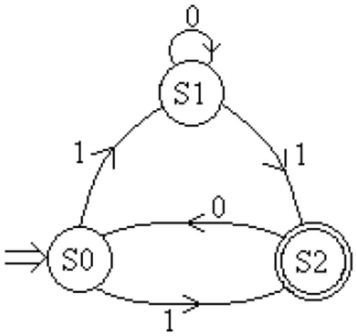
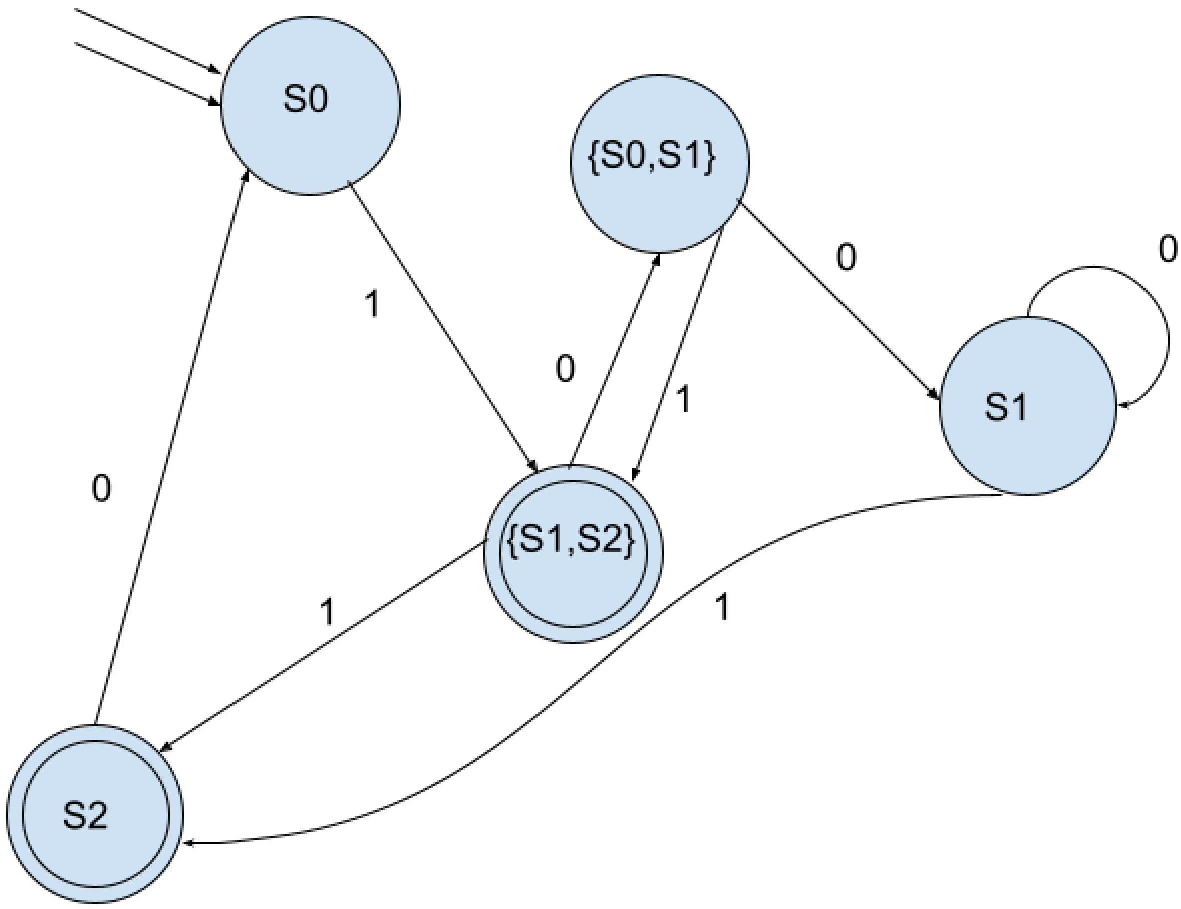
Série 3: Automates d'états finis et expressions régulières

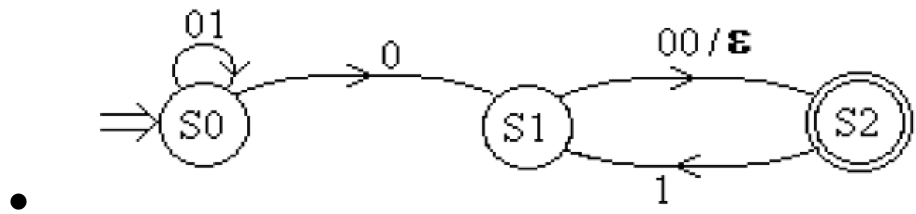
3.1. Trouver l'automate simple déterministe B tel que L(B) = L(A). A étant l'automate suivant:



etats	0	1
S0	/	{S1,S2}
{S1,S2}	{S0,S1}	S2
{S0,S1}	S1	{S1,S2}
S2	S0	/
S1	S1	S2



3.2

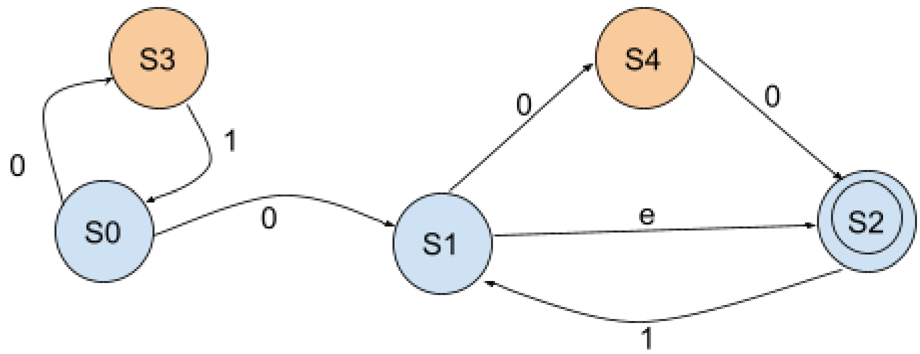


1/ Trouver l'automate d'états finis simple déterministe B tel que L(L'automate A est généralisé. Pour trouver l'automate B équivalent, il faut suivre les étapes suivantes:
1/ S'assurer que l'automate est réduit sinon, le réduire (ici, A est réduit)
2/ Rendre A partiellement généralisé
3/ Rendre A simple
4/ Rendre A déterministe

2.1/ L'automate A ne comporte aucun état non accessible ou non co-accessible, il est donc réduit.

2.2/ Rendre l'automate partiellement généralisé

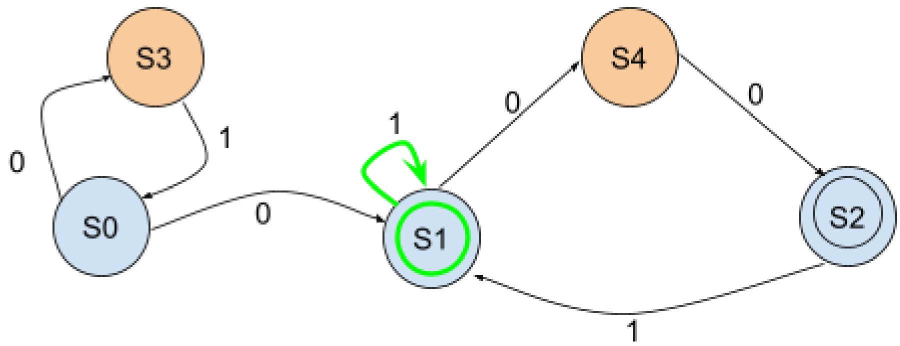
On voit que l'automate A est généralisé car il comporte une transition au niveau de S0 par le mot '01' , une deuxième entre S1 et S2 par le mot '00' et une autre par le mot vide. Pour le rendre partiellement généralisé il faut supprimer les transitions qui se font par des mots différents du mot vide en créant des états intermédiaires.



1.3/ Rendre l'AEF A simple

Pour cela il faut éliminer la transition spontanée entre S1 et S2. Pour cela il suffit que les successeurs de S2 deviennent les successeurs de S1 (avec les mêmes étiquettes) et si S2 est final, S1 devient final.

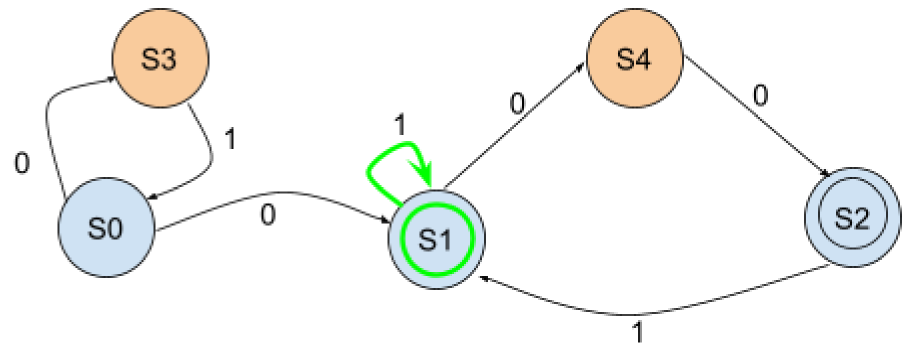
Reste à faire la représentation graphique de lAEF déterministe obtenu



2/ Trouver l'automate Complet C tel que L(C) = L(B)

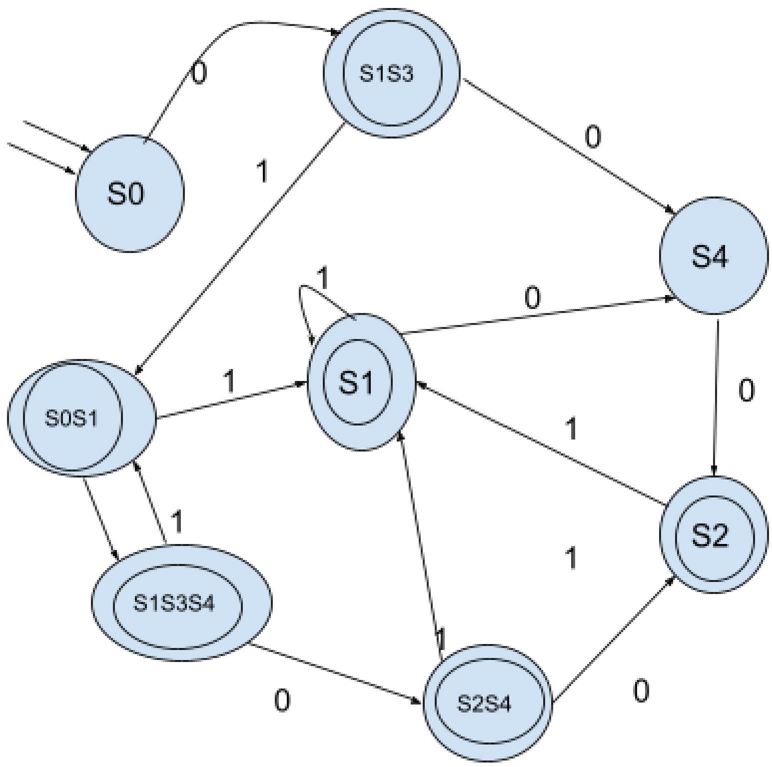
Etat	0	1

S0	S1, S3	-
S1	S4	S1
S2	-	S1
S3	-	S0
S4	S2	-
S0	S1, S3	-
S1,S3	S4	S0,S1
S4	S2	-
S0,S1	S1;S3,S4	S1
S2	-	S1
S1,S3,S4	S2,S4	S0,S1
S1	S4	S1
S2,S4	S2	S1



etats	0	1
S0	S3.S1	/
{S3.S1}	S4	{S0,S1}
{S0,S1}	{S3, S1 ,S4 }	S1
S4	S2	/
S1	S4	S1

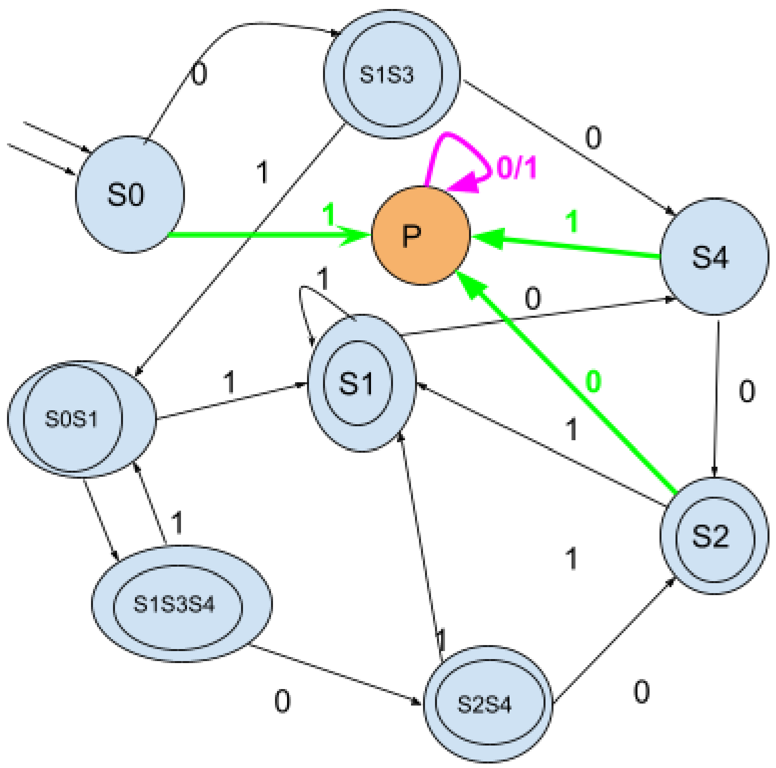
{S3, S1 ,S4 }	{S4, S2}	{S0, S1}
S2	-	S1
{S4, S2}	{S2 ,S4}	S1



2/ Trouver l'automate Complet C tel que L(C) = L(B)
Pour rendre l'AEF B complet, il suffit d'ajouter un état puits P

Etat	0	1
S0	{S1,S3}	**P**
{S1,S3}	S4	{S0,S1}
S4	S2	**P**
{S0,S1}	{S1,S3,S4}	S1

S2	**P**	S1
{S1,S3,S4}	{S2,S4}	{S0,S1}
S1	S4	S1
{S2,S4}	S2	S1
P	**P**	**P**



3/ Trouver l'AEF D tel que L(D) = complément de L(C)
Pour obtenir l'AEF reconnaissant le langage complément, il suffit d'inverser la nature de chaque état : un état final devient non final, et un état non final devient final.

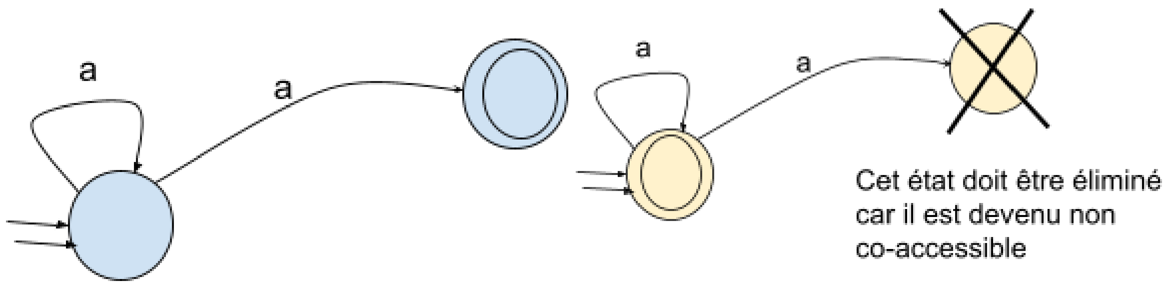
Attention: il faut d'abord que l'automate soit simple, déterministe et complet.
Explication:

L(D) et L(C) sont complémentaires ssi

- L(D) ∩ L(C) = l'ensemble vide (ceci est garanti par le fait que l'AEF C est déterministe)

- L(D) ∪ L(C)) = X* (ceci est garanti par le fait que l'AEF C est complet, ce qui permettra dans l'AEF D d'accepter les mots dont la lecture se terminait à l'état puits.)

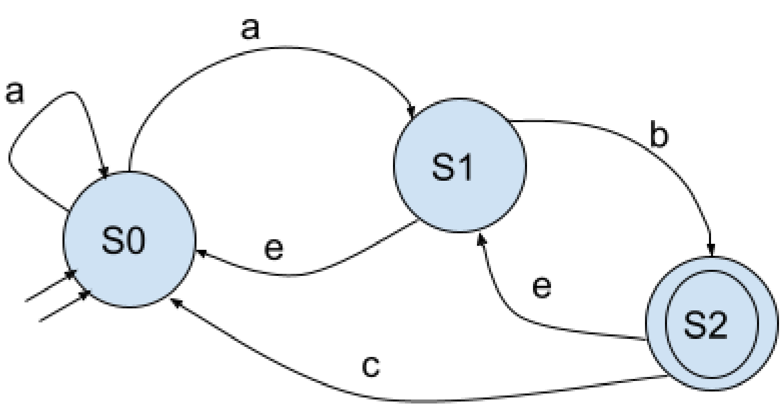
Voici un exemple qui explique l'importance du déterminisme. Si on considère l'alphabet W= {a}, on voit que dans l'automate à gauche qui reconnaît le langage L = a*a, le mot "a" est accepté, et le fait d'inverser la nature des états ne suffit pas pour obtenir l'AEF du complément car le mot "a" sera aussi accepté dans l'automate à droite.

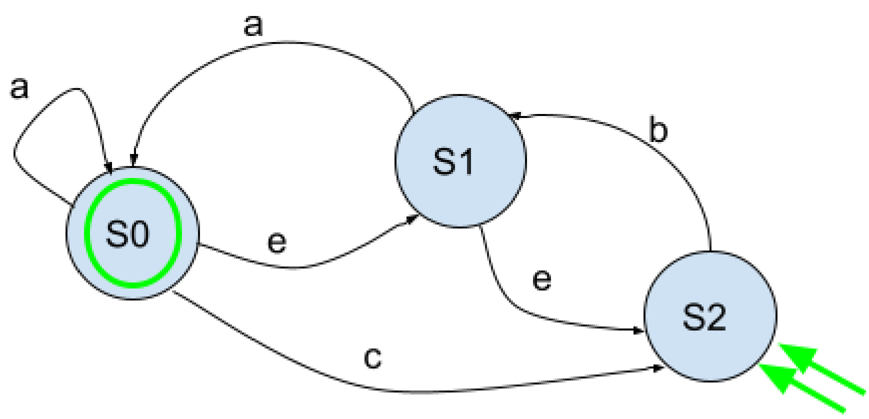


3.3. Soit A<X, (S0,S1,S2}, S0, (S2}, II > l'automate d'états finis dont la table de transition est donnée ci-dessous:

	ε	a	b	c
S0	-	S0,S1	-	-
S1	S0	-	S2	-
S2	S1	-	-	S0

Construire l'automate *B* tel que L(*B*) = $L^R(A)$





D

L'algorithme de construction de Ar :

Début

S'= {S} ∪ {S0'}

F' = {S0}

II'={(S0',ε,Si) pour tout Si ∈ F}

Pour chaque transition (Si ,w ,Sj) dans II

Faire II'=II' ∪ {(Sj, wr,Si)}

Fin Pour

Fin

Pour prouver que L(Ar)=(L(A))r

1/ **L(Ar)⊆(L(A))r**

∀W ∈ L(Ar) W =W1 . . . Wn ∈ L(Ar) donc il existe un chemin dans l'automate Ar de l'état initial vers l'état final
C à dire, S'0├ eS1' ├W1S2'......├ Sf' or Sf' = S0 (l'état initial dans l'automate initial A)
Donc S0├ WnSn' ├.WnWn-1Sn'......├ S1' ∈ F (L'état initial dans Ar est un état final dans A) On en déduit que le mot W appartient à L(A)

La démonstration du sens inverse est similaire.

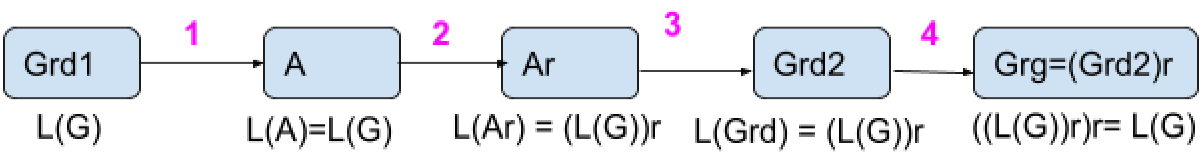
Passage d’une grammaire régulière droite à une grammaire régulière gauche équivalente (engendrant le même langage)

Soit la grammaire G suivante

S -> aA / bS
A -> aA / bB
B -> aB / bB / e
Quelle est la nature de cette grammaire ? C’est une grammaire régulière droite
Quel est le langage engendré par cette grammaire ? $b^*aa^*b(a \cup b)^*$

Soit la grammaire G’ obtenue en appliquant l’opérateur miroir sur les règles de G
S -> Aa / Sb
A -> Aa / Bb
B -> Ba / Bb / e
Le langage engendré par cette grammaire est $(a \cup b)^*ba^*ab^{**}$

Conclusion: En appliquant le miroir aux productions d’une grammaire régulière droite, on obtient une grammaire régulière gauche engendrant le langage miroir du langage L(G)
Problème : Trouver la grammaire régulière gauche équivalente à une grammaire régulière droite donnée.



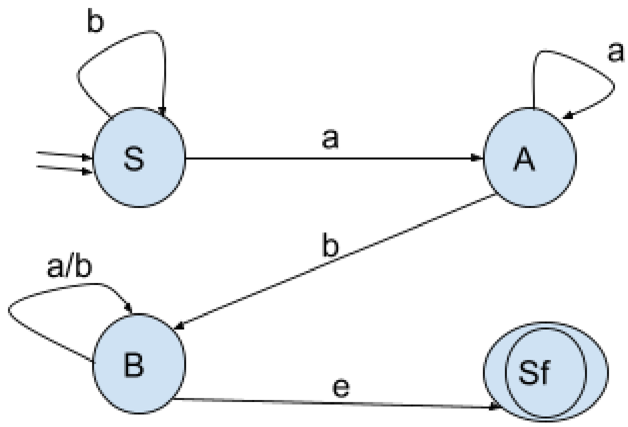
Remarque Lr = miroir de L

1/ Construction de A tel que L(A) = L(G)

Algo 1: Algorithme de passage d’une Grd à un AEF
Soit G<X, V, S, P> Soit A<X’, S’, S0’, F’, I’> tq X’ = X, S’ = V ∪ {Sf}, S0’ = {S}, F’ = {Sf} et pour toute production de P **Si** A -> wB tq A et B ∈ V, et w ∈ X* **Alors** I’ <- I’ ∪ {(A, w, B)} **Si** A -> w tq A ∈ V et w ∈ X* **Alors** I’ <- I’ ∪ {(A, w, Sf)}

Application à l'exemple

S -> aA / bS
A -> aA / bB
B -> aB / bB / e



Démonstration:

Montrons que L(G) = L(A)
C'est à dire, quelque soit w de X*, w ∈ L(G) ⇒ w ∈ L(A)
Pour prouver que L(G) = L(A) on doit prouver la double inclusion

w ∈ L(G) S |- w ⇒ S |-w1 V1 |- w1w2V2 |- w1w2….wn-1Vn-1|- w1w2….wn , wi∈X*
Or, selon l'algorithme de construction de l'AEF A, pour toute production dans la grammaire G il existe une transition équivalente dans l'automate, donc on peut faire exactement le même chemin dans l'AEF

S0 |- w ⇒ S0 |-w1 S1 |- w1w2S2 |- w1w2….wn-1Sn-1|- w1w2….wn Sf , wi∈X*

On en déduit que w ∈ L(A)

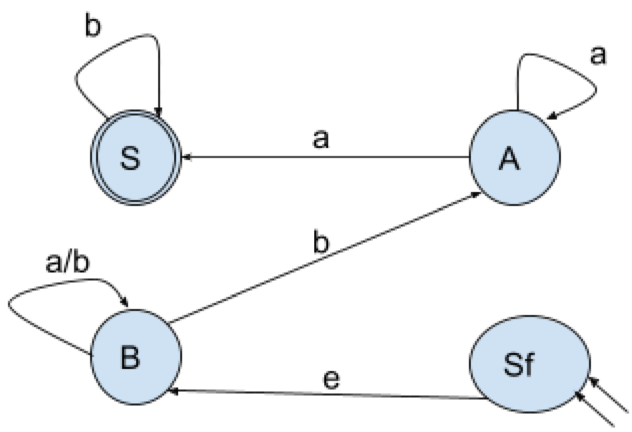
La démonstration de l'autre inclusion est similaire

$S \vdash_G^{*} w$

2/ Construction de Ar tel que L(Ar) = L(A)r

Algo 2: Algorithme de construction d'un AEF reconnaissant le langage miroir
Soit A<X', S', S0', F', I'> Soit Ar<X', S', S0', F', I'> tq X' = X, S' = S, S0' = {S}, F' = S0 et pour toute production de P Pour toute instruction i (si, w , sj)∈ I' <- I' ∪ {(sj, wr,si)} (NB:wr est le miroir du mot w)

Application à l'exemple



3/ Construction de Grd2 tel que L(GRD) = L(Ar)

Algo 3: Algorithme de passage d'un AEF à une grammaire régulière droite
Soit A<X, S, S0, F, I > S0 : une seul état initial Soit G<X', V', S', P'> tq X' = X, V'= S, S' = S0, et **<u>Pour</u>** toute instruction (si, w , sj) ∈ I **<u>faire</u>** P <- P ∪ {(si -> w Sj)} **<u>Pour</u>** tout état final si dans ∈ F **<u>faire</u>** P <- P ∪ {(si -> e)}

Application à l'exemple

Sf -> B

B -> aB / bB / bA

A -> aA / aS

S -> bS/ e

4/ Construction de (Grd2)r

Algo 4: Algorithme de passage d’une Grd à une Grg telle que L(Grd) = (L(Grg))r
Soit Grd<X, V, S, P> régulière droite Soit Grg<X’, V’, S’, P’> tq X’ = X, V’ = V, S’ = S et **pour** toute production A -> wB de P (A, B ∈ V et w ∈ X*) **faire** P’ <- P’ ∪ {(A-> B wr)} (NB: Wr est le miroir de w) **pour** toute production A -> w de P (A ∈ V, et w ∈ X*) **faire** P’ <- P’ ∪ {(A-> wr)}

Application à l’exemple

Sf -> B
B -> Ba / Bb / Ab
A -> Aa / Sa
S -> Sb/ e

Construction d'un automate à partir d'une expression régulière

Définition : On définit sur les expressions régulières une opération appelée **dérivée** comme suit :
Soit E une expression régulière on a $E//u = \{w \in X^* \,/\, uw \in E\}$, u est un mot de X^*.

3.2 Exemple : $E = ab^*(a \cup b)^*$,
$E//a = b^*(a \cup b)^* = E1$
$E//b = \varnothing = E2$, aucun mot de L(E) ne commence par un b.

3.3 Proposition : Soient E_1 et E_2, deux expressions régulières on a :

1. $(E_1 \cup E_2)//u = (E_1//u) \cup (E_2//u)$, $u \in X^*$
2. $(E_1 . E_2)//u_i = (E_1//u_i) . E_2$, $u_i \in X$, si $e \notin L(E_1)$
   $= (E_1//u_i) . E_2 \cup (E_2//u_i)$, $u_i \in X$, si $e \in L(E_1)$
3. $E_1^*// u_i = (E_1// u_i) . E_1^*$
4. $E_1//(u.v)= (E_1//u)//v$.

Propriétés : Soient E_1, E_2 et E_3, trois expressions régulières, on a :

1. $E_1 \cup E_2 = E_2 \cup E_1$
2. $E_1 \cup (E_2 \cup E_3) = (E_1 \cup E_2) \cup E_3$
3. $E_1 . (E_2 . E_3) = (E_1 . E_2) . E_3$
4. $E_1 . (E_2 \cup E_3) = (E_1 . E_2) \cup (E_1 . E_3)$
5. $(E_1 \cup E_2) . E_3 = (E_1 . E_3) \cup (E_2 . E_3)$
6. $E_1 \cup \varnothing = \varnothing \cup E_1 = E_1$
7. $E_1 . e = e . E_1 = E_1$
8. $E_1 . \varnothing = \varnothing . E_1 = \varnothing$
9. $E_1 \cup E_1 = E_1$
10. $E_1^* . E_1^* = E_1^*$
11. $E_1 . E_1^* = E_1^* . E_1 = E_1^+$
12. $(E_1^* \cup E_2^*)^* = (E_1 \cup E_2)^* = (E_1^* . E_2^*)^*$
13. $(E_1 \cup E_2)^* = E_1^* . (E_2 . E_1^*)^*$

Proposition :

A toute expression régulière E, il existe un automate d'états finis A<X,S,S_0,F,II> simple déterministe complet tq L(E)= L(A)

Nous allons montrer les étapes de construction de l'automate d'états finis avec la méthode des dérivées à travers un exemple, puis nous démontrerons la proposition:

Soit **E0 = a* b***

1. On associe à l'expression régulière initiale E0, l'état initial S0

2. On dérive E0 par rapport à toutes les lettres de X. A chaque nouvelle expression régulière obtenue est associé un état dans l'AEF. On s'arrête lorsqu'il n'y a plus de nouvelles expressions régulières.

3. Tout état associé à une expression régulière contenant epsilon (e) est un état final.

<u>Rappel des propositions:</u>

1. $(E_1 \cup E_2)//u = (E_1//u) \cup (E_2//u)$, $u \in X^*$
2. $(E_1 . E_2)//u_i = (E_1//u_i) . E_2$, $u_i \in X$, si $e \notin L(E_1)$
   $= (E_1//u_i) . E_2 \cup (E_2//u_i)$, $u_i \in X$, si $e \in L(E_1)$
3. $E_1^*// u_i = (E_1// u_i) . E_1^*$

E0 = a* b*

E0//a = (a* b*) //a = (a* // a).b* ∪ b*//a (proposition 2 avec e ∈ L(a*))

= (a//a).a* b* ∪ ∅ = e.a* b* = a*b* = E0 (proposition 3)

E0//b = (a* b*) //b = (a* // b).b* ∪ b*//b (proposition 2)

= ((a//b) . a*) b* ∪ b*//b =

= ∅.b* ∪ (b//b). b* = ∅ ∪ e.b* = b* = E1 (proposition 3)

E1//a = b*//a = ∅ = E2

E1//b = b*//b = b* = E1 (proposition 3)

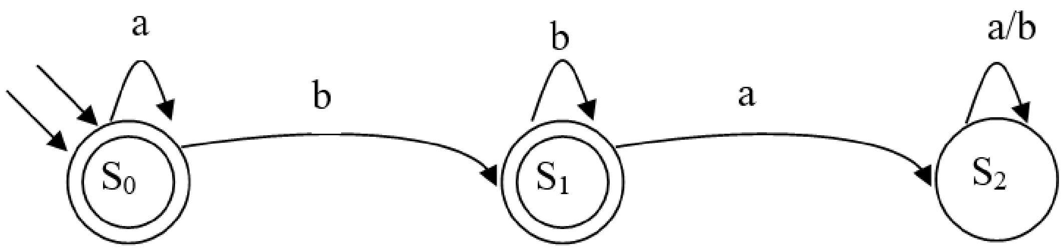
E2//a = ∅ // a = ∅ = E2 (état puits)

E2//b = ∅ // b = ∅ = E2

Algorithme sarrête à ce niveau car Il n'y a plus de nouvelles expressions régulières à traiter. Nous allons maintenant tracer l'AEF en associant chaque expression régulière Ei à un état Si de l'automate.

Epsilon appartient à L(E0) et à L(E1), les états S0 et S1 sontt donc des états finaux.

L'automate qui reconnaît les mots dénotés par E0 est le suivant :



<u>**Exercice :**</u> Trouver l'AEF reconnaissant le langage dénoté par l'expression régulière suivante **X*abX***

E0 = X*abX*

E0// a = (X*abX*) // a = (X*//a) (abX*) ∪ (abX*)//a (proposition 2)

= X*abX* ∪ bX*= E0 ∪ bX* = E1

> **A retenir:**
> X* // a = (a ∪b)* // a = (a ∪ b) // a . (a ∪b)* = (a//a ∪ b// a) . (a ∪b)* = (e ∪ ∅).(a ∪b)* = (a ∪b)* = X*
> Plus généralement, quelque soit l'alphapet X et quelque soit la lettre x de l'alphabet X, X*//x = X*

E0 // b = (X*abX*) // b = (X*//b) (abX*) ∪ (abX*)//b (proposition 2)

= X*abX* ∪ ∅ = X*abX* = E0

Remarque : on peut exploiter les calculs effectués précédemment car E1 = E0 ∪ bX*

E1 // a = E0// a ∪ bX* // a = E1 ∪ ∅ = E1

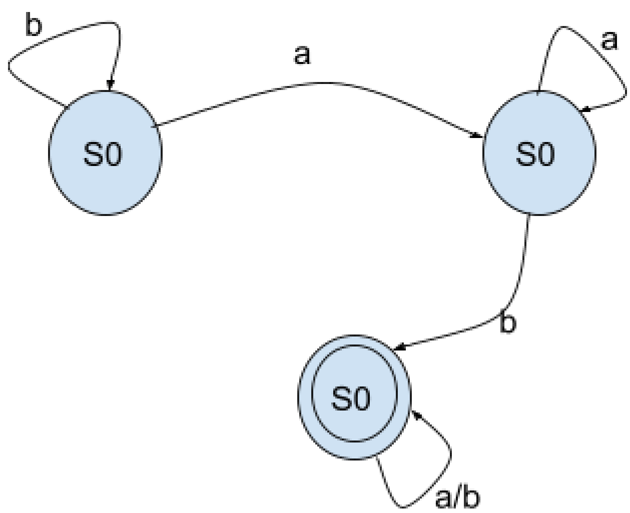
E1 // b = E0// b ∪ bX* // b = E0 ∪ X* = X* = E2

A retenir:
Pendant le calcul des dérivées, il est utile de remarquer qu'une expression régulière en contient une autre afin de "compacter" leur union et accélérer les calculs.
Par exemple: quelque soit Ei, $X^* \cup Ei = X^*$

E2 = X*

E2 // a = E2 // b = E2

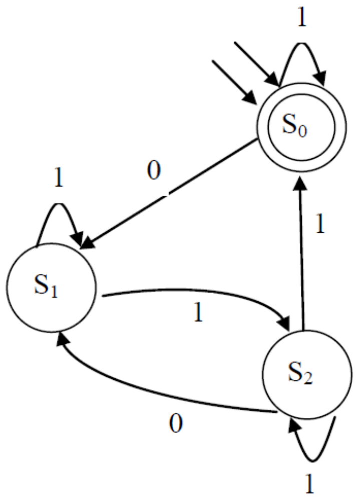
Il n'y a plus aucune nouvelle expression régulière à traiter, l'AEF va donc contenir trois états et le seul état final est S2.



Trouver l'expression régulière à partir d'un AEF

Proposition: A tout automate d'état finis A<X, S, S0, F, II>, il existe une expression régulière E tel que L(A) = L(E).

Démonstration :
Nous allons présenter les étapes de construction d'une expression régulière à partir d'un automate simple à travers un exemple, puis nous généraliserons les étapes.
Soit l'exemple ci-dessous:



1. Nous allons associer à chaque état de l'automate, une expression régulière qui permet de savoir comment s'écrivent les mots qui font passer l'automate de cet état aux états finaux. Nous nous intéresserons aux mots qui font passer l'automate de l'état initial à un des états finaux.

Par exemple, à l'état S1, nous allons associer l'expression régulière E1. A partir de S1, et à la lecture d'un 1, l'automate peut passer à l'état S2 ou revenir à l'état S1. Les mots que l'automate lit à partir de S1 commencent forcément par un 1. L'expression régulière associé à S1 est la suivante E1 = 1 E1 ∪ 1 E2.

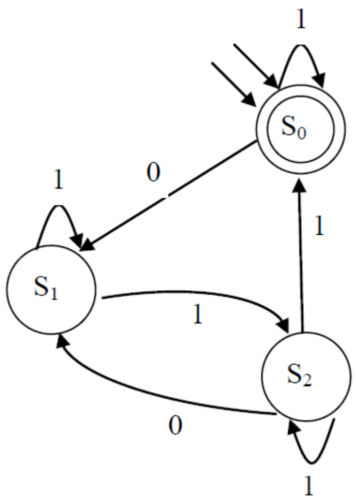
Les expressions régulières E2 et E0, associées respectivement à S2 et S0 sont définies comme suit :
E2 = 1 E0 ∪ 0E1 ∪ 1E2
E0 = 1 E0 ∪ 0E1 ∪ e , le mot vide est ajouté car S0 est un état final.

Nous avons établi trois équations à trois inconnues. Maintenant, nous allons résoudre ces équations en éliminant étape par étape les inconnues par une série de remplacements afin de trouver l'expression régulière de E0 qui dénote le langage reconnu par l'AEF. Pour cela nous utiliserons le théorème d'Arden.

Théorème d'Arden: Soient A et B deux langages, A*B est une solution de l'équation L = AL ∪ B

Dans cet exemple, l'état initial est aussi l'état final.

$$\begin{cases} E_0 = 1\ E_0 \cup 0E_1 \cup \varepsilon \\ E_1 = 1\ E_1 \cup 1\ E_2. \\ E_2 = 1\ E_0 \cup 0E_1 \cup 1E_2 \end{cases}$$



2. Elimination de E2:

$$\begin{cases} E_0 = 1\ E_0 \cup 0E_1 \cup \varepsilon \\ E_1 = 1\ E_1 \cup 1\ E_2. \\ \underset{L}{E_2} = \underbrace{1\ E_0 \cup 0E_1}_{B} \cup \overset{A}{1}\underset{L}{E_2} \end{cases}$$

$E_2 = 1^*\ (1E_0 \cup 0E_1)$, (Application de la règle d'Arden)

3. Remplacement de E2 dans E1 et élimination de E1:

$$\overset{L}{E_1} = 1E_1 \cup 1[1^*\ (1E_0 \cup 0E_1)] = \overset{B}{11^*1E_0} \cup \overset{A}{(1 \cup 11^*0)}\ \overset{L}{E_1}$$

$$E_1 = (1 \cup 11^*0)^*\ 11^*1E_0$$

Remplacement de E1 dans E0 et élimination de l'inconnue E0

$$E_0 = 1\,E_0 \cup 0E_1 \cup \varepsilon = 1E_0 \cup 0(1 \cup 11^*0)^*\ 11^*1E_0 \cup \varepsilon$$
$$E_0 = \overbrace{\boxed{(1 \cup 0(1 \cup 11^*0)^*\ 11^*1)}}^{A}E_0 \cup \varepsilon \quad B$$
$$E_0 = (1 \cup 0(1 \cup 11^*0)^*\ 11^*1)^*.\ \varepsilon = (1 \cup 0(1 \cup 11^*0)^*\ 11^*1)^*$$

Généralisation :

- On associe à chaque état, une expression régulière. Il y a autant d'expressions régulières qu'il y a d'états, soit n ce nombre.

$$E_k = \bigcup_{j=1}^{n} V_{kj}E_J \cup w_k \text{ avec } w_k = \begin{cases} \varepsilon \text{ si } S_k \in F \\ \varnothing \text{ Sinon} \end{cases} \qquad 1 \leq k \leq n$$

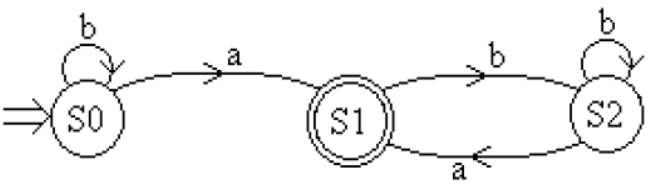
Avec $V_{kj} = \{x_i \in X \text{ tq } (S_k, x_i, S_J) \in II\}$

- Elimination des inconnues, étape par étape (application de la règle d'Arden):

$$E_1 = \bigcup_{j=1}^{n} V_{1J}E_J \cup w_1 = V_{11}\ E_1 \bigcup_{j=2}^{n} V_{1J}E_J \cup w_1$$

$$E_1 = V_{11}^{\ *}\ (\bigcup_{j=2}^{n} V_{1J}E_J \cup w_1)$$

Exercice Donner l'expression régulière dénotant le langage reconnu par l'AEF suivant



E0 = bE0 ∪ aE1
E1 = bE2 ∪ e
E2 = bE2 ∪ aE1

E2=b*aE1
E1=b(b*aE1) u e
E1=(bb*a)*
E0 = bE0 ∪ a(bb*a)*
E0 = b*a (bb*a)*

Autre solution

1/ Ecriture des équations
E0 = bE0 ∪ aE1
E1 = b E2 ∪ e
E2 = a E1 ∪ bE2

2/ résolution des équations

E2 = a E1 ∪ bE2 = a(bE2 ∪ e) ∪ bE2 = abE2 ∪ a ∪ bE2 = (ab ∪b)E2 ∪ a = (ab ∪ b)*.a
E1 = bE2 ∪ e = b(ab∪b)*.a ∪e
E0 = bE0 ∪ a E1 = b*aE1 = b*a(b(ab∪b)*.a ∪e) = b*ab(ab∪b)*a ∪b*a

E0 = b*ab(ab∪b)*a ∪b*a

Remarque: les expressions régulières obtenues dans les deux solutions ne sont pas identiques mais elles sont équivalentes.

Exercices

1/ donner l'AEF reconnaissant le langage dénoté par cette expression régulière

(ab*a(a∪b)*∪aa(a∪b)*)

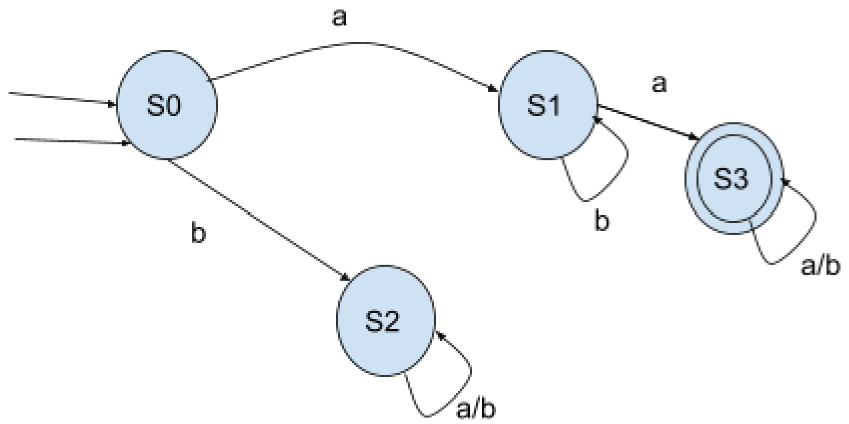
1. $(E_1 \cup E_2)//u = (E_1//u) \cup (E_2//u),\ u \in X^*$

E0//a = (ab*a(aUb)* Uaa (aUb)*)//a
=(ab*a(aUb)*// a) U(aa (aUb)* //a)
= b*a(aUb)* U a (aUb)* = b*a(aUb)* = E1

E0//b = ∅ = E2

E1 = b*a(aUb)*
E1//a = (b*a(aUb)*)//a = (b*//a)a(aUb)*U (a(aUb)*)//a = ∅ U (aUb)* = (aUb)* = E3
E1//b = (b*//b).(a(aUb)*)U (a(aUb)*)//b) = b*a(aUb)* U ∅ = b*a(aUb)* = E1

E2//a = E2//b = E2
E3//a = E3//b = E3



Remarque: Nous pouvons aussi remarquer que l'expressions initiale (ab*a(aUb)* U aa(aUb)*) est l'union de deux expressions dont l'une est incluse dans l'autre. En effet, aa(aUb)* est incluse dans ab*a(aUb)*.

Donc (ab*a(aUb)* U aa(aUb)*) = ab*a(aUb)*

E0//a = ab*a(aUb)* // a = b*a(aUb) = E1

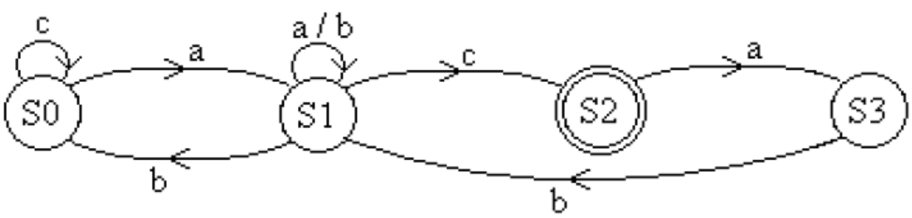
E0//b = ab*a(aUb)*//b = ∅ = E2

E1//a = b*a(aUb)//a = (aUb)* = E3

E1// b = b*a(aUb)//b = E1

E2// a = E2 // b = E2
E3// a = E3 // b = E3

2/ Donner l'expression régulière dénotant le langage reconnu par l'AEF ci-dessous



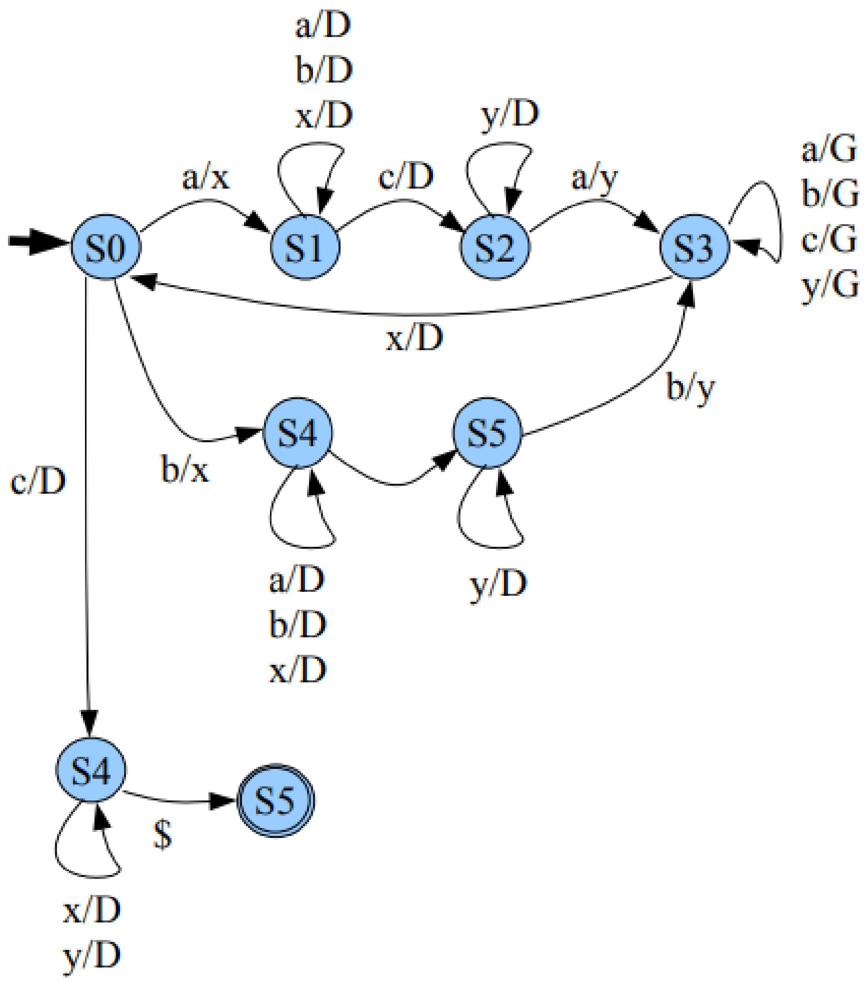
E0 = c.E0 U a.E1
E1= (a U b).E1 U b.E0 U c.E2
E2= a.E3 U e
E3 = b.E1

\> E3 = b.E1
\> E2 = a.b.E1 U e
\> E1 = (a U b).E1 U b.E0 U c.(a.b.E1 U e)
= (a U b U c.a.b).E1 U b.E0 U c
= (a U b U c.a.b)*.(b.E0 U c)
\> E0 = c.E0 U a.((a U b U c.a.b)*.(b.E0 U c))
= (c U a.(a U b U c.a.b)*. b).E0 U a.(a U b U c.a.b)*.c
= (c U a.(a U b U c.a.b)*. b)*.a.(a U b U c.a.b)*.c

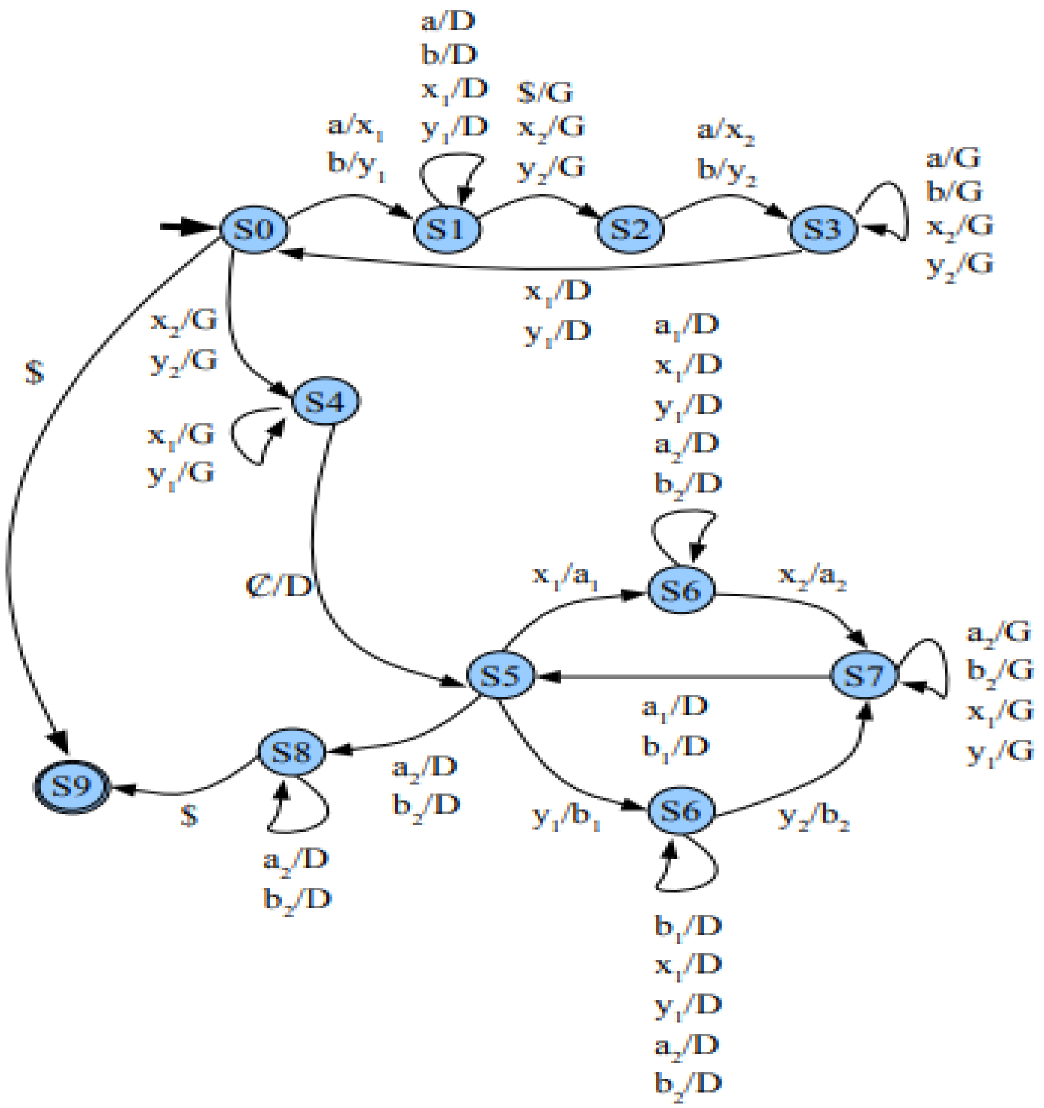
Automates à pile et Automates à Bornes Linéaires

Exo 13

L= { wcw / w e {a,b}}



L= { ww / w e {a,b}}


a/D
b/D
x_1/D
y_1/D
$/G
x_2/G
y_2/G
a/x_1
b/y_1
a/x_2
b/y_2
a/G
b/G
x_2/G
y_2/G
S0
S1
S2
S3
x_1/D
y_1/D
x_2/G
y_2/G
$
S4
x_1/G
y_1/G
a_1/D
x_1/D
y_1/D
a_2/D
b_2/D
Ȼ/D
x_1/a_1
S6
x_2/a_2
S5
S7
a_2/G
b_2/G
x_1/G
y_1/G
a_1/D
b_1/D
S8
S9
$
a_2/D
b_2/D
y_1/b_1
S6
y_2/b_2
a_2/D
b_2/D
b_1/D
x_1/D
y_1/D
a_2/D
b_2/D

Série 1: Calculabilité

Rappel:

1. **La calculabilité**
   Elle répond à la question: Quelles sont les fonctions calculables ? càd, celles pour lesquelles il existe un algorithme de calcul.
   1. Fonctions lambda calculables
   2. Fonctions récursives
   3. Fonctions Turing calculables

2. **Les fonctions primitives récursives**

Elles forment une sous-classe des fonctions récursives
Ce sont toutes celles dont nous avons besoin en pratique

Elles sont définies à partir de:

- Fonctions de base:
  - $Z(x) = 0 \quad \forall\ x \in N$
  - $S(x) = x+1 \quad \forall\ x \in N$
  - Projection $P_i^n(x_1, \ldots, x_n) = x_i$
- 2 opérations sur les fonctions de base
  - La récursion
  - La composition

2.1. La règle de récursion Étant données g une fonction à n arguments (n> 0), et h une fonction à n + 2 arguments, une fonction f à n + 1 arguments est obtenue par récursion à partir de g et h, si et seulement si, pour tout x1, x2, ...,xn, y appartenant à N, on a :

$$f(x_1, x_2, \ldots, x_n, 0) = g(x_1, x_2, \ldots, x_n)$$
$$f(x_1, x_2, \ldots, x_n, y+1) = h(x_1, x_2, \ldots, x_n, y, f(x_1, x_2, \ldots, x_n, y))$$

Cas particulier (n = 0)
f(0) = constante
f(y+1) = h (y, f(y)).

Cas particulier (n = 1)
f(x,0) = g(x)
f(x,y+1) = h (x,y, f(x,y)).

2. La composition

Etant données g1,...,gm, m fonctions à n arguments (n >0), et h une fonction à m arguments, une fonction f à n arguments est définie par composition à partir de g1,...,gm et de h, si et seulement si pour tout x1,.....,xn appartenant à N, on a :

$$f(x_1, x_2, \ldots, x_n) = h(g_1(x_1, x_2, \ldots, x_n), \ldots, g_m(x_1, x_2, \ldots, x_m))$$

Une fonction f est primitive récursive si et seulement si :
i. f est une fonction initiale ;
ii. f peut être obtenue à partir des fonctions initiales à l'aide des règles de composition et/ou de récursion.

Fonctions démontrées PR en cours:

$$Sg(x) = \begin{cases} 0 \text{ si } x = 0 \\ 1 \text{ sinon} \end{cases}$$

$$\overline{Sg}(x) = \begin{cases} 1 \text{ si } x = 0 \\ 0 \text{ sinon} \end{cases}$$

$$Pr(x) = \begin{cases} 0 \text{ si } x = 0 \\ x-1 \text{ sinon} \end{cases}$$

x+y, x∸y , abs(x,y)

Exercice 1 Montrer que les fonctions suivantes sont primitives récursives:

1/ f(x,y) = x*y

Nous devons trouver une fonction g à 1 argument et une fonction h à 3 arguments telles que :

f(x,0) = g(x)
f(x,y+1) = h (x,y, f(x,y)).

On a

f(x,0) = x * 0 = 0 = Z(x) *(la fonction g que nous cherchions est la fonction Zéro)*

f(x,y+1) = x * (y +1) = x*y + x = +(x, x*y) = $+ (P^3_1, P^3_3)$ (x,y,x*y)

(la fonction h que nous cherchions est la fonction $+ (P^3_1, P^3_3)$*)*

2/ f(x,y) = x^y

f(x, 0) = $x^0 = 1 = SoZ(x)$

f(x, y+1) = $x^{y+1} = x * x^y$

f(x, y+1) = * (P^3_1, P^3_3)(x,y, f(x,y))

3/ f(x) = fact(x)

f(0) = cte

f(x+1) = h (x, f(x)) .

f(0) = 1

f(x+1) = (x+1)*f(x) = *$(S\ o\ P^2_1, P^2_2)$ (x,f(x)) = *$(S\ o\ P^2_1, P^2_2)$ (x,fact(x))

4/ r(x, y) : reste de la division de y sur x

r(x,0) = 0 = Z(x)

r(x, y+1) = ?

Petit exemple numérique

r(3,6) = 0

r(3,7) = 1

r(3,8) = 2

r(3,9) = 0

On remarque que

$$r(x, y+1) = \begin{cases} r(x,y) + 1 & \text{si } r(x,y) + 1 < x \\ 0 & \text{si } r(x,y) + 1 = x \end{cases}$$

Maintenant, on doit chercher la fonction h telle que f(x, y+1) = h (x, f(x)) .

$$r(x, y+1) = \begin{cases} S(r(x,y)) & \text{si } S(r(x,y)) < x \\ 0 & \text{si } S(r(x,y)) = x \end{cases}$$

$$r(x, y+1) = \begin{cases} S(r(x,y)) & \text{si } x - S(r(x,y)) > 0 \\ 0 & \text{si } x - S(r(x,y)) = 0 \end{cases}$$

$$r\,(x, y+1) = S(r(x,y)) * \begin{cases} 1 & \text{si } x - S(r(x,y)) > 0 \\ 0 & \text{si } x - S(r(x,y)) = 0 \end{cases}$$

$$r\,(x, y+1) = S(r(x,y)) * \begin{cases} 1 & \text{si } x - S(r(x,y)) > 0 \\ 0 & \text{si } x - S(r(x,y)) = 0 \end{cases}$$

A ce niveau on peut remplacer le - par l'opérateur $\dot{-}$; ce qui nous donne

$$r\,(x, y+1) = S(r(x,y)) * \begin{cases} 1 & \text{si } x \dot{-} S(r(x,y)) > 0 \\ 0 & \text{si } x \dot{-} S(r(x,y)) = 0 \end{cases}$$

On remarque ici que l'on obtient la définition de la fonction Sg(S(r(x,y)))
Après remplacement on obtient

r(x, y+1) = S(r(x,y) * Sg (x $\dot{-}$ S(r(x,y))
r(x, y+1) = * (S(r(x,y)) , Sg (x $\dot{-}$ S(r(x,y)))
r(x, y+1) = * (S(r(x,y)) , Sg $\circ$ $\dot{-}$ (x,S(r(x,y)))
r(x, y+1) = * (S(r(x,y)) , Sg $\circ$ $\dot{-}$ (P^3_1, $S \, o \, P^3_3$))) (x, y , r(x,y))
r(x, y+1) = * (SoP^3_3 , Sg $\circ$ $\dot{-}$ (P^3_1, $S \, o \, P^3_3$))) (x, y , r(x,y))

5/ f(x,y) = q(x,y) : quotient de la division de y par x

q(x,0) = 0 = Z(x)
q(x, y+1) = ?
Petit exemple numérique
r(4,5) = 1
r(4,6) = 1
r(4,7) = 1
r(4,8) = 2

On remarque que

$$q\,(x, y+1) = q(x,y) + \begin{cases} 1 & \text{si } x*(q(x,y)+1) = y+1 \\ 0 & \text{si } x*(q(x,y)+1) = y+1 \end{cases}$$

Maintenant, on doit chercher la fonction h telle que f(x, y+1) = h (x, f(x)) .

$$q(x, y+1) = q(x,y) + \begin{cases} 1 & \text{si } x^* (q(x,y)+1) - (y+1) = 0 \\ 0 & \text{si } x^* (q(x,y)+1) - (y+1) > 0 \end{cases}$$

A ce niveau on peut remplacer le - par l'opérateur $\dot{-}$; ce qui nous donne

$$q(x, y+1) = q(x,y) + \begin{cases} 1 & \text{si } x^* (q(x,y)+1) \dot{-} (y+1) = 0 \\ 0 & \text{si } x^* (q(x,y)+1) \dot{-} (y+1) > 0 \end{cases}$$

On remarque ici que l'on a la définition de la fonction $\overline{Sg}$(x* (r(x,y) +1) $\dot{-}$ (y+1))

Après remplacement on obtient

$q(x, y+1)$= q(x,y) +$\overline{Sg}$(x* (q(x,y) +1) $\dot{-}$ (y+1))

= + $(P^3_3, \overline{Sg\ o} \dot{-} (* \ (S\ o\ P^3_3, SoP^3_2)))$ (x,y q(x,y))

5'/ quotient avec la fonction reste

On remarque que

$$r(x, y+1) = \begin{cases} r(x,y) + 1 & \text{si } r(x,y) + 1 < x \\ 0 & \text{si } r(x,y) + 1 = x \end{cases}$$

$$r(x, y+1) = \begin{cases} S(r(x,y)) & \text{si } S(r(x,y)) < x \\ 0 & \text{si } S(r(x,y)) = x \end{cases}$$

$$r(x, y+1) = \begin{cases} S(r(x,y)) & \text{si } x - S(r(x,y)) > 0 \\ 0 & \text{si } x - S(r(x,y)) = 0 \end{cases}$$

$$r(x, y+1) = S(r(x,y)) * \begin{cases} 1 & \text{si } x - S(r(x,y)) > 0 \\ 0 & \text{si } x - S(r(x,y)) = 0 \end{cases}$$

$r(x, y+1)$= S(r(x,y)) * Sg (x $\dot{-}$ S(r(x,y)))
$r(x, y+1)$= *(S o P_3^3, Sg o $\dot{-}$ (P_1^3, S o P_3^3)) (x, y, r(x,y))

6/ racc (x): racine carrée de x
racc(0) = 0 = Z(X)

Racc (x+1) = ?
Petit exemple numérique:
racc(2) = 1.41
racc(3) = 1.73
racc(4) = 2
racc(5) = 2.23
racc(6) =2.44
racc(7) =2.64
racc(8) =2.82
racc(9) =3

On en déduit que

	⎧	1 si $(racc(x) + 1)^2$= x+1
racc(x+1) = racc (x)+	⎨	
	⎩	0 si $(racc(x) + 1)^2$> x+1

Maintenant, on doit chercher la fonction h telle que
f(x, y+1) = h (x, f(x)) .

	⎧	1 si $(racc(x) + 1)^2$- x+1=0
racc(x+1) = q(x,y) +	⎨	
	⎩	0 si $(racc(x) + 1)^2$-(x+1) >0

	⎧	1 si $(racc(x) + 1)^2 \dot{-}$ (x+1)=0
racc(x+1) = q(x,y) +	⎨	

	⌊	0 si $(racc(x)\ +\ 1)^2 \dot{-} ($ x+1)>0

remarque que le deuxième terme représente la fonction $\overline{Sg}((racc(x)\ +\ 1)^2 \dot{-}$ (x+1))
Après remplacement, on a
racc(x+1) = racc(x)+ $\overline{Sg}((racc(x)\ +\ 1)^2 \dot{-}$ (x+1))
= racc(x) + $\overline{Sg}\ o \dot{-} ((racc(x)\ +\ 1)^2$,(x+1))
= racc(x) + $\overline{Sg}\ o \dot{-} ((S(racc(x)))^2$, S(x))
=+(P^2_1, $\overline{Sg}\ o \dot{-} ((SoP^2_2)^2$, So P^2_1)) (x, racc(x))

On peut utiliser le produit au lieu de la puissance

= + (racc(x) , $\overline{Sg}$((*(S(racc(x)),S(racc(x)) $\dot{-}$ S(x)))
= +(P^2_1, $\overline{Sg}$ o $\dot{-}$(*(S(P^2_2) , (S(P^2_2)),S(P^2_1))) (x,racc(x))

7/ f(x,y,z) = $(x^y)^z$

Méthode 1: composition

$(x^y)^z$ **= puiss(**$x^y,\ z) \ =\ puiss(puiss(x, y), z)\ =\ puiss(puiss(P^3_1, P^3_2)(x, y, z), P^3_3\ (x, y, z))$

Ou bien

$(x^y)^z$ **=** $x^{y*z} = \ puiss\ (x,\ \ *\ (y, z))\ =\ puiss\ (P^3_1(x,\ y,\ z),\ \ *\ (P^3_2, P^3_3)(x, y, z))$

Méthode 2: récursion

$(x^y)^0\ =\ 1\ =\ S\ o\ Z(x)$
$(x^y)^{z+1}$ **=** $(x^y)^z\ *\ \ x^y\ =\ \ *\ \ (P^4_4,\ puiss\ (P^4_1, P^4_2)\ (x,\ y,\ z,\ f(x, y, z))$

8/ f(x,y,z) = $x^{|y-z|}$

$x^{abs(y,z)}$= puiss(x, abs(y,z))
=puiss (P^3_1 (x, y, z), abs(P^3_1, P^3_2)(x,y,z))