Inf 431 – Cours 15

Grammaires formelles

jeanjacqueslevy.net

secrétariat de l'enseignement:
Catherine Bensoussan
cb@lix.polytechnique.fr
Aile 00, LIX,
01 69 33 34 67

www.enseignement.polytechnique.fr/informatique/IF

Plan

1. La classification de Chomsky
2. Langages réguliers et expressions régulières
3. Règles avec parties droites vides dans les grammaires algébriques
4. Forme normale de Chomsky
5. Propriétés de fermeture
6. Non expressivité
7. Langages récursivement énumérables

Bibliographie

John E. Hopcroft , Rajeev Motwani , Jeffrey D. Ullman, *Introduction to Automata Theory, Languages, and Computation (2nd Edition)*, Addison-Wesley, 2000.

Dexter C. Kozen, *Automata and Computability*, Springer, 1999.

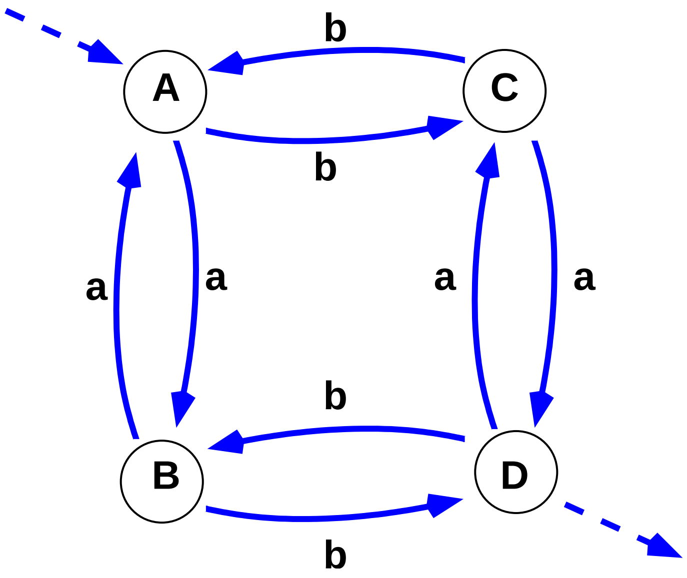
Langage régulier (rationnel)

- Nombre impair de a et nombre impair de b

$$
\begin{array}{llll}
A \longrightarrow aB & B \longrightarrow aA & C \longrightarrow aD & D \longrightarrow aC \\
A \longrightarrow bC & B \longrightarrow bD & C \longrightarrow bA & D \longrightarrow bB \\
 & B \longrightarrow b & C \longrightarrow a &
\end{array}
$$

Exemple de dérivation :

$\underline{A} \longrightarrow a\underline{B} \longrightarrow aa\underline{A} \longrightarrow aab\underline{C} \longrightarrow aaba\underline{D} \longrightarrow aabaa\underline{C} \longrightarrow aabaaa$



Langage algébrique (*context-free*)

- langage $\{a^n b^n \mid n > 0\}$

  $S \to aSb \qquad S \to ab$

  Exemple de dérivation : $\underline{S} \to a\underline{S}b \to aa\underline{S}bb \to aaa\underline{S}bbb \to aaaabbbb$

- système de parenthèses

  $S \to aSbS \qquad S \to abS \qquad S \to aSb \qquad S \to ab$

  Exemple de dérivation :

  $\underline{S} \to a\underline{S}bS \to aa\underline{S}bbS \to aaabbb\underline{S} \to aaabbba\underline{S}b \to aaabbbaabb$

- automates à pile [Chomsky, Schutzenberger]

  On rajoute une pile au fonctionnement d'un automate fini

  Par exemple pour $\{a^n b^n \mid n > 0\}$. Table de transitions (q_1 état initial, Z dans la pile, $F = \{q_4\}$)

	a, Z	a, X	b, X	ϵ, Z
q_1	(q_2, ZX)			
q_2		(q_2, XX)	(q_3, ϵ)	
q_3			(q_3, ϵ)	(q_4, Z)
q_4				

Langage *context-sensitive*

- langage $\{a^n b^n c^n \mid n > 0\}$

$$S \rightarrow \epsilon \qquad S \rightarrow A$$

$$A \rightarrow aABC \qquad CB \rightarrow BC \qquad aB \rightarrow ab \qquad bC \rightarrow bc$$

$$A \rightarrow aBC \qquad bB \rightarrow bb \qquad cC \rightarrow cc$$

Exemple de dérivation :

$$\underline{S} \rightarrow \underline{A} \rightarrow a\underline{A}BC \rightarrow aa\underline{A}BCBC \rightarrow aaaB\underline{CB}CBC$$
$$\rightarrow aaaBBC\underline{CB}C \rightarrow aaaBB\underline{CB}CC \rightarrow aa\underline{aB}BBCCC$$
$$\rightarrow aaa\underline{bB}BCCC \rightarrow aaab\underline{bB}CCC \rightarrow aaabb\underline{bC}CC$$
$$\rightarrow aaabbb\underline{cC}C \rightarrow aaabbbc\underline{cC} \rightarrow aaabbbccc$$

Le contexte intervient ; les dérivations ne font pas décroitre la taille des mots.

Exercice 1 Donner une grammaire pour les langages suivants :

- $\{a^n b^n c^n d^n \mid n > 0\}$
- $\{a^{n^2} \mid n \geq 0\}$
- $\{uu \mid u \in \Sigma^*\}$
- $\{a^n b^n c^p \mid n \geq 0, p \geq 0\} \cup \{a^n b^p c^p \mid n \geq 0, p \geq 0\}$

Langages et Grammaires

Un langage L est un ensemble de mots sur l'alphabet Σ ($L \subset \Sigma^*$).

Une grammaire est un 4-uplet $G = (\Sigma, V_N, S, \mathcal{P})$

- Σ alphabet des symboles terminaux
- V_N ensemble de non terminaux, $V = \Sigma \cup V_N$
- $S \in V_N$ axiome
- $\mathcal{P}$ ensemble fini de règles de productions de la forme $\alpha_i \rightarrow \beta_i$ ($\alpha_i \in V^+ - \Sigma^*$, $\beta_i \in V^*$)

Classification de Chomsky

type	productions	catégorie
0	$\alpha \longrightarrow \beta \qquad \alpha \longrightarrow \epsilon$	
1	$\alpha \longrightarrow \beta \qquad \lvert\alpha\rvert \leq \lvert\beta\rvert$	*context sensitive*
2	$A \longrightarrow \beta$	*context-free*
3	$A \longrightarrow aB \qquad A \longrightarrow a$	régulier

où $\alpha \in V^+ - \Sigma^*$, $\beta \in V^+$ et $A, B \in V_N$, $a \in \Sigma$.

Une exception est possible pour l'axiome S :

- il peut figurer dans une règle $S \longrightarrow \epsilon$
- alors il ne doit pas apparaître dans une partie droite de production.

Langage généré par une grammaire

- Une dérivation élémentaire $u\alpha v \longrightarrow u\beta v$ s'obtient par application d'une règle $\alpha \to \beta$ de production.
- Une dérivation $u \xrightarrow{*} v$ est une suite de dérivations élémentaires $u = u_0 \longrightarrow u_1 \longrightarrow u_2 \ldots u_n = v \ (n \geq 0)$
- Le langage généré par une grammaire $G = (\Sigma, V_N, S, \mathcal{P})$ est :

$$L(G) = \{u \mid S \xrightarrow{*} u \in \Sigma^*\}$$

- Un langage est de type i s'il existe une grammaire de type i qui le génère $(0 \leq i \leq 3)$. Soit $\mathcal{L}_i$ les langages de type i sur Σ.
- Comme G de type $i+1$ implique G de type i, on a

$$\boxed{\mathcal{L}_3 \subset \mathcal{L}_2 \subset \mathcal{L}_1 \subset \mathcal{L}_0}$$

.

- $\{a^n b^n \mid n \geq 0\} \in \mathcal{L}_2 - \mathcal{L}_3$
- $\{a^n b^n c^n \mid n \geq 0\} \in \mathcal{L}_1 - \mathcal{L}_2$

Notations abrégées

- Souvent on factorise les productions qui ont une même partie gauche grâce au (méta)symbole | (barre verticale) :

$$\alpha \rightarrow \beta_1 \mid \beta_2 \mid \ldots \beta_n$$

  pour

$$\alpha \rightarrow \beta_1 \quad \alpha \rightarrow \beta_2 \quad \ldots \quad \alpha \rightarrow \beta_n$$

- Une autre notation fréquente est proche de la BNF (Backus Naur Form), la flèche est remplacé par "::=" :

$$\alpha ::= \beta_1 \mid \beta_2 \mid \ldots \beta_n$$

- Par exemple, l'espace des termes t est défini par

$$t ::= t + t \mid t * t \mid t - t \mid t/t \mid nb$$

  qui ressemble à une définition récursive. Parfois on duplique les symboles non terminaux pour distinguer les occurences dans les parties droites :

$$t, t' ::= t + t' \mid t * t' \mid t - t' \mid t/t' \mid nb$$

Langage régulier et automate fini

Théorème 1 Un langage est régulier si et seulement s'il est généré par un automate fini.

- Démonstration : Si G est la grammaire (de type 3) générant L, on considère l'automate non-déterministe $\mathcal{A} = (\Sigma, V_N, S, \{q_f\}, \delta)$ tel que

  $B \in \delta(A, a)$ si $A \rightarrow aB$
  $q_f \in \delta(A, a)$ si $A \rightarrow a$

  Alors $T(\mathcal{A}) = L(G)$.

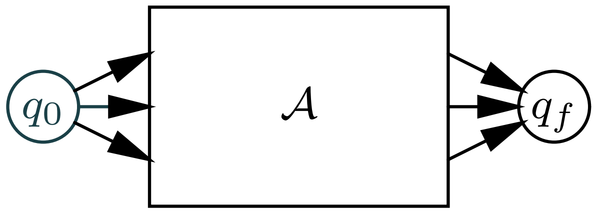
- Réciproquement, si $L = T(A)$ et $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$, on considère la grammaire $G = (\Sigma, Q, q_0, \mathcal{P})$ telle que

  $A \rightarrow aB$ si $B \in \delta(A, a)$
  $A \rightarrow a$ si $\delta(A, a) \cap F \neq \emptyset$

  Alors $L(G) = T(\mathcal{A})$. $\square$

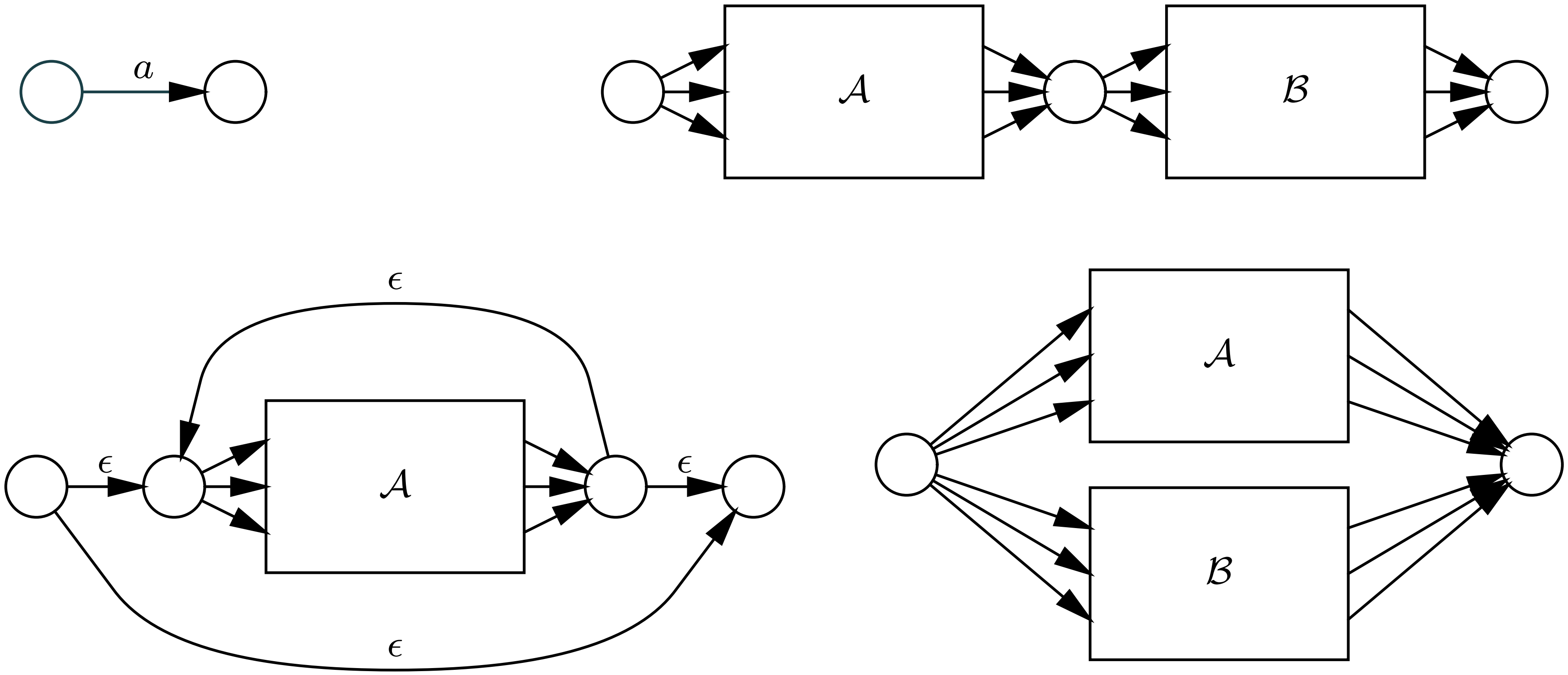
Expression régulière et automate (1/5)

- On admet l'équivalence entre automates finis avec ou sans ϵ-transitions ;
- on suppose que les automates finis n'ont qu'un seul état de fin q_f avec $\delta(q_f, a) = \emptyset$ pour tout $a \in \Sigma$;
- de même, on suppose $q_0 \notin \delta(q, a)$ pour tout $q \in Q$ et $a \in \Sigma$;



- alors les automates finis reconnaissent les expressions régulières par les constructions suivantes.

Expression régulière et automate (2/5)



- [construction due à [Thompson] (l'inventeur du système Unix)]

Expression régulière et automate (3/5)

Théorème 2 [Kleene] Un langage est régulier si et seulement s'il est décrit par une expression régulière.

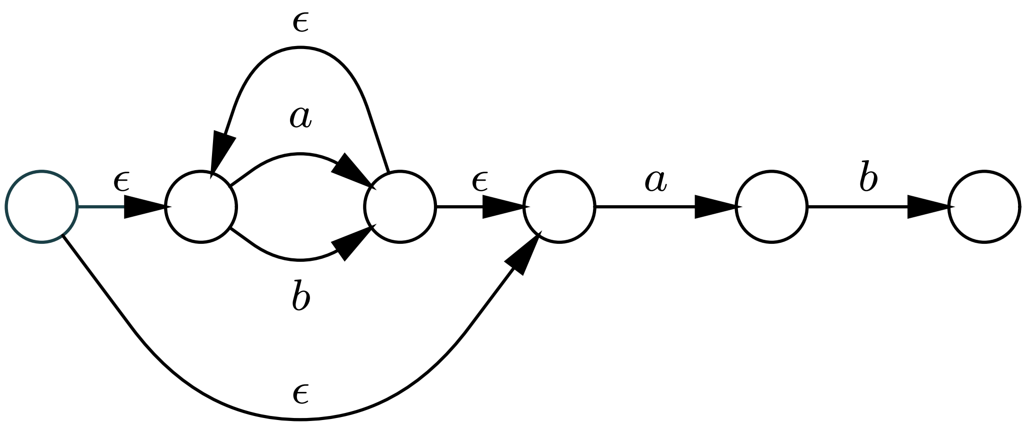
- Démonstration : Par la construction précédente, on montre par récurrence sur la taille de l'expression régulière que toute expression régulière est reconnue par un automate.
- Réciproquement, soit $A = (\Sigma, Q, q_0, F)$ avec $Q = \{q_0, q_1, \dots q_{n-1}\}$. Soit $L^k_{i,j}$ l'ensemble des mots permettant d'aller de q_i à q_j en passant par des états q_ℓ vérifiant $\ell < k$. Alors

$$\begin{aligned} L^0_{i,i} &= \{\epsilon\} \cup \{a \mid \delta(q_i, a) = q_i\} \\ L^0_{i,j} &= \{a \mid \delta(q_i, a) = q_j\} \\ L^{k+1}_{i,j} &= L^k_{i,j} \cup L^k_{i,k}(L^k_{k,k})^* L^k_{k,j} \end{aligned}$$

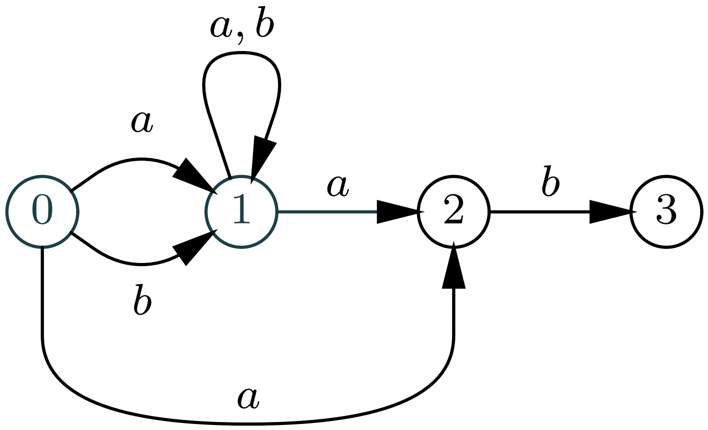
Et $L = T(A) = \bigcup_{q_i \in F} L^n_{0,i}$. $\square$

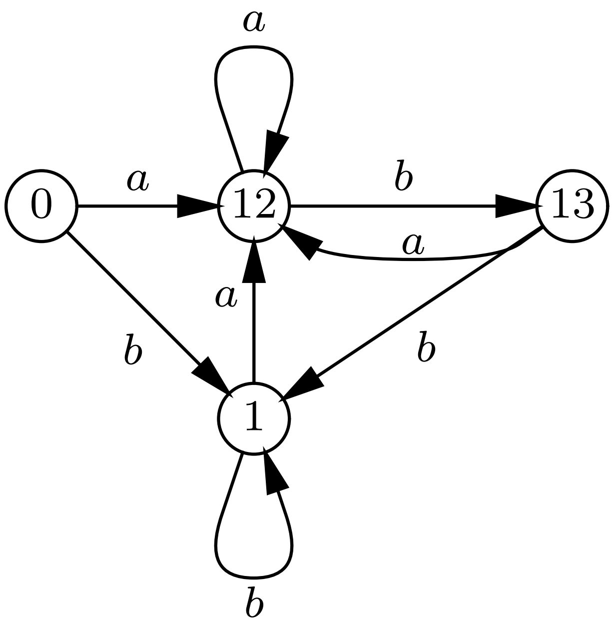
Expression régulière et automate (4/5)

Pour $(a+b)^*ab$, l'automate est :



En supprimant les ϵ-transitions et en déterminisant, on obtient :





Expression régulière et automate (5/5)

- L'automate obtenu par la construction précédente est non-déterministe, il a beaucoup de ϵ-transitions.
- On peut toujours déterminiser l'automate obtenu, et considérer l'automate minimal.
- Il existe des constructions plus directes pour obtenir de bons automates (peut-être pas minimaux) [Glushkov ; Brzozowski ; Berry-Sethi ; Berstel-Pin] en considérant les expressions régulières linéaires, et les expressions régulières dérivées.

Mot vide dans une grammaire algébrique

- Les productions $A \rightarrow \alpha$ doivent avoir une partie droite non vide ($\alpha \neq \epsilon$) à l'exception de l'axiome.
- Or, dans le cours 6, on avait par exemple autorisé
  $S \rightarrow aSbS \quad S \rightarrow \epsilon.$
- Mais les deux définitions de grammaires algébriques génèrent les mêmes langages algébriques.
  En effet, si on a $B \xrightarrow{*} \epsilon$ et une règle $A \rightarrow \alpha B \beta$, on ajoute la règle $A \rightarrow \alpha\beta$. Puis on supprime toutes les productions $C \rightarrow \epsilon$.
  Si on avait $S \xrightarrow{*} \epsilon$, on considère un nouvel axiome S' et on ajoute les nouvelles règles $S' \rightarrow \epsilon$ et $S' \rightarrow S$.

- Sur l'exemple précédent, cela donne successivement
  $S \rightarrow aSbS \quad S \rightarrow \epsilon \quad S \rightarrow abS \quad S \rightarrow aSb \quad S \rightarrow ab$
  $S' \rightarrow \epsilon \quad S' \rightarrow S \quad S \rightarrow aSbS \quad S \rightarrow abS \quad S \rightarrow aSb \quad S \rightarrow ab$

Exercice 2 Supprimer les parties droites vides dans ces deux grammaires vues au cours 6

$S \rightarrow \epsilon \quad S \rightarrow SS \quad S \rightarrow aSb$

$A \rightarrow \epsilon \quad A \rightarrow [\ A\ nb\ A\ ]$

Forme normale de Chomsky

Toute grammaire algébrique a une grammaire équivalente dont les règles sont de la forme $A \rightarrow a$, $A \rightarrow BC$ $(a \in \Sigma)$, à l'exception de l'axiome qui peut avoir une règle $S \rightarrow \epsilon$.

Démonstration :

- On trouve les non-terminales B et C telles que $B \xrightarrow{*} C$.
- Si on a $A \rightarrow \alpha B \beta$ avec $C \rightarrow \gamma$ et $|\gamma| \geq 2$, on ajoute $A \rightarrow \alpha\gamma\beta$.
- Puis on supprime les règles $A \rightarrow B$.
- A présent, les règles sont de la forme $S \rightarrow \epsilon$, $A \rightarrow a$, $A \rightarrow \alpha$ avec $|\alpha| \geq 2$.
  Pour chaque production $A \rightarrow \alpha = X_1 X_2 \ldots X_n$, on ajoute de nouvelles non-terminales Y_i et on pose
  - $Y_i \rightarrow X_i$ quand $X_i \in \Sigma$
  - $A \rightarrow X_1 Y_2 \quad Y_2 \rightarrow X_2 Y_3 \quad \ldots \quad Y_{n-1} \rightarrow X_{n-1} X_n$.

Exercice 3 Mettre en forme normale de Chomsky les grammaires de l'exercice précédent.

La forme normale de Chomsky a une application : l'algorithme CYK pour analyser toute grammaire context-free en temps $O(n^3)$

Propriétés de clôtures (1/2)

- Un morphisme ϕ est toute fonction de Σ dans Σ_1^* étendue sur les mots et sur les langages de manière naturelle :
  $\phi(uv) = \phi(u)\phi(v)$ et $\phi(L) = \{\phi(w) \mid w \in L\}$
- Les langages réguliers sont clos par morphisme.
- Les langages algébriques sont clos par morphisme.

- Les langages réguliers forment une algèbre de Boole (clos par union, intersection et complément).
- Les langages algébriques sont clos par union ; ils ne sont pas clos par intersection puisque :
  $\{a^n b^n c^n \mid n \geq 0\} = \{a^n b^n c^* \mid n \geq 0\} \cap \{a^* b^n c^n \mid n \geq 0\}$

Propriétés de clôtures (2/2)

- Les langages algébriques sont clos par intersection avec un régulier : L algébrique, R régulier $\Rightarrow L \cap R$ algébrique
- Démonstration :
  - $G = (\Sigma, V_N, S, \mathcal{P})$ génère L ;
    $\mathcal{A} = (\Sigma, Q, q_0, F)$ reconnait R ;
  - la grammaire G' suivante génère $L \cap R$:
    $G' = (\Sigma, V, S', \mathcal{P}')$
    $V = \{A_{q,q'} \mid A \in V_N$ où $q, q' \in Q\} \cup \{S'\}$
    $\mathcal{P}'$ contient les règles productions suivantes :
    - $A_{q'_0,q'_n} \longrightarrow A^1_{q'_0,q'_1} A^2_{q'_1,q'_2} \cdots A^n_{q'_{n-1},q'_n}$
      si $A \longrightarrow A^1 A^2 \cdots A^n$ est une production de $\mathcal{P}$ avec :
      $A^i \in V_N \cup \Sigma$ et $\delta(q'_{i-1}, A^i) = q'_i$ si $A^i \in \Sigma$
    - $S' \longrightarrow S_{q_0,q}$ avec $q \in F$

Non expressivité

- Lemme de l'étoile

  Si L est régulier, il existe n tel que :
  $\forall w \in L \ \ |w| > n \ \Rightarrow \ w = uvx$ avec $|v| > 0$ et $uv^n x \in L$ pour tout $n \geq 0$.

  Exercice 4 Montrer que $\{a^n b^n \mid n \geq 0\}$ n'est pas régulier
  Exercice 5 Montrer que $\{a^n b a^n \mid n \geq 0\}$ n'est pas régulier

- Lemme de la pompe

  Si L est algébrique, il existe n tel que :
  $\forall w \in L \ \ |w| > n \ \Rightarrow \ w = uvxyz$ avec $|vy| > 0$ et $uv^n xy^n z \in L$ pour tout $n \geq 0$.

  Exercice 6 Montrer que $\{a^n b^n c^n \mid n \geq 0\}$ n'est pas algébrique.

Langage récursivement énumérable

Théorème 3 Un langage est de type 0 (dans la classification de Chomsky) ss'il est récursivement énumérable.

- Démonstration : D'abord si un langage est de type 0, on peut construire une machine de Turing qui le génère.
- Réciproquement, supposons qu'un langage L vérifie $L = T(M)$ avec $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ est une machine de Turing. On construit la grammaire $G = (\Sigma \cup \{\$\}, Q \cup \Gamma \cup \{S, S'\}, S, \mathcal{P})$ avec comme productions :

  $qX \rightarrow Yp$ si $(u, q, Xv) \longrightarrow (uY, p, v)$
  $q\$ \rightarrow Yp\$$ si $(u, q, \epsilon) \longrightarrow (uY, p, \epsilon)$
  $ZqX \rightarrow pZY$ si $(uZ, q, Xv) \longrightarrow (u, p, ZY)$
  $Zq\$ \rightarrow pZY\$$ si $(uZ, q, \epsilon) \longrightarrow (u, p, ZY)$
  $S \rightarrow q_0 S'$
  $S' \rightarrow aS'$
  $S' \rightarrow \$$

  Alors $u\$ \in L(G)$ ssi $u \in T(M)$. $\square$

Quelques exercices

Exercice 7 Montrer qu'il existe un algorithme pour reconnaître un langage contexte-sensitif, c'est-à-dire répondant oui si le mot d'entrée est dans le langage, et non s'il n'appartient pas.

Exercice 8 Montrer qu'il n'existe qu'un semi-algorithme pour reconnaître un langage récursivement énumérable, c'est-à-dire répondant oui si et seulement si le mot d'entrée est dans le langage. Le programme peut ne pas fournir de réponse si le mot n'est pas dans le langage.

Exercice 9 Montrer qu'il existe une analyse descendante fonctionnant pour tout langage algébrique, mais avec retours en arrière (*backtracking*) dans le mot d'entrée. Complexité ?

Exercice 10 Quel est la notion d'arbre syntaxique pour les langages non algébriques ?

Exercice 11 Donner un algorithme pour trouver les non-terminales A telles que $A \xrightarrow{*} \epsilon$ dans un langage algébrique.

Exercice 12 Donner un algorithme pour trouver les non-terminales A telles que $A \xrightarrow{*} B$ $(B \in V_N)$ dans un langage algébrique.