\mathrm{FO}(a)$ est l'ensemble des formules défini par la grammaire

$$\varphi ::= x = y \mid a(x,y) \mid \varphi \vee \varphi \mid \neg\varphi \mid \exists x\ \varphi$$

où $x,y \in \mathcal{V}$ sont des variables. On utilisera les abbréviations usuelles : $\forall x\ \varphi = \neg\exists x\ \neg\varphi$, $\varphi \wedge \psi = \neg(\neg\varphi \vee \neg\psi)$, $\varphi \longrightarrow \psi = \neg\varphi \vee \psi$, ...

Par exemple, $\varphi_1 = \forall x \forall y((\exists z\ a(x,z) \wedge a(z,y)) \rightarrow a(x,y))$ est une formule close de $\mathrm{FO}(a)$ et $\varphi_2(x,y) = \exists z\ a(x,z) \wedge a(z,y)$ est une formule de $\mathrm{FO}(a)$ ayant deux variables libres x et y.

Soient $G = (S,A)$ un graphe et $\sigma : \mathcal{V} \rightarrow S$ une assignation. On définit $G,\sigma \models \varphi$ par induction sur les formules $\varphi \in \mathrm{FO}(a)$ comme suit :

- $G,\sigma \models x = y$ si $\sigma(x) = \sigma(y)$,
- $G,\sigma \models a(x,y)$ si $(\sigma(x),\sigma(y)) \in A$,
- $G,\sigma \models \varphi \vee \psi$ si $G,\sigma \models \varphi$ ou $G,\sigma \models \psi$,
- $G,\sigma \models \neg\varphi$ si $G,\sigma \not\models \varphi$
- $G,\sigma \models \exists x\ \varphi$ s'il existe $s \in S$ tel que $G,\sigma' \models \varphi$ où $\sigma'(x) = s$ et $\sigma'(y) = \sigma(y)$ pour tout $y \neq x$.

Remarque : $G,\sigma \models \varphi$ ne dépend que des valeurs de σ sur les variables libres de φ.

Soit $\varphi \in \mathrm{FO}(a)$. On considère le problème des φ-graphes défini comme suit :

Données : Un graphe $G = (S,A)$ et une assignation $\sigma : V \rightarrow S$ définie sur l'ensemble V des variables libres de φ.

Question : $G,\sigma \models \varphi$?

On utilisera la représentation usuelle d'un graphe par le nombre de sommets n écrit en binaire sur une bande N et la matrice d'adjacence $a \in \{0,1\}^{n^2}$ sur une bande A. L'ensemble des sommets du graphe est $S = \{1,\ldots,n\}$. Pour chaque variable libre x_i de φ, l'assignation $\sigma(x_i) \in S$ sera représentée en binaire sur une bande B_i.
Remarque : la formule φ ne faisant pas partie des données, le nombre de bandes d'entrée est indépendant des données.

a) Supposons que l'on dispose d'une machine de Turing déterministe $\mathcal{M}_\varphi$ qui décide du problème des φ-graphes en espace logarithmique. Montrer que le problème des $(\exists x_k\ \varphi)$-graphes est décidable en espace logarithmique par une machine de Turing déterministe.

b) Montrer que pour toute formule $\varphi \in \mathrm{FO}(a)$, le problème des φ-graphes est décidable en espace logarithmique par une machine de Turing déterministe.