Maths 4.2

Expressions régulières et automates

Florent Madelaine

BUT 2 Informatique

Plan

Une impression de déjà vu

Ce que nous avons vu sur les automates en cours et TD :

- Technique 1 : Comment déterminiser un automate
  c'est-à-dire construire un automate déterministe à partir d'un automate qui ne l'est pas, tels que les 2 automates sont équivalents
- Technique 2 : Comment vérifier l'équivalence de 2 automates déterministes
  c'est-à-dire vérifier qu'ils reconnaissent le même langage
- Techniques 1 & 2 : Comment vérifier l'équivalence de 2 automates

Ce que nous avons vu concernant les automates en TP :

- Que la meilleure façon de définir un langage c'est de donner un automate ou une expression régulière
- Car une description en français n'est pas assez précise
- Que ce n'est pas toujours facile de passer d'une expression régulière à un automate et vice-versa

Cette semaine

Des automates aux langages et vice-versa

- (En cours/TD)
  transformer une expression régulière en un automate ...
- (En TP)
  transformer un automate en une expression régulière ...

... de telle sorte que le langage reconnu par l'automate et le langage décrit par l'expression régulière soient le même

Expression régulière vers automate

Objectif

Construire un automate qui reconnaît le langage

$$ab^* + ((ba)^*a^*)^*$$

Méthode

Combiner des automates simples

Expression régulière vers automate

On dispose d'un automate A_1 qui reconnaît le langage L_1 et d'un automate A_2 qui reconnaît le langage L_2.

Question

Comment construire à partir de A_1 et de A_2 des automates qui reconnaissent les langages :

- $L_1 L_2$
- $L_1 + L_2$
- L_1^*

Pour pouvoir passer progressivement d'une expression régulière à un automate

Rappel : concaténation de langages

On s'intéresse au langage :

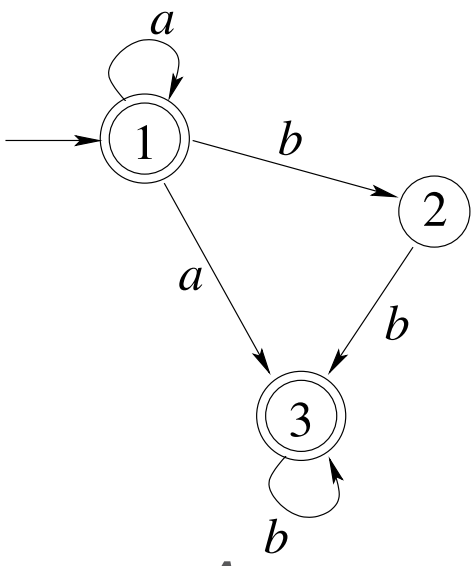
$$L = L_1 L_2$$

$$L = \{w_1 w_2 \text{ tel que } w_1 \in L_1 \text{ et } w_2 \in L_2\}.$$

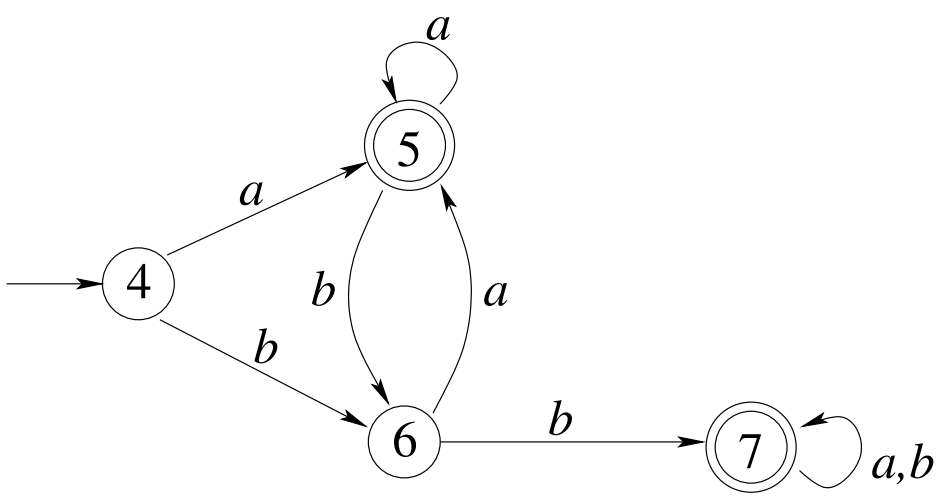
Autrement dit L est l'ensemble des mots formés comme une concaténation d'un mot de L_1 suivi d'un mot de L_2.

Concaténation d'automates

- Soient A_1 un automate permettant de reconnaître le langage L_1, et A_2 un automate permettant de reconnaître le langage L_2.



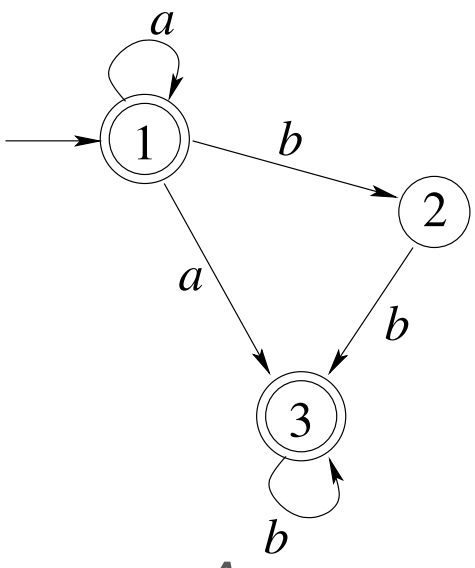
A_1



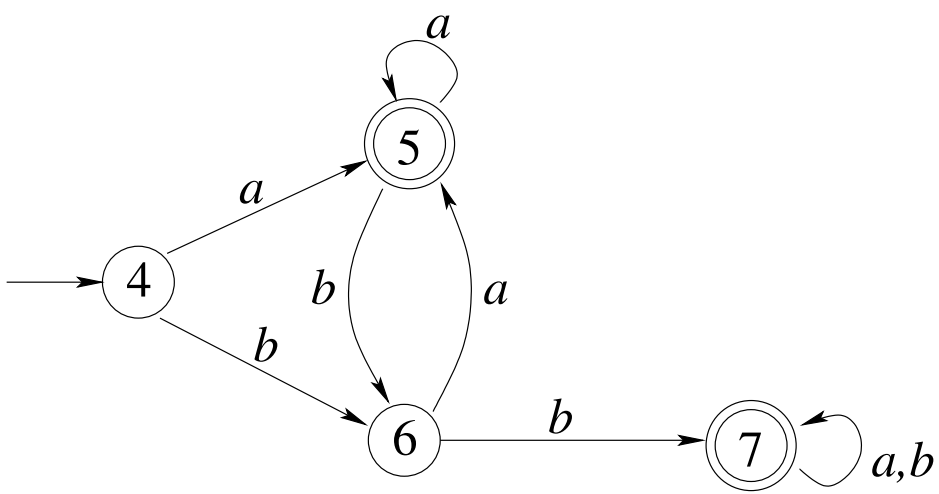
A_2

Concaténation d'automates

- Pour reconnaître le langage L_1L_2, l'automate A à construire doit alors être conçu comme la concaténation de A_1 et de A_2.



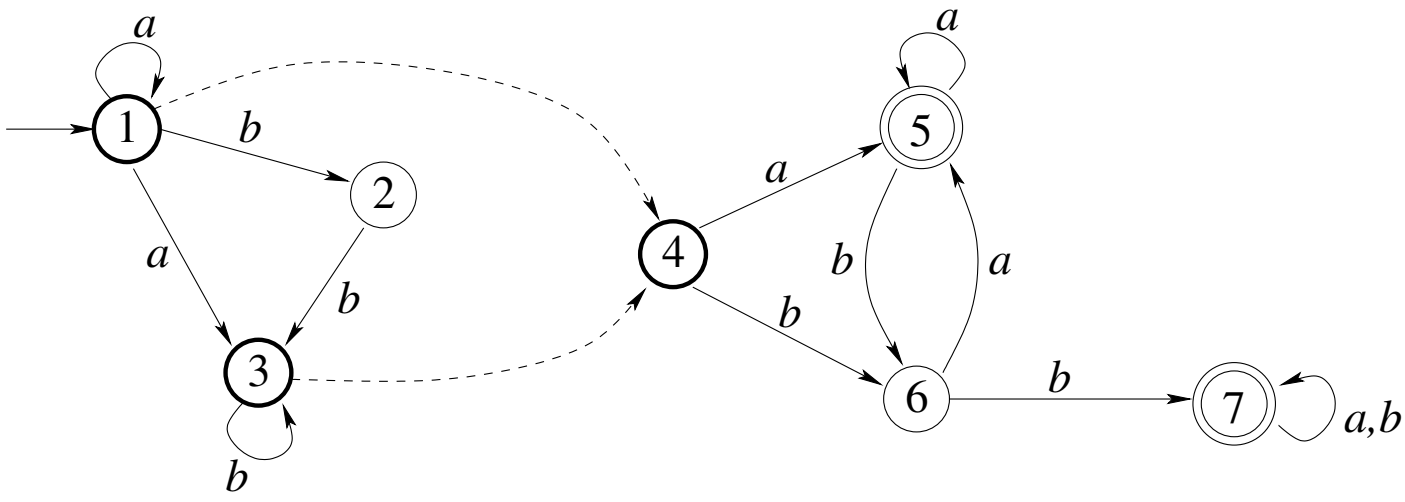
A_1



A_2

Concaténation d'automates

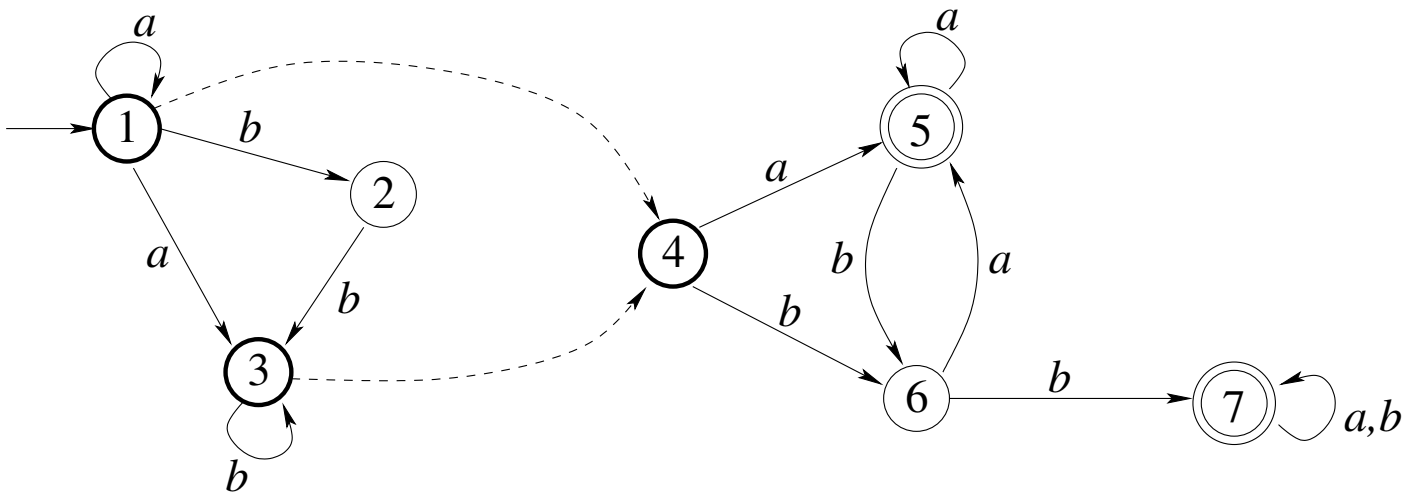
- On aimerait pouvoir réaliser cette concaténation en reliant les états acceptants de A_1 à l'état initial de A_2 par des transitions vides (sans étiquette).



Automate non valide, c'est juste une étape au brouillon

Concaténation d'automates

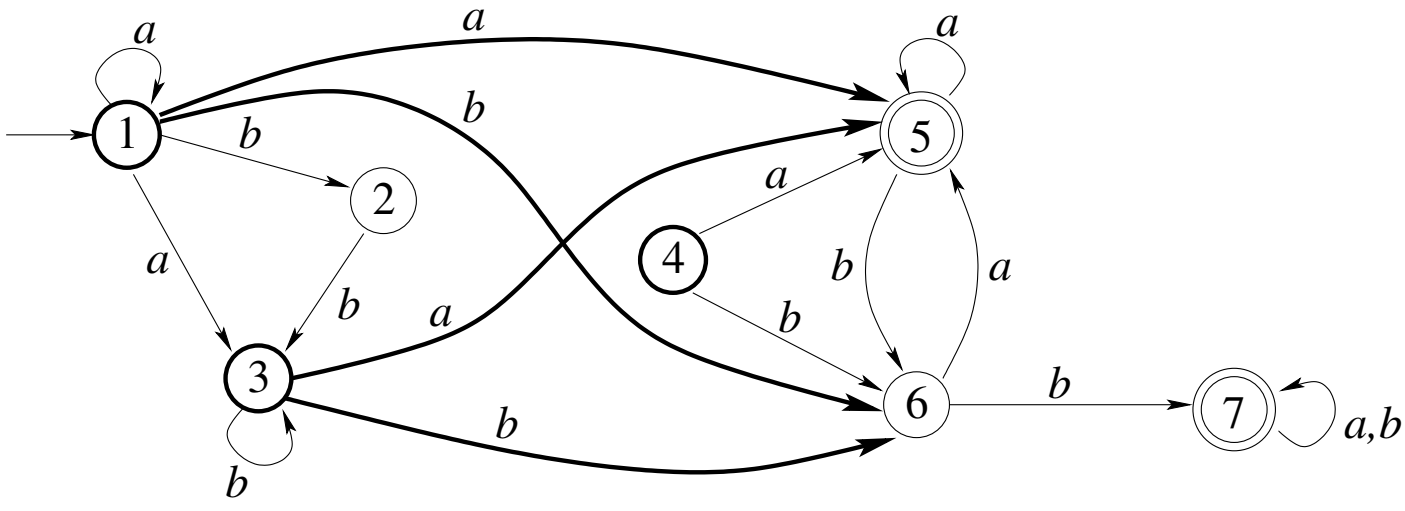
- Mais un automate ne peut pas avoir des transitions vides, donc il faudra s'en débarrasser par la suite.



Automate non valide, c'est juste une étape au brouillon

Concaténation d'automates

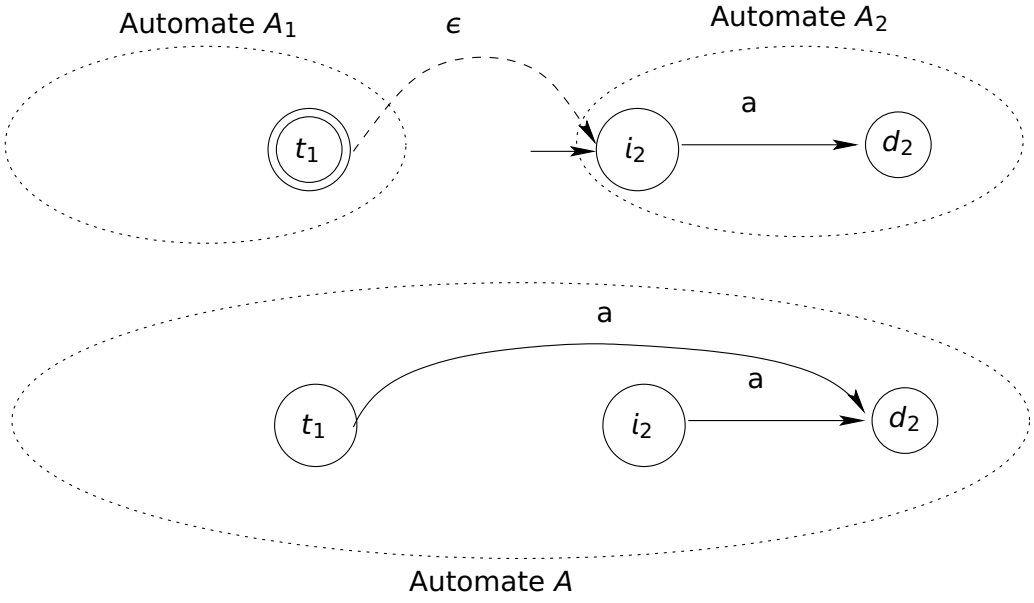
- Mais un automate ne peut pas avoir des transitions vides, donc il faudra s'en débarrasser par la suite.



Automate A valide reconnaissant les mots de $L = L_1 L_2$

Remplacement des transitions vides : idée générale

Appelons t_1 un état terminal de A_1 et i_2 l'état initial de A_2.



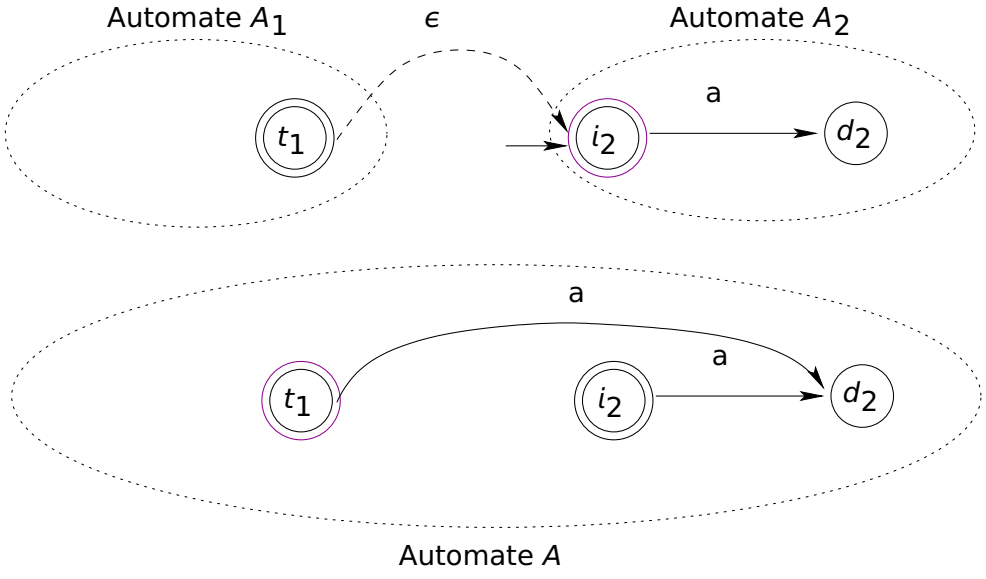
$t_1 \xrightarrow{a} d_2$ si il existe
$i_2 \xrightarrow{a} d_2$ dans A_2

Pourquoi ?

Car passer de t_1 à i_2 sans rien faire puis de i_2 à d_2 en lisant un a est équivalent à aller directement de t_1 à d_2 en lisant un a

Remplacement des transitions vides : Idée générale

Appelons t_1 un état terminal de A_1 et i_2 l'état initial de A_2.

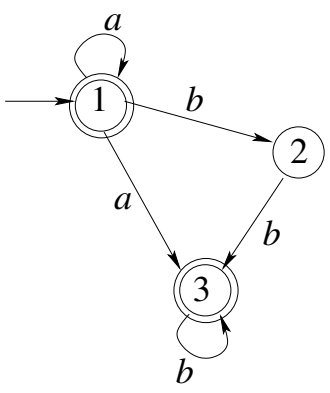


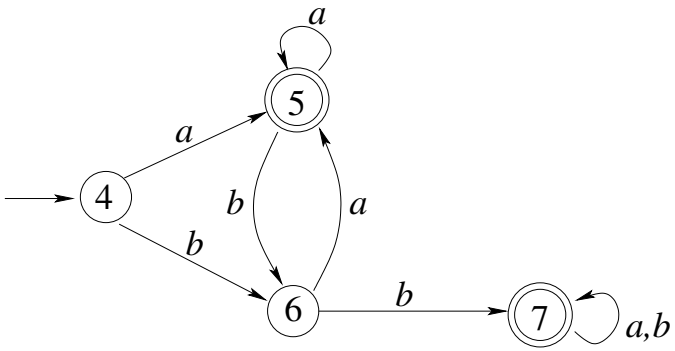
t_1 sera terminal dans A si i_2 est terminal

Pourquoi ?

Car passer de t_1 à i_2 sans rien faire, puis sortir directement de A_2 est équivalent à sortir directement de A par t_1.

Exemple

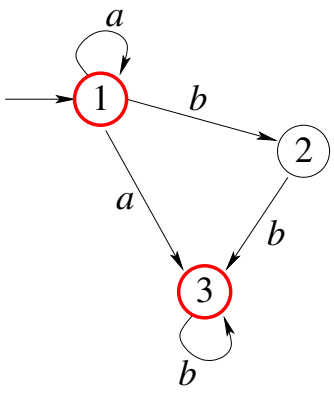


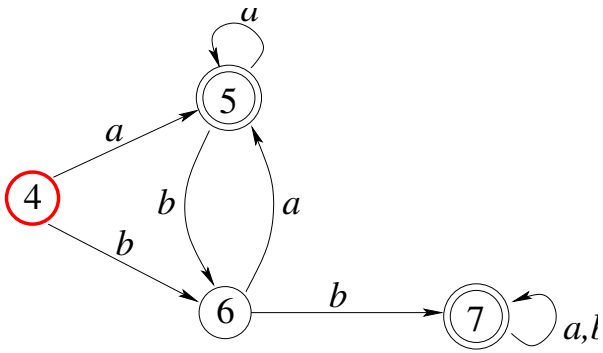


Étapes effectuées :

- on pose A_1 à gauche de A_2 ;
- l'état 4 n'est plus initial, les états 1 et 3 ne sont plus acceptants ;
- on ajoute les transitions :
  - $1 \xrightarrow{a} 5$ et $3 \xrightarrow{a} 5$ car il y avait $4 \xrightarrow{a} 5$ dans A_2
  - $1 \xrightarrow{b} 6$ et $3 \xrightarrow{b} 6$ car il y avait $4 \xrightarrow{b} 6$ dans A_2

Exemple

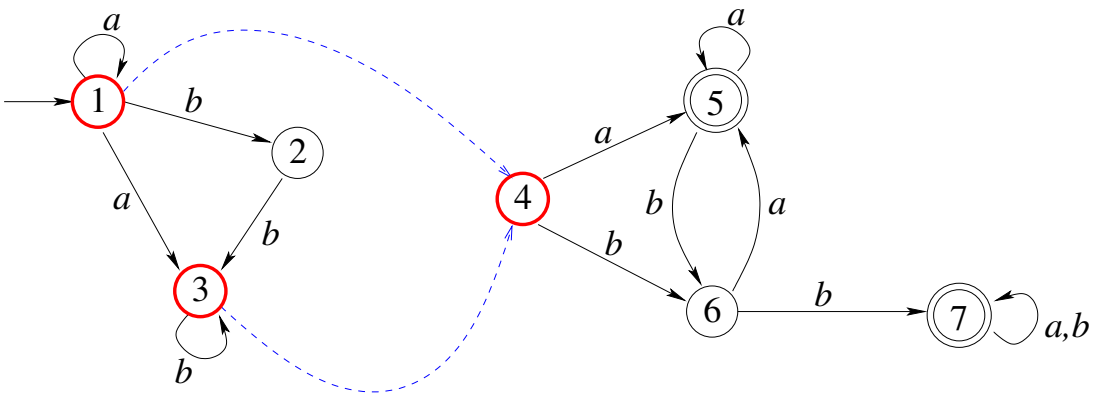




Étapes effectuées :

- on pose A_1 à gauche de A_2 ;
- l'état 4 n'est plus initial, les états 1 et 3 ne sont plus acceptants ;
- on ajoute les transitions :
  - $1 \xrightarrow{a} 5$ et $3 \xrightarrow{a} 5$ car il y avait $4 \xrightarrow{a} 5$ dans A_2
  - $1 \xrightarrow{b} 6$ et $3 \xrightarrow{b} 6$ car il y avait $4 \xrightarrow{b} 6$ dans A_2

Exemple

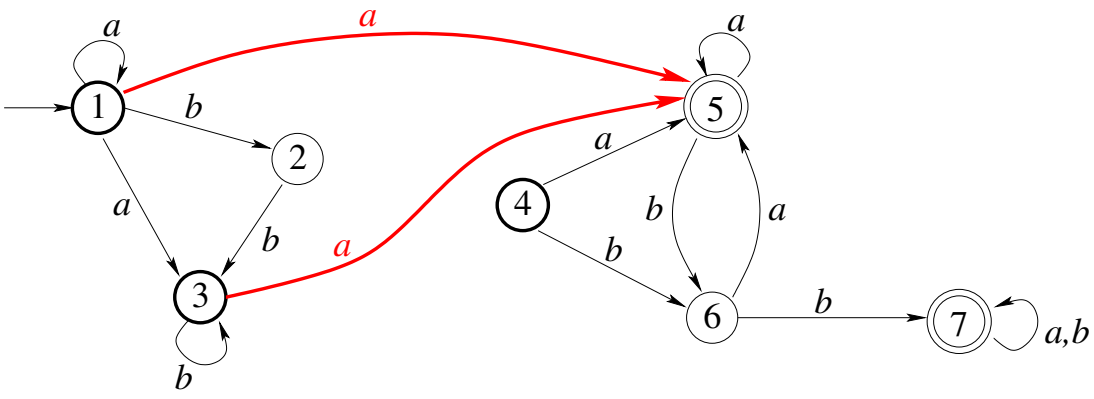


- on ajoute les transitions :
  - $1 \xrightarrow{a} 5$ et $3 \xrightarrow{a} 5$
    car il y avait $4 \xrightarrow{a} 5$ dans A_2
  - $1 \xrightarrow{b} 6$ et $3 \xrightarrow{b} 6$
    car il y avait $4 \xrightarrow{b} 6$ dans A_2

Étapes effectuées :

- on pose A_1 à gauche de A_2 ;
- l'état 4 n'est plus initial, les états 1 et 3 ne sont plus acceptants ;

Exemple

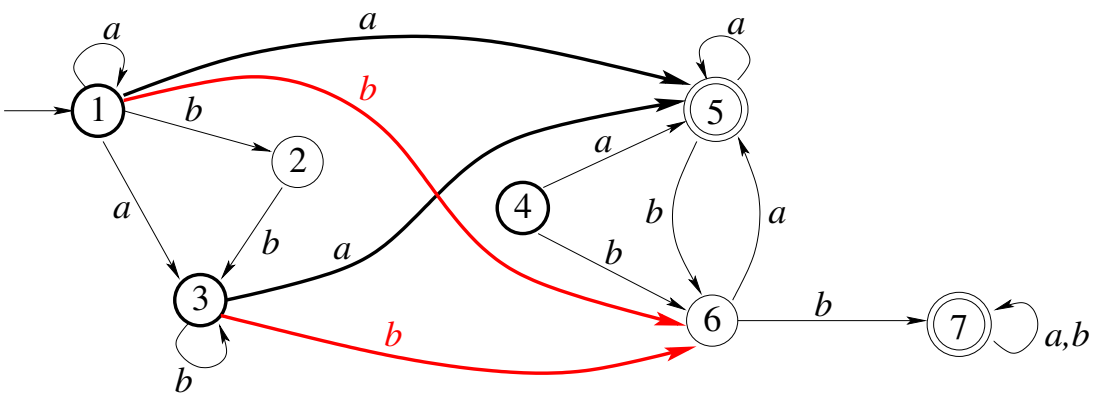


- on ajoute les transitions :
  - $1 \xrightarrow{a} 5$ et $3 \xrightarrow{a} 5$
    car il y avait $4 \xrightarrow{a} 5$ dans A_2
  - $1 \xrightarrow{b} 6$ et $3 \xrightarrow{b} 6$
    car il y avait $4 \xrightarrow{b} 6$ dans A_2

Étapes effectuées :

- on pose A_1 à gauche de A_2 ;
- l'état 4 n'est plus initial, les états 1 et 3 ne sont plus acceptants ;

Exemple



- on ajoute les transitions :
  - $1 \xrightarrow{a} 5$ et $3 \xrightarrow{a} 5$
    car il y avait $4 \xrightarrow{a} 5$ dans A_2
  - $1 \xrightarrow{b} 6$ et $3 \xrightarrow{b} 6$
    car il y avait $4 \xrightarrow{b} 6$ dans A_2

Étapes effectuées :

- on pose A_1 à gauche de A_2 ;
- l'état 4 n'est plus initial, les états 1 et 3 ne sont plus acceptants ;

Méthode point par point

- On pose les deux automates l'un à coté de l'autre (A_1 à gauche de A_2)
- L'état initial i_1 de A_1 deviendra l'état initial du nouvel automate
- L'état initial i_2 de A_2 ne sera plus initial
- Les états acceptants t de A_1 ne seront plus acceptants (dans un premier temps en tout cas)
- Les états acceptants de A_2 seront les états acceptants du nouvel automate
- On ajoute les transitions nécessaires :
  Si $i_2 \xrightarrow{a} d_2$ (avec $a \in \Sigma$) alors on ajoute une transition $t_1 \xrightarrow{a} d_2$ pour tout état acceptant t_1 de A_1
- On ajoute les états acceptants nécessaires : si i_2 est acceptant dans A_2, alors t_1 sera acceptant dans A

Méthode point par point

- On pose les deux automates l'un à coté de l'autre (A_1 à gauche de A_2)
- L'état initial i_1 de A_1 deviendra l'état initial du nouvel automate
- L'état initial i_2 de A_2 ne sera plus initial
- Les états acceptants t de A_1 ne seront plus acceptants (dans un premier temps en tout cas)
- Les états acceptants de A_2 seront les états acceptants du nouvel automate
- On ajoute les transitions nécessaires :
  Si $i_2 \xrightarrow{a} d_2$ (avec $a \in \Sigma$) alors on ajoute une transition $t_1 \xrightarrow{a} d_2$ pour tout état acceptant t_1 de A_1
- On ajoute les états acceptants nécessaires : si i_2 est acceptant dans A_2, alors t_1 sera acceptant dans A

Méthode point par point

- On pose les deux automates l'un à coté de l'autre (A_1 à gauche de A_2)
- L'état initial i_1 de A_1 deviendra l'état initial du nouvel automate
- L'état initial i_2 de A_2 ne sera plus initial
- Les états acceptants t de A_1 ne seront plus acceptants (dans un premier temps en tout cas)
- Les états acceptants de A_2 seront les états acceptants du nouvel automate
- On ajoute les transitions nécessaires :
  Si $i_2 \xrightarrow{a} d_2$ (avec $a \in \Sigma$) alors on ajoute une transition $t_1 \xrightarrow{a} d_2$ pour tout état acceptant t_1 de A_1
- On ajoute les états acceptants nécessaires : si i_2 est acceptant dans A_2, alors t_1 sera acceptant dans A

Méthode point par point

- On pose les deux automates l'un à coté de l'autre (A_1 à gauche de A_2)
- L'état initial i_1 de A_1 deviendra l'état initial du nouvel automate
- L'état initial i_2 de A_2 ne sera plus initial
- Les états acceptants t de A_1 ne seront plus acceptants (dans un premier temps en tout cas)
- Les états acceptants de A_2 seront les états acceptants du nouvel automate
- On ajoute les transitions nécessaires :
  Si $i_2 \xrightarrow{a} d_2$ (avec $a \in \Sigma$) alors on ajoute une transition $t_1 \xrightarrow{a} d_2$ pour tout état acceptant t_1 de A_1
- On ajoute les états acceptants nécessaires : si i_2 est acceptant dans A_2, alors t_1 sera acceptant dans A

Méthode point par point

- On pose les deux automates l'un à coté de l'autre (A_1 à gauche de A_2)
- L'état initial i_1 de A_1 deviendra l'état initial du nouvel automate
- L'état initial i_2 de A_2 ne sera plus initial
- Les états acceptants t de A_1 ne seront plus acceptants (dans un premier temps en tout cas)
- Les états acceptants de A_2 seront les états acceptants du nouvel automate
- On ajoute les transitions nécessaires :
  Si $i_2 \xrightarrow{a} d_2$ (avec $a \in \Sigma$) alors on ajoute une transition $t_1 \xrightarrow{a} d_2$ pour tout état acceptant t_1 de A_1
- On ajoute les états acceptants nécessaires : si i_2 est acceptant dans A_2, alors t_1 sera acceptant dans A

Méthode point par point

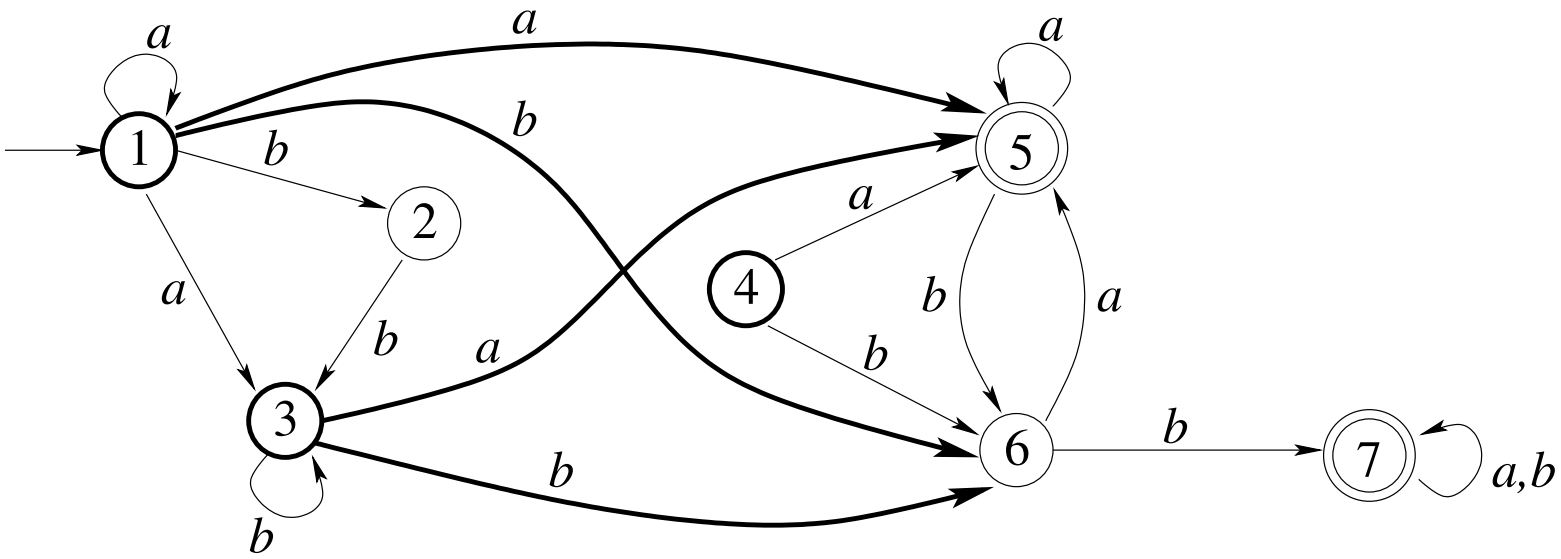
- On pose les deux automates l'un à coté de l'autre (A_1 à gauche de A_2)
- L'état initial i_1 de A_1 deviendra l'état initial du nouvel automate
- L'état initial i_2 de A_2 ne sera plus initial
- Les états acceptants t de A_1 ne seront plus acceptants (dans un premier temps en tout cas)
- Les états acceptants de A_2 seront les états acceptants du nouvel automate
- On ajoute les transitions nécessaires :
  Si $i_2 \xrightarrow{a} d_2$ (avec $a \in \Sigma$) alors on ajoute une transition $t_1 \xrightarrow{a} d_2$ pour tout état acceptant t_1 de A_1
- On ajoute les états acceptants nécessaires : si i_2 est acceptant dans A_2, alors t_1 sera acceptant dans A

Méthode point par point

- On pose les deux automates l'un à coté de l'autre (A_1 à gauche de A_2)
- L'état initial i_1 de A_1 deviendra l'état initial du nouvel automate
- L'état initial i_2 de A_2 ne sera plus initial
- Les états acceptants t de A_1 ne seront plus acceptants (dans un premier temps en tout cas)
- Les états acceptants de A_2 seront les états acceptants du nouvel automate
- On ajoute les transitions nécessaires :
  Si $i_2 \xrightarrow{a} d_2$ (avec $a \in \Sigma$) alors on ajoute une transition $t_1 \xrightarrow{a} d_2$ pour tout état acceptant t_1 de A_1
- On ajoute les états acceptants nécessaires : si i_2 est acceptant dans A_2, alors t_1 sera acceptant dans A

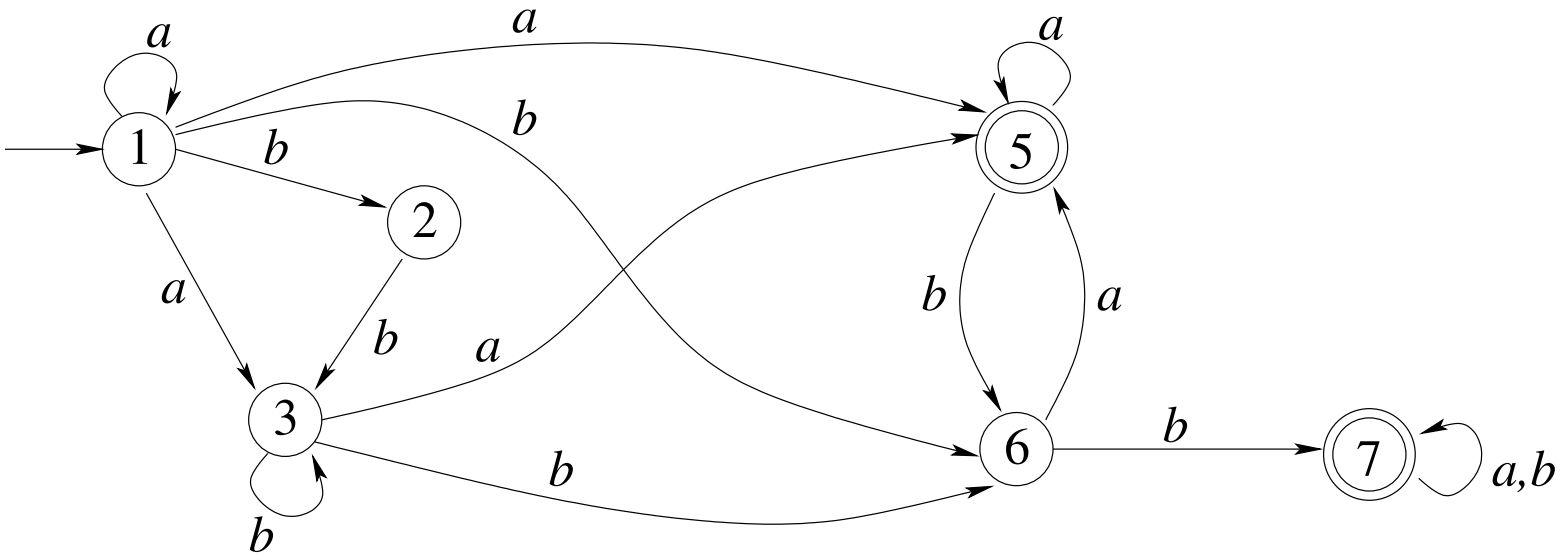
Remarque

On peut simplifier l'automate obtenu en enlevant l'état 4 car aucune transition ne permet d'arriver dessus (état non accessible) ce qui nous donne finalement l'automate suivant :

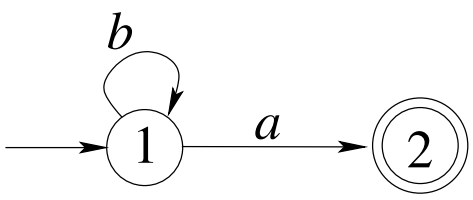


Remarque

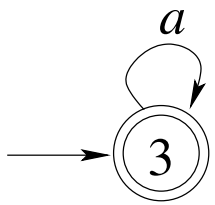
On peut simplifier l'automate obtenu en enlevant l'état 4 car aucune transition ne permet d'arriver dessus (état non accessible) ce qui nous donne finalement l'automate suivant :



Autre exemple



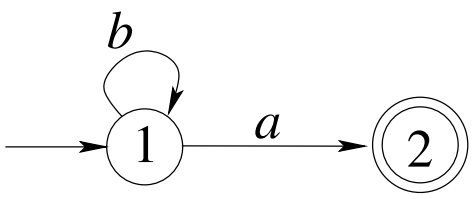
A_1



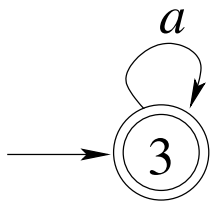
A_2

Que proposez-vous ?

Autre exemple



A_1

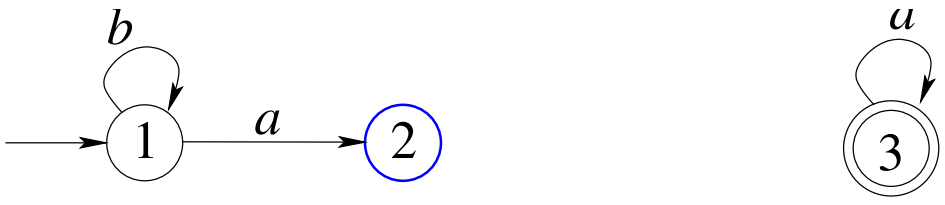


A_2

Étapes effectuées :

- On pose A_1 à gauche de A_2
- L'état 3 n'est plus initial
  L'état 2 n'est plus acceptant
- On ajoute la transition $2 \xrightarrow{a} 3$ car il y avait $3 \xrightarrow{a} 3$ dans A_2
- On remet 2 comme état acceptant de A car 3 était initial et acceptant dans A_2

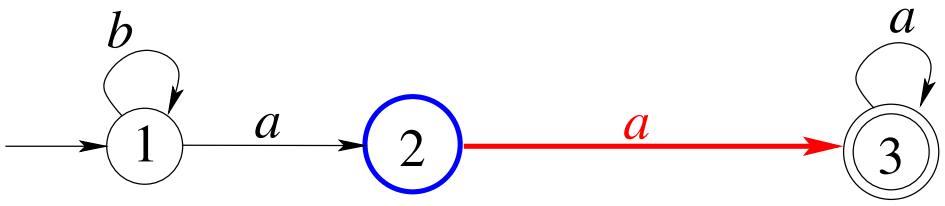
Autre exemple



Étapes effectuées :

- On pose A_1 à gauche de A_2
- L'état 3 n'est plus initial
  L'état 2 n'est plus acceptant
- On ajoute la transition $2 \xrightarrow{a} 3$ car il y avait $3 \xrightarrow{a} 3$ dans A_2
- On remet 2 comme état acceptant de A car 3 était initial et acceptant dans A_2

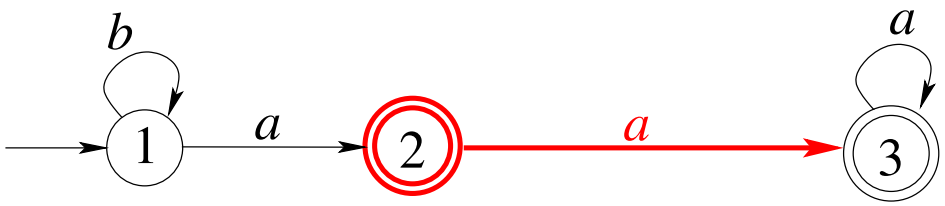
Autre exemple



Étapes effectuées :

- On pose A_1 à gauche de A_2
- L'état 3 n'est plus initial
  L'état 2 n'est plus acceptant
- On ajoute la transition $2 \xrightarrow{a} 3$ car il y avait $3 \xrightarrow{a} 3$ dans A_2
- On remet 2 comme état acceptant de A car 3 était initial et acceptant dans A_2

Autre exemple



Automate A reconnaissant le langage $L = L_1 L_2$

Étapes effectuées :

- On pose A_1 à gauche de A_2
- L'état 3 n'est plus initial
  L'état 2 n'est plus acceptant
- On ajoute la transition $2 \xrightarrow{a} 3$ car il y avait $3 \xrightarrow{a} 3$ dans A_2
- On remet 2 comme état acceptant de A car 3 était initial et acceptant dans A_2

Rappel : somme de langages

On s'intéresse au langage :

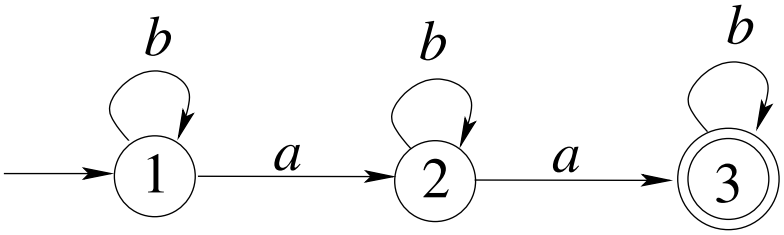
$$L = L_1 + L_2$$

$$L = \{w \text{ tel que } w \in L_1 \text{ ou } w \in L_2\}.$$

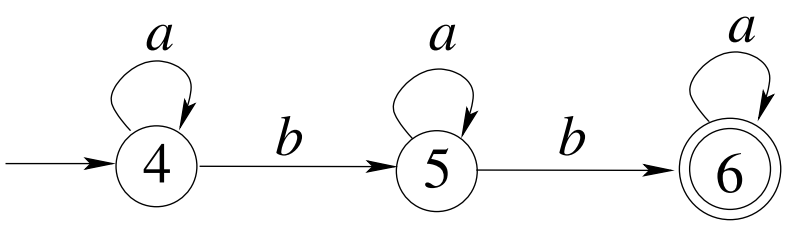
Autrement dit L est l'ensemble des mots qui sont soit des mots de L_1, soit des mots de L_2.

Somme d'automates

- Notons A_1 un automate permettant de reconnaître les mots de L_1, et A_2 un automate permettant de reconnaître les mots de L_2.



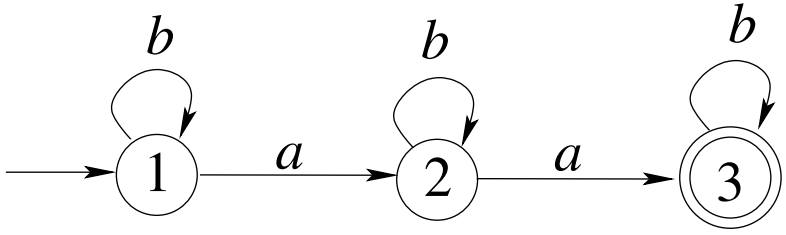
A_1 reconnaissant L_1 = {mots ayant exactement 2 a}



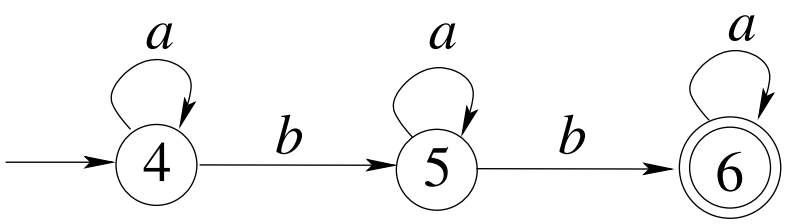
A_2 reconnaissant L_2 = {mots ayant exactement 2 b}

Somme d'automates

- Pour reconnaître les mots de $L = L_1 + L_2$, l'automate A à construire doit alors être conçu comme donnant la possibilité de parcourir soit A_1 soit A_2.



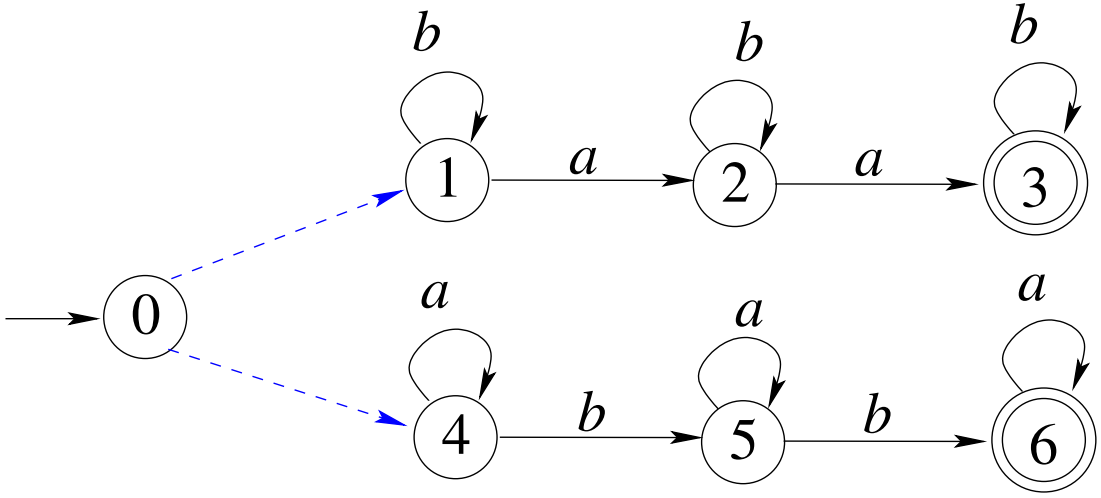
A_1 reconnaissant L_1 = {mots ayant exactement 2 a}



A_2 reconnaissant L_2 = {mots ayant exactement 2 b}

Somme d'automates

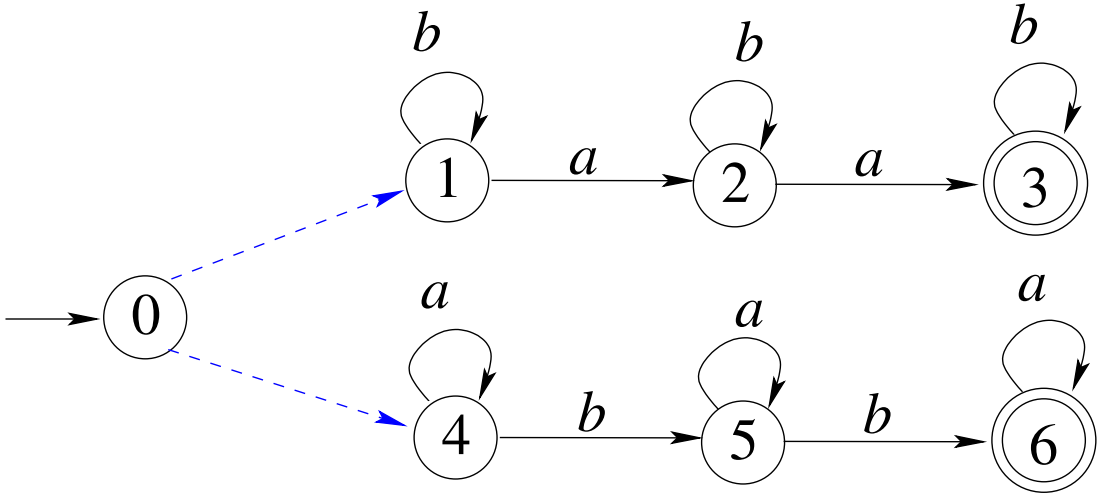
- On aimerait pouvoir réaliser ceci en posant l'un au-dessus de l'autre A_1 et A_2, en ajoutant un nouvel état initial et en reliant celui-ci aux états initiaux de A_1 et A_2 par des transitions vides.



Idée pour A (brouillon)

Somme d'automates

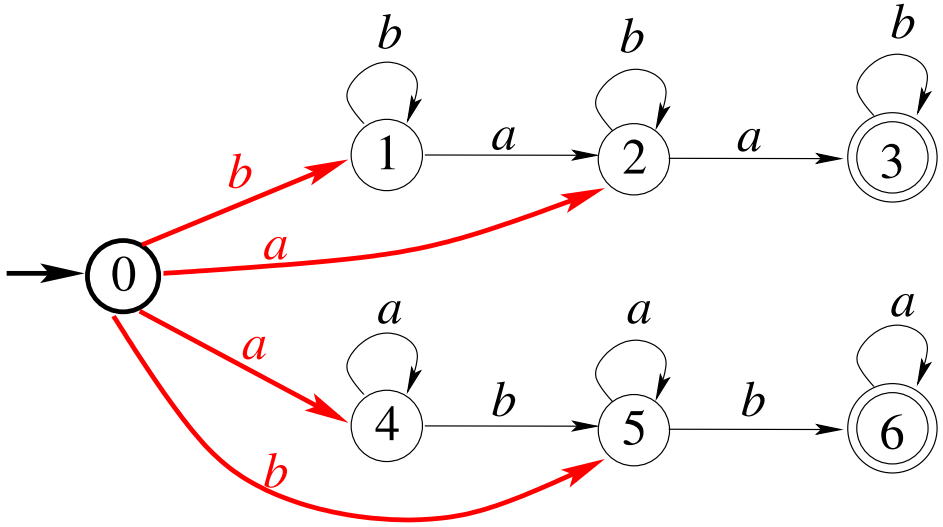
- Mais un automate ne peut pas avoir des transitions vides, donc il va falloir s'en débarrasser.



Idée pour A (brouillon)

Somme d'automates

- Mais un automate ne peut pas avoir des transitions vides, donc il va falloir s'en débarrasser.

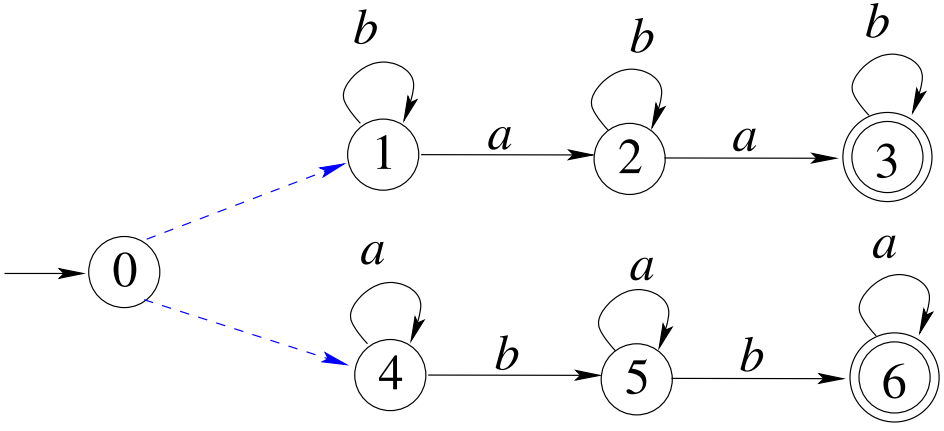


A reconnaissant $L = L_1 + L_2$

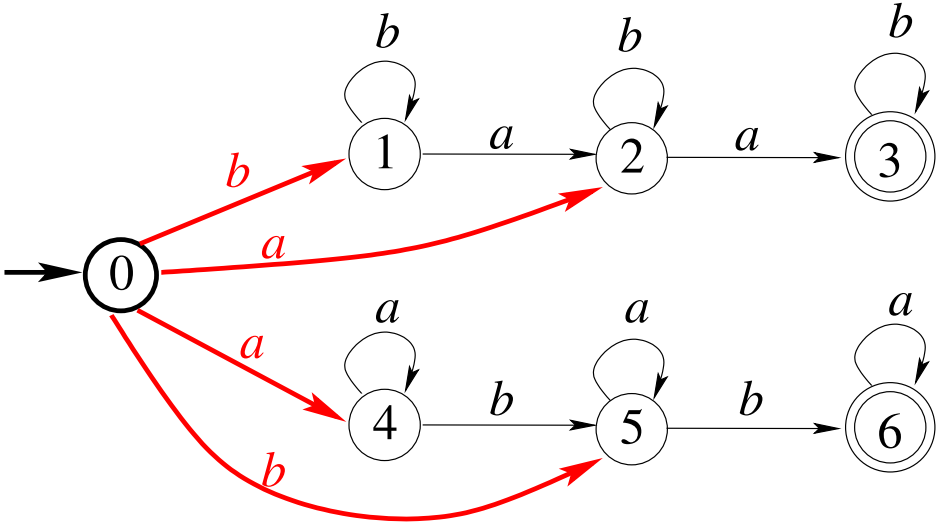
Idée générale

- Appelons i, i_1 et i_2 les états initiaux de A, A_1 et A_2.
- On remplace la transition vide de i vers i_1 par :
  - $i \xrightarrow{a} d$ s'il existe $i_1 \xrightarrow{a} d$ dans A_1 (car passer de i à i_1 sans rien faire puis de i_1 à d en faisant un a est équivalent à faire un a pour aller directement de i à d).
  - i sera terminal dans A si i_1 est terminal (car passer de i à i_1 sans rien faire, puis sortir directement de A_1 est équivalent à sortir directement de A par i).
- On fait pareil pour remplacer la transition vide de i vers i_2.

Exemple

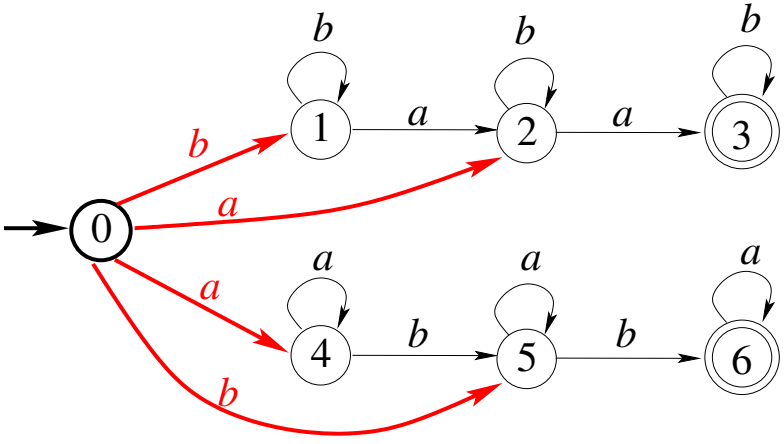


Idée pour A



A reconnaissant $L = L_1 + L_2$

Exemple



A reconnaissant $L = L_1 + L_2$

Ce qu'on a fait :

- on pose A_1 et A_2 l'un au-dessus de l'autre ;
- les états 1 et 4 ne sont plus initiaux ;
- on ajoute le nouvel état initial 0 ;
- on ajoute les transitions :
  - $0 \xrightarrow{b} 1$ car il y avait $1 \xrightarrow{b} 1$ dans A_1
  - $0 \xrightarrow{a} 2$ car il y avait $1 \xrightarrow{a} 2$ dans A_1
  - $0 \xrightarrow{a} 3$ car il y avait $3 \xrightarrow{a} 3$ dans A_2
  - $0 \xrightarrow{b} 4$ car il y avait $3 \xrightarrow{b} 4$ dans A_2

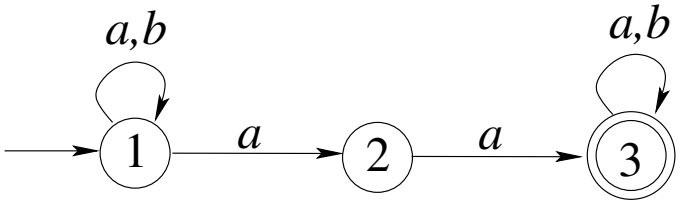
Méthode point par point

- On pose les deux automates l'un au dessus de l'autre
- Les états initiaux i_1 et i_2 des automates A_1 et A_2 ne sont plus initiaux pour le nouvel automate A
- On ajoute un état i qui sera l'état initial du nouvel automate
- On ajoute les transitions nécessaires :
  (en notant d_1 un état de A_1 et d_2 un état de A_2)
  - si $i_1 \xrightarrow{a} d_1$ alors on ajoute une transition $i \xrightarrow{a} d_1$
  - si $i_2 \xrightarrow{a} d_2$ alors on ajoute une transition $i \xrightarrow{a} d_2$
  - si i_1 ou i_2 est également terminal alors i aussi

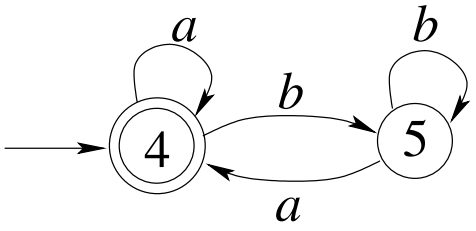
Méthode point par point

- On pose les deux automates l'un au dessus de l'autre
- Les états initiaux i_1 et i_2 des automates A_1 et A_2 ne sont plus initiaux pour le nouvel automate A
- On ajoute un état i qui sera l'état initial du nouvel automate
- On ajoute les transitions nécessaires :
  (en notant d_1 un état de A_1 et d_2 un état de A_2)
  - si $i_1 \xrightarrow{a} d_1$ alors on ajoute une transition $i \xrightarrow{a} d_1$
  - si $i_2 \xrightarrow{a} d_2$ alors on ajoute une transition $i \xrightarrow{a} d_2$
  - si i_1 ou i_2 est également terminal alors i aussi

Autre exemple



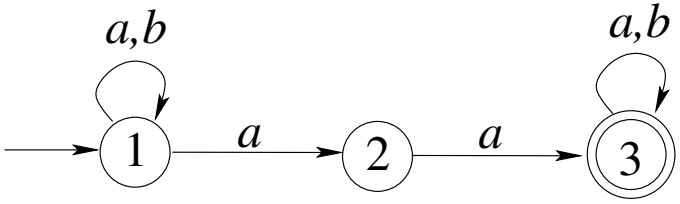
A_1 reconnaissant
L_1 = {mots ayant 2 a consécutifs}



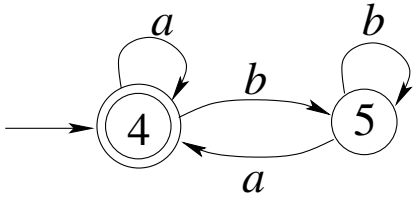
A_2 reconnaissant
$L_2 = \{\epsilon\} \cup \{$ mots terminant par $a\}$

Que proposez-vous ?

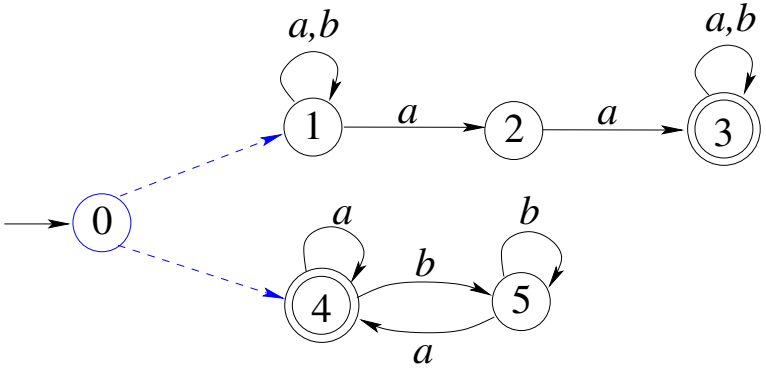
Autre exemple



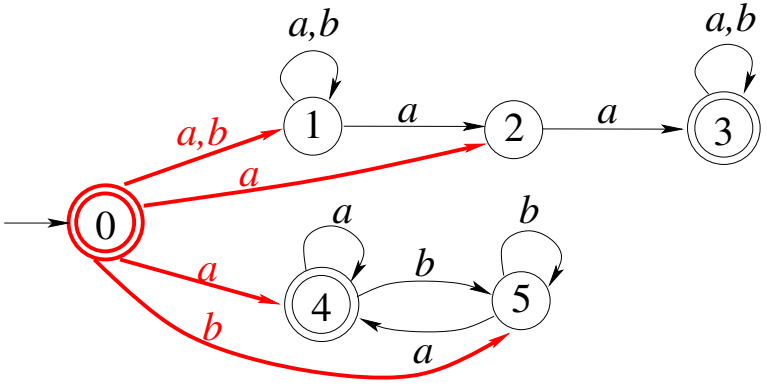
A_1 reconnaissant
L_1 = {mots ayant 2 a consécutifs}



A_2 reconnaissant
$L_2 = \{\epsilon\} \cup \{$ mots terminant par $a\}$



idée pour A



A reconnaissant $L = L_1 + L_2$

Rappel : Étoile d'un langage

On s'intéresse au langage :

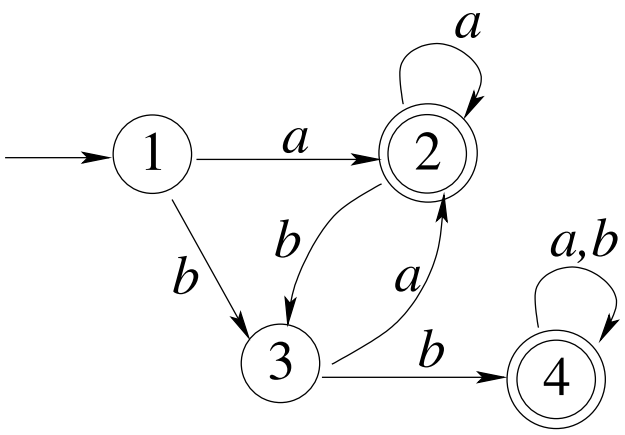
$$L = L_1^*$$

$$L = \{w_1 w_2 \dots w_i \text{ tel que } w_1, w_2, \dots w_i \in L_1 \text{ et } 0 \leqslant i\}.$$

Autrement dit L est l'ensemble des mots formés comme une concaténation de mots de L_1 (0 ou plusieurs) les uns à la suite des autres.

Étoile d'un automate

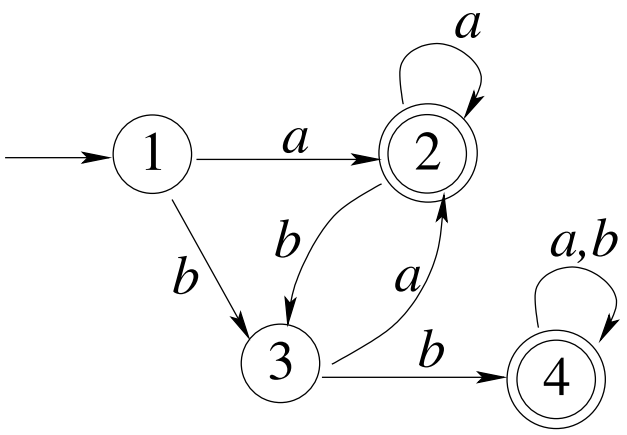
- Soit A_1 l'automate permettant de reconnaître les mots de L_1.



A_1 reconnaissant L_1

Étoile d'un automate

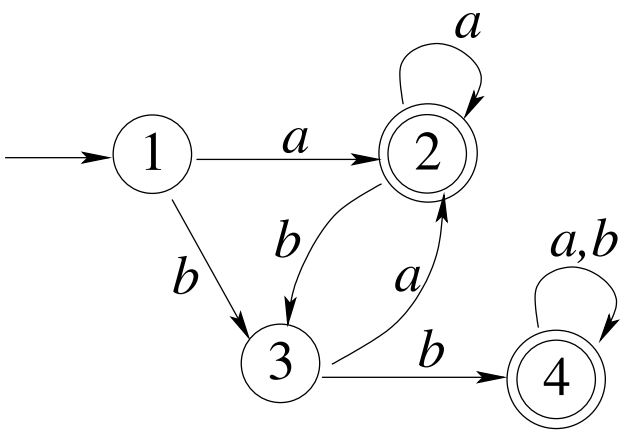
- On a donc envie de pouvoir :
  - sortir tout de suite et reconnaître ainsi le mot vide (ϵ),
  - ou parcourir l'automate autant de fois qu'on veut pour concaténer plusieurs mots de A_1 et ainsi former notre mot.



A_1 reconnaissant L_1

Étoile d'un automate

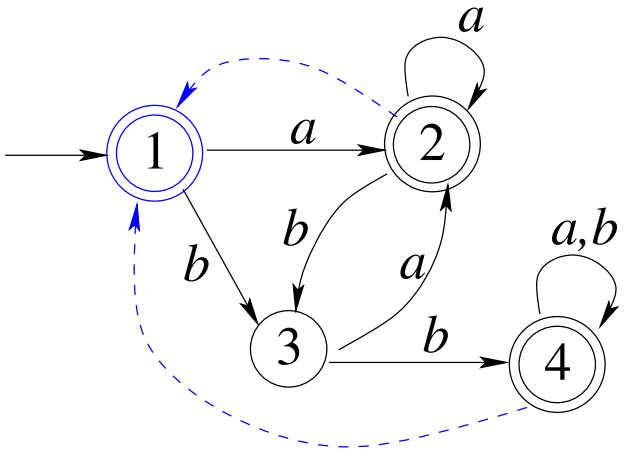
- Pour reconnaître les mots de L, l'automate A à construire doit alors être conçu comme « une boucle » de A_1 sur lui-même, permettant de sortir dès l'état initial.



A_1 reconnaissant L_1

Étoile d'un automate

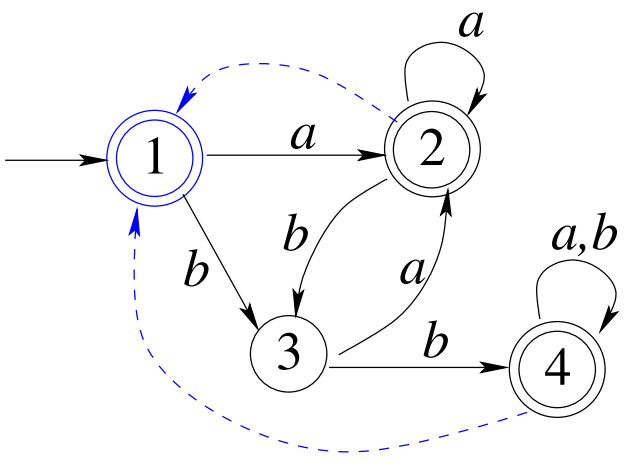
- On aimerait pouvoir réaliser cette boucle en reliant les états terminaux de A_1 à l'état initial de A_1 par des transitions vides (sans étiquette) et en faisant de l'état initial i_1 de A_1 un état également terminal.



idée pour A

Étoile d'un automate

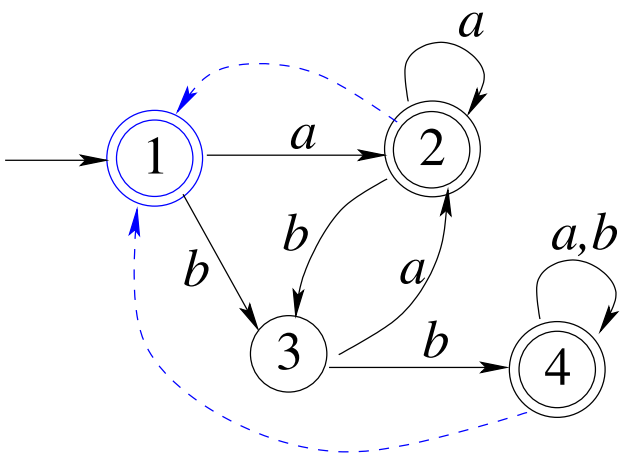
- Ensuite, si la vie était simple, on n'aurait plus qu'à remplacer les transitions vides par des transitions étiquetées comme dans les constructions précédentes.



idée pour *A*

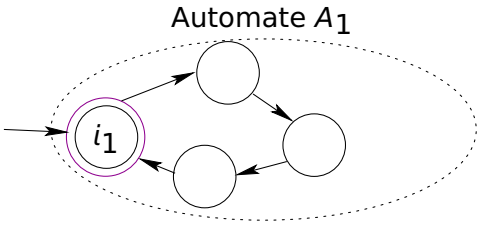
Étoile d'un automate

- Mais en réalité c'est un peu plus compliqué : il peut être nécessaire de passer par une étape préalable de standardisation.



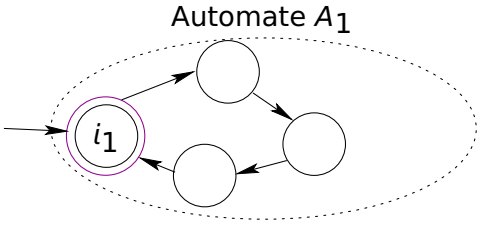
idée pour A

Problème



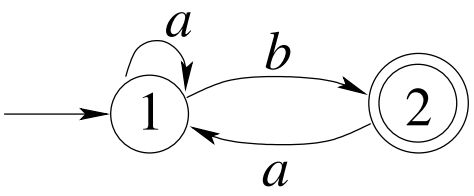
- On transforme l'état initial i_1 en état acceptant pour pouvoir accepter le mot vide.
- Mais il ne faut pas que cette transformation permette de sortir de l'automate au milieu de la reconnaissance d'un mot de L_1.
- Donc si i_1 est aussi une étape intermédiaire lors de la reconnaissance d'un mot de A_1, il faut tout d'abord transformer A_1 pour différencier l'étape initiale des autres étapes se réalisant en cet état.

Problème



- On transforme l'état initial i_1 en état acceptant pour pouvoir accepter le mot vide.
- Mais il ne faut pas que cette transformation permette de sortir de l'automate au milieu de la reconnaissance d'un mot de L_1.
- Donc si i_1 est aussi une étape intermédiaire lors de la reconnaissance d'un mot de A_1, il faut tout d'abord transformer A_1 pour différencier l'étape initiale des autres étapes se réalisant en cet état.

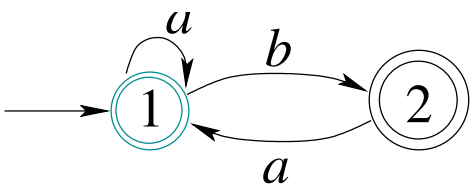
Standardisation



1 est un état initial par lequel on peut repasser lors du parcours de l'automate pour reconnaître un mot. En faisant de 1 un état également terminal, on permettrait de sortir n'importe quand au milieu d'un tel parcours, ce qu'on ne veut pas.

Pour que ce ne soit pas le cas on « éclate » l'état 1 en deux états : 1 et 1', et on relie 1' à tous les états auxquels 1 était lié (recopiage des transitions partant de 1).

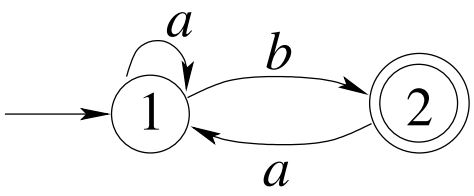
Standardisation



1 est un état initial par lequel on peut repasser lors du parcours de l'automate pour reconnaître un mot. En faisant de 1 un état également terminal, on permettrait de sortir n'importe quand au milieu d'un tel parcours, ce qu'on ne veut pas.

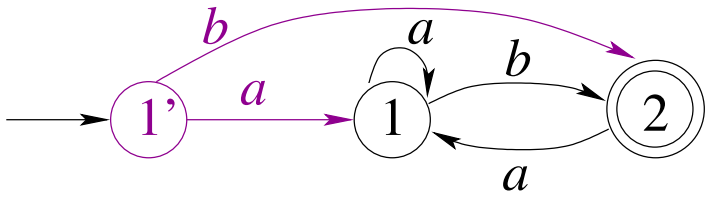
Pour que ce ne soit pas le cas on « éclate » l'état 1 en deux états : 1 et 1', et on relie 1' à tous les états auxquels 1 était lié (recopiage des transitions partant de 1).

Standardisation

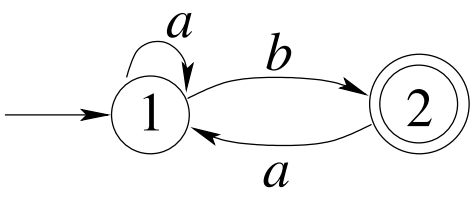


1 est un état initial par lequel on peut repasser lors du parcours de l'automate pour reconnaître un mot. En faisant de 1 un état également terminal, on permettrait de sortir n'importe quand au milieu d'un tel parcours, ce qu'on ne veut pas.

Pour que ce ne soit pas le cas on « éclate » l'état 1 en deux états : 1 et 1', et on relie 1' à tous les états auxquels 1 était lié (recopiage des transitions partant de 1).

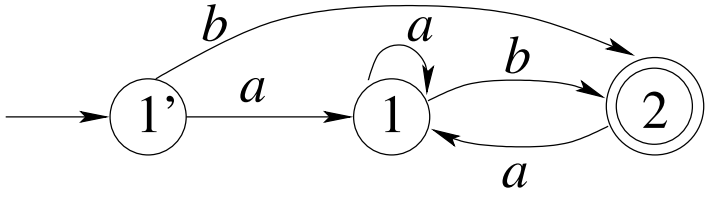


Standardisation



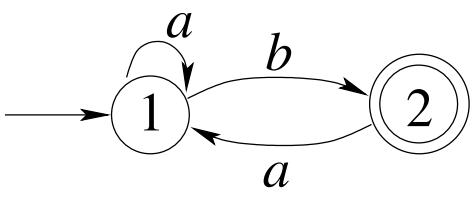
1 est un état initial par lequel on peut repasser lors du parcours de l'automate pour reconnaître un mot. En faisant de 1 un état également terminal, on permettrait de sortir n'importe quand au milieu d'un tel parcours, ce qu'on ne veut pas.

Pour que ce ne soit pas le cas on « éclate » l'état 1 en deux états : 1 et 1', et on relie 1' à tous les états auxquels 1 était lié (recopiage des transitions partant de 1).



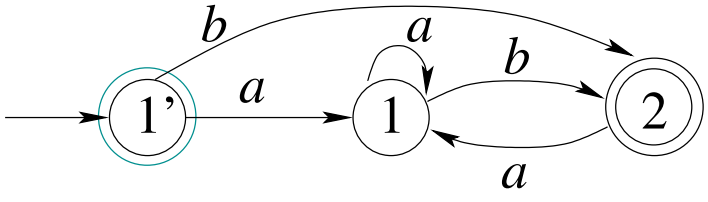
À partir de ce dernier automate, si on tranforme 1' en état également terminal, on permet bien de reconnaître le mot vide ou un mot de L_1 et non pas autre chose.

Standardisation



1 est un état initial par lequel on peut repasser lors du parcours de l'automate pour reconnaître un mot. En faisant de 1 un état également terminal, on permettrait de sortir n'importe quand au milieu d'un tel parcours, ce qu'on ne veut pas.

Pour que ce ne soit pas le cas on « éclate » l'état 1 en deux états : 1 et 1', et on relie 1' à tous les états auxquels 1 était lié (recopiage des transitions partant de 1).

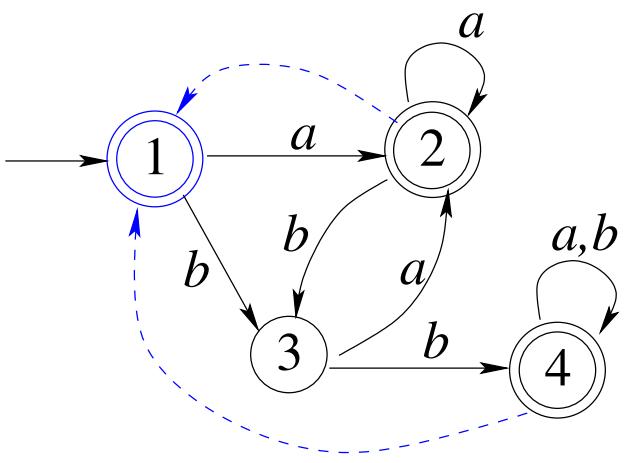


À partir de ce dernier automate, si on tranforme 1' en état également terminal, on permet bien de reconnaître le mot vide ou un mot de L_1 et non pas autre chose.

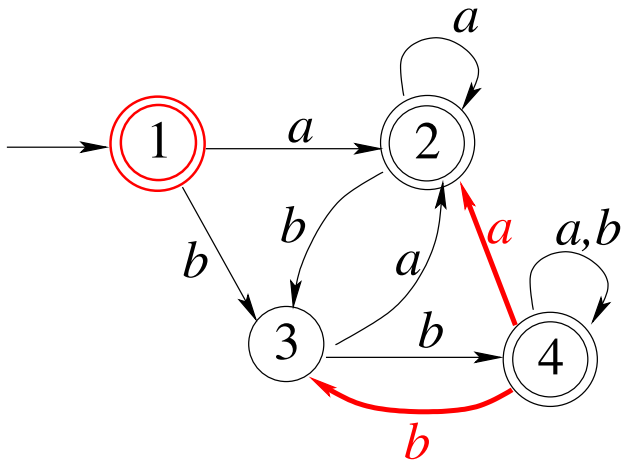
Retour à la construction

- Un automate ne peut pas avoir des transitions vides, donc nous devons remplacer ces transitions vides par ce qui pourrait leur être équivalent
- Sur un automate standard, on procède de la même manière que dans le cas de la concaténation ou de la somme :
  - ne faire rien entre t_1 et i_1 suivi d'un a ($a \in \Sigma$) pour aller de i_1 vers un état d de A_1 est équivalent à aller directement de t_1 vers d par une transition étiquetée a.

Exemple



idée pour A



A reconnaissant $L = L_1^*$

Méthode point par point

- Rendre A_1 **standard** en faisant en sorte qu'aucune transition ne revienne sur i_1 : si besoin dupliquer l'état initial ainsi que les transitions qui en partent.
- L'état initial i_1 de A_1 devient également un état terminal (car $\epsilon \in L_1^*$).
- Pour chaque lettre a et chaque état q
  - si $i \xrightarrow{a} q$ est une transition de A_1
  - alors on ajoute les transitions $t \xrightarrow{a} q$ pour chaque état terminal t.

Méthode point par point

- Rendre A_1 **standard** en faisant en sorte qu'aucune transition ne revienne sur i_1 : si besoin dupliquer l'état initial ainsi que les transitions qui en partent.
- L'état initial i_1 de A_1 devient également un état terminal (car $\epsilon \in L_1^*$).
- Pour chaque lettre a et chaque état q
  - si $i \xrightarrow{a} q$ est une transition de A_1
  - alors on ajoute les transitions $t \xrightarrow{a} q$ pour chaque état terminal t.

Méthode point par point

- Rendre A_1 **standard** en faisant en sorte qu'aucune transition ne revienne sur i_1 : si besoin dupliquer l'état initial ainsi que les transitions qui en partent.
- L'état initial i_1 de A_1 devient également un état terminal (car $\epsilon \in L_1^*$).
- Pour chaque lettre a et chaque état q
  - si $i \xrightarrow{a} q$ est une transition de A_1
  - alors on ajoute les transitions $t \xrightarrow{a} q$ pour chaque état terminal t.

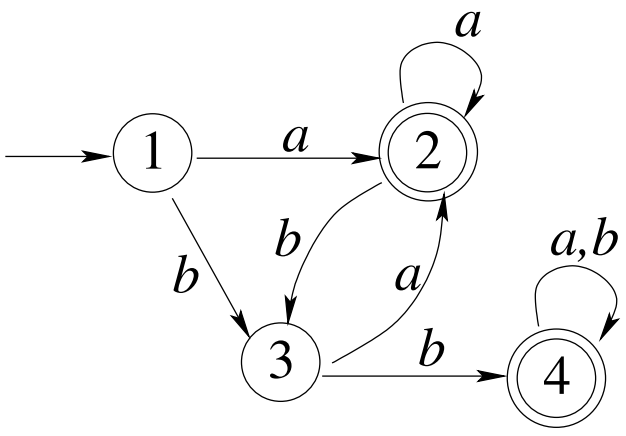
Méthode point par point

- Rendre A_1 **standard** en faisant en sorte qu'aucune transition ne revienne sur i_1 : si besoin dupliquer l'état initial ainsi que les transitions qui en partent.
- L'état initial i_1 de A_1 devient également un état terminal (car $\epsilon \in L_1^*$).
- Pour chaque lettre a et chaque état q
  - si $i \xrightarrow{a} q$ est une transition de A_1
  - alors on ajoute les transitions $t \xrightarrow{a} q$ pour chaque état terminal t.

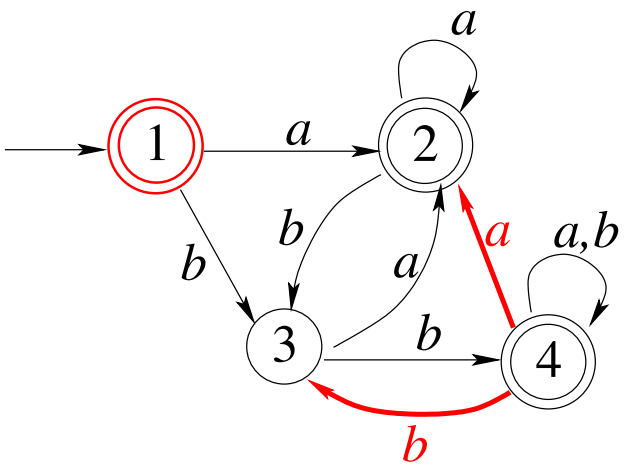
Méthode point par point

- Rendre A_1 **standard** en faisant en sorte qu'aucune transition ne revienne sur i_1 : si besoin dupliquer l'état initial ainsi que les transitions qui en partent.
- L'état initial i_1 de A_1 devient également un état terminal (car $\epsilon \in L_1^*$).
- Pour chaque lettre a et chaque état q
  - si $i \xrightarrow{a} q$ est une transition de A_1
  - alors on ajoute les transitions $t \xrightarrow{a} q$ pour chaque état terminal t.

Exemple



A_1 reconnaissant L_1

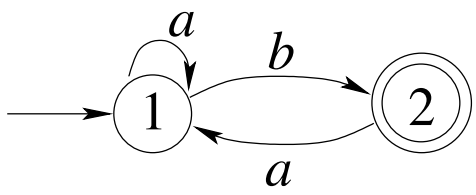


A reconnaissant $L = L_1^*$

Ce qu'on a fait :

- A_1 est déjà standard (aucune transition ne revient sur l'état initial). L'état initial devient aussi terminal.
- on ajoute $4 \xrightarrow{b} 3$ et $2 \xrightarrow{b} 3$ car il y a dans A_1 $1 \xrightarrow{b} 3$
- on ajoute $4 \xrightarrow{a} 2$ et $2 \xrightarrow{a} 2$ car il y a dans A_1 $1 \xrightarrow{a} 2$

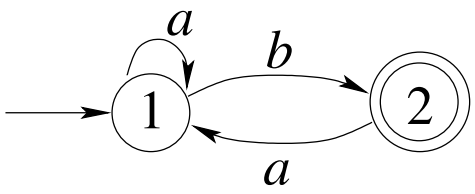
Autre exemple



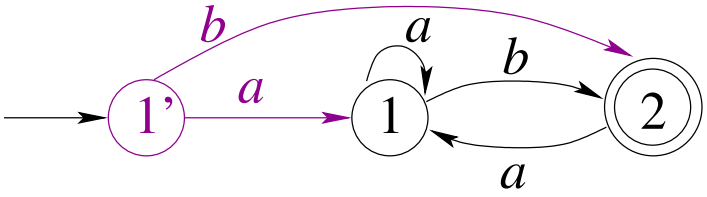
A_1

Que proposez-vous ?

Autre exemple

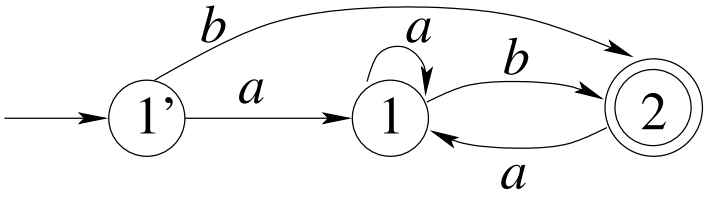


A_1 non standard

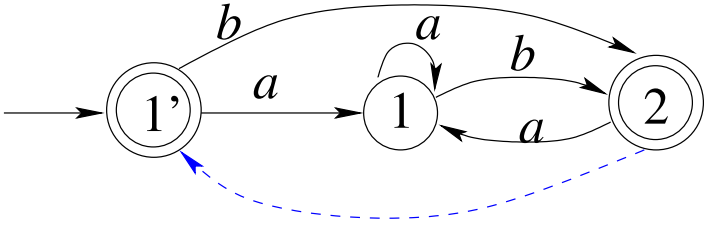


A_1 standard

Autre exemple

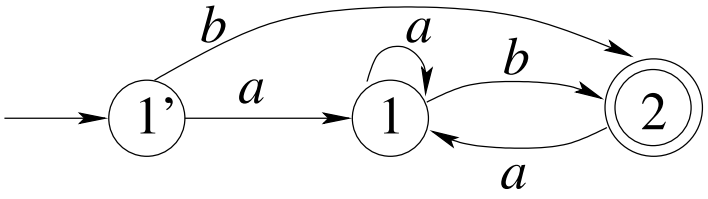


A_1 standard

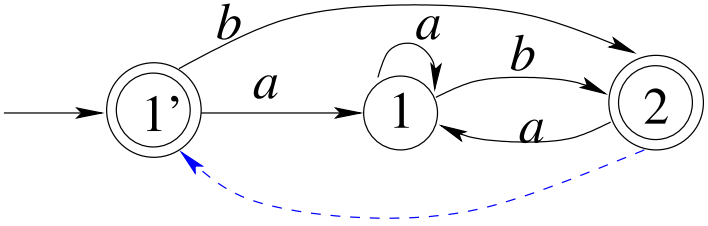


idée pour A

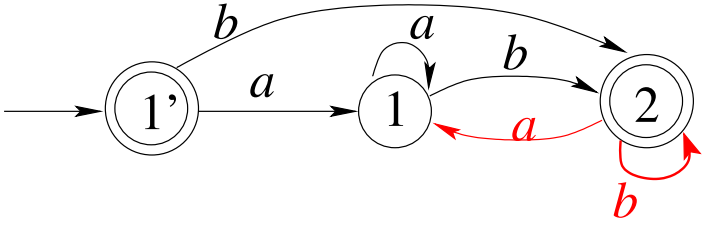
Autre exemple



A_1 standard

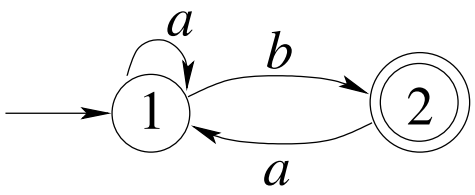


idée pour A

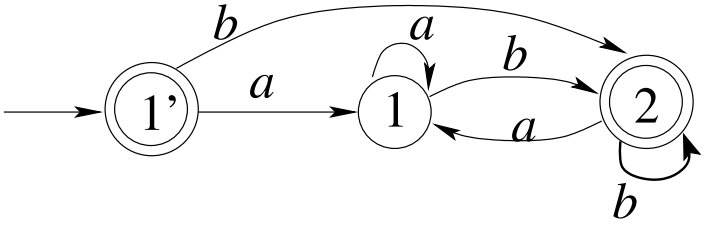


A reconnaissant $L = L_1^*$

Autre exemple



A_1 non standard

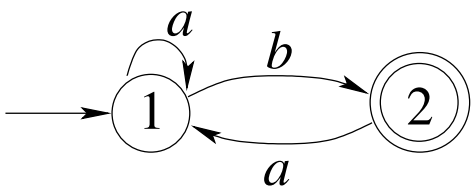


A reconnaissant $L = L_1^*$

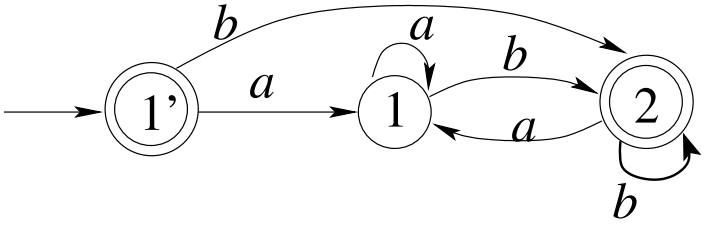
Ce qu'on a fait :

- On standardise A_1 (en dupliquant 1 en 1 et 1') ainsi que les transitions qui en partent (ajout de $1' \xrightarrow{a} 1$ et $1' \xrightarrow{b} 2$ car il y avait $1 \xrightarrow{a} 1$ et $1 \xrightarrow{b} 2$).
- On rend l'état initial de A aussi un état terminal.
- On ajoute $2 \xrightarrow{b} 2$ et $2 \xrightarrow{a} 1$ car il y a dans A_1 $1' \xrightarrow{b} 2$ et $1' \xrightarrow{a} 1$.

Autre exemple



A_1 non standard

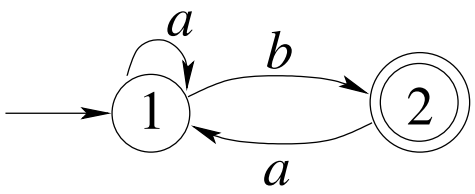


A reconnaissant $L = L_1^*$

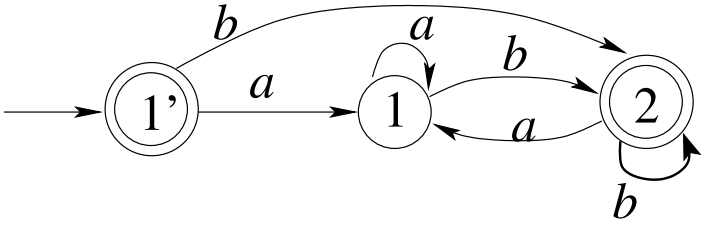
Ce qu'on a fait :

- On standardise A_1 (en dupliquant 1 en 1 et 1') ainsi que les transitions qui en partent (ajout de $1' \xrightarrow{a} 1$ et $1' \xrightarrow{b} 2$ car il y avait $1 \xrightarrow{a} 1$ et $1 \xrightarrow{b} 2$).
- On rend l'état initial de A aussi un état terminal.
- On ajoute $2 \xrightarrow{b} 2$ et $2 \xrightarrow{a} 1$ car il y a dans A_1 $1' \xrightarrow{b} 2$ et $1' \xrightarrow{a} 1$.

Autre exemple



A_1 non standard

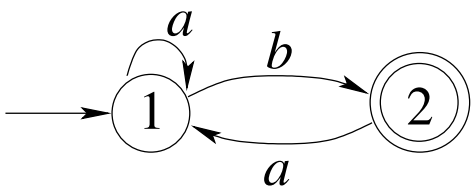


A reconnaissant $L = L_1^*$

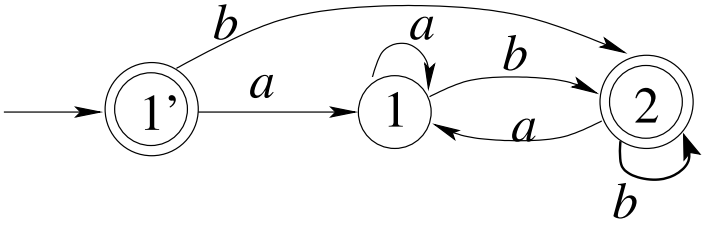
Ce qu'on a fait :

- On standardise A_1 (en dupliquant 1 en 1 et 1') ainsi que les transitions qui en partent (ajout de $1' \xrightarrow{a} 1$ et $1' \xrightarrow{b} 2$ car il y avait $1 \xrightarrow{a} 1$ et $1 \xrightarrow{b} 2$).
- On rend l'état initial de A aussi un état terminal.
- On ajoute $2 \xrightarrow{b} 2$ et $2 \xrightarrow{a} 1$ car il y a dans A_1 $1' \xrightarrow{b} 2$ et $1' \xrightarrow{a} 1$.

Autre exemple



A_1 non standard

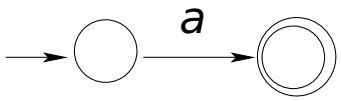


A reconnaissant $L = L_1^*$

Ce qu'on a fait :

- On standardise A_1 (en dupliquant 1 en 1 et 1') ainsi que les transitions qui en partent (ajout de $1' \xrightarrow{a} 1$ et $1' \xrightarrow{b} 2$ car il y avait $1 \xrightarrow{a} 1$ et $1 \xrightarrow{b} 2$).
- On rend l'état initial de A aussi un état terminal.
- On ajoute $2 \xrightarrow{b} 2$ et $2 \xrightarrow{a} 1$ car il y a dans A_1 $1' \xrightarrow{b} 2$ et $1' \xrightarrow{a} 1$.

Comment faire ?

- On analyse la structure de l'expression rationnelle qu'on veut transformer en automate.
- Pour les langages d'une lettre on a facilement un automate.
- Par exemple, pour la lettre a : 

- On utilise nos constructions pour $.$ $*$ et $+$ pour combiner des automates de plus en plus complexes.

Exemple

Question

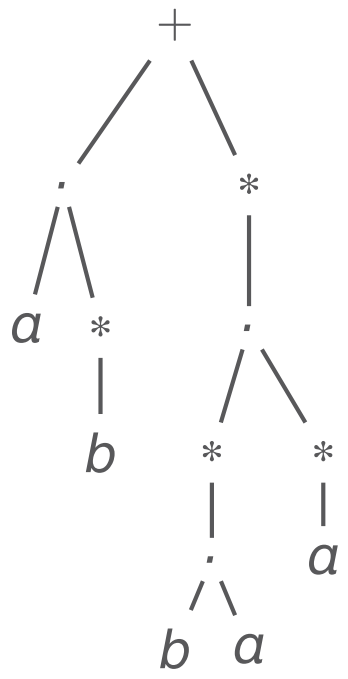
Construire un automate qui reconnaît le langage $ab^* + ((ba)^*a^*)^*$.

Toutes les étapes : cf correction au tableau
Automate final

Exemple

Question

Construire un automate qui reconnaît le langage $ab^* + ((ba)^*a^*)^*$.



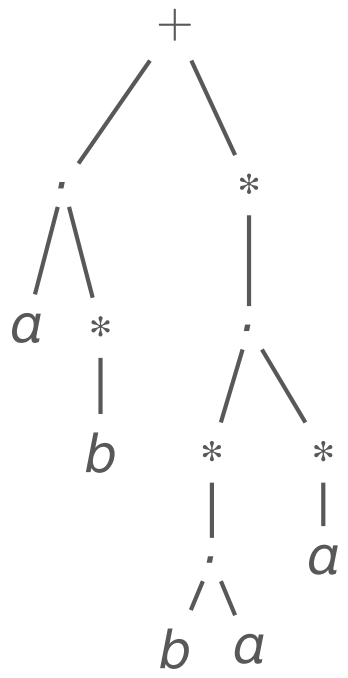
Toutes les étapes : cf correction au tableau

Automate final

Exemple

Question

Construire un automate qui reconnaît le langage $ab^* + ((ba)^*a^*)^*$.



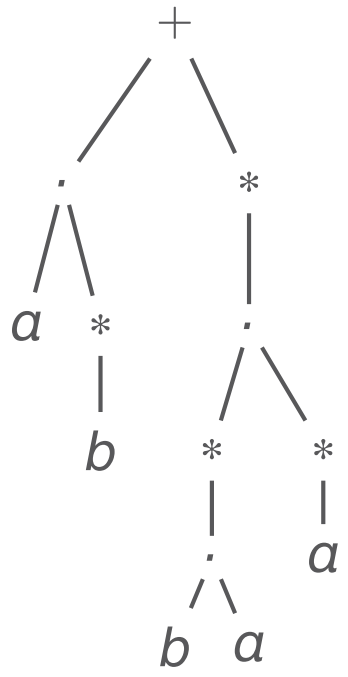
Toutes les étapes : cf correction au tableau

Automate final

Exemple

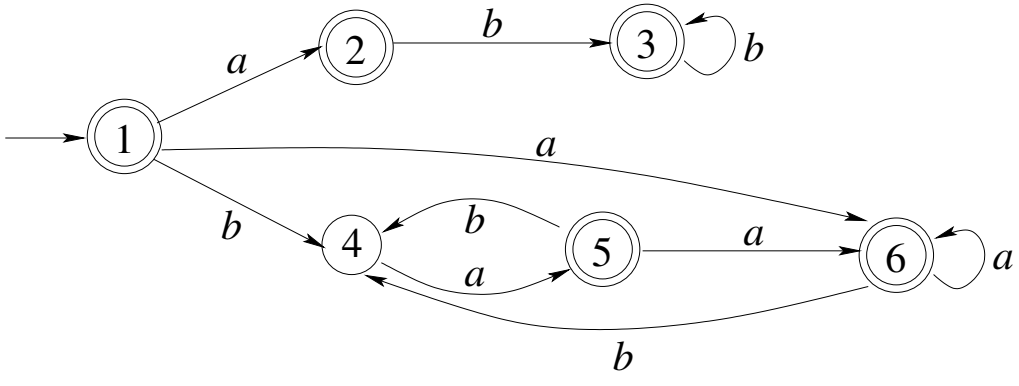
Question

Construire un automate qui reconnaît le langage $ab^* + ((ba)^*a^*)^*$.



Toutes les étapes : cf correction au tableau

Automate final



Conclusion

Aujourd'hui

- Technique 1 Comment « concaténer »deux automates en transformant des transitions vides.
- Technique 2 Comment « faire l'union » de 2 automates.
- Technique 3 Comment « calculer l'étoile »d'un automate après l'avoir standardisé si nécessaire.
- Techniques 1, 2 & 3 Comment synthétiser un automate pour n'importe quelle expression rationnelle.