

Calculabilité - Décidabilité (ICC)

Cours n° 3

Stef Graillat

Sorbonne Université



- **ε -transitions** : elles permettent d'étendre un AFN en autorisant un changement d'état en lisant une entrée vide (c'est-à-dire en ne lisant aucun symbole). Les ε -AFN peuvent être convertis en AFD acceptant le même langage.
- **Expression régulière** : notation algébrique décrivant exactement les mêmes langages que les automates finis. Les opérateurs sont l'**union**, la **concaténation** et la **fermeture**.
- **Équivalence entre expressions régulières et automates finis** : on peut convertir un AFN en ER. On peut aussi convertir une ER en un ε -AFN.

- ER sous UNIX :
 - . pour n'importe quel caractère
 - $[a_1a_2\cdots a_k]$ pour $a_1 + a_2 + \cdots + a_k$
 - `[:digit:]` pour les 10 chiffres
`[:alpha:]` pour les caractères de l'alphabet (aussi [A-Z])
`[:alnum:]` pour les chiffres et l'alphabet.
 - | en lieu et place du +
 - R^* pour R répété zéro ou plusieurs fois
 - $R?$ pour $\varepsilon + R$
 - R^+ pour RR^*
 - $R\{n\}$ pour une copie de R n fois.
- `lex`, `flex`
- Recherche de motifs.

Commutativité – Associativité :

- $L + M = M + L$
- $(L + M) + N = L + (M + N)$
- $(LM)N = L(MN)$

Identité – éléments neutres – annulateur :

- $\emptyset + L = L + \emptyset = L$
- $\varepsilon L = L\varepsilon = L$
- $\emptyset L = L\emptyset = \emptyset$

Distributivité :

- $L(M + N) = LM + LN$
- $(M + N)L = ML + NL$

Attention : en général $LM \neq ML$

Lois algébriques pour les expressions régulières (suite)

- $L + L = L$
- $(L^*)^* = L^*$
- $\emptyset^* = \varepsilon$
- $\varepsilon^* = \varepsilon$
- $L^+ = LL^* = L^*L$
- $L^* = \varepsilon + L^+$

Décider si une loi algébrique sur des ER est vraie

Tester la véracité de lois algébriques sur des ER.

Exemple : $(L + M)^* = (L^* M^*)^*$

Théorème 1

Soit E une ER avec les variables L_1, L_2, \dots, L_m et C l'ER formée en remplaçant chaque occurrence de L_i par a_i pour $i = 1, 2, \dots, m$. Alors tout mot $w \in L(E)$ peut s'écrire sous la forme $w = w_1 w_2 \dots w_k$, où chaque w_i est dans un des langages, par exemple L_{j_i} , et le mot $a_{j_1} a_{j_2} \dots a_{j_k}$ est dans $L(C)$

Autrement dit, on peut construire $L(E)$ en partant de mots de $L(C)$, par exemple $a_{j_1} a_{j_2} \dots a_{j_k}$, et en remplaçant chaque a_{j_i} par n'importe quel mot de L_{j_i} .

Décider si une loi algébrique sur des ER est vraie

Tester $E = F$:

- Remplacer chaque variable par un symbole. On obtient $C = D$
- Tester si $L(C) = L(D)$

Prouver que $(L + M)^* = (L^* M^*)^*$ revient à prouver que $(\mathbf{a}_1 + \mathbf{a}_2)^* = (\mathbf{a}_1^* \mathbf{a}_2^*)^*$

- **Le lemme de pompage** : un outil pour prouver qu'un langage n'est pas régulier.
- **Propriétés de fermeture des langages réguliers** : construire des langages réguliers à partir d'autres langages réguliers.
- **Propriétés de décision** : algorithme décidant par exemple si deux automates reconnaissent le même langage.
- **Minimisation des automates** : construire un automate avec le moins d'états possibles

Intuition du lemme de pompage

Supposons que $L_{01} = \{0^n 1^n : n \geq 1\}$ soit régulier. Il est donc reconnu par un AFD A avec par exemple k états.

Quand A lit 0^k on a :

ε	p_0
0	p_1
00	p_2
\dots	\dots
0^k	p_k

$\Rightarrow \exists i < j$ tel que $p_i = p_j$. Appelons $q = p_i = p_j$

- Si $\widehat{\delta}(q, 1^i) \in F$ alors A accepte $0^j 1^i$
- Si $\widehat{\delta}(q, 1^i) \notin F$ alors A rejete $0^i 1^i$

Par conséquent L_{01} n'est pas régulier

Lemme 1 (Lemme de pompage)

Soit L un langage régulier. Alors il existe $n \in \mathbb{N}$ (qui dépend de L) tel que pour tout $w \in L$ tel que $|w| \geq n$, w se décompose en $w = xyz$ avec

- $y \neq \varepsilon$,
- $|xy| \leq n$,
- pour tout $k \geq 0$, on a $xy^kz \in L$.

Preuve.

Comme L est régulier, il est reconnu par un AFD A avec n états.

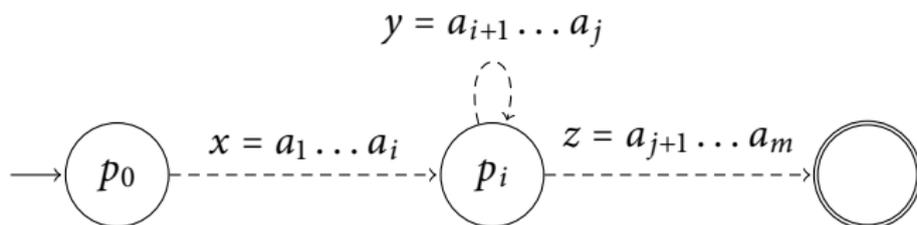
Posons $w = a_1a_2 \dots a_m \in L$ avec $m \geq n$ et $p_i = \widehat{\delta}(q_0, a_1a_2 \dots a_i)$ pour $i = 0, 1, \dots, n$ (on remarque que $p_0 = q_0$).

$\Rightarrow \exists i, j$ vérifiant $0 \leq i < j \leq n$ tel que $p_i = p_j$ (principe des tiroirs)

Lemme de pompage (suite)

On sépare $w = xyz$ comme suit :

- 1 $x = a_1 a_2 \dots a_i$
- 2 $y = a_{i+1} a_{i+2} \dots a_j$
- 3 $z = a_{j+1} a_{j+2} \dots a_m$



Il est alors clair que $xy^kz \in L$ pour tout $k \geq 0$. ■

Prouver que les langages suivants ne sont pas réguliers :

- Le langage L constitué des mots contenant autant de 0 que de 1 :
considérons $w = 0^n 1^n \in L$ pour le n correspondant à l'énoncé. D'après le lemme de pompage $w = xyz$, $|xy| \leq n$, $y \neq \varepsilon$ et $xy^kz \in L$

$$w = \underbrace{000\dots}_{x} \underbrace{\dots 0}_{y} \underbrace{0111\dots 11}_{z}$$

En particulier $xz \in L$, or xz a moins de 0 que de 1

- Le langage L des mots ne contenant que des 1 et dont la longueur est un nombre premier : considérons un premier $p \geq n + 2$ et $w = 1^p$ pour n donné par le lemme de pompage

$$w = \overbrace{111 \cdots \cdots 1 \cdots 1111 \cdots 11}^p$$

$x \quad y, |y|=m \quad z$

Alors $xy^{p-m}z \in L$.

Or $|xy^{p-m}z| = |xz| + (p-m)|y| = p - m + (p-m)m = (1+m)(p-m)$
qui n'est pas premier

En effet $y \neq \varepsilon \Rightarrow 1+m > 1$ et $m = |y| \leq |xy| \leq n$, $p \geq n+2$

$\Rightarrow p-m \geq n+2-n = 2$

Énoncés du type : si certains langages sont réguliers et un langage L est obtenu via certaines opérations sur ces langages réguliers, alors L est régulier.

- Opérations de nature ensembliste
- Concaténation, renversement, fermeture

Propriétés de fermeture (suite)

Soit L et M deux langages réguliers. Alors les langages suivants sont réguliers :

- Union : $L \cup M$
- Intersection : $L \cap M$
- Complémentaire : \overline{L}
- Différence : $L \setminus M$
- Renversement : $L^R = \{w^R : w \in L\}$
- Fermeture : L^*
- Concaténation : LM

Théorème 2

Soit L et M deux langages réguliers. Alors $L \cup M$ est régulier.

Preuve : Soit $L = L(E)$ et $M = L(F)$ avec E et F des ER. On a alors $L \cup M = L(E) \cup L(F) = L(E + F)$ par définition. ■

Théorème 3

Soit L et M deux langages réguliers. Alors LM est régulier.

Preuve : Soit $L = L(E)$ et $M = L(F)$ avec E et F des ER. On a alors $LM = L(E).L(F) = L(EF)$ par définition. ■

Théorème 4

Soit L un langage régulier. Alors L^ est régulier.*

Preuve : Soit $L = L(E)$ avec E une ER. On a alors $L^* = L(E)^* = L(E^*)$ par définition. ■

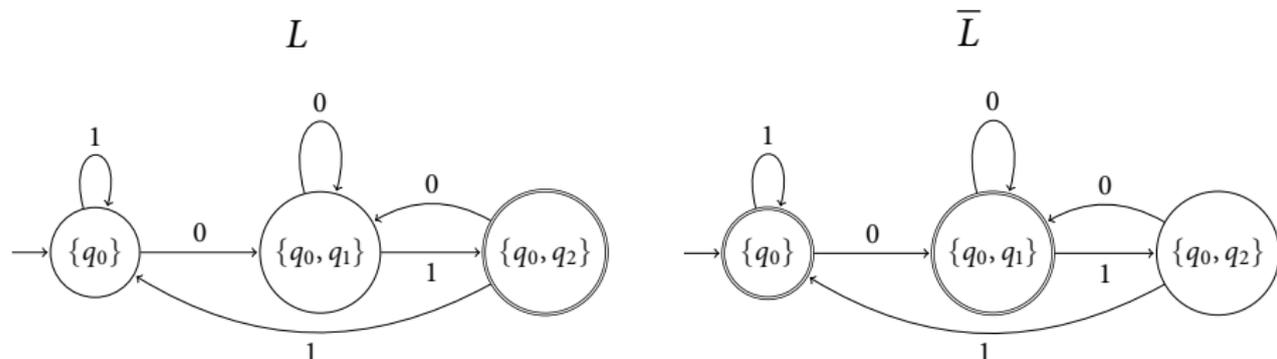
Complémentaire, intersection et différence

Théorème 5

Soit L un langage régulier. Alors le complémentaire de L dans Σ^* noté \bar{L} est régulier.

Preuve : Convertir l'expression régulière en un ε -AFN, puis convertir cet ε -AFN en un AFD $A = (Q, \Sigma, \delta, q_0, F)$. Soit alors l'automate $B = (Q, \Sigma, \delta, q_0, Q \setminus F)$. On a $\bar{L} = L(B)$. ■

Exemple : complémentaire du langage $L = (0 + 1)^* 01$



Théorème 6

Soit L et M deux langages réguliers. Alors $L \cap M$ est régulier.

Preuve : Par les lois de De Morgan, on a $L \cap M = \overline{\overline{L} \cup \overline{M}}$. Par conséquent, $L \cap M$ est régulier comme complémentaire et l'union de langages réguliers. ■

Corollaire 1

Soit L et M deux langages réguliers. Alors $L \setminus M$ est régulier.

Preuve : Comme $L \setminus M = L \cap \overline{M}$, on en déduit que $L \setminus M$ est régulier. ■

Théorème 7

Soit L et M deux langages réguliers. Alors $L \cap M$ est régulier.

Preuve : Soit $A_L = (Q_L, \Sigma, \delta_L, q_L, F_L)$ et $A_M = (Q_M, \Sigma, \delta_M, q_M, F_M)$ deux automates **déterministes** reconnaissant respectivement L et M .

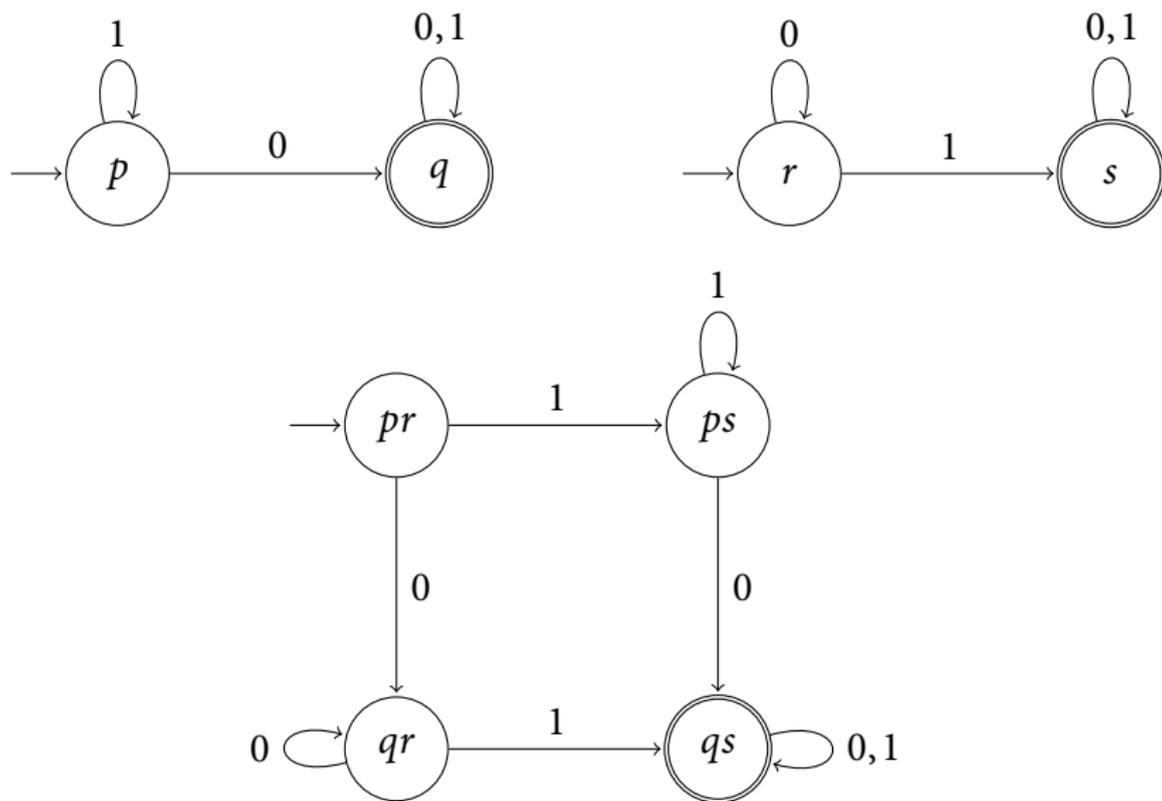
→ on va construire un automate qui simule A_L et A_M en parallèle et qui accepte si et seulement si A_L et A_M acceptent.

Posons $A_{L \cap M} = (Q_L \times Q_M, \Sigma, \delta_{L \cap M}, (q_L, q_M), F_L \times F_M)$ avec

$$\delta_{L \cap M}((p, q), a) = (\delta_L(p, a), \delta_M(q, a))$$

⇒ $A_{L \cap M}$ reconnaît le langage $L \cap M$. ■

Retour sur l'intersection (exemple)



Renversement : $a_1 a_2 \cdots a_{n-1} a_n \rightarrow a_n a_{n-1} \cdots a_2 a_1$

Pour un mot w , on note w^R son miroir

Étant donné un langage L , on note $L^R = \{w^R : w \in L\}$

Théorème 8

Soit L un langage régulier. Alors L^R est régulier.

Preuve 1 : Si L est régulier, il existe un AFD A qui le reconnaît :

- Renverser toutes les flèches dans le diagramme de transition de A
- L'état de départ de A devient seul et unique état acceptant.
- Créer un nouvel état p_0 de départ avec ε -transitions sur chaque état acceptant de A .

L'automate ainsi construit reconnaît L^R . ■

Théorème 9

Soit L un langage régulier. Alors L^R est régulier.

Preuve 2 : Si $L = L(E)$ avec E une ER, on va construire un ER E^R telle que $L(E^R) = L(E)^R$ par induction structurelle

Base : Si E est égale à ε , \emptyset ou \mathbf{a} alors $E^R = E$

Induction :

- si $E = F + G$ alors $E^R = F^R + G^R$
- si $E = F.G$ alors $E^R = G^R.F^R$
- si $E = F^*$ alors $E^R = (F^R)^*$

On vérifie que E^R ainsi construit satisfait $L(E^R) = (L(E))^R$. ■

Exemple : si $L = (\mathbf{0} + \mathbf{1})\mathbf{0}^*$ alors $L^R = \mathbf{0}^*(\mathbf{0} + \mathbf{1})$

Représentation *finie* des langages réguliers :

- AFD
- AFN, ε -AFN
- Expressions régulières

Questions :

- Le langage décrit est-il vide?
- Un mot donné appartient-il au langage décrit?
- Deux descriptions différentes définissent-elles le même langage?

Conversion entre représentations

On en a déjà vu quelques-unes... mais combien coûtent-elles?

Des AFN aux AFD :

On part d'un ε -AFN à n états :

- Calcul des ε -fermetures : $\mathcal{O}(n^3)$
- Construction des sous-ensembles : il y en a 2^n mais il faut aussi obtenir le diagramme de transition ! Chaque transition coûte $\mathcal{O}(n^2)$.
- Cout global : $\mathcal{O}(n^3 2^n)$

Des AFD aux AFN :

C'est le plus simple et ça ne coûte que $\mathcal{O}(n)$ opérations.

Conversion entre représentations (suite)

Des automates aux expressions régulières :

À chaque étape, la longueur de l'expression peut quadrupler et il y a n étapes!

Écrire les n^3 expressions coûte $\mathcal{O}(n^3 4^n)$.

L'élimination d'états ne change pas la complexité (juste la constante).

Si l'entrée est un ε -AFN à n états, le coût de cette conversion est $\mathcal{O}(n^3 4^{2n})$!

Des expressions régulières aux automates :

Conversion vers les ε -AFN linéaire en la longueur de l'expression.

Pour passer à un AFN : $\mathcal{O}(n^3)$

Tester si un langage régulier est vide

Représentation : un automate fini.

Reformulation de la question : existe-t-il un moyen de joindre un état acceptant à partir de l'état de départ (parcours de graphe)?

Par induction :

Base : L'état de départ est atteignable par l'état de départ.

Induction : Si l'état q est atteignable par l'état de départ, tout état atteignable depuis q est atteignable par l'état de départ.

Coût total : $\mathcal{O}(n^2)$.

Tester si un langage régulier est vide (suite)

On peut aussi étudier une ER E et dire si $L(E) = \emptyset$.

On procède de manière récursive :

- $E = F + G$. Alors $L(E)$ est vide ssi $L(F)$ **et** $L(G)$ sont vides
- $E = F.G$. Alors $L(E)$ est vide ssi $L(F)$ **ou** $L(G)$ sont vides
- $E = F^*$. Alors $L(E)$ n'est jamais vide puisque $\varepsilon \in L(E)$
- $E = \varepsilon$. Alors $L(E)$ est non vide
- $E = \mathbf{a}$. Alors $L(E)$ est non vide
- $E = \emptyset$. Alors $L(E)$ est vide

Tester l'appartenance à un langage régulier

- Pour tester si $w \in L(A)$ avec A un AFD, on en simule le fonctionnement. Si $|w| = n$ alors cela coûte $\mathcal{O}(n)$.
- Si A est un AFN à s états, simuler A sur w coûte $\mathcal{O}(ns^2)$.
- Si A est un ε -AFN à s états (il faut alors calculer les ε -fermetures), simuler A sur w coûte $\mathcal{O}(ns^3)$.
- Si $L = L(E)$ pour une ER E de longueur s , on convertit E en un ε -AFN avec $2s$ états. Simuler w sur cette machine coûte $\mathcal{O}(ns^3)$.

Tester si deux langages réguliers définissent le même langage.

Conséquence : Minimisation d'un AFD en un AFD unique.

Définition 1

Soit $A = (Q, \Sigma, \delta, q_0, F)$ un AFD et $p, q \in Q$. On dit que p et q sont *équivalents* et on note $p \equiv q$ si pour $w \in \Sigma^*$,

$\widehat{\delta}(p, w)$ est un état acceptant ssi $\widehat{\delta}(q, w)$ est un état acceptant.

Si $p \not\equiv q$, on dit que p et q sont *distinguable*s.

Autrement dit, $p \not\equiv q$ ssi il existe $w \in \Sigma^*$ tel que

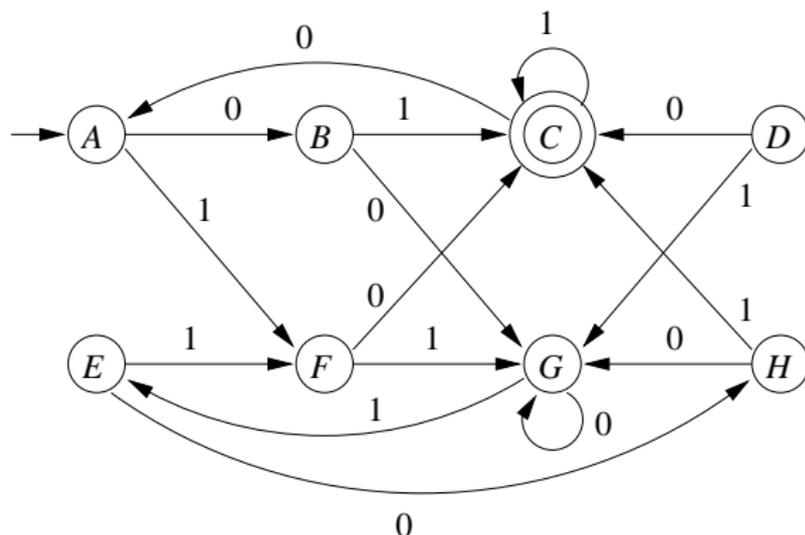
- $\widehat{\delta}(p, w) \in F$ et $\widehat{\delta}(q, w) \notin F$

ou

- $\widehat{\delta}(p, w) \notin F$ et $\widehat{\delta}(q, w) \in F$

Tester l'équivalence de deux états d'un AFD

Exemple :

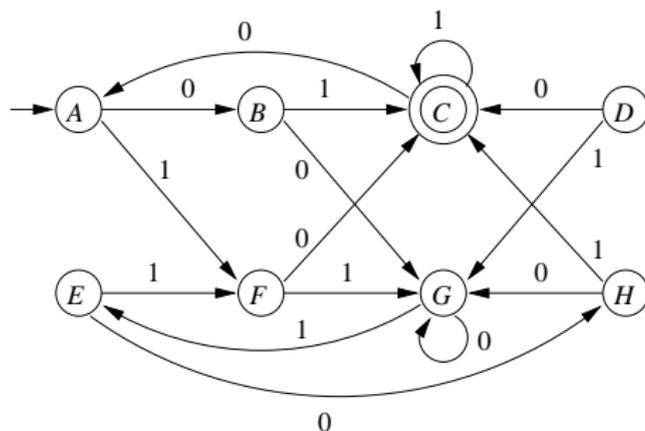


$$\widehat{\delta}(C, \varepsilon) \in F, \widehat{\delta}(G, \varepsilon) \notin F \Rightarrow C \neq G$$

$$\widehat{\delta}(A, 01) = C \in F, \widehat{\delta}(G, 01) = E \notin F \Rightarrow A \neq G$$

Tester l'équivalence de deux états d'un AFD (suite)

Que dire de A et E ?



$\widehat{\delta}(A, \varepsilon) = A \notin F$, $\widehat{\delta}(E, \varepsilon) = E \notin F$ et $\widehat{\delta}(A, 1) = F = \widehat{\delta}(E, 1)$

Ainsi $\widehat{\delta}(A, 1x) = \widehat{\delta}(E, 1x) = \widehat{\delta}(F, x)$ pour tout $x \in \{0, 1\}^*$

De plus $\widehat{\delta}(A, 00) = G = \widehat{\delta}(E, 00)$ et $\widehat{\delta}(A, 01) = C = \widehat{\delta}(E, 01)$

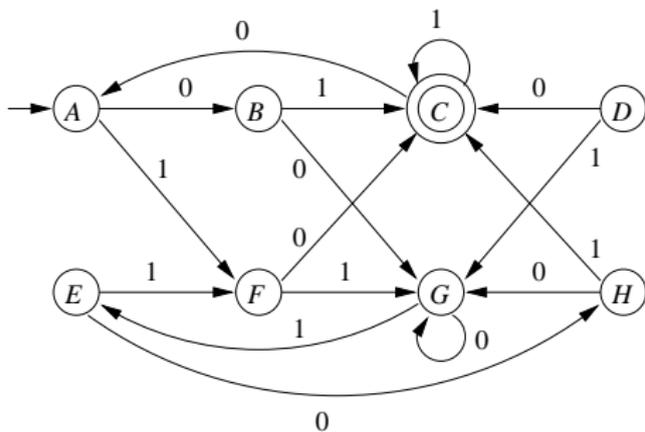
Par conséquent : $A \equiv E$

Algorithme de remplissage de table

Algorithme récursif :

Base : si $p \in F$ et $q \notin F$ alors $p \neq q$

Induction : si pour un $a \in \Sigma$, $\delta(p, a) \neq \delta(q, a)$ alors $p \neq q$



<i>B</i>	<i>x</i>						
<i>C</i>	<i>x</i>	<i>x</i>					
<i>D</i>	<i>x</i>	<i>x</i>	<i>x</i>				
<i>E</i>		<i>x</i>	<i>x</i>	<i>x</i>			
<i>F</i>	<i>x</i>	<i>x</i>	<i>x</i>		<i>x</i>		
<i>G</i>	<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>	
<i>H</i>	<i>x</i>		<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>
	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	<i>F</i>	<i>G</i>

Tester l'équivalence de langages réguliers

Les langages L et M deux langages réguliers

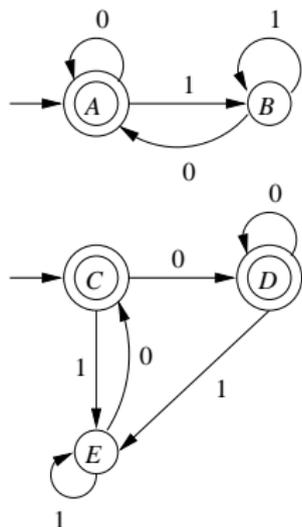
Pour tester si $L = M$:

- 1 Convertir L et M en AFD A_1 et A_2
- 2 Imaginer l'AFD qui est l'union des deux AFD
- 3 Si les deux états de départ de A_1 et A_2 sont équivalents, alors $L = M$
autrement $L \neq M$

Complexité : $\mathcal{O}(n^4)$

On peut la faire chuter à $\mathcal{O}(n^2)$.

Tester l'équivalence de langages réguliers (exemple)



<i>B</i>	<i>x</i>			
<i>C</i>		<i>x</i>		
<i>D</i>		<i>x</i>		
<i>E</i>	<i>x</i>		<i>x</i>	<i>x</i>
	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>

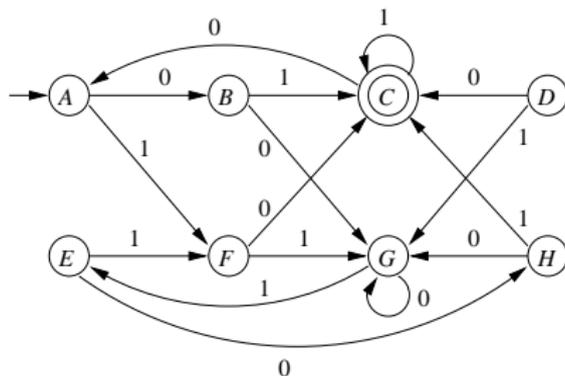
On voit bien que les 2 automates acceptent le langage $\varepsilon + (\mathbf{0} + \mathbf{1})^* \mathbf{0}$

Minimisation *unique* d'un AFD.

- Éliminer tout état ne pouvant pas être atteint par l'état de départ.
- Partitionner les états restants en blocs tels que tous les états se trouvant dans un même bloc sont équivalents.
- Les blocs deviennent des états.
- L'état de départ est le bloc contenant l'état de départ
- Les blocs acceptants sont ceux qui contiennent des états acceptants
- Si S est un bloc d'états équivalents et $a \in \Sigma$ alors $\delta(S, a)$ (réunion) est un bloc d'états équivalents.

Minimisation d'AFD (exemple)

Minimiser l'automate



<i>B</i>	<i>x</i>						
<i>C</i>	<i>x</i>	<i>x</i>					
<i>D</i>	<i>x</i>	<i>x</i>	<i>x</i>				
<i>E</i>		<i>x</i>	<i>x</i>	<i>x</i>			
<i>F</i>	<i>x</i>	<i>x</i>	<i>x</i>		<i>x</i>		
<i>G</i>	<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>	
<i>H</i>	<i>x</i>		<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>	<i>x</i>
	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	<i>F</i>	<i>G</i>

