Langages formels

Juin 2011

A. Grazon

Langages Formels

A. Grazon

Juin 1999

Je remercie mes collègues de l'IFSIC : Gilles Lesventes, pour m'avoir permis de choisir dans son polycopié de Grammaires et Sémantique de nombreux extraits qui sont reproduits dans ce document, et Martine Vergne pour avoir relu celui-ci avec le plus grand soin.

Ces notes de cours sont destinées aux étudiants de la licence d'informatique de l'Université de Rennes I. Elles leur permettent de retrouver les définitions et théorèmes de ce cours. Elles sont volontairement pauvres en exemples et preuves.
En aucun cas la lecture de ces notes ne peut se substituer à la présence des étudiants en cours.

Table des matières

Chapitre 1

Monoïde libre, langages

1.1 Préliminaires mathématiques

1.1.1 Relations

Etant donnés deux ensembles E et F, on appelle relation entre E et F tout sous-ensemble R du produit cartésien E × F. On note xRy pour (x,y)∈R.

Etant donnée une relation R⊆E×F, pour tout x appartenant à E, on note R(x) l'image de x par R, c'est à dire l'ensemble des y appartenant à F tels que xRy.

On appelle relation symétrique de R et on note R^{-1} la relation entre F et E définie comme l'ensemble des couples (y,x) tels que (x,y)∈R.

R est dite :

- injective si $\forall x_1 \in E,\ \forall x_2 \in E,\ x_1 \neq x_2 \ \Leftrightarrow\ R(x_1) \cap R(x_2) = \emptyset$.
- surjective si $\forall y \in F,\ \exists x \in E\ tel\ que\ xRy$.
- bijective si R est injective et surjective.

Une relation R sur un ensemble E est une partie de E×E. Une telle relation R est dite :

- réflexive si $\forall x \in E\ xRx$
- anti-réflexive si $\forall x, y \in E\ xRy\ \Rightarrow\ x \neq y$
- symétrique si $\forall x \in E\ \forall y \in E\ xRy\ \Rightarrow\ yRx$
- anti-symétrique si $\forall x \in E\ \forall y \in E\ xRy\ et\ yRx\ \Rightarrow\ x = y$
- transitive si $\forall x, y, z \in E\ (xRy\ et\ yRz)\ \Rightarrow\ xRz$

Une relation sur un ensemble E est une relation d'équivalence si elle est réflexive, symétrique et transitive.

Etant donnés une relation d'équivalence R sur un ensemble E et un élément x de E,

on appelle classe d'équivalence de x modulo R et on note $\bar{x}$ l'ensemble des éléments y de E tels que xRy. On appelle ensemble quotient de E par R et on note E/R l'ensemble de ces classes d'équivalence. Il est facile de voir que pour tous x et y de E, $\bar{x}$ et$\bar{y}$ sont soit égaux soit disjoints et que d'autre part $\cup_{x\in E}\bar{x}$=E. Les classes d'équivalence modulo une relation d'équivalence R sur E constituent une partition de E.

Une relation R sur un ensemble E est une relation d'ordre large si elle est réflexive, anti-symétrique et transitive. C'est une relation d'ordre strict si elle est anti-réflexive, anti-symétrique et transitive.

La fermeture réflexive d'une relation R$\subseteq$E$\times$E est R$\cup$R^0 où R^0=$\{$(x,x)$|$x$\in$E$\}$. La fermeture transitive d'une relation R$\subseteq$E$\times$E est $\cup_{k=1}^{\infty}$R^k où:

$$R^1 = R \; et \; \forall k > 1 \; R^k = \{(x,z) \in E \times E \mid \exists y \in E \; (x,y) \in R^{k-1}, \; (y,z) \in R^1\}$$

R^k est appelé l'exponentiation à l'ordre k de R.

La fermeture réflexive et transitive d'une relation R$\subseteq$E$\times$E est la fermeture réflexive de sa fermeture transitive ou la fermeture transitive de sa fermeture réflexive. Une relation R$_1$ est plus fine qu'une relation R$_2$ (R$_2$ est plus grossière que R$_1$) si R$_1$ $\subseteq$R$_2$ c'est-à-dire si pour tous x et y, xR$_1$y implique xR$_2$y. Les classes modulo R$_2$ sont alors des unions de classes modulo R$_1$.

1.1.2 Applications

Une relation R$\subseteq$E$\times$F est une fonction partielle de E dans F si R^{-1} est injective. C'est une application de E dans F si R^{-1} est injective et surjective.

Toute relation R$\subseteq$E$\times$F définit une application de E dans $\mathcal{P}$(F) (ensemble des parties de F) qui à tout x de E associe son image R(x). Cette application peut être confondue avec R dans le cas où R est une application.

Une application de E$\times$E dans E est appelée loi de composition interne ou opération sur E. Si op désigne une opération sur E, l'image par op d'un couple (x,y) est généralement notée x op y.

1.1.3 Congruences

Etant donné un ensemble E muni d'une opération $\diamond$, une relation d'équivalence R$\subseteq$E$\times$E est une congruence si pour tous x,y,z,t$\in$E, xRy et zRt implique x$\diamond$z R y$\diamond$t.

Sur l'ensemble E/R des classes d'équivalence modulo R, on définit l'opération $\bar{\diamond}$ par $\bar{x}\bar{\diamond}\bar{y} = \bar{z}$ si et seulement si $x \diamond y = z$.

Etant donnés un ensemble E et une équivalence R sur E, on appelle application canonique l'application φ de E dans E/R définie par $\varphi(x) = \bar{x}$ pour tout x dans E.

1.1.4 Quelques notations

Etant donnés deux ensembles E et F, on note E\F l'ensemble {x|x∈ E,x∉ F}.

On note [i,j] l'ensemble {k|i≤k≤j}. On note [j] l'ensemble [1,j]. [0] dénote donc l'ensemble vide.

1.1.5 Raisonnement par récurrence

Proposition 1 (Principe de récurrence)
Soit S une partie de N *telle que :*

1. $0 \in S$
2. $\forall n \in S,\ n+1 \in S$

alors $S = \mathrm{N}$

En appliquant ce principe à $S = \{n \in \mathrm{N} \mid P(n)\ est\ vraie\}$, où P est une propriété définie sur les entiers, on obtient le corollaire suivant qui est le fondement du raisonnement par récurrence :

Théorème 2 *Soit P une propriété définie sur* N*, et soit a un entier naturel donné. Si*

1. *$P(a)$ est vraie (on notera cela simplement $P(a)$)*
2. *pour tout entier naturel $n \geq a$, $P(n) \Rightarrow P(n+1)$ est vraie*

alors $P(n)$ est vraie pour tout entier naturel n supérieur ou égal à a.

Par la suite, on utilisera bien souvent un corollaire de ce théorème, dans lequel on suppose une hypothèse plus forte que $P(n)$, à savoir "$P(i)$ est vraie pour tous les entiers i compris entre a et n" :

Théorème 3 (Récurrence forte) *Soit P une propriété définie sur* N*, et soit a un entier naturel donné. Si*

1. *$P(a)$ est vraie*
2. *pour tout entier $n \geq a$, $(P(a) \wedge P(a+1) \ldots \wedge P(n)) \Rightarrow P(n+1)$*

alors $P(n)$ est vraie pour tout entier naturel n supérieur ou égal à a.

Remarque :
Il faut prendre les précautions suivantes lors de la construction d'une preuve par récurrence :

- énoncer la propriété $P(n)$ que l'on veut montrer par récurrence
- ne pas oublier de déterminer et de traiter le (les) cas de base

- pour montrer que $P(n)$ est vraie, n'utiliser comme hypothèses que des $P(i)$ avec i **strictement** inférieur à n
- pour chaque condition $P(i)$ de la forme *si* $A\ldots$ *alors* $B\ \ldots$ utilisée dans la preuve, s'assurer que **toutes** les conditions pour que $A\ldots$ soit vrai sont remplies avant d'affirmer que $B\ldots$ est vrai.
- signaler à quel endroit on applique l'hypothèse de récurrence

La plupart des preuves de ce cours sont des preuves par récurrence.

On trouvera dans le lemme 25 un exemple assez détaillé de preuve par récurrence. Dans le paragraphe 2.2 figure une preuve pouvant servir de modèle à un type d'exercice assez courant en théorie des langages. D'autres exemples se trouvent dans les propositions 26 et 28.

1.2 Monoïdes

Définition 4 (monoïde) *Un monoïde est la donnée d'un ensemble M et d'une opération binaire interne $\circ$ sur M qui est associative et admet un élément neutre noté $\mathbb{1}_M$.*

Définition 5 (morphisme de monoïde)
Soient (M,$\circ$) et ($M^{'}, \circ^{'}$) deux monoïdes, un morphisme φ de M dans $M^{'}$ est une fonction telle que :

1. $\forall m_1, m_2 \in M \quad \varphi(m_1 \ \circ \ m_2) = \varphi(m_1) \ \circ^{'} \ \varphi(m_2)$
2. $\varphi(\mathbb{1}_M) = \mathbb{1}_{M'}$

Proposition 6 *Si $(M, \circ)$ est un monoïde alors $(\mathcal{P}(M), \circ)$ est aussi un monoïde, où l'opération sur $\mathcal{P}(M)$ est définie par :*

$$\forall X, Y \in \mathcal{P}(M) \qquad X \circ Y = \{x \circ y \mid x \in X, y \in Y\}$$

Preuve : Il est clair que cette opération est associative et qu'elle admet l'ensemble réduit à $\{\mathbb{1}_M\}$ comme élément neutre. Remarquez que l'ensemble vide, $\emptyset$, est un élément absorbant (élément zéro).

Puisque l'opération $\circ$ sur $\mathcal{P}(M)$ est associative, pour tout X de $\mathcal{P}(M)$ on peut définir l'exponentiation :

$$X^0 = \{\mathbb{1}_M\}$$
$$X^{n+1} = X \circ X^n$$

Définition 7 (sous-monoïde) *Une partie T d'un monoïde M est un sous-monoïde si :*

$$\mathbb{1}_M \in T \ , \ T^2 \subseteq T$$

Définition 8 (monoïde engendré) *Soit $(M, \circ)$ un monoïde, si A est une partie de M, on note A^* le monoïde engendré par A, c'est-à-dire le plus petit sous-monoïde de M contenant A. Les éléments de A sont appelés <u>générateurs</u> de A^*. Il est facile de montrer que $A^* = \bigcup_{n=0}^{\infty} A^n$.*

Notation : Il est traditionnel de noter A^+ l'ensemble suivant :

$$A^+ \ = \ \bigcup_{n=1}^{\infty} A^n \ = \begin{cases} A^* & si \ \mathbb{1}_M \in A \\ A^* \setminus \{\mathbb{1}_M\} & sinon \end{cases}$$

Définition 9 (alphabet) *Un alphabet est un ensemble fini non vide de symboles appelés lettres.*

Définition 10 (mot) *Un mot sur un alphabet A est une suite finie d'éléments de A, c'est-à-dire une application de $[n]$ dans A, pour quelque n. Une telle application sera représentée par $(a_1, \ldots, a_n)$ ou, plus simplement, par $a_1 \ldots a_n$. L'application de $[0]$ dans A est notée $\mathbb{1}_A$, abrégé en $\mathbb{1}$ en l'absence d'ambiguïté, et nommée mot vide.*

A^* est l'ensemble des mots sur A (cette notation est cohérente avec la précédente grâce à ce qui suit), la longueur d'un mot est l'entier n associé à sa définition en tant que suite, si bien que $\mathbb{1}$ a pour longueur zéro.

Définition 11 (monoïde libre) *Soient A un alphabet et A^* l'ensemble des mots sur A, on munit A^* de l'opération de concaténation définie ainsi :*

$$\forall s, t \in A^* \ , s \ : \ [n] \to A \text{ et } t \ : \ [m] \to A$$

alors le mot s.t noté également st est la suite suivante :

$$st \ : \ [n+m] \to A \ :$$

$$\begin{array}{c} \forall i, \ 1 \leq i \leq n, \quad st(i) = s(i) \\ \forall j, \ n+1 \leq j \leq n+m, \quad st(j) = t(j-n) \end{array}$$

Si s et t sont représentées respectivement par $a_1 \ldots a_n$ et $b_1 \ldots b_m$, st peut être représentée par $a_1 \ldots a_n b_1 \ldots b_m$. Il est alors clair que la concaténation est une opération associative et admettant $\mathbb{1}$ comme élément neutre. Si maintenant on identifie tout mot f de longueur 1 à la lettre f(1), A^* est appelé le monoïde libre engendré par A (à cet isomorphisme près).

Remarques :

1. Un monoïde M est appelé libre s'il admet un système de générateurs A tel que chacun des éléments de M se décompose de façon unique en produit d'éléments de A.
2. Il faut donc voir le mot $a_1 \ldots a_n$ comme le résultat des concaténations des lettres $a_1, a_2, \ldots, a_n$.

Fait fondamental : Soient $u = a_1 \ldots a_p, \ v = b_1 \ldots b_q, \ u = v$ si et seulement si $p = q$ et pour tout $i, \ 1 \leq i \leq p, \ a_i = b_i$

Définition 12 (facteur) *Soient s et t deux mots de A^*, t est un facteur de s si, et seulement si, il existe u et v de A^* tels que $s = utv$.*
Si u est le mot vide alors t est dit facteur gauche (ou préfixe) de s, si v est le mot vide alors t est dit facteur droit (ou suffixe) de s.
Si t est distinct de s et du mot vide, il est dit facteur propre de s.
Une factorisation d'un mot consiste en une écriture de ce mot en un produit de facteurs.

Théorème 13 *Soient $(M, \circ)$ un monoïde et f une application de A dans M, alors cette application s'étend en un unique morphisme de A^* dans M.*

Preuve : Nous démontrons d'abord l'existence puis l'unicité du morphisme qui étend f.

Existence : Nous définissons une application φ de A^* dans M de la façon suivante :

$$\varphi(\mathbb{1}) = \mathbb{1}_M$$

$$\begin{array}{rcl} \forall a \in A & \varphi(a) & = f(a) \\ \forall s \in A^*, s = a_1 \dots a_n & \varphi(s) & = f(a_1) \circ \dots \circ f(a_n) \end{array}$$

Tout d'abord il est clair que φ est une extension de f et que, par ailleurs, il suffit de montrer que $\varphi(st) = \varphi(s) \circ \varphi(t)$ pour s'assurer que φ est un morphisme de monoïde.

Si t=$\mathbb{1}$ alors φ(st) = φ(s) et $\varphi(s) \circ \varphi(t) = \varphi(s) \circ \varphi(\mathbb{1}) = \varphi(s) \circ \mathbb{1}_M = \varphi(s)$, d'où la propriété. De même si s=$\mathbb{1}$. Si s est le mot $a_1 \dots a_n$ et t le mot $b_1 \dots b_m$, alors par définition de φ :

$$\varphi(st) = f(a_1) \circ f(a_2) \circ \dots \circ f(a_n) \circ f(b_1) \circ \dots \circ f(b_m)$$

par associativité de $\circ$:

$$\varphi(st) = (f(a_1) \circ f(a_2) \circ \dots \circ f(a_n)) \circ (f(b_1) \circ \dots \circ f(b_m))$$

par définition de φ :

$$\varphi(st) = \varphi(s) \circ \varphi(t)$$

Unicité : Soit ψ un morphisme de A^* dans M étendant f, alors :

$$\psi(\mathbb{1}) = \mathbb{1}_M = \varphi(\mathbb{1})$$

$$\forall a \in A \ \psi(a) = f(a) = \varphi(a)$$

Tout mot s de longueur supérieure à 1 s'écrit $a_1 \dots a_n$ et puisque ψ est un morphisme :

$$\psi(a_1 \dots a_n) = \psi(a_1) \circ \dots \circ \psi(a_n) = f(a_1) \circ \dots \circ f(a_n) = \varphi(a_1) \circ \dots \circ \varphi(a_n)$$

d'où : $\forall s \in A^* \ \psi(s) = \varphi(s)$ et ainsi $\psi = \varphi$.

Corollaire 14 *Tout morphisme de A^* dans M est entièrement déterminé par l'image des lettres de A.*

1.3 Langages et grammaires

Dans la suite de ce cours, nous allons nous intéresser aux sous-ensembles de A^*, appelés langages, et à diverses méthodes pour engendrer et reconnaître ces langages.

Définition 15 (langage) *Un langage sur A est un sous-ensemble de A^*.*

Opérations sur les langages : Comme vu dans la proposition 6 , $(\mathcal{P}(A^*), .)$ est un monoïde. On peut ainsi parler du produit L.M de deux langages L et M et de l'exponentiation L^k, ainsi que de L^* et L^+. On sera amené à utiliser également l'union $L \cup M$, l'intersection $L \cap M$ et la complémentation $L \setminus M$.

On a ainsi les propriétés remarquables suivantes :

- $\emptyset^* = \{\mathbb{1}\}$
- $\emptyset.L = L.\emptyset = \emptyset$
- $(L^*)^* = L^*$
- $L.(M \cup N) = (L.M) \cup (L.N)$
- $(L \cup M).N = (L.N) \cup (M.N)$
- $(L \cup M)^* = (L^*M)^*.L^*$

Définition 16 (grammaire)
Une grammaire est définie par un quadruplet (X,V,S,P) où :

- *X est un alphabet dit terminal*
- *V est un autre alphabet, disjoint de X, dit non-terminal (ou auxiliaire)*
- *S est une lettre distinguée de V dite axiome*
- *P est un ensemble fini de couples de $(X \cup V)^* V (X \cup V)^* \times (X \cup V)^*$ dits règles de production.*

Définition 17 (dérivation dans une grammaire) *Un mot u de $(X \cup V)^*$ se dérive directement en un mot v de $(X \cup V)^* ssi$*
$\exists (g, d) \in P \ \exists u_1, u_2 \in (X \cup V)^*$ tels que $u = u_1 g u_2$ et $v = u_1 d u_2$.

On note $u \underset{G}{\longrightarrow} v$ cette relation.

On note $\underset{G}{\overset{n}{\longrightarrow}}$ l'exponentiation à l'ordre n de $\underset{G}{\longrightarrow}$:

$$\forall u, v \in (X \cup V)^* \ \forall n > 0 \ u \underset{G}{\overset{n}{\longrightarrow}} v \ ssi$$
$$\exists u_0, u_1, \ldots, u_n \ tels \ que \ u_0 = u, \ u_n = v \ et \ \forall i \in [0, n-1] \ u_i \underset{G}{\longrightarrow} u_{i+1}$$

On note $\xrightarrow[G]{*}$ *la fermeture réflexive et transitive de* $\xrightarrow[G]{}$ *:*

$$\forall u, v \in (X \cup V)^* \; u \xrightarrow[G]{*} v \text{ ssi}$$
$$\text{soit } u = v$$
$$\text{soit } \exists n \; u \xrightarrow[G]{n} v$$

Définition 18 (langage engendré) *On appelle langage engendré par une grammaire G d'axiome S l'ensemble* $\{f \in X^* \mid S \xrightarrow[G]{*} f\}$ *et on le note L(G,S) (ou parfois L(G)). On appelle langage élargi, l'ensemble* $\{f \in (X \cup V)^* \mid S \xrightarrow[G]{*} f\}$ *et on le note* $\hat{L}$*(G,S).*

Suivant l'allure des règles de production, on opère une classification des grammaires dite classification de Chomsky.

Définition 19 (classification de Chomsky) *Soit G une grammaire (X,V,S,P).*

- *Si P est inclus dans* $(X \cup V)^* V (X \cup V)^* \times (X \cup V)^*$*, elle est dite de type 0 ou à structure de phrase.*
- *Si les règles de P sont de la forme* (lTr, lmr) *avec* $l, r \in (X \cup V)^*, T \in V, m \in (X \cup V)^+$*, ou éventuellement de la forme* $(S, \mathbb{1})$ *si S ne figure dans aucun membre droit de règle, alors G est dite de type 1 ou contextuelle ou encore context-sensitive*
- *Si P est inclus dans* $V \times (X \cup V)^*$*, elle est dite de type 2 ou algébrique ou encore context-free (indépendante du contexte).*
- *Si P est inclus dans* $V \times X^* V \cup X^*$ *ou dans* $V \times V X^* \cup X^*$ *, elle est dite de type 3 ou encore linéaire droite ou linéaire gauche.*

Remarque importante : Si on note respectivement $\mathcal{L}_0$, $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$ les familles des langages engendrés par les grammaires de types 0, 1, 2, 3, on peut annoncer:

$$\textit{Les inclusions } \mathcal{L}_3 \subset \mathcal{L}_2 \subset \mathcal{L}_1 \subset \mathcal{L}_0 \textit{ sont strictes}$$

Exemple 20

1. *type 0* $X=\{a, b, c\}$, $V=\{S, T\}$

$$\begin{cases} S & \to & ab \\ S & \to & aST \\ ST & \to & b \\ a & \to & bS \end{cases}$$

2. *type 1* $X=\{a,\ b,\ c\},\ V=\{S,\ T\}$

$$\begin{cases} S & \to & abc \\ S & \to & aTBc \\ TB & \to & TW \\ TW & \to & BW \\ BW & \to & BT \\ Tc & \to & XBcc \\ BX & \to & BY \\ BY & \to & XY \\ XY & \to & XB \\ aX & \to & aaT \\ aX & \to & aa \\ B & \to & b \end{cases}$$

Cette grammaire engendre le langage $\{a^n b^n c^n, n > 0\}$

3. *type 2* $X=\{a,\ b\},\ V=\{S,\ T\}$

$$\begin{cases} S & \to & ST \\ S & \to & \mathbb{1} \\ T & \to & aSb \end{cases}$$

4. *type 3* $X=\{a,\ b\},\ V=\{S,\ T,\ U\}$

$$\begin{cases} S & \to & aS \\ S & \to & aT \\ T & \to & bU \\ T & \to & b \\ U & \to & aU \\ U & \to & b \end{cases}$$

Notation : Pour raccourcir la description des productions d'une grammaire nous regroupons les règles ayant le même membre gauche de la façon suivante :

Exemple 21

1.

$$\begin{cases} S & \to & ab + aST \\ ST & \to & b \\ a & \to & bS \end{cases}$$

2.

$$\begin{cases} S & \to & abc + aTBc \\ TB & \to & TW \\ TW & \to & BW \\ BW & \to & BT \\ Tc & \to & XBcc \\ BX & \to & BY \\ BY & \to & XY \\ XY & \to & XB \\ aX & \to & aaT + aa \\ B & \to & b \end{cases}$$

3.

$$\begin{cases} S & \to & ST + \mathbb{1} \\ T & \to & aSb \end{cases}$$

4.

$$\begin{cases} S & \to & aS + aT \\ T & \to & bU + b \\ U & \to & aU + b \end{cases}$$

Nous aborderons directement les grammaires algébriques.

Exercices :

Exercice 1 :
Montrer que l'ensemble des relations sur un ensemble E muni de l'opération composition est un monoïde.

Exercice 2 :
Montrer que l'intersection de 2 sous-monoïdes est un sous-monoïde.

Exercice 3 :
Montrer que tous les mots de $\{a, b\}^*$ de longueur supérieure ou égale à 4 contiennent au moins une répétition.

Exercice 4 : (Equidivisibilité du monoïde libre)
Soient $u_1.v_1 = u_2.v_2$ deux factorisations d'un mot de X^*. Montrer qu'il existe un mot $w \in X^*$ tel que :

- soit $u_1.w = u_2$ et $w.v_2 = v_1$
- soit $u_2.w = u_1$ et $w.v_1 = v_2$

Exercice 5 : (plus difficile)
Donner une grammaire de type 0 qui engendre l'ensemble :

$$L = \{a^i b^j a^i b^j \mid i, j \geq 1\}$$

Expliquer clairement comment la grammaire fonctionne, et prouver qu'elle engendre L, c'est-à-dire que tout mot de L(G) est dans L et tout mot de L est dans L(G).

Exercice 6 :
Caractériser les solutions de l'équation $f.g = g.f$, où f et g sont des mots de X^*.

Chapitre 2

Grammaires algébriques

2.1 Dérivation, Lemme fondamental

Nous avons vu en fin de chapitre précédent, qu'une grammaire algébrique G est définie par un quadruplet (X,V,S,P) où P est inclus dans $V \times (X \cup V)^*$.

Par définition de la relation de dérivation directe, il est facile de voir que :

Proposition 22 *Soient G une grammaire algébrique, f et g deux mots de* $(X \cup V)^*$*, f* $\underset{G}{\longrightarrow}$ *g si et seulement si :*

$$\exists u, v, m \in (X \cup V)^*,\ \exists T \in V\ f = uTv,\ g = umv \text{ et } (T, m) \in P$$

Soient f et g deux mots de $(X \cup V)^*$, f se dérive à l'ordre k en g si et seulement si f $\overset{k}{\underset{G}{\longrightarrow}}$ g et f se dérive proprement en g si il existe k ≥ 1 tel que f $\overset{k}{\underset{G}{\longrightarrow}}$ g, ce que nous noterons f $\overset{+}{\underset{G}{\longrightarrow}}$ g (fermeture transitive de la relation $\underset{G}{\longrightarrow}$). k est appelé l'ordre de la dérivation.

Rappelons que L(G,S) (respectivement $\hat{\text{L}}$(G,S)) désigne le langage (respectivement langage élargi) engendré par G.

Définition 23 (langage algébrique) *Un langage algébrique est une partie L de* X^* *telle qu'il existe une grammaire algébrique (X,V,S,P) qui l'engendre i.e. L=L(G,S). Alg*(X^*) *est l'ensemble des langages algébriques sur* X^* *(on parlera plutôt de la famille des langages algébriques).*

Proposition 24 *Soit G = (X,V,S,P) une grammaire algébrique, les propositions suivantes sont vraies :*

1. $\forall S \in V\ \forall m \in (X \cup V)^*\ (S, m) \in P\ \Rightarrow\ S \underset{G}{\longrightarrow} m$

2. $\forall f, g, u, v \in (X \cup V)^*\ f \xrightarrow[G]{*} g \ \Rightarrow\ ufv \xrightarrow[G]{*} ugv$
3. $\forall f, g, h \in (X \cup V)^*\ f \xrightarrow[G]{*} g\ et\ g \xrightarrow[G]{*} h \ \Rightarrow\ f \xrightarrow[G]{*} h$
4. $\forall f, g, f', g' \in (X \cup V)^*\ f \xrightarrow[G]{*} g\ et\ f' \xrightarrow[G]{*} g' \ \Rightarrow\ ff' \xrightarrow[G]{*} gg'$
5. $\forall f, g, f' \in (X \cup V)^*\ \forall u, v \in X^*\ (f = uf'v\ et\ f \xrightarrow[G]{*} g) \ \Rightarrow$
   $\exists g' \in (X \cup V)^*\ (g = ug'v\ et\ f' \xrightarrow[G]{*} g')$

Preuve : Les quatre premières sont des corollaires triviaux de la définition de la relation de dérivation qui est une congruence. Avant de prouver la dernière, nous énonçons un lemme technique dont la preuve est laissée en exercice :

Lemme 25 *Quels que soient $u_1,\ u_2\ , v_1\ , v_2$ de X^* tels que $u_1v_1 = u_2v_2$, il existe un mot w de X^* tel que :*

$$(u_1w = u_2\ et\ wv_2 = v_1)\ ou\ (u_2w = u_1\ et\ wv_1 = v_2)$$

Ce lemme énonce la propriété dite d'équidivisibilité du monoïde libre.

La cinquième proposition peut être maintenant prouvée par récurrence sur l'ordre de dérivation n de $f \xrightarrow[G]{n} g$. On va en fait montrer plus précisément :

$$\forall n \in \mathrm{N},\ \forall f, g, f' \in (X \cup V)^*,\ \forall u, v \in X^*,$$
$$(f = uf'v\ et\ f \xrightarrow[G]{n} g)\ \Rightarrow \exists g' \in (X \cup V)^*,\ (g = ug'v\ et\ f' \xrightarrow[G]{*} g')$$

- Si cet ordre est nul, la propriété est évidente car $f = g$. Il suffit en effet de prendre $g' = f'$.
- Si l'ordre est 1 alors f $\xrightarrow[G]{}$ g donc :

  $\exists \bar{u}, \bar{v} \in (X \cup V)^*, \exists T \in V, \exists m \in (X \cup V)^*\ t.q.\ f = \bar{u}T\bar{v}\ ,\ (T, m) \in P\ et\ g = \bar{u}m\bar{v}$

  On a donc d'une part, $f = uf'v$ et d'autre part, $f = \bar{u}T\bar{v}$. T ne peut pas être dans u ni dans v puisque $u, v \in X^*$. D'après le lemme précédent, il existe deux mots u' et v' tels que :

  $$\bar{u} = uu',\ \bar{v} = vv'\ et\ f' = u'Tv'$$

  d'où, en posant $g' = u'mv'$, on déduit :

  $$f' = u'Tv' \xrightarrow[G]{} u'mv' = g'$$

  et :

  $$g = \bar{u}m\bar{v} = (uu')m(v'v) = u(u'mv')v = ug'v$$

– Soit $n \geq 1$. Supposons la propriété vraie à l'ordre n et montrons qu'elle est alors vraie à l'ordre n+1. Soit donc $f = uf'v$ et $f \xrightarrow[G]{n+1} g$. Il existe f_1 tel que $f \xrightarrow[G]{1} f_1 \xrightarrow[G]{n} g$ par définition de la relation de dérivation. Comme $f = uf'v$, d'après le point ci-dessus (cas où l'ordre de dérivation est 1), il existe f'_1 tel que $f_1 = uf'_1v$ et $f' \xrightarrow[G]{*} f'_1$. Mais l'hypothèse de récurrence s'applique à $f_1 \xrightarrow[G]{n} g$ et donc il existe g' tel que $g = ug'v$ et $f'_1 \xrightarrow[G]{*} g'$. Comme $f' \xrightarrow[G]{} f'_1$, on en déduit $f' \xrightarrow[G]{*} g'$.

Notons une particularité de cette preuve :

- étude du cas $n = 0$
- étude du cas $n = 1$: ce cas peut sembler superflu !
- étude du cas $n >= 1$: passage de n à $n + 1$. Dans l'étude du cas $n + 1$ on se ramène aux deux cas $n' = 1$ et $n'' = n$. On peut sans problème appliquer l'hypothèse de récurrence au cas $n'' = n$. Par contre, il est nécessaire de s'assurer que la propriété est vraie à l'ordre 1.

Cette dernière propriété s'étend en la proposition suivante, que l'on appelle souvent le Lemme fondamental.

Proposition 26 (Lemme fondamental)
Soit une grammaire algébrique G=(X,V,S,P) et f un mot de $(X \cup V)^*$ *de la forme :*

$$f = f_0 S_1 f_1 S_2 \dots f_{k-1} S_k f_k \ , \ k \geq 1$$

où $f_0, f_1 \dots f_k \in X^*, \ S_1, \dots S_k \in V$ *, alors* $f \xrightarrow[G]{*} g$ *si, et seulement si, g est de la forme :*

$$g = f_0 h_1 f_1 h_2 \dots f_{k-1} h_k f_k$$

$$avec \ \forall i, \ 1 \leq i \leq k, \ h_i \in (X \cup V)^* \ et \ S_i \xrightarrow[G]{*} h_i.$$

De façon plus précise, nous allons prouver que si f $\xrightarrow[G]{p}$ g alors $S_i \xrightarrow[G]{p_i} h_i$ et $p = \sum_{i=1}^{k} p_i$.

Preuve : La condition suffisante est triviale. La condition nécessaire se prouve par récurrence sur p.
Soit à montrer :

$$\forall k \geq 1, \forall f \in (X \cup V)^*,$$

$$\forall f_0, f_1 \dots f_k \in X^*, \ S_1, \dots S_k \in V \ tels \ que \ f = f_0 S_1 f_1 S_2 \dots f_{k-1} S_k f_k,$$

$$\forall p \in \mathrm{N},\ \forall g \in (X \cup V)^*,\ f \xrightarrow[G]{p} g \text{ si et seulement si,}$$

$$\exists h_0, h_1, \ldots, h_k \in (X \cup V)^*, \exists p_0, p_1, \ldots, p_k \in \mathrm{N},$$

$$g = f_0 h_1 f_1 h_2 \ldots f_{k-1} h_k f_k \text{ et } \forall i,\ 1 \leq i \leq k,\ S_i \xrightarrow[G]{p_i} h_i \text{ et } p = \sum_{i=1}^{k} p_i$$

- cas $p = 0$: si $f \xrightarrow[G]{0} g$ alors $f = g$; on pose $\forall i,\ 1 \leq i \leq k, h_i = S_i$ et $p_i = 0$. On a bien ainsi $g = f = f_0 h_1 f_1 h_2 \ldots f_{k-1} h_k f_k$, $S_i \xrightarrow[G]{p_i = 0} h_i$ et $p = 0 = \sum_{i=1}^{k} p_i$
- cas $p \geq 0$: Supposons la propriété vraie à l'ordre p et $f \xrightarrow[G]{p+1} g$: alors il existe g' tel que $f \xrightarrow[G]{p} g' \xrightarrow[G]{} g$. L'hypothèse de récurrence s'applique à $f \xrightarrow[G]{p} g'$ donc il existe h'_i et p'_i, $1 \leq i \leq k$ tels que :

$$S_i \xrightarrow[G]{p'_i} h'_i\ ,\ p = \sum_{i=1}^{k} p'_i \text{ et } g' = f_0 h'_1 f_1 \ldots f_{k-1} h'_k f_k$$

$g' \xrightarrow[G]{} g$ entraine $g' = uTv$ et $g = umv$, avec $(T, m) \in P$ et comme les f_i sont dans X^* cela entraine l'existence d'un $j, 1 \leq j \leq k$, tel que $h'_j = \bar{u}T\bar{v},\ u = f_0 h'_1 f_1 \ldots f_{j-1}\bar{u}$, $v = \bar{v} f_j h'_{j+1} \ldots h'_k f_k$ d'où $g = f_0 h'_1 f_1 \ldots f_{j-1}\bar{u}m\bar{v} f_j h'_{j+1} \ldots h'_k f_k$ et h'$_j \xrightarrow[G]{} \bar{u}m\bar{v}$.
En posant :

$$\forall i,\ 1 \leq i \leq k,\ i \neq j,\ h_i = h'_i \text{ et } h_j = \bar{u}m\bar{v}\ ,\ p_i = p'_i \text{ et } p_j = p'_j + 1$$

On en déduit :

$$g = f_0 h_1 f_1 \ldots f_{j-1} h_j f_j h_{j+1} \ldots h_k f_k$$

avec :

$$S_i \xrightarrow[G]{p_i} h_i \text{ et } p + 1 = \sum_{i=1}^{k} p_i$$

2.2 Deux exemples de langages algébriques

2.2.1 Un exemple classique

On a coutume de prendre comme exemple-type de langage algébrique le langage suivant :

$$L = \{a^n b^n \mid n \in \mathrm{N}\}$$

Montrons que ce langage est engendré par la grammaire suivante :
$G = (\{a, b\}, \{S\}, S, P)$ où :

$$P = \left\{ \begin{array}{rcl} S & \rightarrow & aSb \mid \mathbb{1} \end{array} \right.$$

Pour ceci, on procède en montrant deux inclusions :

1. $L(G, S) \subseteq L$
   La preuve se fait par récurrence sur l'ordre de dérivation. On va montrer :

   $$\forall n \in \mathrm{N}, \forall f \in X^*,\ S \xrightarrow[G]{n} f \ \Rightarrow f \in L$$

   Pour alléger la preuve, notons $\mathcal{P}(n)$ la propriété $\forall f \in X^*,\ S \xrightarrow[G]{n} f \ \Rightarrow f \in L$.

   - cas $n = 0$: $S \xrightarrow[G]{0} f$ implique $f = S$ et donc $f \notin X^*$. $f \in X^*$ *et* $S \xrightarrow[G]{0} f$ est toujours faux et ainsi l'implication ($\mathcal{P}(0)$) est vraie. On a coutume de ne pas étudier ce genre de situation.
   - cas $n = 1$: $S \xrightarrow[G]{1} f$ *et* $f \in X^*$ implique $f = \mathbb{1}$ par définition de P. Or $\mathbb{1} \in L$ car $\mathbb{1}$ est de la forme $a^0 b^0$. Donc $\mathcal{P}(1)$ est vraie.
   - soit $n \geq 1$ et supposons $\mathcal{P}(n)$ vraie. On va montrer que $\mathcal{P}(n+1)$ est alors vraie. Soit donc $f \in X^*$ tel que $S \xrightarrow[G]{n+1} f$. Cette dérivation se décompose en $S \xrightarrow[G]{1} aSb \xrightarrow[G]{n} f$ (attention, on ne peut envisager cette décomposition que si $n+1$ est au moins égal à 2, c'est pourquoi le cas $n = 1$ doit absolument être traité avant). D'après le lemme fondamental, il existe un mot g de X^* tel que $f = agb$ et $S \xrightarrow[G]{n} g$. On peut alors appliquer l'hypothèse de récurrence, et déduire que $g \in L$. D'après la forme des mots de L, il existe un entier i tel que $g = a^i b^i$. On en déduit que $f = agb = a^{i+1} b^{i+1}$ et ainsi $f \in L$. On a bien montré que $\mathcal{P}(n+1)$ est vraie.
2. $L \subseteq L(G, S)$
   La preuve se fait par récurrence sur la longueur des mots de L, mais cette fois-ci c'est une récurrence forte que l'on utilise. Montrons :

   $$\forall n \in \mathrm{N}, \forall f \in X^*,\ (\mid f \mid = n \ et \ f \in L) \Rightarrow f \in L(G, S)$$

   Pour alléger la preuve, notons $\mathcal{Q}(n)$ la propriété

   $$\forall f \in X^*,\ (\mid f \mid = n \ et \ f \in L) \Rightarrow f \in L(G, S)$$

   - cas $n = 0$: $\mid f \mid = 0$ signifie $f = \mathbb{1}$, et on a bien $\mathbb{1} \in L$. Il faut vérifier que l'on a bien $\mathbb{1} \in L(G, S)$. Ceci découle de l'application de la règle $S \rightarrow \mathbb{1}$. Donc $\mathcal{Q}(0)$ est vraie.
   - soit $n \geq 0$ et supposons $\mathcal{Q}(k)$ vraie pour tout entier k tel que $0 \leq k \leq n$. On va montrer que $\mathcal{Q}(n+1)$ est alors vraie.

Soit $f \in L$ tel que $\mid f \mid = n+1$. D'après la définition de L, f est de la forme $f = a^i b^i$ pour un entier $i > 0$.
Posons $g = a^{i-1}b^{i-1}$, ce qui est toujours possible puisque $i > 0$.
On remarque que $g \in L$. De plus, g étant de longueur $n-1$, et $n-1$ étant inférieur à n on peut appliquer l'hypothèse de récurrence $\mathcal{Q}(n-1)$: $(\mid g \mid = n-1 \; et \; g \in L) \Rightarrow g \in L(G,S)$. Ceci signifie qu'il existe un entier $m \geq 0$ tel que $S \xrightarrow[G]{m} g$. Mais alors $S \xrightarrow[G]{1} aSb \xrightarrow[G]{m} agb = f$ d'après le lemme fondamental, et on a bien $f \in L(G,S)$, d'où $\mathcal{Q}(n+1)$.

2.2.2 Un autre exemple

Soit le langage M suivant :

$$M = \{a^n b^m \mid n, m \in \mathrm{N},\ n = m+1 \; ou \; m = n+1\}$$

Montrons que ce langage est engendré par la grammaire suivante, d'axiome T : $G' = (\{a,b\}, \{S,T\}, T, P')$ où :

$$P' = \begin{cases} T & \rightarrow & aS \mid Sb \\ S & \rightarrow & aSb \mid \mathbb{1} \end{cases}$$

On montre en recopiant la preuve précédente que $L(G',S) = L$. On va montrer que $L(G',T) = M$, en montrant successivement les deux inclusions :

1. $L(G',T) \subseteq M$
   Soit $f \in X^*$ tel que $T \xrightarrow[G']{*} f$. Cette dérivation se décompose
   - soit en $T \xrightarrow[G']{1} aS \xrightarrow[G]{*} f$
     D'après le lemme fondamental, f est de la forme ag avec $S \xrightarrow[G]{*} g$. Remarquons que cette sous-dérivation n'utilise que des règles de G. On peut donc en déduire que $g \in L$. Donc g est de la forme $a^n b^n$ pour un certain entier n. D'où $f = aa^n b^n = a^{n+1}b^n$ et ce mot est bien un mot de M,
   - soit en $T \xrightarrow[G']{1} Sb \xrightarrow[G]{*} f$
     De la même façon, f est de la forme gb avec $S \xrightarrow[G]{*} g$. On a alors $g \in L$, et donc g est de la forme $a^n b^n$ pour un certain entier n, d'où $f = a^n b^n b = a^n b^{n+1}$ qui est bien un mot de M.
2. $M \subseteq L(G',T)$
   Soit $f \in M$. Deux cas peuvent se présenter :
   - soit f possède un a de plus que de b, $f = a^{n+1}b^n$ pour un certain entier n. Posons $g = a^n b^n$. On a $f = ag$. De plus, $g \in L$. On en déduit que $S \xrightarrow[G]{*} g$. En

complétant cette dérivation par une règle de G', on obtient, grâce au lemme fondamental, $T \xrightarrow[G']{1} aS \xrightarrow[G]{*} ag = f$. On a ainsi montré que $f \in L(G', T)$.

– soit f possède un b de plus que de a. On reprend la même argumentation en utilisant cette fois-ci la règle $T \xrightarrow[G']{1} Sb$.

2.3 Arbres de dérivation

Définition 27 (dérivation gauche) *Soit G une grammaire, une dérivation gauche d'un mot f de $(X \cup V)^*$ en un mot g de X^* est obtenue en dérivant à chaque pas le symbole non-terminal le plus à gauche, c'est à dire une dérivation telle que:*

$$\exists f_0, f_1, \ldots, f_n \in (X \cup V)^*,\ f = f_0,\ g = f_n,\ \forall i,\ 0 \leq i \leq n,\ f_i \xrightarrow[G]{} f_{i+1}$$

avec

$$\begin{cases} f_i = u_i T_i v_i,\ f_{i+1} = u_i m_i v_i,\ (T_i, m_i) \in P \\ \forall i,\ 0 \leq i \leq n,\ u_i \in X^*,\ v_i \in (X \cup V)^* \end{cases}$$

Remarque : Si on n'impose pas que tous les u_i soient dans X^*, on retrouve la définition d'une dérivation.

Proposition 28 *Soit G une grammaire, il existe une dérivation gauche de f en g dans G si, et seulement si, il existe une dérivation de f en g dans G.*

Preuve : par récurrence sur l'ordre de dérivation. Soit à montrer : $\forall n \in \mathrm{N}, \forall f, g \in (X \cup V)^*$, si $f \xrightarrow[G]{n} g$ alors il existe une dérivation gauche de f en g.

– Si $g \in X^*$ dérive à l'ordre 1 de f alors $f = g'Tg$" , $g = g'hg$" avec g', g" $\in X^*$ et $(T, h) \in P$. La dérivation est gauche par définition.

– Supposons montré la propriété vraie à tout ordre inférieur ou égal à n. Soient f et g tels que $f \xrightarrow[G]{n+1} g$
Posons $f = f_1 T'_1 f_2 \ldots f_q T'_q f_{q+1}$ où $\forall j, f_j \in X^*,\ T'_j \in V$.
D'après le lemme fondamental, on a $g = f_1 h_1 f_2 \ldots f_q h_q f_{q+1}$ où pour tout j, h_j dérive de T'_j à un ordre inférieur ou égal à n et l'hypothèse de récurrence s'applique q fois.
Par conséquent, pour tout j, il existe une dérivation gauche de T'_j en h_j. En partant de $f = f_1 T'_1 f_2 \ldots f_q T'_q f_{q+1}$ et en effectuant pour j croissant de 1 à q les dérivations gauches de T'_j en h_j, on obtient une dérivation gauche de f en $g = f_1 h_1 f_2 \ldots f_q h_q f_{q+1}$. La propriété est donc démontrée à l'ordre $n + 1$.

Nous appellerons les dérivations gauches de T'_j en h_j sous-dérivations gauches de la dérivation gauche de f en g.

De cette proposition, découle la remarque suivante : étant donnée une grammaire G=(X,V,S,P), le langage L(G,S) est l'ensemble des mots de X^* obtenus par dérivation gauche à partir de l'axiome S dans G.

Succintement, nous présentons la définition de domaine d'arbre, nécessaire à la définition d'arbre (de dérivation).

Précisons d'abord que nous notons N_+^*, le monoïde libre sur l'ensemble des entiers strictement positifs N_+. Nous noterons par le symbole . l'opération de concaténation et par $\mathbb{1}$ l'élément neutre pour cette opération.

Définition 29 (domaine d'arbre) *Un domaine d'arbre est une partie (non vide) Δ de N_+^* telle que :*

- $u \in \Delta$ *et* $u = v.w \ \Rightarrow \ v \in \Delta$
- $\forall n \in \mathrm{N}\ (u.n \in \Delta \ \Rightarrow \forall j,\ 1 \leq j \leq n,\ u.j \in \Delta)$

Remarque : De la définition précédente, il découle que si Δ est un domaine d'arbre alors $\mathbb{1}$ appartient nécessairement à Δ. En effet Δ est non vide, donc il existe un élément de N_+^* appartenant à Δ que nous appelerons u ; ou bien u est l'élément $\mathbb{1}$ ou bien u peut s'écrire u=$\mathbb{1}$.u et donc, par définition, $\mathbb{1}$ appartient à Δ.

Définition 30 (arbre de dérivation) *Un arbre de dérivation τ, sur une grammaire G=(X,V,S,P), est une application de Δ (un domaine d'arbre) dans $X \cup V \cup \{\mathbb{1}\}$ telle que :*
$\forall w \in \Delta$

1. *Soit* $\tau(w) \in X \cup \{\mathbb{1}\}$ *et* $\forall n \in \mathrm{N}_+\ w.n \notin \Delta$
2. *Soit* $\tau(w) \in V$ *et* $\exists n \in \mathrm{N}_+\ w.n \in \Delta$ *et* $\forall m > n\ w.m \notin \Delta$ *et* $(\tau(w), \tau(w.1)\tau(w.2)\ldots\tau(w.n)) \in P$

Exemple 31 *Soit la grammaire $S \longrightarrow SS + aSb + \mathbb{1}$. Soit l'arbre de dérivation représenté page suivante.*

Son domaine est Δ=$\{\mathbb{1}$,1,2,1.1,1.2,1.3,2.1,2.2,2.3,1.2.1,1.2.2, 1.2.3,2.2.1,1.2.2.1$\}$.

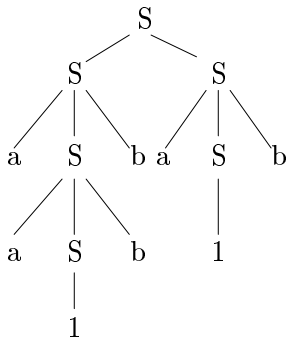
Sur cet exemple, on remarque que, lisant les feuilles de l'arbre de gauche à droite, on trouve le mot a^2b^2ab qui est dans L(G,S).

Définition 32 (feuillage d'un arbre) *Soit τ un arbre de dérivation, on définit l'application $\check{\tau}$ de Δ, le domaine de τ, dans $(X \cup V)^*$ par :*

- $\check{\tau}(w) = \tau(w)$ *si* $\tau(w) = \in X \cup \{\mathbb{1}\}$

– $\check{\tau}(w) = \check{\tau}(w.1)\ldots\check{\tau}(w.n)$ *si* $\tau(w) \in V$ *et* $w.n \in \Delta$ *et* $\forall m > n\ w.m \notin \Delta$

Le feuillage de l'arbre est le mot $\check{\tau}(\mathbb{1}_{\mathrm{N}_+^*})$.



Définition 33 (arbre associé à une dérivation gauche) *Soient* τ *un arbre de dérivation et* $S \xrightarrow[G]{+} f$, *où* $f \in X^*$, *une dérivation gauche. Notons* $S_{k_1}, \ldots, S_{k_n}$ *les* n *non-terminaux dérivés et* $h_1, \ldots, h_n$ *les membres droits des règles de* P *correspondants. (On suppose que* n *est l'ordre de dérivation et* $S = S_{k_1}$*).*

On dit que la dérivation et l'arbre sont <u>associés</u> *s'il existe une bijection* φ *de* $[n]$ *dans* $\Delta = \{w \in dom(\tau) \mid \tau(w) \in V\}$ *vérifiant:*

1. $i < j \Rightarrow \varphi(i) < \varphi(j)$ $(ordre\ lexicographique)$
2. $\forall i \in [n]\ \tau(\varphi(i)) = S_{k_i}$
3. $\forall i \in [n]\ \tau(\varphi(i).1)\ldots\tau(\varphi(i).a(\varphi(i))) = h_i$ *où* $a(\varphi(i))$ *est le plus grand entier* p *tel que* $\varphi(i).p \in \Delta$

Proposition 34 *A toute dérivation gauche correspond un seul arbre de dérivation associé* τ *et réciproquement.*

La preuve découle directement de la définition 33.

2.4 Ambiguïté

Définition 35 (grammaire ambiguë) *Une grammaire algébrique G est dite non-ambiguë si pour tout mot f de L(G,S), il existe une seule dérivation gauche de S en f. Elle est dite ambiguë sinon.*

Un langage algébrique est dit non-ambigu si il existe une grammaire non-ambiguë qui l'engendre. Il est dit inhéremment ambigu, ou plus simplement ambigu, sinon.

Proposition 36 *L'union de deux langages algébriques disjoints non-ambigus est un langage algébrique non-ambigu.*

2.5 Propriétés de cloture de Alg(X^*)

Proposition 37 *Si L et M sont deux langages algébriques, $L \cup M$ est algébrique.*

Cette propriété s'énonce également : Alg(X^*) est close par union.

Proposition 38 *Alg(X^*) est close par produit de concaténation et par étoile.*

Preuve : Il s'agit, étant donnée une grammaire de L (resp. une grammaire de L et une grammaire de M), d'en déduire par construction une grammaire de L^* (resp. de $L \cup M$, *de* $L.M$). Soit $G = (X, V, S, P)$ une grammaire du langage L, et $G' = (X', V', S', P,)$ une grammaire de M. Au besoin, on renomme les non-terminaux de V' pour que V et V' soient disjoints. Soit S" un nouveau non-terminal.

Posons $G_1 = (X \cup X', V \cup V' \cup \{S"\}, S", P \cup P' \cup \{S" \longrightarrow S + S'\})$. On montre alors que la grammaire G_1 engendre le langage $L \cup M$ (laissé en exercice au lecteur). De la même façon $G_2 = (X, V \cup \{S"\}, S", P \cup \{S" \longrightarrow SS" + \mathbb{1}\})$ engendre L^* et $G_3 = (X \cup X', V \cup V' \cup \{S"\}, S", P \cup P' \cup \{S" \longrightarrow SS'\})$ engendre $L.M$.

Remarque : Alg(X^*) n'est pas close par intersection.

2.6 Formes de grammaires

Dans ce qui suit nous allons définir trois formes de grammaires particulières, auquelles on peut ramener pratiquement toutes les grammaires algébriques. Les preuves n'ont aucun intérêt pour l'étudiant de licence autre que celui d'être constructives ; elles fournissent donc chacune un algorithme de mise en forme d'une grammaire donnée en une forme déterminée. Nous ne donnons pas la construction pour les grammaires réduites, le lecteur pourra s'inspirer des deux autres constructions pour s'en faire une sienne.

2.6.1 Grammaires propres :

Définition 39 (grammaire propre) *Une grammaire algébrique G=(X,V,S,P) est dite propre si, et seulement si, P est inclus dans $V \times ((X \cup V)^* \backslash (V \cup \{\mathbb{1}\}))$. Un langage algébrique est propre s'il est engendré par une grammaire algébrique propre.*

Proposition 40 *Un langage L sur X^* est algébrique si, et seulement si, $L\backslash\{\mathbb{1}\}$ est un langage algébrique propre.*

Par conséquent, un langage algébrique est propre si et seulement si il ne contient pas le mot vide.

Preuve : Si L\{𝟙} est un langage algébrique propre, il est clair que L est algébrique. Il faut donc prouver que pour tout langage algébrique L=L(G,S), engendré par une grammaire G=(X,V,S,P), il existe une grammaire algébrique propre $\bar{G} = (X, \bar{V}, \bar{S}, \bar{P})$ telle que $L(\bar{G}, \bar{S})$ =L\{𝟙}.

La construction de $\bar{G}$ se fait en deux étapes :

1. Suppression des règles de la forme T $\xrightarrow[G]{}$ 𝟙
2. Suppression des règles de la forme T $\xrightarrow[G]{}$ T'

1- On considère la suite $\{V_i \mid i \in \mathbb{N}\}$ des parties de V définie ainsi :

$$V_0 = \{T \in V \mid (T \to \mathbb{1}) \in P\}$$

$$V_{i+1} = \{T \in V \mid \exists w \in V_i^+ \ t.q. \ (T \to w) \in P\} \cup V_i$$

Pour tout i ∈ ℕ, on a $V_i \subseteq V_{i+1} \subseteq V$; il est facile de montrer par récurrence que la suite est constante à partir d'un certain rang, disons k, et que $V_k = \{T \in V \mid \mathbb{1} \in L(G,T)\}$.

Construisons alors la grammaire G'=(X,V,S,P') telle que :

1. $P\backslash(V \times \{\mathbb{1}\}) \subseteq P'$
2. Pour toute règle $(T \to m_1 T_1 \ldots m_p T_p m_{p+1})$ de P, où, $1 \leq j \leq p$, $T_j \in V_k$ et $m_j \in ((X \cup V)\backslash V_k)^*$, on met dans P' les règles $\{T\} \times \{m_1 W_1 m_2 \ldots m_p W_p m_{p+1}\}$ avec $W_j = \{T_j, \mathbb{1}\}$, $1 \leq j \leq p$
   (excepté si $m_1 W_1 m_2 \ldots m_p W_p m_{p+1}$=𝟙).

Il est évident que P' ne contient aucune règle de la forme $T \to \mathbb{1}$; d'autre part, il faut vérifier que, pour tout T de V, $L(G', T) = L(G, T)\backslash\{\mathbb{1}\}$ (par récurrence sur l'ordre de dérivation).

2- La relation de dérivation restreinte à V définit un préordre sur V. A ce préordre nous associons la relation d'équivalence, notée ≡, définie par :

$$\forall T, T' \in V \quad T \equiv T' \Leftrightarrow T \xrightarrow[G]{*} T' \ et \ T' \xrightarrow[G]{*} T$$

(notons que $T \equiv T'$ est vérifiable, pour tous T et T', en un nombre fini de pas)

On note $\bar{V}$ l'ensemble quotient $V/\equiv$ et on considère cet ensemble comme nouvel alphabet de symboles non-terminaux. On en déduit la définition d'un nouvel ensemble de productions $\bar{P}'$:

$$\begin{array}{l} \bar{T} \ \to m_1 \bar{T}_1 m_2 \ldots m_p \bar{T}_p m_{p+1} \ \in \ \bar{T}' \\ \qquad\qquad \Leftrightarrow \\ T \ \to m_1 T_1 m_2 \ldots m_p T_p m_{p+1} \ \in \ T' \end{array}$$

Posant $\bar{G}' = (X, \bar{V}, \bar{S}, \bar{P}')$ on a, pour tout t de V, $L(\bar{G}', \bar{T}) = L(G', T) = L(G, T) \backslash \{\mathbb{1}\}$.
Dans $\bar{V}$ on peut noter $\leq$ l'ordre quotient $\rightarrow^*/\equiv$ et l'on a la proposition :

$$(\bar{T} \rightarrow \bar{T}') \in \bar{P}' \;\Rightarrow\; \bar{T} \leq \bar{T}'$$

On peut tout d'abord supprimer toutes les règles de la formes $(\bar{T} \rightarrow \bar{T})$ qui n'apportent rien, si bien que maintenant on a, avec $<$ l'ordre strict associé à $\leq$:

$$(\bar{T} \rightarrow \bar{T}') \in \bar{P}' \;\Rightarrow\; \bar{T} < \bar{T}'$$

Pour obtenir les règles finales de $\bar{G}$, il suffit de prendre les variables dans l'ordre décroissant et de remplacer pour chacune des variables $\bar{T}$ toute règle $(\bar{T} \rightarrow \bar{T}')$ par l'ensemble des règles :

$$\{(\bar{T} \rightarrow m) \mid (\bar{T}' \rightarrow m) \in \bar{P}'\}$$

Note : L'ordre dans lequel on effectue les deux opérations de suppression des règles $(T \rightarrow \mathbb{1})$ et des règles $(T \rightarrow T')$ n'est pas indifférent. En effet, la procédure employée pour éliminer les règles $(T \rightarrow \mathbb{1})$ peut introduire des règles de la forme $(T \rightarrow T')$.

Exemple 41 *Soit la grammaire $G = (X, V, S, P)$ où :*

$$P = \begin{cases} S & \rightarrow & aAb \\ A & \rightarrow & B \mid aAb \mid AA \\ B & \rightarrow & x \mid C \mid D \\ C & \rightarrow & \mathbb{1} \\ D & \rightarrow & x \mid B \end{cases}$$

1. *Elimination des règles de la forme $(T \rightarrow \mathbb{1})$:*
   On construit la suite des parties de V, définie par :

$$V_0 = \{T \in V \mid (T \rightarrow \mathbb{1}) \in P\}$$

$$V_{i+1} = \{T \in V \mid (T \rightarrow m) \in P,\ m \in V_i^+\} \cup V_i$$

   On obtient la suite suivante :

$$\begin{array}{rcl} V_0 & = & \{C\} \\ V_1 & = & \{B, C\} \\ V_2 & = & \{A, B, C, D\} \\ V_3 & = & \{A, B, C, D\} \end{array}$$

   On en déduit la grammaire $G' = (X, V, S, P')$ où:

$$P' = \begin{cases} S & \rightarrow & aAb \mid ab \\ A & \rightarrow & B \mid aAb \mid ab \mid AA \mid A \\ B & \rightarrow & x \mid C \mid D \\ D & \rightarrow & x \mid B \end{cases}$$

2. *Elimination des règles de la forme* $(T \rightarrow T')$ *:*
La relation d'ordre partiel induite par la relation de dérivation restreinte à V *est la suivante :*

$$A \leq B,\ B \leq C,\ B \leq D,\ D \leq B$$

On en déduit : $B \equiv D$*. On construit alors la grammaire* $\bar{G}' = (X, \bar{V}, \bar{S}, \bar{P}')$ *où :*

$$\bar{P}' = \begin{cases} \bar{S} & \rightarrow & a\bar{A}b \mid ab \\ \bar{A} & \rightarrow & \bar{B} \mid a\bar{A}b \mid ab \mid \bar{A}\bar{A} \\ \bar{B} & \rightarrow & x \mid \bar{C} \mid \bar{B} \end{cases}$$

On a maintenant une relation d'ordre strict : $\bar{A} < \bar{B} < \bar{C}$*. Toutes les règles* $(T \rightarrow T)$ *sont supprimées. Toutes les règles* $(T \rightarrow T')$ *sont remplacées par* $\{(T \rightarrow m \mid (T' \rightarrow m) \in \bar{P}'\}$*. Si un non-terminal ne se dérive pas, on ote les parties droites où il apparaît. On obtient ainsi la grammaire* $\bar{G} = (X, \bar{V}, \bar{S}, \bar{P})$ *où :*

$$\bar{P} = \begin{cases} \bar{S} & \rightarrow & a\bar{A}b \mid ab \\ \bar{A} & \rightarrow & x \mid a\bar{A}b \mid ab \mid \bar{A}\bar{A} \end{cases}$$

2.6.2 Grammaires réduites

Définition 42 (grammaire réduite) *Une grammaire G=(X,V,S,P) est dite réduite vis à vis de S si, et seulement si, les deux conditions suivantes sont satisfaites :*

1. $\forall T \in V\ L(G,T) \neq \emptyset$
2. $\forall T \in V\ \hat{L}(G,S) \cap (X \cup V)^* T (X \cup V)^* \neq \emptyset$

Proposition 43 *Un langage L sur* X^* *est algébrique si, et seulement si, il existe une grammaire algébrique G=(X,V,S,P), réduite vis à vis de S, qui l'engendre.*

2.6.3 Forme normale de Greibach

Définition 44 *Une grammaire algébrique G=(X,V,S,P) est dite sous forme normale de Greibach si* $P \subseteq V \times XV^*$*.*

Théorème 45 (Greibach) *Pour tout langage algébrique de* X^+*, il existe une grammaire G=(X,V,S,P) sous forme normale de Greibach telle que L=L(G,S).*

Preuve : Elle est longue et technique. Si L $\subseteq X^+$ est un langage algébrique, L est égal à L(G,S) pour une certaine grammaire algébrique propre G=(X,V,S,P). On numérote les variables: V=$\{S_i \mid 1 \leq i \leq l\}$ et on supposera que S_1 est l'axiome. D'autre part P sera représenté par un ensemble de couples (S_j, p_j) où p_j est une partie de $(X \cup V)^*$(p_j regroupe tous les membres droits de S_j).

A partir de G on construit la grammaire $\bar{G} = (X, \bar{V}, \bar{P})$ où $\bar{V} = V \cup V'$ (V'=$\{S'_j \mid 1 \leq j \leq l\}$) et dont les règles $\bar{P}$ sont un ensemble de (l+l) couples $(S_j, \bar{p}_j)$ et $(S'_j, \bar{p}'_j)$ avec $\bar{p}_j, \bar{p}'_j$ éléments de $(X \cup \bar{V})^*$.

Les parties $\bar{p}_j$ *et* $\bar{p}'_j$ sont définies par récurrence à partir de p_j de la façon suivante:

- $\forall j\ 1 \leq j \leq l\ \bar{p}_j^{(0)} = p_j$
- $\forall i\ 1 \leq i \leq l$ **si** $\bar{p}_i^{(i-1)} = S_i m + m'$
  avec
  $$m' \cap S_i(X \cup \bar{V})^* = \emptyset$$
  alors
  $$\bar{p}_i^{(i)} = m' S'_i + m' \text{ et } \bar{p}_i^{\prime(0)} = m S'_i + m$$
- $\forall i\ 1 \leq i \leq l\ \forall j\ 1 \leq j \leq l\ j \neq i$, **si** $\bar{p}_j^{(i-1)} = S_i m + m'$
  avec
  $$m' \cap S_i(X \cup V)^* = \emptyset$$
  alors
  $$\bar{p}_j^{(i)} = \bar{p}_i^{(i)} m + m'$$
- $\forall j\ 1 \leq j \leq l\ \bar{p}_j = \bar{p}_j^{(l)}$
- $\forall j\ 1 \leq j \leq l$ **si** $\bar{p}_j^{\prime(0)} = \sum_{k=1}^{l} S_k m_k + \sum_{k=1}^{j-1} S'_k m'_k + m$
  avec
  $$m \cap (V \cup \{S'_1, \ldots, S'_{j-1}\})(X \cup \bar{V})^* = \emptyset$$
  alors
  $$\bar{p}'_j = \sum_{k=1}^{l} \bar{p}_k m_k + \sum_{k=1}^{j-1} \bar{p}'_k m'_k + m$$

Lemme 46 $\forall j\ 1 \leq j \leq l\ \bar{p}_j \subseteq X(X \cup \bar{V})^*$

Par récurrence sur i, on montre que : $\forall i\ 1 \leq i \leq l,\ \forall j\ 1 \leq j \leq l$:

$$\begin{array}{ll} \forall k\ \ 1 \leq k \leq i & \bar{p}_j^{(i)} \cap S^k(X \cup \bar{V})^* = \emptyset \\ \forall k\ \ 1 \leq k \leq l & \bar{p}_j^{(i)} \cap S'^k(X \cup \bar{V})^* = \emptyset \\ & \bar{p}_j^{(i)} \cap (\bar{V} \cup \{\mathbb{1}\}) = \emptyset \end{array}$$

Lemme 47 $\forall j\ 1 \leq j \leq l\ \bar{p}'_j \subseteq X(X \cup \bar{V})^*$

Par récurrence sur j.

Pour que $\bar{G}$ soit une grammaire sous forme normale de Greibach on lui ajoute les variables $\{S_x \mid x \in X\}$, les règles $(S_x \to x)$ et on substitue S_x aux x qui ne sont pas en début de membres droits dans tous les $\bar{p}_j$ et les $\bar{p}'_j$. Il reste alors à montrer que :

$$\forall j\ 1 \leq j \leq l\ \ L(\bar{G}, S_j) = L(G, S_j),$$

ce qui découle de la constatation suivante : si on remplace les règles $(S_i \to S_i m_i + m'_i)$ par $(S_i \to m'_i S'_i + m'_i)$ et $(S'_i \to m_i S'_i + m_i)$, L(G,S_i) reste égal à $L(G, m'_i).(L(G, m_i))^*$. Toutes les autres opérations que l'on effectue sont des substitutions qui conservent trivialement le langage engendré.

Pour faire passer cette construction quelque peu indigeste, examinons un exemple.

Exemple 48 *Soit la grammaire G définie par X={a,b}, V={S,T} et les règles :*

$$\begin{cases} 1 & S & \to & SaT + TT + bb \\ 2 & T & \to & TS + a \end{cases}$$

On a donc les deux équations suivantes :

- $\bar{p}_1^{(0)} = SaT + TT + bb$
- $\bar{p}_2^{(0)} = TS + a$

Pour i=1 :

- $\bar{p}_1^{(1)} = (TT + bb)S' + TT + bb = TTS' + bbS' + TT + bb$
- $\bar{p}_2^{(1)} = TS + a$
- $\bar{p}_1'^{(0)} = aTS' + aT$

Pour i=2 :

- $\bar{p}_2 = \bar{p}_2^{(2)} = aT' + a$
- $\bar{p}_2'^{(0)} = ST' + S$
- $\bar{p}_1 = \bar{p}_1^{(2)} = (aT' + a)(TS' + T) + bbS' + bb$
  $= aT'TS' + aT'T + aTS' + aT + bbS' + bb$
- $\bar{p}'_1 = aTS' + aT$
- $\bar{p}'_2 = (aT'TS' + aT'T + aTS' + aT + bbS' + bb)(T' + \mathbb{1})$
  $= aT'TS'T' + aT'TT' + aTS'T' + aTT' + bbS'T' + bbT' + aT'TS' + aT'T + aTS' + aT + bbS' + bb$

D'où la grammaire sous forme normale de Greibach :

$$\begin{cases} S & \to & aT'TS' + aT'T + aTS' + aT + bBS' + bB \\ T & \to & aT' + a \\ S' & \to & aTS' + aT \\ T' & \to & aT'TS'T' + aT'TT' + aTS'T' + aTT' + bBS'T' + bBT' \\ & & + aT'TS' + aT'T + aTS' + aT + bBS' + bB \\ B & \to & b \end{cases}$$

Exercices :

Exercice 1 :
Soit n un entier fixé. Soit la grammaire G=(X,V,S,P) où X=$\{a_1, \ldots, a_n, \neg, \vee, \wedge, [,]\}$, $V = \{S\}$ et P=$\{S \longrightarrow a_i \mid i = 1, \ldots, n\} \cup \{S \longrightarrow \neg S, S \longrightarrow [S \wedge S], S \longrightarrow [S \vee S]\}$.
Qu'est-ce que L(G,S)?

Exercice 2 :
Soit la grammaire $G = (\{0, 1\}, \{S, T\}, S, P)$ où :

$$P = \begin{cases} S & \rightarrow & 0S \mid \ 1S \mid \ 0T \\ T & \rightarrow & 0 \end{cases}$$

Montrer que le langage engendré est $\{w00 \mid \ w \in \{0, 1\}^*\}$

Exercice 3 :
Soit la grammaire G=(X,V,S,P) où X=$\{$a,b$\}$, V=$\{$S$\}$ et P=$\{$S $\longrightarrow$ aaS, S $\longrightarrow$ b$\}$. Montrer que L(G,S)=$\{a^{2n}b \mid n \in \mathrm{N}\}$.

Exercice 4 :
Si X est un alphabet et u $\in X^*$, on appelle mot miroir de $u = x_1x_2 \ldots x_{p-1}x_p$, et l'on note $\tilde{u}$, le mot $\tilde{u} = x_px_{p-1} \ldots x_2x_1$ où, pour tout i, x_i est une lettre de x.

Montrer que les langages suivants sont algébriques :

1. $L_1 = \{uc\tilde{u} \mid u \in \{a, b\}^*\}$
2. $L_2 = \{u\tilde{u} \mid u \in \{a, b\}^*\}$
3. $L_3 = \{u \in \{a, b\}^* \mid u = \tilde{u}\}$
4. $L_4 = \{uc\tilde{v} \mid u, v \in \{a, b\}^* \ et \ u \neq v\}$

Exercice 5 :
Soit l'alphabet X=$\{$0,1,2,3,4,5,6,7,8,9, . ,: ,; , (,) $\}$. On désire décrire par une grammaire les mots de X^* composés d'une suite de couples d'entiers séparés de points-virgules, les deux entiers d'un même couple étant séparés de deux points, le dernier couple de la suite étant suivi d'un point. Donner une telle grammaire.

Exercice 6 :
Soit G une grammaire réduite vis-à-vis de la variable V_0. Montrer que L(G,V_0) est infini si, et seulement si, il existe une variable V et deux mots α et β appartenant à X^* et non tous les deux vides tels que $V \xrightarrow[G]{*} \alpha V \beta$.

Exercice 7 :
Mettre sous forme normale de Greibach les grammaires définies par les ensembles de règles suivants :

1. $\begin{array}{lcl} S_1 & \longrightarrow & S_1aS_2 \ + \ S_2S_2 \ + \ b \\ S_2 & \longrightarrow & S_2d \ + \ S_2S_1a \ + \ aS_1 \ + \ c \end{array}$

2.
$$\begin{array}{lcl} S & \longrightarrow & AA \ + \ a \\ A & \longrightarrow & SS \ + \ b \end{array}$$

3.
$$\begin{array}{lcl} S_1 & \longrightarrow & S_1S_1 \ + \ S_1S_2 \ + \ S_2S_3 \ + \ S_2b \\ S_2 & \longrightarrow & S_2S_1 \ + \ S_1S_2a \ + \ S_3S_3 \ + \ S_1a \\ S_3 & \longrightarrow & S_1S_3 \ + \ S_2S_3a \ + \ S_3S_1 \ + \ a \end{array}$$

Exercice 8 :

Montrer que si L est un langage algébrique, il en est de même de $\tilde{L}$, ensemble des mots miroirs des mots de L.

Exercice 9 :

Donner une grammaire des nombres réels du langage Pascal.

Exercice 10 :

Quel est le langage engendré par la grammaire suivante ?
$G = (\{a, b\}, \{S, A, B\}, S, P)$ où :

$$P = \left\{ \begin{array}{lcl} S & \rightarrow & aB \mid bA \\ A & \rightarrow & a \mid aS \mid bAA \\ B & \rightarrow & b \mid bS \mid aBB \end{array} \right.$$

Indication : on devra faire une preuve en parallèle sur les trois langages L(G,S), L(G,A) et L(G,B).

Chapitre 3

Automates finis

Nous allons dans les deux chapitres suivants étudier sous divers points de vue la même famille de langages, appelés selon les cas langages reconnaissables, rationnels, voire réguliers. Nous introduirons plusieurs types d'outils pour définir et manipuler ces langages. Selon les cas nous aurons intérêt à utiliser tel outil plutôt que tel autre, mais nous aurons auparavant vérifié qu'ils sont tous équivalents.

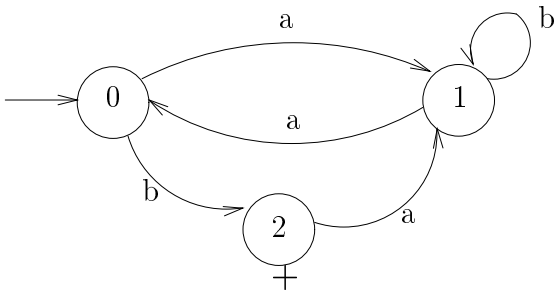
3.1 Automates finis et langages reconnaissables

Définition 49
Un automate fini $\mathcal{A}$ est défini par un quintuplet $\mathcal{A} = (X, Q, q_0, A, \delta)$ où:

- *X est un alphabet*
- *Q est un ensemble fini de symboles : les états de l'automate*
- *q_0 est un état distingué de Q, l' état initial*
- *A est un sous-ensemble de Q, les états terminaux*
- *δ est une application de $Q \times X$ dans $\mathcal{P}(Q)$ appelée fonction de transition*

Traditionnellement, les automates sont dessinés sous forme de graphe orienté dont les sommets sont les états de l'automate et les arcs représentent les transitions de l'automate. Plus précisément :

Définition 50
Soit X un alphabet, un automate sur X est un graphe $G=(Q,\Gamma, q_0, A)$, dont Q est l'ensemble des sommets, Γ l'ensemble des arcs étiquetés, i.e. une partie de $Q \times X \times Q$, le triple (q_1, x, q_2) signifiant qu'il existe un arc de q_1 à q_2 étiqueté par la lettre x.



Par convention on peut déduire les informations suivantes à la lecture de ce graphe :

- X={a,b}
- Q={0,1,2}
- q_0=0
- A={2}
- δ(0,a)={1} etc...

Pour assurer la cohérence des deux définitions il suffit, si $\mathcal{A}$ est donné sous la forme $\mathcal{A} = (X, Q, q_0, A, \delta)$, de prendre pour Γ l'ensemble $\{(q_1, x, q_2) \mid q_1, q_2 \in Q,$
$x \in X, q_2 \in \delta(q_1, x)\}$. Pour cette raison, on verra parfois δ comme un sous-ensemble de $Q \times X \times Q$, les deux notations cohabitant dans la suite de ce texte. De la terminologie de la théorie des graphes, nous empruntons les notions de chemin, de cycle, de longueur d'un chemin, de début et de fin d'un chemin. Bien entendu étant donné un chemin, il est légitime de parler de l'étiquette de ce chemin, on dira plutôt la trace de ce chemin, cette trace étant le mot de X^* obtenu par concaténation des étiquettes des arcs constituant ce chemin.

Définition 51 (mot reconnu)
Un mot f de X^* est reconnu par un automate $\mathcal{A}$ si, et seulement si :

$$\exists q \in A,\ \exists \text{ un chemin de } q_0 \text{ à } q \text{ de trace } f$$

Définition 52 (langage reconnu)
Le langage reconnu par un automate $\mathcal{A}$, noté $L(\mathcal{A})$, est l'ensemble des mots f de X^* reconnus par $\mathcal{A}$.

Définition 53 (langage reconnaissable)
Un langage L est reconnaissable s'il existe un automate fini $\mathcal{A}$ tel que L=$L(\mathcal{A})$.
La famille des langages reconnaissables sur X^* est notée $Rec(X^*)$.

3.2 Propriétés de $Rec(X^*)$

Remarque :

- Le langage vide est reconnaissable.
- Pour tout alphabet X, X^* est reconnaissable.
- Tout ensemble fini de mots est reconnaissable.

Proposition 54 *La famille $Rec(X^*)$ est close par union.*

Preuve : Soient L et M deux langages reconnaissables.
Il existe alors un automate $\mathcal{A}_1 = (X_1, Q_1, q_1, A_1, \delta_1)$ reconnaissant L et un autre automate $\mathcal{A}_2 = (X_2, Q_2, q_2, A_2, \delta_2)$ reconnaissant M. A un renommage-près, on suppose que $Q_1 \cap Q_2$ est vide. Soit q_0 un nouveau symbole.
Posons alors $\mathcal{A} = (X_1 \cup X_2, Q_1 \cup Q_2 \cup \{q_0\}, q_0, A_1 \cup A_2 \cup F, \delta)$, avec

- $\delta = \delta_1 \cup \delta_2 \cup \{(q_0, x, q) \mid (q_1, x, q) \in \delta_1 \text{ ou } (q_2, x, q) \in \delta_2\}$
  i.e. on duplique toutes les flèches partant de l'état initial de l'un ou l'autre des deux automates,
- $F = \emptyset$ *si* $q_1 \notin A_1$ *et* $q_2 \notin A_2, F = \{q_0\}$ sinon,
  de façon à reconnaître le mot vide s'il est dans l'un des deux langages.

On laisse au lecteur le soin de prouver l'égalité $L \cup M = L(\mathcal{A})$

Proposition 55 *La famille $Rec(X^*)$ est close par produit, étoile, complémentation, intersection.*

3.3 Equivalence automates finis - grammaires linéaires droites

Nous allons montrer que les langages reconnaissables sont également algébriques, et donc que la famille $Rec(X^*)$ est une sous-famille de la famille $Alg(X^*)$. Rappelons la définition suivante :

Définition 56 (grammaire linéaire droite)
Soit G=(X,V,S,P) une grammaire, elle est dite linéaire droite si, et seulement si :

$$P \subseteq V \times X^*(V \cup \mathbb{1})$$

Proposition 57 *Soit $\mathcal{A}$ un automate fini, il existe une grammaire linéaire droite G telle que L($\mathcal{A}$)=L(G,S).*

Preuve : Soit $\mathcal{A} = (X, Q, q_0, A, \delta)$ l'automate, on construit la grammaire G de la façon suivante :

- V=$\{S_q \mid q \in Q\}$, i.e. V est un alphabet de symboles indexés par l'ensemble des états de $\mathcal{A}$.
- P contient les règles suivantes :

$$\begin{array}{rcl} S_q & \rightarrow & xS_{q'} \quad \Leftrightarrow \quad q' \in \delta(q,x) \\ S_q & \rightarrow & \mathbb{1} \quad \Leftrightarrow \quad q \in A \end{array}$$

- l'axiome S est S_{q_0}
- X est le même alphabet que celui de $\mathcal{A}$

G=(X,V,S,P) est à l'évidence une grammaire linéaire droite. Il faut montrer l'équivalence logique :
$\forall f \in X^*,\ S \xrightarrow[G]{*} f \ \Leftrightarrow$ il existe un chemin de q_0 à $q \in A$ de trace f.
Pour ce faire, on montre une propriété légèrement plus forte que la précédente :
$\forall f \in X^*,\ \forall q \in Q, S_q \xrightarrow[G]{*} f \ \Leftrightarrow$ il existe un état q'$\in A$ et un chemin dans $\mathcal{A}$ de q à q' de trace f.
Ceci se montre par récurrence sur la longueur du mot f.

De façon analogue on peut prouver la proposition réciproque :

Proposition 58 *Soit G une grammaire régulière, il existe un automate fini $\mathcal{A}$ tel que L(G,S)=L(A).*

3.4 Formes d'automates

On va définir maintenant certaines formes d'automates permettant ainsi de faciliter la preuve de certains résultats sur les langages reconnaissables.

3.4.1 Automates finis déterministes

Définition 59 (automate déterministe complet)

- *Un automate fini $\mathcal{A} = (X, Q, q_0, A, \delta)$ est déterministe si et seulement si :*

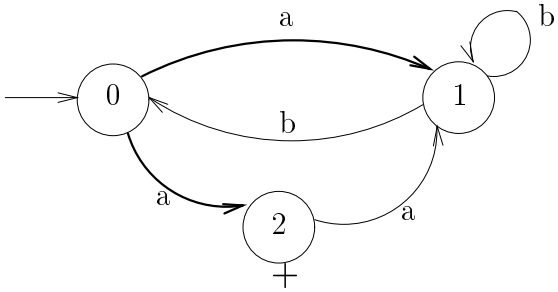
$$\forall q \in Q, \forall x \in X, Card(\{q' \in Q \mid q' \in \delta(q,x)\}) \leq 1$$

i.e dans le graphe associé il part de chacun des états au plus une flèche étiquetée x, ceci pour toute lettre x.

- *Un automate fini* $\mathcal{A} = (X, Q, q_0, A, \delta)$ *est complet si et seulement si :*

$$\forall q \in Q, \forall x \in X, Card(\{q' \in Q \mid q' \in \delta(q, x)\}) \geq 1$$

- *Un automate est déterministe complet si et seulement si il est à la fois déterministe et complet(!), c'est-à-dire que de chaque état part une et une seule flèche étiquetée x.*

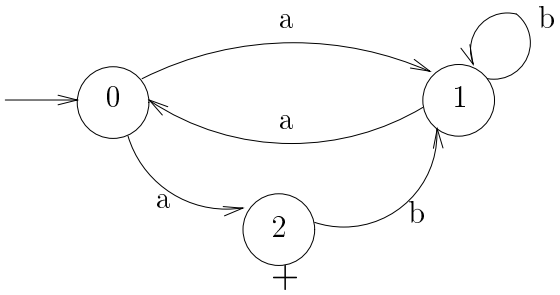


Automate non déterministe.

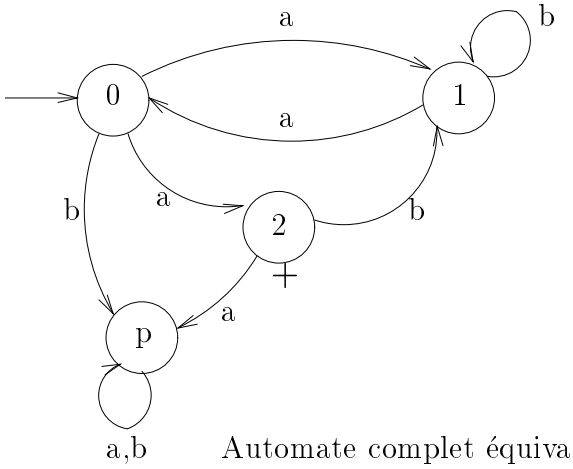
Proposition 60 *Pour tout automate* $\mathcal{A} = (X, Q, q_0, A, \delta)$ *il existe un automate complet équivalent* $\mathcal{B}$*, c'est-à-dire tel que* $L(\mathcal{A})$=$L(\mathcal{B})$.

Preuve :
Soit $\mathcal{A} = (X, Q, q_0, A, \delta)$ un automate non complet. Soit p un nouvel état, appelé état puits. On construit l'automate suivant $\mathcal{B} = (X, Q \cup \{p\}, q_0, A, \delta_1)$, où la nouvelle fonction de transition est définie par :

$$\begin{aligned} \delta_1 \;=\; \delta \;& \cup \{(q, x, p),\ q \in Q,\ x \in X,\ Card(\{q' \in Q \mid q' \in \delta(q, x)\}) = 0\} \\ & \cup \{(p, x, p),\ x \in X\} \end{aligned}$$



Automate non complet.



Automate complet équivalent.

Par la suite on dira déterministe pour déterministe complet.

Propriété :
Si $\mathcal{A}$ est un automate déterministe complet, la fonction de transition δ peut être vue comme une application de Q × X dans Q plutôt que de Q × X dans $\mathcal{P}$(Q), en identifiant tout singleton à son unique élément.

On étend δ en une application $\hat{\delta}$ de $Q \times X^*$ dans $\mathcal{P}(Q)$ de la façon suivante :

- $\forall q \in Q,\ \hat{\delta}(q,\mathbb{1}) = q$
- $\forall q \in Q,\ \forall f \in X^*,\ \forall x \in X,\ \hat{\delta}(q, xf) = \hat{\delta}(\delta(q,x), f)$

Remarques :

- $\forall q \in Q,\ \forall x \in X,\ \hat{\delta}(q,x) = \hat{\delta}(q, x\mathbb{1}) = \hat{\delta}(\delta(q,x),\mathbb{1}) = \delta(q,x)$
- Lorsqu'un automate est déterministe complet, pour tout état q et tout mot u, il existe un unique état q' tel qu'il existe un chemin de q à q' de trace u. Ce chemin est lui aussi unique, et l'on constate que $q' = \hat{\delta}(q,u)$.
  De là, on déduit trivialement : $L(\mathcal{A})$=$\{f \mid \hat{\delta}(q_0, f) \in A\}$.
  Ceci permet de décider effectivement pour chaque mot f si il est reconnu ou non.

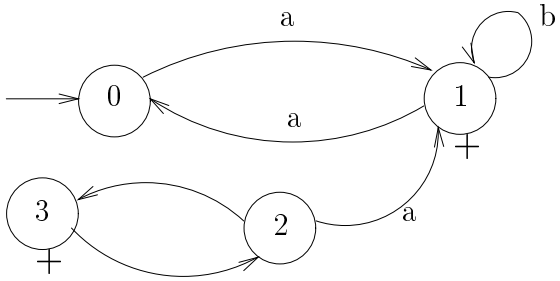
– $\forall q \in Q,\ \forall u, v \in X^*,\ \hat{\delta}(q, uv) = \hat{\delta}(\hat{\delta}(q, u), v)$. Ceci se montre par récurrence sur la longueur de u.

3.4.2 Automates accessibles, co-accessibles

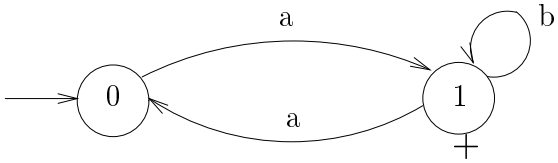
Définition 61 (automate accessible, co-accessible)
Un automate $\mathcal{A} = (X, Q, q_0, A, \delta)$ est dit accessible si, et seulement si, $\forall q \in Q, \exists f \in X^*, \hat{\delta}(q_0, f) = q$. Un automate $\mathcal{A} = (X, Q, q_0, A, \delta)$ est dit co-accessible si, et seulement si, $\forall q \in Q,\ \exists f \in X^*\ \hat{\delta}(q, f) \in A$. Un automate est dit net si, et seulement si, il est à la fois accessible et co-accessible.

Proposition 62 *Soit un automate quelconque $\mathcal{A} = (X, Q, q_0, A, \delta)$, il existe un automate net $\mathcal{A}_1 = (X, Q_1, q_0, A_1, \delta_1)$ tel que $L(\mathcal{A})$=$L(\mathcal{A}_1)$*

Preuve : Elle se fait en deux temps : d'abord construction d'un automate accessible $\mathcal{A}'$ tel que $L(\mathcal{A})$=$L(\mathcal{A}')$, puis à partir du nouvel automate, construction d'un automate $\mathcal{A}_1$, co-accessible qui reste accessible et qui est donc l'automate désiré. La construction détaillée est laissée au lecteur.



Automate co-accessible mais non accessible.



Automate net équivalent.

On remarquera que si l'on est parti d'un automate déterministe $\mathcal{A}$ la construction renvoie un automate résultat également déterministe, mais pas nécessairement complet. Désormais, nous admettrons que l'automate utilisé est accessible, co-accessible ou net selon les besoins du moment.

3.4.3 Automates standards

Définition 63 *Un automate déterministe est dit standard si et seulement si :*

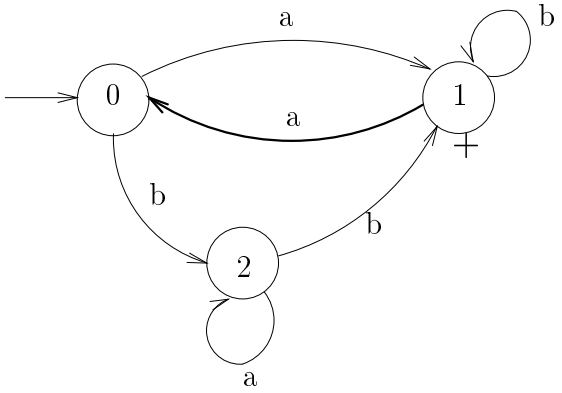
$$\forall q \in Q, \forall x \in X, \delta(q,x) \neq q_0$$

Proposition 64 *Soit un automate déterministe quelconque $\mathcal{A}$, il existe un automate standard $\mathcal{A}_s$ tel que $L(\mathcal{A})$=$L(\mathcal{A}_s)$*

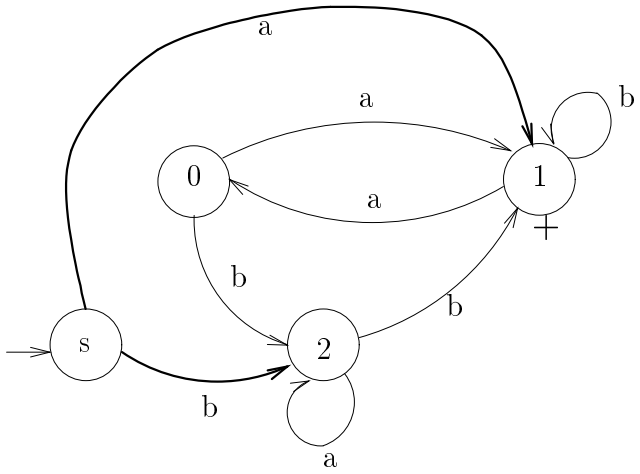
Preuve : Soit $\mathcal{A} = (X, Q, q_0, A, \delta)$ un automate déterministe. Si A n'est pas standard, on choisit un nouvel état s et l'on pose :

- $Q_s = Q \cup \{s\}$
- $A_s = A$ *si* $q_0 \notin A, A \cup \{s\}$ *sinon*
- $\delta_s = \delta \cup \{(s,x,q),\ (q_0,x,q) \in \delta\}$

On vérifie que l'automate $\mathcal{A}_s = (X, Q_s, s, A_s, \delta_s)$ convient.



Automate non standard.



Automate standard équivalent.

3.5 Equivalence automates finis - automates déterministes

Proposition 65 *Un langage est reconnaissable si et seulement si il est reconnu par un automate déterministe.*

Preuve : Soit L un langage reconnaissable, il existe donc un automate $\mathcal{A} = (X, Q, q_0, A, \delta)$ tel que L=$L(\mathcal{A})$. Construisons l'automate déterministe $\mathcal{A}'$ dont les états sont les éléments de $\mathcal{P}$(Q).

$$\mathcal{A}' = (X, \mathcal{P}(Q), \{q_0\}, A', \delta')$$

avec :

- A'=$\{R \mid R \in \mathcal{P}(Q),\ R \cap A \neq \emptyset\}$)
- δ' est une application de $\mathcal{P}(Q) \times X$ dans $\mathcal{P}(Q)$ qui au couple (R,x) associe le sous-ensemble de Q, $\delta'(R, x)$, défini par :

$$\delta'(R, x) = \bigcup_{q \in R} \delta(q, x)$$

Cet automate est déterministe, encore faut-il montrer qu'il reconnaît $L(\mathcal{A})$. La preuve est laissée au lecteur.

3.6 Propriétés de clôture des reconnaissables

Nous avons annoncé dans le chapître précédent la fermeture de $Rec(X^*)$ par union, produit, étoile, intersection et complémentation. Pour la clôture par étoile, nous allons utiliser l'une des formes d'automates introduites dans le présent chapître.

Proposition 66 *$Rec(X^*)$ est fermée pour l'opération étoile.*

Preuve : Soit L $\in Rec(X^*)$, et soit $\mathcal{A} = (X, Q, q_0, A, \delta)$ un automate standard qui reconnait L (on sait qu'un tel automate existe toujours). Posons $\mathcal{A}^* = (X, Q, q_0, A \cup \{q_0\}, \delta^*)$, où $\delta^* = \delta \cup \{(q, x, q_0) \mid \exists q' \in A, (q, x, q') \in \delta\}$, i.e. on duplique vers le nouvel état de départ les flèches menant à un état final de l'ancien automate $\mathcal{A}$. On vérifie aisément que ce nouvel automate reconnait $L(\mathcal{A})$, on peut cependant constater qu'il n'est plus en général déterministe.

Les preuves des clôtures par les autres opérations sont laissées en exercice.

3.7 Lemme de l'étoile

Proposition 67 *Soit $\mathcal{A} = (X, Q, q_0, A, \delta)$ un automate net, le langage $L(\mathcal{A})$est fini si et seulement si il n'y a pas de circuits (dans l'automate) de longueur strictement positive.*

Preuve : L'existence d'un circuit signifie que :

$$\exists q \in Q,\ \exists f \in X^+,\ q \in \hat{\delta}(q, f)$$

Puisque l'automate est net, il existe u,$v \in X^*$ tels que $q \in \hat{\delta}(q_0, u)$ (l'état q est accessible) et $q_+ \in A$ avec $q_+ \in \hat{\delta}(q, v)$ (l'état q est co-accessible). Par suite, le mot ufv appartient à $L(\mathcal{A})$, mais il en est de même pour uf^2v, uf^3v, d'une façon générale $uf^*v \subseteq L(\mathcal{A})$ et donc $L(\mathcal{A})$ est infini.

Réciproquement si $L(\mathcal{A})$ est infini, alors il contient un mot f dont la longueur est strictement supérieure au nombre d'états de Q. Par conséquent lors de la reconnaissance de ce mot, l'automate passe deux fois par le même état et donc il existe un circuit.

Plus précisément nous avons :

Lemme 68 (dit de l'étoile)
Soit L une partie de X^* reconnaissable, il existe un entier N tel que pour tout mot f de X^*, si $f \in L$ et $\mid f \mid \geq N$ alors $\exists \alpha,\ u,\ \beta \in X^*$ tels que :

$$u \neq \mathbb{1},\quad f = \alpha u \beta \text{ et } \forall n \in \mathrm{N}\ \alpha u^n \beta \subseteq L$$

Il existe une autre version de ce lemme imposant la contrainte supplémentaire $\mid u \mid \leq N$.

Preuve : Découle de la proposition précédente.
Remarque importante : L'intérêt de ce lemme de l'étoile est de permettre de montrer que de nombreux langages ne sont pas reconnaissables.

Exercices :
Exercice 1 :
Soit $X = \{a, b\}$.
Donner un automate reconnaissant :

- les mots de X^* de longueur paire,
- les mots de X^* contenant le facteur aa,
- les mots de X^* ne contenant pas le facteur aa,
- les mots contenant un nombre pair de a et impair de b.

Exercice 2 :
Soit $X = \{0, 1\}$.
Donner un automate reconnaissant les mots de X^* correspondant à l'écriture en base 2 d'un entier multiple de 3. Généraliser en prenant $X = \{0, 1, ..., b-1\}$ et les mots écrits en base b qui sont des multiples de l'entier q.
Indication : cet automate aura exactement q états, pour $q \geq 0$.

Exercice 3 :
Donner un automate fini reconnaissant l'ensemble des nombres réels du langage Pascal.

Exercice 4 :
Montrer que $Rec(X^*)$ est close par produit. Montrer que $Rec(X^*)$ est close par intersection.

Exercice 5 :
Montrer que si L est un langage reconnaissable, il en est de même de FG(L), ensemble des facteurs gauches des mots de L. Idem avec FD(L) et F(L), facteurs droits et facteurs.

Exercice 6 :
Montrer que $Rec(X^*)$ est close par miroir.

Exercice 7 :
Soit L_1 un langage reconnaissable et L_2 un langage non reconnaissable, tels que $L_1 \cap L_2 = \emptyset$. Montrer que $L_1 \cup L_2$ n'est pas reconnaissable. On utilisera les propriétés de clôture de $Rec(X^*)$.
Application :
Admettons que le langage $\{a^n b^n, n \geq 0\}$ n'est pas reconnaissable. La justification en sera donnée au chapitre suivant.
Montrer que $L = \{a^n b^m \mid 0 \leq n, m,\ n \neq m\}$ n'est pas reconnaissable.

Exercice 8 :
Donner la construction d'un automate accessible équivalent à un automate donné. Par équivalent on entend que les langages reconnus par les deux automates sont identiques.

Exercice 9 :
Montrer que $Rec(X^*)$ est fermée par complémentation. Pour cette preuve il est fondamental de considérer des automates déterministes complets.

Exercice 10 :
Donner un automate déterministe reconnaissant l'ensemble des mots sur $\{a, b\}^*$ contenant le facteur $abaab$.

Exercice 11 :
Donner un automate déterministe reconnaissant l'ensemble des mots sur $\{a, b\}^*$ contenant au moins trois occurrences de la lettre a.

Lemme de l'étoile :
Exercice 12 :
Montrer que le langage $\{a^n b^n, n \geq 0\}$ n'est pas reconnaissable.

Exercice 13 :
Montrer que le langage $\{a^p, p$ est un nombre premier$\}$ n'est pas reconnaissable.

Exercice 14 :
Montrer que le langage $\{a^p, p$ est une puissance de 2$\}$ n'est pas reconnaissable.

Exercice 15 :
Montrer que le langage $L = \{u\tilde{u},\ u \in \{a, b\}^*\}$ vérifie les conditions du lemme de l'étoile (à savoir il existe un entier N tel que ...) mais n'est cependant pas reconnaissable.

Chapitre 4

Langages rationnels

4.1 Langages rationnels

Définition 69 *Soit X un alphabet, on définit inductivement une suite de sous-ensembles de X^* de la façon suivante :*

$$\begin{aligned} &Rat_0(X^*) \text{ est l'ensemble des parties finies de } X^* \\ &Rat_{n+1}(X^*) = \{R \subseteq X^* \mid \exists p \in \mathbb{N}, \exists k_1,\ k_2,\ \ldots\ k_p \in \mathbb{N},\ R = R_{11}R_{12}\ldots R_{1k_1} \cup \ldots \cup R_{p1}R_{p2}\ldots R_{pk_p} \\ &\qquad\qquad \forall i,\ 1 \leq i \leq p,\ \forall j,\ 1 \leq j \leq k_i,\ R_{ij} \in Rat_n(X^*) \\ &\qquad\qquad \text{ou } R_{ij} = A^* \text{ avec } A \in Rat_n(X^*)\ \} \\ &\qquad \cup\ Rat_n(X^*) \end{aligned}$$

On désigne par $Rat(X^*)$ l'union infinie des $Rat_n(X^*)$ et on l'appelle l'ensemble des parties rationnelles (ou langages rationnels) de X^*.

Proposition 70 *$Rat(X^*)$ est la plus petite famille contenant les parties finies de X^* et fermée par union finie, produit fini et étoile.*

Preuve :

- Les parties finies sont dans $\mathrm{Rat}(X^*)$ de par sa définition.
- $$\left\{ \begin{array}{lcl} L_1, L_2 \in Rat(X^*) & \Rightarrow & \exists n_1, n_2, L_1 \in Rat_{n_1}(X^*), L_2 \in Rat_{n_2}(X^*) \\ & \Rightarrow & L_1, L_2 \in Rat_n(X^*) \text{ avec } n = max(n_1, n_2) \\ & \Rightarrow & L_1 \cup L_2,\ L_1.L_2,\ {L_1}^*,\ {L_2}^* \in Rat_{n+1}(X^*) \end{array} \right.$$

Il reste alors à montrer que cette famille est la plus petite possédant cette propriété.
Soit $\mathcal{L}$ une autre famille contenant les parties finies de X^* et fermée par union finie, produit fini et étoile. On a :

$$Rat_0(X^*) \subseteq \mathcal{L}$$

$$Rat_n(X^*) \subseteq \mathcal{L} \ \Rightarrow\ Rat_{n+1}(X^*) \subseteq \mathcal{L}$$

D'où, par récurrence sur n :

$$Rat(X^*) = \cup_i Rat_i(X^*) \subseteq \mathcal{L}$$

Rappelons maintenant les propriétés de clôture de Rec(X^*) :

Proposition 71 *Soient L, L_1, L_2 des sous-ensembles de X^*.*

1. *Si L est une partie finie de X^* alors L est reconnaissable.*
2. *Si L_1 et L_2 sont reconnaissables alors $L_1 \cup L_2$ et $L_1.L_2$ sont reconnaissables.*
3. *si L est reconnaissable alors L^* est reconnaissable.*

Comme $Rat(X^*)$ est la plus petite famille contenant les parties finies de X^* et fermée par union finie, produit fini et étoile, on a donc $Rat(X^*) \subseteq Rec(X^*)$.

Corollaire 72 (Reformulation de la proposition 71)
Tout langage rationnel est reconnaissable.

Proposition 73 *Tout langage reconnaissable est rationnel.*

Preuve : Nous allons construire effectivement, à partir de l'automate, la décomposition (dite de Kleene) d'un langage reconnaissable. Soit L ce langage, que l'on peut supposer non vide.
Notons $\mathcal{A}$ un automate le reconnaissant tel que Q=$\{ q_0, \ldots, q_p, q_{p+1}, \ldots, q_n\}$, $A = \{q_p, \ldots, q_n\}$ $(0 \leq p \leq n)$.
Alors L=$\bigcup_{j=p}^{n} L_j$ avec $L_j = \{f \in X^* \mid q_j \in \hat{\delta}(q_0, f)\}$. En fait L_j est l'ensemble des mots trace de chemin menant de q_0 à q_j.

Posons pour i, j$\in [n]$, et k$\in$ N, l'ensemble suivant : $L_{ij}^k = \{f \in X^* \mid$ il existe un chemin de q_i à q_j ne passant pas par un état d'indice$l \ > \ k\}$. On a alors les relations suivantes :

- $L_{ii}^0 = \{\mathbb{1}\} \cup \{x \in X \mid q_i \in \delta(q_i, x)\}$ (cet ensemble peut être réduit à $\{\mathbb{1}\}$)
- $\forall i, j, \ i \neq j, \ L_{ij}^0 = \{x \in X \mid q_j \in \delta(q_i, x)\}$ (cet ensemble peut être vide)
- $L_{ij}^{k+1} = L_{ij}^k \cup L_{i,k+1}^k (L_{k+1,k+1}^k)^* L_{k+1,j}^k$

Il est donc facile de montrer (par récurrence sur k) que les L_{ij}^k sont des parties rationnelles de X^*, en particulier pour k égal à n. On en déduit que $L_j = L_{0j}^n$ est rationnel pour tout j compris entre p et n donc L est rationnel.

Théorème 74 (Théorème de Kleene)
Les deux familles Rec(X^*) et Rat(X^*) sont identiques. Autrement dit un langage de X^* est rationnel si, et seulement si, il est reconnaissable.

4.2 Expressions rationnelles

Définition 75 *Soit X un alphabet, on définit ER(X), ensemble des expressions rationnelles sur X, comme le plus petit ensemble de mots sur X∪{(,),+,.,*,0} vérifiant :*

1. $\forall x \in X, x \in ER(X)$
2. $0 \in ER(X)$
3. $\forall e_1,\ e_2 \in ER(X), (e_1 + e_2),\ (e_1.e_2)$ *et* ${e_1}^*\ \in ER(X)$.

Définition 76 (Langage dénoté par une expression rationnelle)
A chaque expression rationnelle $e \in ER(X)$ on va associer le langage $\mid e \mid$ sur X^* de la façon suivante :

- $\forall a \in X,\ \mid a \mid = \{a\}$
- $\mid 0 \mid = \emptyset$
- $\mid (e_1 + e_2) \mid = \mid e_1 \mid \cup \mid e_2 \mid$
- $\mid (e_1.e_2) \mid = \mid e_1 \mid . \mid e_2 \mid$
- $\mid e^* \mid = \mid e \mid^*$

Proposition 77
A tout langage rationnel L de X^* on sait associer une expression rationnelle e de ER(X) telle que e dénote L.

Preuve :
Par récurrence sur l'entier n tel que L$\in Rat_n(X^*)$.

Définition 78
Deux expressions rationnelles e_1 et e_2 sont dites équivalentes si elles dénotent le même langage. On le notera $e_1 \equiv e_2$.

Proposition 79
Pour toutes expressions $e_1,\ e_2,\ e_3$ de ER(X), on a les équivalences suivantes :

- $(e_1 + e_2) \equiv (e_2 + e_1)$
- $((e_1 + e_2) + e_3) \equiv (e_1 + (e_2 + e_3))$
- $((e_1.e_2).e_3) \equiv (e_1.(e_2.e_3))$
- $(e_1.(e_2 + e_3)) \equiv ((e_1.e_2) + (e_1.e_3))$
- ${e_1}^{**} \equiv {e_1}^*$

Proposition 80 *Il existe un nombre fini de propriétés permettant de détablir l'équivalence de deux expressions rationnelles quelconques (à condition qu'elles soient équivalentes!).*

Axiomatisation de Salomaa :
Les propriétés annoncées dans la proposition ci-dessus sont les suivantes :

1. $(\alpha + (\beta + \gamma)) \equiv ((\alpha + \beta) + \gamma)$
2. $(\alpha(\beta\gamma)) \equiv ((\alpha\beta)\gamma)$
3. $((\alpha + \beta)\gamma) \equiv ((\alpha\gamma) + (\beta\gamma))$
4. $(\alpha(\beta + \gamma)) \equiv ((\alpha\beta) + (\alpha\gamma))$
5. $(\alpha + \beta) \equiv (\beta + \alpha)$
6. $(0^*\alpha) \equiv \alpha$
7. $(0\alpha) \equiv 0$
8. $\alpha^* \equiv ((\alpha^*\alpha) + 0^*)$
9. $\alpha^* \equiv (\alpha + 0^*)^*$

ainsi que les deux règles d'inférence suivantes :

- I) de $\alpha \equiv ((\alpha\beta) + \gamma)$ et $\mathbb{1} \notin | \beta |$ on déduit $\alpha \equiv \gamma\beta^*$
- II) de $\alpha \equiv \beta$ et $\gamma \equiv \delta$, si γ' est le résultat du remplacement de α par β dans γ, on déduit $\gamma' \equiv \delta$ et $\gamma' \equiv \gamma$.

4.3 Résiduels

Définition 81 *Soit $L \subseteq X^*$ un langage et $u \in X^*$ un mot. On appelle résiduel du langage L par rapport à u, et l'on note $u^{-1}.L$, le langage*

$$u^{-1}.L = \{v \in X^*,\ u.v \in L\}$$

Remarque :

$$(u.v)^{-1}.L = v^{-1}.(u^{-1}.L)$$

Proposition 82 *Un langage est rationnel si et seulement si il possède un nombre fini de résiduels.*

Preuve : Soit L un langage reconnaissable. Il est reconnu par un automate $\mathcal{A} = (X, Q, q_0, A, \delta)$ déterministe, complet et accessible. Pour tout état q, posons $L_q = \{v \in X^*, \hat{\delta}(q, v) \in A\}$, ce langage étant l'ensemble des mots reconnus par $\mathcal{A}$ à partir de l'état q. On montre que l'ensemble des résiduels de L est exactement l'ensemble $\{L_q,\ q \in A\}$. Le nombre de résiduels distincts de L est alors au maximum égal au cardinal de Q, deux états distincts q et q' pouvant donner un unique résiduel $L_q = L_{q'}$.

Réciproquement, soit L possédant un nombre fini de résiduels. On construit l'automate $\mathcal{A} = (X, Q, q_0, A, \delta)$ où :

- Q=$\{[u^{-1}.L],\ u \in x^*\}$, fini par hypothèse
- $q_0 = [\mathbb{1}_{-1}.L] = L$
- A=$\{[u^{-1}.L],\ \mathbb{1} \in u^{-1}.L\}$
- $\delta([u^{-1}.L], x) = [ux^{-1}.L]$, si bien que $\hat{\delta}(q_0, u) = \hat{\delta}(L, u) = [u^{-1}.L]$.

Cet automate est déterministe et complet, il reste à vérifier qu'il reconnait L. On a ainsi montré que L est reconnaissable.

4.4 Automate minimal

On se place dans le cadre des automates finis déterministes, complets et accessibles (en abrégé a.f.d.c).

Définition 83 *Soit $\mathcal{A} = (X, Q, q_0, A, \delta)$ un a.f.d.c. Un mot $u \in X^*$ sépare deux états q et q' dans $\mathcal{A}$ si, des deux états $\hat{\delta}(q, u)$ et $\hat{\delta}(q', u)$, l'un est final et l'autre pas.*

Définition 84 (Equivalence de Nérode)
Soit $\mathcal{A} = (X, Q, q_0, A, \delta)$ un a.f.d.c., on dit que deux états q et q' sont Nérode-équivalents, ou simplement équivalents, et on note $q \sim q'$, si aucun mot de X^* ne les sépare.

Remarque : Le lecteur vérifiera qu'il s'agit bien d'une relation d'équivalence.

Définition 85 (Automate quotient)
Notons, pour tout $q \in Q$, $\bar{q}$ sa classe d'équivalence.
On définit l'automate $\overline{\mathcal{A}} = (X, \overline{Q}, \overline{q_0}, \overline{A}, \overline{\delta})$ où :

- $\overline{Q} = \{\bar{q}, q \in Q\} = Q/\sim$
- $\overline{A} = \{\bar{q}, q \in A\} = A/\sim$
- $\overline{\delta} = \{(\bar{q}, x, \bar{q}'), \exists q_1 \in \bar{q},\ \exists q_2 \in \bar{q}',\ (q_1, x, q_2) \in \delta\}$

Proposition 86 *Les automates $\mathcal{A}$ et $\overline{\mathcal{A}}$ reconnaissent le même langage.*

La preuve en est laissée au lecteur.
Pour que cette construction soit effective, il faut pouvoir calculer les classes d'équivalence modulo $\sim$.

Proposition 87 *Soit $\mathcal{A}$ un a.f.d.c., on peut calculer de façon effective son équivalence de Nérode associée, et donc construire $\overline{\mathcal{A}}$.*

Preuve : On construit par induction une suite finie de relations d'équivalence de plus en plus fines, notées $\equiv_i$.
$\equiv_0$ contient deux classes : A et Q \ A.
$\equiv_i$ est définie par : $q \equiv_i q'$ ssi $q \equiv_{i-1} q'$ et $\forall x \in X, \delta(q,x) \equiv_{i-1} \delta(q',x)$.
Q étant fini, on arrive au bout d'un nombre fini de calculs à une équivalence plus fine que toutes les autres, qui n'est autre que $\sim$.

Proposition 88 *Soit L un langage reconnaissable, et $\overline{\mathcal{A}}$ l'automate quotient fabriqué à partir d'un quelconque a.f.d.c.reconnaissant L. $\overline{\mathcal{A}}$ est isomorphe à l'automate possédant le minimum d'états construit à partir des résiduels de L. On l'appelle automate minimal de L.*

Preuve : Pour tous états q et q', $L_q = L'_q$ ssi $q \sim q'$. On a donc bien une bijection entre les résiduels et les classes d'équivalence.

Conséquence : On sait construire effectivement l'automate minimal de tout langage reconnaissable. Ceci permet de décider d'un grand nombre de questions concernant les automates finis et les langages rationnels, comme l'égalite de deux langages, l'équivalence de deux automates ou celle de deux expressions rationnelles (ceci sans passer par l'axiomatisation de Salomaa).

Exercices :

Exercice 1 :
Calculer les résiduels du langage suivant : $\{a^{2^n} \mid n \geq 0\}$
En déduire que ce langage n'est pas rationnel. Même question avec le langage $\{aba^2ba^3b...a^nb \mid 0 \leq n\}$

Exercice 2 :
A partir d'un automate fini reconnaissant un quelconque langage (on pourra reprendre des exemples dans la liste d'exercices du chapitre précédent), donner une expression rationnelle du langage reconnu.

Exercice 3 :
Donner une expression rationnelle de l'ensemble des mots de $\{a,b\}^*$ ne contenant pas le facteur abaab.

Exercice 4 :
Soit L le langage dénoté par l'expression rationnelle $(aa^*ba + ab)^*a$. Calculer les résiduels de L et en déduire un automate fini le reconnaissant.
Donner l'automate minimal de L.

Exercice 5 :
Vérifier l'équivalence des deux expressions rationnelles $((a+ab)^*b)^*$ et $(a^*b)^*$. (au moins deux méthodes possibles)

Exercice 6 :

Donner l'automate minimal du langage $L = \{f \in \{a, b\}^*, \mid f \mid$ est divisible par 3, f commence par a et finit par b, f ne contient pas le facteur $aaa\}$

Chapitre 5

Automates à pile

Nous allons dans ce chapitre définir des classes d'automates permettant de reconnaître les langages algébriques, tout comme les automates finis reconnaissent les langages rationnels. A peu de choses près ces automates sont équivalents entre eux, ce qui sera prouvé dans la suite. L'intérêt de la diversité réside dans le fait que, selon les circonstances, il sera plus naturel ou plus aisé d'utiliser un automate de tel type plutôt que de tel autre.

5.1 Automates à pile simples

Définition 89 (Automate à pile simple)
Un automate à pile simple est un quadruplet A=(X, Y,y_0, λ) où:

- *X est un alphabet dit alphabet d'entrée*
- *Y est un alphabet dit alphabet de pile (pas nécessairement disjoint de X)*
- *y_0 est une lettre de Y dit symbole initial de pile*
- *λ est une application de $X \times Y$ dans $\mathcal{P}_f(Y^*)$(ensemble des parties finies de Y^*)*

λ peut être étendu (de façon unique) en une application $\hat{\lambda}$ de $X^* \times \mathcal{P}(Y^*)$ dans $\mathcal{P}(Y^*)$ de la façon suivante :

$$\forall \Omega \in \mathcal{P}(Y^*)\ \hat{\lambda}(\mathbb{1}, \Omega) = \Omega$$

$$\forall x \in X,\ \forall m \in X^*,\ \forall \Omega \in \mathcal{P}(Y^*),\ \hat{\lambda}(xm, \Omega) = \hat{\lambda}(m, \bigcup\{\lambda(x, y)w \ \mid\ y \in Y, yw \in \Omega\})$$

Remarquons que :

$$\forall x \in X, \hat{\lambda}(x, \mathbb{1}) = \emptyset$$

$$\forall x \in X, \hat{\lambda}(x, \emptyset) = \emptyset$$
$$\forall x \in X, \forall y \in Y, \hat{\lambda}(x, y) = \lambda(x, y)$$
$$\forall m_1, m_2 \in X^*,\ \hat{\lambda}(m_1 m_2, \Omega) = \hat{\lambda}(m_2, \hat{\lambda}(m_1, \Omega))$$

Définition 90 (langage reconnu) *Le langage reconnu par un automate à pile simple A est :*

$$L(A) = \{m \in X^* \ \mid \ \mathbb{1} \in \hat{\lambda}(m, \{y_0\})\}$$

Proposition 91 *Un langage L est algébrique propre si, et seulement si, il est reconnu par un automate à pile simple.*

Preuve : Soit L un langage algébrique propre de X^* et G=(X,V,S,P) une grammaire de L en forme normale de Greibach. On associe à G l'automate A=(X,V,S,λ) où λ est définie par :

$$\forall x \in X,\ \forall T \in V,\ \lambda(x, T) = \{w \in V^* \ \mid \ (T, xw) \in P\}$$

Alors L(A) = L(G,S) = L :

Soit f de L(G,S) et $\mid f \mid$=n. Alors il existe une dérivation gauche de S en f de longueur n que l'on peut décrire par $(S = u_0, u_1, \ldots, u_n = f)$ avec:

- $\forall i,\ 0 \leq i < n,\ u_i \to u_{i+1}$
- $\forall i,\ 0 \leq i < n,\ u_i = u_i' u_i$" $avec\ u_i' \in X^*, u_i$" $\in V^*\ et\ \mid u_i' \mid = i, \mid u_i$" $\mid \leq n - i$

Les u_i' sont les facteurs gauches de f de longueur i. On montre par induction finie que :

$$\forall i,\ u_i" \in \hat{\lambda}(u_i', \{S\})$$

Or u_n"=$\mathbb{1}$ et u_n'=f d'où :

$$\mathbb{1} \in \hat{\lambda}(f, \{S\})\ et\ L(G, S) \subseteq L(A)$$

Réciproquement, soit f un mot de L(A), i.e. $\mathbb{1} \in \hat{\lambda}(f, \{S\})$, on pose :

- f=$x_1 \ldots x_n$
- $\forall i,\ 0 \leq i \leq n,\ v_i = x_1 \ldots x_i\ (v_0 = \mathbb{1})$
- $\forall i,\ 0 \leq i \leq n,\ v_i' = x_{i+1} \ldots x_n\ (v_n' = \mathbb{1})$

On montre par induction finie que :

$$\forall i,\ 0 \leq i \leq n,\ \exists w_i \in \hat{\lambda}(v_i, \{S\}),\ w_i \xrightarrow[G]{*} v_i'$$

or :

$$\hat{\lambda}(v_0, \{S\}) = \hat{\lambda}(\mathbb{1}, \{S\}) = \{S\}$$

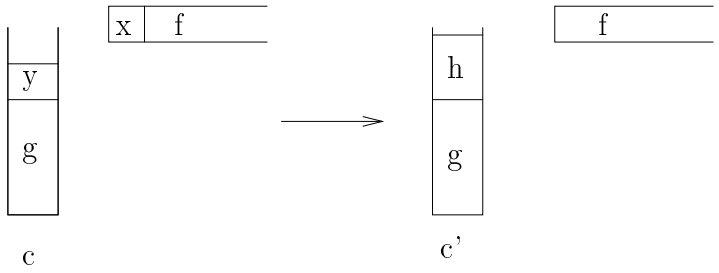
d'où : $S \xrightarrow[G]{*} v'_0 = f$ *et* $L(A) \subseteq L(G, S)$.

La condition nécessaire est ainsi prouvée. Pour la condition suffisante, partant d'un langage L reconnu par un automate à pile simple A=(X,Y,y_0,λ), on construit une grammaire G=(X,V,y_0,P) avec $T \xrightarrow[G]{} xw \in P$ ssi $(x, T, w) \in \lambda$. C'est la construction symétrique de la précédente et elle marche pour les mêmes raisons.

Autre approche : Il peut sembler plus aisé de parler des automates à pile en des termes similaires à ceux employés pour les automates finis. Nous allons donc donner une autre version (équivalente) du mode d'emploi des automates à pile.

Définition 92 *Une configuration d'un automate à pile A=(X, Y,y_0, λ) est un couple (f,g), où $f \in X^*$ et $g \in Y^*$.*
Il y a transition entre une configuration c et une autre c' si et seulement si c=(xf,yg) et c'=(f,hg) avec $h \in \lambda(x, y)$.
Un calcul valide est une suite de configurations $c_1, \ldots, c_n$ telle qu'il existe un calcul valide entre c_i et c_{i+1} pour tout i.
Un mot f est reconnu par A ssi il existe un calcul valide de (f,y_0) à ($\mathbb{1}$,$\mathbb{1}$).

La plupart des définitions peuvent être revues en ces termes.



Transition entre c et c'.

5.2 Automates à pile généraux

Définition 93 (Automate à pile avec états) *Un automate à pile est un sextuplet A=(Q,X, Y,q_0, y_0, λ) où:*

- *Q est un ensemble d'états*
- *X est l'alphabet d'entrée*
- *Y est l'alphabet de pile*

- q_0 *est un élément distingué de* Q *appelé état initial*
- y_0 *est le symbole initial de pile*
- λ *est une application de* $Q \times (X \cup \{\mathbb{1}\}) \times Y$ *dans* $\mathcal{P}_f(Q \times Y^*)$

Notations : On note souvent les régles de l'automate sous la forme

$$q, x, y \longrightarrow q', w$$

On appelle ϵ-règle une règle de la forme

$$q, \mathbb{1}, y \longrightarrow q', w$$

On vient de donner une nouvelle définition de la notion d'automate dont on montrera l'équivalence à la précédente.

λ est, de nouveau, étendue en une application $\hat{\lambda}$ de $X^* \times \mathcal{P}(Q \times Y^*)$ dans $\mathcal{P}(Q \times Y^*)$:

$$\begin{array}{l} \forall x \in X, \forall m \in X^*, \forall \theta \in \mathcal{P}(Q \times Y^*), \\ \hat{\lambda}(xm, \theta) = \hat{\lambda}(m, \bigcup\{\lambda(q, x, y)w \mid y \in Y, (q, yw) \in \theta\}) \\ \qquad\qquad \cup\ \hat{\lambda}(xm, \bigcup\{\lambda(q, \mathbb{1}, y)w \mid y \in Y, (q, yw) \in \theta\}) \end{array}$$

$$\begin{array}{l} \forall \theta \in \mathcal{P}(Q \times Y^*), \\ \hat{\lambda}(\mathbb{1}, \theta) = \theta \ \cup\ \hat{\lambda}(\mathbb{1}, \bigcup\{\lambda(q, \mathbb{1}, y)w \mid y \in Y, (q, yw) \in \theta\}) \end{array}$$

Remarquons que :

- $\forall m_1, m_2 \in X^*, \forall \theta \in \mathcal{P}(Q \times Y^*),\ \hat{\lambda}(m_1 m_2, \theta) = \hat{\lambda}(m_2, \hat{\lambda}(m_1, \theta))$
- On n'est plus dans les parties finies de $Q \times Y^*$.

5.2.1 Langage reconnu par pile vide

Définition 94 *Le langage reconnu par pile vide par un tel automate à pile est :*

$$N(A) = \{m \in X^* \mid \exists q \in Q,\ (q, \mathbb{1}) \in \hat{\lambda}(m, \{(q_0, y_0)\})\}$$

Proposition 95 *Soit A=(Q,X,Y,*q_0, y_0, λ*) un automate à pile, N(A) est algébrique.*

Preuve : Il faut construire une grammaire associée à A. Soit G=(X,V,S,P) la grammaire définie de la façon suivante:

1. V $= Q \times Y \times (Q \cup \{F\})$ où F est un nouveau symbole n'appartenant pas à $X \cup Y \cup Q$.

2. Les règles de P sont:
   $\forall q, q' \in Q$, $\forall y \in Y$
   - [q,y,q'] $\rightarrow$ $x[q_n, y_n, q_{n-1}]$ ... $[q_1, y_1, q']$ pour tous les $q_{n-1}, \ldots, q_1$ si λ(q,x,y) contient $(q_n, y_n \ldots y_1)$
   - [q,y,q'] $\rightarrow$ $[q_n, y_n, q_{n-1}]$ ... $[q_1, y_1, q']$ pour tous les $q_{n-1}, \ldots, q_1$ si λ(q,𝟙,y) contient $(q_n, y_n \ldots y_1)$
   - [q,y,q'] $\rightarrow$ x si λ(q,x,y) contient (q',𝟙)
   - [q,y,q'] $\rightarrow$ 𝟙 si λ(q,𝟙,y) contient (q',𝟙)
   - [q,y,F] $\rightarrow$ $x[q_n, y_n, q_{n-1}]$ ... $[q_1, y_1, F]$ pour tous les $q_{n-1}, \ldots, q_1$ si λ(q,x,y) contient $(q_n, y_n \ldots y_1)$
   - [q,y,F] $\rightarrow$ $[q_n, y_n, q_{n-1}]$ ... $[q_1, y_1, F]$ pour tous les $q_{n-1}, \ldots, q_1$ si λ(q,𝟙,y) contient $(q_n, y_n \ldots y_1)$
   - [q,y,F] $\rightarrow$ x si $\exists$ q' tel que λ(q,x,y) contient (q',𝟙)
   - [q,y,F] $\rightarrow$ 𝟙 si $\exists$ q' tel que λ(q,𝟙,y) contient (q',𝟙)
3. L'axiome est $[q_0, y_0, F]$.

On peut vérifier que cette grammaire engendre N(A).

Proposition 96 *Un langage L est algébrique si et seulement si il existe un automate à pile A tel que L=N(A).*

Preuve : On vient de voir que la condition est suffisante, montrons qu'elle est nécessaire.

Soit L un langage algébrique. Si L est propre, la question est déja réglée, puisque L est reconnu par un automate à pile simple, cas particulier d'automate à pile général sans état, ce qui équivaut à un unique état, et sans ϵ-règle.

Si L contient le mot vide, alors $M = L \setminus \{\mathbb{1}\}$ est algébrique propre.
Soit A=$(\{q_0\}, X, Y, q_0, y_0, \lambda)$ un automate à pile sans ϵ-règle et à un seul état tel que M=N(A). Posons A'=$(\{q_0, q_1, q_2\}, X, Y, q_0, y_0, \lambda')$ avec

$$(q_0, \epsilon, y_0) \longrightarrow (q_2, \mathbb{1}) \in \lambda'$$

et

$$(q_0, x, y_0) \longrightarrow (q_1, w) \in \lambda' \; ssi \; (q_0, x, y_0) \longrightarrow (q_0, w) \in \lambda$$

et

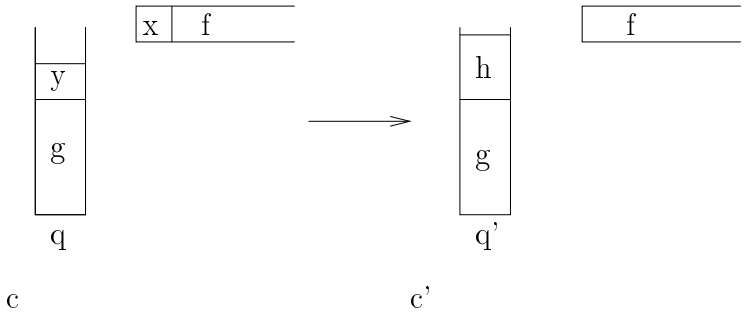
$$(q_1, x, y) \longrightarrow (q_1, w) \in \lambda' \; ssi \; (q_0, x, y) \longrightarrow (q_0, w) \in \lambda$$

Les seules règles de λ' sont celles citées ci-dessus.
L'automate A' reconnait L par pile vide.

Autre approche : Redonnons les notions de configuration et de calcul vues dans le cas des automates sans état.

Définition 97 *Une configuration d'un automate à pile A=(Q,X,Y,q_0, y_0, λ) est un triplet (f,q,g), où $f \in X^*$, $q \in Q$ et $g \in Y^*$.*

Il y a transition entre une configuration c et une autre c' si et seulement si c=(xf,q,yg) et c'=(f,q',hg) avec $(q', h) \in \lambda(q, x, y)$.
Un calcul valide est une suite de configurations $c_1, \ldots, c_n$ *telle qu'il existe un calcul valide entre* c_i *et* c_{i+1} *pour tout i.*
Un mot f est reconnu par A par pile vide ssi il existe un calcul valide de (f,q_0, y_0*) à (*$\mathbb{1}, q, \mathbb{1}$*), pour un quelconque état q.*



Transition dans un automate à pile.

5.2.2 Langage reconnu par états terminaux

Soit A un automate à pile. On se donne un ensemble Q_T d'états dits terminaux.

Définition 98 *Le langage reconnu par états terminaux par A est :*

$$T(A) = \{f \in X^* \mid \exists q \in Q_T,\ \exists w \in Y^*,\ (q, w) \in \hat{\lambda}(f, (q_0, y_0))\}$$

Autre approche :

Définition 99 *Un mot f est reconnu par A par états terminaux ssi il existe un calcul valide de (f,*q_0, y_0*) à (*$\mathbb{1}, q, g$*), pour un quelconque mot* $g \in Y^*$*, et pour un état terminal q.*

Proposition 100 *Les deux modes de reconnaissance pile vide ou états terminaux sont équivalents, c'est-à-dire :*

- *Si L est le langage reconnu par un automate A par états terminaux, alors il existe un automate B reconnaissant L par pile vide, et*
- *si L est reconnu par pile vide par un automate A, il existe un automate B le reconnaissant par états terminaux.*

Corollaire 101 *Un langage L est algébrique si et seulement si il existe un automate A tel que L=T(A).*

5.2.3 Langages déterministes

Nous introduisons une sous-classe des automates à pile.

Définition 102 (automate déterministe) *Un automate à pile déterministe est un septuplet A=(Q,X,Y,q_0,Q_T,y_0,λ) où:*

- *Q est un ensemble d'états*
- *X est l'alphabet d'entrée*
- *Y est l'alphabet de pile*
- *q_0 est un élément distingué de Q appelé état initial*
- *Q_T est un sous-ensemble de Q d'états dits terminaux*
- *y_0 est le symbole initial de pile*
- *λ est une application partielle de $Q \times (X \cup \{\mathbb{1}\}) \times Y$ dans $(Q \times Y^*)$ telle que, quels que soient q et y, si λ(q,$\mathbb{1}$,y) est défini alors, quel que soit x de X, λ(q,x,y) n'est pas défini.*

On constate que dans ce type d'automate le mouvement est entièrement déterminé par l'état, la lettre lue en entrée et le sommet de pile.

Comme d'habitude λ est étendue en une application $\hat{\lambda}$ de $X^* \times (Q \times Y^*)$ dans $Q \times Y^*$ de la façon suivante :

$\forall x \in X, \forall f \in X^*, \forall q \in Q, \forall y \in Y, \forall w \in Y^*$

$$\begin{array}{lcll}
\hat{\lambda}(xf,(q,yw) & = & \hat{\lambda}(f,(q',zw)) & si \quad \lambda(x,q,y) = (q',z) \\
 & = & \hat{\lambda}(xf,(q',zw)) & si \quad \lambda(\mathbb{1},q,y) = (q',z) \\
\hat{\lambda}(\mathbb{1},(q,yw) & = & \hat{\lambda}(\mathbb{1},(q',zw)) & si \quad \lambda(\mathbb{1},q,y) = (q',z) \\
 & = & (q,yw) & si \quad \lambda(\mathbb{1},q,y)\ n'est\ pas\ défini \\
\hat{\lambda}(\mathbb{1},(q,\mathbb{1})) & = & (q,\mathbb{1}) &
\end{array}$$

Proposition 103 *Le langage reconnu par pile vide par un automate à pile déterministe est un langage préfixe, c'est-à-dire :*

$$\forall f,g \in X^* \ (f \in N(A) \ et \ fg \in N(A)) \ \Rightarrow \ g = \mathbb{1}$$

Preuve : Raisonnons par l'absurde : supposons f et g, g$\neq \mathbb{1}$, tels que f et fg appartiennent à N(A) alors il existe q_1 et q_2, deux états tels que :

$$(q_1,\mathbb{1}) \in \hat{\lambda}(f,(q_0,y_0)) \ et \ (q_2,\mathbb{1})\hat{\lambda}(fg,(q_0,y_0))$$

Or :

$$\hat{\lambda}(fg,(q_0,y_0)) = \hat{\lambda}(g,\hat{\lambda}(f,(q_0,y_0)) = \hat{\lambda}(g,(q_1,\mathbb{1}))$$

Or si g est différent de $\mathbb{1}$, $\hat{\lambda}(g,(q_1,\mathbb{1}))$ n'est pas défini. D'où contradiction.

Définition 104 (langage déterministe) *On appelle langage déterministe, un langage reconnu par états terminaux par un automate à pile déterministe.*

Proposition 105 *La famille $Det(X^*)$ des langages déterministes sur X^* est une sous-famille stricte de la famille $Alg(X^*)$, contrairement à ce qui se passe dans le cas des langages reconnaisables.*

Proposition 106 *Si L un langage algébrique et K un langage rationnel, alors $L \cap K$ est algébrique. On dit que la famille $Alg(X^*)$ est close par intersection rationnelle.*

Preuve : Soit L reconnu par un automate à pile $A = (Q, X, Y, q_0, Q_T y_0, \lambda)$ par états terminaux.
Soit K reconnu par $\mathcal{A}' = (X', Q', q_0', A', \delta)$.
On pose $B = (X \cap X', Q \times Q', Y, (q_0, q_0'), y_0, \lambda_B)$ avec comme fonction de transition λ_B :

$$\lambda_B = \{(x,(q,r),y,(q',r'),h),\ (x,q,y,q',h) \in \lambda \text{ et } (x,r,r') \in \delta\}$$
$$\cup\ \{(\mathbb{1},(q,r),y,(q',r),h),\ r\ \in Q',\ (\mathbb{1},q,y,q',h) \in \lambda\}$$

$L \cap K$ est reconnu par B par états terminaux en prenant $Q_T \times A'$ comme ensemble d'états terminaux. Il est donc algébrique.

Corollaire 107 *$Det(X^*)$ est close par intersection rationnelle.*

Preuve : La construction ci-dessus conserve le caractère déterministe des automates.

Exercices :

Exercice 1 :
Donner un automate à pile simple reconnaissant le langage $\{a^n b^n, n \geq 1\}$

Exercice 2 :
Donner un automate à pile simple reconnaissant le langage $\{uc\tilde{u} \mid u \in \{a,b\}^*\}$

Exercice 3 :
Donner un automate à pile simple reconnaissant le langage $\{u\tilde{u} \mid u \in \{a,b\}^*,\ u \neq \mathbb{1}\}$

Exercice 4 :
Soit $\mathcal{A}$ un automate fini dont l'état initial n'est pas terminal. Montrer que l'on peut

construire un automate à pile simple simulant $\mathcal{A}$ (i.e. les langages reconnus sont les mêmes).

Exercice 5 :

Montrer, en utilisant des automates à pile, que les langages suivants sont algébriques :

1. $L_1 = \{u\tilde{u} \mid u \in \{a,b\}^*\}$
2. $L_2 = \{u \in \{a,b\}^* \mid u = \tilde{u}\}$
3. $L_3 = \{uc\tilde{v} \mid u,v \in \{a,b\}^* \ et \ u \neq v\}$ Montrer que ce langage est déterministe par des propriétés de clôture. Donner un automate à pile déterministe le reconnaissant par états terminaux.
4. $L_4 = \{a^k b^{n_1} c^{n_1} b^{n_2} c^{n_2} \dots b^{n_k} c^{n_k} \mid k > 0 \ et \ \forall i : \ n_i > 0\}$
5. $L_5 = \{a^{n_1} b a^{n_2} b \dots a^{n_k} b \mid k > 0 \ et \ \exists i : \ i \neq n_i\}$
6. $L_6 = \{a,b,c\}^* \setminus \{a^n b^n c^n \mid n > 0\}$
7. $L_7 = \{u \in \{a,b\}^* \mid |u|_a = 2|u|_b\}$
8. $L_8 = \{a^n b^m, \ 1 \leq n \leq m \leq 2n\}$
9. $L_9 = \{a^n b^m, n \neq m\}$ Montrer que le langage est déterministe.

Exercice 6 :

Soient X={a,b} et L=$\{a^n b^n \mid n \geq 1\} \ \cup \ \{b^n a^n \mid n \geq 1\}$.

- Construire un automate à pile à deux états qui reconnaît L par pile vide
- Appelons automate à compteur un automate à pile tel que $Y = \{y_0\}$, et langage à compteur un langage reconnu par pile vide par un tel automate. Montrer que L est un langage à compteur.
- Démontrer que L ne peut pas être reconnu par états terminaux par un automate à compteur.

Exercice 7 :

Soit X={a,b}. Montrer que le langage suivant est algébrique :
$L = \{uv \mid u,v \in X^*, |u| = |v| \ et \ u \neq v\}$

Exercice 8 :

Est-ce qu'à l'aide d'un automate fini on peut tester dans un programme de type Pascal la bonne imbrication des "begin" "end" ?
Montrer qu'un automate à pile peut résoudre le problème.
Donner la grammaire des "begin" "end" correctement imbriqués.

Chapitre 6

Lemme d'itération pour les langages algébriques

Définition 108 *Appelons profondeur d'un arbre la longueur maximale des éléments du domaine de cet arbre.*

Proposition 109 *Soit τ un arbre de dérivation de profondeur r, alors $\check{\tau}(\mathbb{1})$ est de longueur au plus p^r (où p est la longueur maximale des membres droits de P).*

La preuve est par récurrence sur la profondeur r.

Corollaire 110 *Si pour une grammaire G, f est un "long" mot de L(G,S), tout arbre de dérivation de S en f est "haut".*

Théorème 111 (Bar-Hillel,Perles et Shamir)

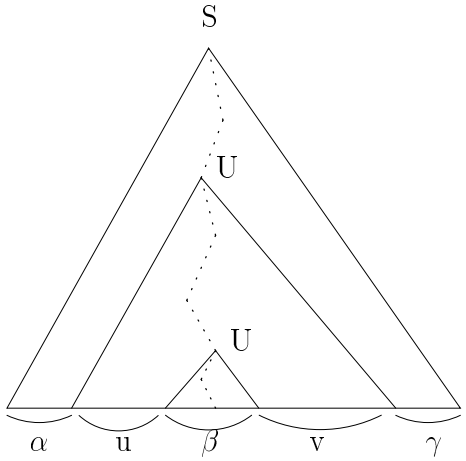
Pour tout langage algébrique L, il existe un entier N tel que tout mot f de L de longueur strictement supérieure à N se factorise en $f = \alpha u \beta v \gamma$ avec:

1. $\mid uv \mid \neq 0$
2. $\mid u\beta v \mid \leq N$
3. $\forall n \in \mathrm{N}\ \alpha u^n \beta v^n \gamma \in L$

Preuve : Soit $G = (X, V, S, P)$ une grammaire de L. On peut faire en sorte que G ne contienne aucune règle de la forme $T \underset{G}{\longrightarrow} T'$.
Posons d'abord $p = max\{\mid m \mid \mid \ (W \underset{G}{\longrightarrow} m) \in P\}$ et $r = Card\{V\}$.
Posons aussi $N = p^{r+1}$. On a $N > 0$ dès que $L \neq \{\mathbb{1}\}$ car $p \neq 0$.
Si f est un mot de $L(G, S)$ avec $\mid f \mid > N$, un arbre de dérivation τ de f aura une profondeur au moins égale à $r + 2$. Choisissons dans le domaine de cet arbre un mot w

de longueur maximale; w est de longueur supérieure ou égale à $r+2$ et $\tau(w) \in X \cup \{\mathbb{1}\}$. Le nombre des facteurs gauches de w est supérieur à $r+2$; prenons alors les $r+2$ plus grands facteurs gauches de w notés $w_1, \ldots, w_{r+2}(=\mathrm{w})$.
On a $\tau(w_{r+2}) = \tau(w) \in X \cup \{\mathbb{1}\}$ et $\forall i,\ 1 \leq i \leq r+1,\ \tau(w_i) \in V$.

Comme V contient r éléments, on en déduit qu'il existe un non-terminal U et qu'il existe i et j tels que $1 \leq i < j \leq r+1$ et $\tau(w_i) = \tau(w_j) = U$.



On pose :

1. $\beta = \check{\tau}(w_j)$
2. si w_j peut s'écrire $w_i\theta_1 \ldots \theta_{j-i}$ où $\theta_1 \ldots \theta_{j-1} \in [p]$:

$$u = \prod_{k=0}^{j-i-1} \Big(\prod_{\theta=1}^{(\theta_{k+1})-1} \check{\tau}(w_i\theta_1 \ldots \theta_k\theta) \Big)$$

$$v = \prod_{k=0}^{j-i-1} \Big(\prod_{\theta=(\theta_{k+1})+1}^{a(w_i\theta_1 \ldots \theta_k)} \check{\tau}(w_i\theta_1 \ldots \theta_k\theta) \Big)$$

3. si w_i peut s'écrire $\lambda_1 \ldots \lambda_q,\ \ \lambda_1 \ldots \lambda_q \in [p]$

$$\alpha = \prod_{k=0}^{q-1} \Big(\prod_{\lambda=1}^{(\lambda_{k+1})-1} \{\check{\tau}(\lambda_1 \ldots \lambda_k\lambda) \Big)$$

$$\gamma = \prod_{k=0}^{q-1} \Big(\prod_{\lambda=(\lambda_{k+1})-1}^{a(\lambda_1 \ldots \lambda_k)} \{\check{\tau}(\lambda_1 \ldots \lambda_k\lambda) \Big)$$

Il est facile de voir que $\check{\tau}(\mathbb{1}) = f = \alpha u \beta v \gamma$, $S \xrightarrow[G]{*} \alpha U \gamma$, $U \xrightarrow[G]{*} uUv$ et $U \xrightarrow[G]{*} \beta$.

La condition $uv \neq \mathbb{1}$ est assurée par le choix de la grammaire.

Par ailleurs le sous-arbre de racine w_i est au plus de profondeur $r+1$, on a donc :

$$| \check{\tau}(w_i) | = | u\beta v | \leq \ p^{r+1} \ = \ N$$

En effaçant ou itérant un nombre arbitraire de fois la partie de l'arbre comprise entre les deux occurences marquées de U, on vérifie la dernière condition du théorème, à savoir $\forall n \in \mathrm{N} \ \alpha u^n \beta v^n \gamma \in L$.

Remarque importante : L'intérêt de ce théorème est de permettre de montrer que de nombreux langages ne sont pas algébriques.

Exercices :

Exercice 1 :

Montrer que les langages suivants ne sont pas algébriques :

- $L_1 = \{a^n b^n c^n, \ 0 \leq n\}$
- $L_2 = \{a^n b^n c^i, \ 1 \leq i \leq n\}$
- $L_3 = \{ww, \ w \in \{a,b\}^*\}$
- $L_4 = \{aba^2ba^3b...a^nb, \ 0 \leq n\}$

Exercice 2 :

Montrer que tout langage fini vérifie les conditions du théorème de B-H,P&S.

Exercice 3 :

Proposer un langage L vérifiant les conditions du théoréme de B-H,P&S qui ne soit cependant pas algébrique. Indication : essayer d'ajouter une partie "molle" (un langage rationnel contenant "beaucoup" de mots, par exemple) au langage L_4 de l'exercice 1. Ne pas oublier de montrer que L n'est pas algébrique !

Bibliographie

[1] J.M. Autebert, *Théorie des langages et des automates,* MASSON 1994.

[2] J.E. Hopcroft, J.D. Ullman, *Introduction to automata theory, language and computation,* ADDISON WESLEY 1979.

[3] M. Harrison, *Introduction to the formal languages theory,* ADDISON WESLEY 1978.

[4] A. Aho, J. Ullman, *Concepts fondamentaux de l'informatique,* DUNOD 1993.

[5] A. Salomaa, *Formal languages,* ACADEMIC PRESS 1973.