

# Graphes et automates

## I / Automates Finis

Cas particulier des systèmes de transition avec nombre fini

états  
étiquettes  
transition

### Définition (Automates Finis déterministes) :

Quintuplet:  $Q \times A \times i \times T \times S$  ( $Q, A, i, T, S$ )

$Q$ : ensemble fini d'états  $\{q_1, \dots, q_n\}$

$A$ : ensemble fini de symboles de l'alphabet  $\{s_1, \dots, s_p\}$

$i \in Q$ : état initial

$T \subseteq Q$ : états terminaux

$\delta: Q \times A \rightarrow Q$

fonction de transition

$q': \delta(q, s)$ ,  $q, q' \in Q$ ,  $s \in A$

Exemple:  $Q = \{A, E\}$   $i = E$   $T = Q$

$A = \{I\}$

$\delta(E, I) = A$

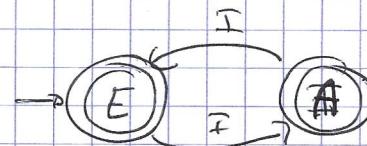
$\delta(A, I) = E$

$\delta = \{(E, I) \mapsto A, (A, I) \mapsto E\}$

Représentation graphique.

état: nom entouré cercle

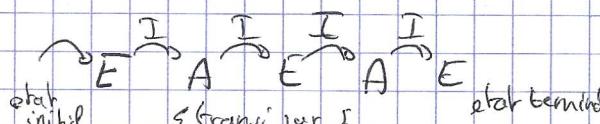
état initial:  $\rightarrow$



état final: double cercle

transition: flèche numérotée avec symbole alphabet.

Intuition: exécution d'un automate dépend : état initial. Tant qu'il y a un symbole de l'alphabet, faire une transition jusqu'à atteindre état terminal



l'automate accepte le mot IIII  
langage accepté  $\{I^n | n \in \mathbb{N}\} - I^*$

### Définition (Automate Fini indéterministe)

$Q$ : ens. fini d'états

$A$ : ens fini symboles  $\alpha/\beta$

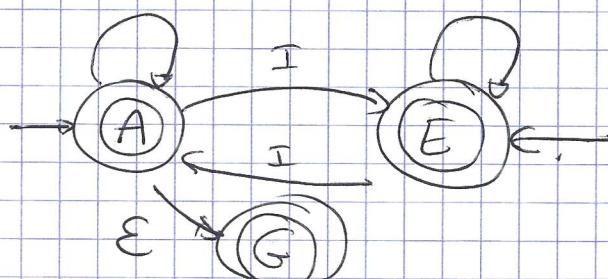
$T \subseteq Q$ : terminaux.

$I \subseteq Q$ : initiaux

$\epsilon$ : transition arbitraire/inconnue

$\delta: Q \times (A \cup \{\epsilon\}) \rightarrow \wp(Q)$

Ex:



$\delta(E, I) = \{A, E\}$

$\delta(A, \epsilon) = \{A, E\}$

3 causes indéterministes :

- $\text{card } (\Sigma) > 1$  plusieurs états initiaux.
- $\Sigma$  transition arbitraire sur  $\Sigma$ .
- $\text{card } (\delta(q, \Sigma)) > 1$  plusieurs états cibles de transition.

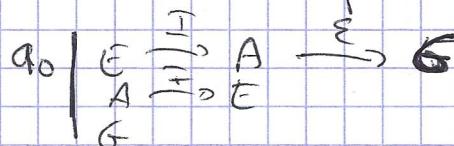
**Théorème:** tout automate fini indéterministe peut être transformé en un automate qui accepte le même langage.

Preuve : algorithme de déterminisation

**Principe:** états de l'automate déterministe équivalents sont des ensembles d'états de l'automate indéterministe.

Ex:  $q_0 = I \cup G^y$   $\rightarrow \Sigma\text{-fermeture }(I)$ ,  
car lié par  $\Sigma$  avec  $A$

Etude des transitions possibles depuis  $q_0$ .



Automate fini

$$\begin{aligned} Q &\text{ états} \\ A &\text{ alphabet} \\ I \subseteq Q &: \text{états initiaux} \\ T \subseteq Q &: \text{terminaux.} \\ \delta: Q \times (A \cup \{\varepsilon\}) &\rightarrow P(Q) \text{ d: } Q \times A \rightarrow Q \\ &\text{f° transition} \end{aligned}$$

Automate fini déterministe

$$\begin{aligned} \textcircled{1} \quad \text{card } (I) &= 1 \\ \textcircled{2} \quad \forall q \in Q, \delta(q, \varepsilon) &= \emptyset \\ \textcircled{3} \quad \forall q \in Q, \forall a \in A, \text{ card } (\delta(q, a)) &\leq 1 \end{aligned}$$

Langage accepté par un automate déterministe :

$$\underline{L(Q, A, i, T, S)} \subseteq A^* \quad \delta^*: Q \times A^* \rightarrow Q \text{ extension de } \delta \text{ à des mots.}$$

enr. de mots sur  $\alpha \beta \in A$

$$\forall q \in Q, \delta^*(q, \varepsilon) = q \text{ (réflexive).}$$

$$\forall q \in Q, \forall a \in A, \forall m \in A^*, \delta^*(q, a.m) = \delta^*(\delta(q, a), m) \text{ transitif}$$

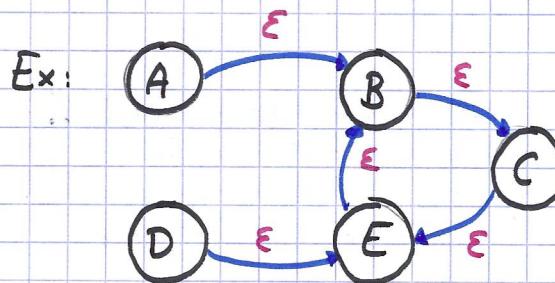
$\delta^*$ : fermeture réflexive et transitive de  $\delta$ .

$$L(Q, A, i, T, S) = \{m \in A^* \mid \delta^*(i, m) \in T\}$$

définir  $T$  état terminal  
lire  $m$

lecture de  $m$

## Definition : $\epsilon$ -fermeture



$$\epsilon\text{-fermeture}(\{A, D\}) = \underline{\{A, B, C, D, E\}}$$

ens des états accessibles depuis  $\{A, D\}$  (compris) sur  $\epsilon$ .

états accessibles depuis  $E$  sur  $\epsilon$

$$\epsilon\text{-fermeture}(E) = \boxed{E} \cup \underbrace{\epsilon\text{-fermeture}}_{\text{réflexif}} \left( \bigcup_{q \in E} \delta(q, \epsilon) \right) \underbrace{\epsilon\text{-fermeture}}_{\text{transitif}} \left( \bigcup_{q \in E} \delta(q, \epsilon) \right) \text{ définition récursive, terminaison assurée car } Q \text{ fini.}$$

$$\epsilon\text{-fermeture}(E) \quad E_0 = E$$

$$E_{i+1} = E_i \cup \bigcup_{q \in E_i} \delta(q, \epsilon) \quad \forall q \quad E_i \subseteq E_{i+1}.$$

critère d'arrêt :  $E_i = E_{i+1}$

$$\text{Ex: } E_0 = \{A, D\}$$

$$E_1 = E_0 \cup \bigcup_{q \in E_0} \delta(q, \epsilon) = \{A, D\} \cup \delta(A, \epsilon) \cup \delta(D, \epsilon)$$

$$E_1 = \{A, B, D, E\}$$

$$E_2 = E_1 \cup \bigcup_{q \in E_1} \delta(q, \epsilon) = \{A, B, D, E\} \cup \delta(A, \epsilon) \cup \delta(B, \epsilon) \cup \delta(D, \epsilon) \cup \delta(E, \epsilon).$$

$$E_2 = \{A, B, C, D, E\}.$$

$$E_3 = E_2 \text{ arrêt}$$

$$\epsilon\text{-fermeture}(E) = E_n \quad \text{avec } n = \min \{ i \mid E_i = E_{i+1} \}.$$

Definition:  $\Delta : \underbrace{\mathcal{P}(Q)}_{\text{ens. état}} \times A \rightarrow \underbrace{\mathcal{P}(Q)}_{\text{ens. état}}$

$$\Delta(E, a) = \epsilon\text{-fermeture} \left( \bigcup_{q \in E} \delta(q, a) \right)$$

une étape de  $\delta$  sur chaque état de  $E$

$\epsilon$ -fermeture

$\Delta$ : extension de  $\delta$  pour intégrer les transitions sur  $\epsilon$

Definition: langage accepté par un automate

$$L(Q, A, I, T, \delta) = \{ m \in A^* \mid \begin{array}{l} \textcircled{1} \quad (\epsilon\text{-fermeture}(I), m) \in T \\ \textcircled{2} \quad \text{éxistante} \\ \text{et les états finis} \\ \text{au moins 1 état permis} \end{array} \}$$

Theoreme: pour tout automate fini  $(Q, A, I, T, \delta)$ , il existe un automate fini déterministe  $(Q', A, I', T, \delta')$  qui accepte le même langage.

Preuve: par construction :  $I' = \underline{\epsilon\text{-fermeture}(I)}$

$\textcircled{2} \quad Q'_0 = \{I'\}$  nouvel ensemble d'état initialisé avec nouvel état initial

$$Q'_m = Q'_0 \cup \{ \Delta(q', a) \mid q' \in Q'_0, a \in A \} \quad q' \in Q'_m$$

Cas d'arrêt: Ainsi :  $Q_i \cap Q' \neq \emptyset$  et  $Q' \subseteq F(Q)$  les états de  $Q$  sont des états de  $Q'$ .  
 $n = \min \{ i \in \mathbb{N} \mid Q_{i+1} = Q'_i \}$ .

$$Q' = Q_n$$

$$T' = \{ q' \in Q' \mid q' \cap T \neq \emptyset \}$$

$$\delta' = \Delta$$

Les états terminaux du nouvel automate déterministe contiennent au moins 1 état terminal de l'automate indéterminé.

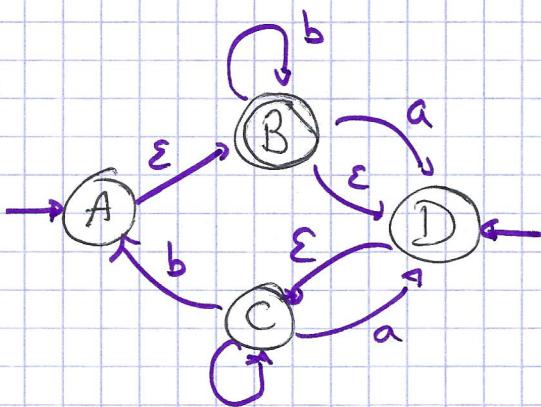
→  $\mathcal{D}'$  est déterministe.

1 seul état initial dans  $Q'$   
pas de transition  $\epsilon$

$$\text{card}(\delta'(q', a)) \leq 1 \quad \forall q' \in Q' \quad \text{car par const de } Q' \text{ le nouv automate est de taille expérimentale } \leq n \text{ mais prototypique c'est vraiment } \leq n \text{ car l'automate de départ est fini car } \text{card}(A) \leq 2.$$

par const de  $Q'$  le nouv automate est de taille expérimentale  $\leq n$  mais prototypique c'est vraiment  $\leq n$  car l'automate de départ est fini car l'automate de départ est faiblement indéterminé.

Exemple:



$$Q = \{A, B, C, D\}$$

$$I = \{A, D\}$$

$$\begin{aligned}\delta(A, \epsilon) &= \{B\} \\ \delta(B, a) &= \{D\} \\ \delta(C, \epsilon) &= \{C\} \\ \delta(D, b) &= \{A\},\end{aligned}$$

$$\begin{aligned}\delta(B, \epsilon) &= \{D\}, \\ \delta(B, b) &= \{B\}, \\ \delta(C, a) &= \{C\}, \\ \delta(C, b) &= \{A\}.\end{aligned}$$

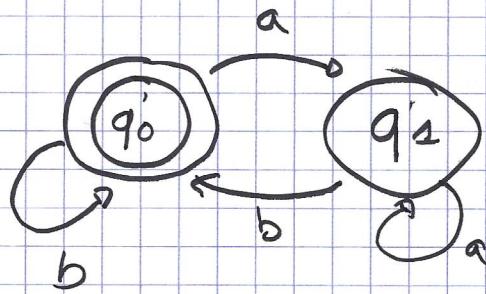
$$\begin{aligned}i' = q_0' &= \text{EF}(I) = \text{EF}(\{A, D\}) = \{A, D\} \cup \text{EF}(\delta(A, \epsilon) \cup \delta(D, \epsilon)) \\ &= \{A, D\} \cup \text{EF}(\{B, C\}), \\ &= \{A, D\} \cup \{B, C\} \cup \text{EF}(\delta(B, \epsilon) \cup \delta(C, \epsilon)), \\ &= \{A, D\} \cup \{B, C\} \cup \text{EF}(\{B, C\}), \\ &= \{A, B, C, D\}.\end{aligned}$$

$$\begin{aligned}\Delta(q_0', a) &= \text{EF}(\delta(A, a) \cup \delta(B, a) \cup \delta(C, a) \cup \delta(D, a)) \\ &= \text{EF}(\{D, C\}) = \{C, D\} = q_1'\end{aligned}$$

$$\begin{aligned}\Delta(q_0', b) &= \text{EF}(\delta(A, b) \cup \delta(B, b) \cup \delta(C, b) \cup \delta(D, b)) \\ &= \text{EF}(\{\epsilon B, A\}) = \{A, B, C, D\} = q_0'\end{aligned}$$

$$\Delta(q_1', a) = \text{EF}(\delta(C, a) \cup \delta(D, a)) = \text{EF}(\{C, D\}) = \{C, D\} = q_1'$$

$$\Delta(q_1', b) = \text{EF}(\delta(C, b) \cup \delta(D, b)) = \text{EF}(\{A\}) = q_0'$$


 $(b \mid aa = b)^*$ 

Q : quelle est la relation entre automate, expression régulière

## II / Relations entre formalismes de description de langages et automates finis.

**Théorème :** Les langages acceptés par les automates, décrits par les expressions régulières, décrits par les grammaires régulières sont de la même catégories : langage réguliers ou rationnels

Preuve par construction :

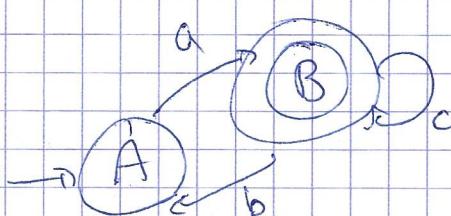
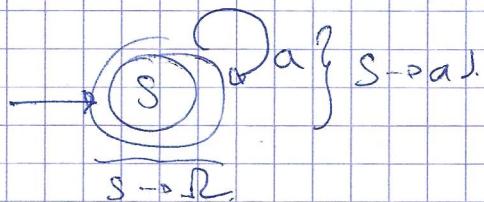
Grammaires régulières : A alphabet terminal      V vocabulaire non-terminal      SEV axiome  
 $P$  règles de production de la forme       $X \rightarrow 1 \quad X \in V$ .  
 $\quad \quad \quad Y \rightarrow aZ \quad X, Z \in V \quad a \in A$ .

L'automate  $Q - V \quad i = S$ .  
 $T = \{x \in V \mid x \rightarrow 1 \in P\}$

$\delta(Xa) = Z$  si  $X \rightarrow aZ \in P$ . accepte le langage  
 décrit par la grammaire  
 axiome :  $X \quad V = (X, X)$   
 $P = \{X \rightarrow 1, X \rightarrow aY, X \rightarrow bX,$   
 $Y \rightarrow aY, Y \rightarrow bX\}$

$$L(A, V, \delta, P) = \{m \in \Sigma^* \mid \delta = \delta^m\}$$

Ex:  $S \rightarrow aS$   
 $S \rightarrow 1$

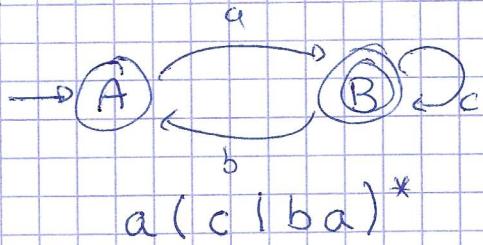


axiome A.  
 $B \rightarrow 1$

A  $\rightarrow aB$ ,  
 $B \rightarrow bA$ ,  
 $B \rightarrow cC$ .

La grammaire décrit le langage accepté par l'automate

Automates  $\rightarrow$  expression régulière : J'associe à chaque état de l'automate une variable de nature expression régulière qui représente le langage qui va de cet état vers les états terminaux en suivant les transitions.



$$a(c \mid ba)^*$$

Exemple: 2 variables A et B.

$$\left| \begin{array}{l} A = aB \\ B = \underbrace{cB \mid bA}_{\text{B est terminal}} \end{array} \right.$$

équation linéaire car structure algébrique des langages (monoides)

Q: Peut-on résoudre ce système ?

$$x = ax + by \quad B = cB \mid (R \mid bA) \rightarrow \text{Par Arden.}$$

$$(1-a)x = by \quad \text{si } a \neq 1 \quad B = c^* (R \mid bA).$$

$$x = \left( \frac{b}{1-a} \right) y.$$

Lemme de Arden: (éq X:  $e_1 X e_2$  sur des expr rég. admet  $e_1^* e_2$  comme s°.

$$\text{Preuve: remplaçons } X \text{ par } e_1^* e_2: \quad e_1^* e_2 = e_1 e_1^* e_2 = \cancel{(e_1 e_1^* \cancel{n})} e_2. \quad \text{CQFD.}$$

$$\begin{aligned} &= e_1^* e_2 \\ &= e_1^* e_2. \end{aligned}$$

Substitution de A par aB.

$$B = c^* (R \mid baB) = c^* \mid c^* baB.$$

$$\text{Par Arden} \quad B = (c^* ba)^* c^* \quad A = aB = \underline{a(c^* ba)^* c^*}$$

Langage accepté par automate car A état initial.

Cherchons une stratégie plus efficace.

$$B = R \mid bA \mid cB \quad \text{substitution}$$

$$A = aB \quad \left. \begin{array}{l} \\ B = R \mid bA \mid cB = R \mid (ba)c \mid B. \end{array} \right.$$

$$\text{par Arden} \quad B = (ba)c^* R = (ba)c^*$$

$$\text{substitution: } A = a(ba)c^*$$

Thm: Automate  $(Q, A, i, T, \delta)$

Var. associées :  $Q$  Syst lin associé:  $q_i = \{a_i\} q_i$  avec  $q_i = \delta(q_i, a_i)$

$$q = a_1 q_1 | a_2 q_2 | \dots | a_n q_n | \Sigma$$

si  $q \in T$

est tel que l'expression régulière associée à  $i$  décrire le langage accepté par automate.

Preuve: par co-induction sur structure de l'automate.

Expressions régulières vers automate:

Suivons la même correspondance: expr reg  $\rightarrow$  syst d'éq dont elle est s.o.

$$i = a \underbrace{(ba|c)^*}_{q_1} = aq_1$$

$\hookrightarrow$  automate

$$q_1 = (ba|c)^* = (ba|c) \underbrace{(ba|c)^*}_{q_2} \# \Sigma = \overline{baq_1 | cq_1 | \Sigma}$$

~~q2~~

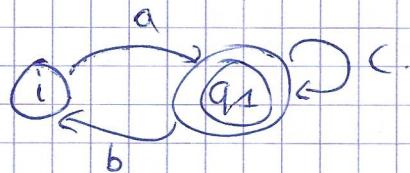
$$= \overline{b \cancel{q_2} | cq_1 | \Sigma}$$

$$i = aq_1 \quad q_1 = b\cancel{q_2} | cq_1 | \Sigma$$

~~q2 = aq1~~

$$q_2 = \overline{b \cancel{i} | cq_1 | \Sigma}$$

~~q1~~  $\in T$



Thm: Dérivée d'une expression régulière par un symbole de l'alphabet.

$$e' = Da(e) \text{ telle que } L(e) = \{w \in \Sigma^* \mid w \in L(e')\}.$$

$e'$  représente le langage des suffixes de  $a$  dans  $e$ .

$$e = a_1 Da_1(e) | a_2 Da_2(e) | \dots | a_n Da_n(e) | \Sigma \quad \text{pour } a_i \in A.$$

si  $\Sigma \subseteq L(e)$

Définition de  $Da$  par induction sur structure expr reg.

$$Da(\emptyset) = \emptyset$$

$$Da(b) = \emptyset. \quad a \neq b.$$

$$Da(\Sigma) = \Sigma$$

$$Da(e_1 | e_2) = Da(e_1) | Da(e_2).$$

$$Da(e_1 e_2) = Da(e_1) e_2 \text{ si } \Sigma \not\subseteq L(e_1)$$

$$Da(e_1 e_2) = Da(e_1) e_2 | Da(e_2) \text{ si } \Sigma \subseteq L(e_1)$$

$$e_1 = e_1' | \Sigma \text{ avec } \Sigma \not\subseteq L(e_1') \quad e_1 e_2 = e_1' e_2 | e_2.$$

Notation  $\delta(e) = \Sigma \text{ si } \Sigma \subseteq L(e)$     } discriminant  $\delta$ .

$$\delta(e) = \emptyset \text{ si } \Sigma \not\subseteq L(e)$$

$$Da(e_1 e_2) = Da(e_1) e_2 | \delta(e_1) Da(e_2)$$

$$Da(e^*) = Da(e) e^* \text{ car } e^* = ee^* | \Sigma$$

$$Da(e^*) = Da(ee^* | \Sigma) = Da(ee^*) | \emptyset = Da(e)e^* | \delta(e)\cancel{Da(e)}$$

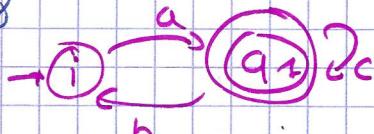
non obtenu

$$Ex: i = a \quad (c|ba)^* \quad A = \{a, b, c\} \quad \delta(a) = \emptyset$$

$$\begin{aligned} D_a(i) &= D_a(a(c|ba)^*) = D_a(a)(c|ba)^* \mid \underbrace{\delta(a)}_{\emptyset} D_a((c|ba)^*) \\ &= \emptyset \quad (c|ba)^* \mid \emptyset \dots = (c|ba)^* = q_1. \end{aligned}$$

car  $\delta(A)$

$$D_b(i) = \underbrace{D_b(a)}_{\emptyset} (c|ba)^* \mid \underbrace{\delta(a)}_{\emptyset} D_b((c|ba)^*) = \emptyset$$

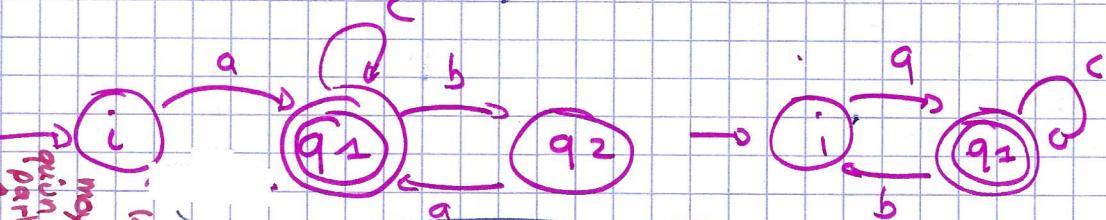


$$D_c(i) = \underbrace{D_c(a)}_{\emptyset} (c|ba)^* \mid \underbrace{\delta(a)}_{\emptyset} D_c((c|ba)^*) = \emptyset$$

$$D_a(q_1) = D_a((c|ba)^*) = \underbrace{D_a(c)}_{q_1} (c|ba)^* = (\underbrace{D_a(c)}_{\emptyset} \mid \underbrace{D_a(ba)}_{\emptyset}) q_1 = \emptyset.$$

$$D_b(q_1) = D_b((c|ba)^*) = (\underbrace{D_b(c)}_{\emptyset} \mid \underbrace{D_b(ba)}_{\emptyset}) q_1 = a q_1 = i.$$

$$D_c(q_1) = (\underbrace{D_c(c)}_{\emptyset} \mid \underbrace{D_c(ba)}_{\emptyset}) q_1 = q_2.$$



acceptent le même langage décrit par l'expr rég.

### Thm: l'intégration des automates

Pour tout automate fini, il existe un automate fini minimal (en nombre d'état) qui accepte le même langage au renommage près des états.

Preuve:  $q \cong q' \Leftrightarrow \{m \mid \delta^*(a, m) \in T\} = \{m \mid \delta^*(q, m) \in T\}$ .

Automate minimal : classes d'éq. de cette relation.

Algo: Point de départ : 2 états  $T, Q \setminus T$

on encl. d'états inclus  
soit de  $T$  ou  $Q \setminus T$

$a \in A \quad \Delta(T, a) \quad \left. \begin{array}{l} \Delta(Q \setminus T, a) \end{array} \right\} \text{ si ces transitions produisent } T \text{ ou } Q \setminus T$   
 $\Delta(Q \setminus T, a) \quad \left. \begin{array}{l} \text{arrêt de la procédure.} \end{array} \right\}$

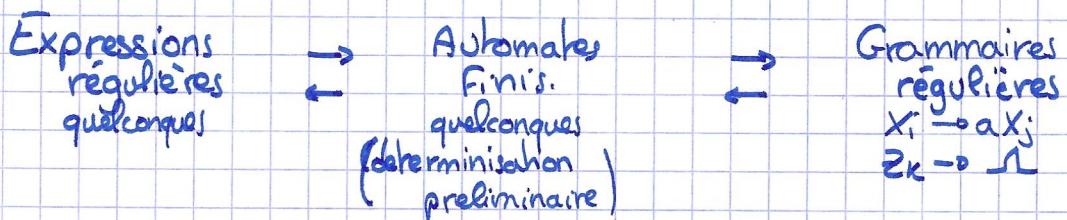
Sinon, il faut décomposer les états jusqu'à obtenir un ens. stable par  $A$ .

Ex:  $|T = \{q_1\} \quad Q \setminus T = \{i, q_2\}|$  stable par  $A$  donc automini.

$$\Delta(\{q_1\}, a) = \emptyset \quad \Delta(\{i, q_2\}, a) = \{q_1\} = T$$

$$\Delta(\{q_1\}, b) = \{q_2\} \subseteq Q \setminus T \quad \Delta(\{i, q_2\}, b) = \emptyset \rightarrow \boxed{i, q_2} \xrightarrow{a} \boxed{q_1} \xrightarrow{b} \boxed{q_2} D_c$$

$$\Delta(\{a\}, c) = \{q_2\} = T \quad \Delta(\{i, q_2\}, c) = \emptyset$$



Grammaires non contextuelles:  $X_i \rightarrow \alpha_i \quad \alpha_i \in (A \cup V)^*$

plus général que les grammaires rég.

partie droite est une séquence de (non) terminaux

Question: Peut-on transformer n'importe quelle grammaire non contextuelle en grammaire régulière?

Non

Ex: Langages de Dick qui associent symboles ouvrants (Ex: (, [, {, begin...)) à un symbole fermant (Ex: ), ], }, end...)). à chaque ouvrant doit être associé un fermant.

$$X_i \rightarrow \alpha_i \mid \beta_i \mid F_i \quad \alpha_i, \beta_i, F_i \in (A \cup V)^*$$

$$S \rightarrow (S) \quad \left| \begin{array}{l} \alpha_i = \text{N} \\ \beta_i = ( \\ F_i = ) \end{array} \right. \quad L = \{ (n \text{id})^n \mid n \in \mathbb{N}^* \} \text{ n'est pas régulier}$$

$$S \rightarrow \text{id} \quad \left| \begin{array}{l} \alpha_i = \text{id} \\ \beta_i = \text{id} \\ F_i = \text{id} \end{array} \right. \text{ ne peut être accepté par un automate.}$$

$$S \rightarrow (X) \quad \left| \begin{array}{l} \text{régulier à droite} \\ X \rightarrow S \quad \text{régulier à gauche} \\ S \rightarrow \text{id} \end{array} \right. \text{ n'est pas un langage régulier.}$$

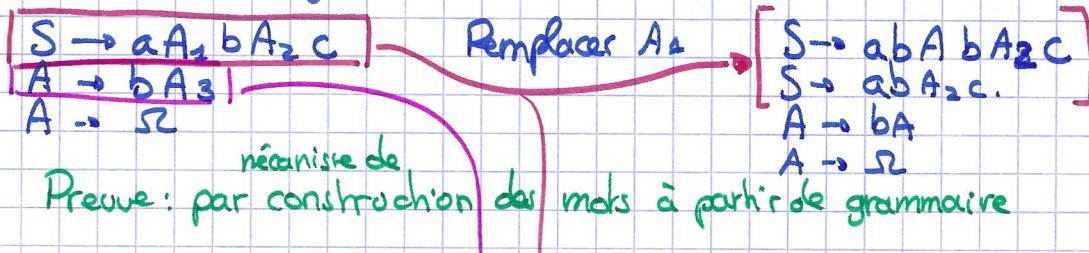
Pour déterminer si un langage est régulier, il faut essayer de transformer ses règles de production en:

1) grammaire régulière à droite : sous → langage régulier.

2) règle de la forme:  $X_i \rightarrow \alpha_i \mid \beta_i \mid F_i$ . éclate → langage pas régulier → algébrique.

Déf: Substitution de non terminaux.

$\{X \rightarrow \alpha_i\}$ : ensemble de toutes les règles pour le non terminal  $X$ . La grammaire obtenue en remplaçant  $X$  dans la partie droite d'une règle de production:  $Y \rightarrow \beta X \gamma$  par  $\{\beta_i \text{ règles à } Y \rightarrow \beta_i X_i \gamma\}$  déchir le même langage.



Réplacer  $A_2$ :

$$\begin{aligned} S &\rightarrow a A_1 b b A_2 c. \\ S &\rightarrow a A_1 b c \\ A &\rightarrow b A. \\ A &\rightarrow \Sigma. \end{aligned}$$

Réplacer  $A_3$ :

$$\begin{aligned} S &\rightarrow a A_2 b A_3 c \\ A &\rightarrow b b A \\ A &\rightarrow b \\ A &\rightarrow \Sigma. \end{aligned}$$

Rq:  $\left. \begin{array}{l} S \rightarrow (X) \\ X \rightarrow S \\ S \rightarrow id \end{array} \right\} \quad \left. \begin{array}{l} S \rightarrow (S) \\ X \rightarrow S \\ S \rightarrow id \end{array} \right\}$  ← on peut supprimer cette règle car inaccessible dp axiome.

Def: élimination des non terminaux inutiles.

Les non terminaux qui ne sont pas accessibles depuis l'axiome en appliquant les règles peuvent être éliminés.

Def: élimination des règles de production inutiles / mal fondées.

les règles qui ne peuvent pas être utilisées pour construire un mot fixe peuvent être éliminées.

Ex:  $S \rightarrow aA \quad A \rightarrow aA \quad A \rightarrow \Sigma. \quad L(A) = \{ a^n \mid n \in N \} = a^*$   
 $S \rightarrow bB \quad B \rightarrow bB \quad B \rightarrow \Sigma. \quad L(B) = \{ b^\omega \}$   
 infini de  $b$ .

Def: élimination de la récursivité à gauche.

$X \rightarrow Y\alpha$ , reg. à gauche.

$$\left\{ \underbrace{X \rightarrow X\alpha_i}_{\text{récursivité à gauche.}} \right\} \text{ et } \left\{ X \rightarrow \beta_j \right\}.$$

toutes les productions de  $X$ .

remplacer les règles de  $X$  par:

$$\left\{ X \rightarrow \beta_j L \right\} \text{ et } \left\{ L \rightarrow \alpha_i L \right\}, \text{ et } L \rightarrow \Sigma$$

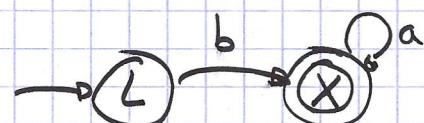
ne modifie pas le langage décrit.

↪  $L$  est un nouveau non terminal.

Ex:  $L \rightarrow La$      $\alpha_1 = a$   
 $L \rightarrow b$ .     $\beta_1 = b$ .  
 reg gauche récursive

$$\begin{array}{l} L \rightarrow bX \\ X \rightarrow aX \\ X \rightarrow \Sigma. \end{array}$$

reg droite



Ex:  $\left. \begin{array}{l} X \rightarrow Ya \\ Y \rightarrow Xb \\ Y \rightarrow \Sigma. \end{array} \right\}$  n'est pas directement récursive mais indirectement : substitution  
 pour faire apparaître récursivité directe

Substitution  $Y$ :  $X \rightarrow Xba$      $X \rightarrow a$ .

$Y$  inaccessible.

éliminer la récursivité à gauche.

$$X \rightarrow aL \quad L \rightarrow baL \quad L \rightarrow \Sigma$$

Def: Factorisation à gauche

(intro suffis P new non terminale)

$\{X \rightarrow \alpha \beta_i\}$  peuvent être remplacées par:  $X \rightarrow \alpha P$  ( $P \rightarrow \beta_i$ ).  
 sans modifier le langage décrir.  
 $\alpha$  prefixe commun

Ex:  $C \rightarrow R s N_i$   
 $C \rightarrow R$   
 $C \rightarrow I$

axiome C.

$R \rightarrow s N$   
 $R \rightarrow N$   
 $I \rightarrow R_i$   
 $N \rightarrow N_e$   
 $N \rightarrow c$

1  
Essayons de la transformer en grammaire régulière à droite factorisée à gauche.

1 - Éliminons la récursivité à gauche.

$N \rightarrow c X$   
 $X \rightarrow c X$   
 $X \rightarrow \Omega$

 $X \rightarrow c X \mid \Omega$ 

2 - Point de départ : axiome

 $C \rightarrow R s N_i \mid R \mid I$ 

3 - Substituons les non-terminaux à gauche pour faire apparaître des termes ou ↗

 $C \rightarrow s N s N_i \mid N s N_i \mid s N \mid N \mid R_i$  $C \rightarrow s N s N_i \mid c X s N_i \mid s N \mid c X \mid s N_i \mid N_i$ 
 $C \rightarrow s N s N_i \mid c X s N_i \mid s N \mid c X \mid s N_i \mid c X_i$   
 Y Z Y Z Y Z

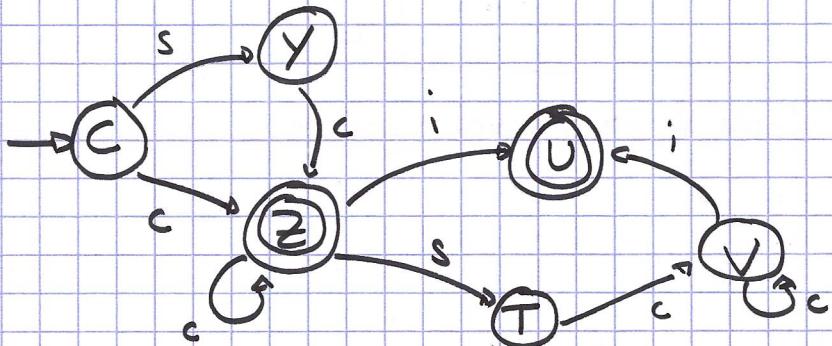
4 - Factorisons s et c

 $C \rightarrow s Y \mid c Z$  régulière à droite $Y \rightarrow N s N_i \mid N_i N_i$  $Y \rightarrow c \underbrace{X s N_i}_z \mid c \underbrace{X}_z \mid c \underbrace{X_i}_z$  $Y \rightarrow c Z$  $Z \rightarrow X s N_i \mid X \mid X_i$  $Z \rightarrow c \underbrace{X s N_i}_z \mid s N_i \mid c \underbrace{X}_z \mid \Lambda \mid c \underbrace{X_i}_z$  $Z \rightarrow c Z \mid S T \mid ; U \mid \Lambda$ 
 $T \rightarrow N_i$   
 $T \rightarrow c X_i$   
 $T \rightarrow c V$ 
 $V \rightarrow X_i$   
 $V \rightarrow c X_i \mid i \mid U$ 
 $V \rightarrow c V \mid i \mid U$

## Grammaire régulière à gauche:

$$\begin{aligned}
 C &\rightarrow sY \mid cZ \\
 Y &\rightarrow cZ \\
 Z &\rightarrow cZ \mid sT \mid iU \mid \lambda \\
 T &\rightarrow cv \\
 V &\rightarrow cv \mid iU \\
 U &\rightarrow \lambda
 \end{aligned}$$

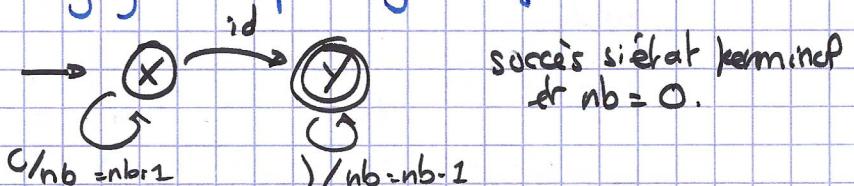
L'héuristique permet de savoir si la grammaire est régulière à droite, le langage est régulier.



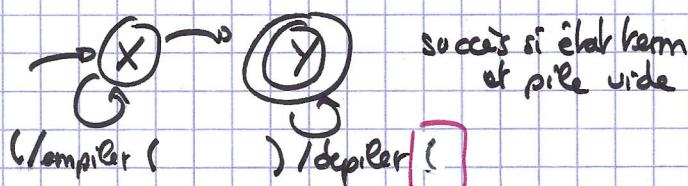
Perdre de la compréhensibilité de la grammaire

Question: Que faire si le langage n'est pas régulier?

Ex:  $S \rightarrow (S)$   
 $S \rightarrow id$



Automates avec compléments



Vérif que le sommet contient (.

## III / Automates finis à pile

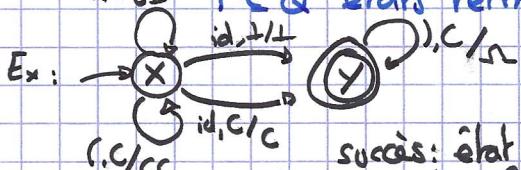
Def: Automate fini à pile

A : alphabet

Q : états

i  $\in Q$ : état initial

T  $\subseteq Q$  états terminaux.



succès: état termin. sommet pile: 1

$\Gamma$ : alphabet de pile

$\perp \in \Gamma$ : fin de pile

symbole en entrée ou arbitraire E

$\delta$ :  $Q \times A \cup \{\epsilon\} \times \Gamma \rightarrow P(Q \times \Gamma^*)$

étalement sommet de pile état vide qui remplace le sommet de la pile

$\delta(q, a, x) = \{ (q_1, x_1 \dots x_n) \}$

dépile  $a$ , empile  $x_1 \dots x_n$

+ Configuration de l'automate à pile :

$$(q, m, \underline{x}) \quad q \in Q, m \in A^*, x \in \Gamma^*$$

étalement mot en cours pile d'analyse

+ 2 transitions possibles :

• arbitraire (sur  $\epsilon$ ): dépend de la présence de  $x$  au sommet de la pile

$$(q, m, \underline{X}x) \rightarrow (q', m, \alpha \underline{x}) \text{ si } (q', \alpha) \in \delta(q, \underline{\epsilon}, \underline{X})$$

sommet de la pile.

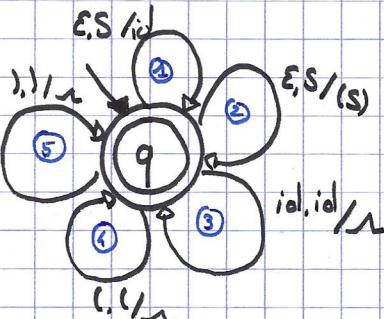
• lecture d'un symbole

$$(q, am, \underline{X}x) \rightarrow (q', m, \alpha \underline{x}) \text{ si } (q', \alpha) \in \delta(q, a, X).$$

Une transition dépile  $X$  et le remplace par  $\alpha$ .

$\alpha$  peut être le mot vide.

Ex :



$$\delta(q, \epsilon, S) = \{(q, id); (q, (S))\}$$

indéterminisme.

$$\delta(q, id, id) = \{(q, \lambda)\}$$

$$\delta(q, (.), (.) = \{(q, \lambda)\}$$

$$\delta(q, ., \lambda) = \{(q, \lambda)\}.$$

$$S \rightarrow (S)$$

$$S \rightarrow id$$

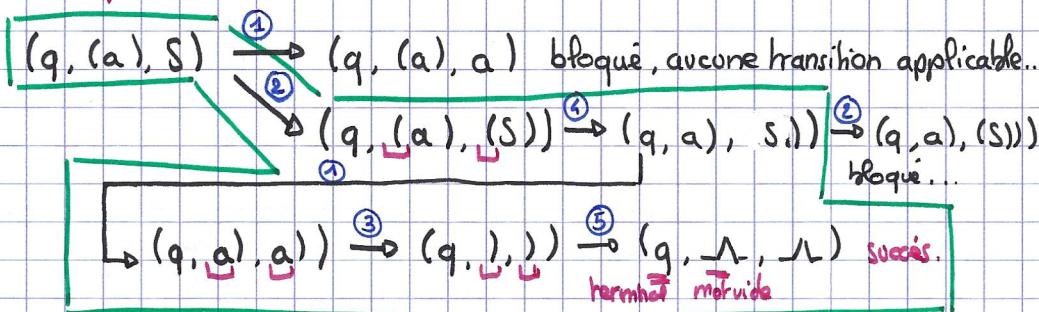
$$A = \{ ., id \}$$

$$\Gamma = A \cup S$$

$$\perp = S$$

fond de pile

Analyse de l'entrée (a)



Remarque : construction de (a) à partir de la grammaire

$$S \Rightarrow (S) \Rightarrow (a)$$

$$S \Rightarrow (S) \quad S \Rightarrow \lambda$$

transition ② transition ③

cet automate à pile simule la construction de la dérivation de  $S$  vers  $(a)$ , c'est à dire : il applique dans le même ordre les règles de productions. (les autres transitions consomment des symboles du mot).

Définition : (correspondance grammaire non contextuelle et automate à pile).

$(A, V, S, P)$  une grammaire non contextuelle  
 $P = \{x_i \rightarrow \alpha_i\}_{i \in \mathbb{N}}, x_i \in V, \alpha_i \in (A \cup V)^*$

des mots du langage décrit par cette grammaire sont acceptés par l'automate à pile défini

$Q = \{q_0, q_1, \dots, q_n\}, \Sigma = T \cup \{ \lambda \}, \delta: Q \times \Sigma \times \Gamma \rightarrow Q \times \Gamma, q_0 \in Q, \lambda \in \Gamma$

$\forall X_i \rightarrow \alpha_i \in P, \delta(q, \epsilon, X_i) = \{(\alpha_i, \cdot)\}$  application des règles de production.

$\forall a \in A, \delta(q, a, a) = \{(\alpha, \cdot)\}$  consommation des terminaux.

Preuve: l'automate simule la construction d'une dérivation.

Rq: La transformation automate à pile vers grammaire non contextuelle peut également être définie mais sort du contexte du cours.

Rq: L'automate ainsi construit est en général indéterministe  $\{X \rightarrow \beta_j\}_{j \in \{1, n\}} \subseteq P$  les différentes règles du non terminal  $X \quad \delta(q, \epsilon, X) = \{(\alpha, \beta_j)\}_{j \in \{1, n\}}$

Def: (Automate à pile indéterministe)

Un automate à pile est indéterministe si IP existe :

$\text{card}(\delta(q, \epsilon, X)) > 1$  ou  $\text{card}(\delta(q, a, X)) > 1 \rightarrow$  plusieurs états pour une même transition.

ou  $[\text{card}(\delta(q, \epsilon, X)) > 1 \text{ et } \text{card}(\delta(q, a, X)) > 1]$

↳ Transition sur  $\epsilon$  et  $X$  avec transition sur  $a$  et  $X$

Rq: l'utilisation de la pile réduit l'indéterminisme sur  $\epsilon$ , par rapport aux automates finis.

$\text{card}(\delta(q, \epsilon, X)) = 1$  et  $\forall a \in A, \text{card}(\delta(q, a, X)) = 0$  n'est pas indé-

Théorème: Les automates à pile indéterministes sont strictement plus puissants (plus expressifs) que les automates à pile déterministes.  
Il existe des langages qui ne sont pas acceptés par des automates à pile déterministes et sont acceptés par des automates à pile indéterministes.

Ex: Palindrome :  $S \rightarrow aSa \quad a \in A$   
 $S \rightarrow \lambda \quad S \lambda a \quad a \in A$

⇒ Conséquence: IP existe des langages décrits par des grammaires non contextuelles pour lesquels il n'existe pas d'automates finis à pile déterministe qui les accepte.

Rq: Catégories de langage (Chomsky)

Type 0

grammaire croisee  
production:  
 $\alpha \rightarrow \beta$   
 $\alpha \in (A \cup V)^*$   
 $\beta \in (A \cup V)^*$   
machines de Thuring

Type 1

grammaire contextuelle  
 $\alpha \beta \gamma \rightarrow \gamma$   
 $\alpha \in V$   
 $\gamma, \beta, \gamma \in (A \cup V)^*$

Type 2

(algébrique)  
grammaire non contextuelle  
automates à pile  
indéterministe  
analyse syntaxique

Type 3

(régulier)  
équivalent aux automates  
grammaire régulière  
expression régulière.

analyse lexicale

### ► Grammaire LR (k) :

Left to Right

Right most

(K look ahead)

lecture de gauche à droite

construction de la dérivation la plus à droite

lecture de K

symboles du mot pour décider de faire une transition

(Algorithmie)

Technique de construction d'automates à pile à partir d'une grammaire non contextuelle. Si cet automate est déterministe, la grammaire est LR(k).

Rôle de la plupart des algorithmes de génération des syntactiques (Kenhir, Yacc, Bison, ...)

$\Sigma : k = 0$   
 $a.m : k = 1$

Théorème: Tous les langages déterministes ont une grammaire LR(k).

### ► Grammaire LL (k)

Left to Right Leftmost (k look ahead).

construction de la dérivation la plus à gauche.

Technique de construction d'automate à pile à partir d'une grammaire non contextuelle strictement moins puissante que LR(k) : il existe des langages déterministes qui n'ont pas de grammaire LL(k).

Principe:  $\{ X \rightarrow \alpha_i \}_{i \in \{1, n\}} \subseteq P$  l'ensemble de toutes les règles de prod. pour le non term.

La grammaire est LL(k) si l'on peut choisir une unique règle de prod.  $X \rightarrow \alpha_i$  pour analyser un mot m en utilisant uniquement les k premiers symboles de m.

On associe à chaque règle ses symboles directeurs SD<sub>k</sub> ( $X \rightarrow \alpha$ ) l'ensemble des mots de longueur  $\leq k$  qui permettent de choisir d'utiliser  $X \rightarrow \alpha$ .

Une grammaire est LL(k)  $\Leftrightarrow \forall i, j \in \{1, \dots, n\} \quad i \neq j \Rightarrow SD_k(X \rightarrow \alpha_i) \cap SD_k(X \rightarrow \alpha_j) = \emptyset$

$SD_k(X \rightarrow \alpha) = \text{Premiers}_k(\alpha) \cup \{\lambda\} \cup \text{Suivants}_k(X)$

Premiers<sub>k</sub>( $\alpha$ ) : ens. des mots de taille  $\leq k$  qui sont dérivés de  $\alpha$   
 Suivants<sub>n</sub>(X) : ens. des \_\_\_\_\_ qui peuvent apparaître après X dans les mots du langage.