Chap. 2 Expressions rationnelles

Il existe un autre moyen de définir les langages reconnaissables. Sont reconnaissables les langages qui peuvent être exprimés par les expressions rationnelles (voir Théorème de Kleene ci-dessous).

Expression rationnelle, définition :

On construit une expression rationnelle de la façon suivante :
Soit *A* un alphabet fini, non vide. Nous avons déjà défini *A** comme ensemble de tous les mots sur l'alphabet *A* plus le mot vide. On définit deux *formules atomiques :* ∅ et ε, où ∅ correspond à un langage vide, et ε est le mot vide (qu'on confond avec le langage qui contient un seul mot, ε).

- ∅ est une expression rationnelle
- le mot vide ε est une expression rationnelle
- toute lettre $a \in A$ est une expression rationnelle
- si X, Y sont des expressions rationnelles, alors X + Y, XY, X* sont des expressions rationnelles.

Notation

On définit trois opérations :

- X + Y désigne l'union des expressions X et Y.
  X + Y = {u| u∈X ou u∈Y}
- XY désigne le produit (concaténation, parfois notée avec un point comme dans l'algèbre normale)
  XY={uv| u∈X, v∈Y}
- L'étoile désigne une séquence de longueur arbitraire (y compris zéro) d'éléments de l'expression étoilée :
  $\mathbf{X^* = \{u_1u_2\ldots u_n |\ n \geq 0,\ u_i \in X\}}$
  (On convient ici que le mot vide appartient à **X* -** cela correspond au cas n=0).
- Règles de priorité : * a priorité sur · qui a priorité sur +. Autrement, on utilise les parenthèses comme dans l'arithmétique.
  Ainsi, a·b + c* = ((a·b) + (c*), a·(b+c)*=(a·((b+c)*), (a·b+c)*=((a·b)+c)*).

Il y a une différence entre les concepts de langage rationnel et expression rationnelle : un langage est un ensemble, et non pas une expression. Par exemple, le langage consistant en deux mots « **papa** » et « **maman** » construit sur l'alphabet français s'écrit L={ **papa**, **maman** }. L'expression rationnelle lui correspondant s'écrit **papa** + **maman**. Une expression rationnelle est donc un moyen de décrire quels mots appartiennent au langage en tant que membres de l'ensemble. Mais nous allons très fréquemment confondre les deux concepts et dire des choses par tout à fait exactes mais très commodes, du genre L = **papa** + **maman.**

Exemple 1 Si **X = ab+ba,** alors **X*** désigne l'ensemble des mots de la forme $w=\prod_{i=1}^{n} u_i$ avec $\mathbf{u_i}$ **= ab** ou $\mathbf{u_i}$ **= ba** pour $\forall n\ 0 \leq n < \infty$. Les mots appartenant à cet ensemble sont, par exemple, **ε, ab, ba, abba, abab, baba, baab**, mais non pas **aabb** ni **abaa**, donc on ne peut pas simplifier l'écriture de cet ensemble comme **(a+b)***.

Exemple 2 Les éditeurs de textes permettent l'usage d'expressions rationnelles pour rechercher une chaîne de caractères dans un fichier.

Dans EMACS on écrit :
[a-c][a-b]* au lieu de **(a+b+c)(a+b)***
. au lieu de la somme de tous les caractères sauf new-line.
x$ au lieu de x suivi du caractère new-line.

Quelques identités rationnelles
L'algèbre des expressions rationnelles est très compliquée. Assez souvent on découvre que deux expressions rationnelles qui ne se ressemblent pas du tout, sont égales. Pour l'établir, on doit le plus fréquemment passer par des automates (voir ci-dessous). Mais on peut quand-même établir quelques identités. Dans ce qui suit, E, F etc. sont des expressions rationnelles.

- Une loi commutative pour + : E + F = F + E
  (Attention : il n'y a pas de loi commutative pour la concaténation ! EF≠FE *!)*
- Une loi associative pour + : (E + F) + G=E + (F + G)
- Une loi associative pour ·: (E F) G=E (F G)
- Une loi distributive : E(F + G)=EF + EG et (E + F)G = EG + FG
- E + ∅ = ∅ + E = E
- E∅=∅E=∅
- Eε=εE=E
- E + E = E
- (E*)*=E*

Quelques identités (liste très partielle !) qui relient l'étoile et les autres opérations :

- (E + F)*=(E*F)*E*=E*(FE*)* (1)
- (EF)*=ε+E(FE)*F (2)
- E* = ε + EE* = ε + E*E (on se rappelle qu'on peut l'écrire E* = ε + E^+) (3)
  (il est très facile de montrer (2) et (3), tandis que (1) est moins évident)

En combinant ces identités, on peut obtenir d'autres. Par exemple, on peut obtenir (a*b)*a^+ + ε =(b*a)* de façon suivante :

(a*b)*a^+ + ε=(a*b)*a*a+ε	utilisons (1) avec E=a, F=b
=(a+b)*a+ε = (b+a)*a+ε	et encore une fois
= (b*a)*b*a+ε	utilisons (3)
=(b*a)*	

Résolution d'équations
Lemme d'Arden, sans demonstration

Soit Y et Z deux parties de A* où Y ne contient pas ε. Alors X=Y*Z est l'unique solution de l'équation
X = YX+Z (on peut écrire X = YX+Z ⇔ X=Y*Z) (4a)
et X=ZY* est l'unique solution de l'équation
X = XY + Z (on peut écrire X =XY+Z ⇔ X=ZY*) (4b)
Remarque Si Y contient ε, une solution de (4a) et de (4b) est donnée par A* ; Y*Z (pour 4a) [ou ZY* (pour 4b)] n'est plus l'unique solution, c'est la plus petite solution de (4a) [ou (4b)].

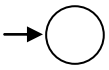
Théorème de KLEENE (les années 50).

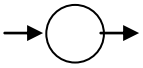
Un langage est reconnaissable par automate fini *ssi* il peut être décrit par une expression rationnelle.

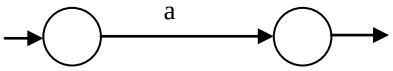
La preuve doit se faire séparément dans chacune des directions en donnant un algorithme pour passer des expressions rationnelles aux automates finis et inversement.

Expression rationnelle → Automate fini

On construit pour chaque expression rationnelle un automate asynchrone qui reconnaît le langage défini par cette expression.

Pour x=∅, on marque  (cet automate ne reconnaît *aucun* mot)

Pour x = ε, on marque : 

Pour x = a, a∈A, on marque : 

En ce qui concerne le reste des opérations, on doit choisir un jeu de règles bien défini, sinon on ne pourra pas programmer la construction d'un automate suivant une expression rationnelle. Le choix de ce jeu de règles est assez libre, mais une fois fait, il faut le respecter. Dans ce cours, on impose la règle selon laquelle

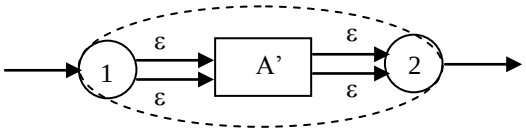
1. dans tout état entrent deux flèches au maximum,
2. de tout état sortent deux flèches au maximum,
3. tout automate a un unique état initial
4. tout automate a un unique état terminal.

D'autres éléments des règles figurent ci-dessous.

Opérations +, produit (concaténation) et * :

Soient X, Y deux expressions rationnelles. On suppose déjà construits pour X et Y les automates A et B qui vérifient les trois règles que nous venons de spécifier. On suppose qu'aucune flèche étiquetée ne sort de l'état terminal (ici on ne parle pas de la flèche sans étiquette qu'on utilise pour marquer l'état final, dont la signification est tout à fait différente).

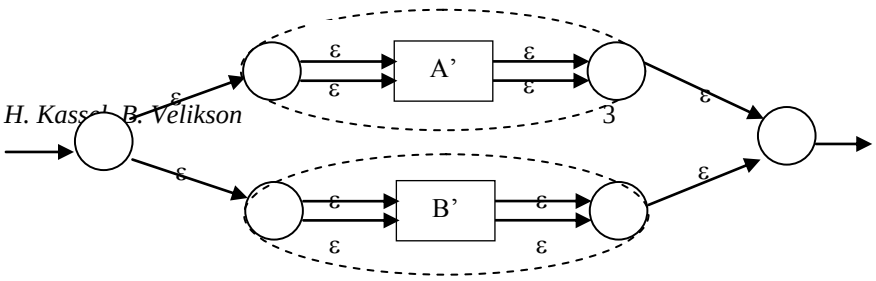
On notera les automates A et B de la façon suivante :



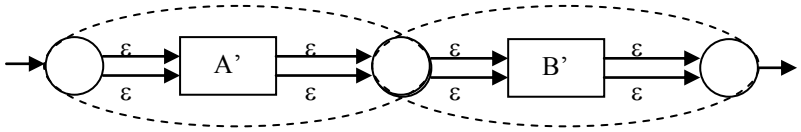
de telle façon qu'on puisse voir explicitement ses états initial et terminal sans plus de détail pour d'autres états, et où A est le nom de tout l'automate, et A' celui de la « boite noire » à l'intérieur. Les deux flèches entre 1 et A', ainsi qu'entre A' et 2, correspondent au nombre maximum de flèches - il peut n'y avoir qu'une seule.

Nous allons construire des automates ayant les mêmes quatre propriétés pour X+Y, XY et X*.

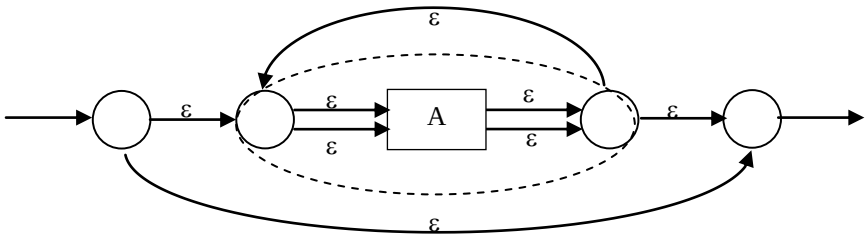
Pour X+Y on prend l'automate asynchrone suivant :



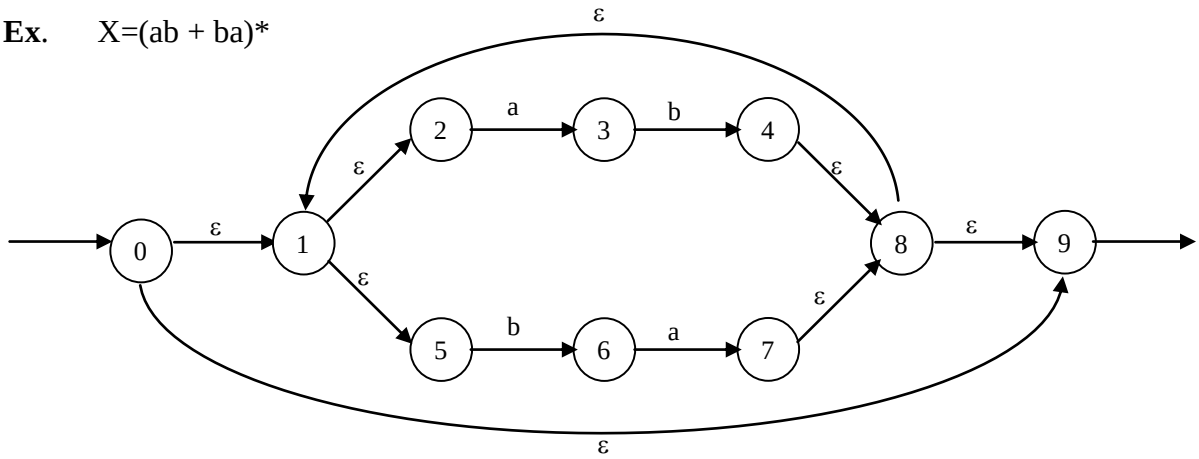
Pour XY on prend l'automate asynchrone suivant, obtenu en confondant l'état initial de B avec l'état terminal de A :



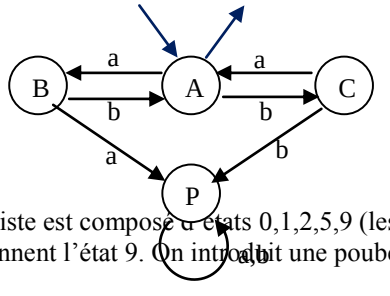
Pour X* on prend l'automate asynchrone suivant :



Ex. X=(ab + ba)*



Cet automate asynchrone peut être déterminisé et minimisé, en produisant un automate à trois états + poubelle :



Voici comment :

Déterminisation

L'état initial de l'automate déterministe est composé d'états 0,1,2,5,9 (les états où on peut arriver en lisant le mot vide). Les états terminaux sont ceux qui contiennent l'état 9. On introduit une poubelle pour que l'automate déterministe obtenu soit complet.

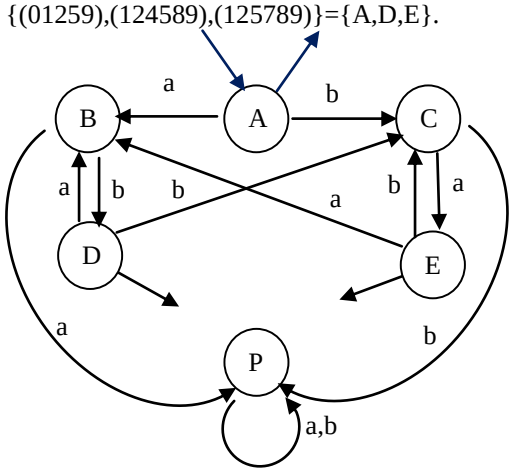
Ou bien, en modifiant la notation,

Etat	a	B
(01259)	3	6
3	P	(124589)
6	(125789)	P
(124589)	3	6
(125789)	3	6

Etat	a	b
A	B	C
B	P	D
C	E	P
D	B	C
E	B	C

L'état initial : I = {(01259)} = {A}, les états terminaux T = {(01259),(124589),(125789)}={A,D,E}.

Voici l'image graphique de l'automate déterminisé :



Minimisation

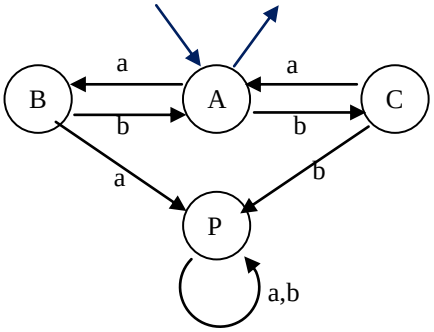
Groupe d'états non terminaux : I = {B,C,P}. Groupe d'états terminaux : II = {A,D,E}.

Etat	a	b	A	b
B	P	D	I	II
C	E	P	II	I
P	P	P	I	I
A	B	C	I	I
D	B	C	I	I
E	B	C	I	I

Les états B, C et P se séparent. Les états A , D et E ne peuvent jamais se séparer car ils mènent aux mêmes états de l'automate initial (B en lisant a, et C en lisant b).
On obtient donc 4 états : B,C,P et (ADE) ; pour simplifier la notation, on va utiliser A au lieu de (ADE).
La table de transitions de l'automate minimal est donc

Etat	a	b
A	B	C
B	P	D
C	E	P
P	P	P

avec un seul état initial et un seul état terminal A. On retrouve donc l'image graphique suivant :



Automate fini → Expression rationnelle

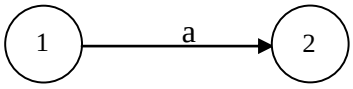
Il y a plusieurs méthodes pour déterminer le langage reconnu par un automate. Ici nous allons présenter deux méthodes, toutes les deux basées sur la construction d'un système d'équations. Les deux systèmes d'équations en résultant s'appellent le **système de l'arrivée** et **le système du départ.**

Tout en présentant les deux méthodes, nous n'allons utiliser que le système de l'arrivée en TD.

On n'a pas besoin de déterminiser l'automate pour utiliser ces méthodes.

A. Système de l'arrivée

- Chaque état représente une expression rationnelle (***langage de l'arrivée de cet état)*** correspondant à l'ensemble des mots qu'on peut lire pour l'atteindre à partir d'un état initial). Par exemple, si on a



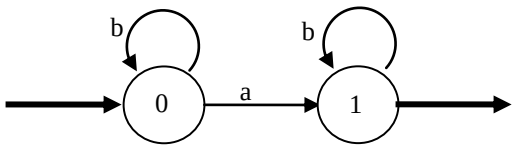
l'état 1 correspond à l'expression rationnelle X_1 et l'état 2 à X_2, alors $X_2 = X_1a$. *(Cela veut dire : l'état 2 lit tous les mots lus par l'état 1 suivis de a).*

- La flèche non étiquetée qui entre dans un état initial correspond à ε. *(Un état d'entrée lit certainement le mot vide !)*
- On peut utiliser la loi distributive (du genre (a+b)c=ac+bc).
- εX=Xε=X
- On peut utiliser la règle (4b) (le lemme d'Arden) :

$$X = XY + Z \Leftrightarrow X = ZY^* \quad (*)$$
$$\varepsilon \notin Y$$

- Le langage reconnu par l'automate est la somme des langages reconnus par ses états terminaux.

Exemple A1 :



Le premier état, qu'on note X0, reçoit les flèches suivantes :

- Une flèche initiale (ε) (il lit le mot vide),
- Une flèche partant de lui-même, marquée b, ce qui donne X0b.

Le second état, noté X1, reçoit une flèche partant de X0, marquée a, ce qui donne X0a, et une partant de lui-même, marquée b, ce qui donne X1b.

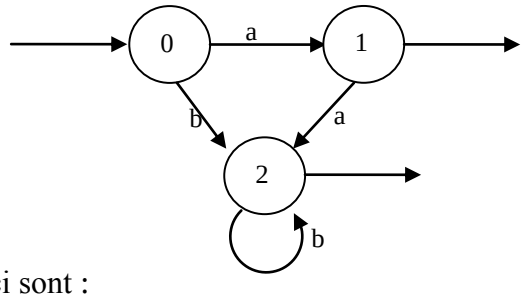
On a donc les équations suivantes :

$X0 = \varepsilon + X0b$ (1)

X1 = X0a + X1b (2)

Le lemme d'Arden appliqué à (1) donne X0 = εb* = b*, puis X1 = X0a + X1b = b*a + X1b, et une nouvelle utilisation du lemme d'Arden donne X1 = b*ab* ; 1 étant le seul état terminal, le langage reconnu par l'automate est L = b*ab*.

Exemple A2:

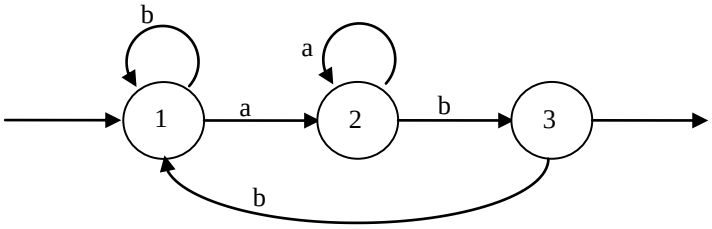


Les équations ici sont :
X0 = ε, (3)
X1 = X0a, (4)
X2 = X0b + X1a + X2b. (5)

On obtient : X1 = X0a = εa = a,
X2 = X0b +X1a + X2b = εb + aa + X2b = X2b + (b+aa) = (b+aa)b*, où on a appliqué le lemme d'Arden.

Le langage reconnu par l'automate est L = X1 + X2 = a + (b+aa)b*.

Exemple A3, plus compliqué :



Le premier état (état 1) reçoit des flèches suivantes :

- Une flèche initiale (ε),
- Une flèche partant de lui-même, marquée b, ce qui donne X1b,
- Une flèche partant de l'état 3, marquée b, ce qui donne X3b.

On obtient donc une équation
X1 = X1b + X3b + ε.
En continuant de la même façon pour deux autres états, on obtient un système d'équations :

X1 = X1b + X3b + ε. (6)
X2 = X1a + X2a (7)
X3 = X2b (8)

Remplaçant (8) dans (6), on obtient

$X1 = X1b + X2bb + \varepsilon$ (9)

Appliquant la règle (*) à (7), on obtient

$X2 = X1a + X2a \quad \Leftrightarrow \quad X2 = X1aa^*$ (10)

Remplaçant X2 dans (9), on obtient

$X1 = X1b + X1aa^*b^2 + \varepsilon$

$= X1(b + aa^* b^2) + \varepsilon$ (11)

Appliquant la règle (*) à (11), on obtient

$X1 = \varepsilon (b + aa^*b^2)^* = (b + aa^*b^2)^*$ (12)

et puis, de (10)

$X2 = (b+aa^*b^2)^*aa^*$ (13)

et de (8)

$X3 = (b + aa^*b^2)^*aa^*b$ (14)

L'état 3 étant l'état terminal, l'automate reconnait le langage décrit par l'expression rationnelle $(b + aa^*b^2)^*aa^*b$.

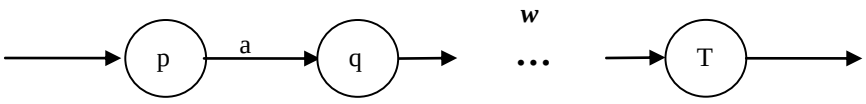
Ce n'est pas la seule forme possible de cette expression! En effectuant les opération dans un ordre différents, on aurait pu obtenir une autre expression équivalente (exprimant le même langage).

B. Système du départ

- Chaque état représente maintenant une expression rationnelle (***langage du départ de cet état)*** correspondant au langage qui serait reconnu par l'automate si cet état était la seule entrée. **Attention ! On ne parle plus du langage « lu par l'état » !**

- Dans cette optique, une sortie correspond au mot vide : si l'état terminal en question est initial, l'automate lit certainement le mot vide.

Le langage correspondant à un état est la somme des mots qui peuvent arriver à une sortie si cet état est initial ; si l'état ***p*** est lié à un autre état, ***q***, par une transition ***p.a.q***, alors, si le mot ***w*** fait partie du langage X_q correspondant à ***q,*** le mot ***aw*** fait partie du langage X_p correspondant à ***p***.



(Ici, ***q*** et ***T*** sont liés par un ***chemin*** $q \xrightarrow{w} T$, ce chemin pouvant consister en plusieurs transitions).

On peut exprimer ceci par une formule :

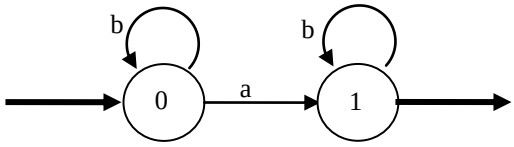
- Dans la méthode du langage du départ, si l'état ***p*** a les transitions $p.a_i.q_i$,l'expression rationnelle lui correspondant est égale à $\sum_i a_i q_i$. Evidemment, cela s'applique, en particulier, dans le cas où un des q_i est l'état ***p*** lui-même. Ainsi, une boucle en ***a*** sur l'état ***p*** correspond au terme ***ap***.

- Comme dans la méthode de l'arrivée, on peut utiliser le lemme d'Arden, mais cette fois dans sa version (4a) :

$$X = YX + Z \Leftrightarrow X = Y^*Z \qquad (**)$$
$$\varepsilon \notin Y$$

- Le langage reconnu par l'automate est la somme des langages reconnus par les états **initiaux**.

Exemple B1 : on prend le même automate que celui de l'exemple A1 :



Le système d'équation s'écrit :

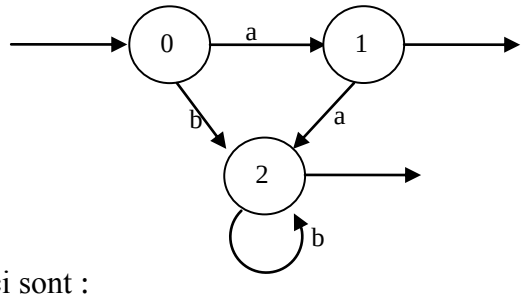
$X0 = aX1 + bX0$ (15)

$X1 = \varepsilon + bX1$ (16)

(16) résulte en $X1 = b^*\varepsilon = b^*$, et en le mettant dans (15), on obtient $X0 = ab^* + bX0$.

En appliquant le lemme d'Arden (**) encore une fois, on a $X0 = b^*ab$. L'état 0 étant la seule entrée, c'est le langage reconnu par l'automate : $L = b^*ab$. Nous avons obtenu exactement la même chose qu'avec la méthode de l'arrivée.

Exemple B2 : on prend le même automate que celui de l'exemple A2 :



Les équations ici sont :

$X0 = aX1 + bX2,$ (17)

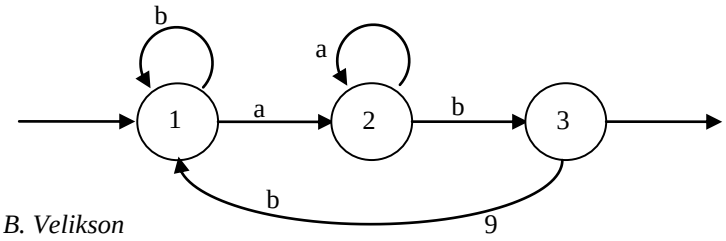
$X1 = \varepsilon + aX2,$ (18)

$X2 = \varepsilon + bX2$. (19)

(19) résulte en $X2 = b^*\varepsilon = b^*$, en le mettant dans (18) on obtient $X1 = \varepsilon + ab^*$, et finalement $X0 = a(\varepsilon + ab^*) + bb^* = a + aab^* + bb^*$, ce qui est la même expression que celle qu'on a obtenu avec la méthode de l'arrivée.

.

Exemple B3 : on prend le même automate que celui de l'exemple A3 :



Les équations:
X3 = ε +bX1, (20)
X2 = aX2 + bX3, (21)
X1 = bX1 + aX2. (22)

En mettant (20) dans (21), on obtient X2 = aX2 + b + bbX1 (23)
En utilisant le lemme d'Arden sur (22), on obtient X1 = b*aX2 (24)
En utilisant (24) dans (23), on obtient X2 = aX2 + b + bbb*aX2 (25)

Réécrivant (25) comme X2 = (a + bbb*a)X2 + b et en utilisant le lemme d'Arden, on obtient X2 = (a + bbb*a)*b.

Donc, X1 = b*a(a + bbb*a)*b, et l'état 1 étant la seule entrée, L = b*a(a + bbb*a)*b. (26)

On a obtenu une expression rationnelle différente de celle obtenue dans l'exemple A3 (éq. 14 : L = (b + aa*b^2)*aa*b.

Peut-on obtenir la même expression que celle de l'éq. 14 ? Dans ce cas particulier, ce n'est pas trop compliqué : il suffit de commencer par l'application du lemme d'Arden à (21) au lieu de (22) :
X2 = a*bX3, et en replaçant X3 par (20), X2=a*b + a*bbX1.
Maintenant remplaçant X2 dans (22), on obtient
X1 = bX1 + a(a*b + a*bbX1) = bX1 + aa*b + aa*bbX1 = (b + aa*bb)X1 + aa*b, d'où

X1 = (b+aa*bb)*aa*b, et on retrouve l'expression (14).

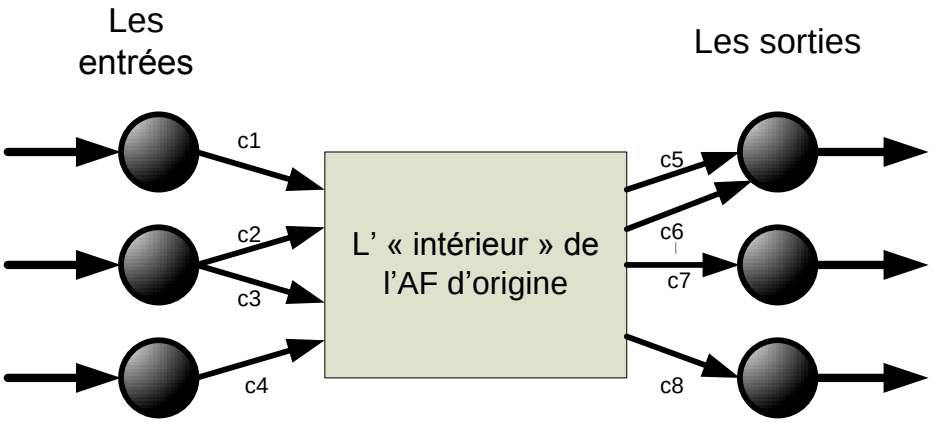
En résolvant un système d'équations sur des expressions rationnelles dans un ordre différent, ou par des méthodes differentes, on peut obtenir un résultat différent. De tels résultats différents seront, bien sûr, équivalents, mais il est difficile de le voir immédiatement car l'algèbre des expressions rationnelles est fort compliquée (nous en avons vu un seul exemple – les règles (4a,b)). Une des meilleures façons de voir que deux expressions rationnelles sont équivalentes est de construire à partir d'elles deux automates asynchrones, les déterminiser et minimiser et arriver au même automate minimal.

C. Méthode d'élimination d'états

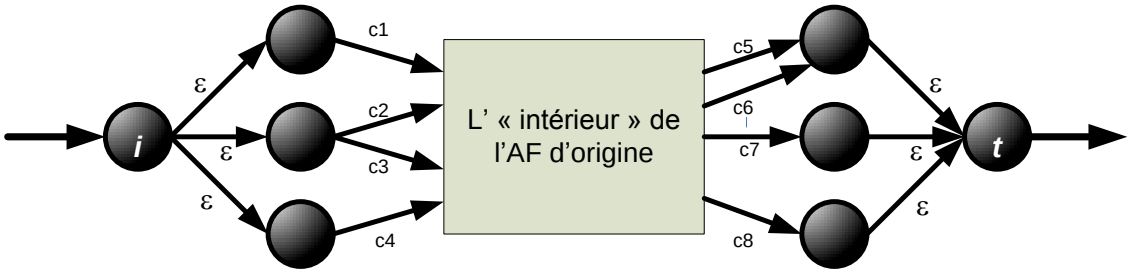
On définit d'abord un **Automate Fini Généralisé** : la seule différence avec un AF « normal » consiste en ce que les transitions peuvent être marquées par n'importe quelle expression rationnelle.

Voici la méthode :

1) On prend l'AF dont le langage on veut obtenir en tant qu'expression rationnelle. On y ajoute un état d'entrée ***i*** et un état de sortie ***t***, on crée des ε-transitions partant de l'état ***i*** vers toutes les entrées de l'automate d'origine, et arrivant dans l'état ***t*** de toutes les sorties de l'automate d'origine :

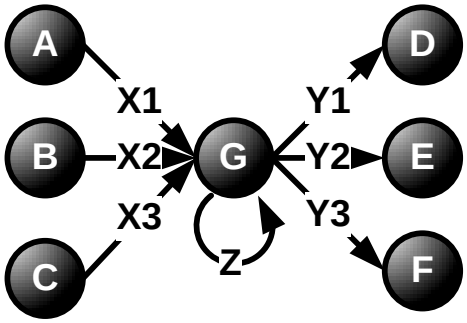


devient



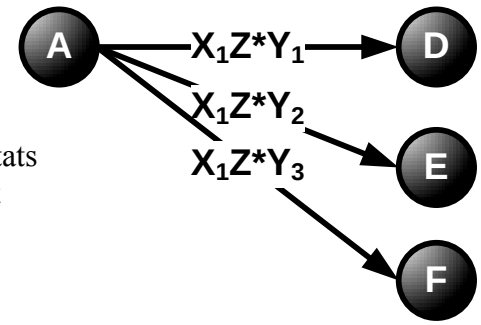
2) On « élimine » tous les états sauf ***i*** et ***t*** l'un après l'autre, dans un ordre quelconque, de la façon suivante : soit l'état G qu'on veut éliminer, avec des transitions entrantes, transitions sortantes et éventuellement avec une boucle. Dans le cadre d'un AF généralisé, je marque toutes ces transitions par des expressions rationnelles X_i, Y_i, Z :

(Il faut remarquer que parmi les états D,E, F certains peuvent coïncider avec certains des états A, B, C)



On liste tous les chemins passant par G : AGD, AGE, AGF, BGD, BGE, BGF, CGD, CGE, CGF. Chaque chemin donne lieu à une ER du type $X_iZ^*Y_j$. On obtient des transitions généralisées $A.X_1Z^*Y_1.D$, $A.X_1Z^*Y_2.E$, $A.X_1Z^*Y_3.F$, $B.X_2Z^*Y_1.D$ etc. :

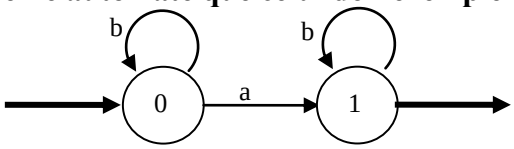
(je n'ai pas dessiné les états B et C et leurs transitions généralisées vers D, E et F pour ne pas encombrer le dessin).



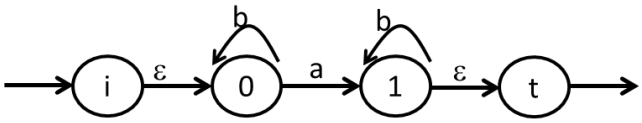
On procède ainsi jusqu'à l'élimination de tous les états sauf ***i*** et ***t*** . L'ER marquant la transition généralisée reliant ***i*** et ***t*** est le langage reconnu par l'automate.

Prenons les mêmes exemples qu'auparavant.

Exemple C1 : on prend le même automate que celui de l'exemple A1 :

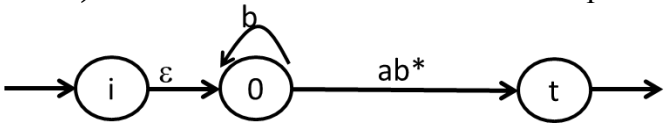


Etape 1 : l'ajout des nouveaux états initial et terminal

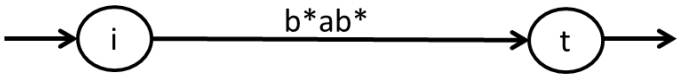


Etape 2, élimination itérative d'états :

a) Eliminons l'état 1 : le seul chemin passant par 1 est 01t, ce qui donne l'ER ab*ε=ab* :

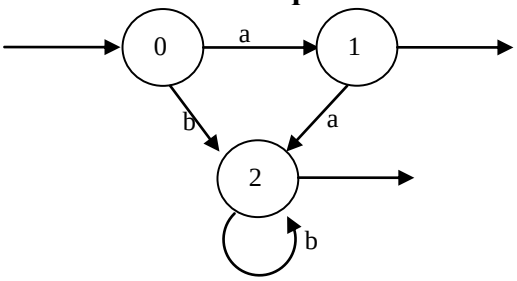


b) Eliminons l'état 0 : le seul chemin passant par 1 est i0t, ce qui donne l'ER εb*ab*=b*ab* :

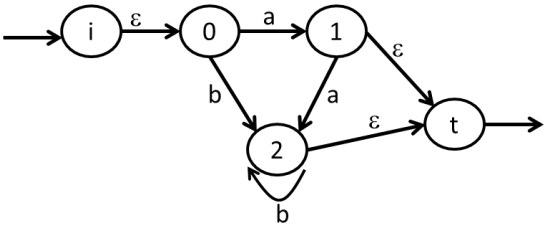


L = b*ab*.

Exemple C2 : on prend le même automate que celui de l'exemple A2 :

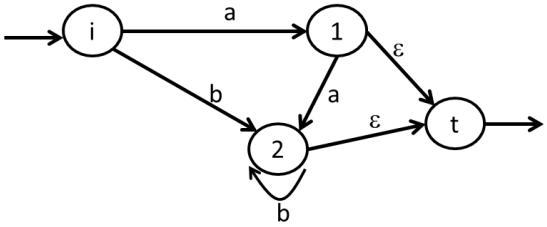


Etape 1 :

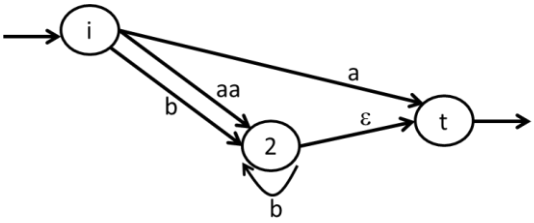


Etape 2 :

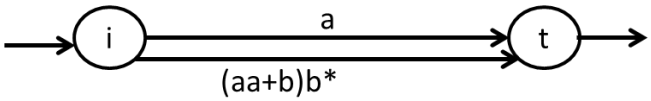
Eliminons 0 : il y a 2 chemins passant par 0, ce qui donne iεa1=ia1, iεb2=ib2



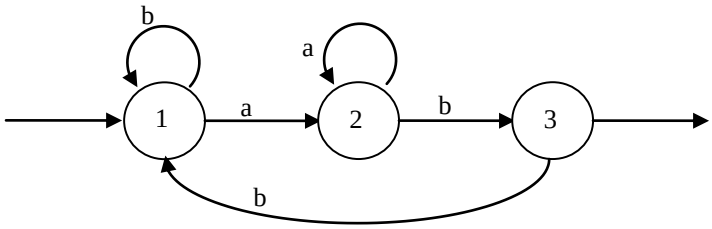
Eliminons 1: il y a 2 chemins passant par 1, ce qui donne iaεt=iat, iaa2



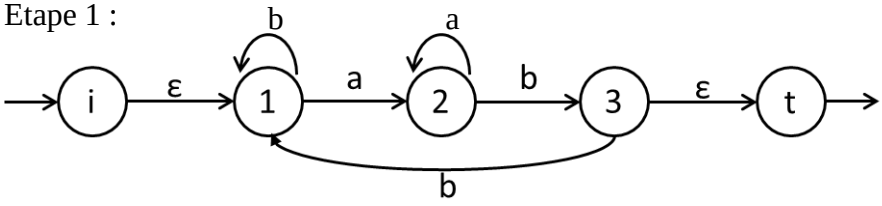
Eliminons 2: il y a un chemin passant par 2, ce qui donne i(aa+b)b*εt=i(aa+b)b*t. Avec la transition déjà existante iat cela donne
i(a+(aa+b)b*)t et donc L = a+(aa+b)b* = a + aab* + bb*.



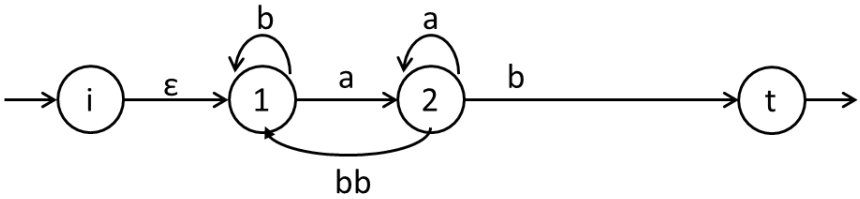
Exemple C3 : on prend le même automate que celui de l'exemple A3 :



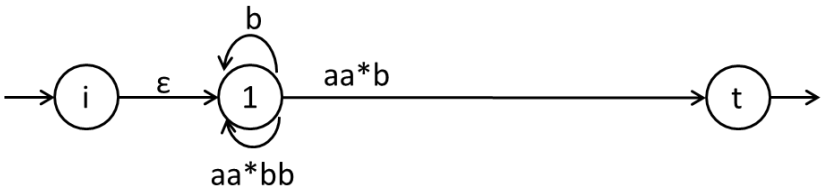
Etape 1 :



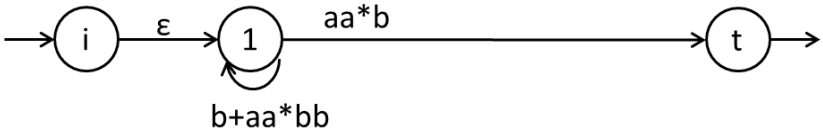
Etape 2 :
Eliminons 3 : il y a 2 chemins passant par 3, ce qui donne 2bεt=2bt ; 2bb1.



Eliminons 2 : il y a 2 chemins passant par 2, ce qui donne 1aa*bt, 1aa*bb1.



ce qu'on peut dessiner avec une seule boucle sur 1, marquée par la somme des deux expressions :



Eliminons 1 : il y a 1 chemin passant par 1, ce qui donne iε(b+aa*bb)*aa*bt = i(b+aa*bb)*aa*bt , donc L = (b+aa*bb)*aa*b.