Chapitre 4

Langages réguliers et hors contexte

Alphabets et mots

Définition 4.1. Un alphabet est un ensemble fini et non vide de symboles. ▲

Exemples 4.2.

$$\Sigma_1 = \{\mathtt{a}, \mathtt{b}, \ldots, \mathtt{z}\}$$

$$\Sigma_2 = \{\mathtt{a}, \mathtt{b}\}$$

$$\Sigma_3 = \{\mathtt{0}, \mathtt{1}\}$$

$$\Sigma_4 = \{\mathtt{0}, \mathtt{1}, \ldots, \mathtt{9}\}$$

$$\Sigma_5 = \{\clubsuit, \diamondsuit, \heartsuit, \spadesuit\}$$

$$\Sigma_6 = \{\mathtt{0}, \mathtt{1}, \ldots, 9, +, -, *, /, (,)\}$$

$$\Sigma_7 = \mathsf{ASCII}$$

$$\Sigma_8 = \mathsf{ADN} = \{\langle \textsc{Adénine} \rangle, \langle \textsc{Cytosine} \rangle, \langle \textsc{Guanine} \rangle, \langle \textsc{Thymine} \rangle\}$$

▲

Définition 4.3. Un mot est une suite finie, possiblement vide, de symboles appartenant à un alphabet. ▲

Exemple 4.4. `abba` est un mot sur l'alphabet $\{\mathtt{a}, \mathtt{b}\}$. ▲

Définition 4.5. La lettre grecque epsilon minuscule, ε, dénote le mot vide, quel que soit l'alphabet en usage. ▲

Définition 4.6. Σ^* est l'ensemble de tous les mots pouvant être formés à partir de l'alphabet Σ. ▲

Remarque 4.7. On a toujours $\varepsilon \in \Sigma^*$, quel que soit Σ. ▲

Exemples 4.8.

$$\begin{aligned}
\Sigma_2 &= \{\mathtt{a}, \mathtt{b}\} \\
\Sigma_2^* &= \{\varepsilon, \mathtt{a}, \mathtt{b}, \mathtt{aa}, \mathtt{ab}, \mathtt{ba}, \mathtt{bb}, \mathtt{aaa}, \mathtt{aab}, \ldots\}
\end{aligned}$$

$$\begin{aligned}
\Sigma_3 &= \{0, 1\} \\
\Sigma_3^* &= \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, \ldots\}
\end{aligned}$$

$$\begin{aligned}
\Sigma_6 &= \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -, *, /, (,)\} \\
\Sigma_6^* &\ni (25 + 50) * 56 \\
\Sigma_6^* &\ni (()25 + + + 6
\end{aligned}$$

▲

Définition 4.9. Soit $w = w_1 w_2 \dots w_k \in \Sigma^*$, alors

$$|w| = k.$$

L'entier k est la longueur du mot w, c'est-à-dire le nombre de symboles dans w.

▲

Exemple 4.10.

$$|\texttt{allo}| = 4.$$

▲

Définition 4.11. Soit $w = w_1 w_2 \dots w_k \in \Sigma^*$ et $\mathtt{a} \in \Sigma$, alors

$$|w|_{\mathtt{a}} = \#\{w_i \mid w_i = \mathtt{a}\},$$

c'est-à-dire le nombre d'occurrences du symbole a dans le mot w. ▲

Exemples 4.12.

$$|\mathtt{yabadabadoo}|_{\mathtt{a}} = 4 \qquad \text{et} \qquad |\mathtt{yabadabadoo}|_{\mathtt{b}} = 2.$$

▲

Définition 4.13. Soient

$$x = x_1 x_2 \dots x_k \in \Sigma^* \quad \text{et} \quad y = y_1 y_2 \dots y_l \in \Sigma^*,$$

alors

$$x \cdot y = x_1 x_2 \dots x_k y_1 y_2 \dots y_l$$

est la concaténation de x et y. ▲

Remarque 4.14. $w \cdot \varepsilon = w$, quel que soit le mot w. ▲

Exemples 4.15.

$$\begin{aligned} \texttt{bon} \cdot \texttt{jour} &= \texttt{bonjour} \\ \texttt{bon} \cdot \varepsilon &= \texttt{bon} \end{aligned}$$

▲

Notations 4.16. Soient un mot $w \in \Sigma^*$, et un entier positif i, alors

$$w^i = \underbrace{w \cdot w \cdots w}_{i \text{ fois}},$$

et

$$w^0 = \varepsilon.$$

De plus, on omet souvent l'opérateur de concaténation :

$$xy = x \cdot y.$$

▲

Notation 4.17. Soit un mot $w \in \Sigma^*$, alors w^{R} est le renversé de w, c'est-à-dire le mot formé avec les symboles de w pris de droite à gauche. ▲

Définition 4.18. Soit $<_\Sigma$ une relation d'ordre sur l'alphabet Σ. On appelle ordre lexicographique la relation d'ordre suivante sur Σ^* :

$$|w| < |w'| \quad \Rightarrow \quad w <_{\Sigma^*} w'$$

$$\left.\begin{array}{l} |w| = |w'| \\ w = x \cdot \mathtt{a} \cdot y \\ w' = x \cdot \mathtt{b} \cdot z \\ \mathtt{a} <_\Sigma \mathtt{b} \end{array}\right\} \quad \Rightarrow \quad w <_{\Sigma^*} w'$$

▲

Exemple 4.19.

$$\begin{array}{rcl} \Sigma & = & \{\mathtt{0}, \mathtt{1}\} \\ \Sigma^* & = & \{\varepsilon, \mathtt{0}, \mathtt{1}, \mathtt{00}, \mathtt{01}, \mathtt{10}, \mathtt{11}, \mathtt{000}, \mathtt{001}, \mathtt{010}, \ldots\} \end{array}$$

▲

Définition 4.20. Un langage sur l'alphabet Σ est un sous-ensemble de Σ^*. ▲

Définition 4.21. On note $\{\}$ ou $\emptyset$ le langage vide, c'est-à-dire le langage qui ne contient aucun mot. ▲

Remarque 4.22. $\varepsilon \notin \emptyset$. ▲

Exemples 4.23. Divers langages sur l'alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}\}$:

$$
\begin{aligned}
L_1 &= \{\mathtt{a}, \mathtt{b}, \mathtt{abba}, \mathtt{abbb}\} \\
\\
L_2 &= \{w \mid \ |w| = 2k, k \in \mathbb{N}\} \\
&= \{\varepsilon, \mathtt{aa}, \mathtt{ab}, \mathtt{ba}, \mathtt{bb}, \mathtt{aaaa}, \mathtt{aaab}, \mathtt{aaba}, \mathtt{aabb}, \ldots\} \\
&= \text{les mots qui contiennent un nombre pair de symboles} \\
\\
L_3 &= \{w \mid w = aw',\ w' \in \Sigma^*\} \\
&= \{\mathtt{a}, \mathtt{aa}, \mathtt{ab}, \mathtt{aaa}, \mathtt{aab}, \mathtt{aba}, \mathtt{abb}, \mathtt{aaaa}, \ldots\} \\
&= \text{les mots qui commencent par } \mathtt{a}
\end{aligned}
$$

$$
\begin{aligned}
L_4 &= \{w \mid |w|_{\mathtt{a}} = |w|_{\mathtt{b}}\} \\
&= \{\varepsilon, \mathtt{ab}, \mathtt{ba}, \mathtt{aabb}, \mathtt{abab}, \mathtt{abba}, \mathtt{baab}, \mathtt{baba}, \ldots\} \\
&= \text{les mots qui contiennent autant de } \mathtt{a} \text{ que de } \mathtt{b}
\end{aligned}
$$

$$
\begin{aligned}
L_5 &= \{w \mid |w|_{\mathtt{a}} - |w|_{\mathtt{b}} \text{ est un nombre premier}\} \\
&= \{\mathtt{aaab}, \ldots, \mathtt{aabbabaaaaa}, \ldots\}
\end{aligned}
$$

▲

Exemples 4.24. Divers langages sur l'alphabet $\Sigma = \text{ASCII}$:

$$L_6 = \{w \mid w \text{ est une fonction syntaxiquement correcte dans le langage de programmation Java et cette fonction retourne toujours } \langle \text{TRUE} \rangle\}$$

$$L_7 = \{w \mid w \text{ compile sans erreur en C++}\}$$

$$L_8 = \{w \mid w \text{ est un programme C++ qui écrit “Hello world !”}\}$$

▲

Exemples 4.25. Divers langages sur l'alphabet $\Sigma = \mathsf{ADN}$:

$$L_9 = \{w \mid w \text{ est la séquence d'un gène bactérien}\}$$

$$L_{10} = \{w \mid w \text{ est le génome d'un individu}\}$$

▲

Opérations sur les langages ou sur les ensembles

- l'union
- l'intersection
- le complément
- le produit cartésien
- l'ensemble des parties

Définition 4.26. L'union de deux langages L_1 et L_2 :

$$L_1 \cup L_2 = \{w \mid w \in L_1 \text{ ou } w \in L_2\}.$$

▲

Exemple 4.27. Soient

$$\Sigma = \{\mathtt{a}, \mathtt{b}\} \text{ et } L_1 = \{\varepsilon, \mathtt{a}, \mathtt{aa}\} \text{ et } L_2 = \{\mathtt{a}, \mathtt{bb}\},$$

alors

$$L_1 \cup L_2 = \{\varepsilon, \mathtt{a}, \mathtt{aa}, \mathtt{bb}\}.$$

▲

Définition 4.28. L'intersection de deux langages L_1 et L_2 :

$$L_1 \cap L_2 = \{w \mid w \in L_1 \text{ et } w \in L_2\}.$$

▲

Exemples 4.29. Soient

$$\Sigma = \{\mathtt{a}, \mathtt{b}\} \text{ et } L_1 = \{\varepsilon, \mathtt{a}, \mathtt{aa}\} \text{ et } L_2 = \{\mathtt{a}, \mathtt{bb}\},$$

alors

$$\begin{aligned} L_1 \cap L_2 &= \{\mathtt{a}\}, \\ L_1 \cap \emptyset &= \emptyset. \end{aligned}$$

▲

Définition 4.30. Le complément d'un langage L sur l'alphabet Σ :

$$\overline{L} = \{w \in \Sigma^* \mid w \notin L\}.$$

▲

Exemple 4.31. Soit

$$\Sigma = \{\mathtt{a}, \mathtt{b}\} \text{ et } L = \{\varepsilon, \mathtt{a}, \mathtt{aa}\},$$

alors

$$\overline{L} = \{\mathtt{b}, \mathtt{ab}, \mathtt{ba}, \mathtt{bb}, \mathtt{aaa}, \mathtt{aab}, \mathtt{aba}, \mathtt{baa}, \mathtt{bba}, \ldots\}.$$

▲

Définition 4.32. Le produit cartésien de deux ensembles $\mathcal{A}$ et $\mathcal{B}$:

$$\mathcal{A} \times \mathcal{B} = \{(x, y) \mid x \in \mathcal{A} \text{ et } y \in \mathcal{B}\}.$$

▲

Remarques 4.33. $\#(\mathcal{A} \times \mathcal{B}) = (\#\mathcal{A})(\#\mathcal{B})$ et $\mathcal{A} \times \emptyset = \emptyset$. ▲

Exemple 4.34. Soient

$$\Sigma = \{\mathtt{a}, \mathtt{b}\},\ L_1 = \{\varepsilon, \mathtt{a}, \mathtt{aa}\} \text{ et } L_2 = \{\mathtt{a}, \mathtt{bb}\},$$

alors

$$L_1 \times L_2 = \{(\varepsilon, \mathtt{a}), (\varepsilon, \mathtt{bb}), (\mathtt{a}, \mathtt{a}), (\mathtt{a}, \mathtt{bb}), (\mathtt{aa}, \mathtt{a}), (\mathtt{aa}, \mathtt{bb})\}.$$

▲

Notation 4.35. Pour un ensemble $\mathcal{A}$ et un entier $k \geq 2$:

$$\begin{aligned} \mathcal{A}^k &= \underbrace{\mathcal{A} \times \mathcal{A} \times \cdots \times \mathcal{A}}_{k \text{ fois}} \\ &= \text{l'ensemble des } k\text{-tuplets d'éléments de } \mathcal{A}. \end{aligned}$$

▲

Remarque 4.36. $\#(\mathcal{A}^k) = (\#\mathcal{A})^k$. ▲

Définition 4.37. L'ensemble des parties d'un ensemble $\mathcal{A}$:

$$\mathcal{P}(\mathcal{A}) = \{\mathcal{B} \mid \mathcal{B} \subseteq \mathcal{A}\}.$$

▲

Remarque 4.38.

$$\#\mathcal{P}(\mathcal{A}) = 2^{\#\mathcal{A}}.$$

▲

Exemple 4.39. Soient

$$\Sigma = \{\mathtt{a}, \mathtt{b}\} \text{ et } L = \{\varepsilon, \mathtt{a}, \mathtt{aa}\},$$

alors

$$\mathcal{P}(L) = \{\emptyset, \{\varepsilon\}, \{\mathtt{a}\}, \{\mathtt{aa}\}, \{\varepsilon, \mathtt{a}\}, \{\varepsilon, \mathtt{aa}\}, \{\mathtt{a}, \mathtt{aa}\}, \{\varepsilon, \mathtt{a}, \mathtt{aa}\}\}$$

▲

Classe de langages

Définition 4.40. Une classe de langages est un ensemble de langages. ▲

La plupart du temps, on n'étudie que les classes qui sont associées à certains modèles calculatoires.

Par exemple, "la classe des langages qui sont reconnus par un type de machine ayant telles ressources et telles limitations..."

Définition 4.41. FINI désigne la classe de tous les langages finis, c'est-à-dire l'ensemble des langages qui contiennent un nombre fini de mots. ▲

Exemples 4.42. Si

$$\begin{aligned} \Sigma &= \{\mathtt{a}, \mathtt{b}\} \\ L &= \{\mathtt{abba}, \mathtt{baba}, \varepsilon\}, \end{aligned}$$

alors

$$\begin{aligned} L &\in \mathsf{FINI} \\ \emptyset &\in \mathsf{FINI} \\ \Sigma^* &\notin \mathsf{FINI} \end{aligned}$$

▲

Définition 4.43. On désigne la classe de tous les langages par UNIVERS. ▲

Fermeture d'une classe de langages par rapport à une opération

Définition 4.44. On dit qu'une classe est fermée par rapport à une opération, ou fermée pour une opération, si le résultat de l'opération est toujours un élément de la classe lorsque les opérandes sont pris dans la classe. ▲

Théorème 4.45. La classe FINI est fermée pour l'union.

Preuve. Il suffit de voir que l'union de deux langages finis est un langage fini. ■

Théorème 4.46. La classe FINI est fermée pour l'intersection.

Preuve. L'intersection de deux langages finis donne toujours un langage fini. ■

Théorème 4.47. La classe FINI n'est pas fermée pour l'opération complément.

Preuve. On a : $\emptyset \in \mathsf{FINI}$, mais $\overline{\emptyset} = \Sigma^* \notin \mathsf{FINI}$. ■

Remarque 4.48. Le complément d'un langage fini, quel qu'il soit, ne donne jamais un langage fini. ▲

Résumé

Concept	Définition	Peut être vide ?	Peut être infini ?	Exemple
symbole	—	—	—	`a`
alphabet	ensemble de symboles	non	non	{`a`, `b`, `c`}
mot	suite de symboles	oui (ε)	non	`abaac`
langage	ensemble de mots	oui ($\emptyset$)	oui	{ε, `ba`, `bba`, `bbba`, ...}
classe	ensemble de langages	oui (inintéressant)	oui (toutes les intéressantes sont infinies)	REG

Automates

Définition 4.49. Un automate est un quintuplet $(Q, \Sigma, \delta, q_0, F)$ où :

- Q est un ensemble fini d'états ;
- Σ est un alphabet ;
- $\delta : Q \times \Sigma \to Q$ est la fonction de transition ;
- $q_0 \in Q$ est l'état initial ;
- $F \subseteq Q$ est l'ensemble états acceptants.

▲

Exemple 4.50. Soit

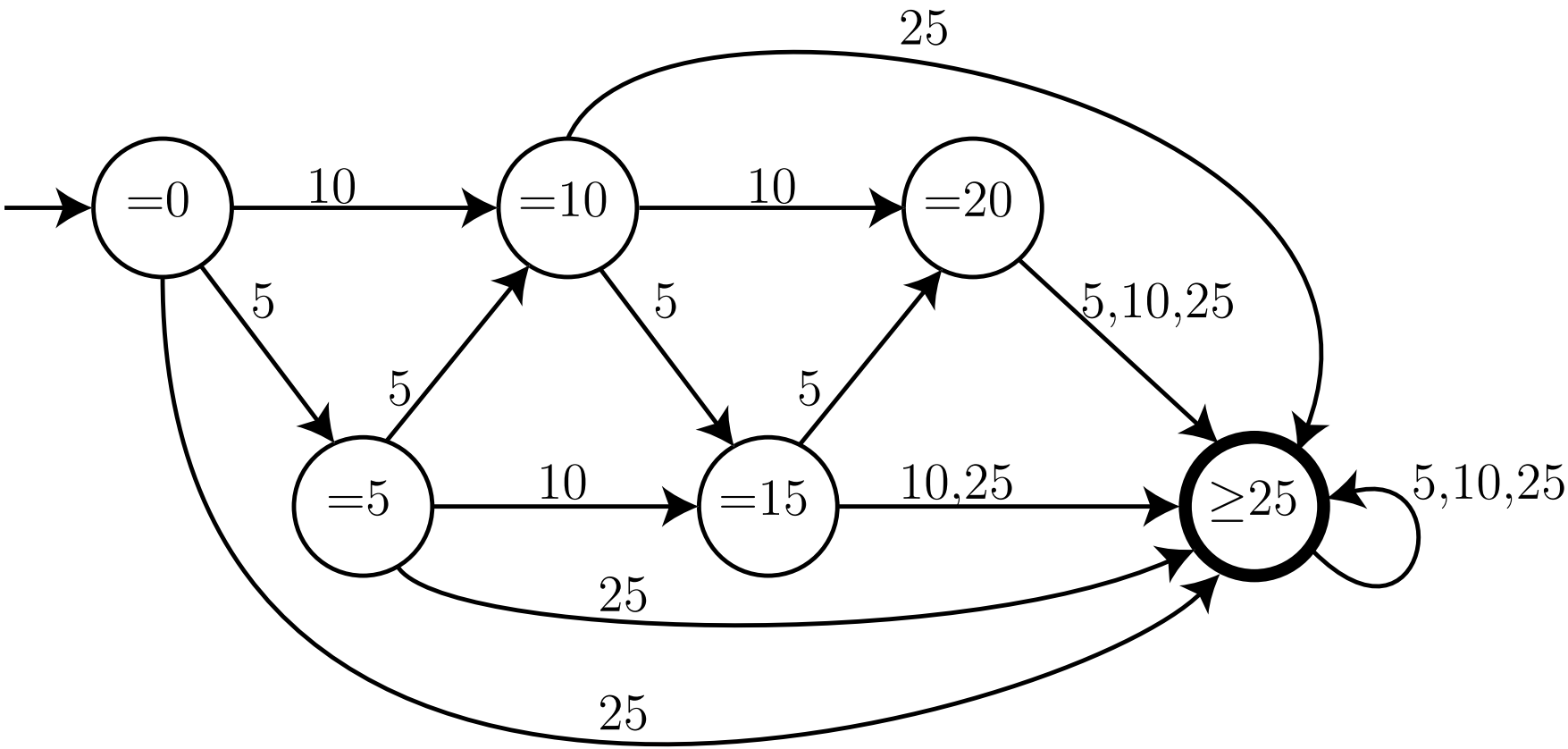
$$M = (Q, \Sigma, \delta, q_0, F)$$

où

$$\begin{aligned} Q &= \{\langle = 0\rangle, \langle = 5\rangle, \langle = 10\rangle, \langle = 15\rangle, \langle = 20\rangle, \langle \geq 25\rangle\}, \\ \Sigma &= \{\langle 5\rangle, \langle 10\rangle, \langle 25\rangle\}, \\ q_0 &= \langle = 0\rangle, \\ F &= \{\langle \geq 25\rangle\}, \end{aligned}$$

et où la fonction δ de transition est donnée par le diagramme de la figure 4.1.

Figure 4.1.



Le langage

$$\{p_1 \cdot p_2 \cdots p_k \mid \Sigma_{i=1}^{k} V(p_i) \geq 25\}.$$

est l'ensemble des mots qui amènent l'automate M de son état initial à l'état $\langle \geq 25 \rangle$. ▲

Définition 4.51. Soit $M = (Q, \Sigma, \delta, q_0, F)$ un automate.

Une configuration de l'automate M est une paire $[w, q]$ où $w \in \Sigma^*$ et $q \in Q$. ▲

Définition 4.52. Soit $M = (Q, \Sigma, \delta, q_0, F)$ un automate.

Si $\mathtt{a} \in \Sigma$ et $\delta(q, \mathtt{a}) = q'$ alors

$$[\mathtt{a} \cdot w, q] \vdash [w, q']$$

dénote une transition de l'automate M de la configuration $[\mathtt{a} \cdot w, q]$ à la configuration $[w, q']$. ▲

Définition 4.53. Soit $M = (Q, \Sigma, \delta, q_0, F)$ un automate.

Pour $x \in \Sigma^*$ et $w \in \Sigma^*$,

$$[x \cdot w, q] \overset{n}{\vdash} [w, q']$$

dénote n transitions successives amenant l'automate M de la configuration $[x \cdot w, q]$ à la configuration $[w, q']$, et

$$[x \cdot w, q] \overset{*}{\vdash} [w, q']$$

indique que

$$[x \cdot w, q] \overset{n}{\vdash} [w, q']$$

pour $n = |x|$. ▲

Exemples 4.54.

$$
\begin{aligned}
[\langle 10\rangle\langle 5\rangle\langle 10\rangle, \langle = 0\rangle] \quad &\vdash \quad [\langle 5\rangle\langle 10\rangle, \langle = 10\rangle] \\
&\vdash \quad [\langle 10\rangle, \langle = 15\rangle] \\
&\vdash \quad [\varepsilon, \langle \geq 25\rangle]
\end{aligned}
$$

$$
\begin{aligned}
[\langle 25\rangle\langle 10\rangle, \langle = 0\rangle] \quad &\vdash \quad [\langle 10\rangle, \langle \geq 25\rangle] \\
&\vdash \quad [\varepsilon, \langle \geq 25\rangle]
\end{aligned}
$$

$$
\begin{aligned}
[\langle 5\rangle\langle 5\rangle\langle 5\rangle, \langle = 0\rangle] \quad &\vdash \quad [\langle 5\rangle\langle 5\rangle, \langle = 5\rangle] \\
&\vdash \quad [\langle 5\rangle, \langle = 10\rangle] \\
&\vdash \quad [\varepsilon, \langle = 15\rangle]
\end{aligned}
$$

▲

Définition 4.55. Soit $M = (Q, \Sigma, \delta, q_0, F)$ un automate.

On dira que l'automate M accepte le mot $w \in \Sigma^*$ si $\exists q \in F$ tel que

$$[w, q_0] \overset{*}{\vdash} [\varepsilon, q].$$

▲

Définition 4.56. Soit $M = (Q, \Sigma, \delta, q_0, F)$ un automate.

L'ensemble

$$L(M) = \{w \in \Sigma^* \mid M \text{ accepte } w\}$$

est le langage reconnu par l'automate M. ▲

Définition 4.57. Un langage K est régulier si il existe un automate M tel que $K = L(M)$. ▲

Définition 4.58. La classe des langages réguliers est notée par REG. ▲

Exemple 4.59. Soit

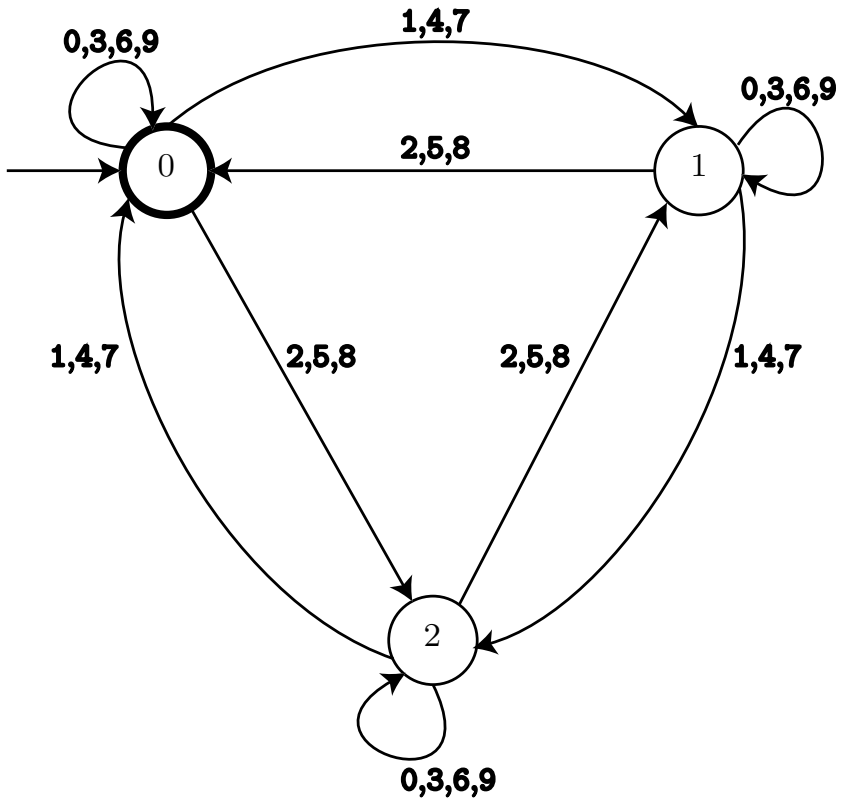
$$M = (Q, \Sigma, \delta, q_0, F)$$

où

$$\begin{aligned} Q &= \{\langle 0\rangle, \langle 1\rangle, \langle 2\rangle\}, \\ \Sigma &= \{\mathtt{0}, \mathtt{1}, \mathtt{2}, \mathtt{3}, \mathtt{4}, \mathtt{5}, \mathtt{6}, \mathtt{7}, \mathtt{8}, \mathtt{9}\}, \\ q_0 &= \langle 0\rangle, \\ F &= \{\langle 0\rangle\}, \end{aligned}$$

et où la fonction δ de transition est donnée par le diagramme de la figure 4.2.

Figure 4.2.



▲

Exemple 4.60.

$$
\begin{aligned}
[68725, \langle 0 \rangle] &\vdash [8725, \langle 0 \rangle] \\
&\vdash [725, \langle 2 \rangle] \\
&\vdash [25, \langle 0 \rangle] \\
&\vdash [5, \langle 2 \rangle] \\
&\vdash [\varepsilon, \langle 1 \rangle]
\end{aligned}
$$

Donc $68725 \notin L(M)$. ▲

Proposition 4.61. L'automate de l'exemple 4.59 reconnaît le langage contenant le mot vide et les chaînes de chiffres décimaux qui forment un nombre divisible par 3.

Preuve. L'inspection du diagramme de la fonction de transition δ, révèle que M accepte ε de même que les nombres dont la somme des chiffres est congrue à 0 modulo 3.

Or, si le développement décimal de l'entier n est $d_i d_{i-1} \ldots d_0$, on a :

$$
\begin{aligned}
& n \equiv 0 \pmod 3 \\
\Leftrightarrow \quad & \sum_{j=0}^{i} d_j 10^j \equiv 0 \pmod 3 \\
\Leftrightarrow \quad & \sum_{j=0}^{i} d_j \equiv 0 \pmod 3 \quad (\text{car } 10^j \equiv 1)
\end{aligned}
$$

■

Théorème 4.62. La classe REG est fermée pour l'opération complément.

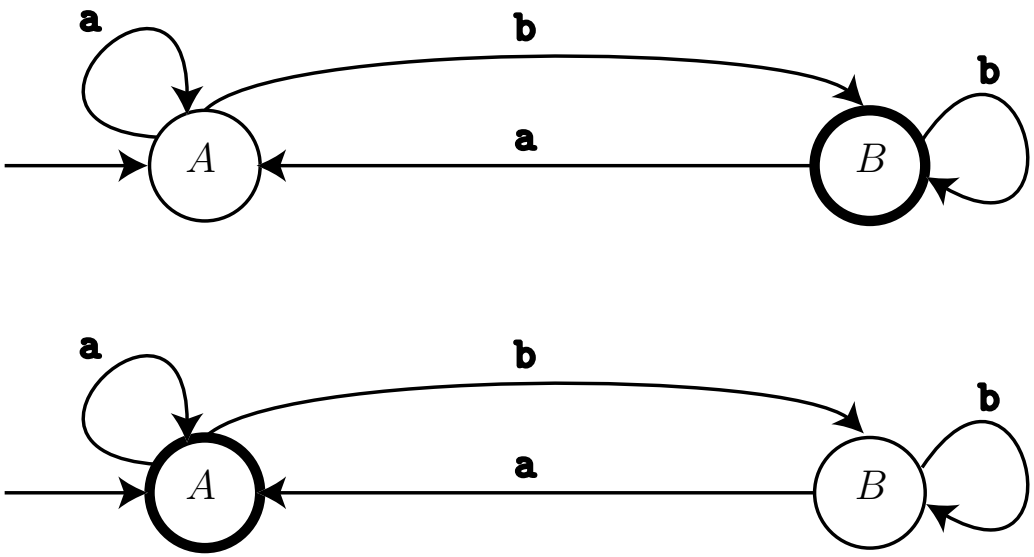
Preuve. Si $K \in \mathsf{REG}$, alors il est reconnu par un automate M.

Soit M' l'automate construit comme M, mais où les états acceptants et non-acceptants ont été interchangés.

Alors M' reconnaît $\overline{K}$. Donc $\overline{K} \in \mathsf{REG}$ et la classe est fermée pour l'opération.

■

Figure 4.3.



Théorème 4.63. La classe REG est fermée pour l'intersection.

Preuve. Soient $K_1 \in \mathsf{REG}$ et $K_2 \in \mathsf{REG}$ deux langages réguliers reconnus respectivement par les automates M_1 et M_2 :

$$\begin{aligned} M_1 &= (Q_1,\ \Sigma,\ \delta_1,\ q_{0,1},\ F_1) \\ M_2 &= (Q_2,\ \Sigma,\ \delta_2,\ q_{0,2},\ F_2). \end{aligned}$$

Considérons l'automate M suivant :

$$M = (Q_1 \times Q_2,\ \Sigma,\ \delta,\ (q_{0,1}, q_{0,2}),\ F_1 \times F_2),$$

où

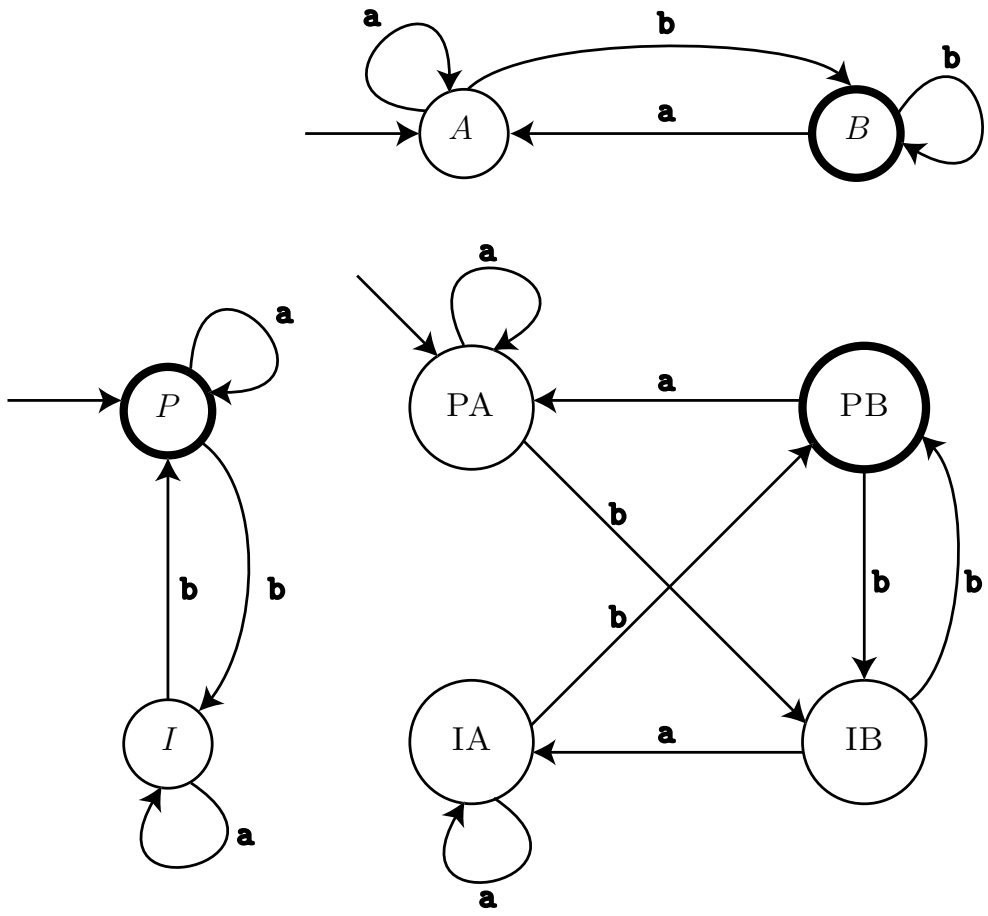
$$\delta(q_1, q_2, x) = (q_1', q_2')$$

si, et seulement si

$$\delta_1(q_1, x) = q_1' \qquad \text{et} \qquad \delta_2(q_2, x) = q_2',$$

comme dans l'exemple de la figure 4.4.

Figure 4.4.



Soit $w \in K_1 \cap K_2$.

Alors w amène M_1 de $q_{0,1}$ à un état final $f_1 \in F_1$ et amène également M_2 de $q_{0,2}$ à $f_2 \in F_2$.

Par construction, le mot w amène l'automate M de $(q_{0,1}, q_{0,2})$ à $(f_1, f_2) \in F_1 \times F_2$.

Par conséquent, M accepte w, et $w \in L(M)$.

Inversement, si $f_1 \notin F_1$ ou $f_2 \notin F_2$, alors $(f_1, f_2) \notin F_1 \times F_2$, et M n'accepte pas w.

On a donc $L(M) = K_1 \cap K_2$, et $K_1 \cap K_2 \in \mathsf{REG}$. ■

Théorème 4.64. La classe REG est fermée pour l'union.

Preuve. Soient $K_1 \in \mathsf{REG}$ et $K_2 \in \mathsf{REG}$ deux langages réguliers.

On a :

$$K_1 \cup K_2 = \overline{\overline{K_1} \cap \overline{K_2}}.$$

Comme la classe REG est fermée pour le complément et l'intersection, par les théorèmes 4.62 et 4.63, alors les langages $\overline{K_1}$, $\overline{K_2}$, $\overline{K_1} \cap \overline{K_2}$, et enfin $\overline{\overline{K_1} \cap \overline{K_2}}$ sont réguliers.

On a donc $K_1 \cup K_2 \in \mathsf{REG}$. ■

Théorème 4.65.

$$\mathsf{FINI} \subseteq \mathsf{REG}$$

Preuve. Tout d'abord, il est facile de voir que l'automate

$$(\{q_0\},\ \Sigma,\ \delta,\ q_0,\ \emptyset)$$

où

$$\delta(q_0, x) = q_0, \quad \forall x \in \Sigma$$

reconnaît le langage vide.

Ensuite, on remarque qu'un automate accepte le mot vide, si, et seulement si son état initial est acceptant.

Nous allons maintenant considérer les langages finis, non vides, qui ne contiennent pas le mot vide.

Soit $L = \{w_1, \ldots, w_k\}$ un langage fini de k mots, $k \geq 1$, $\varepsilon \notin L$, avec

$$\begin{aligned} w_1 &= x_{1,1}\, x_{1,2}\, \ldots\, x_{1,l_1} \\ w_2 &= x_{2,1}\, x_{2,2}\, \ldots\, x_{2,l_2} \\ &\vdots \\ w_k &= x_{k,1}\, x_{k,2}\, \ldots\, x_{k,l_k} \end{aligned}$$

où $l_i = |w_i|$ et $x_{i,j} \in \Sigma$.

Soit

$$l = \max_{1 \leq i \leq k} l_i$$

la longueur maximale des mots de L.

Considérons l'automate suivant :

$$M = (Q, \Sigma, \delta, q_0, F)$$

où

$$\begin{aligned} Q &= \{\langle \textsc{Init} \rangle\} \cup \Sigma \cup (\Sigma \times \Sigma) \cup \cdots \cup \Sigma^l \cup \{\langle \textsc{Puits} \rangle\} \\ q_0 &= \langle \textsc{Init} \rangle \\ F &= \{\langle x_{1,1}\, x_{1,2} \ldots x_{1,l_1} \rangle,\ \langle x_{2,1}\, x_{2,2} \ldots x_{2,l_2} \rangle,\ \ldots,\ \langle x_{k,1}\, x_{k,2} \ldots x_{k,l_k} \rangle\} \end{aligned}$$

et où la fonction δ de transition est donnée par :

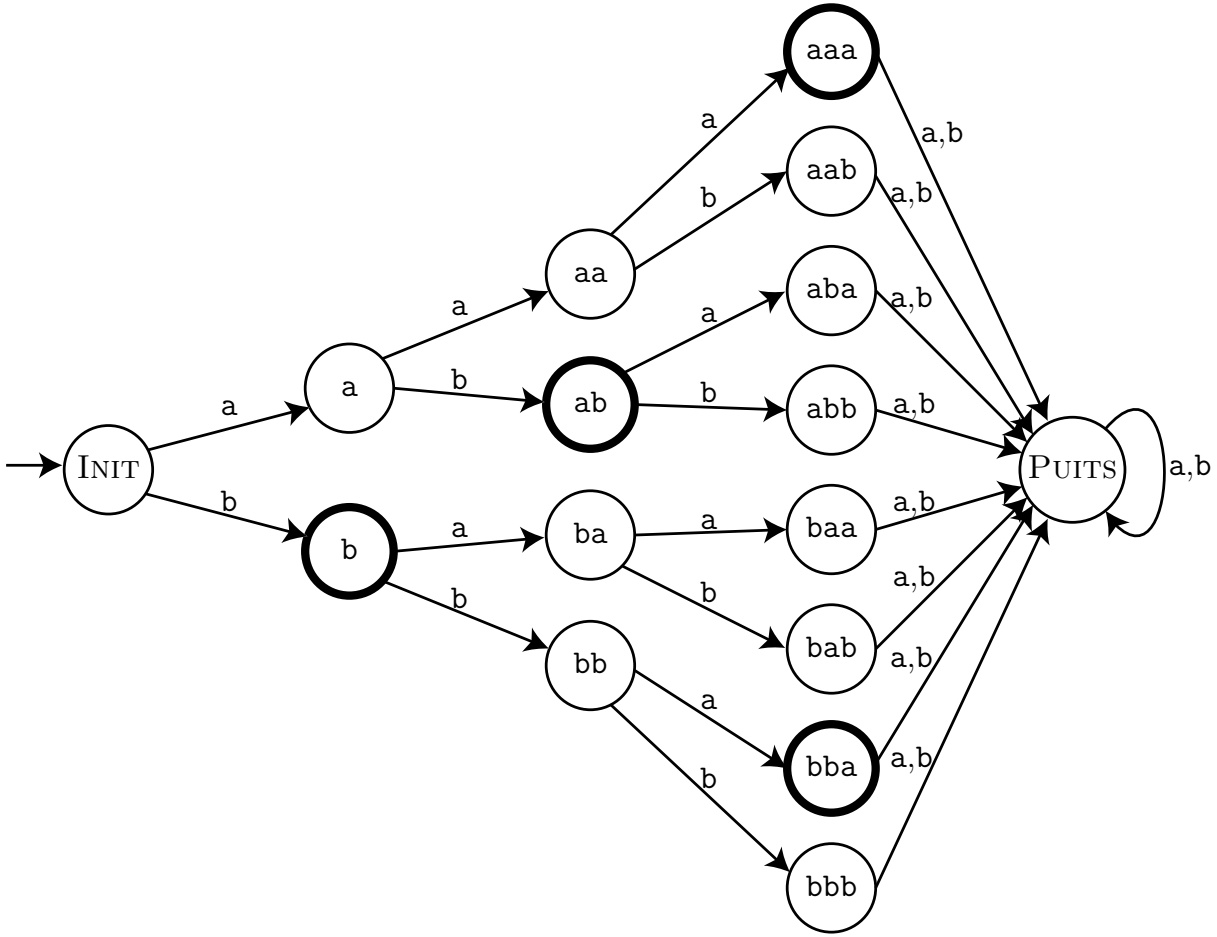
$\forall z,\ z_1,\ z_2,\ \ldots,\ z_l \in \Sigma\ :$

$$
\begin{aligned}
\delta(\langle \textsc{Init} \rangle,\ z) &= \langle z \rangle \\
\delta(\langle z_1\ z_2\ \ldots\ z_h \rangle,\ z) &= \langle z_1\ z_2\ \ldots\ z_h\ z \rangle, \quad \text{si } h < l \\
\delta(\langle z_1\ z_2\ \ldots\ z_l \rangle,\ z) &= \langle \textsc{Puits} \rangle \\
\delta(\langle \textsc{Puits} \rangle,\ z) &= \langle \textsc{Puits} \rangle
\end{aligned}
$$

comme dans l'exemple de la figure 4.5.

Figure 4.5.

$$\Sigma = \{\mathtt{a},\ \mathtt{b}\} \quad L = \{\mathtt{b},\ \mathtt{ab},\ \mathtt{aaa},\ \mathtt{bba}\} \quad k = 4 \quad l = 3$$

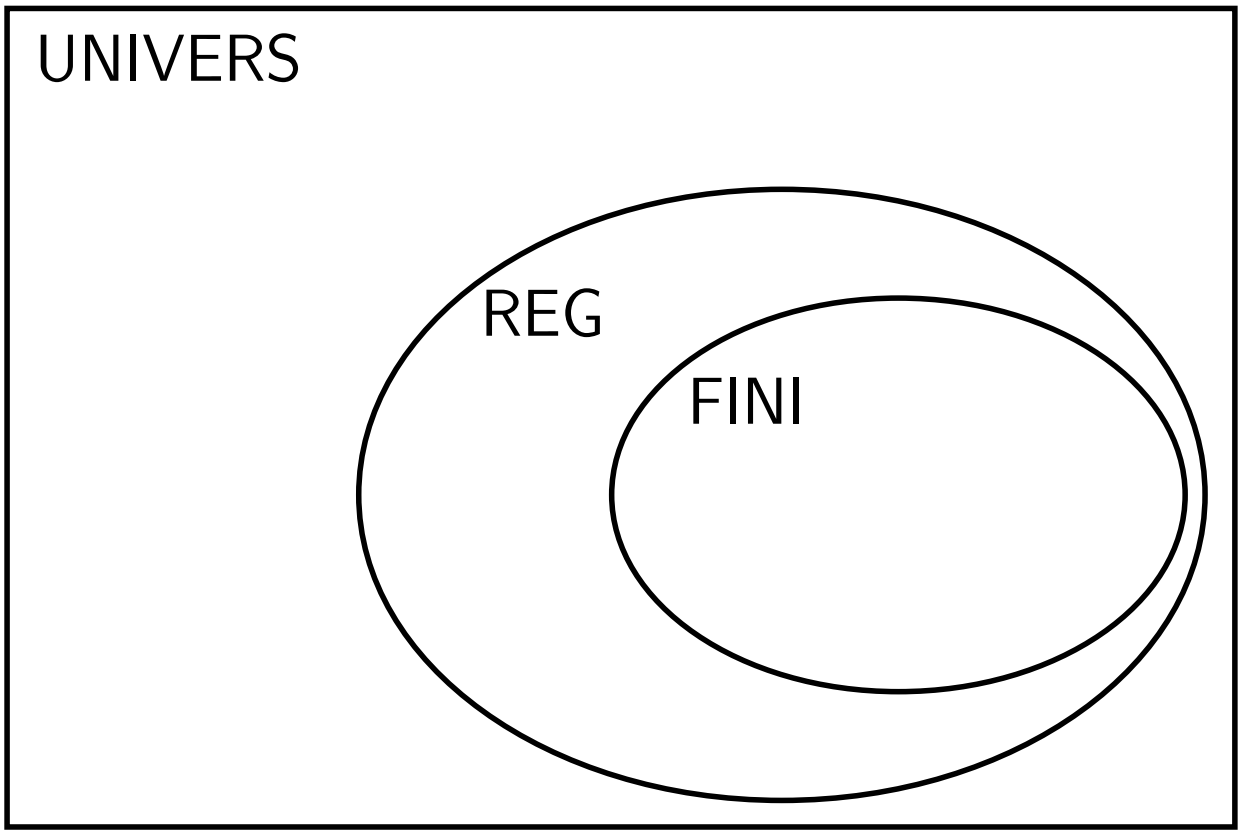


L'automate ainsi construit acceptera les mots du langage L et uniquement ces mots-là.

Donc $L \in \mathsf{REG}$, et $\mathsf{FINI} \subseteq \mathsf{REG}$. ■

Hiérarchie

Figure 4.6.



Pour prouver qu'un langage L est régulier

On peut...

- construire un automate qui reconnaît L,
- utiliser les propriétés de fermeture de la classe REG,
- montrer que L est fini.

Théorème 4.66 (*lemme du pompiste*). Soit $L \in \mathsf{REG}$. Il existe un entier $p \geq 1$ tel que pour tout $w \in L$ avec $|w| \geq p$ on peut écrire $w = x \cdot y \cdot z$ où

- $|y| > 0$,
- $|x \cdot y| \leq p$,
- $\forall i \geq 0 \ : \ x \cdot y^i \cdot z \in L$.

Preuve. Soit $M = (Q, \Sigma, \delta, q_0, F)$ un automate qui reconnaît le langage L.

Choisissons $p = \#Q$.

Soit $w \in L$ tel que $|w| \geq p$.

L'automate M va passer par au moins $p + 1$ états avant d'accepter w.

Par le principe du pigeonnier, il existe un état par lequel M va passer au moins deux fois, parmi les $p + 1$ premiers états visités.

Soit $r \in Q$ le premier état par lequel M passe deux fois.

Soit x la partie de w qui est lue entre l'état initial q_0 et la première visite de r.

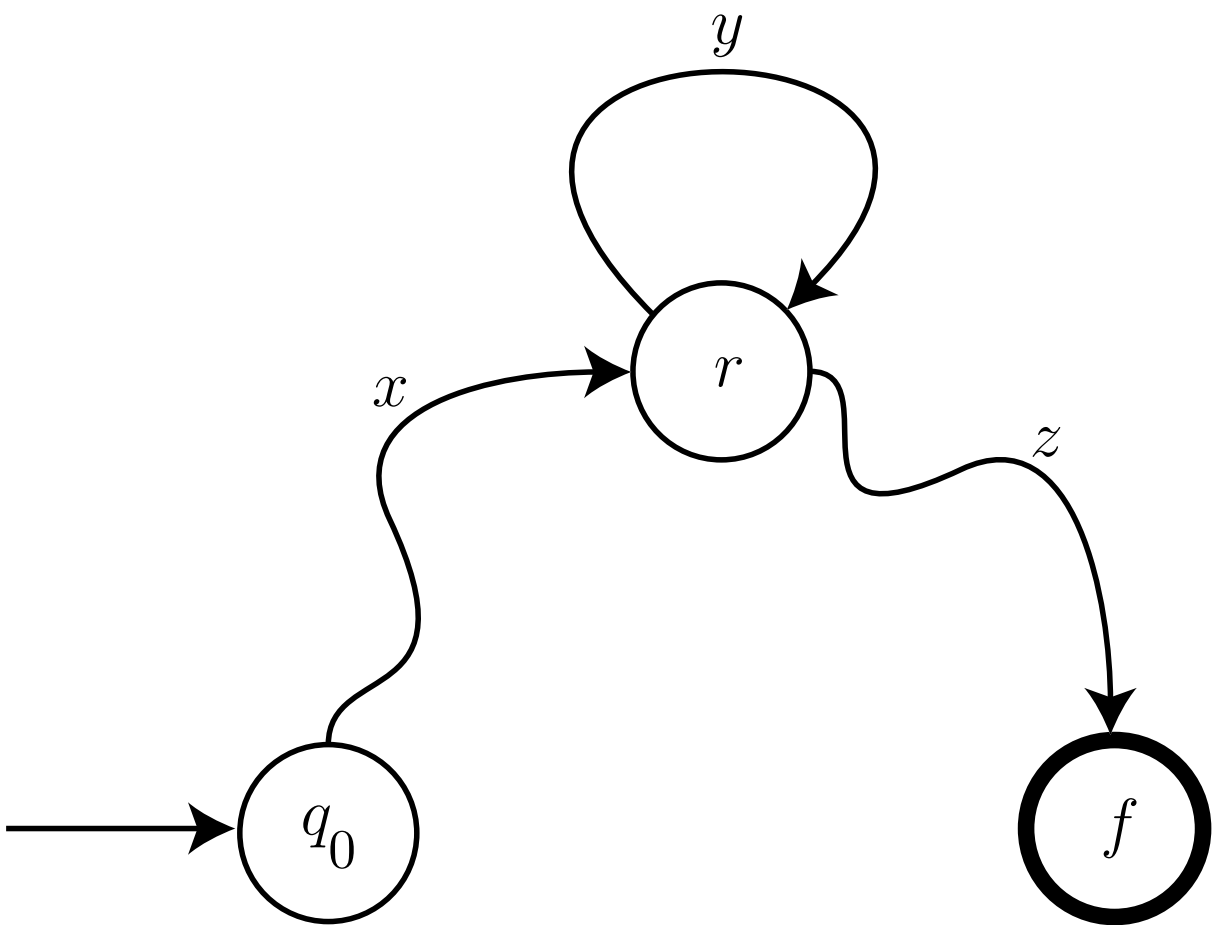
Soit y la partie de w lue entre les deux premières visites de l'état r.

Soit z la partie restante de w après $x \cdot y$. On a donc :

$$\begin{aligned} w &= x \cdot y \cdot z, \\ |y| &> 0, \\ |x \cdot y| &\leq p, \end{aligned}$$

comme illustré dans la figure 4.7.

Figure 4.7.



Il est maintenant clair que les mots suivants seront acceptés par M :

$$x \cdot z$$

$$x \cdot y \cdot z$$

$$x \cdot y \cdot y \cdot z$$

$$x \cdot y \cdot y \cdot y \cdot z$$

$$\vdots$$

et donc tous ces mots appartiennent à L. ■

Pour prouver qu'un langage L n'est pas régulier

On peut...

- utiliser les propriétés de fermeture de la classe REG,
- appliquer le lemme du pompiste.

Exemple 4.67. On sait que le langage

$$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid \ |w|_{\mathtt{b}} \text{ est pair}\}$$

est régulier parce qu'un automate qui le reconnaît apparaît à la figure 4.4.

Cet automate a 2 états.

Par la preuve du lemme du pompiste, on peut prendre $p = 2$.

Considérons le mot $w = \mathtt{abb}$.

On a bien $w \in L$ et $|w| \geq p$.

Choisissons la décomposition $w = x \cdot y \cdot z$ où

$$\begin{aligned} x &= \varepsilon, \\ y &= \mathtt{a}, \\ z &= \mathtt{bb}. \end{aligned}$$

On a bien :

- $|y| > 0$,
- $|x \cdot y| \leq p$,
- $\mathtt{bb},\ \mathtt{abb},\ \mathtt{aabb},\ \mathtt{aaabb},\ \mathtt{aaaabb},\ \ldots \in L$.

▲

Théorème 4.68. Le langage

$$L = \{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}$$

n'est pas régulier.

Preuve. Supposons au contraire que $L \in \mathsf{REG}$, afin d'arriver à une contradiction.

Soit $p \geq 1$ tel que donné par le lemme du pompiste.

Considérons le mot $w = \mathtt{a}^p\mathtt{b}^p$.

On a bien $w \in L$ et $|w| = 2p \geq p$.

Soient $w = x \cdot y \cdot z$, $|y| > 0$, $|x \cdot y| \leq p$, tels que donnés par le lemme du pompiste.

Il est clair que y ne contient que des $\mathtt{a}$.

Considérons $i = 2$ dans l'expression $x \cdot y^i \cdot z$.

Ce mot contient plus de a que de b, et donc $x \cdot y^i \cdot z \notin L$, en contradiction avec l'énoncé du lemme.

Donc $L \notin \mathsf{REG}$. ■

Théorème 4.69. Le langage

$$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid \ |w|_{\mathtt{a}} = |w|_{\mathtt{b}}\}$$

n'est pas régulier.

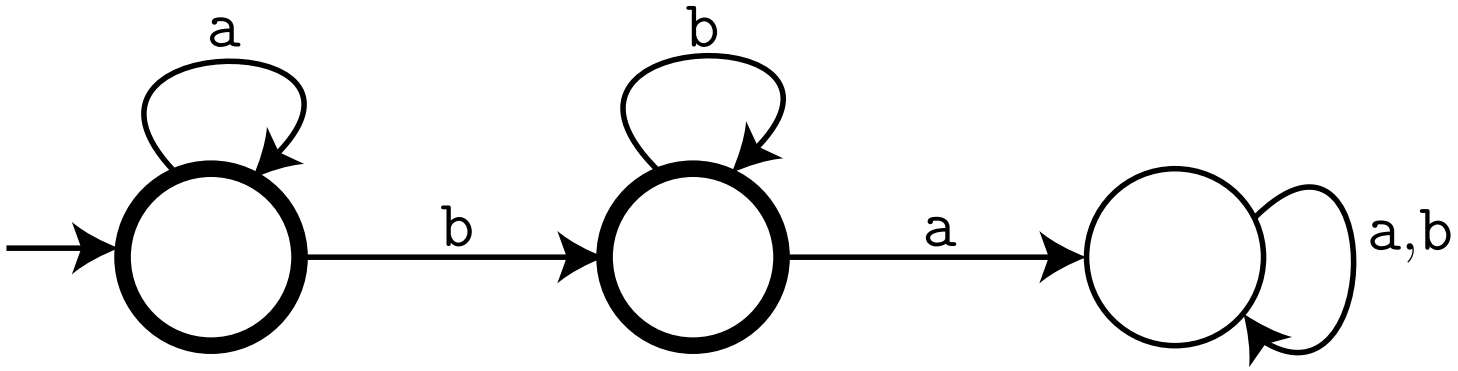
Preuve. Supposons au contraire que $L \in \mathsf{REG}$, afin d'arriver à une contradiction.

Considérons le langage

$$R = \{\mathtt{a}^k\mathtt{b}^l \mid k \geq 0, l \geq 0\}.$$

Il est facile de voir que $R \in \mathsf{REG}$, car il est reconnu par l'automate de la figure 4.8.

Figure 4.8.



On a donc $L \cap R \in \mathsf{REG}$, par la fermeture de la classe REG par rapport à l'opération d'intersection (théorème 4.63).

Mais $L \cap R = \{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}$ qui n'est pas régulier par le théorème 4.68.

On a une contradiction et le théorème est montré. ■

Grammaires hors contexte

Exemple 4.70.

$$
\begin{array}{rcl}
\langle \text{PROGRAMME} \rangle & \to & \texttt{program}\langle \text{ID} \rangle; \\
 & & \langle \text{DECL} \rangle \\
 & & \langle \text{CORPS} \rangle \\
\langle \text{DECL} \rangle & \to & \texttt{var}\langle \text{LISTE VAR} \rangle \\
\langle \text{LISTE VAR} \rangle & \to & \langle \text{LISTE ID} \rangle : \langle \text{TYPE} \rangle; \\
\langle \text{LISTE ID} \rangle & \to & \langle \text{ID} \rangle \langle \text{AUTRES ID} \rangle \\
\langle \text{AUTRES ID} \rangle & \to & \varepsilon \mid , \langle \text{ID} \rangle \langle \text{AUTRES ID} \rangle \\
 & \vdots &
\end{array}
$$

▲

Définition 4.71. Une grammaire hors contexte (GHC) est un quadruplet (V, Σ, R, S) où :

- V est un ensemble fini de variables ;
- Σ est un alphabet, c'est-à-dire un ensemble non vide et fini de symboles appelés terminaux, $V \cap \Sigma = \emptyset$;
- R est un ensemble fini de règles de la forme $v \to z$ où $v \in V$ et $z \in (V \cup \Sigma)^*$;
- $S \in V$ est la variable initiale.

▲

Exemple 4.72.

$$G = (V, \Sigma, R, S)$$

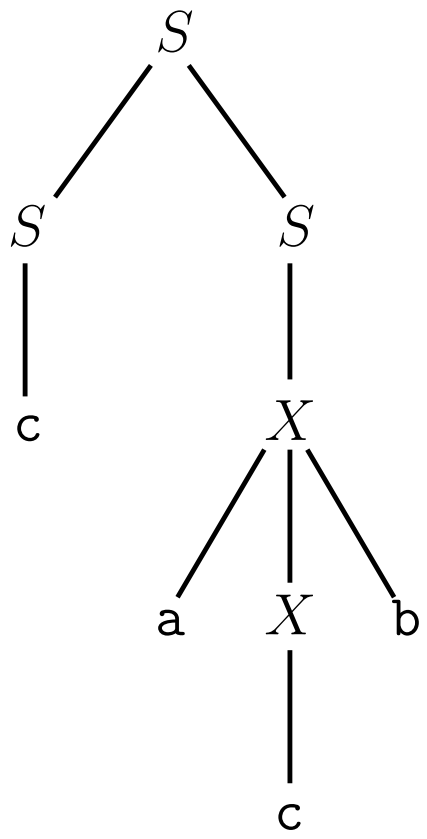
où :

$$\begin{aligned} V &= \{S, X\} \\ \Sigma &= \{\mathtt{a}, \mathtt{b}, \mathtt{c}\} \\ R &= \{S \to SS \mid X \mid \mathtt{c}, \quad X \to \mathtt{a}X\mathtt{b} \mid \mathtt{c}\} \end{aligned}$$

Dérivation du mot `cacb` :

$$
\begin{aligned}
S &\Rightarrow SS \\
&\Rightarrow \texttt{c}S \\
&\Rightarrow \texttt{c}X \\
&\Rightarrow \texttt{ca}X\texttt{b} \\
&\Rightarrow \texttt{cacb}
\end{aligned}
$$

Figure 4.9 (*arbre de dérivation*).



▲

Définition 4.73. Soit $G = (V, \Sigma, R, S)$ une GHC et soient $u, v, w \in (V \cup \Sigma)^*$.

On dit que uAv donne ou produit ou engendre uwv, noté

$$uAv \Rightarrow uwv,$$

si $A \to w \in R$.

On note $u \overset{n}{\Rightarrow} v$ si u donne v en n étapes :

$$u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \cdots \Rightarrow u_n = v.$$

Aussi, $u \overset{*}{\Rightarrow} v$ s'il existe un $n \geq 0$ tel que $u \overset{n}{\Rightarrow} v$. ▲

Définition 4.74. Soit $G = (V, \Sigma, R, S)$ une GHC.

$$L(G) = \{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\}$$

est le langage engendré par G. ▲

Définition 4.75. Un langage L est hors contexte si il existe une GHC G telle que $L = L(G)$. ▲

Définition 4.76. La classe des langages hors contexte est notée HC. ▲

Exemple 4.77. Soit $G = (\{S\}, \{\mathtt{a}, \mathtt{b}\}, R, S)$ une GHC où

$$R = \{S \to \mathtt{a}S\mathtt{b} \mid \varepsilon\}.$$

Alors

$$\begin{aligned} L(G) &= \{\varepsilon, \mathtt{ab}, \mathtt{aabb}, \mathtt{aaabbb}, \ldots\} \\ &= \{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}. \end{aligned}$$

Dérivation du mot aaabbb :

$$\begin{aligned} S &\Rightarrow \mathtt{a}S\mathtt{b} \\ &\Rightarrow \mathtt{aa}S\mathtt{bb} \\ &\Rightarrow \mathtt{aaa}S\mathtt{bbb} \\ &\Rightarrow \mathtt{aaabbb} \end{aligned}$$

▲

Exemple 4.78. Soit $G = (V, \Sigma, R, S)$ une GHC où

$$\begin{aligned} V &= \{S, M, N\} \\ \Sigma &= \{\mathtt{0}, \mathtt{1}, \ldots, \mathtt{9}, +, *, (,)\} \end{aligned}$$

$$R : \begin{cases} S & \to & N \mid (S{+}S) \mid (S{*}S) \\ N & \to & \mathtt{1}M \mid \mathtt{2}M \mid \ldots \mid \mathtt{9}M \\ M & \to & \varepsilon \mid \mathtt{0}M \mid \mathtt{1}M \mid \ldots \mid \mathtt{9}M \end{cases}$$

La variable N donne les nombres :

$$N \;\Rightarrow\; \mathtt{2}M \;\Rightarrow\; \mathtt{23}M \;\Rightarrow\; \mathtt{23}.$$

La variable S donne des expressions arithmétiques :

$$S \;\Rightarrow\; (S{+}S) \;\Rightarrow\; (S{+}(S{*}S)) \;\overset{*}{\Rightarrow}\; (N{+}(N{*}N)) \;\overset{*}{\Rightarrow}\; (\mathtt{123} + (\mathtt{245} * \mathtt{19}))$$

▲

Proposition 4.79. Soit $G = (\{S\}, \{\mathtt{a}, \mathtt{b}\}, R, S)$ une GHC où

$$R \ : \ S \to \mathtt{a}S\mathtt{b} \mid \mathtt{b}S\mathtt{a} \mid SS \mid \varepsilon,$$

et soit

$$L = \{w \mid \ |w|_{\mathtt{a}} = |w|_{\mathtt{b}}\},$$

alors

$$L(G) = L.$$

Preuve. Il est facile de voir que $L(G) \subseteq L$ car toutes les règles de G produisent un nombre égal de `a` et de `b`.

Il reste à montrer que $L \subseteq L(G)$.

Soit $w \in L$.

La preuve fonctionne par induction sur la longueur $n = |w|$.

Base de l'induction. Soit $n = 0$. Alors :

$$\begin{aligned} S &\Rightarrow \varepsilon \\ &= w, \end{aligned}$$

donc $w \in L(G)$.

Hypothèse d'induction. Pour $n > 0$, supposons que

$$\forall z \in L, |z| < n \ : \ S \stackrel{*}{\Rightarrow} z.$$

Pas d'induction. Remarquons que tous les mots de L ont un nombre pair de symboles.

Considérons 4 cas pour w :

1. $w = \mathtt{a}z\mathtt{b}$. Dans ce cas, on a la dérivation :

$$\begin{array}{lcll} S & \Rightarrow & \mathtt{a}S\mathtt{b} & \\ & \stackrel{*}{\Rightarrow} & \mathtt{a}z\mathtt{b} & \text{(par hypothèse d'induction)} \\ & = & w. & \end{array}$$

2. $w = \mathtt{b}z\mathtt{a}$:

$$\begin{array}{lcll} S & \Rightarrow & \mathtt{b}S\mathtt{a} & \\ & \stackrel{*}{\Rightarrow} & \mathtt{b}z\mathtt{a} & \text{(par hypothèse d'induction)} \\ & = & w. & \end{array}$$

3. $w = \mathtt{a}z\mathtt{a}$. Dans ce cas, z contient deux $\mathtt{b}$ de plus que de $\mathtt{a}$, et on peut donc le décomposer en $z = z_1 \cdot z_2$, où z_1 et z_2 contiennent chacun un $\mathtt{b}$ de plus que de $\mathtt{a}$. D'où :

$$
\begin{array}{rcl}
S & \Rightarrow & SS \\
 & \stackrel{*}{\Rightarrow} & \mathtt{a}z_1 S \quad (\text{car } \mathtt{a}z_1 \in L \text{ et par hypothèse d'induction}) \\
 & \stackrel{*}{\Rightarrow} & \mathtt{a}z_1 z_2 \mathtt{a} \quad (\text{car } z_2\mathtt{a} \in L \text{ et par hypothèse d'induction}) \\
 & = & \mathtt{a}z\mathtt{a} \\
 & = & w.
\end{array}
$$

4. $w = \mathtt{b}z\mathtt{b}$. Ce cas se traite comme le cas 3, *mutatis mutandis*.

Dans tous les cas, on a $w \in L(G)$. ■

Construction de grammaires

Construire une GHC pour un langage donné demande souvent de la créativité.

Cependant, il y a un certain nombre de techniques qui aident beaucoup. Entre autres :

- décomposer le langage en l'union de deux langages plus simples ;
- utiliser des règles qui produisent des listes ;

comme dans l'exemple qui suit.

Exemple 4.80. Soit

$$L = \{\mathtt{a}^i\mathtt{b}^j\mathtt{a}^k\mathtt{b}^l \mid (i = k \text{ ou } i = l),\ j \geq 0,\ k \geq 1,\ l \geq 1\},$$

c'est-à-dire :

$$\begin{aligned} L &= \{\mathtt{a}^i\mathtt{b}^j\mathtt{a}^i\mathtt{b}^l \mid i \geq 1,\ j \geq 0,\ l \geq 1\} \\ &\cup \{\mathtt{a}^i\mathtt{b}^j\mathtt{a}^k\mathtt{b}^i \mid i \geq 1,\ j \geq 0,\ k \geq 1\}. \end{aligned}$$

Une GHC pour L est donnée par :

$$G = (\{S, S_1, S_2, A, B, C, D\},\ \{\mathtt{a}, \mathtt{b}\},\ R,\ S)$$

avec

$$R\ :\ \begin{cases} S & \to & S_1 \mid S_2 \\ S_1 & \to & C\mathtt{b}B \\ C & \to & \mathtt{a}C\mathtt{a} \mid \mathtt{a}B\mathtt{a} \\ B & \to & \mathtt{b}B \mid \varepsilon \\ S_2 & \to & \mathtt{a}S_2\mathtt{b} \mid \mathtt{a}D\mathtt{b} \\ D & \to & B\mathtt{a}A \\ A & \to & \mathtt{a}A \mid \varepsilon \end{cases}$$

▲

Forme normale de Chomski

Définition 4.81. La GHC $G = (V, \Sigma, R, S)$ est sous forme normale de Chomski (FNC) si toutes ses règles sont d'une des formes

$$\begin{aligned} A &\to BC, \\ A &\to x, \end{aligned}$$

avec

$$\begin{aligned} A, B, C &\in V, \\ B &\neq S, \\ C &\neq S, \\ x &\in \Sigma, \end{aligned}$$

en permettant aussi

$$S \to \varepsilon.$$

▲

Théorème 4.82. Il existe un algorithme qui prend en entrée une GHC G et qui retourne une grammaire FNC G' telle que $L(G) = L(G')$, et cet algorithme fonctionne dans un temps proportionnel à la taille de G, c'est-à-dire en temps linéaire.

Aperçu de la preuve. Nous nous limitons à expliciter l'algorithme sans démontrer formellement son bon fonctionnement ni son temps d'exécution linéaire.

Algorithme 4.1. Prendre en entrée : $G = (V, \Sigma, R, S)$, une GHC.

1. Ajouter S_0 à V, la nouvelle variable initiale, et ajouter la règle $S_0 \to S$ à R.

2. Éliminer de R toutes les règles de la forme $A \to \varepsilon$, sauf possiblement la règle $S_0 \to \varepsilon$:

   2.1 Enlever $A \to \varepsilon$.

   2.2 $\forall (B \to uAv) \in R$: ajouter $B \to uv$, mais seulement si $B \to uv$ n'a pas été éliminée.

3. Éliminer de R toutes les règles de la forme $A \to B$, où $B \in V$:

   3.1 Enlever $A \to B$.

   3.2 $\forall (B \to w) \in R$, ajouter $A \to w$, mais seulement si $A \to w$ n'a pas été éliminée.

4. Éliminer de R toutes les règles de la forme $A \to u_1 u_2 \dots u_k$ où $k \geq 3$ et où $u_1, u_2, \dots, u_k \in V \cup \Sigma$:
   - 4.1 Enlever $A \to u_1 u_2 \dots u_k$.
   - 4.2 Ajouter $A_1,\ A_2,\ \dots,\ A_{k-2}$ à V.
   - 4.3 Ajouter les règles $A \to u_1 A_1,\ A_1 \to u_2 A_2,\ \dots,\ A_{k-2} \to u_{k-1} u_k$.

5. Éliminer de R toutes les règles de la forme $A \to xB$ où $x \in \Sigma$ et $B \in V$:

   5.1 Enlever $A \to xB$.

   5.2 Ajouter A' à V.

   5.3 Ajouter les règles $A \to A'B$ et $A' \to x$.

   Faire de même pour les règles de la forme $A \to Bx$.

Produire en sortie la grammaire ainsi modifiée.

■

Exemple 4.83.

$$G = (\{S, A, B\}, \{\mathtt{a}, \mathtt{b}\}, R, S)$$

où

$$R \ : \ \begin{cases} S & \rightarrow & ASA \mid \mathtt{a}B \\ A & \rightarrow & B \mid S \\ B & \rightarrow & \mathtt{b} \mid \varepsilon \end{cases}$$

1. Ajouter une nouvelle variable initiale.

Avant :

$$\begin{aligned} S &\to ASA \mid \mathtt{a}B \\ A &\to B \mid S \\ B &\to \mathtt{b} \mid \varepsilon \end{aligned}$$

Après :

$$\begin{aligned} S_0 &\to S \\ S &\to ASA \mid \mathtt{a}B \\ A &\to B \mid S \\ B &\to \mathtt{b} \mid \varepsilon \end{aligned}$$

2. Éliminer les règles ε.

Avant :

$$\begin{aligned} S_0 &\to S \\ S &\to ASA \mid \mathtt{a}B \\ A &\to B \mid S \\ B &\to \mathtt{b} \mid \varepsilon \end{aligned}$$

Après :

$$\begin{aligned} S_0 &\to S \\ S &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ A &\to B \mid S \\ B &\to \mathtt{b} \end{aligned}$$

3. Éliminer les règles unitaires.

Avant :

$$\begin{aligned} S_0 &\to S \\ S &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ A &\to B \mid S \\ B &\to \mathtt{b} \end{aligned}$$

Après :

$$\begin{aligned} S_0 &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ S &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ A &\to \mathtt{b} \mid ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ B &\to \mathtt{b} \end{aligned}$$

4. Éliminer les règles à plus de deux symboles.

Avant :

$$S_0 \to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \qquad A \to \mathtt{b} \mid ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS$$
$$S \to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \qquad B \to \mathtt{b}$$

Après :

$$S_0 \to \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \mid AS_{0,1} \qquad A \to \mathtt{b} \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \mid AA_1$$
$$S_{0,1} \to SA \qquad A_1 \to SA$$
$$S \to \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \mid AS_1 \qquad B \to \mathtt{b}$$
$$S_1 \to SA$$

5. Éliminer les règles mixtes.

Avant :

$$S_0 \to \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \mid AS_{0,1} \qquad A \to \mathtt{b} \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \mid AA_1$$
$$S_{0,1} \to SA \qquad A_1 \to SA$$
$$S \to \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \mid AS_1 \qquad B \to \mathtt{b}$$
$$S_1 \to SA$$

Après :

$$S_0 \to S_0{}'B \mid \mathtt{a} \mid SA \mid AS \mid AS_{0,1} \qquad S_1 \to SA$$
$$S_0{}' \to \mathtt{a} \qquad A \to \mathtt{b} \mid A'B \mid \mathtt{a} \mid SA \mid AS \mid AA_1$$
$$S_{0,1} \to SA \qquad A' \to \mathtt{a}$$
$$S \to S'B \mid \mathtt{a} \mid SA \mid AS \mid AS_1 \qquad A_1 \to SA$$
$$S' \to \mathtt{a} \qquad B \to \mathtt{b}$$

▲

Remarque 4.84. L'exemple précédent peut être simplifié à la GHC suivante, également sous FNC :

$$
\begin{aligned}
S_0 &\to DB \mid \mathtt{a} \mid SA \mid AS \mid AC \\
S &\to DB \mid \mathtt{a} \mid SA \mid AS \mid AC \\
A &\to \mathtt{b} \mid DB \mid \mathtt{a} \mid SA \mid AS \mid AC \\
B &\to \mathtt{b} \\
C &\to SA \\
D &\to \mathtt{a}
\end{aligned}
$$

▲

Les remarques 4.85 et 4.86 sont cruciales pour la compréhension de l'algorithme 4.2.

Remarque 4.85. Si $G = (V, \Sigma, R, S)$ est sous FNC, et si $\varepsilon \in L(G)$, alors la seule façon de produire ε est

$$S \Rightarrow \varepsilon.$$

▲

Remarque 4.86. Si $G = (V, \Sigma, R, S)$ est sous FNC, si $x \in \Sigma$ et $x \in L(G)$, alors la seule façon de produire x est

$$A \Rightarrow x$$

où $A \in V$. ▲

Appartenance d'un mot à un langage hors contexte

Théorème 4.87. Il existe un algorithme qui prend en entrée un mot w et une GHC G et qui détermine si $w \in L(G)$. Pour une grammaire G fixée, l'algorithme fonctionne en temps $O(n^3)$ où n est la taille du mot donné.

Aperçu de la preuve. Nous donnons l'algorithme et argumentons pour son temps d'exécution.

Algorithme 4.2. Prendre en entrée : $G = (V, \Sigma, R, S)$, une GHC, et $w \in \Sigma^*$.

Soit $w = w_1 w_2 \dots w_n$, avec $w_1, \dots, w_n \in \Sigma$.

1. Mettre G sous FNC à l'aide de l'algorithme 4.1.

   Soit $G' = (V', \Sigma, R', S_0)$ cette nouvelle grammaire.

2. Si $w = \varepsilon$, alors :

   si $(S_0 \to \varepsilon) \in R'$, alors retourner $\langle \text{VRAI} \rangle$,

   sinon retourner $\langle \text{FAUX} \rangle$.

3. $n \geq 1$.

   Pour $1 \leq i \leq j \leq n$, soit $f(i, j)$ l'ensemble des variables dans V' qui engendrent le sous-mot $w_i \ldots w_j$.

   Calculer $f(1, n)$ en appelant la sous-routine décrite à l'étape 4.

   Si $S_0 \in f(1, n)$, alors retourner $\langle$VRAI$\rangle$, sinon retourner $\langle$FAUX$\rangle$.

4. Calcul de $f(i,j)$, pour $1 \leq i \leq j \leq n$ donnés.

   Conserver les résultats obtenus au fur et à mesure dans une cache. Si $f(i,j)$ a déjà été calculé, alors retourner le résultat tout de suite, sans faire de calculs.

   Si $i = j$, alors $f(i,i) \leftarrow \{A \in V' \mid (A \to w_i) \in R'\}$.

   Si $i < j$, alors :

   $F \leftarrow \emptyset$,

   pour $k \leftarrow i$ à $j-1$ faire :

   $F_g \leftarrow f(i,k)$,

   $F_d \leftarrow f(k+1,j)$,

   $F \leftarrow F \cup \{A \in V' \mid \exists B \in F_g,\ \exists C \in F_d\ :\ (A \to BC) \in R'\}$,

   $f(i,j) \leftarrow F$.

Pour une grammaire G fixée, et si la grammaire G' est calculée une fois pour toutes, le temps d'exécution de l'algorithme 4.2 sur un mot de longueur n est dominé par le nombre d'appels à la fonction f de l'étape 4.

La fonction récursive f est équivalente à un algorithme de programmation dynamique qui consiste à remplir une table de taille $O(n^2)$ où chaque élément prend un temps $O(n)$ à calculer. Le temps d'exécution total est dans $O(n^3)$. ■

Théorème 4.88.

$$\mathsf{REG} \subseteq \mathsf{HC}$$

Preuve. Soit le langage $L \in \mathsf{REG}$ et soit $M = (Q, \Sigma, \delta, q_0, F)$ un automate qui le reconnaît.

Alors il est facile de voir que la GHC $G = (V, \Sigma, R, S)$ où

$$\begin{aligned} V &= Q, \\ R &= \{(A \to xB) \mid A \in Q,\ x \in \Sigma,\ B \in Q,\ \delta(A, x) = B\} \\ &\quad \cup \{(A \to \varepsilon) \mid A \in F\}, \\ S &= q_0, \end{aligned}$$

engendre L. ■

Corollaire 4.89. La classe REG est strictement incluse dans la classe HC.

Preuve. Soit $L = \{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}$.

Nous avons vu que $L \in$ HC à l'exemple 4.77, mais $L \notin$ REG par le théorème 4.68. ■

Langages non hors contexte

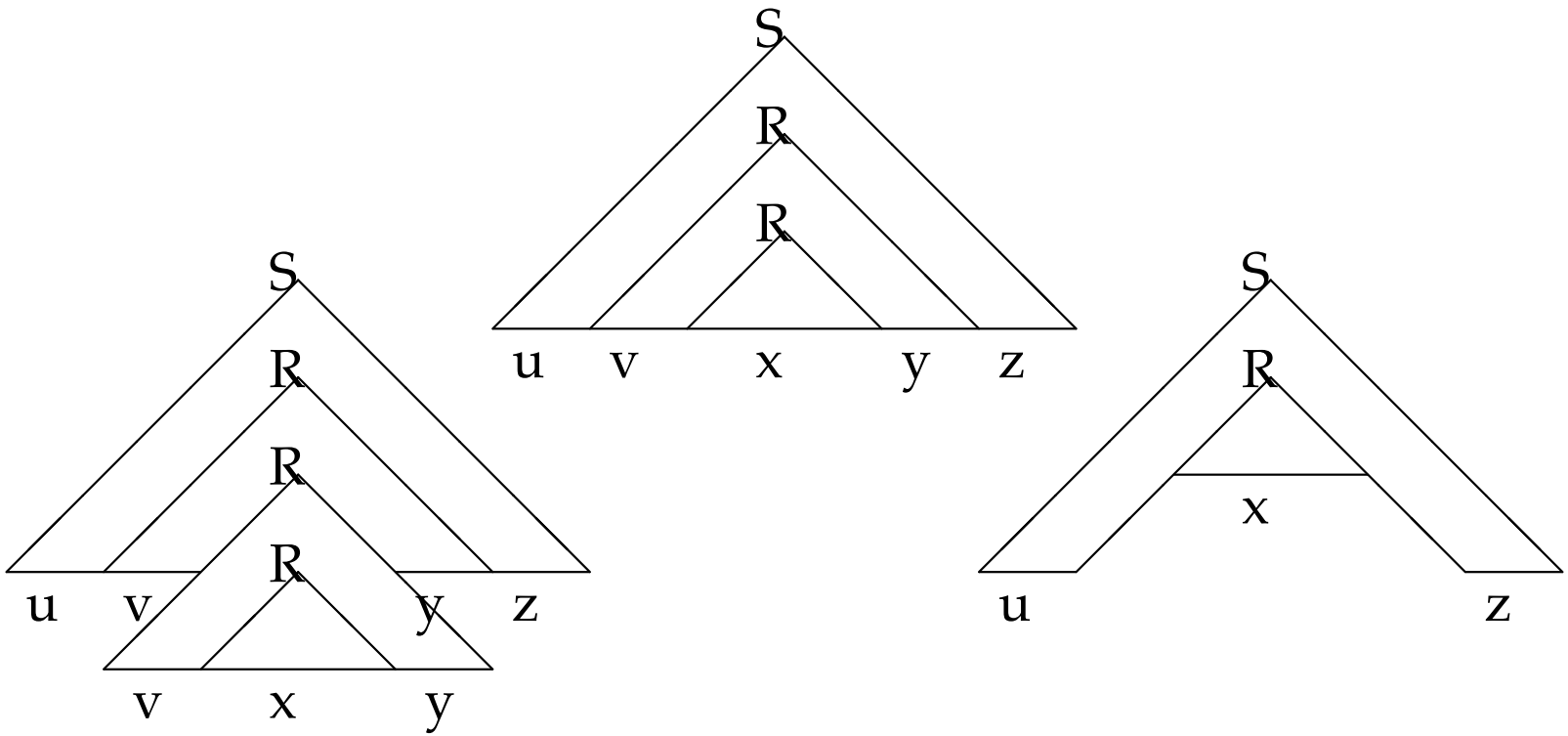
Théorème 4.90 (*lemme du pompiste pour les langages hors contexte*). Soit $L \in \mathsf{HC}$. Il existe un entier $p \geq 1$ tel que pour tout $w \in L$ avec $|w| \geq p$ on peut écrire $w = uvxyz$ où

- $|vy| > 0$,
- $|vxy| \leq p$,
- $\forall i \geq 0 \ : \ uv^i xy^i z \in L$.

Aperçu de la preuve.

Si un mot $w \in L$ est suffisamment long, alors il aura un arbre de dérivation très haut. Dans cet arbre, il y aura un chemin, du sommet jusqu'à la base, avec une répétition de variable. Voir la figure 4.10.

Figure 4.10.



■

Théorème 4.91. Le langage

$$L = \{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \mid n \geq 0\}$$

n'est pas hors contexte.

Preuve. Supposons au contraire que $L \in \mathsf{HC}$, afin d'arriver à une contradiction.

Soit $p \geq 1$ tel que donné par le lemme du pompiste pour les langages hors contexte.

Considérons le mot $w = \mathtt{a}^p\mathtt{b}^p\mathtt{c}^p$.

On a bien $w \in L$ et $|w| = 3p \geq p$.

Soient $w = uvxyz$, $|vy| > 0$, $|vxy| \leq p$, tels que donnés par le lemme.

Considérons les deux cas suivants :

1. La partie v, ou y, contient plus d'une sorte de symboles parmi `a`, `b` et `c`.

   Choisissons $i = 2$ dans l'expression uv^ixy^iz.

   Ce mot n'est pas de la forme $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$, et donc $uv^ixy^iz \notin L$, en contradiction avec l'énoncé du lemme.

2. La partie v ne contient qu'une seule sorte de symbole, de même que y.

   Choisissons $i = 0$ dans l'expression uv^ixy^iz, c'est-à-dire uxz.

   Ce mot contient pas un nombre égal des trois symboles et n'est pas de la forme $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$, et donc $uv^ixy^iz \notin L$, de nouveau une contradiction l'énoncé du lemme du pompiste.

Donc $L \notin \mathsf{HC}$. ■

Propriétés de fermeture

Théorème 4.92. La classe HC est fermée pour l'union.

Preuve. Soient

$$L_1 = L(G_1) \text{ où } G_1 = (V_1, \Sigma, R_1, S_1)$$

et

$$L_2 = L(G_2) \text{ où } G_2 = (V_2, \Sigma, R_2, S_2),$$

en supposant que V_1 et V_2 sont disjoints, de même que R_1 et R_2.

Alors la GHC suivante :

$$G = (V_1 \cup V_2 \cup \{S\},\ \Sigma,\ R_1 \cup R_2 \cup \{S \to S_1 \mid S_2\},\ S)$$

engendre le langage $L_1 \cup L_2$. ■

Théorème 4.93. La classe HC n'est pas fermée pour l'intersection.

Preuve. Soient

$$\begin{aligned} L_1 &= \{\mathtt{a}^m\mathtt{b}^n\mathtt{c}^n \mid m \geq 0, n \geq 0\}, \\ L_2 &= \{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^m \mid m \geq 0, n \geq 0\}. \end{aligned}$$

Alors

$$L_1 \cap L_2 = \{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \mid n \geq 0\}.$$

Or on a vu au théorème 4.91 que ce langage n'est pas hors contexte. ■

Théorème 4.94. La classe des langages hors contexte n'est pas fermée pour la complémentation.

Preuve. Supposons au contraire que la classe est fermée afin d'arriver à une contradiction.

Soient $L_1 \in \mathsf{HC}$ et $L_2 \in \mathsf{HC}$.

En considérant que

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$

et le théorème 4.92 (HC est fermée pour l'union), on obtient une contradiction avec le théorème 4.93 (HC n'est pas fermée pour l'intersection). ■

Hiérarchie

Figure 4.11.