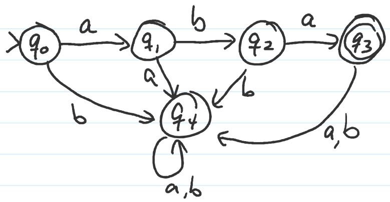
Automates à états finis

28 mars 2023 10:20

Un automate à états finis sert à détecter des mots qui satisfont une syntaxe qu'on appelle régulière.

ex :



○ = états

⟶ = transitions

>○ = état initial

◎ = état finaux

δ

(q_0, a, q_1)

(q_0, b, q_4)

(q_1, b, q_2)

$\vdots$

(q_3, a, q_4)

(q_3, b, q_4)

ex : aba : $q_0 \to q_1 \to q_2 \to q_3$ ✓ **accepté**

abab : $q_0 \to q_1 \to q_2 \to q_3 \to q_4$ ✗ **rejeté**

- Sert à faire de la recherche de mots dans un texte
- Sert à évaluer des expressions régulière (regex)
  ex : [a-z]*@[a-z]*.[a-z]*

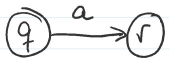
Automate fini déterministe (AFD)

Un AFD est un quintuplet $\langle Q, \Sigma, \delta, q_0, F \rangle$ où

- Q est un ensemble fini d'états
- Σ est un ensemble fini de symboles, appelé l'alphabet // $\Sigma = \{a, b\}$
- $q_0 \in Q$ est l'état initial
- $F \subseteq Q$ est l'ensemble des états finaux
- $\delta \in (Q \times \Sigma) \to Q$ // δ est une fonction qui reçoit $(q, a) \in Q \times \Sigma$ et "retourne" $r \in Q$

$q \xrightarrow{a} r \quad ((q, a), r) \in \delta$

// et "retourne" $r \in Q$



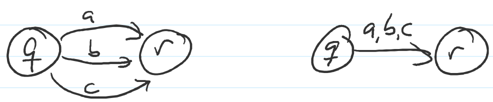
$((q,a),r) \in \delta$

Souvent, on écrit $((q,a),r) \in \delta$ avec

(q,a,r) ou encore

$q \xrightarrow{a} r$

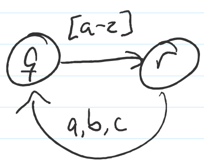
- On peut étiqueter une transition par un ensemble de symboles au lieu d'avoir une trans. par symbole.



- On dénote $[a-z] = \{a, b, c, \dots, z\}$

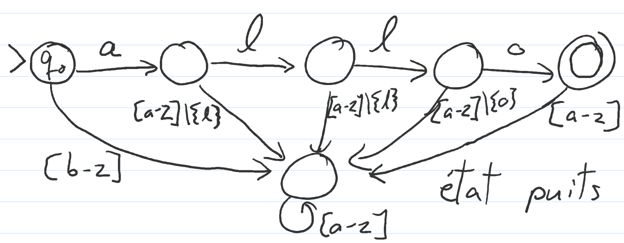
  $[0-9] = \{0, 1, 2, \dots, 9\}$

ex:



- Un mot w est accepté par l'automate si, en suivant les caractères de w sur l'automate, on termine dans un état de F.

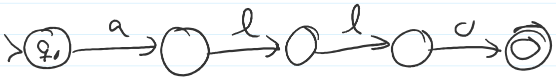
- ex: accepte seulement "allo"



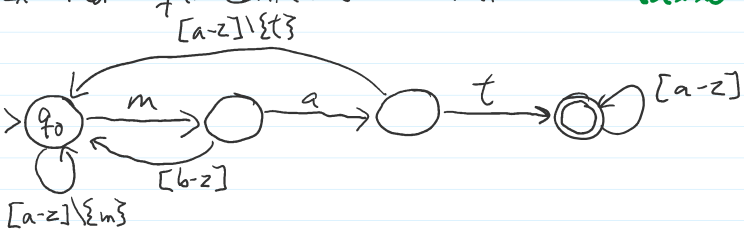
- Un état puits est un état duquel on ne peut pas sortir.
  On peut omettre les états puits. Si on a une transition non-spécifiée, on suppose qu'elle va à un

sortir.

On peut omettre les états puits. Si on a une transition non-spécifiée, on suppose qu'elle va à un puits.



- ex: mots qui contiennent "mat" mallomatxy



Accepter vs rejeter

- Un <u>mot</u> est une séquence de symboles tous dans Σ

  Plusieurs dénotations: $[\sigma_1, \sigma_2, \sigma_3, \dots, \sigma_n]$, $\forall i, \sigma_i \in \Sigma$

  $(\sigma_1, \sigma_2, \sigma_3, \dots, \sigma_n)$

  $\sigma_1 \sigma_2 \sigma_3 \cdots \sigma_n$

  ex: allo $[a, l, l, o]$

  (a, l, l, o)

  allo

  Synonyme: mot, chaîne de caractères, suite, séquence

- Σ^* : l'ensemble (infini) de tous les mots possibles sur alphabet Σ
- [ ] est le mot vide (0 caractères)
  parfois, $[\,] = \varepsilon$ (petit epsilon)
- Concaténation: soit $u \in \Sigma^*$ et $v \in \Sigma^*$, on

- Concaténation : soit $u \in \Sigma^*$ et $v \in \Sigma^*$, on dénote par $u \frown v$ la concaténation de u et v, i.e. u suivi de v.

ex : $u =$ allo $v =$ salut, $u \frown v =$ allosalut

$u \frown [\,] = u \qquad [\,] \frown u = u$

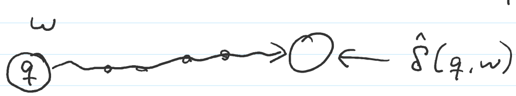
- Soit $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ un AFD, on définit

$L(A) = \{ w \mid w \in \Sigma^* \text{ et } w \text{ est accepté par } A \}$

On dit que L est le <u>langage</u> de A.

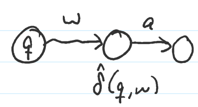
- On définit $\hat{\delta} \in (Q \times \Sigma^*) \to Q$

l'extension de δ à des mots. $\hat{\delta}(q, w)$ est l'état où on termine si on démarre à q et on lit w.



$\hat{\delta}(q, [\,]) = q$

$\hat{\delta}(q, w \frown [a]) = \delta(\hat{\delta}(q, w), a)$



$L(A) = \{ w \mid w \in \Sigma^* \wedge \hat{\delta}(q_0, w) \in F \}$

<u>Automate fini non-déterministe</u> (AFND)

Un AFND est quintuplet $\langle Q, \Sigma, \delta, q_0, F \rangle$

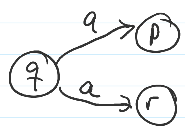
avec

- Q, Σ, q_0, F sont comme dans les AFD avec

- Q, Σ, q_0, F sont comme dans les AFD
- $\delta \subseteq (Q \times \Sigma) \times Q$

possible : $(q, a, p) \in \delta$

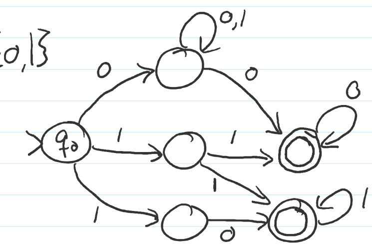
$(q, a, r) \in \delta$ (en même temps)



Interprétation : état courant q, voit a, on choisit si on va à p ou r.

- Un mot w est accepté par l'automate s'il existe un choix de séquence de transitions en lisant w qui termine dans un état de F.

ex : $\Sigma = \{0,1\}$



100101 ✗

101111 ✓

0011011100000 ✓

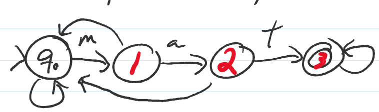
001101 ✗

- Soit M les mots qui contiennent "mat"

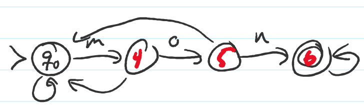
ou qui contiennent "mon"

Produire un AFND A tel que $L(A) = M$

- AFD pour mat



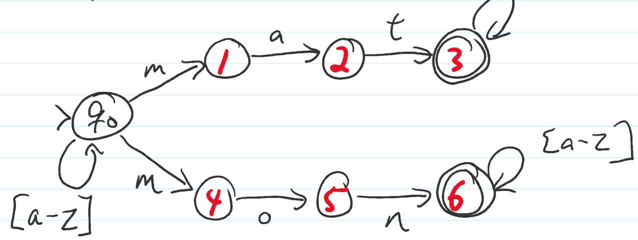
mon



- AFND pour mat ou mon

$q_0 q_0 q_0 q_0 q_0 1\ 2\ 3\ 3\ \underline{3}$

- AFND pour mat ou mon



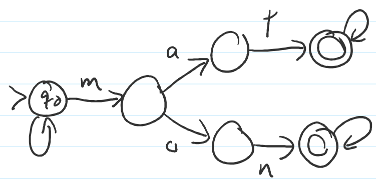
$q_0 q_0 q_0 q_0 q_0 1 2 3 3 \underline{3}$

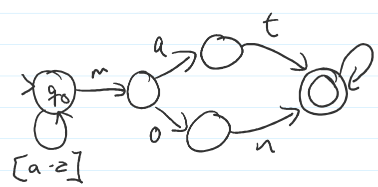
xymaxmatxy

xymaxmonxy

$q_0 q_0 q_0 q_0 q_0 4 5 6 6 \underline{6}$

xymonxmaty





Théorème: soit A un AFND. Alors il existe un AFD D tel que
$$L(A) = L(D).$$

En d'autres termes, on peut "convertir" un AFND en AFD.

- Acceptation par un AFND

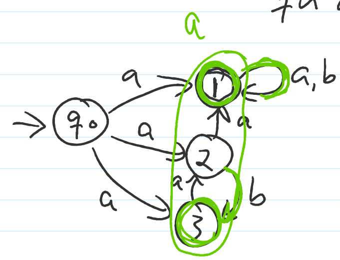
Soit $A = \langle Q, \Sigma, q_0, F, \delta \rangle$ un AFND.
On définit

Soit $A = \langle Q, \Sigma, q_0, F, \delta \rangle$ un AFND.

On définit

$\hat{\delta}(q, w)$ = ensemble des états accessibles si on est dans l'état q et qu'on lit w.

ex:

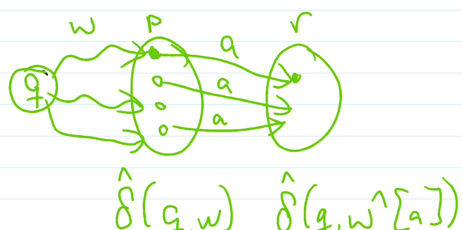


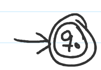
$\hat{\delta}(q_0, [a]) = \{1, 2, 3\}$

$\hat{\delta}(q_0, [a,b]) = \{1, 3\}$

$$\hat{\delta}(q, [\,]) = \{q\}$$

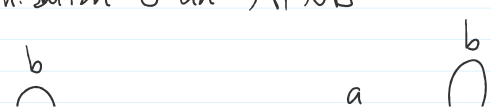
$$\hat{\delta}(q, w \frown [a]) = \{ r \mid r \in Q \ \wedge \ \exists p \cdot (p \in \hat{\delta}(q,w) \ \wedge \ (p, a, r) \in \delta) \}$$

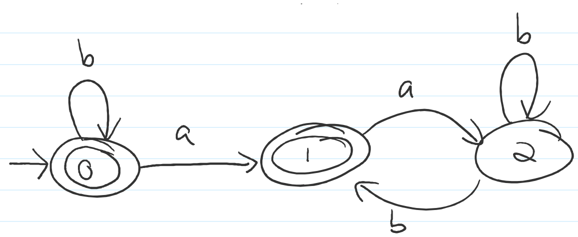




<u>Déf.</u>: Un AFND A accepte w si $\hat{\delta}(q_0, w) \cap F \neq \emptyset$.

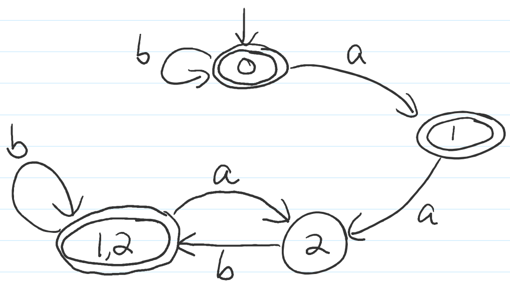
Déterminisation d'un AFND

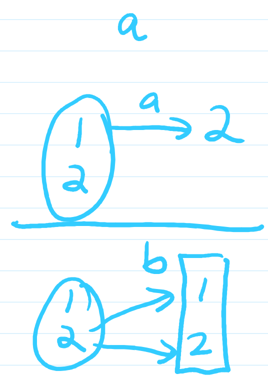




AFD:

$w = baaba$



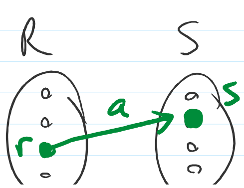


représente: on pourrait être dans 1, ou dans 2 (on "choisit")
final car on pourrait être dans $1 \in F$

- Pour déterminiser $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ (AFND) en AFD $A' = \langle Q', \Sigma, \delta', q_0', F' \rangle$

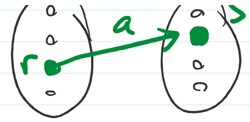
  1) États de A': $Q' = \mathbb{P}(Q) = \{ R \mid R \subseteq Q \}$
  2) États de départ: $q_0' = \{ q_0 \}$
  3) États finaux: $F' = \{ R \mid R \subseteq Q \wedge R \cap F \neq \emptyset \}$
  4) Transitions δ': $R \xrightarrow{a} S$ si, $\forall s \in S$, $\exists r \in R$ tel que $(r, a, s) \in \delta$

AFD



chaque état de S
est accessible par

AFD

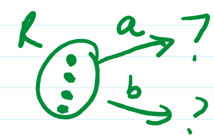


chaque état de S est accessible par un état de R quand il voit a

$R, S \in Q'$

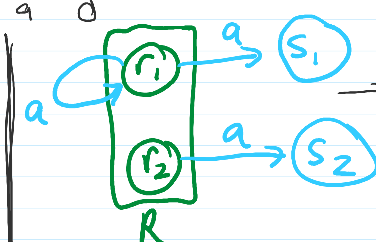
Méthode algorithmique

- AFND $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ un AFND
- Démarrer à $Q' = \emptyset$, $F' = \emptyset$, $\delta' = \emptyset$
- Définir un ensemble T // états à traiter
- Ajouter $\{q_0\}$ à T, ajouter $\{q_0\}$ à Q'
- Tant que $T \neq \emptyset$

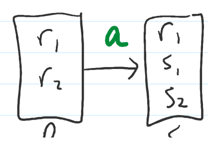


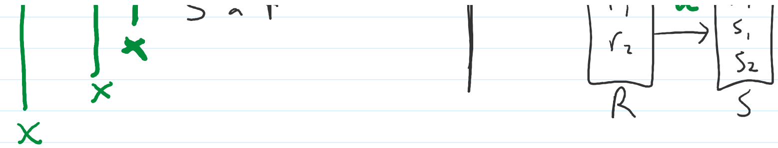
    choisir $R \in T$
    retirer R de T
    Pour chaque symbole $a \in \Sigma$
        $S = \{ s \mid s \in Q \wedge \exists r \cdot ((r, a, s) \in \delta) \}$
        Ajouter (R, a, S) à δ'

        si $S \notin Q'$
            Ajouter S à T
            Ajouter S à Q'
            si $S \cap F \neq \emptyset$, ajouter S à F'

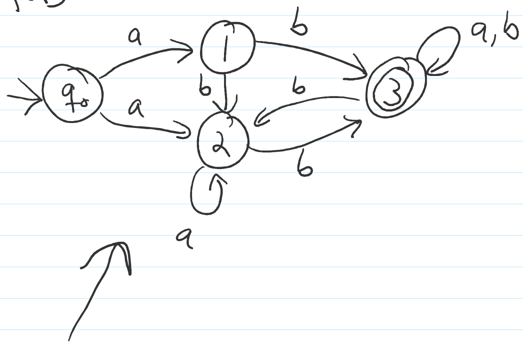


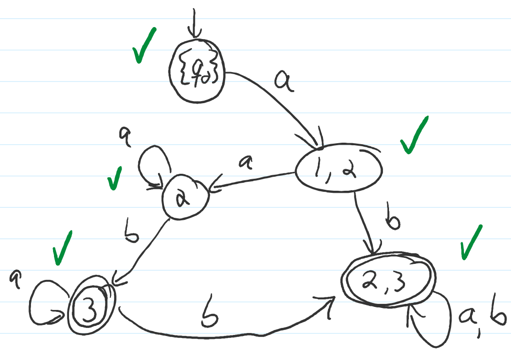
$\rightarrow \{r_1, s_1, s_2\}$

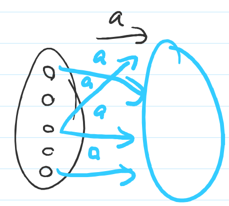


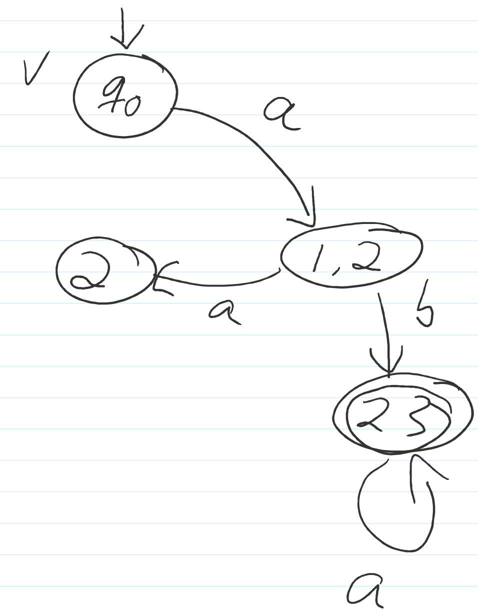


AFND





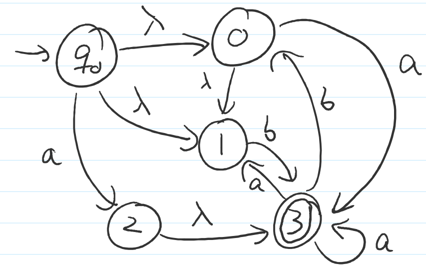




λ-Transitions (λ : lambda)

- Certains les appellent ε-transitions
- Dans un AFND, certains permettent des transitions "vides"

veut dire "si on est dans p, on peut aller à q sans lire de caractère" ex:

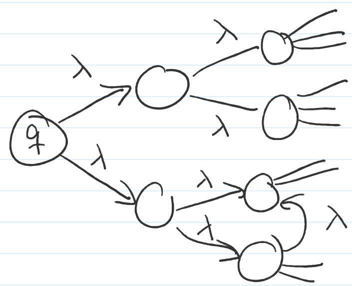


accepte [a]

$q_0 \xrightarrow{a} 2 \xrightarrow{\lambda} 3$

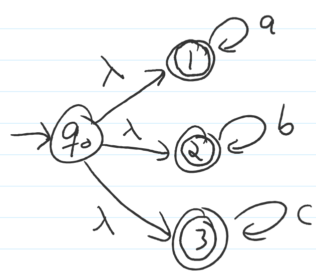
$q_0 \xrightarrow{\lambda} 0 \xrightarrow{a} 3$

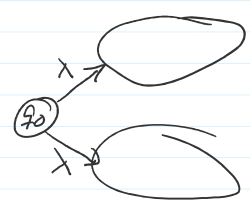
accepte: bbbbb $\quad q_0 \xrightarrow{\lambda} 1 \xrightarrow{b} 3 \xrightarrow{b} 0 \xrightarrow{\lambda} 1 \xrightarrow{b} 3 \xrightarrow{b} 0 \xrightarrow{\lambda} 1 \xrightarrow{b} 3$



- Parfois utile pour "simuler" plusieurs états de départ.

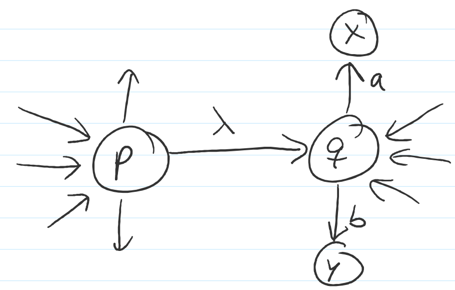
M = mots qui ne contiennent que des "a",
ou que des "b",
ou que des "c"

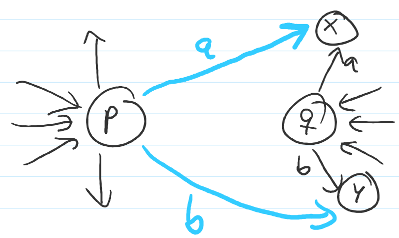


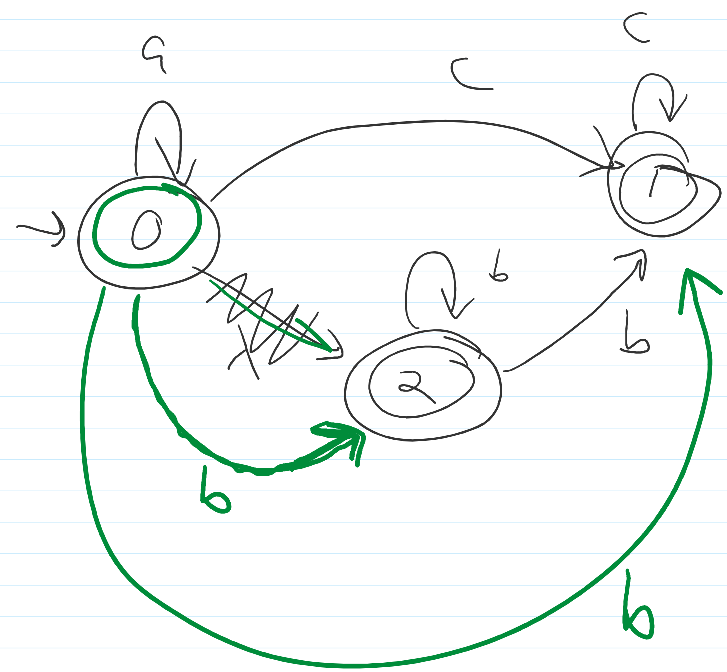


- Un mot w est accepté par un AFND A avec λ-transitions s'il existe un choix de transitions qui suit les caractères de w ou des λ-transitions (sans consommer de caractère de w), et qui mène à un état final.

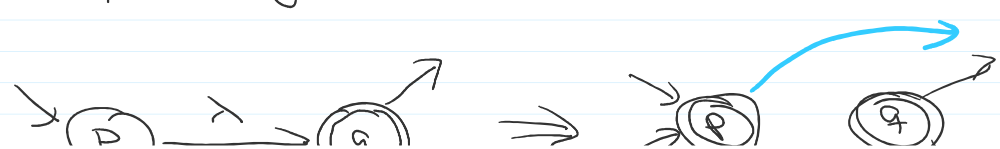
- Comment enlever <u>une</u> λ-transition



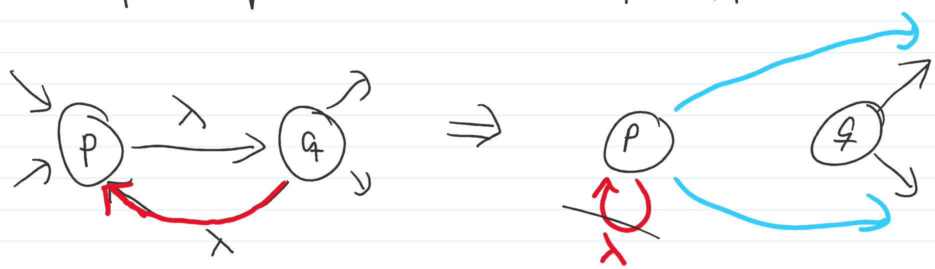




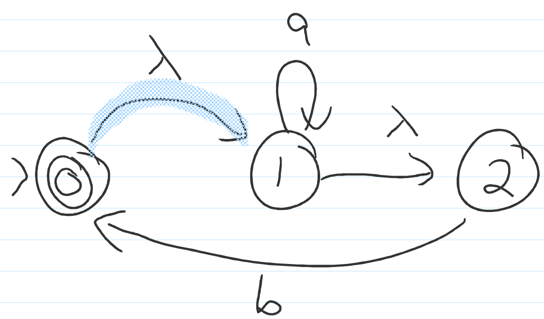
- Retirer (p, λ, q)
- Pour chaque $(q, a, r) \in \delta$
  ajouter (p, a, r) à δ
- si $q \in F$, ajouter p à F



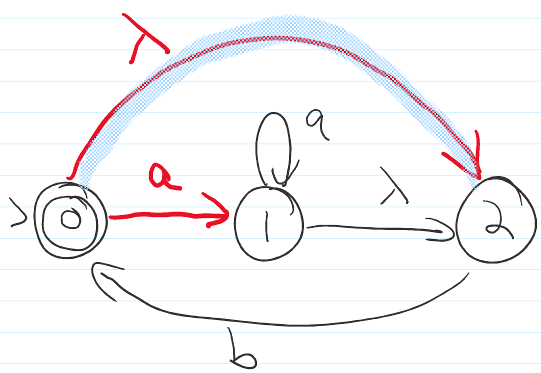
• si $(p, \lambda, p) \in \delta$, retirer (p, λ, p) de δ

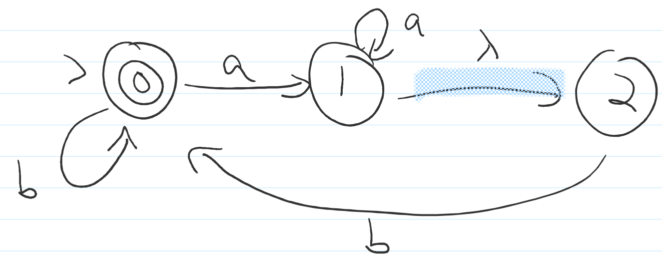


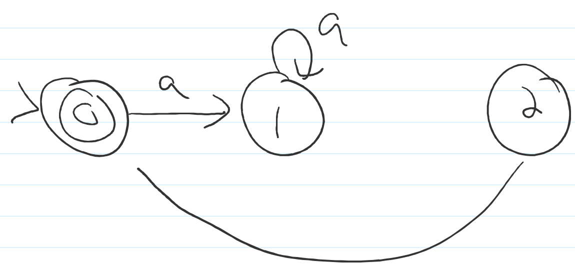
ex:

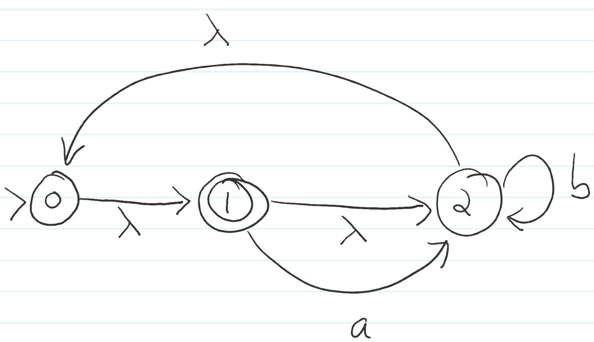


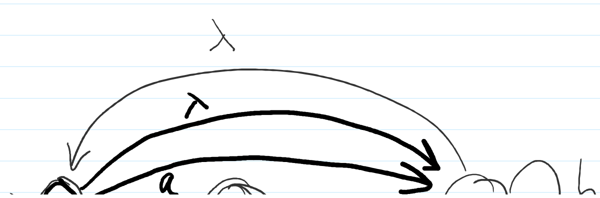
But: retirer les λ-transitions une par une.

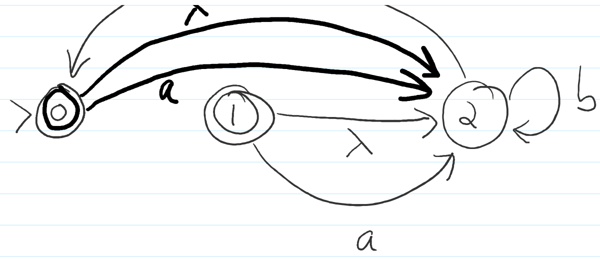


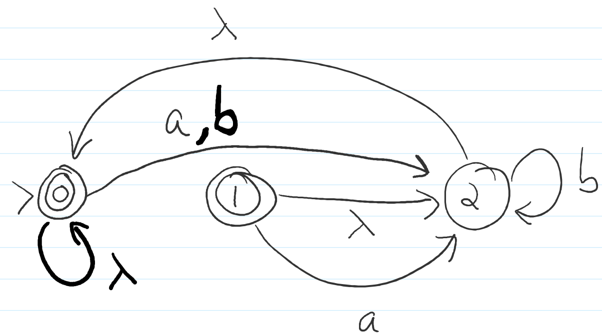

0
a
1
a
λ
2
b
b

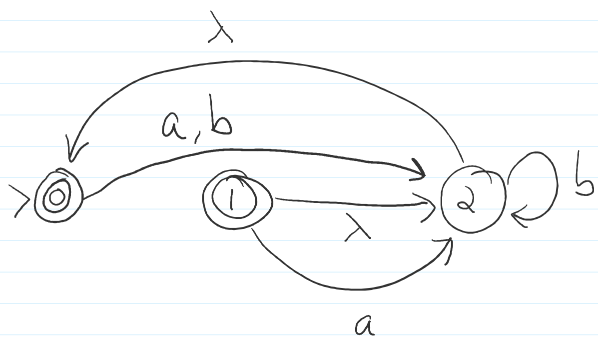

0
a
1
a
2

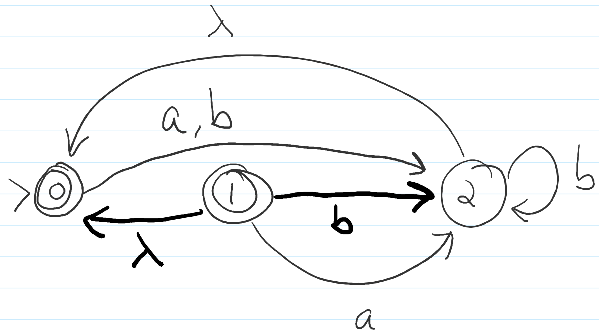

λ
0
λ
1
λ
2
b
a

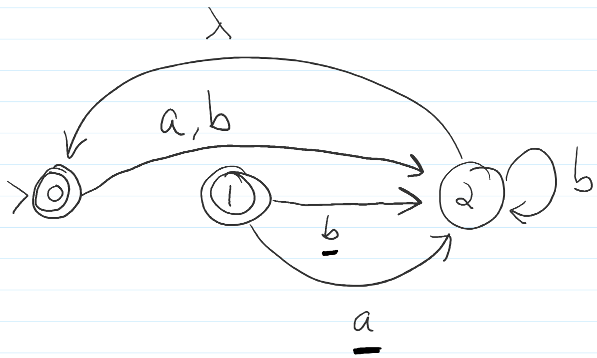

λ
λ
a
b

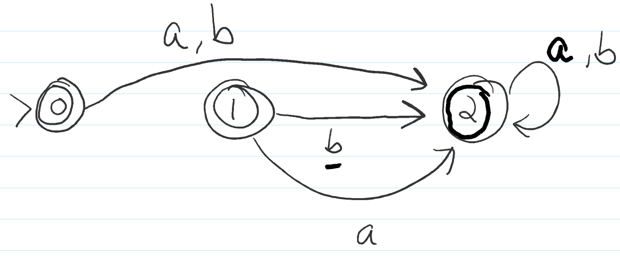

λ
a
0
1
λ
2
b
a

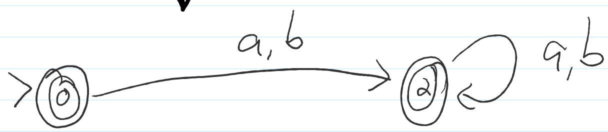

λ
a,b
0
1
λ
2
b
λ
a


λ
a,b
0
1
λ
2
b
a


λ
a,b
0
1
2
b
b
λ
a


λ
a,b
0
1
2
b
b
a


a,b
0
1
2
a,b
b
a


0
a, b
2
a, b