

Automates et langages: quelques algorithmes

Eugene Asarin

Saddek Bensalem

1 Avertissement

Dans l'état actuel ce document est archi-sec et peut servir seulement d'un aide-mémoire. Pour comprendre les algorithmes ci-dessous il faut suivre les cours (et/ou lire un livre sur les automates). Il faut également pratiquer ces algorithmes (au moins faire toutes les exercices des feuilles de TD sur ma page web).

2 Détermination

2.1 Problème

Etant donné un automate $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$, non-déterministe avec ϵ -transitions, construire un automate déterministe \mathcal{A}' , complet et sans *epsilon*, acceptant le même langage.

2.2 Quelques notations utiles

2.2.1 a -successeurs

Pour un état $q \in Q$ et un symbole $a \in \Sigma$ on définit

$$S(q, a) = \{p \mid (q \xrightarrow{a} p) \in \Delta\}$$

$S(q, a)$ dénote l'ensemble des états qui sont les successeurs immédiats de q , en d'autres termes tous les états que l'on pourrait atteindre par la transition étiquetée par a .

On peut étendre cette opération aux ensembles. Soient M sous-ensemble de Q , où Q est un ensemble d'états et a un élément de *Sigma*. Les successeurs immédiats de M par la transition étiquetée par a est définie par l'équation suivante :

$$S(M, a) = \bigcup_{q \in M} S(q, a)$$

En d'autres termes, $S(M, a)$ dénote l'ensemble de tous les états que l'on pourrait atteindre à partir des états dans M en effectuant une transition étiquetée par a .

2.2.2 ϵ -fermeture

La définition de l' ϵ -fermeture d'un état q , que l'on note par $E(q)$, et de son extension sont utiles pour l'élimination des ϵ -transitions.

Étant donné état $q \in Q$, on définit l' ϵ -fermeture de q par :

$$E(q) = \{q' \mid q \xrightarrow{\epsilon^*} q'\}$$

En d'autres termes $E(q)$ est l'ensemble de tous les états dans Q que l'on pourrait atteindre à partir de q en effectuant 0, 1 ou plusieurs ϵ -transitions.

L'opération E , comme pour S , pourrait être étendue aux ensembles :

$$E(M) = \bigcup_{q \in M} E(q)$$

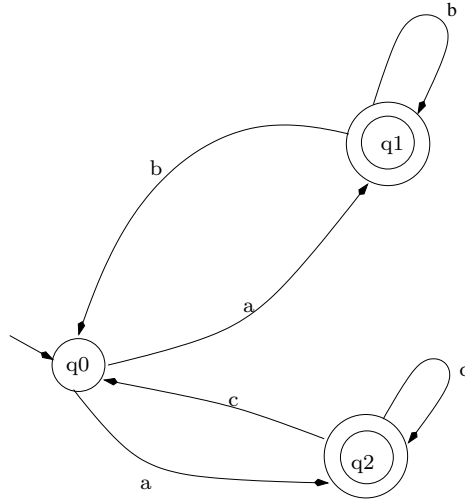


FIG. 1 – Automate A_1

Exemples : Le calcul des successeurs immédiats des états de l'automate A_1 ci-dessus :

$$S(q_0, a) = \{q_1, q_2\}, \quad S(q_0, b) = \emptyset, \quad S(q_0, c) = \emptyset$$

$$S(q_1, a) = \emptyset, \quad S(q_1, b) = \{q_0, q_1\}, \quad S(q_1, c) = \emptyset$$

$$S(q_2, a) = \emptyset, \quad S(q_2, b) = \emptyset, \quad S(q_2, c) = \{q_0, q_2\}$$

l' ϵ -fermeture de chaque état de l'automate A_2 :

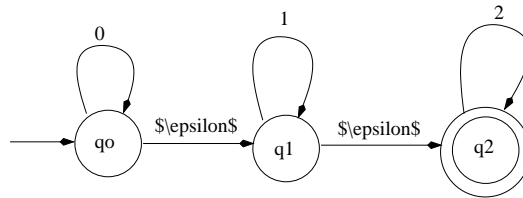


FIG. 2 – Automate A_2

$$E(q_0) = \{q_0, q_1, q_2\},$$

$$E(q_1) = \{q_1, q_2\},$$

$$E(q_2) = \{q_2\},$$

$$E(\{q_1, q_2\}) = E(q_1) \cup E(q_2) = \{q_1, q_2\} \cup \{q_2\} = \{q_1, q_2\}$$

2.3 Construction – le cas simple sans ϵ

Si l'automate \mathcal{A} que l'on veut rendre déterministe ne contient pas d' ϵ -transitions, alors l'automate déterministe équivalent \mathcal{A}' , c'est-à-dire acceptant le même langage, peut être construit de la manière suivante:

$$\mathcal{A}' = (Q', \Sigma', \Delta', q'_0, F')$$

Où

$$Q' = 2^Q$$

l'ensemble de toutes les parties de Q

$$\Sigma' = \Sigma$$

le même 'alphabet que celui de l'automate \mathcal{A}

$$\delta'(M, a) = S(M, a)$$

c'est ça l'astuce

$$q'_0 = \{q_0\}$$

est le singleton qui contient l'état initial de l'automate \mathcal{A}

$$F' = \{M \mid M \cap F \neq \emptyset\}$$

M est final s'il contient au moins un état final de \mathcal{A}

Ci-dessus M désigne un sous-ensemble quelconque de Q

2.3.1 Exemple

Soit l'automate non-déterministe ci-dessous :

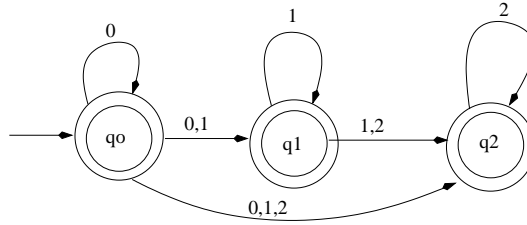


FIG. 3 – Automate A_3 non-déterministe sans ϵ -transition

Pour construire l'automate déterministe acceptant le même langage que l'automate A_3 , nous appliquons la construction ci-dessus car il s'agit d'un automate non-déterministe sans ϵ -transition. Donc, l'automate déterministe que nous obtenons est le suivant :

- L'ensemble des états est l'ensemble des parties de $Q = \{q_0, q_1, q_2\}$
- L'alphabet est le même, c'est-à-dire $\Sigma = \{0, 1, 2\}$
- l'état initial est $\{q_0\}$
- La relation de transition est définie par :

	0	1	2
$\{q_0\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$\{q_1, q_2\}$	\emptyset	$\{q_1, q_2\}$	$\{q_2\}$
$\{q_0\}$	\emptyset	\emptyset	$\{q_2\}$

- Comme $F = \{q_0, q_1, q_2\}$ alors $F' = \{\{q_0\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}\}$

2.4 Construction – le cas général

Dans le cas général, il faut prendre en considération les ϵ -transition:

$Q' = 2^Q$	comme dans le cas simple sans ϵ -transition
$\Sigma' = \Sigma$	l'alphabet n'est pas modifié
$\delta'(M, a) = E(S(M, a))$	on ferme par ϵ
$q'_0 = E(q_0)$	il y a plus d'états initiaux
$F' = \{M \mid M \cap F \neq \emptyset\}$	l'ensemble des états finals est le même que dans le cas simple sans ϵ -transition

2.4.1 Exemple

Pour illustrer la méthode de construction d'automate déterministe dans le cas général, d'automate non-déterministe avec ϵ -transition, nous considérons l'automate A_4 figure ?? ci-dessous :

- L'ensemble, Q' , des états de l'automate déterministe est l'ensemble des parties de $Q = \{0, 1, 2, 3, 4\}$, où Q est l'ensemble des états de A_4 ,
- Le vocabulaire, Σ' est le même que celui de l'automate A_4 , c'est-à-dire $\Sigma' = \Sigma = \{a, b\}$,

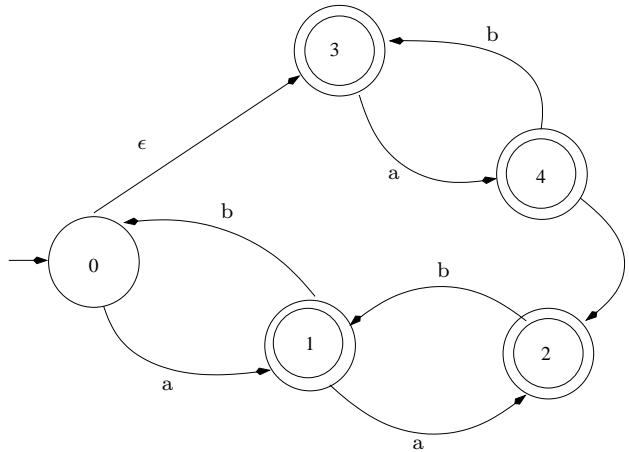


FIG. 4 – Automate A_4 non-déterministe avec ϵ -transition

- L'état initial de A_4 est définie par ϵ -fermeture de l'état 0, état initial de A_4 : $E(0) = \{0,3\}$, La relation de transition est définie par :

	a	b
$\{0,3\}$	$\{1,2,4\}$	\emptyset
$\{1,2,4\}$	$\{2\}$	$\{0,1,3\}$
$\{2\}$	\emptyset	$\{1\}$
$\{0,1,3\}$	$\{1,2,4\}$	$\{0,3\}$
$\{1\}$	$\{2\}$	$\{0,3\}$

- Comme l'ensembles des états finals de l'automate A_4 est $F = \{1,2,3,4\}$, alors l'ensemble des états finals de l'automate déterministe est $F' = \{\{1\}, \{2\}, \{0,3\}, \{0,1,3\}, \{1,2,4\}\}$

La représentation graphique de l'automate déterministe ci-dessus est :

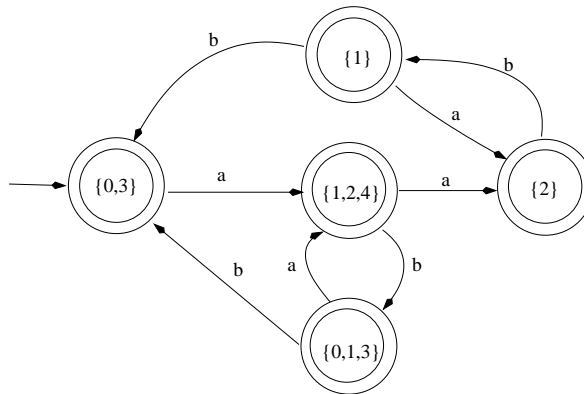


FIG. 5 – Automate déterministe obtenu à partir de A_4

3 Expressions régulières : lois algébriques

$$f + g = g + f$$

$$\begin{aligned}
(f + g) + h &= f + (g + h) \\
f + f &= f \\
f + \emptyset &= f \\
(fg)h &= f(gh) \\
f\epsilon &= \epsilon f = f \\
f\emptyset &= \emptyset f = \emptyset \\
f(g + h) &= fg + fh \\
(g + h)f &= gf + hf \\
\emptyset^* &= \epsilon^* = \epsilon \\
f^* &= ff^* + \epsilon
\end{aligned}$$

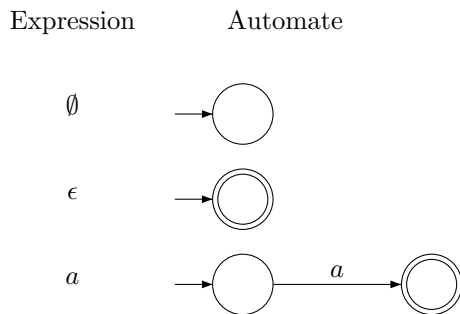
4 Expression régulière \rightarrow automate

4.1 Problème

Étant donnée une expression régulière f construire un automate \mathcal{A} qui accepte le langage dénoté par cette expression.

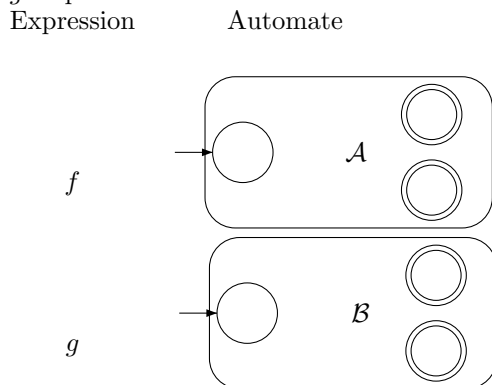
4.2 Construction

4.2.1 Cas de base



4.2.2 Cas inductif

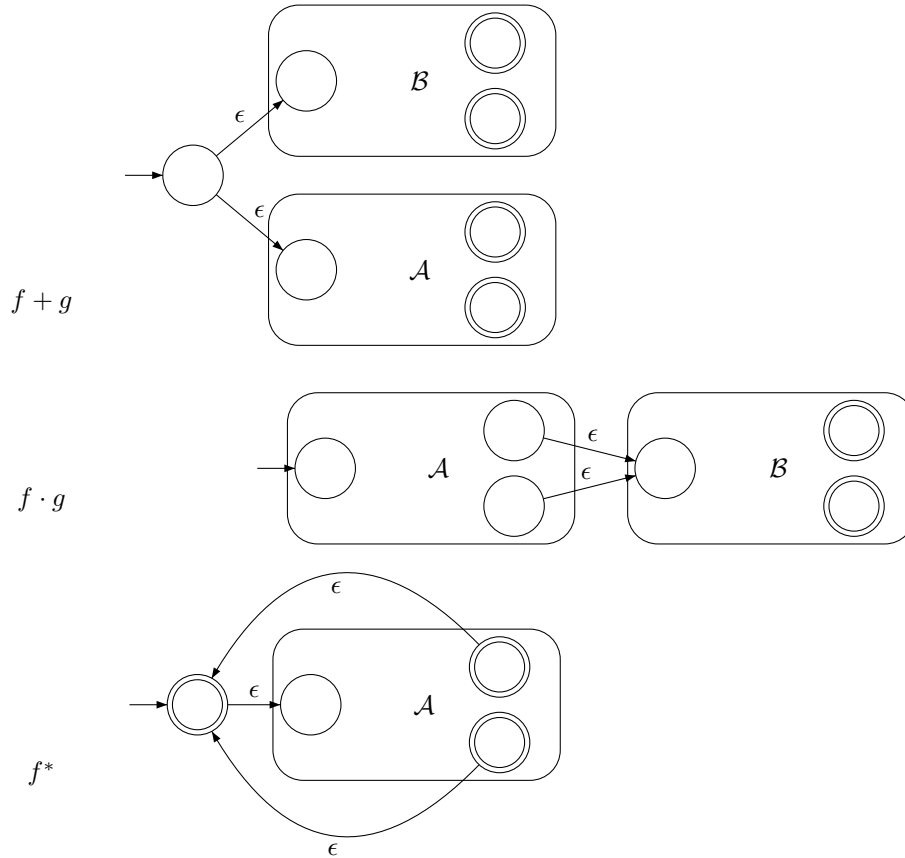
On suppose qu'on a déjà construit les 2 automates \mathcal{A} et \mathcal{B} acceptant les langages des expressions f et g respectivement



On construira à partir de \mathcal{A} et \mathcal{B} les automates pour $f + g$, $f \cdot g$, f^*

Expression

Automate



Cela termine la construction

5 Automate \rightarrow expression régulière

5.1 Problème

Étant donné un automate $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$, on veut construire une expression régulière f telle que $L(f) = L(\mathcal{A})$, c'est à dire que le langage dénoté par f est le même que le langage reconnu par \mathcal{A} .

5.2 Automate \rightarrow système d'équations sur des langages

On introduit une inconnue X_q pour chaque état $q \in Q$. Elle désignera le langage accepté à partir de l'état q . Pour chaque q on écrit une équation suivante:

– si $q \notin F$

$$X_q = \sum_{(q \xrightarrow{a} p) \in \Delta} aX_p;$$

– si $q \in F$

$$X_q = \sum_{(q \xrightarrow{a} p) \in \Delta} aX_p + \epsilon;$$

On obtient un système de N équations avec N inconnues (N est le nombre d'états de l'automate). On cherche une expression régulière pour X_{q_0} .

5.3 Résolution d'une équation

Lemme 1 (Arden) Soient L et M deux langages, X - un langage inconnu. L'équation

$$X = LX + M$$

a une solution

$$X_0 = L^*M$$

Remarque: sous certaines hypothèses cette solution est unique. Dans tous les cas elle est la seule solution pertinente pour notre problème.

5.4 Résolution d'un système

La méthode de résolution suggérée combine le lemme d'Arden pour résoudre les équations individuelles et la méthode d'élimination de variables de Gauss. En 2 mots: pour un système de N équations avec N inconnues $X_1..X_N$ on procède comme suit:

- En utilisant le lemme précédent pour la première équation on exprime X_1 en fonction de variables $X_2..X_N$.
- On substitue toutes les occurrences de X_1 dans les équations 2.. N par cette expression. X_1 est éliminée
- De la même manière on élimine X_2 en utilisant la deuxième équation, etc. jusqu'à X_N
- Maintenant notre système a une forme triangulaire.
- En remontant le système on trouve les expressions régulières pour $X_N, X_{N-1}, \dots, X_2, X_1$
- C'est tout

Pour le système obtenu à partir de l'automate (A), la solution obtenue pour X_{q_0} est l'expression régulière recherchée pour $L(A)$

6 Opérations sur les langages réguliers

On peut représenter les langages réguliers soit par des expressions, soit par des automates. Certaines opérations et certains tests peuvent être effectués aussi bien sur les expressions que sur les automates, tandis que d'autres opérations et tests peuvent se faire seulement sur les automates. Dans ce document, uniquement les algorithmes sur les automates seront présentés.

On suppose que $L = L(A)$ et $M = L(B)$.

Opérations régulières Automates pour $L \cup M$, $L \cdot M$, L^* : voir section ??

Complément Pour construire l'automate acceptant \bar{L} (le complément de L) il faut

1. Déterminiser l'automate \mathcal{A} (en n'oubliant dans aucun cas "l'état d'erreur")
2. Complémenter l'ensemble des états finals.

Intersection Théoriquement comme $L \cap M = \overline{\overline{L} \cup \overline{M}}$, la construction de l'automate pour l'intersection peut se ramener aux opérations précédentes. En pratique cette méthode donne des automates de taille énorme.

Une façon beaucoup plus raisonnable de construire l'automate acceptant $L \cap M$ est la suivante: à partir de deux automates $\mathcal{A} = (P, \Sigma, \Delta_A, p_0, F_A)$ et $\mathcal{B} = (R, \Sigma, \Delta_B, r_0, F_B)$ on construit l'automate produit $\mathcal{A} \times \mathcal{B} = (Q, \Sigma, \Delta, q_0, F)$ avec

$$\begin{aligned} Q &= P \times R && \text{produit cartésien} \\ q_0 &= (p_0, r_0) \\ F &= F_A \times F_B \end{aligned}$$

La relation de transition Δ contient trois types de transitions:

1. $(p,r) \xrightarrow{a} (p',r')$ tels que $(p \xrightarrow{a} p') \in \Delta_A$ et $(r \xrightarrow{a} r') \in \Delta_B$
2. $(p,r) \xrightarrow{\epsilon} (p',r')$ tels que $(p \xrightarrow{\epsilon} p') \in \Delta_A$ et $r = r'$
3. $(p,r) \xrightarrow{\epsilon} (p',r')$ tels que $(r \xrightarrow{\epsilon} r') \in \Delta_B$ et $p = p'$

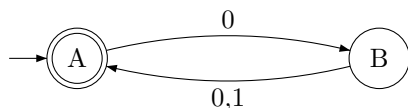
L'automate $\mathcal{A} \times \mathcal{B}$ accepte le langage $L \cap M$. Lors de la construction de l'automate produit il n'est pas nécessaire de considérer tous les états (tout le produit cartésien). On peut se restreindre à l'ensemble des états accessibles (voir l'exemple ci-dessous).

Autres opérations En TD on a vu comment effectuer les opérations "miroir", homomorphisme, "préfixe", "suffixe", "infixe" ...

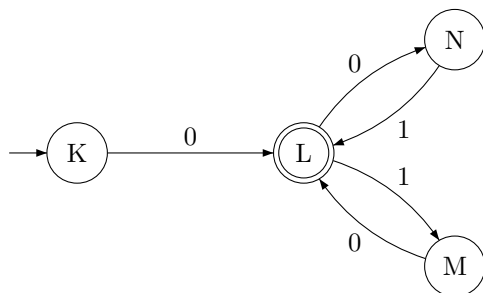
6.1 Exemple

En passant par les automates transformer l'expression régulière étendue $(00+01)^* \cap 0(10+01)^*$ en forme régulière (non étendue).

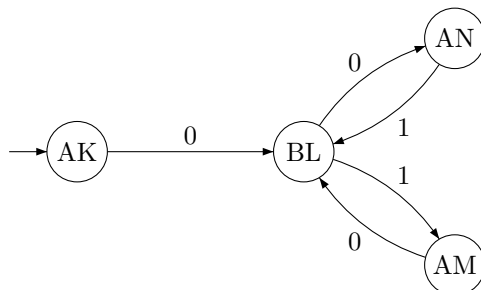
Un automate A1 pour $(00+01)^*$:



Un automate A2 pour $0(10+01)^*$:



L'automate produit de A1 et A2 accepte l'intersection $(00 + 01)^* \cap 0(10 + 01)^*$:



Comme il n'a pas d'états accepteurs, son langage est vide, d'où l'expression régulière : \emptyset

7 Tests sur les langages réguliers

Comme précédemment on suppose que $L = L(\mathcal{A})$ et $M = L(\mathcal{B})$.

Appartenance : $w \in L$ si et seulement si $Succ(q_0, w) \cap F \neq \emptyset$. Il faut donc calculer l'ensemble successeur et tester s'il contient un état final.

Langage vide : $L \neq \emptyset$ si et seulement si dans (A) il existe un chemin de l'état initial vers un état final¹.

Langage fini : L est infini si et seulement si dans (A) il existe un cycle σ avec les propriétés suivantes:

- σ est accessible à partir de l'état initial
- un état final est accessible à partir de σ
- σ contient au moins une transition étiquetée par autre chose qu' ϵ

Inclusion : Comme $L \subset M$ si et seulement si $L \cap \overline{M} = \emptyset$, pour tester l'inclusion il suffit de construire l'automate acceptant $L \cap \overline{M}$ et faire le test de langage vide pour cet automate.

Égalité : Comme $L = M$ si et seulement si $(L \subset M) \wedge (M \subset L)$, pour tester l'égalité il suffit de faire deux tests d'inclusion.

8 Minimisation

8.1 Problème

Étant donné un automate déterministe $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$, on cherche à construire un automate déterministe équivalent contenant un nombre minimum d'états.

8.2 Structure de l'algorithme

1. Supprimer tous les états de \mathcal{A} inaccessibles à partir de q_0 . Dans la suite on suppose que c'est déjà fait.
2. Construire la relation d'équivalence \approx sur l'ensemble Q (voir ??). Elle partitionne Q en plusieurs classes d'équivalence. On va écrire $[q]$ pour la classe contenant q .

1. les algorithmes pour rechercher un tel chemin sont étudiés en LP

3. Construire l'automate minimal $\mathcal{A}' = (Q', \Sigma, \delta', q'_0, F')$ en collant ensemble les états dans chaque classe d'équivalence. Formellement

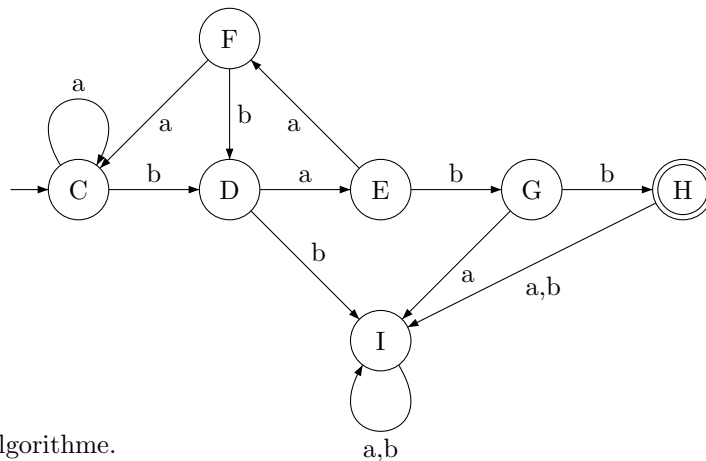
$$\begin{array}{ll}
 Q' = Q / \approx = \{[q] \mid q \in Q\} & \text{l'ensemble de tous les classes d'équivalence} \\
 \delta'([q], a) = [\delta(q, a)] & \text{le successeur d'une classe d'équivalence} \\
 & \text{est la classe d'équivalence du successeur} \\
 q'_0 = [q_0] & \text{la classe d'équivalence de l'état initial de } \mathcal{A} \\
 F' = \{[q] \mid q \in F\} & \text{les classes d'équivalence des états finals de } \mathcal{A}
 \end{array}$$

8.3 Comment construire la relation d'équivalence

1. On construit un tableau de taille $N \times N$ (N est le nombre d'états de notre automate). Chaque case dans ce tableau correspond à un couple d'états (p, q) . Pour ne pas avoir deux fois le même couple $((p, q)$ et (q, p)) ni les couples inutiles (p, p) , on garde seulement la partie triangulaire supérieure du tableau. (*Notre but final est de marquer toute les cases (p, q) pour des états p et q non-équivalents*)
2. Au début on marque chaque case (p, q) correspondant à un état final et à un état non-final (c'est à dire $p \in F$ et $q \notin F$ ou à l'envers, $p \notin F$ et $q \in F$).
3. On parcourt toutes les cases vides du tableau d'une manière quelconque en le modifiant de façon suivante: pour chaque case (p, q) et lettre a , on calcule $\delta(p, a) = p'$ et $\delta(q, a) = q'$. On marque (p, q) si et seulement si la case (p', q') est déjà marquée. On répète le balayage jusqu'au moment quand le tableau ne se modifie plus.
4. La relation d'équivalence se cache dans le tableau final: deux états p et q sont équivalents si et seulement si la case (p, q) n'est pas marquée.

8.4 Exemple

On veut minimiser l'automate



On applique l'algorithme.

	D	E	F	G	H	I
C					X	
D	-				X	
E	-	-			X	
F	-	-	-		X	
G	-	-	-	-	X	
H	-	-	-	-	-	X

Initialisation:

	D	E	F	G	H	I
C				X	X	
D	-			X	X	
E	-	-		X	X	
F	-	-	-	X	X	
G	-	-	-	-	X	X
H	-	-	-	-	-	X

Première itération:

	D	E	F	G	H	I
C		X		X	X	
D	-	X		X	X	
E	-	-	X	X	X	X
F	-	-	-	X	X	
G	-	-	-	-	X	X
H	-	-	-	-	-	X

Deuxième itération:

	D	E	F	G	H	I
C	X	X		X	X	
D	-	X	X	X	X	X
E	-	-	X	X	X	X
F	-	-	-	X	X	
G	-	-	-	-	X	X
H	-	-	-	-	-	X

Troisième itération:

	D	E	F	G	H	I
C	X	X		X	X	X
D	-	X	X	X	X	X
E	-	-	X	X	X	X
F	-	-	-	X	X	
G	-	-	-	-	X	X
H	-	-	-	-	-	X

Quatrième itération:

	D	E	F	G	H	I
C	X	X		X	X	X
D	-	X	X	X	X	X
E	-	-	X	X	X	X
F	-	-	-	X	X	X
G	-	-	-	-	X	X
H	-	-	-	-	-	X

L'itération suivante ne modifie pas le tableau. En observant les cases non-cochées on trouve la relation d'équivalence: $C \approx F$, les autres états ne sont pas équivalents. On construit l'automate minimal:

