Année 2008-2009

Examen de rattrapage d'Automates Finis (AF4)

éléments de corrections


corrigé partiel et non officiel,
à lire d'un œil critique


Exercice 1 :

Rappels de cours

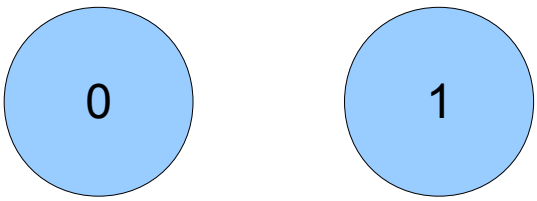
On peut définir un automate fini *A* comme un quintuplet *A* = < A, Q, δ, I, T >, où :

- A est l'alphabet lu par l'automate (souvent {a,b} dans ce cours)
- Q est l'ensemble des états de l'automate
- δ (delta minuscule) est l'ensemble des transitions (flèches) de l'automate
- I est l'ensemble des états initiaux (un état unique si l'automate est déterministe)
- T est l'ensemble des états terminaux

Méthode :

Dessinons tout d'abord l'automate *A*.

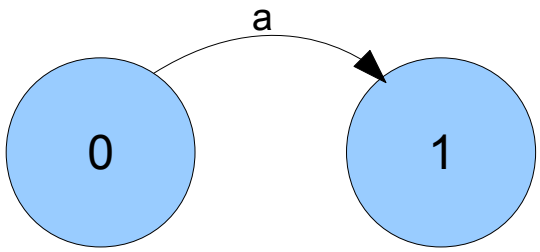
On comprend, en regardant le tableau des transitions, que l'automate comporte 2 états nommés 0 et 1 (impossible de confondre les états et les lettres du langage, puisqu'il est dit que l'alphabet de l'automate est {a,b}, et que 0 et 1 apparaissent respectivement dans les états initiaux et terminaux), ce qui donne :



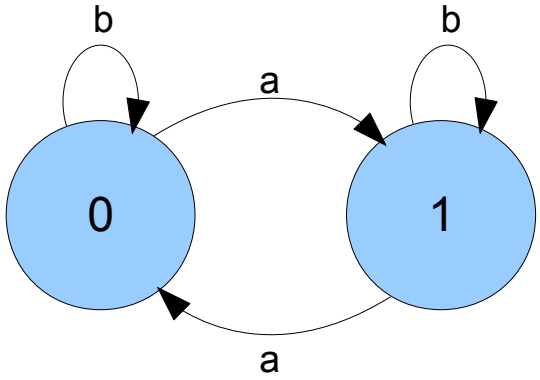
D'après le croisement de la ligne 0 et de la colonne *a* du tableau, on comprend qu'une transition portant la lettre *a* va de 0 en 1

δ	a	b
0	1	0
1	0	1

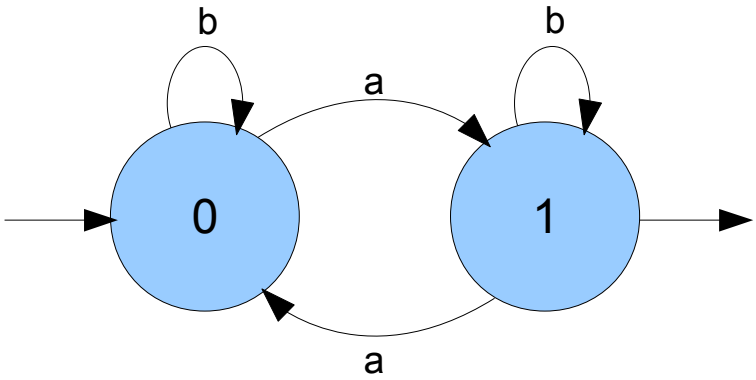
Ce qui donne :



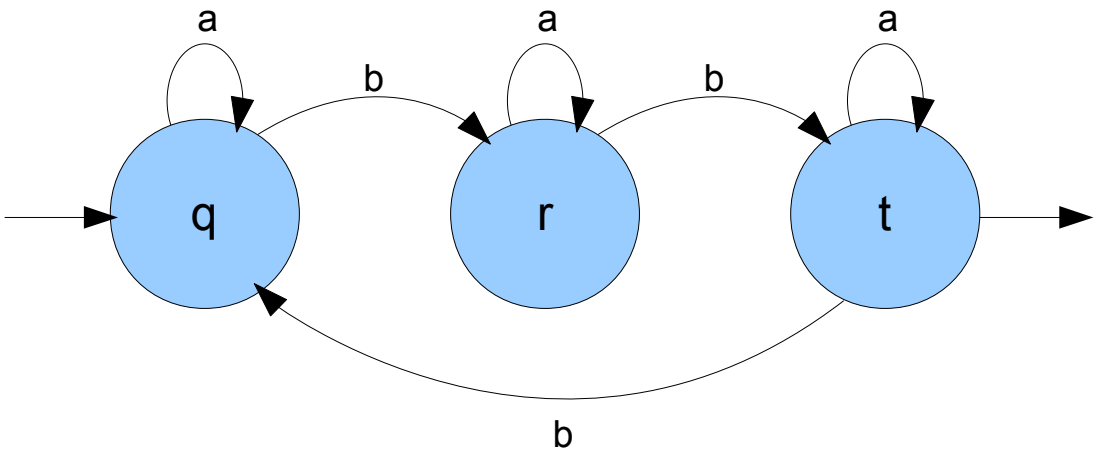
On fait de même pour les 3 autres transitions du tableau :



Enfin, on place l'état initial (0) et l'état final (1) qui sont donné dans la définition de l'automate ($A = < A, Q, \delta, \underline{0}, \{\underline{1}\} >$) et on obtient l'automate A :



En appliquant la même méthode pour l'automate B on obtient :



Construisons à présent C et D.

Rappels de cours

Pour construire un automate reconnaissant l'union ou l'intersection des langages reconnus par 2 automates finis (déterministes, complets, et lisant le même alphabet), il existe un algorithme assez simple :

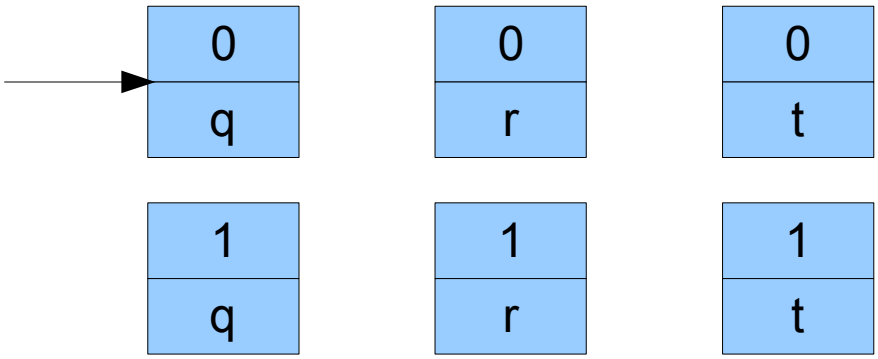
On commence par faire le produit cartésien des 2 automates, c'est-à-dire un nouvel automate construit de la façon suivante :

- Pour chaque était *i* du premier automate et pour chaque était *j* du second, on crée un état "*i-j*".
- L'état qui porte le nom des 2 états initiaux devient initial.
- Pour chacun de ces états *i-j*, et pour chaque lettre α de l'alphabet des automates, on crée une transition vers portant la lettre α vers l'état *k-l*, où *k* est la cible de *i* avec la lettre α dans le 1er automate, et *l* la cible de *j* avec la lettre α dans le 2nd.

Pour faire l'automate de l'intersection : tous les états i-j tels que i est terminal dans le 1er automate et j est terminal dans le 2nd deviennent terminaux.

Pour faire l'automate de l'union : tous les états i-j tels que i est terminal dans le 1er automate ou j est terminal dans le 2nd deviennent terminaux.

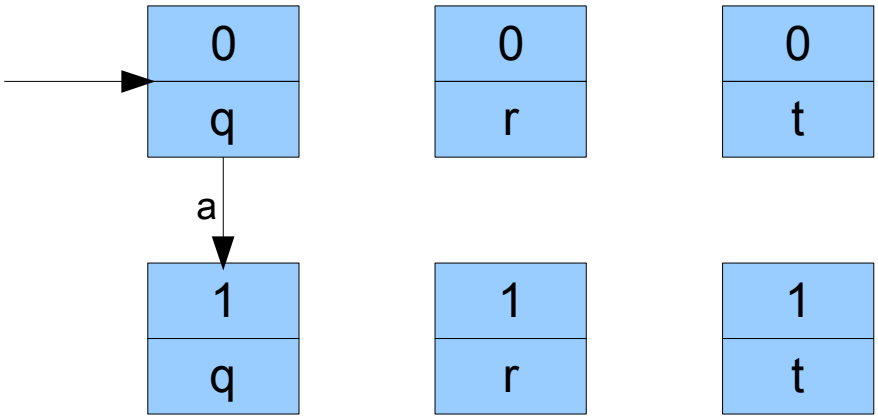
On crée les états du produit cartésien, en mettant comme état initial *0-q* (puisque *0* est initial dans le premier automate, et *q* est initial dans le second) :



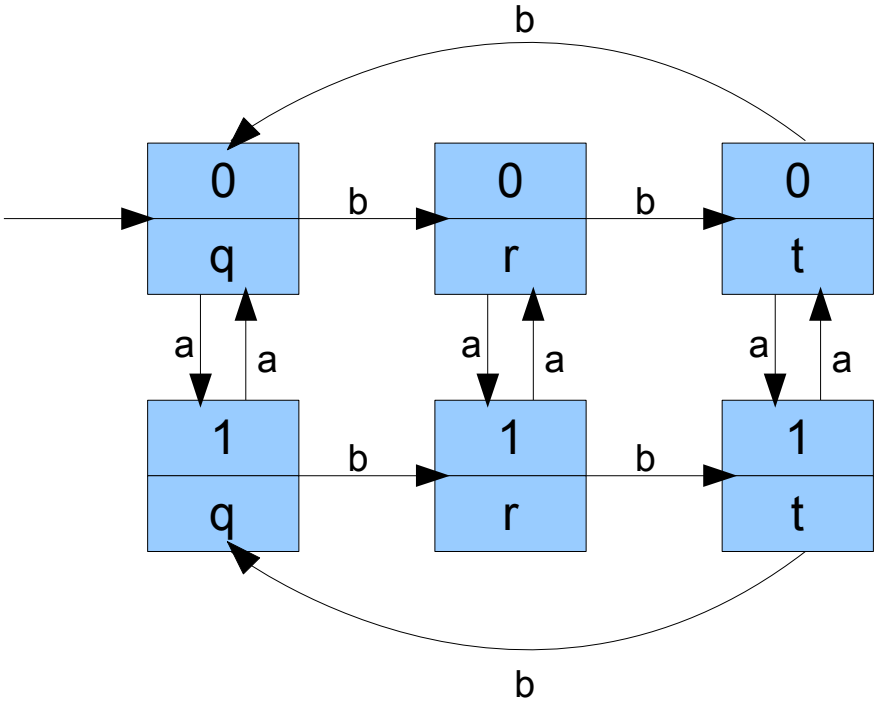
Pour chaque état de départ et pour chaque lettre on va placer une transition.
Commençons par l'état *0*-*q* et la lettre *a* par exemple :

- La transition partant de *0* et portant un *a* va vers *1*.
- La transition partant de *q* et portant un *a* va vers *q*.

Donc la transition partant de *0*-*q* portant un *a* va vers *1*-*q* :



De même pour chaque autre état et lettre :

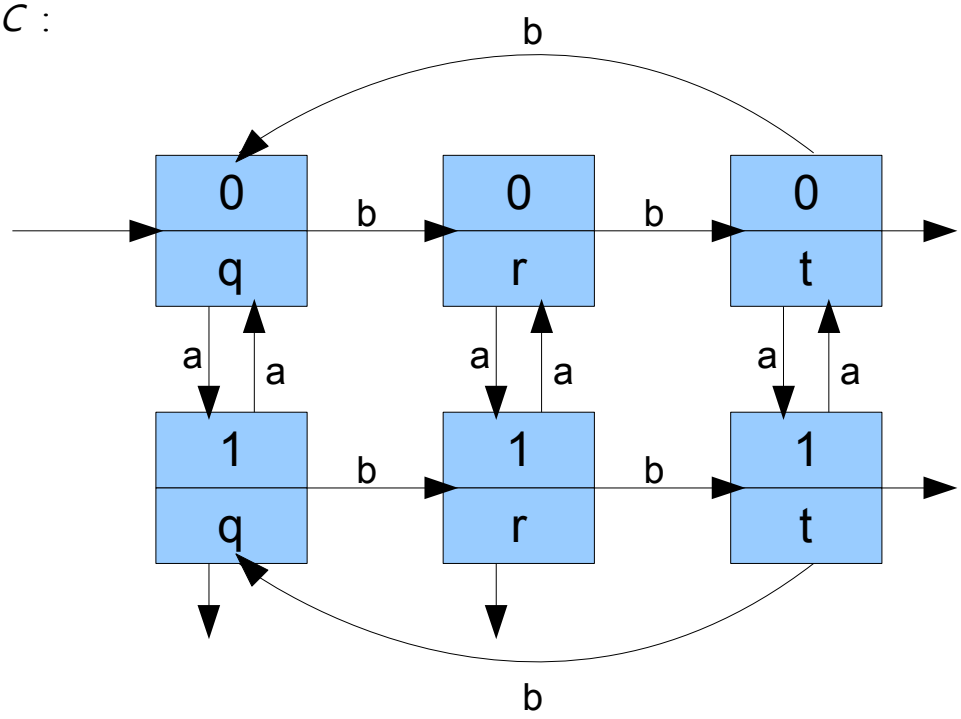


Pour l'automate de l'union C, on rend terminaux les états portant *1* ou *t* (les états terminaux respectifs de A et B), c'est-à-dire *0-t*, *1-q*, *1-r* et *1-t*.

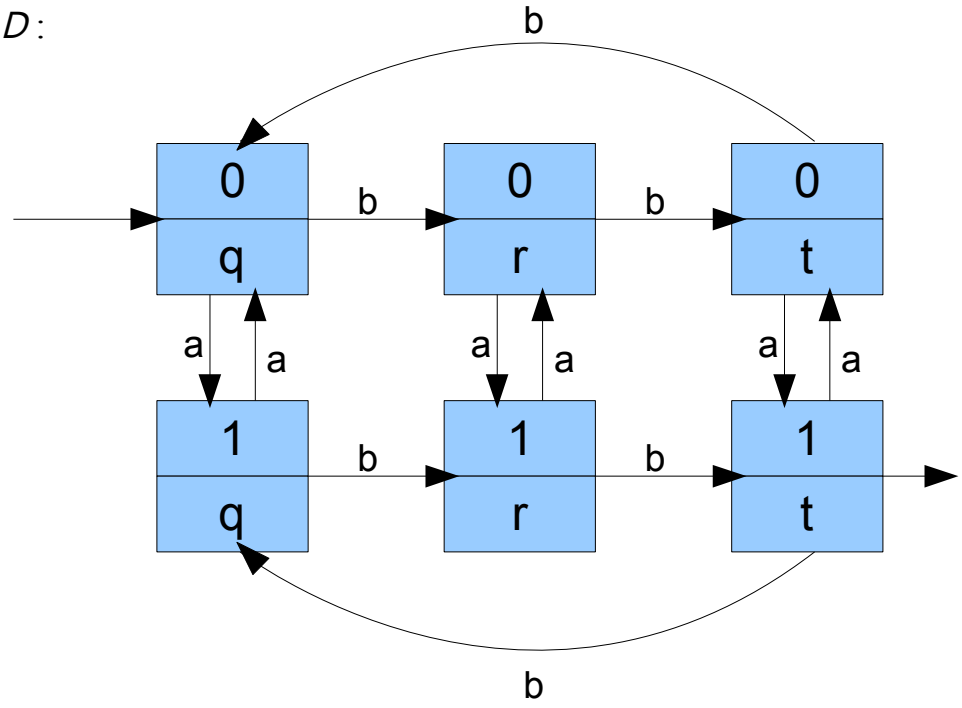
Pour l'automate de l'intersection D, on rend terminaux l'état portant *1* et *t* (les états terminaux respectifs de A et B), c'est-à-dire *1-t*.

Correction :

Automate C :



Automate D :



Remarque : Il n'était pas forcement indispensable de construire les automates *A* et *B* pour trouver *C* et *D*, mais ça évite parfois des erreurs : le fait qu'on reconnaisse l'automate *A* en regardant les colonnes du produit cartésien, et l'automate *B* en regardant les lignes, confirme qu'on ne s'est pas trompé.

Exercice 2 :

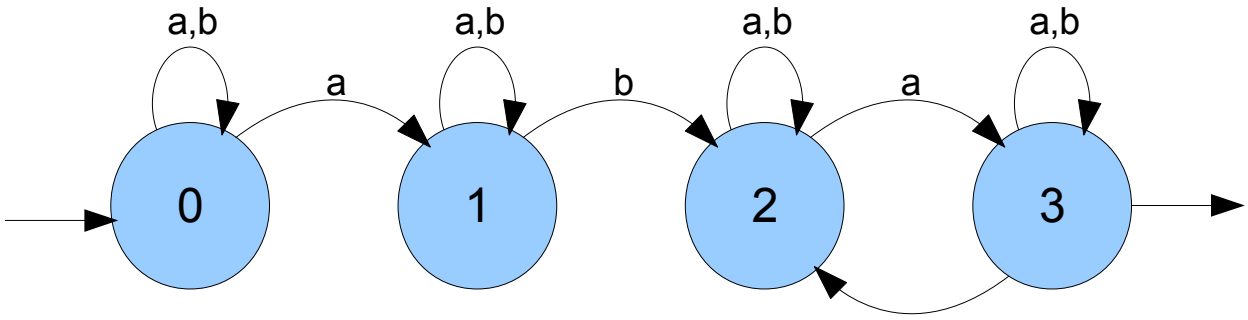
Rappels de cours

Un automate déterministe est un automate qui n'a qu'un seul état initial, et dont chaque état porte, au plus, une seule transition par lettre (par exemple, si depuis l'un des états d'un automate partent 2 flèches portant la lettre *a*, alors cet automate n'est pas déterministe).

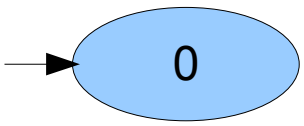
On appelle déterminiser un automate A le fait de construire un automate B équivalent à A (c'est-à-dire reconnaissant le même langage) mais qui soit déterministe.

Méthode :

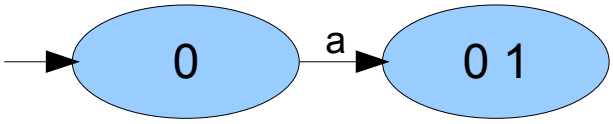
Ici, il n'est pas vraiment nécessaire de dessiner l'automate, raisonner sur le tableau suffit, mais pour ceux qui préfèrent se le représenter, voilà à quoi il devrait ressembler :



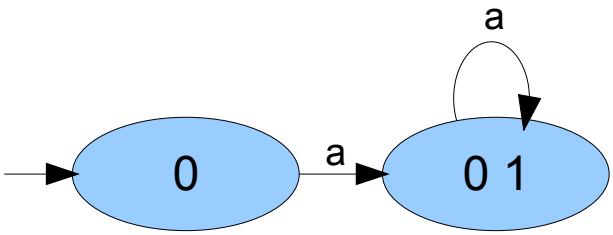
Pour construire l'automate déterministe B, on commence par mettre un état initial portant le nom de tous les états initiaux de A (A n'étant pas déterministe, il pourrait en avoir plusieurs) :



On place sur cet état une transition portant la lettre *a*.
Dans A, 0 pointe vers 0 et 1 avec la lettre a.
Donc, dans B, 0 pointe vers 0-1 avec la lettre a.
Comme 0-1 n'existe pas encore, on le crée :

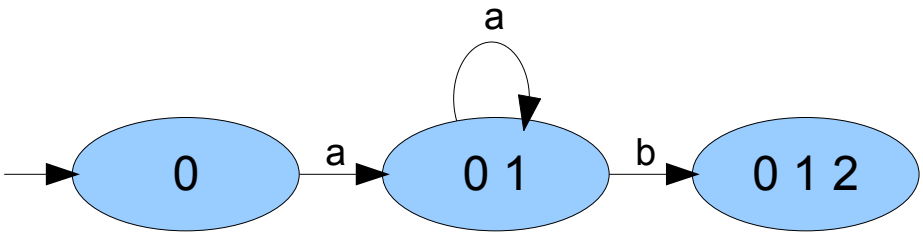


On recommence avec 0-1 :
On place sur cet 0-1 une transition portant la lettre *a*.
Dans A, 0 pointe vers 0 et 1, et 1 pointe vers 1 avec la lettre *a*.
Donc, dans B, 0-1 pointe vers 0-1 avec la lettre *a*.
0-1 existe déjà, on ne le crée pas :

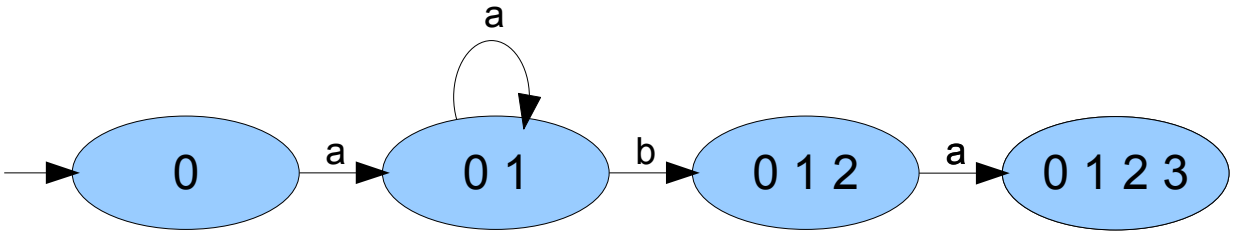


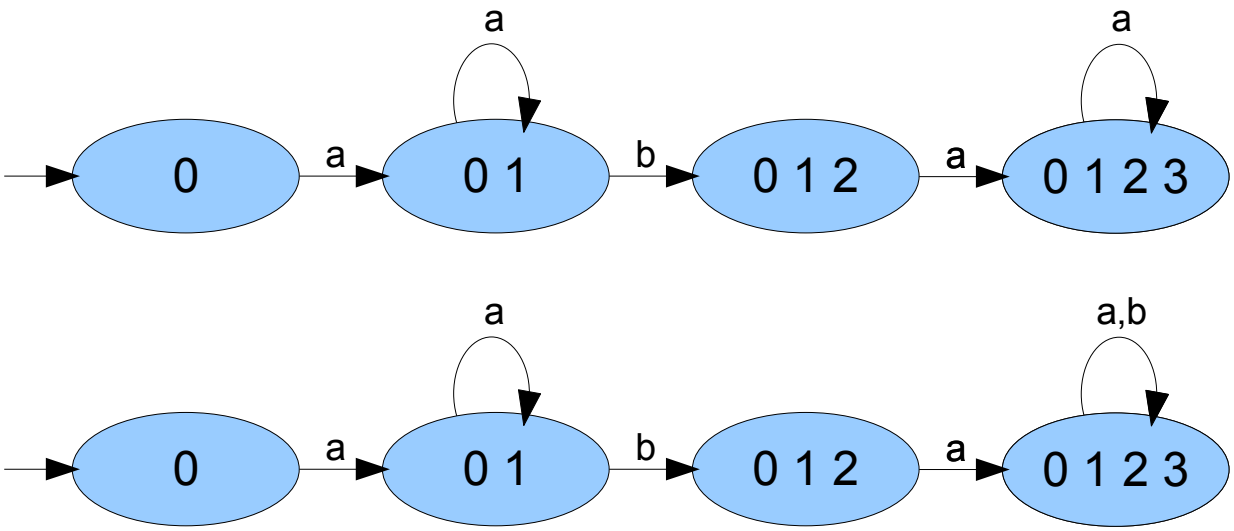
Remarque : L'ordre dans lequel on prend les lettres et les états importe peu, tant qu'à la fin chaque état porte une transition pour chaque lettre. Néanmoins, il est recommandé de suivre un algorithme de construction précis pour ne pas oublier des transition. Ici, on fait un parcours en profondeur : on essaye toujours d'abord de mettre un *a* sur le dernier état dessiné, s'il en a déjà un, on essaye mettre un b, et s'il en a déjà un, on revient à l'état précédent, jusqu'à revenir avant le 1er état.

On place maintenant sur cet 0-1 une transition portant la lettre *b*.
Dans A, 0 pointe vers 0, et 1 pointe vers 1 et 2 avec la lettre *b*.
Donc, dans B, 0-1 pointe vers 0-1-2 avec la lettre *b*.
0-1-2 n'existe pas encore, on le crée :

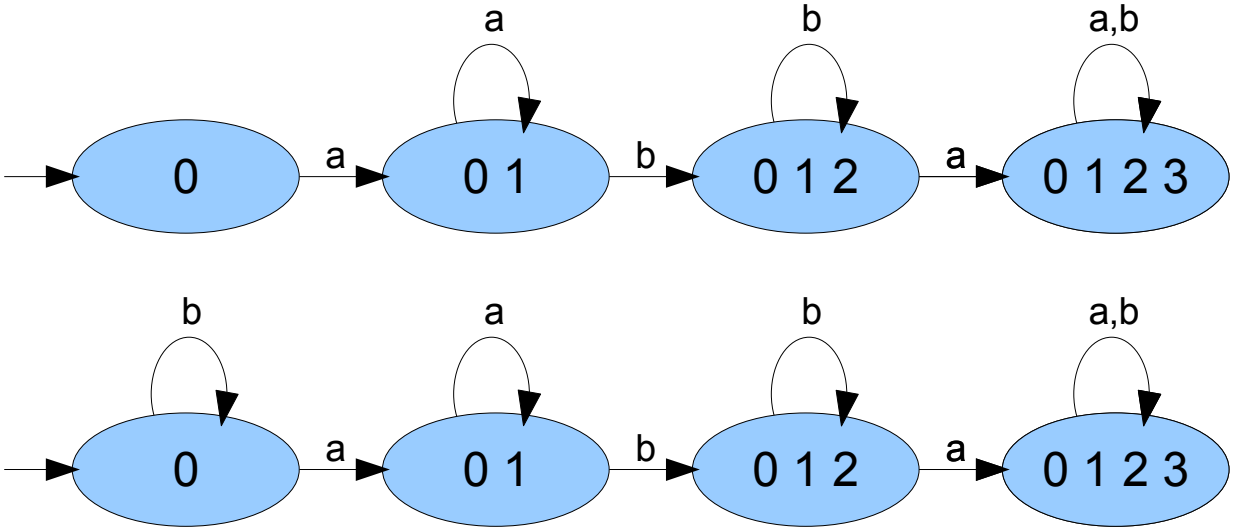


Et on continue, ce qui donne successivement :



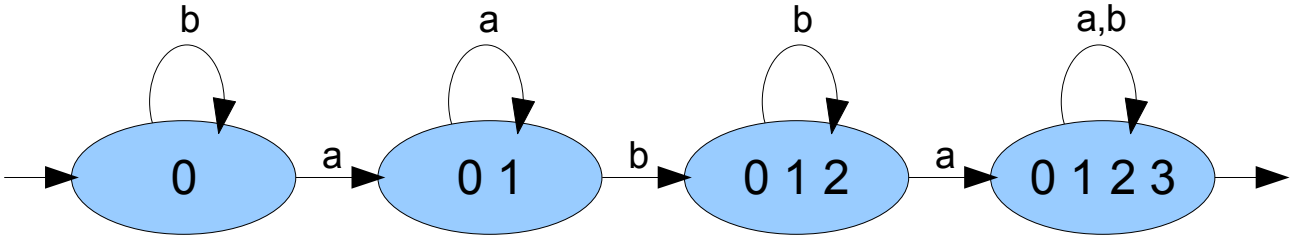


Comme l'état 0-1-2-3 porte un *a* et un *b*, on revient sur 0-1-2 :



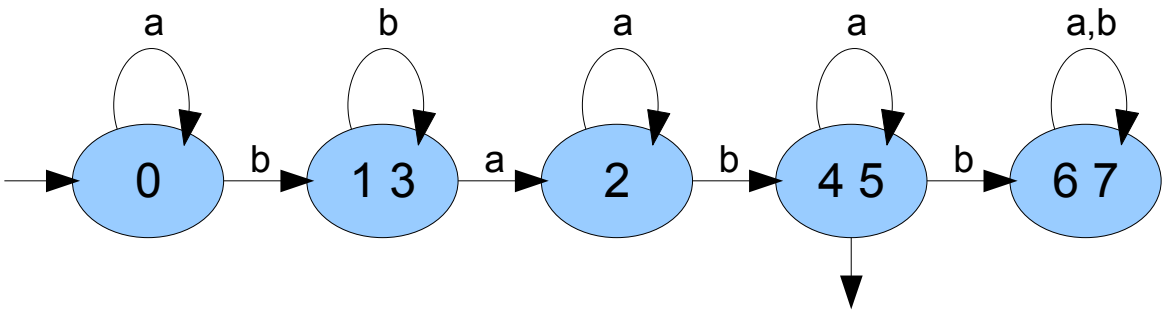
Et enfin, on met terminaux tous les états de B portant le nom d'un ou plusieurs états terminaux de A.
Donc, ici, on met terminaux le ou les états portant 3 dans leur nom, c'est-à-dire l'état 0-1-2-3, ce qui achève la construction de l'automate B.

Correction :



Exercice 3 :

Correction :



(Suite non traitée pour l'instant)