Examen

Documents interdits

Durée: 2h

8 janvier 2013

Note importante : répondre aux questions suivantes de manière la plus lisible possible, en particulier, dans la question 6 où il vous est demandé d'écrire du code en langage C. Les parties illisibles ne seront pas lues !

1 Minimisation

Minimiser l'automate suivant :

		a	b
$\rightarrow$	0	1	4
	1	2	4
$\leftarrow$	2	3	1
	3	6	3
	4	5	1
$\leftarrow$	5	6	4
	6	3	6

2 Calcul de résiduels

Calculer les résiduels suivants en détaillant les différentes étapes :

Q.2.1. $(a+b)/b$

Q.2.2. $(a+b)^*/b$

Q.2.3. $(a+b)^*ba/b$

Q.2.4. $(a+b)^*ba/bba$

3 Automate des résiduels

Q.3.1. Calculer tous les résiduels du langage décrit par l'expression régulière $E = (a+b)^*b^*ab$.

Q.3.2. Combien d'états possède l'automate minimal reconnaissant le langage dénoté par E ? justifiez votre réponse.

Q.3.3. Construire l'automate minimal reconnaissant le langage dénoté par E en suivant la méthode des résiduels.

4 Grammaire hors-contexte

Q.4.1. Ecrire une grammaire hors-contexe G générant le langage $L = \{m \in \{a,b\}^* || m|_a = |m|_b\}$

Q.4.2. Ecrire l'arbre ou les arbres de dérivation que G associe au mot *abbaab*.

Q.4.3. G est elle ambiguë ? justifiez votre réponse.

5 Automate à pile

Q.5.1. Construire l'automate à pile A reconnaissant le langage L des mots sur $\{a,b\}^*$ contenant 2 fois plus de a que de b ou 3 fois plus de b que de a.

Q.5.2. A est il déterministe ? justifiez votre réponse.

Q.5.3. L peut il être reconnu par un automate à pile déterministe ? Si oui, construire un tel automate. Sinon, expliquer de façon convaincante pourquoi.

6 Question d'implémentation

Ecrire la fonction `af_mat *af2mat(af *a)` qui prend en entrée un automate représenté sous la forme d'une structure de type `af` et qui retourne le même automate représenté sous forme matricielle. Vous utiliserez pour cela la structure `af_mat` définie ci-dessous. Les structures `af_etat`, `af_arc` et `af` sont là pour vous rafraîchir la mémoire.

```
typedef struct{int ne;int ns;char *accept;int init;char ***delta;} af_mat;
typedef struct{int n; liste *trans; int is_accept;} af_etat;
typedef struct{int in; af_etat *orig; af_etat *dest;} af_arc;
typedef struct{liste *etats; af_etat *init;} af;
```

Les champs `ne` et `ns` indiquent respectivement le nombre d'états de l'automate et le nombre de symboles dans l'alphabet. Le tableau `accept` est de taille `ne`. Etant donnée l'entier `i` correspondant à un état, `accept[i]` vaut `1` si `i` est un état d'acceptation. Il vaut `0` sinon. Le champ `init` indique l'état initial. Le tableau `delta` représente la fonction de transition. S'il existe une transition étiquetée `e` allant de l'état `o` à l'état `d` alors `delta[o][e][d]` vaut `1`. Il vaut `0` sinon.
Vous considèrerez que vous disposez de la fonction suivante :
`af_mat *af_mat_allocate(int ne, int ns)`
qui permet d'allouer toute la mémoire nécessaire pour une structure de type `af_mat` (la fonction alloue en particulier la mémoire des tableaux `accept` et `delta`).