TP1

Présentation

Le but de ce TP est d'approfondir l'utilisation des types structurés et des matrices, via l'étude d'automates finis.

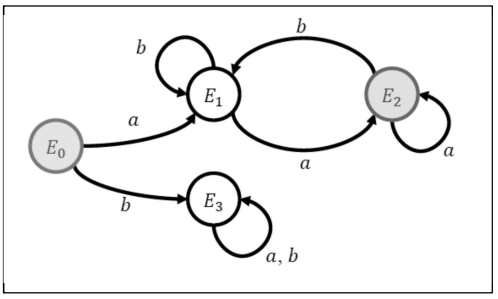
Définition

Un automate fini est un modèle mathématique d'ordinateur très simple, capable de reconnaître des *mots* formés de *lettres* contenus dans un *alphabet* prédéfini et respectant une certaine *structure*. On appelle *longueur* d'un mot le nombre de lettres qui le composent. On note le mot **ε** de longueur 0 appelé *mot vide*.
Exemple *:* avec l'alphabet A ={*a*,*b*}, on peut produire des mots composés de *a* et de *b*, tels que *b*, *aab*, *ababb*, etc.

Les automates les plus simples ont une sortie binaire : tout mot est soit *accepté* par l'automate, soit *refusé*. On peut représenter un automate par un *graphe orienté* :

- Chaque nœud s'appelle un *état* ;
- Chaque lien est un *changement d'état*, associé à une lettre de l'alphabet.

Exemple *:* soit l'automate représenté par le graphe suivant :



Cet automate comporte 4 états E_0, E_1, E_2 et E_3 :

- L'état E_0 est un état *initial*.
- L'état E_2 un état *acceptant*
- Les états E_0, E_1 et E_3 sont des états *refusant*.

Les liens de cet automate sont associés aux lettres ***a*** ou ***b***, ce qui signifie qu'il est défini pour l'alphabet {***a***, ***b***}.

En parcourant le graphe d'un automate depuis l'un de ses états initiaux, on peut décider quels sont les mots qu'il accepte. Ce parcours est réalisé de la façon suivante :

1. On démarre d'un état initial, qui devient l'état courant.
2. On considère chaque lettre constituant le mot, en commençant par la première. À chaque fois, on se déplace dans l'automate en suivant le lien associé à la lettre considérée.
3. Si le parcours se termine sur un état *acceptant*, alors le mot est accepté. Sinon, il est refusé.

Exemple *:* dans notre graphe d'exemple, on n'a qu'un seul état *initial* $\boldsymbol{E_0}$, donc tout parcours démarre obligatoirement de ce nœud. On n'a qu'un seul état *acceptant* $\boldsymbol{E_2}$, donc tout mot accepté se termine obligatoirement sur ce nœud.

- Quelques mots acceptés par cet automate, avec le parcours correspondant :
  - ***aaba*** : E_0,E_1,E_2,E_1,E_2
  - ***abbaba*** :$E_0,E_1,E_1,E_1,E_2,E_1,E_2$
- Quelques mots refusés par cet automate :
  - ***baa*** : E_0,E_3,E_3,E_3
  - ***abab*** : $E_0,E_1,E_2,E3,E_2$

On peut représenter un automate sous la forme d'une *matrice de transition*, qui indique les changements possibles pour chaque lettre, à partir de chaque état.
Exemple : pour l'automate précédent, on a la matrice de transition suivante :

	a	***b***
E_0	E_1	E_3
E_1	E_2	E_1
E_2	E_2	E_1
E_3	E_3	E_3

Travail demandé

Exercice 1 : construction d'un automate a états finis (l'automate déterministe avec un seule état initial et un seule état final)

1. Écrivez la fonction `saisis_automate(…)` qui demande à l'utilisateur de saisir les différents champs (*nombres d'états, l'état initiale, l'alphabet, l'état final – un seule état final*) et *les transitions*) constituant un automate. Les valeurs saisies sont utilisées pour initialiser l'automate passé en paramètre.
2. Écrivez la fonction `affiche_automate(…)` qui affiche les champs de l'automate a passé en paramètre.
3. Écrivez une fonction `teste_mot(…)` qui réalise les opérations suivantes :
   a. Afficher l'automate reçu en paramètre ;
   b. Demander à l'utilisateur de saisir une chaine de caractères contenant seulement des lettres de l'alphabet ;
   c. Tester si le mot représenté par cette chaine de caractères est accepté ou pas par l'automate.
   d. Afficher le résultat de ce test.
4. Testez votre programme avec quelques automates

NB/

Exercice 2 : Elimination des ε-transitions d'un automate

Donnez un programme C qui permet de :

- Entrer au clavier la matrice de codage d'un automate ***Autep*** non déterministe et avec **ε-transitions** qui correspond à un langage L
- Puis afficher la matrice de codage de l'automate ***Aut*** sans **ε-transitions** (équivalant à ***Autep***) qui correspond au langage L