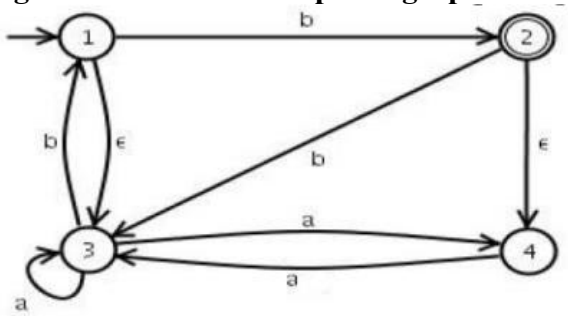
Université de Batna 2
Département d'informatique
2ème année Licence académique
Module : théorie des langages

Correction de la série de la première semaine:
L'automate d'états finis et son langage reconnu

Objectif du TD **: est d'appliquer quelques algorithmes sur un automate d'états finis tel que l'algorithme d'*élimination des transitions* vides, l'algorithme de *détermination*, l'algorithme de *minimisation* et l'algorithme du *passage à son langage reconnu*.**

Exercice :

Soit A l'automate d'états finis généralisé caractérisé par le graphe de transition suivant :



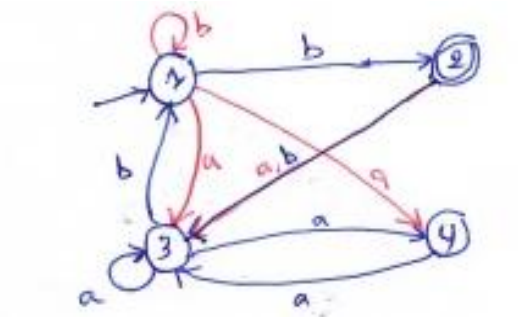
☞**Avant tout, l'étudiant doit savoir c'est quoi un automate d'états finis généralisé.**
Dans le premier temps, on dit que l'automate d'états finis est généralisé s'il contient au moins une transition étiquetée par ε
☞**Aussi à travers cet automate, l'étudiant doit faire la différence entre l'état initial et l'état final.**
Un état initial, c'est chaque nœud précédé d'une flèche. Dans un automate, on peut trouver un ou plusieurs états initiaux.
Un état final, c'est chaque nœud entouré d'un cercle. Dans un automate, on peut trouver un ou plusieurs états finaux.
☞**Aussi à travers cet automate, l'étudiant doit comprendre comment il calcule le ε-successeur d'un état quelconque.**
Le ε-successeur d'un état p noté ε-successeur (p), c'est l'ensemble des états accessibles à partir de cet état en suivant un chemin de transitions étiquetées par ε

1) **Supprimer les ε-transitions de telle sorte on obtient un automate d'états finis simple équivalent à cet automate d'états finis.**

Réponse :

✍ Dans l'automate de l'exercice : ε-successeur (1) = {3} et ε-successeur (2) = {4}
✍ Donc, les états concernés par l'ε-successeur, ce sont *l'état 1* et *l'état 2*
Selon l'algorithme d'élimination des transitions vides :
- ✓ Chaque état concerné par l'ε-successeur qui n'est pas final **devient final** si son ε-successeur contient un état final.

✍ Donc, *l'état 1* ne devient pas final.
✍ Donc, *l'état 2*, il est déjà final.
- ✓ Chaque état concerné par l'ε-successeur **fait aussi les mêmes transitions** de tous les états de son ε-successeur.
  ✍ Donc, *l'état 1* fait aussi les transitions suivantes :
  $\partial(1,a)=3$ C'est-à-dire : *l'état 1* fait la transition vers l'état 3 en lisant le symbole *a*
  $\partial(1,b)=1$ C'est-à-dire : *l'état 1* fait la transition vers lui même en lisant le symbole *b*
  $\partial(1,a)=4$ C'est-à-dire : *l'état 1* fait la transition vers l'état 4 en lisant le symbole *a*
  ✍ Donc, *l'état 2* fait aussi la transition suivante:
  $\partial(2,a)=3$ C'est-à-dire : *l'état 2* fait la transition vers l'état 3 en lisant le symbole *a*
- ✓ Maintenant, dans le graphe on supprime toutes les ε-transitions et on ajoute les nouvelles transitions.
  ✍ Donc, le graphe devient comme ça :



☞L'étudiant doit savoir quand est qu'un automate d'états finis n'est pas déterministe.
On dit qu'un automate d'états finis n'est pas déterministe s'il contient plus d'un état initial ou plus d'une transition à partit d'un même état en lisant le même symbole.

2) L'automate obtenu est déterministe ? Sinon; le déterminiser.

Réponse :

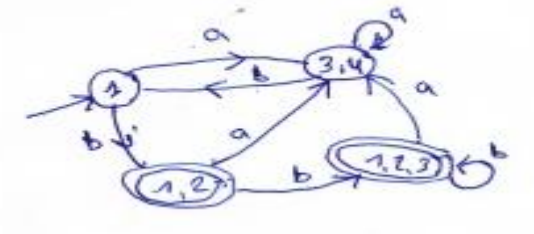
✍ L'automate obtenu n'est pas déterministe puisque il contient par exemple deux transitions sur le même symbole *a* à partit du même *état 1*

Selon l'algorithme de détermination :

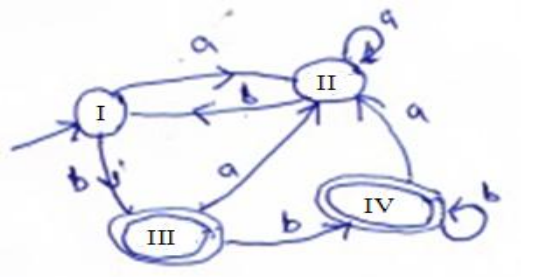
- ✓ Regroupez les états initiaux, cet ensemble devient le nouveau état initial de l'automate déterministe
  - ✍ Dans notre exercice, le nouveau état initial contient seulement *l'état 1*
- ✓ Rajoutez dans la table de transition de l'automate déterministe tous les nouveaux états produits avec leurs transitions.
- ✓ Recommencer cette dernière étape jusqu'à ce qu'il n'y ait plus de nouveau état.
- ✓ Chaque nouveau état qui contient au moins un état final devient final
- ✓ On peut renommer les nouveaux états.
  - ✍ Donc, on trouve cette table de transition:

état	a	b
→ {1}	{3,4}	{1,2}
{3,4}	{3,4}	{1}
{1,2}	{3,4}	{1,2,3}
{1,2,3}	{3,4}	{1,2,3}

✍ D'où le graphe de l'automate déterministe est représenté comme suit :



✍ On peut renommer maintenant notre automate déterministe comme suit :



☞**L'étudiant doit savoir quand est qu'un état est accessible.**
Un état est accessible s'il existe un chemin à partir d'un état initial vers lui.
Remarque : chaque état initial est accessible.
☞**L'étudiant doit savoir quand est qu'un état est co-accessible.**
Un état est co-accessible s'il existe un chemin à partir de cet état vers un état final.
Remarque : chaque état final est co-accessible.

3) Minimiser l'automate déterministe

Réponse :

Selon l'algorithme de minimisation :

- ✓ On élimine de l'automate tous les états inaccessibles et non co-accessibles avec leurs transitions.
  ✍ Dans notre exercice, on n'élimine rien puisqu'il n'y a pas d'état inaccessible
  et puisqu'il n'y a pas d'état non co-accessible
- ✓ On partitionne les états en deux classes, une classe pour les états finaux et une autre classe pour le reste des états.
  ✍ Dans notre exercice, on trouve: A = {III, IV} B = {I, II}
- ✓ La troisième étape consiste à construire une table de transition entre les classes et voir le résultat de transition des états de chaque classe obtenue sur chaque symbole.
  ✍ En premier temps, on a les transitions suivantes:
  $\partial(III, a) =$ **II** l'état **II** où **se trouve ? il est dans la classe B**
  $\partial(III, b) =$ **IV** l'état **IV** où **se trouve ? il est dans la classe A**
  $\partial(IV, a) =$ **II** l'état **II** où **se trouve ? il est dans la classe B**
  $\partial(IV, b) =$ **IV** l'état **IV** où **se trouve ? il est dans la classe A**
  Même chose avec la suite de la table et on trouve:

		Les symboles	
Classe obtenue	Ses états	*a*	*b*
A	III	**B**	**A**
	IV	**B**	**A**
B	I	**B**	**A**
	II	**B**	**B**

- ✓ Ensuite, on doit éclater en d'autres classes seulement la classe d'états ayant des résultats différents de transition sur un symbole quelconque.
  ✍ Dans la dernière table, on trouve que:
  La *classe A* ne s'éclate pas puisque tous ses états ont sur chaque symbole le même résultat de transition. Donc A reste elle-même A = {III, IV}
  La *classe B* doit être éclatée puisque ses états ont sur le symbole *b* des résultats différents de transition. C'est-à-dire la *classe A* sera éclaté en des classes de telle sorte que chacune contient des états qui ont le même résultat de transition sur chaque symbole. Ainsi La *classe B* sera éclatée en deux classes *classe B et classe C*
  B = {I} C= {II}
- ✓ Recommencer la troisième étape jusqu'à ce qu'il n'y ait plus d'éclatement possible.

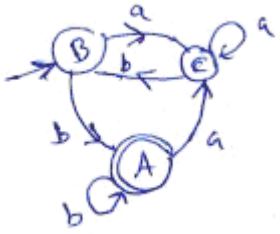
		Les symboles	
Classe obtenue	Ses états	*a*	*b*
A	III	**C**	**A**
	IV	**C**	**A**
B	I	**C**	**A**
C	II	**C**	**B**

  ✍ Comme il n'y ait plus d'éclatement, on s'arrête.
- ✓ La dernière table décrit l'automate minimal.
- ✓ Chaque classe de cette dernière table est un état de l'automate minimal.

- ✓ La classe qui contient l'état initial devient le nouveau état initial.
- ✓ Toute classe contenant un état final devient un nouveau état final.
  - ✍ On résume la dernière table et on obtient :

	Les symboles	
Classe obtenue	*a*	*b*
A	**C**	**A**
→ **B**	**C**	**A**
C	**C**	**B**

✍ D'où le graphe de l'automate minimal est représenté comme suit :



4) Trouver le langage reconnu par cet automate (via le lemme d'Arden).

Réponse :

Selon l'algorithme du passage d'un automate à son langage reconnu, on trouve le système d'équations associé à notre automate minimal:

$$\begin{cases} L(A) = b\,L(A) + a\,L(C) + \varepsilon & .....(1) \\ L(B) = b\,L(A) + a\,L(C) & .....(2) \\ L(C) = a\,L(C) + b\,L(B) & .....(3) \end{cases}$$

- ✓ Le langage reconnu par un automate est donnée par le langage engendré par ses états initiaux.

✍ Donc le langage reconnu par notre automate est donnée par le langage engendré par l'état initial *B*.
C'est-à-dire $L(B)$.

- ✓ On cherche $L(B)$ en utilisant le **lemme d'Arden.**

En appliquant par exemple le lemme sur *(3)*, on trouve que:

$L(C) = a^* b\,L(B)$ .....*(4)*

En remplaçant *(4)* dans *(1)*, on trouve que :

$L(A) = b\,L(A) + a\,a^* b\,L(B) + \varepsilon$ .....*(5)*

En appliquant maintenant le lemme sur *(5)*, on trouve que:

$L(A) = b^*\left(a\,a^* b\,L(B) + \varepsilon\right)$ .....*(5)* Après la distributivité, cette formule devient :

$L(A) = b^* a\,a^* b\,L(B) + b^*$ .....*(6)*

En remplaçant maintenant *(4)*et *(6)*dans *(2)*, on trouve que :

$L(B) = b\left(b^* a\,a^* b\,L(B) + b^*\right) + a\,a^* b\,L(B)$ .....*(2)* Après la distributivité, cette formule devient :

$L(B) = bb^* a\,a^* b\,L(B) + bb^* + a\,a^* b\,L(B)$ .....*(7)* factorisation de $L(B)$, cette formule devient :

$L(B) = \left(bb^* a\,a^* b + a\,a^* b\right)L(B) + bb^*$ .....*(8)*

En appliquant maintenant le lemme sur *(8)*, on trouve que:

$L(B) = \left(bb^* a\,a^* b + a\,a^* b\right)^* bb^*$

✍ Donc le langage reconnu par notre automate est l'expression suivante qui ne contient que des symboles de a et b .

C'est-à-dire $\left(bb^* a\,a^* b + a\,a^* b\right)^* bb^*$