AF 4

Langage

Def Facteur

Un mot $u \in A^*$ est un facteur d'un mot $w \in A^*$ si $\exists\, f, g \in A^*$ tq $w = fug$

Un facteur est dit propre s'il est différent du mot vide

" " " strict " " " " " entier

Def Sous mot

Un sous mot d'un mot $w \in A^*$ est un mot construit à partir de sous-facteurs de la suite des lettres de w

Def Résiduel

On appelle résiduel de f par rapport à L et on note $f^{-1}L$ le langage :

$$f^{-1}L = \{ g \in A^* \mid fg \in L \}$$

Def Rationnelle

Une partie L de A^* est une partie rationnelle de A^* si on peut l'écrire en utilisant uniquement un nombre fini d'opération d'union, de produit ou d'étoile et de parties finies. Une telle expression est dite expression rationnelle

$$A = \{a, b\} \quad L = \{ f \in A^* \mid |f| = 2p \text{ avec } p \in \mathbb{N} \} \cdot \bigcup_{n \geq 0} (A \cdot A)^n = (A \cdot A)^*$$

Notation $Rat(A^*)$ est l'ensemble des parties rationnelles de A^*

Def Reconnaissable

Un langage est dit reconnaissable sur l'alphabet A si il existe un AF qui le reconnait

On note $Rec(A^*)$ la famille des langages sur l'alphabet A qui sont reconnaissables

Def Déterministe

Un AF est déterministe si pour chaque état il n'y a qu'un seul chemin par lettre de A

Def Complet

Un AF est complet si pour chaque état il y a un chemin pour chaque lettre de A

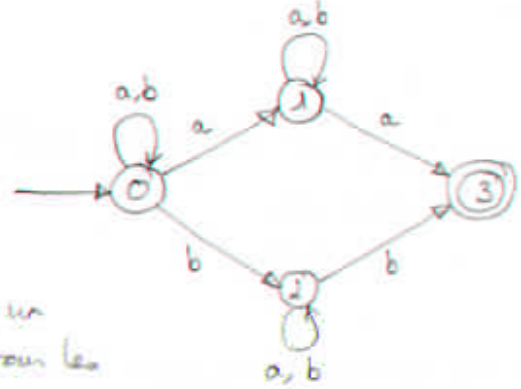
Def Emondé

On appelle automate Emondé, un automate fini dans lequel tous les états sont accessibles à partir de l'état initial et d'où au moins un état final est accessible

Théorème de Kleene :

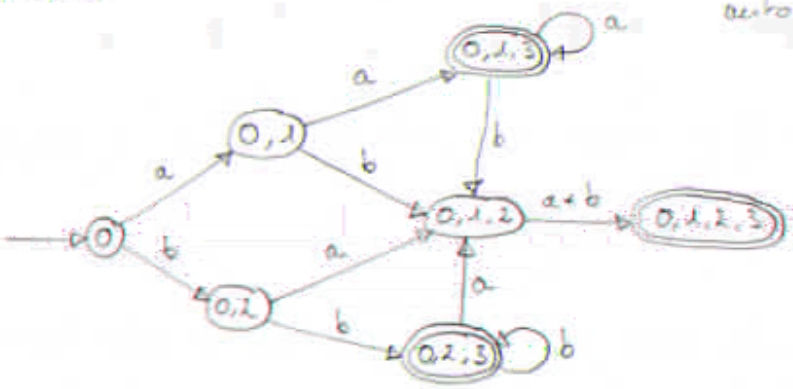
$$Rec(A^*) = Rat(A^*)$$

Déterminisation



L'état initial est un état qui possède tous les numéros des états initiaux de l'automate

Les états finaux sont les états qui ont un numéro d'un état final de l'automate



Pour avoir un AFDC, il suffit de rajouter un état poubelle

Def. Local

Un langage L est local si on peut le mettre sous la forme :

$$L = (XA^* \cap A^*Y) \setminus A^* w A^* \text{ , où } X, Y \subset A, w \in A^2$$

Prop

Si L et M sont deux langages reconnaissables alors $L \cdot M$ est un langage reconnaissable

Si " " " " $L+M$ " " "

$L \cap M$ " "

$\overline{(L \cap M)} = \overline{L} \cup \overline{M}$

Prop:

$L \cdot (M \cup N) = (L \cdot M) \cup (L \cdot N)$ mais ce qui est faux $L \cdot (M \cap N) \neq (L \cdot M) \cap (L \cdot N)$

Opérations sur les automates

[illegible]

Thomson

(A partir d'un langage rationnel fait un AF)

Eléments de base

$x \in A$ reconnait $\{w \in A \mid |w| = 1\}$

reconnait le mot vide

reconnait

reconnait

reconnait

reconnait

reconnait

reconnait

Algorithme

1 — On applique les éléments de base

2 — On ajoute les ε-transitions

dans E

(q, ε, q')

2 — On ajoute les x-transitions

dans E

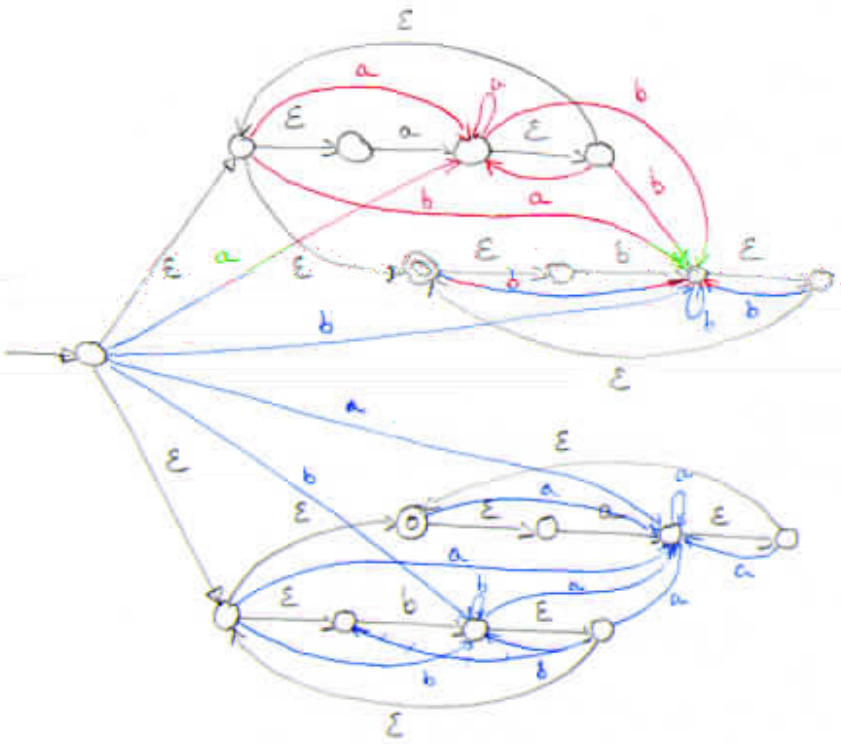
3 — On ajoute les états finaux supplémentaires

On ajoute à F tous les états q tels que $(q, \varepsilon, q') \in \delta$ et $q' \in F$

4 — Puis on enlève toutes les ε-transitions

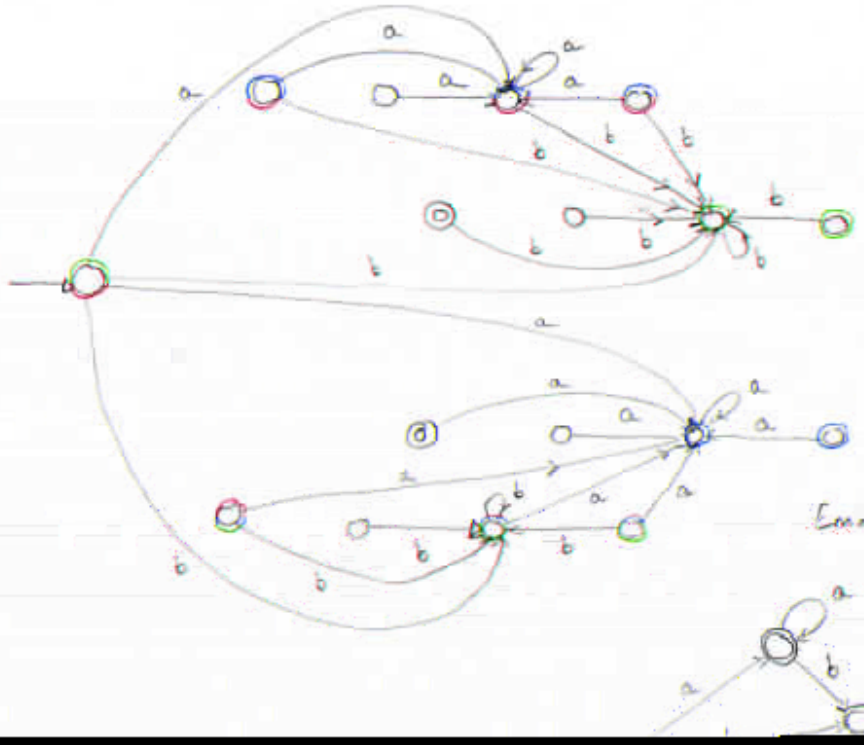
Ex

$L = a^* b^* + b^* a^*$



Je passe l'étape des ε-transitions afin de laisser l'automate plus lisible

Je redessine l'automate avec l'étape 4 avant l'étape 3, puis je fais l'étape 3



Emondons l'automate

Glushkov

(A partir d'un langage rationnel faire un AF)

Ex $L = (a(b+\varepsilon)a)^*$

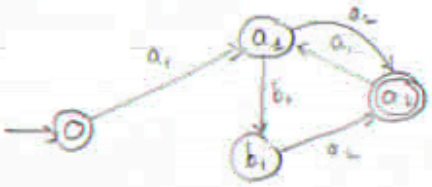
- On met des numéros aux lettres

$$L = (a_1(b_1+\varepsilon)a_2)^*$$

- On fait le tableau

$(a_1(b_1+\varepsilon)a_2)^*$	
a_1	début
a_2	fin
$a_1 \to b_1 + a_2$ $b_1 \to a_2$ $a_2 \to a_1$	Successeurs
oui (ε est accepté)	Vide (pour savoir si l'état de départ est acceptant)

- Dessinons l'automate



On crée un état de départ
Pour chaque lettre on crée un état
On ajoute les transitions, avec les lettres de l'état d'arrivé

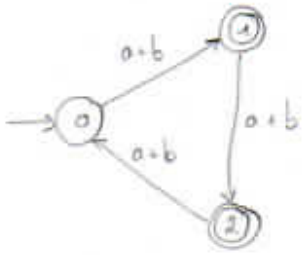
- Puis on supprime les numéros

Une fois qu'on a cet automate on peut faire les algo qu'on veut (déterminisé ⊙ par exemple)

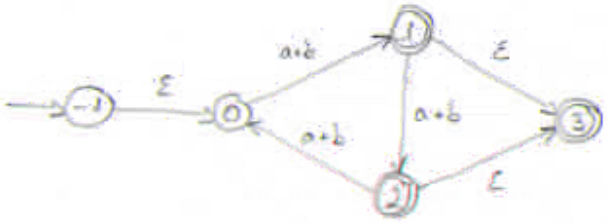
Mc Naughton et Yamada

(A partir d'un AF obtenir une expression Rationnelle)

Ex.



- On ajoute un état initial et un état final tels que :

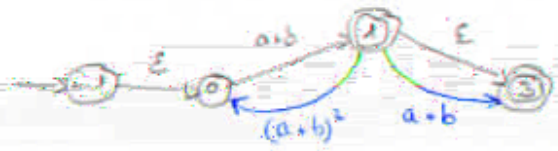


L'état initial pointe vers les états initiaux avec une ε-transition

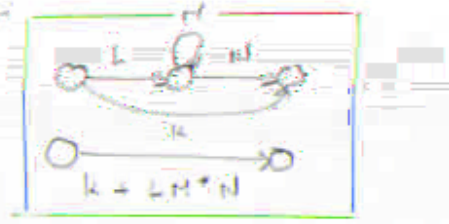
L'état final est accessible à partir des états finaux avec une ε-transition

- Puis on supprime tous les états

- Supprimons (2)

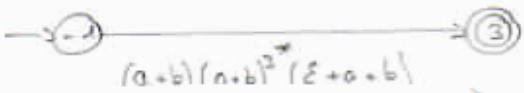


On crée les nouvelles transitions avec :



- Supprimons (0)

- Supprimons (1)



Donc $L = (a+b)[(a+b)^2]^*(\varepsilon+a+b)$

Equation Linéaire Droite

/A quoi ça sert ? Déterminiser des automates

Lemme d'Arden (Pour résoudre les equations)

/* $L_i = (Exp)\,L_i + (Exp)' \Rightarrow L_i = (Exp)^*(Exp)'$ */

La plus petite solution des systèmes $v = Xv + Y$ est $v = X^*Y$

Ex:

L'automate doit être déterministe

On écrit les équations de chaque état

$L_3 = \varepsilon + aL_1 + cL_2$

$L_2 = \varepsilon + aL_1 + bL_2 + cL_3$

$L_1 = \varepsilon + aL_1 + bL_2$

On cherche à avoir que les états initiaux = plusieurs états initiaux ou en fait l'union

$L_2 = b^*(\varepsilon + aL_1 + cL_3)$ par le lemme d'Arden

$L_3 = \varepsilon + bL_2 + ab^*(\varepsilon + aL_1 + cL_3) = \varepsilon + bL_2 + ab^*\varepsilon + ab^*aL_1 + ab^*cL_3$

$L_3 = (ab^*c)^*(\varepsilon + ab^*\varepsilon + (b + ab^*a)L_1)$

$L_1 = a(ab^*c)^*(\varepsilon + ab^*\varepsilon + (b + ab^*a)L_1) + cb^*(\varepsilon + aL_1 + c(ab^*c)^*(\varepsilon + ab^*\varepsilon + (b + ab^*a)L_1))$

$L_1 = a(ab^*c)^*(\varepsilon + ab^*) + cb^*c(ab^*c)^*(\varepsilon + ab^*) + (b + ab^*a)L_1 + cb^*aL_1 + cb^*c(ab^*c)^*(\varepsilon + ab^*) + cb^*c(ab^*c)^*(b + ab^*a)L_1 + cb^*(\varepsilon + c(ab^*c)^*(\varepsilon + ab^*))$

$L_1 = [(a + cb^*c)(ab^*c)^*(ab^*a + b) + cb^*a]^*[cb^* + (a + cb^*c)(ab^*c)^*$

Monoïde de Transition

Il faut que l'automate soit déterministe

Rappel :
Monoïde : ensemble avec une opération et un élément neutre

Ex :

On fait un tableau des états

mot à lire	0	1	2
ε	0	1	2
a	2	1	0
b	1	0	2
aa	1	2	0
ab	0	1	2
ba	2	0	1
bb	0	2	1
aba	1	0	2
bab	2	1	0

← état de départ
← état d'arrivée

On remarque les doubles en écrivant les égalités

donc $aa = \varepsilon$

donc $bb = \varepsilon$

donc $aba = bab$

On a donc

$T = \{\varepsilon, \bar{a}, \bar{b}, \bar{a}\bar{a}, \bar{a}\bar{b}, \bar{b}\bar{a}, \bar{a}\bar{b}\bar{a}\}$

Inutile d'aller plus loin car après on retomberait sur des équivalences

Puis on fait le monoïde de transition

	b	ab	ba	aba
b	b	ab	ba	aba
ab	b	aba	ba	
ε	aba	a	ab	
a	ba	ε	b	
aba	ε	ab	a	
ba	a	b	ε	

	ε	a	
ε	ε	a	
a	a	ε	
b	b	ba	
ab	ab	aba	
ba	ba	b	
aba	aba	ab	

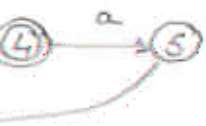
Algo de Moore

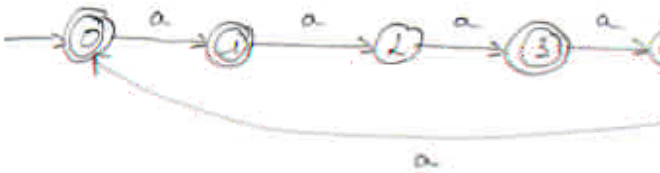
(Minimisation d'un automate)

Idée

Plusieurs états sont de la même classe si pour tous ces états, par une même lettre on accède à une même classe (pour toutes les lettres)

Ex Il faut que l'automate soit déterministe



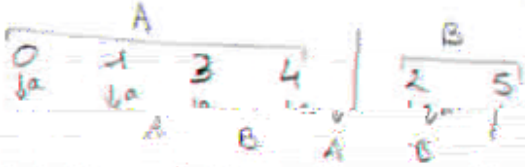


et les états non finaux

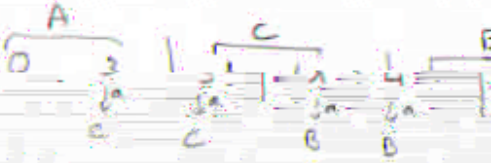
On sépare en deux classes états finaux

Tous les états de B par "a" vont en A

Donc pas besoin de subdiviser cette classe



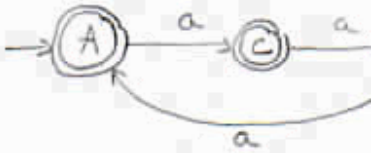
On a divisé au maximum, on a donc les équivalences de Weirdo



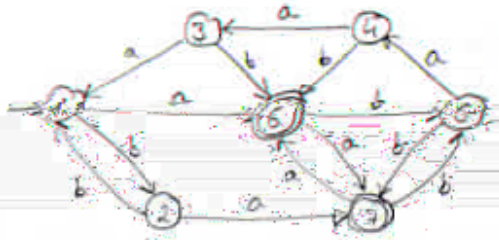
Equivalence de Weirdo (rel

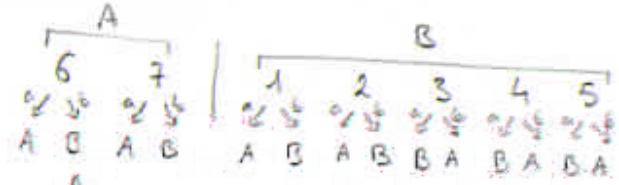
On a 3 et 4 2

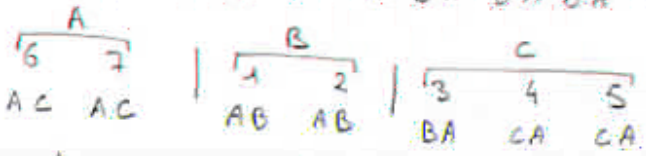
Puis on dessine l'automate mi

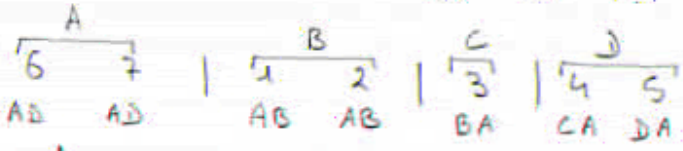


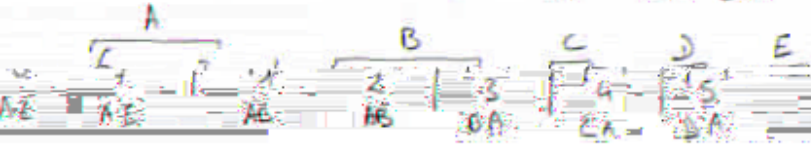
Ex Soit un automate plus que :







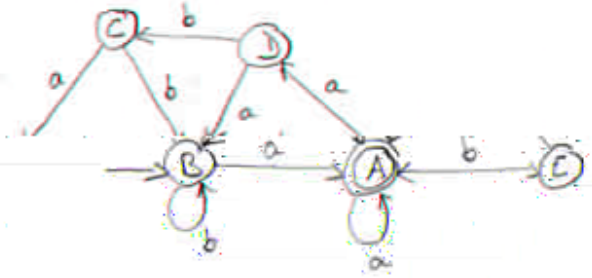




Donc : équivalence de Nérode

$6 \sim 7 \quad 1 \sim 2$

Dessinons l'automate



Formule des dérivées :

$$a^{-1}(L+M) = a^{-1}L + a^{-1}M$$

$$a^{-1}(L^*) = (a^{-1}L)L^*$$

$$a^{-1}(L.M) = \begin{cases} (a^{-1}L).M & \text{si } \varepsilon \notin L \\ (a^{-1}L).M + a^{-1}M & \text{si } \varepsilon \in L \end{cases}$$