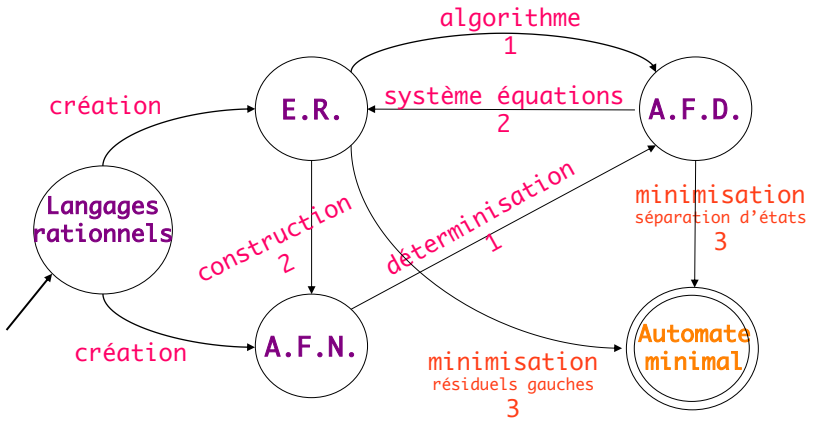
3 - Automate fini minimal

Un méta-automate ...



Minimisation

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

☞ *deux problèmes de minimisation :*

- Donnée : un automate fini déterministe A
- Problème : construire l'automate minimal A_{min} qui reconnaît le langage L(A) reconnu par A
- Idée : *les états équivalents ...*

- Donnée : une expression régulière E
- Problème: construire l'automate minimal A_{min} qui reconnaît le langage L(E) décrit par E
- Idée : *les résiduels ...*

* la minimalité porte sur le nombre d'états d'un automate DETERMINISTE ET COMPLET.

Minimisation (1er problème)

- Donnée : A un A.F.D. complet dont chaque état est accessible depuis l'état initial
- Problème : construire l'automate minimal* qui reconnaît 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 cette opération jusqu'à la stabilisation ...

* *On rappelle que « minimal » sous-entend à la fois « déterministe » ET « complet ».*

RAPPEL !!!

Langage associé à un état

Soit $\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$ un A.F.D.

- 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 l'automate $\mathcal{A}_q = (\Sigma, Q, \delta, q, F)$:
  - c'est presque le même automate que $\mathcal{A}$
  - excepté que q devient état initial à la place de q_0
- en particulier : $L(\mathcal{A}) = L_{q_0}(\mathcal{A})$
- un état q est accessible s'il existe un chemin de q_0 à q dans $\mathcal{A}$.

Etats équivalents

- étant donné $\mathcal{A}$ un A.F.D, deux états p et q sont équivalents si leurs langages associés respectifs sont identiques :

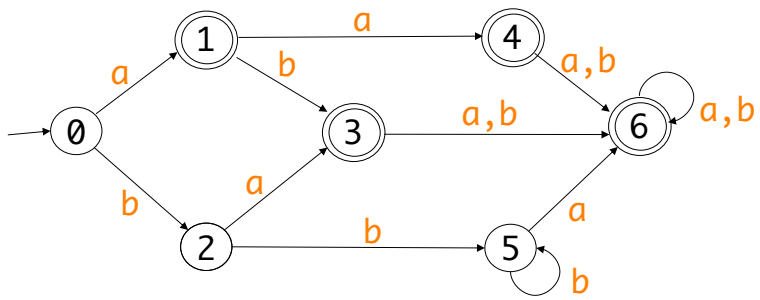
$$p \approx q \quad \text{ssi} \quad L_p(\mathcal{A}) = L_q(\mathcal{A})$$

- 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 $0 \notin F$ et $4 \in F$

$3 \approx 6$? 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 $\mathcal{A}$ un A.F.D. complet* dont chaque état est accessible depuis l'état initial :

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

- l'automate minimal associé à $\mathcal{A}$ est :

$$\boxed{\mathcal{A}_{min} = (\Sigma,\ Q',\ \delta',\ [q_0],\ F')}$$

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

* *En pratique, le « complet » n'est pas nécessaire à la mise en œuvre de l'algorithme, mais l'automate obtenu doit alors être <u>complété</u> a posteriori 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

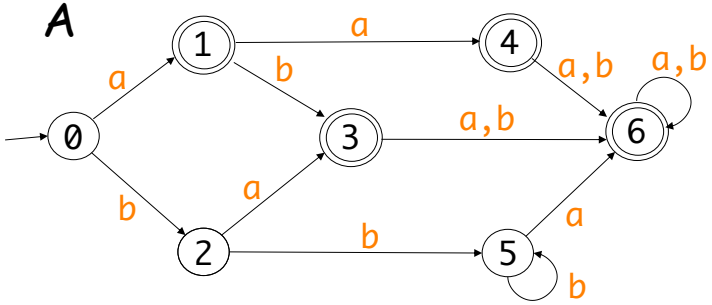
Algorithme de minimisation (1er problème)

Algorithme par « raffinements successifs »

On définit inductivement une suite d'équivalences sur Q :

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

Un premier exemple

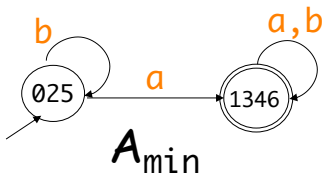


δ	a	b
0	1	2
1	4	3
2	3	5
3	6	6
4	6	6
5	6	5
6	6	6

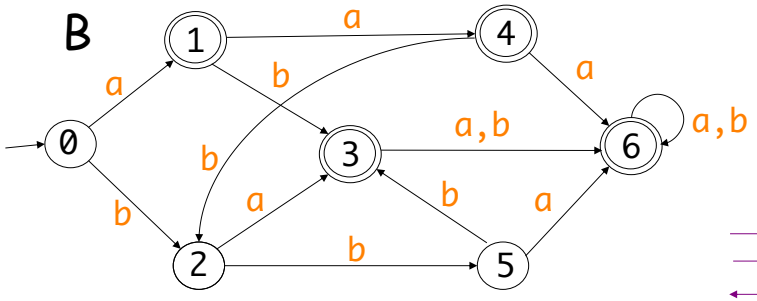
$\approx_0$: {0, 2, 5} {1, 3, 4, 6}

$\approx_1$: {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

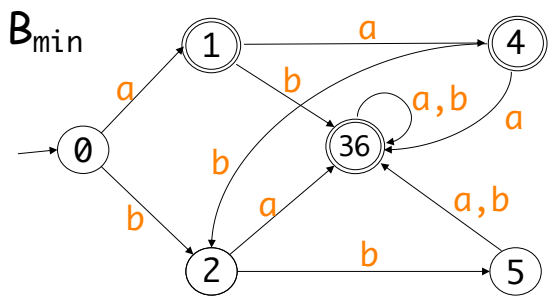
$\approx_0$: { {0, 2, 5} {1, 3, 4, 6} }

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

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

$\approx_3 = \approx_2$: arrêt !

Automate minimal

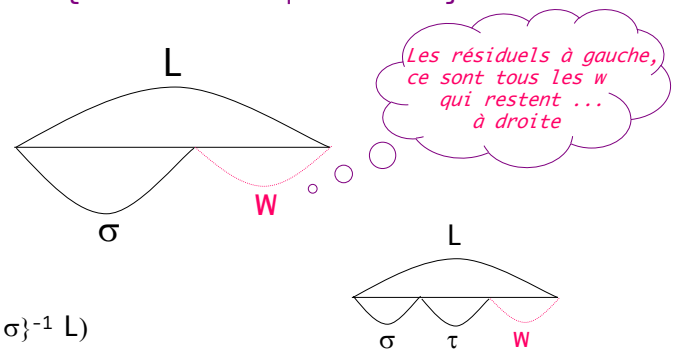


≈ : {{0}, {2}, {5}, {1}, {4}, {3,6}}

Résiduels

- L'ensemble des résiduels à gauche de L, noté R(L), est la réunion pour tous les mots σ de Σ^* des ensembles $\{\sigma\}^{-1}$ L :

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



- *Propriétés :*

$\{\varepsilon\}^{-1}L = L$

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

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

Minimisation (2e 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 immédiate :

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).

Calcul de l'automate minimal

Soit L un langage rationnel sur Σ donné sous forme d'une expression régulière, on construit l'automate minimal $M = (\Sigma, Q, \delta, q_0, F)$ le reconnaissant :

- q_0 correspond au langage $L_{q_0} = L$
- $Q = \{q_0\}$
- $i = 1$
- **pour** tout état q de langage associé L_q non encore traité **faire**
  - **pour** toute lettre σ de Σ **faire**
    - on calcule $\{\sigma\}^{-1}L_q$
    - **si** aucun état ne correspond à $\{\sigma\}^{-1}L_q$ **alors**
      - q_i nouvel état pour $\{\sigma\}^{-1}L_q$
      - $Q = Q \cup \{q_i\}$
      - $\delta = \delta \cup \{(q, \sigma, q_i)\}$
      - $i = i + 1$
- F contient tous les états q tel que $\varepsilon \in L_q$

Exemple

L: $\boxed{1^* \; 0 \; (\; 0 + 1 \;)}$

δ	0	1
→ L	0+1	L
0+1	ε	ε
← ε	∅	∅
∅	∅	∅

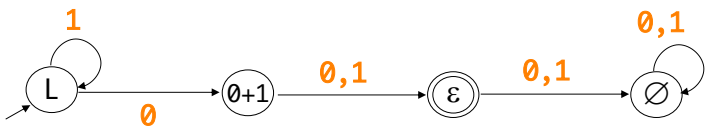
à chaque quotient apparaît un état ...

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

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

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

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



Un autre exemple

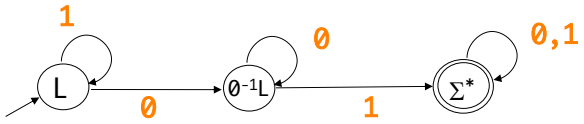
$\boxed{(0+1)^* \; 0 \; 1 \; (0+1)^*}$

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

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

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

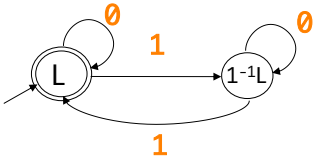
$0^{-1}\,(\Sigma^*) = \Sigma^*$
$1^{-1}\,(\Sigma^*) = \Sigma^*$



Un dernier exemple

$\boxed{0^* \; (\; 1 \; 0^* \; 1 \; 0^*)^*}$

δ	0	1
⇄ L	L	$1^{-1}L$
$1^{-1}L$	$1^{-1}L$	L

$0^{-1}\,L = L$
$1^{-1}\,L = 0^*10^*\,(10^*10^*)^*$

$0^{-1}\,(1^{-1}L) = 1^{-1}L$
$1^{-1}\,(1^{-1}L) = 0^*(10^*10^*)^*$
$= L$