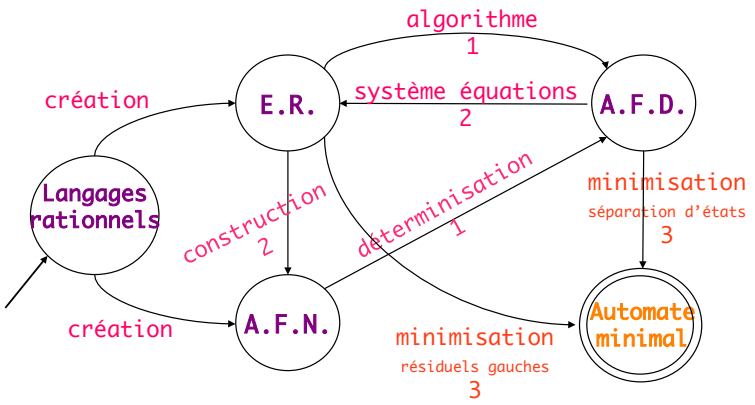
3 - Automate fini minimal

Un méta-automate ...



Minimisation

Théorème : chaque langage rationnel est reconnu par un unique automate déterministe minimal*.

☞ *deux problèmes de minimisation :*

- Donnée : une expression régulière E
- Problème: construire un A.F.D. minimal qui reconnaisse le langage décrit par E

- Idée : *les résiduels ...*

- Donnée : un automate fini déterministe
- Problème : construire un A.F.D. minimal qui reconnaisse le langage décrit par E

- Idée : *l'équivalences d'états ...*

* la minimalité porte sur le nombre d'états d'un automate ... *C O M P L E T.*

Langage associé à un état

Soit un A.F.D. $\mathcal{A} = (\Sigma, Q, \delta, q_o, F)$,

- langage associé à l'état q :

$$L_q(\mathcal{A}) = \{w \in \Sigma^*,\ \delta^*(q,w) \in F\}$$

- $L_q(\mathcal{A})$: langage reconnu par un automate dont l'état initial serait q et qui aurait F comme ensemble d'états finals.

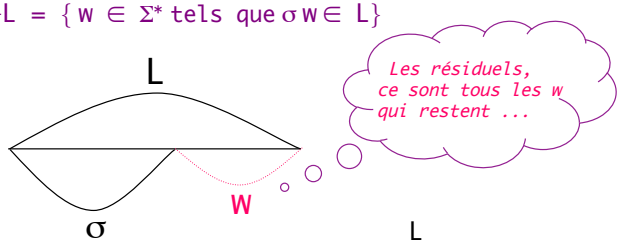
- $L(\mathcal{A}) = L_{q_0}(\mathcal{A})$

- un état q est accessible s'il existe un chemin de q_o à q dans $\mathcal{A}$.

Résiduels

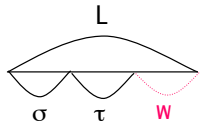
- L'ensemble des résiduels à gauche de L, noté R(L), est la réunion pour toutes les lettres σ de l'alphabet des ensembles $\{\sigma\}^{-1}L$:

$$\{\sigma\}^{-1}L = \{ w \in \Sigma^* \text{ tels que } \sigma w \in L\}$$



- *Petites propriétés :*

$\{\varepsilon\}^{-1}L = L$
$\varnothing^{-1}L = \varnothing$
$(\{\sigma\tau\})^{-1} L = \{\tau\}^{-1} (\{\sigma\}^{-1} L)$



Minimisation (1er problème)

Soit L un langage rationnel,

Théorème : *(admis)*
l'ensemble des résiduels à gauche de L noté R(L) est fini.

Proposition : *(admis)*
soit $A = (\Sigma, Q, \delta, q_0, F)$ un A.F.D. complet dont tous les états sont accessibles, on a :

$$R(L) = \{L_q(A),\ q \in Q\}$$

Conséquence :

A partir d'une expression régulière pour L,
on peut construire l'automate minimal qui reconnaît L
et dont chaque état correspond justement à un élément de R(L).

Exemple

1* 0 (0 + 1)

$0^{-1} L = 0 + 1$
$1^{-1} L = L$

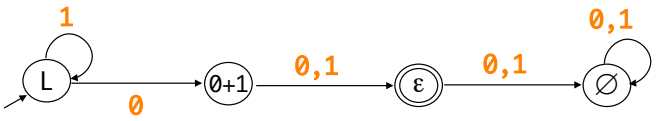
$0^{-1} (0+1) = \varepsilon$
$1^{-1} (0+1) = \varepsilon$

$0^{-1} \varepsilon = \varnothing$
$1^{-1} \varepsilon = \varnothing$

$0^{-1} \varnothing = \varnothing$
$1^{-1} \varnothing = \varnothing$

δ	0	1
→ L	0+1	L
0+1	ε	ε
← ε	∅	∅
∅	∅	∅

dans le tableau, les états apparaissent tout seul ...



Un autre exemple

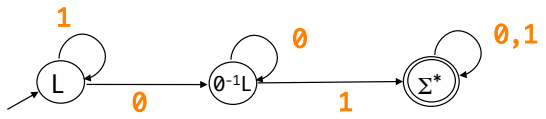
(0+1)* 0 1 (0+1)*

$0^{-1} L = L + 1\ (0+1)^*$
$1^{-1} L = L$

$0^{-1} (0^{-1}L) = 0^{-1}L$
$1^{-1} (0^{-1}L) = \Sigma^*$

$0^{-1} (\Sigma^*) = \Sigma^*$
$1^{-1} (\Sigma^*) = \Sigma^*$

δ	0	1
→ L	$0^{-1}L$	L
$0^{-1}L$	$0^{-1}L$	Σ^*
← Σ^*	Σ^*	Σ^*



Minimisation (2e problème)

- Donnée : A un A.F.D. complet dont chaque état est accessible depuis l'état initial
- Problème : construire un A.F.D. minimal* qui reconnaisse le même langage que A.
- Idée : faire en sorte que les états équivalents soient fusionnés.

En pratique,
l'algo. est fondé sur le principe de « séparation des états » :
- on commence par séparer les états finals des états non finals
- dans chaque classe, on sépare les états non équivalents
- on renouvelle récursivement cette opération ...

* *On rappelle que « minimal » sous-entend « complet ».*

Etats équivalents

- étant donné A un A.F.D, deux états p et q sont équivalents si leur langage associé sont identiques :

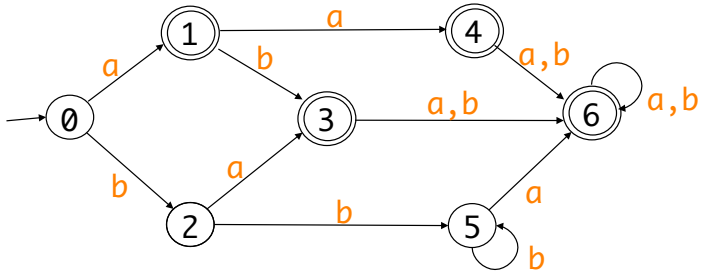
$$p \approx q \quad \text{ssi} \quad L_p(A) = L_q(A)$$

- autrement dit :

$$(p \approx q) \Leftrightarrow \left(\forall w \text{ de } \Sigma^*, \begin{cases} \delta^*(p,w) \in F \text{ et } \delta^*(q,w) \in F \\ \textit{ou bien} \\ \delta^*(p,w) \notin F \text{ et } \delta^*(q,w) \notin F \end{cases} \right)$$

- La relation $\approx$ est une relation d'équivalence.
- Si q est un état, on note $[q]$ l'ensemble des états qui lui sont équivalents.

Exemple



$0 \approx 4$? NON : car $\delta^*(0,ab) \notin F$ et $\delta^*(4,ab) \in F$

$3 \approx 6$? OUI : car d'une part $3 \in F$ et $6 \in F$
et d'autre part :
$\forall w \in \Sigma^*$, $\delta^*(3,aw) \in F$ et $\delta^*(6,aw) \in F$
et $\forall w \in \Sigma^*$, $\delta^*(3,bw) \in F$ et $\delta^*(6,bw) \in F$

Automate minimal

- Soit A un A.F.D. complet* dont chaque état est accessible depuis l'état initial :

$$A = (\Sigma, Q, \delta, q_0, F)$$

- l'automate minimal associé à A est :

$$A_{min} = (\Sigma, Q', \delta', [q_0], F')$$

  - $Q' = \{[q],\ q \in Q\}$
  - $\delta' = \{ ([p],\sigma,[q]) / \exists p' \in [p], \exists q' \in [q]\ (p',\sigma,q') \in \delta \}$
  - $F' = \{[f],\ f \in F\}$

* *En pratique, le « complet » n'est pas nécessaire à la mise en œuvre de l'algorithme, mais l'automate obtenu doit être complété afin d'obtenir l'automate minimal canonique.*

Propriétés

- $A_{min} = (\Sigma, Q', \delta', [q_0], F')$:
  - est bien défini
  - ne peut avoir deux états distincts équivalents
  - reconnaît le même langage que A.
- pour tout A.F.D. *complet* B tel que L(B) = L(A),
  le nombre d'états de B est supérieur ou égal à celui de A_{min}.
- tous les automates minimaux C tels que L(C) = L(A)
  sont identiques à un *renommage* de leurs états près.

➔ on peut parler d'*unicité* de l'automate minimal.

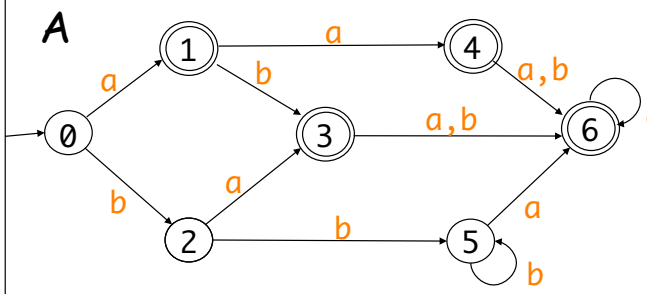
Algorithme de minimisation

Algorithme par « raffinements successifs »

On définit inductivement une suite d'équivalences :

- $p \approx_0 q \Leftrightarrow ((p \in F \text{ et } q \in F) \text{ ou } (p \notin F \text{ et } q \notin F))$
- $i>0 : p \approx_i q \Leftrightarrow \begin{cases} p \approx_{i-1} q \\ \textit{et} \\ \forall \sigma \text{ de } \Sigma : \delta(p,\sigma) \approx_{i-1} \delta(q,\sigma) \end{cases}$
- cas arrêt : $\approx_i$ est identique à $\approx_{i-1}$

Un premier exemple

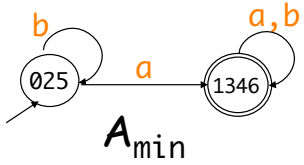


δ	a	b
0	1	2
1	4	3
2	3	5
3	6	6
4	6	6
5	6	5
6	6	6

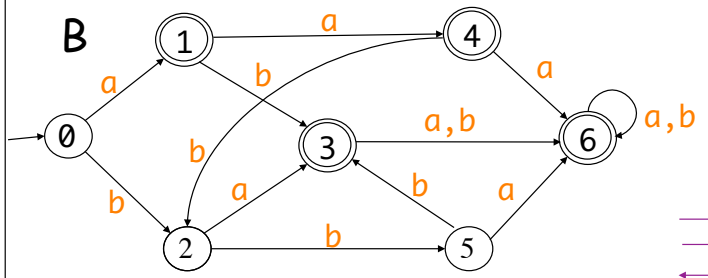
$\approx_0$: {0, 2, 5} {1, 3, 4, 6}

$\approx_1$: {0, 2, 5} {1, 3, 4, 6}

$\approx_0 = \approx_1$ donc arrêt de l'algorithme



Un autre exemple



δ	a	b
0	1	2
1	4	3
2	3	5
3	6	6
4	6	2
5	6	3
6	6	6

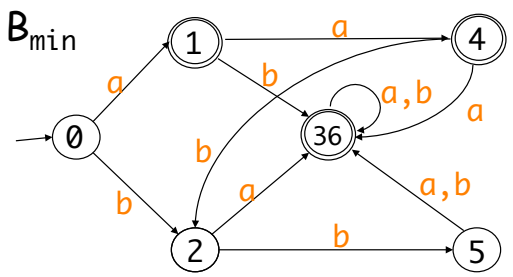
$\approx_0$: {0, 2, 5} {1, 3, 4, 6}

$\approx_1$: {{0, 2}, {5}, {1, 3, 6}, {4}}

$\approx_2$: {{0}, {2}, {5}, {1}, {4}, {3,6}}

$\approx_3 = \approx_2$: arrêt !

Fin !



$\approx$: {{0}, {2}, {5}, {1}, {4}, {3,6}}