Minimisation (suite)

Problème 2

- **Donnée** : A un AFD complet dont chaque état est accessible depuis l'état initial
- **Problème** : construire un AFD minimal qui reconnaisse le même langage que A.
- **Idée** : fusionner les états équivalents.
  En pratique, l'algorithme est fondé sur un principe de *séparation des états* ...

Équivalence d'états

Étant donné *A* un AFD, p et q $\in Q$ sont équivalents ($p \approx q$) si

- $L_p(A) = L_q(A)$

c.à.d.

- $\forall w \in \Sigma^*$

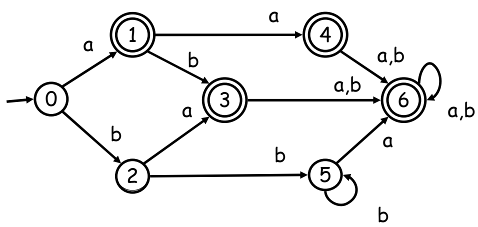
$\delta(p,w) \in F$ et $\delta(q,w) \in F$

ou

$\delta(p,w) \notin F$ et $\delta(q,w) \notin F$

Exemple

$\forall w \in \Sigma^* \begin{cases} \delta(p,w) \in F \text{ et } \delta(q,w) \in F \\ \text{ou} \\ \delta(p,w) \notin F \text{ et } \delta(q,w) \notin F \end{cases}$

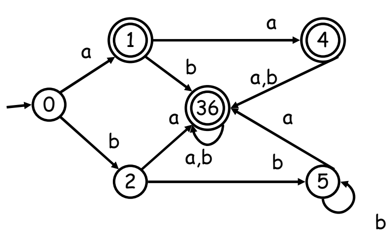


$0 \approx 1$? Non car $0 \notin F$ et $1 \in F$

$3 \approx 6$? Oui car $3,6 \in F$, $\delta(3,a)=\delta(6,a)$ et $\delta(3,b)=\delta(6,b)$

Exemple

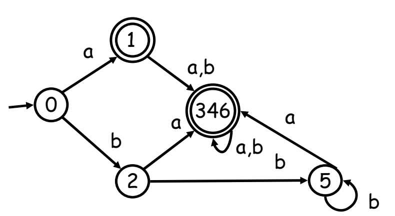
$\forall w \in \Sigma^* \begin{cases} \delta(p,w) \in F \text{ et } \delta(q,w) \in F \\ \text{ou} \\ \delta(p,w) \notin F \text{ et } \delta(q,w) \notin F \end{cases}$



$4 \approx 6$? Oui car $4,6 \in F$, $\delta(4,a)=\delta(6,a)$ et $\delta(4,b)=\delta(6,b)$

Exemple

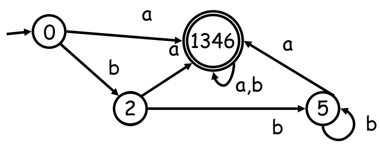
$\forall w \in \Sigma^* \begin{cases} \delta(p,w) \in F \text{ et } \delta(q,w) \in F \\ \text{ou} \\ \delta(p,w) \notin F \text{ et } \delta(q,w) \notin F \end{cases}$



$1 \approx 6$? Oui car $1,6 \in F$, $\delta(1,a) = \delta(6,a)$ et $\delta(1,b) = \delta(6,b)$

Exemple

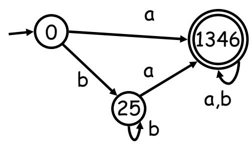
$\forall w\in\Sigma^* \begin{cases} \delta(p,w) \in F \text{ et } \delta(q,w) \in F \\ \text{ou} \\ \delta(p,w) \notin F \text{ et } \delta(q,w) \notin F \end{cases}$



$2\approx5$? Oui car $2,5\notin F$, $\delta(2,a)=\delta(5,a)$ et $\delta(2,b)=\delta(5,b)$

Exemple

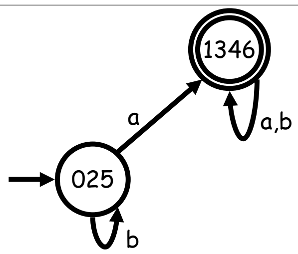
$\forall w\in\Sigma^* \begin{cases} \delta(p,w) \in F \text{ et } \delta(q,w) \in F \\ \text{ou} \\ \delta(p,w) \notin F \text{ et } \delta(q,w) \notin F \end{cases}$



$0\approx2$? Oui car $0,2\notin F$, $\delta(0,a)=\delta(2,a)$ et $\delta(0,b)=\delta(2,b)$

Exemple

$\forall w\in\Sigma^* \begin{cases} \delta(p,w) \in F \text{ et } \delta(q,w) \in F \\ \text{ou} \\ \delta(p,w) \notin F \text{ et } \delta(q,w) \notin F \end{cases}$



$0\approx1$? Non car $0\notin F$ et $1\in F$

Classe d'équivalence et automate associé

La relation ≈ est une relation d'équivalence (elle est réflexive, symétrique, transitive).

Si q est un état, on note [q] l'ensemble des états qui lui sont équivalents et on définit l'automate des classes d'équivalence :

Classe d'équivalence et automate associé

Étant donné un AFD $A = (\Sigma, Q, \delta, q_0, F)$, l'automate minimal associé à A est :

$$\mu A = (\Sigma, Q', \delta', [q_0], F')$$

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

Justification

- 3 étapes:
  - L'automate μA des classes d'équivalence de A est bien défini, réduit et $L(\mu A) = L(A)$.
  - Pour tout AFD B tel que $L(B) = L(A)$,
    $\#\text{états}(B) \geq \#\text{états}(\mu A)$
  - Tout automate minimal B tel que $L(B) = L(A)$, est **isomorphe** à A

Il existe une bijection ϕ entre les états de A et ceux de B qui préserve
- les états spéciaux (initial et d'acceptation)
- les transitions : $\forall p,q\in Q_A$, $\delta_A(p,a)=q \Leftrightarrow \delta_B(\phi(p),a)=\phi(q)$

μA est bien défini, réduit et $L(\mu A) = L(A)$

- Soient p et q deux états de A, $p \approx q$:
  - p et q sont tous deux soit dans F soit dans $Q \setminus F$. Les états terminaux de μA sont bien définis.
  - Si $L_p(A)=L_q(A)$ alors $\forall a \in \Sigma$, $L_{\delta(p,a)}(A) = L_{\delta(q,a)}(A)$. Les transitions de μA sont bien définies.
  - Si $w=w_1 \dots w_n \in \Sigma^*$, et $q_k = \delta(i, w_1 \dots w_k)$ alors
    $[\, q_k \,] = \delta'([i], w_1 \dots w_k)$. $\Rightarrow L(A)=L(\mu A)$
  - μA est un automate réduit par construction

13

Pour tout AFD B tel que $L(B)=L(A)$, $\#\text{états}(B) \geq \#\text{états}(\mu A)$

- On suppose tous les états de B accessibles et B complet.
  Soit $B = \langle \Sigma, Q_B, i_B, F_B, \delta_B \rangle$ tel que $L(B)=L(A)$.
- Soit g: $Q_B \rightarrow Q'$ l'application définie par
  $\forall q \in Q_B, \exists u \in \Sigma^* : \delta_B(i_B,u)=q$.
  $g(q) := \delta_{\mu A}([i],u)$
- Comme μA est réduit et $L(\mu A) = L(A)=L(B)$, cette application est bien définie et surjective
  (donc $\# Q_B \geq \#Q'$)

14

Tout automate minimal B tel que $L(B)=L(A)$, est isomorphe à μA

- En ce cas, comme g surjective et
  $\#\text{états}(\mu A)=\#\text{états}(B)$

g définit une bijection

μA et B sont isomorphes

Reste à construire l'automate réduit

15

Sur les quotients gauches

- L'automate Q(L) des quotients gauches défini comme $\{L_q(A) : q \in Q\} = Q(L)$ est-il bien minimal?
- Supposons, par l'absurde qu'il ne le soit pas. Alors il existe au moins p et q, deux états de l'automate tels que $L_p(A)=L_q(A)$, par définition de Q(L).
  - Si tel est le cas, par définition de l'équivalence, $p \approx q$.
  - Il s'ensuit que p et q peuvent être fusionnés, contredisant la minimalité de l'automate des quotients gauches.

16

Regroupement d'états

- Pour chaque paire d'états, il faut considérer l'ensemble des mots de longueur n sur Σ.
  $O(n^2)$ paires d'états
  $|\Sigma|^n$ mots de longueur n
  (n = nombre d'états de l'AFD)
- Algorithme en $O(n^2 \, |\Sigma|^n)$… catastrophique
- Trouver un meilleur algorithme !

17

Principe

- Au lieu de fusionner les états équivalents,
  - on groupe tous les états;
  - on sépare **inductivement** les états non équivalents;
  - quand on ne peut plus séparer on a terminé.
- La séparation inductive se fait en construisant inductivement $\approx$

18

Construction inductive de ≈

Base :

$p \approx_0 q \Leftrightarrow (p \in F \wedge q \in F) \vee (p \notin F \wedge q \notin F)$

- Règle :

$p \approx_i q \Leftrightarrow (p \approx_{i-1} q) \wedge (\forall a \in \Sigma, \delta(p,a) \approx_{i-1} \delta(q,a))$

La base permet de partitionner Q

La règle affine la partition de Q

Remarque : p $\approx_i$ q si on ne peut pas séparer p de q par un mot de longueur au plus i.

19

Cas d'arrêt

- dès que 2 équivalences successives coïncident

$\approx_i = \approx_{i+1} \Rightarrow \forall k, \approx_i = \approx_{i+k}$

- Par hypothèse, $\approx_i = \approx_{i+1}$. Alors

$p \approx_i q$ et $\forall a \in \Sigma, \delta(p,a) \approx_i \delta(q,a) \Leftrightarrow p \approx_{i+1} q$

$p \approx_{i+1} q$ et $\forall a \in \Sigma, \delta(p,a) \approx_{i+1} \delta(q,a) \Leftrightarrow p \approx_{i+2} q$

- Conséquence : dès qu'il y a coïncidence de 2 équivalences successives, on a obtenu l'automate minimal

20

Cas d'un AFD déjà minimal

- Aucune paire d'états n'est équivalente.

$\approx = \approx_{n-2}$ pour $n = |Q|$

- $\approx_0$ partitionne Q en deux classes;

puisque $\forall i \ \approx_i \neq \approx_{i+1}$

- $\approx_{i+1}$ partitionne Q avec au moins une classe de plus que $\approx_i$.
- on ne peut avoir plus de n classes (n =|Q|), donc

$\approx = \approx_{n-2}$.

21

Minimisation de $A=\langle Q,\Sigma,\delta,i,T\rangle$

- Construire partition ∏ des états en deux groupes
  - un des états terminaux
  - un des autres

Répéter

construire un nouvelle partition ∏' en séparant les états

Si ∏≠∏' alors ∏←∏' fsi

Jusqu'à ∏=∏'

- Terminer

22

Minimisation de $A=\langle Q,\Sigma,\delta,i,T\rangle$

La séparation des états est définie par :

- Pour chaque classe G de ∏ faire
  - p et q sont dans des classes d'équivalence différentes SSI $\exists a \in \Sigma$: δ(p,a) et δ(q,a) sont dans des classes différentes
  - Remplacer G par les sous-groupes ainsi formés.

23

Terminer

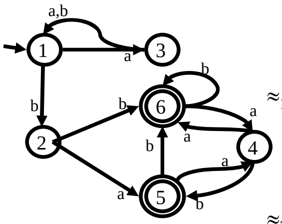
- Choisir un état [p] représentant chaque classe de ∏
- Pour chaque transition δ(p,a)=q de A, ajouter une transition de [p] vers [q] étiquetée par a.
- État initial : l'état représentant la classe de i
- États terminaux : les état représentant les classes contenant des terminaux de A.

24

Complexité

- La définition inductive fournit un algorithme en $O(n^2|\Sigma|)$ pour $n=|Q|$, qui détermine les classes d'équivalence et construit donc l'AFD minimal.
- Avec quelques améliorations, on peut construire l'AFD minimal en $O(n \log n\, |\Sigma|)$

Exemple



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

$\approx_0$

[i]				[f]		
1	2	3	4	5	6	
3	5	1	6	4	4	a
2	6	1	5	6	6	b

[i]				[f]		
1	2	3	4	5	6	
i	f	i	f	i	i	a
i	f	i	f	f	f	b

$\approx_1$

[i]		[j]		[f]		
1	3	2	4	5	6	
3	1	5	6	4	4	a
2	1	6	5	6	6	b

[i]		[j]		[f]		
1	3	2	4	5	6	
i	i	f	f	j	j	a
j	i	f	f	f	f	b

$\approx_2$

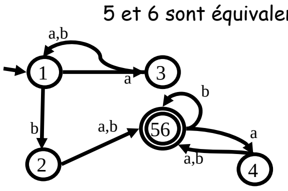
i	k	[j]		[f]		
1	3	2	4	5	6	
3	1	5	6	4	4	a
2	1	6	5	6	6	b

i	k	[j]		[f]		
1	3	2	4	5	6	
k	i	f	f	j	j	a
j	i	f	f	f	f	b

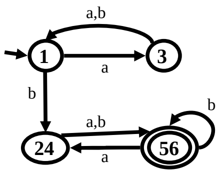
$\approx_3 = \approx_2$ **Plus rien ne peut se séparer**

Exemple

5 et 6 sont équivalents



2 et 4 sont équivalents



i	k	[j]		[f]		
1	3	2	4	5	6	
k	i	f	f	j	j	a
j	i	f	f	f	f	b