ANNÉE UNIVERSITAIRE 2009/2010
2ÈME SESSION DE PRINTEMPS

Parcours :	CSB6
Code UE :	INF253
Épreuve :	Informatique Théorique 2
Date :	21 Juin 2010
Heure :	14 H
Durée :	3H
Documents :	autorisés
Épreuve de :	Mr Géraud Sénizergues

L'exercice et le problème sont indépendants.
La note obtenue à l'examen, sur 20, est le minimum entre 20 et le total des points obtenus à l'exercice et au problème.

EXERCICE [10 points]

On considère un alphabet fini $X := \{a, b\}$. On rappelle que, pour tout langage $A \subseteq X^*$, $A^0 = \{\epsilon\}$ et $A^* = \bigcup_{j\geq 0} A^j$. Etant donnés deux langages H et K, on considère les langages suivants :

$$L_1 = H \cup K \quad ; \quad L_2 = H^* \quad ; \quad L_3 = (X^* - H)^* \quad ; \quad L_4 = H^{57};$$

$$L_5 = \bigcup_{j=0}^{57} H^j \quad ; \quad L_6 = \bigcup_{j\geq 57} H^j \quad ; \quad L_7 = \bigcup_{j\geq 0} H^{57\cdot j} \quad ; \quad L_8 = \bigcap_{j=0}^{57} H^j \quad ;$$

$$L_9 = \bigcap_{j\geq 57} H^j \quad ; \quad L_{10} = \{f \in X^* \mid f \in H \text{ ou } \mid f \mid\equiv 1(mod\ \ 5)\};$$

$$L_{11} = \{f \in X^* \mid f \in H \text{ ou } |f|_a = |f|_b\}; L_{12} = \{f \in X^* \mid f \in H \text{ et } \mid f \mid\equiv 1(mod\ \ 5)\};$$

$$L_{13} = \{f \in X^* \mid f \in H \text{ et } |f|_a = |f|_b\}; L_{14} = H \cdot \{f \in X^* \mid |f|_a = |f|_b\} \cdot K$$

$$L_{15} = \bigcup_{j=0}^{57} H^j \cdot K^j \quad ; \quad L_{16} = \bigcup_{j\geq 0} H^j \cdot K^j \quad ; \quad L_{17} = \bigcup_{j\geq 0} H^{j^2} \quad ; \quad L_{18} = \bigcap_{j\geq 1} H^{j^2}.$$

Pour chaque entier $i \in [1, 18]$:
1- déterminer si, pour tous langages rationnels $H, K \subseteq X^*$, L_i est encore rationnel (on justifiera chaque réponse par une preuve ou un contre-exemple) ;
2- déterminer si, pour tous langages algébriques $H, K \subseteq X^*$, L_i est encore algébrique (on justifiera chaque réponse par une preuve ou un contre-exemple).

PROBLÈME [20 points]

Soit F un ensemble de symboles et $\mathbf{ar} : F \to \mathbb{N}$ une application. Le nombre $\mathbf{ar}(f)$ est l'arité du symbole f. L'ensemble F, muni de $\mathbf{ar}$, est un alphabet *gradué*. On suppose, dans ce problème, que $\mathbf{ar}$ ne prend que les valeurs 2 et 0. Les symboles de F sont représentés par des caractères et les états des automates finis de terme sont représentés par des entiers compris entre 0 et une certaine constante N. Le symbole particulier $\bullet$ est représenté par le caractère $\#$. On définit un type `symbol` dont les valeurs sont tous les symboles de $F \cup \{\#\}$ union tous les entiers de 0 à N. On suppose connue une fonction `arite(f:symbol):int` qui renvoie, pour tout $f \in F \cup \{\#\} \cup [0, N]$, son arité (l'arité d'un élément de $\{\#\} \cup [0, N]$ vaut 0).

On représente les arbres binaires étiquetés sur $F \cup \{\#\} \cup [0, N]$ par le type de données :

```
type ARBREB=^noeud
type noeud=record
    fg: ARBREB;
    fd: ARBREB;
    etiq: symbol
end
```

1- Ecrire une fonction `TERME(t:ARBREB):boolean` qui teste si `t` est bien un terme sur l'alphabet gradué F.
On représente un automate fini de termes déterministe (aftd en abrégé) sur F par le type de données :

```
type AUT=record
    taille: 0..N;
    trans0: array[symbol] of 0..N;
    trans2: array[0..N,0..N, symbol] of 0..N;
    final: array[0..N] of boolean
end
```

Le champs `taille` mémorise le nombre d'états de l'automate, `trans0[f]` contient l'état atteint par l'automate après avoir lu le symbole `f`$\in F$ sur une feuille.
`trans2[i,j,f]` contient l'état atteint par l'automate après avoir lu l'état `i` sur le fils gauche, l'état `j` sur le fils droit et le symbole `f` $\in F$ à la racine.
Exemple :
Soit $\mathcal{B} = \langle F, Q_{\mathcal{B}}, Q_{\mathcal{B},+}, \delta_{\mathcal{B}} \rangle$ où $F = \{g, h, a\}, \mathbf{ar}(g) = \mathbf{ar}(h) = 2, \mathbf{ar}(a) = 0, Q_{\mathcal{B}} := \{0, 1, 2\}, Q_{\mathcal{B},+} := \{0\}$ et $\delta_{\mathcal{B}}$ est formé des transitions suivantes :

$$a \to 1, \quad g(0,0) \to 1, \quad g(0,1) \to 0, \quad g(1,0) \to 0, \quad g(1,1) \to 1$$

$$h(q,r) \to 2, \quad g(q,2) \to 2, \quad g(2,r) \to 2$$

pour tous $q, r \in Q_{\mathcal{B}}$. La structure représentant cet automate aura comme contenu : `trans0[a]`$=$ 1 les trois tableaux `trans2[*,*,g]`, `trans2[*,*,h]`, `final[*]` auront pour valeurs :

$i \setminus j$	0	1	2
0	1	0	2
1	0	1	2
2	2	2	2

$i \setminus j$	0	1	2
0	2	2	2
1	2	2	2
2	2	2	2

i	0	1	2
final[i]	1	0	0

2- Soit $\mathcal{C}$ l'aftd $\langle F, Q_{\mathcal{C}}, Q_{\mathcal{C},+}, \delta_{\mathcal{C}}\rangle$ où F est l'alphabet gradué de l'exemple précédent, $Q_{\mathcal{C}} := \{0,1,2,3\}, Q_{\mathcal{C},+} := \{0\}$ et $\delta_{\mathcal{C}}$ est formé des transitions suivantes :

$$a \rightarrow 1,\ \ g(0,0)\rightarrow 0,\ \ g(0,1)\rightarrow 1,\ \ g(1,0)\rightarrow 1,\ \ g(1,1)\rightarrow 0$$

$$h(0,0)\rightarrow 2,\ \ h(0,1)\rightarrow 1,\ \ h(1,0)\rightarrow 1,\ \ h(1,1)\rightarrow 0$$

$$g(q,2)\rightarrow 2,\ \ g(2,q)\rightarrow 2,\ \ h(q,2)\rightarrow 2,\ \ h(2,q)\rightarrow 2$$

$$g(r,3)\rightarrow 0,\ \ g(3,r)\rightarrow 0,\ \ h(r,3)\rightarrow 0,\ \ h(3,r)\rightarrow 0$$

pour tout $q \in [0,2], r \in [0,3]$. Décrire l'objet de type `AUT` qui représente cet aftd.
3- Ecrire une fonction `ETAT(t:ARBREB;A:AUT):0..N` qui renvoie l'état atteint par l'aftd `A` après lecture du terme `t` (si il existe une course de `A` sur `t`) et la constante `nil` (sinon).
4- Un état i de l'aftd $\mathcal{A}$ est dit *accessible* si et seulement si, il existe un terme $t \in \mathrm{T}(F)$, tel que $t \rightarrow^*_{\mathcal{A}}\ i$.
Ecrire une procedure `ACC(A:AUT;var QACCB: array[0..N] of boolean)` qui calcule les états accessibles de l'aftd `A` : pour tout $i \in [0, A \cdot taille - 1]$, elle place dans la case `QACCB[i]`
- le booléen `true`, si i est accessible
- le booléen `false`, si i n'est pas accessible.

5- Ecrire une procedure `ACC_temoin(A:AUT;var QACCA: array[0..N] of ARBREB)` qui place dans la case `QACCA[i]`
- un terme $t \in \mathrm{T}(F)$ tel que $t \rightarrow^*_{\mathcal{A}} i$, si i est accessible
- l'arbre `nil`, si i n'est pas accessible

6- Un état i de l'aftd $\mathcal{A}$ est dit *co-accessible* si et seulement si, il existe un contexte $C \in T(Q_{\mathcal{A}}, \{\bullet\})$, tel que

$$\exists j \in Q_{\mathcal{A},+}, C[i] \rightarrow^*_{\mathcal{A}} j$$

Ecrire une procédure `COACC(A:AUT;var QCOACCB: array[0..N] of boolean)` qui calcule les états co-accessibles de l'aftd `A`.
7- Ecrire une procedure `COACC_temoin(A:AUT;var QCOACCA: array[0..N] of ARBREB)` qui place dans la case `QCOACCA[i]`
- un contexte $C \in T(Q_{\mathcal{A}}, \{\bullet\})$ tel que $\exists j \in Q_{\mathcal{A},+}, C[i] \rightarrow^*_{\mathcal{A}} j$, si i est co-accessible
- l'arbre `nil`, si i n'est pas co-accessible.

8- Un aftd $\mathcal{A}$ est dit *émondé* lorsque tout état de $\mathcal{A}$ est à la fois accessible et co-accessible.
8.1- Donner un aftd émondé $\mathcal{C}'$ reconnaissant le même langage que l'aftd $\mathcal{C}$ de la question 2.
8.2- Ecrire une procedure `EMONDAGE(A:AUT;var AE:AUT)` qui affecte au paramètre `AE`, un aftd émondé, reconnaissant le même langage que `A`.
9- Etant donné un aftd $\mathcal{A}$, on définit le graphe orienté $G(\mathcal{A})$, qui a comme sommets les états de $\mathcal{A}$ et qui a un arc de i vers j si et seulement si, il existe un état r et un symbole $f \in F$ tels que

$$f(i,r) \rightarrow_{\mathcal{A}} j \text{ ou } f(r,i) \rightarrow_{\mathcal{A}} j.$$

10- Soit $\mathcal{A}$ un aftd émondé. Montrer que les assertions suivantes sont équivalentes :
(A1) L($\mathcal{A}$) est infini
(A2) Le graphe $G(A)$ a un circuit (i.e. un chemin de longueur ≥ 1 allant d'un sommet i au même sommet i)
(A3) il existe un état i de $\mathcal{A}$ et un contexte $C \in T(Q_{\mathcal{A}}, \{\bullet\})$ tels que,

$$\|C\| \geq 2 \text{ et } C[i] \rightarrow^*_{\mathcal{A}} i$$

(A4) il existe des contextes $B, C \in T(F, \{\bullet\})$ et un terme $d \in T(F)$ tels que,

$$\|C\| \geq 2 \text{ et } B \circ C^{*\bullet}[d] \subseteq \mathrm{L}(\mathcal{A})$$

11- Dans le cas de l'aftd $\mathcal{C}'$ fourni à la question 8.1. :
- construire le graphe $G(\mathcal{C}')$ et donner un circuit dans ce graphe
- donner un exemple de couple (i, C) vérifiant (A3)
- donner un exemple de triplet (B, C, d) vérifiant (A4)
12- Définir une structure de données pour représenter les graphes orientés (on pourra puiser dans les structures classiques vues dans l'UE d'ASD). Nous la noterons `GRAPHE` dans la suite.
13- Ecrire une fonction `AUT_to_GRAPHE(A:AUT;var G:GRAPHE)` qui, étant donné un aftd $\mathcal{A}$, affecte à `G` le graphe orienté $G(\mathcal{A})$.
14- Par quel algorithme classique peut-on décider si un graphe orienté G a un circuit ? (on ne demande pas les détails de cet algorithme mais seulement son nom et son principe général) Quelle est la complexité en temps, dans le cas le pire, de l'algorithme que vous avez choisi ?

On suppose maintenant que l'on a écrit une fonction `CIRCUITB(G:GRAPHE):boolean` qui détermine si un graphe orienté G possède un circuit (ou non).
15- Ecrire une fonction `INFINI(A:AUT):boolean` qui teste si un aftd reconnait un ensemble infini de termes.
16- On définit l'ensemble $\mathcal{C}_1$ comme l'ensemble des termes de hauteur 1 de la forme $f(r, \bullet)$ ou $f(\bullet, r)$ pour $f \in F, \mathbf{ar}(f) = 2, r \in Q_{\mathcal{A}}$.
Définir une structure de données pour représenter les graphes orientés étiquetés sur l'ensemble $\mathcal{C}_1$. Nous la noterons `GRAPHEE` dans la suite.
17- Ecrire une fonction `AUT_to_GRAPHEE(A:AUT;var G:GRAPHEE)` qui, étant donné un aftd $\mathcal{A}$, affecte à `G` un graphe $GE(\mathcal{A})$, obtenu à partir du graphe orienté (non-étiqueté) $G(\mathcal{A})$, en étiquetant tout arc (i, j) par un contexte $C \in \mathcal{C}_1$ tel que

$$C[i] \rightarrow_{\mathcal{A}} j.$$

On suppose que l'on a écrit une fonction `CIRCUITL(G:GRAPHEE):liste` qui renvoie
- une liste $i_0, i_1, \ldots, i_n$ de sommets (avec $n \geq 1$) formant un circuit dans `G`
- la liste `nil` si le graphe `G` n'a pas de circuit.

18- Ecrire une fonction `INFINI_temoinQ(A:AUT;var C:ARBREB;var i:0..N):boolean` qui teste si un aftd $\mathcal{A}$, que l'on suppose émondé, reconnait un ensemble infini de termes et, dans l'affirmative, fournit des objets i, C satisfaisant l'assertion (A3) de la question 10.
19- Expliquer comment on pourrait écrire une fonction
`INFINI_temoinF(A:AUT;var B,C, d:ARBREB):boolean` qui teste si un aftd $\mathcal{A}$, que l'on suppose émondé, reconnait un ensemble infini de termes et, dans l'affirmative, fournit des objets B, C, d satisfaisant l'assertion (A4) de la question 10.