Cours de Langages Formels et Systèmes symboliques

Rédaction : Laure Danthony, MIM00,
d'après le cours d'Arnaud Carayol, MIM99

Cours : Marianne Delorme, septembre - décembre 2001

Table des matières

Introduction

On peut faire débuter la théorie des langages formels en 1955, lorsque **Noam Chomski**, linguiste, pour qui le langage est une manifestation humaine, parle pour l'une des premières fois de l'existence d'une grammaire universelle qui permet d'engendrer du langage (du sens ?) : introduction de grammaire génératrice. Tout cela repose sur la possibilité qu'il y a d'engendrer une infinité de choses (mots) à partir d'un ensemble fini (alphabet).

A la même époque, les informaticiens recherchaient une façon de créer des langages de programmation plus élaborés que ceux dont ils disposaient. Ils cherchaient en fait une façon de *spécifier* des langages infinis de façon finie, d'où leur intérêt pour les travaux de Chomsky. Dans le même temps, ils se sont intéressés aux machines (automates et machines de Turing) et ont introduit la notion de reconnaissance en plus de celle de la génération de langage.

Théorie de Chomski
Cette théorie met en évidence des classes de langages formels :

Classes de langages	*Automates*	*Grammaires*
L. Rationnels ou réguliers	A. finis	Régulières
L. Algébriques ou contextués	A. à pile	Algébriques
L.Contextuels	M.T. à ruban linéaire borné	Contextuelles
L. Récursifs	M.T.	pas intéressantes

Remarques sur ces classes :

- Sur les langages rationnels : $\{a^n b^m, n, m \in \mathbb{N}\}$ est rationnel mais pas $\{a^n b^n, n \in \mathbb{N}\}$. On considère les automates finis déterministes et non déterministes.
- Sur les langages algébriques : $\{a^n b^n, n \in \mathbb{N}\}$ est algébrique mais pas $\{a^n b^n c^n, n \in \mathbb{N}\}$.
  Ces deux langages sont importants notamment en compilation. Entre les deux (strictement) on trouve les langages algébriques déterministes.
- Sur les langages contextuels : la tête de la machine de Turing ne peut pas bouger au delà de l'espace induit par la mot d'entrée. La machine de Turing est non déterministe.
- Les machines de Turing considérées pour les langages récursifs peuvent être ou non déterministes.

Dans ce cours, on étudiera les langages rationnels et les langages algébriques, puis on fera une excursion vers les mots infinis (automates finis "reconnaissant" des mots infinis), vers les mots bi-infinis, et on parlera des problèmes de déterminisme.

Chapitre 1

Mots et langages formels

Avant d'étudier systématiquement différentes classes de langages sous l'angle des automates, commençons par définir les éléments de la théorie des langages formels.

1.1 Définitions

On définit ici les notions élémentaires sur les mots puis on énonce les résultats fondamentaux les concernant.

1.1.1 Alphabets et mots

On appellera **alphabet** tout ensemble fini. Les éléments d'un alphabet sont traditionnellement appelés **symboles**, **lettres** ou encore **caractères**.

Considérons maintenant un alphabet Σ. On appelle **mot** sur Σ toute suite finie d'éléments de Σ. Pour un mot $a = (a_1, a_2, \ldots, a_n)$ sur Σ, l'entier n est appelé **longueur** du mot a, on le note $|a|$. L'ensemble des mots sur Σ est noté Σ^*.

Pour tout alphabet Σ, il existe un et un seul mot de longueur nulle, correspondant d'un point de vue ensembliste à l'unique application de l'ensemble vide dans Σ. On l'on appelle **mot vide** et on le notera toujours ε.

Définition 1.1.1 (concaténation)
Soit Σ un alphabet et soient $a = (a_1, \ldots, a_m)$ et $b = (b_1, \ldots, b_p)$ deux mots sur Σ. On appelle **concaténation** de a et b le mot

$$ab = (a_1, \ldots, a_m, b_1, \ldots, b_p)$$

On s'autorisera, pour chaque élément x de Σ, à désigner par x indifféremment la lettre x et le mot (x) de longueur 1. Par conséquent, on s'autorisera pour un mot $a = (a_1, a_2, \ldots, a_n)$ donné à noter $a = a_1 a_2 \ldots a_n$.

Proposition 1.1.1 *Pour tout alphabet Σ, l'ensemble Σ^* muni de la loi de concaténation est le monoïde libre engendré par Σ et ε en est l'élément neutre.*

On admettra cette propriété sachant que la preuve en est élémentaire étant données les définitions.

REMARQUE : "Monoïde libre" signifie que tout mot non vide sur Σ s'écrit de manière unique comme produit de ses lettres.

1.1.2 Structure des mots

La notion fondamentale de concaténation mène à un certain nombre de définitions supplémentaires. On supposera implicitement désormais que l'on dispose d'un alphabet Σ donné.

DÉFINITION 1.1.2 (FACTEUR, PRÉFIXE, SUFFIXE)
Un mot u est dit **facteur** d'un mot v s'il existe deux mots s et t tels que l'on ait $v = sut$. On parle de facteur gauche, ou **préfixe**, lorsque s est vide et de facteur droit, ou **suffixe**, lorsque t est vide. Dire que u est facteur de v est équivalent à dire que u a (au moins) une occurence dans v.

On emploie parfois l'expression *sous-mot* de u pour désigner soit un facteur de u soit une suite extraite de u (c'est-à-dire une partie de u *a priori* non connexe). Étant donnée cette ambiguïté, on n'emploira pas cette expression ici.

DÉFINITION 1.1.3 (PUISSANCES D'UN MOT)
Pour un mot u donné, on définit les **puissances** de u par récurrence en posant
- $u^0 = \varepsilon$
- pour tout entier n, $u^{n+1} = uu^n = u^n u$

DÉFINITION 1.1.4 (DÉPENDANCE MULTIPLICATIVE)
Deux mots u et v sont dits multiplicativement dépendants s'ils sont puissances d'un même troisième, c'est-à-dire s'il existe un mot w et deux entiers m et n tels que $u = w^m$ et $v = w^n$.

Les notions précédentes permettent d'introduire un résultat important :

PROPOSITION 1.1.2 *Deux mots u et v commutent si et seulement s'ils sont multiplicativement dépendants.*

PREUVE : Il est trivial que la dépendance multiplicative de u et v entraîne le fait qu'ils commutent. La réciproque se montrera par récurrence sur la $|uv| = |u| + |v|$:
- Si la concaténation uv est de longueur nulle, alors u et v le sont aussi obligatoirement, donc $u = v = \varepsilon$ et le résultat est acquis.
- Soit alors n un entier strictement positif et supposons que le résultat est acquis lorsque uv est de longueur strictement inférieure à n. Soient alors u et v deux mots dont la concaténation est de longueur n. On distingue alors les cas suivants :
  - si $u = \varepsilon$ ou $v = \varepsilon$, le résultat est trivial ;
  - si $|u| = |v|$, comme $uv = vu$, u et v sont égaux, puisque l'écriture des mots est unique, et le résultat est démontré ;

– sinon on peut supposer $|u| < |v|$. L'égalité $uv = vu$ impose alors que u soit préfixe de v, donc il existe un mot w pour lequel $v = uw$, donc $uuw = uwu$ et donc $uw = wu$. Or w est strictement plus court que v, donc $|uw| < n$, et par hypothèse de récurrence il existe un mot s et deux entiers m et p tels que $u = s^m$ et $w = s^p$, et donc $v = s^{m+p}$.

■

1.1.3 Équations aux mots

Ces définitions étant posées, il est naturel de vouloir étudier des équations plus générales faisant intervenir des mots. Commençons par définir précisément ce que l'on entend par équation.

DÉFINITION 1.1.5 (ÉQUATION AUX MOTS)
Soient Σ un alphabet et $V = \{x_i, \ldots, x_n\}$ un alphabet disjoint de Σ. Une **équation aux mots** sur Σ à inconnues dans V est une paire $\{\xi, \eta\}$ de mots sur $\Sigma \cup V$, que l'on pourra noter abusivement $\xi = \eta$.

On appelle **solution** d'une telle équation toute suite $(u_1, \ldots, u_n)$ de mots sur Σ telle que l'égalité

$$\xi[u_1/x_1, \ldots, u_n/x_n] = \eta[u_1/x_1, \ldots, u_n/x_n]$$

soit vérifiée.

On note naturellement $\xi[u_1/x_1, \ldots, u_n/x_n]$ le mot obtenu à partir de ξ en substituant à chaque x_i le mot u_i. Un tel mot est donc élément de Σ^*. On appellera donc Σ l'alphabet des constantes et V l'alphabet de variables.

THÉORÈME 1.1.1 (MANAKIN, 1977) *Il existe un algorithme donnant une solution ou décidant l'absence de solution pour toute équation aux mots avec constante.*

La preuve de ce théorème est difficile (l'originale prend une dizaine de pages). Elle fournit un algorithme dont la complexité est une tour de six (ou sept) exponentielles. Une solution récente construit cependant un algorithme fonctionnant en espace polynômial.

EXEMPLE 1.1.1 *Soient $\Sigma = \{a, b\}$ et $V = \{x, y\}$, considérons l'équation $ax = yb$. Si (u, v) est une solution de cette équation, on a alors $au = vb$. Les mots u et v sont donc tous deux de longueur au moins 1, et il existe u' et v' tels que $u = u'b$ et $v = av'$. On a alors $au'b = av'b$ donc $u' = v'$. L'ensemble des solutions de cette équation s'écrit donc $\{(wb, aw) \mid w \in \Sigma^*\}$*
On peut également montrer que l'équation $ax = xb$ n'a pas de solution.

Si la preuve du théorème de Manakin est difficile, on peut en revanche prouver plus facilement un résultat complémentaire :

PROPOSITION 1.1.3 *Si (u, v) est solution d'une équation aux mots non triviale à deux inconnues et sans constante, alors u et v sont multiplicativement dépendants.*

PREUVE : exercice... ■

1.2 Langages formels

Une fois définis les alphabets et les mots et établis quelques résultats importants, passons maintenant à la définition des langages formels proporement dits.

1.2.1 Définitions

DÉFINITION 1.2.1 (LANGAGE FORMEL)
Soit Σ un alphabet. On appelle **langage formel** sur Σ, ou plus simplement langage sur Σ, toute partie de l'ensemble Σ^*.

De même que pour les mots, on peut définir un certain nombre d'opérations sur les langages. Outre les opérations ensemblistes classiques de réunion, d'intersection et de passage au complémentaire, on définit les opérations suivantes :

- **le produit** de deux langages L_1 et L_2 est l'ensemble des produits d'un mot de L_1 et d'un mot de L_2, c'est-à-dire

$$L_1 L_2 = \{uv \mid u \in L_1, v \in L_2\}$$

- **les puissances** d'un langage L se déduisent facilement :

$$L^0 = \{\varepsilon\} \qquad L^{n+1} = LL^n = LL^n \text{ pour tout } n$$

- **l'étoile** ou opération de Kleene, qui à un langage L associe la réunion de l'ensemble des puissances de L :

$$L^* = \bigcup_{n \geq 0} L^n$$

  On montre facilement que L^* est le plus petit langage contenant L et le mot vide qui soit stable par concaténation.

REMARQUE : Dans le cas de la réunion ensembliste, on notera parfois $A + B$ pour désigner la réunion $A \cup B$.

PROPOSITION 1.2.1 *Pour tous langages K et L sur un alphabet Σ, on a*

$$(K + L)^* = (K^* L)^* K^*$$

PREUVE :

- $(K + L)^* \subseteq (K^* L)^* K^*$: Considérons un mot u de $(K + L)^*$ et raisonnons par récurrence sur $|u|$. Pour $|u| = 0$, donc $u = \varepsilon$, le cas est trivial puisque ε est élément de $(K^* L)^*$ et de K^*. Pour $|u| > 0$, on peut écrire u comme la concaténation des u_i pour $1 \leq i \leq n$ avec les u_i éléments de $K \cup L$. Si tous les u_i sont éléments de K, u est élément de K^* qui est clairement inclus dans $(K^* L)^* K^*$. Sinon soit k le plus petit indice pour lequel $u_k \in L \setminus K$. On a alors

$$u = \underbrace{u_1 \ldots u_{k-1}}_{K^*} \cdot \underbrace{u_k}_{L} \cdot u'$$

  et u' est un élément de $(K + L)^*$ de longueur strictement inférieure à $|u|$. Par hypothèse de récurrence, u' est donc élément de $(K^* L)^* K^*$, et comme $u_1 \ldots u_k$ est clairement élément de $K^* L$, u est lui aussi élément de $(K^* L)^* K^*$.

- $(K^*L)^*K^* \subseteq (K+L)^*$: Réciproquement, tout mot de $(K^*L)^*K^*$ est clairement une concaténation de mots de K et de L, donc élément de $(K+L)^*$. L'égalité est donc prouvée.

■

REMARQUE : On a aussi les relations suivantes :
- $(KL)^n = K(LK)^*L + \{\varepsilon\}$
- $K^* = (K^n)^*\big(\{\varepsilon\} + K + \ldots + K^{n-1}\big)$

1.2.2 Équations aux langages, lemme d'Arden

La définition des équations aux langages se fait de façon intuitive en utilisant les opérateurs définis précédemment. On n'a cependant pas de résultat aussi général ici que dans le cas des équations aux mots. On a tout de même un résultat intéressant sur une forme particulière d'équations :

LEMME 1.2.1 (LEMME D'ARDEN) *Soient A et B deux langages. L'équation $X = AX + B$ (respectivement $X = XA + B$) a pour plus petite solution A^*B (respectivement BA^*), et cette solution est unique lorsque ε n'est pas élément de A.*

PREUVE : Énonçons une preuve pour l'équation $X = AX + B$, sachant que l'autre cas se prouve de façon identique.

- **validité** En admettant des résultats classiques sur les opérations entre langages, on a facilement

$$A(A^*B) + B = (AA^*)B + B = (AA^* + \{\varepsilon\})B = A^*B$$

donc A^*B est bien solution de l'équation.

- **minimalité** Soit L une solution de l'équation. On a donc $L = AL + B$. On a alors, pour tout entier n, l'égalité $L = A^{n+1}L + \bigcup_{0\leq i\leq n} A^iB$. En effet le cas $n = 0$ est la définition de L et si le résultat est établi pour n donné on a clairement

$$L = A^{n+1}(AL+B) + \bigcup_{0\leq i\leq n} A^iB = A^{n+2}L + A^{n+1}B + \bigcup_{0\leq i\leq n} A^iB$$

d'où le résultat annoncé. Par conséquent, L contient tous les A^nB, ce qui prouve que A^*B est inclus dans L.

- **unicité** Supposons à présent que A ne contient pas le mot vide et considérons une solution L de l'équation. On sait déjà que L contient A^*B. Considérons alors un mot u de L et montrons qu'il est élément de A^*B.
  - si $|u| = 0$, on a $u = \varepsilon$, donc comme u est la concaténation d'un mot de AL et d'un mot de B, B contient le mot vide, et donc A^*B également et $u \in A^*B$;
  - sinon notons n la longueur de u. D'après la construction de proof de minimalité, u est alors élément de $A^{n+1}L + \bigcup\limits_{0\leq i\leq n} A^iB$, mais comme A ne contient pas le mot vide, tout mot de A^{n+1} et donc de $A^{n+1}L$, est

de longueur au moins $n+1$, et donc u est élément de $\bigcup_{0\leq i\leq n} A^iB$ et donc de A^*B. On a donc $L = A^*B$ est l'unicité est démontrée.

■

Cela dit, cette preuve aurait été plus élégante en utilisant le résultat très général suivant, qui est une restriction d'un théorème fondateur sur les ordres complets partiels (ou cpo) :

THÉORÈME 1.2.1 (POINT FIXE DE KNASTER-TARSKI) *Soit E un ensemble ordonné ayant un plus petit élément e et dans lequel tout suite croissante admet une borne supérieure. Soit alors une application g sur E croissante et continue au sens des cpo, c'est-à-dire qui vérifie pour toute suite (u_n) croissante*

$$g\left(\operatorname{Sup}_n u_n\right) = \operatorname{Sup}_n g\left(u_n\right).$$

Alors g admet un plus petit point fixe qui s'exprime $\operatorname{Sup}_n g^n(e)$.

PREUVE : On considère la suite $\left(g^n(e)\right)_{n\geq 0}$. Comme e est plus petit élément, on a $e \leq g(e)$, puis comme g est croissante, par récurrence immédiate on obtient $\forall n, g^n(e) \leq g^{n+1}(e)$. Par hypothèse le sup de $g^n(e)$ existe et

$$g\Big(Sup(g^n(e))\Big) = Sup\big(g^{n+1}(e)\big).$$

Alors $Sup(g^n(e))$ est un point fixe, s'il existe un plus petit point fixe c : $e \leq c$ donc $\forall n, g^n(e) \leq c$ et en passant au Sup, on a le résultat voulu. ■

REMARQUE : On montre alors le lemme d'Arden en considérant sur l'ensemble $\mathcal{P}(\Sigma^*)$ muni de l'inclusion l'application $L \to AL + B$

Chapitre 2

Automates finis

Maintenant que la notion générale de langage formel est introduite, nous pouvons passer au cœur du problème, à savoir la classification de ces langages formels.

La première classe que nous étudions est celle des langages rationnels, ou réguliers. Ils se définissent par l'intermédiaire de la notion d'automate fini que nous allons introduire maintenant.

2.1 Définitions

Le premier modèle que nous introduisons est le modèle des automates déterministes. Il en existe beaucoup de définitions équivalentes. Nous utiliserons (par exemple) celle-ci :

Définition 2.1.1 (automate fini déterministe)
On appelle automate fini déterministe, ou AFD, tout quintuplet $M = (Q, \Sigma, \delta, q_0, F)$ où

- Q est un ensemble fini (les états) ;
- Σ est un alphabet ;
- q_0, élément de Q, est l'état initial ;
- F, partie de Q, est l'ensemble des états finaux ;
- δ, application de $Q \times \Sigma$ dans Q, est la fonction de transition.

L'automate M est dit *complètement spécifié* lorsque δ est définie sur tout $Q \times \Sigma$.

Définition 2.1.2 (Marche)
Si l'on note une transition $q \stackrel{a}{\to} q'$ lorsque $q' = \delta(q, a)$, une marche d'un tel automate est alors une suite de transitions à partir de l'état initial q_0, c'est-à-dire une suite

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} q_{n-1} \xrightarrow{a_n} q_n$$

Formellement, on étend la définition de l'application δ à $Q \times \Sigma^*$ en posant pour tout état q :

$$\delta(q, \varepsilon) = q \qquad \text{et} \qquad \delta(q, au) = \delta(\delta(q, a), u) \quad \text{pour } a \in \Sigma \text{ et } u \in \Sigma^*$$

Définition 2.1.3
Le langage **reconnu par l'automate** M est alors l'ensemble des mots qui mènent à un état d'acceptation, c'est-à-dire

$$L(M) = \{u \mid u \in \Sigma^*, \delta(q_0, u) \in F\}$$

On parle d'**automate déterministe** parce que pour un état q et un symbole a donnés, il existe au plus une transition et donc au plus un état successeur $\delta(q, a)$. On peut cependant définir un modèle non déterministe de façon naturelle :

Définition 2.1.4 (automate fini non déterministe)
Un automate fini non déterministe, ou AFND, est un quintuplet $M = (Q, \Sigma, q_0, F, \delta)$ dont les éléments sont définis comme dans le cas déterministe excepté δ qui est une application de $Q \times \Sigma$ dans l'ensemble $\mathcal{P}(Q)$ des parties de Q.

Dans ce modèle, il est possible qu'à partir d'un état donné et d'un symbole donné, plusieurs transitions soient empruntables. On étend cette fois la fonction δ en posant

$$\delta(q, \varepsilon) = \{q\} \qquad \text{et} \qquad \delta(q, au) = \bigcup_{r \in \delta(q,a)} \delta(r, u) \quad \text{pour } a \in \Sigma \text{ et } u \in \Sigma^*$$

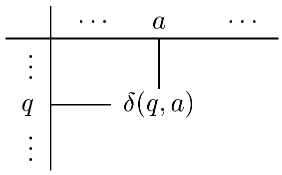
Le langage reconnu par un tel objet sera alors l'ensemble des mots qui *peuvent* mener à un état d'acceptation, c'est-à-dire

$$L(M) = \{u \mid u \in \Sigma^*, \delta(q_0, u) \cap F \neq \emptyset\}$$

2.2 Représentation d'un automate fini

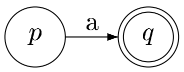
Ces définitions formelles des automates finis ont certes la précision nécessaire, mais elles ne permettent pas de visualiser aisément la nature des objets qu'elles représentent. C'est pourquoi on a coutume de représenter les automates sous des formes différentes.

2.2.1 Table de transition



Une première façon de représenter un automate est simplement de représenter la fonction δ en extension. Cette méthode n'est pas extrêmement visuelle, surtout dans le cas d'automates importants, de plus il n'est pas toujours nécessaire de spécifier complètement un automate. Cependant, on pourra parfois l'utiliser à profit, en particulier lors de la minimisation des automates ou lors de la déterminisation.

2.2.2 Graphe



La représentation la plus fréquemment utilisée est celle sous forme de graphe. Ici un sommet du graphe correspond à un état de l'automate et une transition correspond à une arête étiquetée par le symbole associé à la transition. On représente les états terminaux avec un double contour pour les différencier des autres états.

2.3 Algorithme de déterminisation

Le premier résultat important sur les automates finis est le fait que les deux formalismes (déterministe et non déterministe) ont la même puissance. Plus précisément, on a le résultat suivant :

PROPOSITION 2.3.1 *Il existe un algorithme qui permet de transformer tout automate fini non déterministe en un automate fini déterministe reconnaissant le même langage.*

L'idée à la base de la déterminisation est, pour un automate non déterministe $\mathcal{M} = (Q, \Sigma, q_0, F, \delta)$, de construire un automate déterministe $\mathcal{M}'$ dont les états sont les parties de Q, qui sont en nombre fini puisque Q est fini. La fonction de transition de l'automate déterminisé représentera l'action de δ sur une partie de Q, et les états finaux de $\mathcal{M}'$ seront toutes les parties de Q contenant un élément de F :

$$\mathcal{M}' = \big(\mathcal{P}(Q), \Sigma, \delta', \{q_0\}, F'\big)$$

$$\text{avec} \quad \delta'(E,a) = \bigcup_{q\in E} \delta(q,a) \quad \text{et} \quad F = \{X \mid X \subseteq Q, X \cap F \neq \emptyset\}$$

PROPOSITION 2.3.2 *On a* $\mathcal{L}(\mathcal{M}) = \mathcal{L}(\mathcal{M}')$.

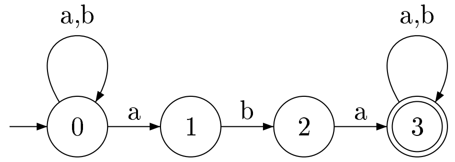
PREUVE : On montre par double inclusion :

- $\mathcal{L}(\mathcal{M}) \subseteq \mathcal{L}(\mathcal{M}')$. Soit $u \in \mathcal{L}(\mathcal{M})$. Si $u = \varepsilon$, cela veut dire que $q_0 \in F$ donc $\{q_0\} \in F'$ et $u \in \mathcal{L}(\mathcal{M}')$. Sinon, u s'écrit $a_1a_2\ldots a_p$ et le chemin défini par : q_0 et $\forall i \leq p-1, q_{i+1} = \delta(q_i, a_{i+1})$ est un chemin "bon" dans l'automate $\mathcal{M}$. De là, on va déduire un chemin "bon" dans l'automate $\mathcal{M}'$. On pose $X_0 = \{q_0\}$ et $\forall i, X_i = \delta'(X_{i-1}, a_i)$. Alors il existe un indice x_0 minimal tel que X_{i_O} contienne q_p (au pire, $i_0 = p$). Alors on a donc $X_{i_O} \in F'$ donc on a bien obtenu un "bon" chemin dans $\mathcal{M}'$.
- $\mathcal{L}(\mathcal{M}') \subseteq \mathcal{L}(\mathcal{M})$. Si $u = a_1 \ldots a_p \in L(\mathcal{M}')$, alors il existe un chemin victorieux (de $\{q_o\}$ à $X_p \in F'$ dans $\mathcal{M}'$) étiqueté par u et vérifiant $\forall i X_{i+1} = \delta'(X_i, a_{i+1})$, et $X_p \cap F' \neq \emptyset$. Comme $X_p \cap F' \neq \emptyset$ il existe $q_f \in F$ et $q_f \in X_p$. Donc $q_f = \bigcup_{x\in X_{p-1}} \delta(x, a_p)$. Donc q_f appartient à un des $\delta(x, a_p)$. Pour q_p on prend donc q_f et $q_{p-1} \in \delta(q_{p-1}, a_p)$, *etc.* On aboutit bien à l'existence d'un chemin victorieux dans $\mathcal{M}$.

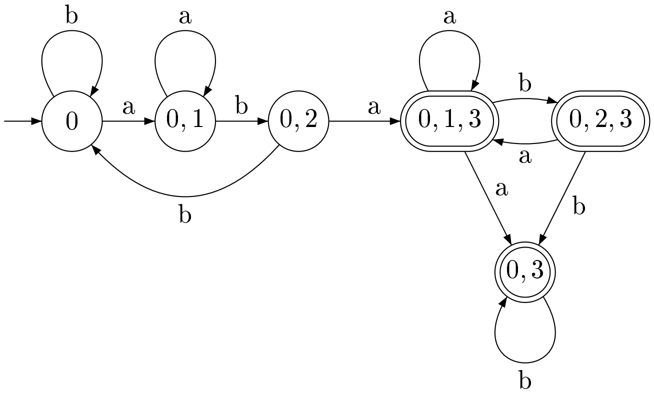
■

En pratique, on n'utilise pas l'ensemble des parties de Q directement mais plutôt l'ensemble des parties atteignables à partir du singleton $\{q_0\}$ afin de limiter le nombre d'états.

Donnons à présent un exemple de déterminisation d'automate. Considérons par exemple l'automate non déterministe suivant :



En appliquant la méthode expliquée au-dessus, on obtient donc l'automate déterminisé suivant :



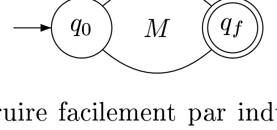
On peut alors remarquer que l'automate obtenu n'est pas minimal. En effet, l'ensemble constitué des états $\{0, 1, 3\}$, $\{0, 2, 3\}$ et $\{0, 3\}$ est stable par transition et les trois états sont acceptants, donc l'automate obtenu en remplaçant ces trois états par un seul état puits acceptant reconnaît le même langage. On reviendra plus précisément sur la minimalité des automates dans la section 2.5.

Quelques propriétés peuvent être démontrées sur ce formalisme. On laissera les suivantes en exercice.

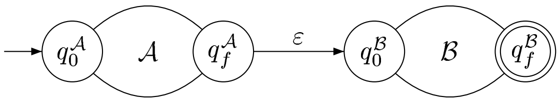
Exercice 2.3.1

- *Soit L un langage reconnu par un automate fini déterministe. Notons x^R le miroir de x, c'est-à-dire que $(u_1 \dots u_n)^R = u_n \dots u_1$. Existe-t-il un automate fini déterministe reconnaissant $L^R = \{x^R \mid x \in L\}$?*
- *Soit L un langage reconnu par un automate fini déterministe. Existe-t-il un automate fini déterministe reconnaissant le complémentaire de L ?*

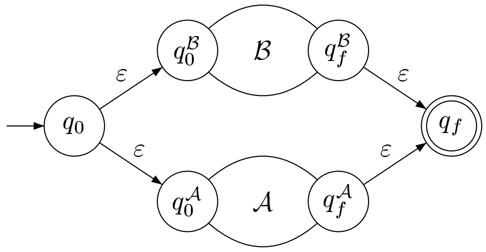
Certains formalismes autorisent les automates non déterministes à comporter un type de transition particulier, les ε-transitions. Il s'agit de transitions qui peuvent être empruntées instantanément, sans lire de caractère. Il est facile de voir que l'introduction de telles transitions ne modifie pas la puissance obtenue. De plus, on peut obtenir une forme standard pour les automates non déterministes avec exactement un état initial et un état final, ce que l'on représentera sous la forme suivante pour un automate M donné :



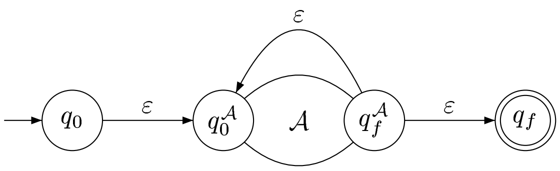
Ceci permet de construire facilement par induction des automates correspondant aux opérations classiques sur les langages rationnels :



La concaténation $\mathcal{A}.\mathcal{B}$



La réunion $\mathcal{A}+\mathcal{B}$



L'étoile $\mathcal{A}^*$

2.4 Lemme de l'étoile

Ce lemme est un résultat important permettant de prouver que certains langages ne peuvent pas être reconnus par des automates finis. On le trouve parfois dans la littérature sous le nom de *lemme de la pompe* ou *pumping lemma* en anglais. Il s'agit du lemme de la pompe *pour les langages reconnaissables par AF*.

LEMME 2.4.1 (DE L'ÉTOILE) *Si un langage L est reconnu par un automate fini, alors il existe un entier n tel que pour tout mot u de L de longueur au moins n il existe des mots x, v et y tels que $u = xvy$ et que*

- *v est non vide ;*
- *xv est de longueur au plus n ;*
- *pour tout entier i, xv^iy est élément de L.*

PREUVE : Soit $M = (Q, \Sigma, q_0, \delta, F)$ un automate fini déterministe reconnaissant le langage L. Soit alors un mot $u = u_1 \ldots u_p$ avec $p \geq |Q|$. Notons q_i les états traversés par l'automate lors de la lecture de u :

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_p} q_p$$

On traverse donc $p+1$ états, donc plus d'états qu'il y en a dans Q. Par conséquent, il existe deux entiers i et j tels que $i < j \leq n$ et $q_i = q_j$. Si on note $x = a_1 \ldots a_i$, $v = a_{i+1} \ldots a_j$ et $y = a_{j+1} \ldots a_p$, on a donc $u = xvy$ et comme $\delta(q_i, v) = q_i$, pour tout entier k on a $\delta(q_i, v^k) = q_i$ d'où $xv^ky \in L$. Or par définition de i et j, xv est de longueur $|Q|$ au plus, donc le résultat est démontré. ■

On en déduit le théorème :

THÉORÈME 2.4.1 *Si L est reconnaissable par automate fini, $\exists N$ tel que $\forall u = svt \in L$ avec $|v| \geq N$, $\exists x, y, z$ tels que $u = sxyzt$ et $1 \leq |y| \leq |N|$, vérifiant $\forall i, sxy^izt \in L$.*

REMARQUE : Il existe des langages non reconnaissables par AF qui satisfont la condition du lemme de l'étoile. $\{a\}\{a\}^*L + \{b\}^*$ où L n'est pas reconnu par AF et $L \in \{b\}^*$ (prendre par exemple $L = \{b^p \mid \text{ p premier}\}$).

Pour obtenir une équivalence, il faut compliquer les hypothèses :

EXEMPLE 2.4.1 *L est reconnaissable par AF ssi $\exists N > 0$ tel que si $u \in \Sigma^*$ est de taille $\geq N$, alors il existe x, y, z, $y \neq \varepsilon$, tels que $u = xyz, |xy| \leq N$, et $\forall v \in \Sigma^*, \forall i \geq 0$, $uv \in L \Leftrightarrow xy^izv \in L$.*

PREUVE : Laissée en exercice ... ■

EXEMPLE 2.4.2 *L'utilisation du lemme de l'étoile permet facilement de montrer, par exemple, que le langage $L = \{a^nb^n \mid n \leq 0\}$ n'est pas reconnaissable par un automate fini. En effet, si tel était le cas, il existerait un entier n vérifiant les proporiétés du lemme. Par conséquent il serait possible de décomposer le mot a^nb^n sous la forme xvy avec $|xv| \leq n$ donc $x = a^{|x|}$ et $v = a^{|v|}$. Par conséquent, le lemme prouverait que tous les xv^ky, c'est-à-dire les $a^{n+(k-1)|v|}b^n$, sont éléments de L, ce qui est manifestement faux.*

EXERCICE 2.4.1 *Montrer que le langage L composé des 1^p où p est premier n'est pas rationnel. Utilisez le lemme de l'étoile.*

2.5 Minimisation des automates déterministes

On a vu en 2.3 que la déterminisation systématique d'un automate non déterministe peut mener à la création d'un certain nombre d'états superflus. Dans le cas des automates finis déterministes (et complètement spécifiés), on dispose en fait de résultats permettant de construire des automates minimaux.

Dans cette section, on présente une méthode de minimisation d'automates déterministes, et on présentera dans la suivante des résultats plus généraux.

Schématiquement, la minimisation d'un automate fini déterministe se fera en deux étapes principales :

1. suppression des états inutiles, c'est-à-dire ceux qui ne sont pas sur un chemin menant de q_0 à F, en les remplaçant par un état puits unique ;
2. passage au quotient pour une relation d'équivalence adaptée sur les états.

La validité de la première étape étant triviale, on n'explique que la seconde.

La relation d'équivalence utilisée se définit de la façon la plus directe qui soit : on définit que deux états sont équivalents lorsque, considérés comme états de départ, ils conduisent à l'acceptation du même langage. Plus formellement, on pose donc :

DÉFINITION 2.5.1 ($\equiv$)

$$q \equiv q' \quad \text{si et seulement si} \quad \forall u \in \Sigma^*,\ \delta(q,u) \in F \Leftrightarrow \delta(q',u) \in F$$

Si l'automate initial est $M = (Q, \Sigma, \delta, q_0, F)$, l'**automate quotient** sera donc $M = (Q', \Sigma, \delta', q'_0, F')$ avec $Q' = Q/\equiv$.

Il est facile de voir que la relation $\equiv$ ainsi construite est compatible avec la fonction de transition, c'est-à-dire que si deux états q et q' sont équivalents, alors pour toute lettre a les états $\delta(q,a)$ et $\delta(q',a)$ le sont également. La fonction δ passe donc au quotient et on peut poser

$$\forall q \in Q,\ \forall a \in \Sigma,\ \delta'(\bar{q}, a) = \overline{\delta(q,a)}$$

REMARQUE : Il est facile également de voir que F passe au quotient, c'est-à-dire que F est une réunion de classes d'équivalences de $\equiv$. Ceci provient du fait qu'un état q_f de F ne peut jamais être équivalent à un état q de $Q \backslash F$, puisque $\delta(q_f, \varepsilon)$ est égal à q_f qui est élément de F alors que $\delta(q, \varepsilon)$ est égal à q qui n'est pas élément de F. On peut donc poser $F' = F/\equiv$.

Par conséquent, si q'_0 est la classe d'équivalence de q_0, l'automate M' reconnaît effectivement le même langage que M, et aucun état n'y est superflu.

On peut définir de façon intéressante une famille similaire de relations d'équivalence. Leur définition consistera à ne considérer que des mots d'un nombre de lettres déterminé pour déterminer l'équivalence entre les états :

DÉFINITION 2.5.2 ($\equiv_k$)
Pour tout entier k, on définit :

$$q \equiv_k q' \quad \text{si et seulement si} \quad \forall u \in \Sigma^*,\ |u| \leq k,\ \delta(q,u) \in F \Leftrightarrow \delta(q',u) \in F$$

Les $\equiv_k$ partitionnent donc Q de plus en plus finement à mesure que k croît. En particulier, $\equiv_0$ partitionne Q en deux parties, F et $Q \backslash F$.

On peut en fait reformuler la définition de ces relations de la façon suivante :

PROPOSITION 2.5.1 $q \equiv_{k+1} q' \Leftrightarrow \begin{cases} q \equiv_k q' \\ \forall a \in \Sigma, \delta(q,a) \equiv_k \delta(q',a) \end{cases}$

PREUVE : On montre par double implication :

- Si $q \equiv_{k+1} q'$ alors en particulier $q \equiv_k q'$. Ensuite, prenons $a \in \Sigma$ et u tel que $|u| \leq k$. Alors

  $$\delta\big(\delta(q,u),a\big) \in F \Leftrightarrow \delta(q,ua) \in F \Leftrightarrow_{|ua| \leq k+1} \delta(q',ua) \in F \Leftrightarrow \delta\big(\delta(q',u),a\big) \in F.$$

  Remarquons que ça marche bien uniquement car c'est déterministe.
- Si $q \equiv_k q'$ et $\forall a \in \Sigma, \delta(q,a) \equiv_k \delta(q',a)$. Si $u \leq k$ est OK. Sinon on pose $u = au'$. Par hypothèse : $\widetilde{\widetilde{q}} = \delta(q',a) \equiv_k \delta(q,a) = \widetilde{q}$.
  Donc $\delta(\widetilde{q}) \in F \Leftrightarrow_{|u'| \leq k} \delta(\widetilde{\widetilde{q}}, u') \in F$

■

Un autre résultat important est le suivant, qui nous servira par la suite dans une caractérisation générale des langages rationnels :

PROPOSITION 2.5.2 *Pour tout automate fini, il existe un entier k_0 pour lequel la relation d'équivalence $\equiv_{k_0}$ est égale à la relation $\equiv$ définie précédemment.*

PREUVE : La relation $\equiv$ peut se définir comme la conjonction des $\equiv_k$ pour k parcourant $\mathbb{N}$, c'est-à-dire que

$$q \equiv q' \iff \forall k \in \mathbb{N},\ q \equiv_k q'$$

Or l'ensemble des relations d'équivalence sur Q est fini puisque Q est fini, donc il y a une nombre fini p de $\equiv_k$ distincts, que l'on peut noter $\equiv_{k_1} \dots \equiv_{k_p}$, avec k_i croissant en fonction de i. On a donc

$$q \equiv q' \iff \bigwedge_{i=1}^{p} (q \equiv_{k_i} q')$$

Or il est clair que si on a $k_i < k_j$, alors $\equiv_{k_j}$ est plus fine que $\equiv_{k_i}$, c'est à dire que $\equiv_{k_i}$ implique logiquement $\equiv_{k_j}$, par conséquent la conjonction des $\equiv_{k_i}$ est égale à $\equiv_{k_p}$, ce qui prouve le résultat voulu. ■

2.6 Automate minimal

On montre donc maintenant des résultats généraux sur la minimalité des automates finis. On définit en particulier une méthode de construction d'un automate minimal reconnaissant un langage rationnel donné.

2.6.1 Résiduels d'un langage

La notion qui nous permet d'aboutir à nos fins est celle de *résiduel* d'un langage. On la définit de la façon suivante :

DÉFINITION 2.6.1 (RÉSIDUEL)
Pour un langage formel L donné sur un alphabet Σ et pour un mot u donné quelconque sur Σ, on définit le **résiduel** (à gauche) de L associé au mot u comme étant le langage

$$u^{-1}L = \{v \mid v \in \Sigma^*,\, uv \in L\}$$

Cette définition mène à une caractérisation importante des langages reconnus par automates finis :

PROPOSITION 2.6.1 *Un langage L est reconnaissable par automate fini si et seulement si l'ensemble de ses résiduels à gauche est fini.*

PREUVE :

- : $\Rightarrow$ Si L est reconnaissable par un automate fini, soit $M = (Q, \Sigma, \delta, q_0, F)$ un automate fini déterministe le reconnaissant. Il est alors clair que le résiduel à gauche associé à un mot u donné est l'ensemble des mots reconnus par M en partant de l'état $\delta(q_0, u)$, donc chaque résiduel est le langage reconnu à partir de l'un des états de M. Or l'ensemble Q des états est fini, donc l'ensemble des résiduels l'est aussi.

- : $\Leftarrow$ Réciproquement, supposons fini l'ensemble des résiduels du langage L. On construit alors un automate $M = (Q, \Sigma, \delta, q_0, F)$ en définissant Q comme l'ensemble des résiduels de L, et en définissant la fonction de transition par $\delta(L', a) = a^{-1}L$. On pose ensuite $q_0 = L$ et un définit qu'un état est terminal si le langage qui lui est associé contient le mot vide. On voit alors facilement que l'automate obtenu reconnaît L, puisqu'un mot u est élément de L si et seulement si $u^{-1}L$ contient ε.

■

COROLLAIRE 2.6.1 *Si $\mathcal{M}$ est un automate fini déterministe complétement spécifié reconnaissant L, alors $\left|\{u^{-1}L \mid u \in \Sigma^*\}\right| \leq \left|\overline{Q}\right|$, et si $\mathcal{M}$ est tel que tout état soit accessible, $\left|\{u^{-1}L \mid u \in \Sigma^*\}\right| = \left|\overline{Q}\right|$*

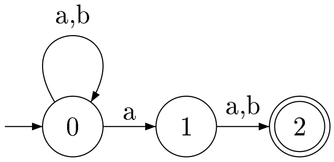
On déduit de ce corollaire la construction de l'automate minimal.

COROLLAIRE 2.6.2 *Il existe "un seul" (i.e. au nom des états près) automate minimal reconnaissant un langage $\mathcal{L}$ reconnu par un AF.*

On sait donc en minimisant déterminer l'égalité de deux langages rationnels.

2.6.2 Exemple de minimisation d'un AF

On va réaliser la minimisation de l'automate suivant :



On commence par déterminiser l'automate :

	a	b
0	01	0
1	2	2
2	-	-

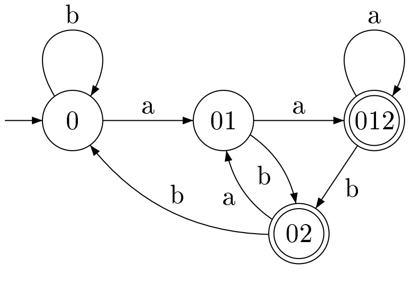
$\longrightarrow_{det}$

	a	b
0	01	0
01	012	02
012	012	02
02	01	0

Puis on "sépare" les états : $\equiv_0 \begin{cases} \{0, 01\} \\ \{012, 02\} \end{cases}$

On sépare avec a les deux paires d'états, d'où $\equiv_1 \begin{cases} \{0\}, \{01\} \\ \{012\}, \{02\} \end{cases}$

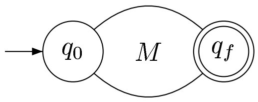
Finalement, on obtient l'automate suivant :



Remarque : Si le nombre d'états de l'AFD est bien minimal, il est ici plus grand que le nombre d'états du non déterministe initial. On a multiplié le nombre de flèches, cela dit.

2.7 Stabilité de la classe des langages reconnaissables par AF

Lemme 2.7.1 (Déjà vu) *Tout langage reconnu par AF l'est par un automate de la forme suivante :*



Ce lemme bien pratique nous permet de montrer les propositions suivantes :

Proposition 2.7.1 *REC est close par union, produit, étoile, complément, intersection.*

Chapitre 3

Langages rationnels

3.1 Introduction

DÉFINITION 3.1.1
Les langages sur l'alphabet Σ reconnus par automates finis forment l'ensemble $REC(\Sigma)$.

Le problème posé ici est le suivant : étant donné un automate $\mathcal{A} = (Q, \Sigma\delta, q_0, F)$ qui reconnaît un certain langage L, peut-on se donner une autre expression de L ?

Dans toute la suite, on notera $Q = \{q_0, q_1, \ldots q_n\}$. Alors, considérons les ensembles suivants :

- $L_i = \{u \in \Sigma^* \mid \delta(q_i, u) \in F\}$
- $X_{i,j} = \{a \in \Sigma, \mid \delta(q_i, a) = q_j\}$
- $Y_i = \begin{cases} \{\varepsilon\} & \text{si } q_i \in F \\ \emptyset & \text{sinon} \end{cases}$

PROPOSITION 3.1.1 $L_i = Y_i + \displaystyle\sum_{j=0}^{n} X_{i,j} L_j$

PREUVE :

- Si $u \in L_i$, alors on peut passer de q_i à $q_f \in F$ en lisant u. Si $u = \varepsilon$, alors $q_i \in F$ donc $u \in Y_i$. Sinon, $u = av$, il existe j tel que $\delta(q_i, a) = q_j$ et on passe de q_j à q_f en lisant v. Donc $\exists j, u \in X_{i,j} L_j$.
- La réciproque se fait de manière analogue.

■

On obtient donc à l'aide de l'automate un système d'équations linéaires à gauche que l'on obtient par substitution et grâce au Lemme d'Arden (notons qu'ici, $\varepsilon \notin X_{i,j}$). Alors on obtient finalement $L_0 = L(\mathcal{A})$ comme somme, produit, étoile de langages rationnels.

EXEMPLE 3.1.1 *On obtient* $\begin{cases} L_0 = \{a\}L_0 + L_1 \\ L_1 = \{a\}L_1 + \{b\}L_0 + \{\varepsilon\} \end{cases}$ *, puis en laissant tomber les accolades lorsque l'ensemble est un singleton :* $L_1 = aL_1 + \big(bL_0 + \varepsilon\big)$*,* *d'où* $L_1 = a^*\big(bL_0 + \varepsilon\big)$ *et en remplaçant dans* L_0, *puis en appliquant Arden, on obtient*

$$L_0 = \big(a + ba^*b\big)^*.\big(ba^*\big)$$

3.2 Théorème de Kleene

3.2.1 La classe des langages rationnels

Définition 3.2.1
Soit Σ un alphabet. On considère la plus petite classe de langages contenant $\emptyset$, $\{\varepsilon\}$ et $\{a\}$, pour $a \in \Sigma$, et close par union, produit et étoile. On note cette classe $RAT(\Sigma)$. C'est la **classe des langages rationnels**.

Proposition 3.2.1 $RAT(\Sigma) = REC(\Sigma)$

Preuve : On montre bien sur ceci par double inclusion :
- $RAT(\Sigma) \subseteq REC(\Sigma)$: On considère $REC(\Sigma)$: elle contient $\emptyset$, $\{\varepsilon\}$ et $\{a\}$, pour $a \in \Sigma$, et est close par union, produit et étoile. Et comme l'autre est plus petite...
- $REC(\Sigma) \subseteq RAT(\Sigma)$ Cette inclusion provient de la résolution du système du paragraphe précédent.

■

3.2.2 Expressions rationnelles

Comment dénoter de façon efficace un langage rationnel ?

Soit Σ un alphabet. On se donne les symboles $\underline{\emptyset}$, $\underline{\varepsilon}$, $\underline{a}$ pour $a \in \Sigma$, $\underline{*}$, $\underline{.}$, et $\underline{+}$. Alors :

Définition 3.2.2 (ER)
$\underline{\emptyset}$, $\underline{\varepsilon}$, $\underline{a}$ pour $a \in \Sigma$ sont des **expressions rationnelles**, et si u et v sont des expressions rationnelles (ER), alors $\underline{*}v$, $\underline{.}uv$, et $\underline{+}uv$ sont des ER. Ne sont ER **que** les expressions obtenues de cette manière. On note cet ensemble $ERAT(\Sigma)$.

Donnons maintenant une interprétation λ de $ERAT(\Sigma)$ dans l'ensemble des parties de Σ^*. On la définit de la façon suivante :
- $\lambda(\underline{\emptyset}) = \emptyset$,
- $\lambda(\underline{\varepsilon}) = \{\varepsilon\}$,
- $\lambda(\underline{+}uv) = \lambda(u) + \lambda(v)$, ...

A partir de maintenant, au lieu de $\underline{+}uv$ on écrit $u\underline{+}v$, au lieu de $\underline{*}u$ on écrit u^*, ... Pour enlever les ambiguités, on dit que l'étoile est prioritaire sur le + et on enlève le soulignement.

Définition 3.2.3
Un langage L sur Σ^* est **dénoté** par une ER si il existe une ER α telle que $L = \lambda(\alpha)$.

Exemple 3.2.1 *Les ER* $(a+b)^*$ *et* $(a^*b)^*a*$ *dénotent toutes deux le langage* $\{a,b\}^*$.

Sur cet exemple, on voit bien qu'il peut y avoir une infinité d'expressions rationnelles qui dénotent le même langage. Alors, pour être tout à fait rigoureux (!!!) on dit qu'une ER est un représentant d'une classe de congruence, cette congruence étant la plus petite congruence contenant les relations entre ces objets, nécessaires pour refléter les propriétés fondamentales des opérations entre les langages. Par exemple on doit avoir $(\alpha+\beta)+\gamma=\alpha+(\beta+\gamma)$, $\varepsilon\alpha=\alpha\varepsilon=\alpha$, $\alpha+\beta=\beta+\alpha$, ...
REMARQUE : On peut aussi dire que l'on considère les classes d'ER modulo λ. En fat, ce que l'on utilise, c'est le fait que deux ER qui dénotent le même langage sont équivalentes.

3.2.3 Théorème de Kleene

THÉORÈME 3.2.1 $RAT(\Sigma)=REC(\Sigma)=ERAT(\Sigma)$.

PREUVE : La première égalité à été démontrée précédemment. Ensuite $RAT\subseteq ERAT$ provient de la résolution du système. On va ici montrer $ERAT\subseteq REC$ en introduisant la notion d'ER dérivée (on aurait pu s'en passer). ■

DÉFINITION 3.2.4 (DÉRIVÉE D'UNE ER PAR RAPPORT À UNE LETTRE)
Soit α une ER sur Σ. Soit $a\in\Sigma$. On définit la dérivée de α par rapport à a, notée $\delta_a(\alpha)$ de la façon suivante :
- $\delta_a(\varepsilon)=\delta_a(b)=\delta_a(\emptyset)=\underline{\emptyset}=\emptyset$ $(b\neq a)$,
- $\delta_a(a)=\varepsilon$,
- $\delta_a(\alpha^*)=\delta_a(\alpha)\alpha^*$,
- $\delta_a(\alpha+\beta)=\delta_a(\alpha)+\delta_a(\beta)$,
- $\delta_a(\alpha.\beta)=\begin{cases}\delta_a(\alpha).\beta+\delta_a(\beta).\alpha & \text{si } \varepsilon\in\lambda(\alpha)\\ \delta_a(\alpha).\beta & \text{sinon}\end{cases}$

On réalise ensuite l'extension suivante :

DÉFINITION 3.2.5 (DÉRIVÉE PAR RAPPORT À UN MOT)
Soit $u\in\Sigma^*$, alors on définit par induction :
- si $u=\varepsilon$, $\delta_u(\alpha)=\alpha$;
- si $u=va$, $\delta_u(\alpha)=\delta_a\big(\delta_v(\alpha)\big)$.

La proposition suivante donnera immédiatement la propriété $ERAT\subseteq REC$:

PROPOSITION 3.2.2 *On obtient les deux résultats suivants :*
1. $\lambda\big(\delta_a(\alpha)\big)=u^{-1}L(\alpha)$
2. *L'automate suivant :*
$$\Big(\{\delta_u(\alpha)\mid u\in\Sigma^*\},\Sigma,\delta,\alpha,\{\delta_u(\alpha)\mid u\in\lambda(\alpha)\}\Big),$$
avec $\delta\big(\delta_u(\alpha),a)\big)=\delta_a\big(\delta_u(\alpha)\big)$ *reconnaît* $\lambda(\alpha)$.

PREUVE :
1. Double inclusion et induction structurelle sur les ER. A faire !

2. Notons que l'ensemble Q est a-priori infini, mais modulo les classes d'équivalences des ER, on ne considère qu'un nombre fini de représentants. La suite est à faire en exercice.

■

3.3 Caractérisations Algébriques

THÉORÈME 3.3.1 (NÉRODE) *Il y a équivalence entre les assertions suivantes :*

1. *L est rationnel.*
2. *L est réunion de classes d'une congruence d'index fini.*
3. *L est réunion de classes d'une congruence à gauche d'index fini.*
4. *La relation $\mathcal{R}_L$ définie par $u\mathcal{R}_L v$ ssi $\forall t \in \Sigma^*, tu \in L \Leftrightarrow tv \in L$ est une congruence à gauche d'index fini.*

PREUVE :

$1 \Rightarrow 2$: Si L est rationnel, $L = L(\mathcal{M})$ où $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$ est un automate fini déterministe. Considérons $u \equiv_{\mathcal{M}} v$ ssi $\delta(\bullet, u) = \delta(\bullet, v)$ avec $\delta(\bullet, u)$ l'application $Q \to Q$ telle que $q \mapsto \delta(q, u)$ (on a en fait une égalité d'applications). C'est une relation d'équivalence, c'est même une congruence car la définition de δ est compatible. De plus, montrons qu'elle est d'index fini : On considère l'application $\Sigma^*_{/\equiv_{\mathcal{M}}} \to Q^Q$, définie par $\overline{u}^{\mathcal{M}} \mapsto \delta(\bullet, u)$. On vérifie que c'est bien une application à l'aide de la définition, et comme elle est surjective sur son image, qui est un ensemble fini, l'ensemble $\Sigma^*_{/\equiv_{\mathcal{M}}}$ est fini. Et $L = L(\mathcal{M}) = \bigcup_{x \in L} \overline{x}^{\mathcal{M}}$ est une union finie.

$2 \Rightarrow 3$ Si il existe une congruence, j'ai une congruence à gauche (une congruence est compatible à droite et à gauche).

$3 \Rightarrow 4$ Par hypothèse, $L = \bigcup_{x \in L} \overline{x}^{\mathcal{R}}$ où $\mathcal{R}$ est une congruence à gauche d'index fini, *i.e.* , $\left|\Sigma^*_{/\mathcal{R}}\right|$ est fini. Considérons $\mathcal{R}$. C'est une congruence à gauche, c'est-à-dire $u\mathcal{R}_L v \Rightarrow \forall t, tu\mathcal{R}_L tv$. En effet, $w(tu) \in L \Leftrightarrow (wt)u \in L \Rightarrow (wt)v \in L$ soit $w(tv) \in L$. Cette congruence est-elle d'index fini ? On considère l'application $\varphi : \Sigma^*_{/\mathcal{R}} \to \Sigma^*_{/\mathcal{R}_L}$, définie par $\overline{u}^{\mathcal{R}} \mapsto \overline{u}^{\mathcal{R}_L}$. Elle est bien définie car $v \equiv u[\mathcal{R}] \Rightarrow v \equiv u[\mathcal{R}_L]$ (en effet, soit $tv \in L$, $\mathcal{R}$ étant congruente à gauche, on a $tu\mathcal{R}tv$. Or$L = \bigcup \overline{x}^{\mathcal{R}}$ donc il existe x, $tv\mathcal{R}x$, et donc $tu\mathcal{R}x$, et finalement $tu \in \overline{x}\mathcal{R} \subseteq L$). De plus elle est surjective donc $\left|\Sigma^*_{/\mathcal{R}_L}\right|$ est fini.

$4 \Rightarrow 1$ Considérons les résiduels de L. Alors $\left(u^{-1}L = v^{-1}L\right) \iff \left(\forall t, tu \in L \Leftrightarrow tv \in L\right)$. Comme la relation est d'index fini, il y a un nombre fini de résiduels, donc L est rationnel (car reconnaissable).

■

Et maintenant, une autre caractérisation alégébrique des langages rationnels :

Définition 3.3.1 (Congruence syntaxique)
On définit sur Σ^* une congruence **dite syntaxique** par :

$$uS_Lv \Longleftrightarrow \forall s,t \in \Sigma^*, \big(sut \in L \Leftrightarrow svt \in L\big)$$

Proposition 3.3.1 *L est rationnel ssi son monoïde syntaxique est fini.*

Preuve :
- $\Rightarrow$: Si L est rationnel, soit $\mathcal{M}$ un automate fini le reconnaissant :$\mathcal{M} = (Q,\Sigma,\delta,q_0,F)$. Alors considérons l'application $\varphi : \Sigma^*_{/\equiv_{\mathcal{M}}} \to \Sigma^*_{/S_L}$, définie par : $\overline{u}^{\mathcal{M}} \mapsto \overline{u}^{S_L}$ est bien définie (vérifier que si $v \equiv_{\mathcal{M}} u$ alors $sut \in L$). C'est une surjection donc comme à gauche c'est fini, c'est aussi fini à droite.
- $\Leftarrow$: On considère cette fois l'application $\varphi : \Sigma^*_{/S_L} \to \Sigma^*_{/\mathcal{R}_L}$, définie par : $\overline{u}^{S_L} \mapsto \overline{u}^{\mathcal{R}_L}$, et on obtient $\Sigma^*_{/\mathcal{R}_L}$ fini, donc L est réunion finies de classes d'une congruence à gauche, donc L est rationnel.

■

Et encore une autre ...

Définition 3.3.2 (local)
On dit que L langage sur Σ est **local** si il existe des parties A et B de Σ, $Cde\Sigma^2$, vérifiant :

$$L = A\Sigma^* \cap \Sigma^*B - \Sigma^*C\Sigma^*.$$

Remarque : Un tel langage est toujours rationnel (pourquoi ?).

Proposition 3.3.2 *Un langage L sur Σ est rationnel ssi il est l'image d'un langage sur Γ local L' par un homomorphisme φ tel que $\varphi(\Gamma) \subseteq \Sigma$.*

Preuve :
- $\Rightarrow$: Si L est rationnel, il existe un automate $\mathcal{M}$ qui le reconnaît. Alors prenons pour alphabet $\Gamma = \{paq \mid p,q \in Q \text{ et } a \in \Sigma\}$, pour application $\varphi : \Gamma \to \Sigma$, $paq \mapsto a$. Alors en prenant les ensembles A, B, C suivants (parties de Γ et de Γ^2 pour C) :
  - $A = \{q_0ap, p \in Q \text{ et } a \in \Sigma\}$
  - $B = \{qaq_f, q \in Q, q_f \in F, a \in \Sigma\}$
  - $C = \{(qap, p'aq'), p, p', q, q' \in Q \text{ et } p \neq p'\}$

  $L' = A\Gamma^* \cap \Gamma^*B - \Sigma^*C\Gamma^*$. est bien local et L est son image réciproque par φ.
- $\Leftarrow$: La réciproque est triviale.

■

3.4 Hauteur d'étoile

DÉFINITION 3.4.1 (HAUTEUR D'ÉTOILE D'UNE E.R.)
Soit α une ER. SA **hauteur d'étoile** est définie par induction sur α de la façon suivante :
- $h(\emptyset = h(a) = h(\varepsilon) = 0$,
- $h(\beta + \gamma) = h(\beta\gamma) = max\{h(\beta), h(\gamma)\}$,
- $h(\beta^*) = h(\beta) + 1$.

DÉFINITION 3.4.2 (HAUTEUR D'ÉTOILE D'UN LANGAGE)
Soit un langage L. On définit sa hauteur d'étoile par :

$$h(L) = Min\{h(\alpha) \mid \lambda(\alpha) = L\}$$

EXEMPLE 3.4.1 $h\big((a + b)^*\big) = 1$, $h\big((a^*b)^*a^*\big) = 2$

Questions :
- Est ce que pour tout n, on peut trouver un langage rationnel de hauteur d'étoile n ? La réponse est OUI. On peut montrer cela en considérant les langages dénotés par $\alpha_0 = \varepsilon$, $\alpha_{n+1} = \Big(a \ldots a\alpha_n b \ldots b\Big)^*$ (2^n fois la lettre "a", 2^n fois la lettre "b").
- Existe-t-il un algorithme qui donne la hauteur d'étoile d'un langage dénoté par une expression rationnelle α ? OUI Hashigushi en 1988 (difficile !).

DÉFINITION 3.4.3 (E.R. GÉNÉRALISÉE)
Dans la définition des ER, on ajoute l'intersection et le complémentaire. On obtient ce que l'on appelle des **expressions rationnelles généralisées**.

On en déduit une notion de hauteur d'étoile généralisée notée h_g. On voit bien que cette hauteur d'étoile va être plus petite du fait de l'intervention des deux nouveaux symboles.

A ce jour, on ne connaît pas de langage de hauteur d'étoile généralisée supérieure ou égale à 2, mais pour $h_g = 0$ et $h_g = 1$, heureusement, on a des résultats :

THÉORÈME 3.4.1 (SCHÜTZENBERGER) $h_g(L) = 0$ *ssi le monoïde syntaxique de L, M_L, a la propriété suivante :*

$$\forall x \in M_L, \exists n \in \mathbb{N}, x^{n+1} = x^n.$$

PREUVE : Laissée en exercice.... Remarquer que l'égalité signifie égal "modulo S_L". On dit alors que M_L est apériodique. ∎

PROPOSITION 3.4.1 *Il existe des langages de hauteur généralisée* 1.

PREUVE : Laissée en exercice.... On notera par ailleurs que $(aa)^*$ est un tel langage. ∎

3.5 Questions de complexité

DÉFINITION 3.5.1 (LANGAGE N-RATIONNEL)
Par **n-langage rationnel** on désigne un langage rationnel reconnu par un automate à moins de n états.

PROPOSITION 3.5.1 *Si on se donne deux langages L_1 (un m-langage rationnel) et L_2 (un n-langage rationnel), on a des résultats sur les opérations suivantes : (bornes atteintes)*
- *La complexité en états du produit $L_1.L_2$ est $m2^n - 2^{n-1}$.*
- *L_2^* : $2^{n-1} + 2^{n-1}$.*
- *Le miroir L_2^R : 2^n.*
- *L'intersection : $L_1 \cap L_2$: mn.*
- *L'union : $L_1 \cup L_2$: mn aussi.*

PREUVE : Dans le cas de l'étoile :

On considère l'automate $A = (Q, \Sigma, \delta, q_0, F)$, on pose $F_0 = F \setminus \{q_0\}$, et $k = |F_0|$. Montrons que $\forall k \geq 1$ et $N > 1$, il existe un automate fini reconnaissant $L(A)^*$ avec un nombre d'états inférieur ou égal à $2^{n-1} + 2^{n-k-1}$:

- **Existence** On considère $A' = (Q', \Sigma, \delta', q_0', F')$ avec $Q' = \{q_0\} \cup \{P \subseteq Q \setminus F_0, P \neq \emptyset\} \cup \{R \subseteq Q, q_0 \in R, R \cap F_0 \neq \emptyset\}$, $\delta'(q_0', a) = \{\delta(q_0, a)\}$ et $\delta'(X, a) = \begin{cases} \delta(X, a) & \text{si } \delta(X, a) \cup F_0 \neq \emptyset \\ \delta(X, a) \cup \{q_0\} & \text{sinon.} \end{cases}$. On vérifie que ça marche !
- **Borne atteinte** Pour $n = 2$ on considère $L = \{w \in a, b\}^* | \, |w|_a$ est impair.$\}$. Pour $n > 2$, on considère l'automate $A_n = (Q_n, \Sigma, \delta_n, 0, \{n-1\}$, avec $Q_n = [\![0, n-1]\!]$, et $\delta : \begin{cases} \delta(i, a) = i + 1 \text{ mod n} & 0 \leq i < n \\ \delta(i, b) = i + 1 \text{ mod n} & 1 \leq i < n \\ \delta(0, b) = 0 \end{cases}$.

■

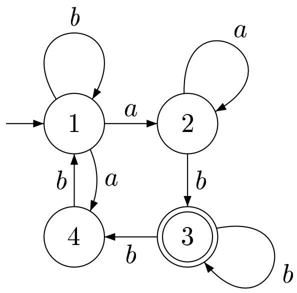
Quelques résultats :

- Le problème de l'indépendance déterministe (resp. non déterministe) est $LOGSPACE$ complet (resp $NLOGSPACE$ complet). C'est le problème $x \in L(A)$, où A automate. En passant par les ER, cela devient $NLOGSPACE$ complet.
- Le problème "est-ce que le langage de l'automate A est vide ou non ?" est $NLOGSPACE$ complet.
- Le problème de l'équivalence de deux automates déterministes est $NLOGSPACE$ complet, pour deus automates non déterministes cela devient $PSPACE$ complet.
- Le problème de l'équivalence de deux ER ($\lambda(\alpha) = \lambda(\beta)$) est aussi $PSPACE$ complet.
- Le problème de la minimisation d'un automate non déterministe est $PSPACE$ complet, ainsi que le passage d'un déterministe à un non déterministe minimal.

3.6 Questions et problèmes ouverts

DÉFINITION 3.6.1 (SYNCHRONISANT)
Soit $\mathcal{A}$ un AFD (Q, Σ, δ). Un mot w est dit **synchronisant** pour $\mathcal{A}$ si $\exists p \in Q$, $\forall q \in Q, \delta(q, w) = p$. L'automate $\mathcal{A}$ est alors dit *synchronisant.*

EXEMPLE 3.6.1 *Sur l'exemple suivant :*



Le mot $w = ba^3ba^3b$ est synchronisant.

CONJECTURE : Si $\mathcal{A}$ à n états synchronisant, il existe un mot synchronisant de longueur $\leq (n-1)^2$.

Chapitre 4

Grammaires et langages algébriques

Après la définition et l'étude des langages rationnels et des automates finis, on passe à présent à une classe de langages plus vaste, celle des langages algébriques. Ces langages se définissent à l'aide de grammaires que nous définissons ici, et nous verrons au chapitre suivant que ces langages sont aussi reconnus par une classe d'automates, appelés automates à pile.

4.1 Définitions

Définition 4.1.1 (Grammaire)
On appelle **grammaire** tout quadruplet $G = (\Sigma, N, P, S)$ où
- Σ est un alphabet, dit alphabet des *terminaux* ;
- N est un alphabet, dit des *non-terminaux* ;
- P est l'ensemble des productions, partie finie de $(\Sigma \cup N)^* \setminus \Sigma^* \times (\Sigma \cup N)^*$;
- S est un élément de N appelé *axiome* de G.

Les éléments de P sont les productions (où règles) de la grammaire, qui s'interprètent comme des règles de réécriture sur les mots de terminaux et de non-terminaux, c'est pourquoi un élément (α, β) de P se note plus souvent $\alpha \to \beta$. De même, lorsque l'on a plusieurs dérivations $S \to \alpha$, $S \to \beta$, $S \to \gamma$, on abrège souvent la notation en $S \to \alpha \mid \beta \mid \gamma$.

Définition 4.1.2 (Grammaires algébriques, contextuelles)
Une grammaire **algébrique** est une grammaire où les productions sont de type $A \to \alpha$, $\alpha \in (\Sigma \cup N)^*$, $A \in N$, une grammaire **contextuelle** ne contient que des productions du type $\beta \to alpha$, avec $|\beta| \leq |\alpha|$.

On déduit de le définition d'une grammaire algébrique une notion de dérivation sur $(\Sigma \cup N)^*$ qui correspond au fonctionnement de la grammaire. La relation de dérivation $\longmapsto$ se définit comme suit, pour μ et ν dans $(\Sigma \cup N)^*$:

$$\mu \longmapsto \nu \quad \Longleftrightarrow \quad \exists\, \theta, \theta', \alpha, \beta \in (\Sigma \cup N)^* \quad \begin{cases} \mu = \theta\alpha\theta' \\ \nu = \theta\beta\theta' \\ \alpha \overset{\beta}{\underset{\in}{\longmapsto}} P \end{cases}$$

La relation $\longmapsto$ est appelée **dérivation directe** ou simple, par opposition à sa clôture transitive $\overset{*}{\underset{G}{\longmapsto}}$. On note aussi quelquefois $\overset{k}{\underset{G}{\longmapsto}}$ pour noter k dérivations de suite.

Tous les langages ne sont pas algébriques :

EXEMPLE 4.1.1 *La grammaire suivante :*

$$G = \begin{cases} 1 & S \to aSBC \\ 2 & S \to aBC \\ 3 & CB \to BC \\ 4 & aB \to ab \\ 5 & bB \to bb \\ 6 & bC \to bc \\ 7 & cC \to cc \end{cases}$$

n'est pas algébrique notamment à cause de la règle 3. *On a :* $S \overset{2}{\underset{G}{\longmapsto}} aBC \overset{4}{\underset{G}{\longmapsto}} abC \overset{6}{\underset{G}{\longmapsto}} abc$, *en faisant* $1, 2, 4, 3, 5, 6, 7$ *on obtient* $a^2b^2c^2$, *on montre que cette grammaire reconnaît* $a^nb^nc^n$ *pour tout* $n \geq 0$.

4.2 Dérivation le plus à gauche

LEMME 4.2.1 (FONDAMENTAL) *Soit G une grammaire algébrique et une dérivation* $\alpha_1\alpha_2 \ldots \alpha_m \overset{n}{\underset{G}{\longmapsto}} \beta$ *(les* α_i *sont dans* $(\Sigma \cup N)^*$*). Alors il existe des mots* $\beta_1, \ldots \beta_m$, *des entiers* $n_1, \ldots n_m$, *tels que* $\beta = \beta_1 \ldots \beta_m$, $n = \sum n_i$ *et* $\alpha_i \overset{n_i}{\underset{G}{\longmapsto}} \beta_i$.

PREUVE : On montre ce résultat par récurrence sur n :

- Pour $n = 0$, $\beta = \alpha_1 \ldots \alpha_m$, prendre $\alpha_i = \beta_i$ et chaque $n_i = 0$.
- Supposons $\alpha_1 \ldots \alpha_m \overset{n}{\underset{G}{\longmapsto}} \beta$. Alors notons γ un mot tel que : $\alpha_1 \ldots \alpha_m \longmapsto \gamma \overset{n-1}{\underset{G}{\longmapsto}} \beta$. Soit $\alpha_i = \alpha_i' A \alpha_i''$. Alors $\alpha_1 \ldots \alpha_m \longmapsto \gamma = \alpha_1 \ldots \alpha_{i-1}\alpha_i'\alpha\alpha_i''\alpha_{i+1} \ldots \alpha_m$ grâce à $A \to \alpha \in P$. En prenant alors $\gamma_i = \alpha_i'\alpha\alpha_i''$, et pour $i \neq j$, $\gamma_j = \alpha_j$, on obtient $\gamma = \gamma_i \ldots \gamma_m$, donc on a la relation $\gamma_1 \ldots \gamma_m \overset{n-1}{\underset{G}{\longmapsto}} \beta$, ce qui permet d'appliquer l'hypothèse de récurrence et de trouver β_i, n_i qui seront convenables en raccrochant avec *pour* $j \neq i \alpha_j \overset{0}{\underset{G}{\longmapsto}} \gamma_j$ et pour $j = i$, $\alpha_i \overset{1}{\underset{G}{\longmapsto}} \gamma_i$. On obtient donc le bon résultat.

■

DÉFINITION 4.2.1 (DÉRIVATION LE PLUS À GAUCHE)
On définit la **dérivation le plus à gauche** par :

$$\mu \underset{g}{\longmapsto} \nu \quad \Longleftrightarrow \quad \exists u \in \Sigma^*, \exists\, \theta', \alpha, \beta \in (\Sigma \cup N)^* \quad \begin{cases} \mu = u\alpha\theta' \\ \nu = u\beta\theta' \\ \alpha \overset{\beta}{\underset{\in}{\longmapsto}} P \end{cases}$$

On définirait de façon similaire une relation de dérivation la plus à droite.

DÉFINITION 4.2.2 (LG)
On note $Lg(G)$ l'ensemble des mots de G obtenus à partir du start symbol par itération de dérivations le plus à gauche.

THÉORÈME 4.2.1

$$Lg(G) = L(G)$$

PREUVE :

- $Lg(G) \subseteq L(G)$ évident.
- $L(G) \subseteq Lg(G)$: On va prouver par récurrence sur la longueur de dérivation que $\forall u \in L(G)$, il existe une dérivation le plus à gauche.
  - Pour $n = 1$, ça marche car $S \overset{1}{\longmapsto} u$ est exactement $S \rightarrow u \in P$.
  - Pour hypothèse de récurrence, on prend "pour toute dérivation $A \rightarrow u$ (et non $S \rightarrow u\,!$), de longueur $\leq n$, il existe une dérivation le plus à gauche". Soit alors la dérivation $S \overset{n}{\longmapsto} u$, soit encore $S \overset{1}{\longmapsto} \gamma \overset{n-1}{\longmapsto} u$. Or $\gamma = \gamma_1 \ldots \gamma_m$, le lemme donne l'existence d'un mot $u = u_1 \ldots u_m$ et d'entiers n_i tels que $\forall i, \gamma_i \overset{n_i}{\longmapsto} u_i$. De plus, $S \rightarrow \gamma \in P$. Donc $\gamma_i \in (\Sigma \cup N)$ donc :
  - Si $\gamma_i \in \Sigma$: alors $u_i = \gamma_i$ et $n_i = 0$.
  - Si $\gamma_i \in N$: alors $\gamma_i \overset{n_i}{\longmapsto} u_i$ et on applique l'hypthèse de récurrence qui donne tout de suite la dérivation à gauche.

■

DÉFINITION 4.2.3 (AMBIGUITÉ D'UNE GRAMMAIRE ALGÉBRIQUE)
Une grammaire algébrique G est dite ambiguë lorsqu'il existe dans $L(G)$ un mot u pour lequel il existe deux suites disjointes de dérivation les plus à gauche menant de l'axiome de G à u.

EXEMPLE 4.2.1 *On considère la grammaire définie sur l'alphabet $\{a, +, \times\}$, avec pour seul non-terminal S, par les règles*

$$S \rightarrow S + S \mid S \times S \mid a$$

alors le mot $a+a\times a$ est obtenu par deux dérivations les plus à gauche différentes :

$$S \underset{g}{\longmapsto} S + S \underset{g}{\longmapsto} a + S \underset{g}{\longmapsto} a + S \times S \underset{g}{\longmapsto} a + a \times S \underset{g}{\longmapsto} a + a \times a$$

$$S \underset{g}{\longmapsto} S \times S \underset{g}{\longmapsto} S + S \times S \underset{g}{\longmapsto} a + S \times S \underset{g}{\longmapsto} a + a \times S \underset{g}{\longmapsto} a + a \times a$$

Il est naturel de chercher à avoir des grammaires non ambiguës autant que possible, car l'unicité de la dérivation de l'axiome en un mot peut être très utile pour certaines démonstrations. Malheureusement, Harrisson a montré le résultat suivant :

THÉORÈME 4.2.2 *L'ambiguité d'une grammaire est indécidable.*

PREUVE : On veut réduire à l'aide de la correspondance de Post. On rappelle le problème : on se donne deux mots A,B, $A = (w_1, \dots, w_m)$, $w_i \in \Sigma^*$ et $B = (x_1, \dots x_m)$, $x_i \in \Sigma^*$. On veut savoir si il existe une suite d'indices $i_1, \dots i_p$ telle que $(w_{i_1} \dots w_{i_p}) = (x_{i_1}, \dots x_{i_p})$. Ce problème est indécidable. On se donne une instance de ce problème, *i.e.* deux mots A et B de taille m, et aussi des mots a_i de Σ^* **distincts** des mots de A et de B.

On construit à partir des données les langages suivants :
- $L_A = \{w_{i_1}, \dots, w_{i_k} a_{i_k} \dots a_{i_j}, w_{i_j} \in \Sigma^*\}$.
- $L_B = \{x_{i_1}, \dots, x_{i_k} a_{i_k} \dots a_{i_j}, x_{i_j} \in \Sigma^*\}$.

On considère aussi la grammaire définie par :

$$\begin{cases} S \to S_A \mid S_B \\ S_A \to w_i S_A a_i \mid w_i a_i \quad \forall i \in [\![1, m]\!] \\ S_B \to x_i S_B a_i \mid x_i a_i \quad \forall i \in [\![1, m]\!] \end{cases}$$

Trivialement, elle génère le langage $L_A \cup L_B$. On va montrer que (A, B) est une instance positive du problème de Post ssi cette grammaire est ambiguë :

- $\Rightarrow$ Si c'est une instance positive du problème de Post, il existe une suite d'entiers $(i_1, \dots i_p)$, $w_{i_1} \dots w_{i_p} = x_{i_1} \dots x_{i_p}$. Maizalors

  $$w_{i_1} \dots w_{i_p} a_{i_p} \dots a_{i_1} = x_{i_1} \dots x_{i_p} a_{i_p} \dots a_{i_1}$$

  admet deux dérivations distinctes, l'une qui mène par S_A et l'autre par S_B.
- $\Leftarrow$ Réciproquement, si la grammaire est ambiguë, alors il existe deux dérivations distinctes de $u \in L_A \cup L_B$. Mais, compte tenu de la définition de la grammaire, on doit diverger sur le deuxième terme, ie

  $$S \longmapsto S_A \overset{*}{\longmapsto} w_{i_1} \dots w_{i_p} a_{i_p} \dots a_{i_1}$$

  et

  $$S \longmapsto S_B \overset{*}{\longmapsto} x_{i_1} \dots x_{i_p} a_{i_p} \dots a_{i_1}$$

  donc $w_{i_1} \dots w_{i_p} = x_{i_1} \dots x_{i_p}$ et c'est une instance positive de Post.

■

EXEMPLE 4.2.2 *$S \to SbS \mid abS \mid a$ est ambiguë, alors que* $\begin{cases} S \to Aa \\ A \to AbA \mid \varepsilon \end{cases}$ *ne l'est pas , mais ces 2 grammaires engendrent le même langage $(ab)^* a$.*

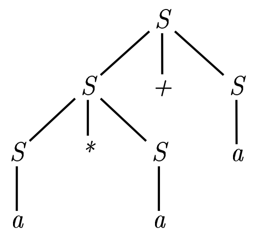
Cela dit il est possible moyennant certaines conditions d'assurer la non-ambiguïté des grammaires, en particulier en modifiant les grammaires tout en engendrant le même langage. Il n'est cependant pas systématiquement possible de rendre une grammaire non-ambiguë, en effet certains langages engendrés par des grammaires sont "intrinsèquement" ambigues, par exemple le langage : $\{a^i b^j c^k \mid i = j \text{ ou } j = k, i, j, k \geq 1\}$.

4.3 Arbres syntaxiques et arbres de dérivation

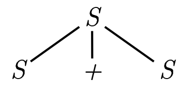
DÉFINITION 4.3.1 (ARBRE DE DÉRIVATION)
Soit G une grammaire algébrique. Les arbres considérés sont des arbres étiquetés et ordonnés :
- Les sommets sont étiquetés par $\Sigma \cup N \cup \{\varepsilon\}$,
- Un sommet étiqueté par ε ou par une lettre est nécessairement une feuille,
- Un sommet étiqueté par une variable A a pour fils $\alpha_1, \dots \alpha_p$, $\alpha_i \in \Sigma \cup N \cup \{\varepsilon\}$ ssi $A \to \alpha_1 \alpha_2 \dots \alpha_p \in P$, ce qui justifie le terme d'arbre "ordonné".

Un arbre dont la racine est S et dont le mot de feuille $\in \Sigma^*$ est appelé **arbre de dérivation** dans G.

EXEMPLE 4.3.1 *On considère la grammaire consituée de la seule production* $S \to S + S \mid S * S \mid a$ *:*
Un arbre de dérivation de $a * a + a$ *est :*



Bien voir que :



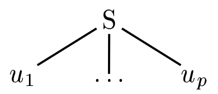
est aussi un arbre de dérivation.

DÉFINITION 4.3.2 (ARBRE SYNTAXIQUE)
C'est la même définition que précédemment sauf que la racine est un élément quelconque de S.

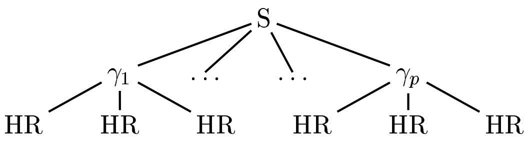
PROPOSITION 4.3.1 $u \in L(G) \Leftrightarrow$ *il est le mot de feuilles d'un arbre de dérivation de* G*.*

PREUVE :
- $\Rightarrow$ On raisonne par récurrence sur la longueur de la chaîne de dérivation. Si $u \in L(G)$, alors $S \overset{n}{\longmapsto} u$.
  - Si $n = 1$, alors $S \to u \in P$, alors si $u = \varepsilon$ on a l'arbre :

    et si $u = u_1 \dots u_p$, on réalise l'arbre :

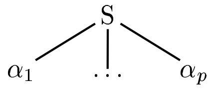


  - HR sur $A \to u$, A variable. Si $n > 1$, alors $S \overset{1}{\longmapsto} \gamma_1 \dots \gamma_p \overset{n-1}{\longmapsto} u$, $\gamma_i \in \Sigma \cup N$. Alors on récupère les sous-arbres construits par récurrence pour réaliser l'arbre :

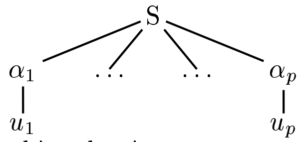


- $\Leftarrow$ On raisonne ici par récurrence sur la hauteur de l'arbre de dérivation dans G dont u est le mot de feuille.
  - si $h = 1$, soit on a l'arbre :

et alors $S \to \varepsilon \in P$; soit on a l'arbre :



auquel cas $S \to \alpha_1 \dots \alpha_p \in P$. On a alors dans les deux cas $S \vdash\!\!- u$.
  - Si $h > 1$, alors on a nécessairement :



...en appliquant l'hypothèse de récurrence.

■

REMARQUE : Un arbre de dérivation n'est pas nécessairement unique

4.4 Grammaires algébriques propres

DÉFINITION 4.4.1 (TRANSFORMATION EN GRAMMAIRE PROPRE) On sait faire en $O(n)$ où (n est le nombre de productions de la grammaire considérée) en sorte que si $L = L(G)$ contient ε, alors ε n'apprait que dans $S \to \varepsilon$, il n'y a pas de production contenant S à droite, et il n'y a pas de production simple (*i.e.* du type $A \to B$). Une telle grammaire est dite **propre**.

LEMME 4.4.1 (ETOILE) *Si L est un langage algébrique, il existe un entier N, tel que quel que soit u mot de L de taille $\geq N$, u se décompose en $u = xvywz$ avec :*
- $|vw| \leq 1$,
- $|vyw| \leq N$,
- $\forall i \geq 0, xv^iyw^iz \in L$.

Pour la preuve, on commence par montrer le :

LEMME 4.4.2 *Si dans un arbre syntaxique aucun chemin n'est de longueur $> i$, alors le mot de feuille est de longueur $\leq m^i$, où m est la longueur maximale d'un second membre de règle de la grammaire considérée.*

PREUVE : La preuve se fait facilement par récurrence sur i en dessinant les arbres correspondant. ■

Voici maintenant la preuve du lemme de l'étoile :
PREUVE : Si on prend $N = m^k + 1$ où k est le nombre de variables de la grammaire. Alors si u de taille $\geq N > m^k$, il existe dans un arbre syntaxique de mot de feuille u un chemin sur lequel deux nœuds sont étiquetés par une même variable A. Alors on a $S \overset{*}{\longmapsto} xAz$, $A \overset{*}{\longmapsto} vAw$ et $A \longmapsto y$. Il en résulte immédiatement la propriété, la première partie venant du fait que l'on a considéré une grammaire propre, la deuxième provenant du fait que l'on peut choisir les deux occurrences de la variable A comme on veut (si les tailles ne conviennent pas, on recommence). ■

EXERCICE 4.4.1 *Montrez que $a^n b^n c^n$ n'est pas algébrique.*

Une conséquence est que la classe des langages algébriques n'est close ni par intersection, ni par complément. Par contre, elle est stable par union, produit, étoile (le montrer).

4.5 Formes normales

4.5.1 Forme normale de Chomski

DÉFINITION 4.5.1 (FNC)
Une grammaire est sous **FNC** si ses productions sont de la forme : $A \to BC$ (A, B, C variables) et $A \to a$, a in Σ. Si $\varepsilon \in L(G)$, on ajoute $S \to \varepsilon$ et nin S, ni ε n'apparaissent dans les autres membres droits de dérivation.

PROPOSITION 4.5.1 *Toute grammaire algébrique G est équivalence à une grammaire algébrique sous FNC. Et la tranformation est effective.*

PREUVE : Soit $G = (\Sigma, N, P, S)$ la grammaire qu'il s'agit de transformer en $G' = (\Sigma, N', P', S)$ grammaire équivalente sous FNC. IL s'agit de transformer les productions de la forme :
- $A \to aB$, $a \in \Sigma, A, B \in N$,
- $A \to ab$,
- $A \to Ba$,
- $A \to \alpha$ avec $|\alpha| \geq 3$.

C'est parti :
- Si on a $A \to aB$, pour $a \in \Sigma$, on introduit une nouvelle variable a' et des nouvelles productions pour remplacer : $A \to a'B$ et $a' \to a$ qui ont le bon format.
- Si on a $A \to \alpha_1 \dots \alpha_p$ avec $\alpha_i \in \Sigma \cup N$ et $p \geq 3$, on réalise les opérations suivantes :
  - Si $\alpha_i \in \Sigma$,on introduit la nouvelle variable α'_i et la production $\alpha'_i \to \alpha_i$,
  - Si $\alpha_i \in N$, on conserve cette variable : $\alpha'_i = \alpha_i$.

  Et on ajoute les productions : ($<$ et $>$ sont des nouvelles variables)

$$\begin{cases} A \to \alpha'_1 < \alpha_2 \dots \alpha_p > \\ < \alpha_2 \dots \alpha_p > \to \alpha'_2 < \alpha_3 \dots \alpha_p > \\ \dots \\ < \alpha_{p-1}\alpha_p > \to \alpha'_{p-1}\alpha'_p \end{cases}$$

On vérifie que la grammaire ainsi obtenue est équivalente par récurrence sur la longueur de dérivation.

■

EXEMPLE 4.5.1 *Considérons la grammaire d'axiome E définie par*

$$\begin{cases} E \to E * T \mid T * F \mid (E) \mid a \\ T \to T * F \mid (E) \mid a \\ F \to (E) \mid a \end{cases}$$

On transforme la première règle en $\begin{cases} E \to E <* T> \\ <* T> \to *' \ T \\ *' \to * \end{cases}$ *etc.*

Cette FNC est très utile car ensuite on peut tester l'appartenance au langage en $O(|w|^3)$: c'est l'algorithme de Cocke, Youger et Kasami :
On suppose que $\varepsilon \notin L$. Soit G une rammaire sous FNC, $G = (\Sigma, N, P, D)$. Soit $w \in \Sigma^*, |w| = n$ et notons $w_{i,j}$ le facteur de longueur j, à partir de la position i de w. Alors :

- Si $j = 1$,$w_{i,j}$ est une lettre, dire que $A \overset{*}{\longmapsto} w_{i,j}$ dans la grammaire signifie que $A \to w_{i,j}$ est une production de la grammaire.
- Si $j \neq 1$, $A \overset{*}{\longmapsto} w_{i,j}$ signifie que cette dérivation est forcément de longueur $m > 1$. Alors on écrit $A \overset{1}{\longmapsto} BC \overset{m-1}{\longmapsto}$. En utilisant le lemme fondamental, il existe $k \in \mathbb{N}$, $w_{i,j} = w_{i,k} w_{j_k+1,j}$, et donc $B \longmapsto w_{i,k}$ et $C \longmapsto w_{j-k+1,j}$.

On va être amené à considérer les ensembles $V_{i,j}$ définis par l'ensemble des variables A, $A \overset{*}{\longmapsto} w_{i,j}$. Alors $V(1,n) = \{A, A \overset{*}{\longmapsto} w\}$.

D'où l'algorithme pour construire ces ensembles :

1. Pour $i := 1$ à n faire
2. $V_{i,1} := \{A \mid A \to a \in P$ et a i-ème caractère de w$\}$
3. Pour $j := 2$ à n faire
4.   Pour $i := 1$ à $n - j + 1$ faire
5. $V_{i,j} := \emptyset$
6.   Pour $k := 1$ à $j - 1$ faire
7. $V_{i,j} := V_{i,j} \cup \{A \mid A \to BC \in P, B \in V_{i,k}$ et $C \in V_{i+k,j-k}\}$

REMARQUE : Tester l'appartenance à un langage est bien en $O(|w|^*)$, si on considère que la dernière ligne est en temps constant, car $|G|$ étant une constante, la complexité de la dernière ligne ne dépend pas de la taille du mot.

4.5.2 Forme Normale de Greibach

DÉFINITION 4.5.2 (FNG)
Une grammaire algébrique est sous **FNG** lorsque ses productions sont du type $A \to a\alpha$, $\alpha \in N^*, a \in \Sigma$.

On suppose que le langage associé ne contient pas ε dans la suite. Sinon, on fait la même bidouille que d'habitude.

LEMME 4.5.1 *Soit G une grammaire qui contient des productions :* $\begin{cases} A \to \alpha_1 B \alpha_2 \\ B \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_p \end{cases}$ *alors on obtient une grammaire équivalente en remplaçant ces productions par* $A \to \alpha_1\beta_1\alpha_2 \mid \ldots \alpha_1\beta_p\alpha_2$.

PREUVE : Evidente. ■

LEMME 4.5.2 *Soit G une grammaire avec des productions* $A \to A\alpha_1 \mid \ldots \mid A\alpha_n$ *et* $A \to \beta_1 \mid \ldots \beta_p$. *Alors il existe une grammaire algébrique* G' *équivalente dans laquelle ces productions sont remplacées par :* $A \to \beta_1 Z \mid \beta_2 Z \mid \ldots \mid \beta_p Z \mid \beta_1 \mid \ldots \mid \beta_p$ *et* $Z \to \alpha_1 Z \mid \ldots \alpha_n Z \mid \alpha_1 \mid \ldots \mid \alpha_n$, *où* Z *est une nouvelle variable.*

PREUVE :

- $\Rightarrow$ Si dans la première grammaire on peut réaliser la dérivation

$$A \longmapsto A\alpha_{i_1} \underset{\alpha}{\overset{A}{\longmapsto}}_{i_2} \alpha_{i_1} \ldots A\alpha_{i_k} \ldots \alpha_{i_1} \longmapsto \beta_j \alpha_{i_k} \ldots \alpha_{i_1}$$

  , dans la deuxième on peut réaliser :

$$A \longmapsto \beta_j Z \longmapsto \beta_j \alpha_{i_k} Z \longmapsto \ldots \longmapsto \beta_j \alpha_{i_k} \ldots \alpha_{i_2} \longmapsto \beta_j \alpha_{i_k} \ldots \alpha_{i_1}.$$

- $\Leftarrow$ Démo identique.

■

On obtient à l'aide des deux lemmes précédents la :

PROPOSITION 4.5.2 *Soit G une grammaire algébrique. Il existe un algorithme qui permet de la mettre sous FNG.*

PREUVE : On part d'une grammaire sous FNC. Les productions sont sous la forme $A \to BC$ et $A \to a$. On énumère les variables $N = \{A_1 \ldots A_m$. On commence par transformer les productions du type $A_i \to A_j\alpha$ avec $j \le i$ pour ne plus garder que les productions du type $A_i \to A_j\alpha$, $(\alpha \in N)$ avec $j > i$ (et les autres). On le fait incrémentalement à partie de $i = 1$. Comment ? On suppose qu'une telle opération a été réalisée pour tout $1 \le i \le k$ et que l'on veut tranformer $A_{k+1} \to A_j\alpha$ avec $j \le k+1$:

- si $j \le k$, on peut "substituer" à A_j une production $A_j \to A_l\beta$ avec $l > j$: $A_{k+1} \to A_l\alpha\beta$. Si $l > k+1$ c'est gagné. Sinon, $l = k+1$ et on utilise le lemme 4.5.2 pour éliminer la récursivité à gauche en introduisant une nouvelle variable.
- si $j = k+1$ On utilise directement le lemme 4.5.2 pour éliminer la récursivité à gauche en introduisant une nouvelle variable.

A la fin de cette phase, les règles sont du type $A_i \to A_j\alpha$ $(j > i)$, $A_i \to a\alpha$, $A_i \to A$ et $Z \to \alpha$ $(\alpha \in (N \cup \bigcup_i \{Z_i\})^+)$. On utilise ensuite le lemme 4.5.1 pour éliminer les règles qui ne conviennent pas. ■

EXERCICE 4.5.1 *Transformer la grammaire suivante afin de la mettre sous FNG :*

$$G = \begin{cases} A_1 \to A_2 A_3 \\ A_2 \to A_1 A_2 \mid 1 \\ A_3 \to A_1 A_3 \mid 0 \end{cases}$$

Chapitre 5

Langages algébriques et automates à pile

Comme pour les langages rationnels, les langages algébriques sont caractérisés par une classe de machines, les automates à pile, qui possèdent “plus de mémoire” que leurs cousins les automates finis, car elles possèdent une pile.

5.1 Automates à pile

Définition 5.1.1 (Automate à pile)
Un automate à pile est un septuplet $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, \gamma_0, F)$ où

- Q est un ensemble d'états,
- Σ est un alphabet fini de terminaux,
- Γ est un alphabet dit **alphabet de pile**,
- $\delta : Q \times \Sigma \cup \{\varepsilon\} \times \Gamma \to \mathcal{P}_{finies}(Q, \Gamma^*)$, $(q, a, z) \mapsto \{(p_1, \gamma_1), \ldots (p_m, \gamma_m\}$ (NON DET) : z correspond au caractère de haut de pile, et chaque γ_i est le mot de pile substitué au caractère z sur la pile. Cette fonction est appelée **fonction de transition de l'automate à pile**,
- $q_0 \in \Sigma$ est l'**état initial** de l'automate,
- $\gamma_0 \in \Gamma$ est le **caractère de fonds de pile**,
- $f \subset \mathcal{P}(Q)$ est l'ensemble des **états d'acceptation**.

Remarque : Le fonctionnement est le suivant : on dispose de 2 têtes de lecture, une sur un ruban de pure lecture où est disposé le mot d'entrée, et un ruban de lecture/ecriture, qui est la pile, où on ne peut lire que le caractère de haut de pile. Pour effectuer une transition à partir d'un certain état q, on peut lire une lettre de l'entrée ou ε (!) et effectuer une transition autorisée par la fonction δ.

Bien remarquer que lorsque la pile est vide, on ne peut plus rien faire.

Formalisons un peu tout ça :

Définition 5.1.2
Une **configuration** d'un tel automate est un triplet (q, u, γ) où q est l'état, u est le mot qui reste à lire sur l'entrée, γ le mot courant de la pile.

DÉFINITION 5.1.3 (DÉRIVATION SIMPLE)
$(q, au, z\gamma) \underset{p}{\overset{(}{\longmapsto}} , u, \gamma, \gamma')$ pour $a \in \Sigma \cup \{eps\}$ ssi $(q, a, z) \in Dom(\delta)$ et $(p, \gamma') \in \delta(q, a, z)$.

La dérivation notée $\overset{*}{\longmapsto}$ est la clôture transitive reflexive de la dérivation simple.

DÉFINITION 5.1.4 (MODES DE RECONNAISSANCE D'UN AP)
Il y a deux modes de reconnaissance différents pour un automate à pile :

- **par pile vide** $L_P(\mathcal{M}) = \{u \in \Sigma^* \mid (q_0, u, \gamma_0) \overset{*}{\longmapsto} (q, \varepsilon, \varepsilon, q \in Q\}$
- **par état final** lorsque l'on arrive dans un état $q \in F$.

REMARQUE : En général, on considère l'un des deux modes de fonctionnement, en précisant lequel. En général, étant donné un automate à pile $\mathcal{M}$, $L_P(\mathcal{M}) \neq L_F(\mathcal{M})$. Mais heureusement on a le résultat suivant :

THÉORÈME 5.1.1 *Si L est un langage reconnu par pile vide, il existe un AP qui le reconnaît par état final. Si L est un langage reconnu par état final, il existe un automate à pile qui le reconnaît par pile vide.*

PREUVE :

- Pile vide $\Rightarrow$ état final : Soit $L = L_P(\mathcal{M})$ avec $M = (Q, \Sigma, \Gamma, \delta, q_0, \gamma_0)$. On cherche $M' = (Q, \Sigma', \Gamma', \delta', q_0', \gamma_0', F')$ tel que $L = L_F(\mathcal{M}')$:
  - $Q' = Q \cup \{q_0', q_f\}$ (nouveaux états).
  - $F' = \{q_f\}$
  - $\Gamma' = \Gamma \cup \gamma_0'$ (nouveau)
  - On cherche à définir δ' pour avoir $(q'0, u, \gamma_0') \underset{\mathcal{M}'}{\overset{*}{\longmapsto}} (q', \varepsilon, q_f)$. On veut donc $(q_0', u, \gamma_0') \underset{\mathcal{M}'}{\longmapsto} (q_0, u, \gamma_0\Gamma_0') \underset{\mathcal{M}'}{\overset{*}{\longmapsto}} (q, \varepsilon, \gamma_0') \underset{\mathcal{M}'}{\longmapsto} (q_f, \varepsilon\gamma_0')$. On évite ainsi d'être bloqué à pile vide, et on force l'évolution vers un état final. L'évolution du milieu peut être réalisée à l'aide des transitions de $\mathcal{M}$: $\delta'(q, a, z) = \{\delta(q, a, z), q \in Q, z \in \Gamma\}$. On rajoute : $\delta'(q_0', \varepsilon, \gamma_0) = \{(q_0, \gamma_0\gamma_0')\}$ et aussi $\delta'(q, \varepsilon, \gamma_0') = \{(q_f, \gamma_0')\}$.

  On a trivialement l'inclusion $L_P(\mathcal{M}) \subseteq L_F(\mathcal{M}')$ (c'est fait pour !). Il reste à montrer $L_F(\mathcal{M}') \subseteq L_P(\mathcal{M})$. Si $(q_0', u, \gamma_0') \underset{\mathcal{M}'}{\overset{*}{\longmapsto}} (q_f, \varepsilon, \gamma)$, alors $(q_0', u\gamma_0') \longmapsto (q_0, u, \gamma_0\gamma_0') \overset{*}{\longmapsto} (q_f, \varepsilon, \gamma_0')$ et voilà !
- Etat final $\Rightarrow$ pile vide : on fait à peu-près la même chose, dans l'autre sens.

■

5.2 Egalité de la classe des AP et la classe des langages algébriques

On commence par le sens "facile" :

PROPOSITION 5.2.1 *Tout langage algébrique est reconnu par un automate à pile.*

PREUVE : Soit G une grammaire qui engendre L algébrique considéré. $G = (\Sigma, N, P, S)$. On cherche $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, \gamma_0)$ qui reconnaît L par pile vide, *i.e.* pour $u \in L$, $(q_0, u, \gamma) \overset{*}{\longmapsto} (q, \varepsilon, \varepsilon)$. On prend $\Gamma = N \cup \Sigma$ pour alphabet de pile, $Q = \{q\}$ (un seul état), $\gamma_0 = S$, $q_0 = q$ et $\delta : (q, \varepsilon, A) \mapsto \{(q, \alpha \mid A \to \alpha \in P\}$ et $(q, a, a) \mapsto \{(q, \varepsilon)\}$ ("vidange"). Il reste à montrer $L_P(\mathcal{M}) = L(G)$:

- $L(G) \subseteq L_P(\mathcal{M})$: Soit $u \in L(G)$. Montrons que $S \underset{G}{\overset{n}{\longmapsto}} u \Rightarrow (q_0, u, S) \underset{\mathcal{M}}{\overset{*}{\longmapsto}} (q, \varepsilon, \varepsilon)$ par récurrence sur n (en remplaçant S par A non terminal dans l'HR) :
  - si $n = 1$: cela signifie $S \to u \in P$ et alors $(q, u, S) \underset{\mathcal{M}}{\longmapsto} (q, u, u) \underset{\mathcal{M}}{\overset{*}{\longmapsto}} (q, \varepsilon, \varepsilon)$.
  - si n quelconque : on applique encore le lemme fondamental pour récupérer l'hytpothèse de récurrence.
- $L_P(\mathcal{M}) \subseteq L(G)$: on montre facilement par récurrence sur n : $\forall A \in N, (q, u, A) \underset{\mathcal{M}}{\overset{n}{\longmapsto}} (q, \varepsilon, \varepsilon) \Rightarrow A \underset{G}{\overset{*}{\longmapsto}} u$.

■

THÉORÈME 5.2.1 *Tout langage reconnu par pile vide par un automate à pile est algébrique.*

PREUVE : Supposons que $L = L_P(\mathcal{M})$ où $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, \gamma_0)$ est un automate à pile. On cherche G algébrique tel que $L = L(G)$. On va créer des variables et des productions à l'aide des caractéristiques de l'automate $\mathcal{M}$:

- Si $(p, \gamma_1\gamma_2 \dots \gamma_n) \in \delta(q, a, z)$ avec $\gamma_i, z \in \Gamma$, $p, q \in Q$, et $a \in \Sigma \cup \{\varepsilon\}$ alors on crée les variables et productions suivantes (pour chaque $p_i \in Q, q' \in Q$) :

$$[qzq'] \to a[p\gamma_1 p_1][p_1\gamma_2 p_2] \dots [p_{m-1}\gamma_m p_m]$$

- Si $(p, \varepsilon) \in \delta(q, a, z)$ alors on crée $[qzp] \to a$.
- On ajoute pour chaque $q' \in Q$, $S \to [q_0\gamma_0 q']$. S est une nouvelle variable qui sera l'axiome de G.

Alors montrons que $(q, u, z) \underset{\mathcal{M}}{\overset{*}{\longmapsto}} (p, \varepsilon, \varepsilon) \Leftrightarrow [qzp] \underset{G}{\overset{*}{\longmapsto}} u$ et on aura bien le résultat voulu :

- $\Rightarrow$: Montrons par récurrence sur n que $(q, u, z) \underset{\mathcal{M}}{\overset{n}{\longmapsto}} (p, \varepsilon, \varepsilon) \Rightarrow [qzp] \underset{G}{\overset{*}{\longmapsto}} u$.
  - Si $n = 1$: cela signifie que $(p, \varepsilon) \in \delta(q, u, z)$ avec $u = \varepsilon$ ou $u = a$ une lettre. Donc d'après la grammaire $[qzp] \to \in P$ donc $[qzp] \underset{G}{\overset{*}{\longmapsto}} u$.
  - Soit n qq. Supposons la propriété vraie pour tout $k < n$. Pour n, la dérivation d'hypothèse se décompose en :

$$(q, u, z) \underset{\mathcal{M}}{\overset{1}{\longmapsto}} (q', u', \gamma') \underset{\mathcal{M}}{\overset{n-1}{\longmapsto}} (p, \varepsilon, \varepsilon)$$

avec $a \in \Sigma \cup \{\varepsilon\}$, $(q', \gamma') \in \delta(q, u, z)$ et $\gamma' \in \Gamma^+$, soit $\gamma' = \gamma_1 \dots \gamma_m$ avec $\gamma_i \in \Gamma$. Donc on a $(q', u', \gamma_1 \dots \gamma_m) \underset{\mathcal{M}}{\overset{*}{\longmapsto}} (p, \varepsilon, \varepsilon)$. Cela implique qu'il existe $u'_1, \dots u'_m$ une décomposition de u' telle que

$$(q', u', \gamma_1 \dots \gamma_m) \underset{\mathcal{M}}{\overset{n_1}{\longmapsto}} (q'_1, u', \gamma_2 \dots \gamma_m)$$

et ceci pour la première fois. De même on a l'existence des n_i suivants : $(q', u'_1, \gamma_1) \underset{\mathcal{M}}{\overset{n_1}{\vdash}} (q'_1, \varepsilon, \varepsilon)$, $(q'_1, u'_2, \gamma_2) \underset{\mathcal{M}}{\overset{n_2}{\vdash}} (q'_2, \varepsilon, \varepsilon)$ et ainsi de suite jusqu'à $(q'_{m-1}, u'_m, \gamma_m) \underset{\mathcal{M}}{\overset{n_m}{\vdash}} (q'_m, \varepsilon, \varepsilon)$. Cela signifie d'après l'hypothèse de récurrence que $\forall i, [q'_{i-1}\gamma_i q'_i] \underset{G}{\overset{*}{\vdash}} u'_i$. Donc

$$[q'\gamma_1 q'_1] \underset{G}{\vdash} \ldots \underset{G}{\vdash} [q'_{m-1}\gamma_m q_m] \underset{G}{\overset{*}{\vdash}} u'_1 \ldots u'_m = u.$$

C'est ce qu'on voulait !

- $\Leftarrow$: Montrons par récurrence sur n que $[qzp] \underset{G}{\overset{n}{\vdash}} u \Rightarrow (q, u, z) \underset{\mathcal{M}}{\overset{*}{\vdash}} (p, \varepsilon, \varepsilon)$.
  - Si $n = 1$: $[qzp] \underset{G}{\overset{1}{\vdash}} u$ donc $[qzp] \to u \in P$. Alors $(p, \varepsilon) \in \delta(q, u, z)$ et donc $(q, u, z) \underset{\mathcal{M}}{\overset{*}{\vdash}} (p, \varepsilon, \varepsilon)$.
  - Soit n quelconque. Supposons la propriété vraie pour tout $k < n$. On décompose encore la dérivation :

$$[qzp] \underset{G}{\overset{1}{\vdash}} \alpha \underset{G}{\overset{n-1}{\vdash}} u$$

Avec $[qzp] \to \alpha \in P$, donc $[qzp] \to a[q'\gamma_1 q_1][q_1\gamma_2 q_2] \ldots [q_{m-1}\gamma_m q_m]$. Ce qui entraîne que $u = au'$ avec $u' = u'_1 \ldots u'_m$ et d'après le lemme fondamental $[q'\gamma_1 q_1] \underset{G}{\overset{n_1}{\vdash}} u'_1, \ldots, [q_i\gamma_{i+1} q_{i+1}] \underset{G}{\overset{n_i}{\vdash}} u'_i$. Mais au premier pas de dérivation dans G correspond un pas de dérivation dans $\mathcal{M}$. On a donc bien obtenu une dérivation dans $\mathcal{M}$ au final.

■

5.3 Automates à pile déterministes

DÉFINITION 5.3.1
Un automate à pile déterministe est un automate à pile $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, \gamma_0, F)$ où en plus on a :

1. $|\delta(q, a, z)| \leq 1$.
2. Si $|\delta(q, \varepsilon, z) = 1|$ alors $\forall a \in \Sigma, |\delta(q, a, z)| = 0$.
3. Si pour un $b \in \Sigma$, on a $|\delta(q, b, z)| = 1$, alors $|\delta(q, \varepsilon, z)| = 0$.

En langage courant, on ne lit ε que si on ne peut lire une lettre.

DÉFINITION 5.3.2
Un langage L possède **la propriété du suffixe** si il vérifie :

$$uv \in L \text{ et } v \neq \varepsilon \Rightarrow u \notin L.$$

PROPOSITION 5.3.1 *Un langage L est reconnu par automate à pile déterministe par pile vide si et seulement si il est reconnu par APD par état final et il a la propriété du suffixe.*

PREUVE : La preuve est similaire à l'équivalence montrée pour les AP non déterministes, sauf que pour une implication, on utilise la propriété du suffixe.

■

EXEMPLE 5.3.1 *$\{a^n b^n\}$ est reconnu par APD par pile vide.*

REMARQUE : Soit $\Sigma = \{a_1 \ldots, a_m\}$ un alphabet et $\overline{\Sigma} = \{\overline{a_1}, \ldots, \overline{a_m}\}$. Considérons la grammaire constituée des productions $S \to S\ a_i\ S\ \overline{a_i}\ S \mid \varepsilon$. Alors $L(G)$ est reconnu par état final par APD et pas par pile vide.

EXERCICE 5.3.1 *Montrez que $\{a^n b^n\} \cup \{a^n b^{2n}\}$ est algébrique, mais il n'est reconnu par aucun APD. Indication : il faut montrer que sur la pile, les mouvements sont nécessairement bornés.*

On a donc :

$$APD_{\text{pile vide}} \subsetneq APD_{\text{final}} \subsetneq AP$$

REMARQUE : Ce sont les langages "du milieu" qui sont en général appelés **Langages algébriques déterministes**.

Chapitre 6

Grammaires LL et LR

Introduction

Il existe deux types d'analyse [1] :

- **L'analyse descendante** : à partir de l'axiome, on construit une dérivation du mot à analyser. Pour l'analse LL, ces dérivations s'effectuent "le plus à gauche".
- **L'analyse ascendante** : on construit une dérivation qui commence par l'axiome (en partant de la fin) . Pour l'anlyse LR ce sont des dérivations "le plus à droite".

REMARQUE : La lecture du mot se fait **toujours** de gauche à droite. D'où la signification du L de LR et LL.

Intuitivement, on note $L?(k)$ une grammaire si le test d'appartenance peut être réalisé avec k "look-ahead". C'est l'idée intuitive du formalisme qui va suivre.

6.1 Grammaires $LL(k)$

DÉFINITION 6.1.1

Si $|w| > k$, $k : w$ désigne le préfixe de longueur k du mot w, sinon $k : w$ désigne w lui-même.

DÉFINITION 6.1.2

Une grammaire algébrique $G = (\Sigma, N, P, S)$ est dite $LL(k)$, $k \geq 0$ lorsque la condition suivante est vérifiée :

$$\begin{cases} (1) & S \overset{*}{\underset{g}{\longmapsto}} uA\theta \longmapsto u\alpha\theta \overset{*}{\underset{g}{\longmapsto}} uw \\ (2) & S \overset{*}{\underset{g}{\longmapsto}} uA\theta \longmapsto u\beta\theta \overset{*}{\underset{g}{\longmapsto}} uw' \Rightarrow \alpha = \beta \\ (3) & k : w = k : w' \end{cases}$$

REMARQUE : Une telle grammaire n'est pas ambiguë.

EXEMPLE 6.1.1

[1] c'est à dire deux manières de savoir si $u \in L(G)$

- $\begin{cases} S \to aAb \mid b \\ A \to a \mid bSA \end{cases}$ *est* $LL(1)$.
- $\begin{cases} S \to \varepsilon \mid abA \\ A \to Saa \mid b \end{cases}$ *est* $LL(2)$.

REMARQUE : Un langage $LL(0)$ est soit vide soit réduit à une seule lettre.

EXERCICE 6.1.1 *Montrez que les langages rationnels sont* $LL(1)$

REMARQUE : Il existe des grammaires algébriques qui ne sont $LL(k)$ pour aucun k, par exemple :

– La grammaire : $\begin{cases} S \to A \mid B \\ A \to aAb \mid 0 \\ B \to aBbb \mid 1 \end{cases}$ qui décrit le langage $\{a^n0b^n\} \cup \{a^n1b^{2n}\}$

– La grammaire : $S \to b \mid S\ a$ qui décrit le langage $\{ba^n, n \geq 1\}$

6.2 Les fonctions $Premier_k$, $Suivant_k$

DÉFINITION 6.2.1

Soit $G = (\Sigma, N, P, S)$ une grammaire. Si $\alpha \in (\Sigma \cup N)^*$, on définit pour tout $k \geq 0$ les ensembles suivants :

- $Premier_k(\alpha) = \{k : w \mid \alpha \overset{*}{\underset{G}{\vdash}} w\}$
- $Suivant_k(\alpha) = \{w \in \Sigma^* \mid \exists \beta, \gamma, S \longmapsto \beta\alpha\gamma$ et $w \in Premier_k(\gamma)\}$

DÉFINITION 6.2.2

Si L est un langage, on note :

$Premier_k(L) = \bigcup_{\alpha \in L} Premier_k(\alpha)$ et $Suivant_k(L) = \bigcup_{\alpha \in L} Suivant_k(\alpha)$.

Nous allons maintenant énoncer différentes caractérisations des grammaires $LL(k)$:

PROPOSITION 6.2.1 *Une grammaire algébrique est* $LL(k)$ *ssi pour toutes productions distinctes* $A \to \alpha$ *et* $A \to \beta$*, l'intersection des ensembles* $Premier_k(\alpha\gamma)$ *et* $Premier_k(\beta\gamma)$ *est vide, pour tout* γ *tel que* $S \overset{*}{\underset{g}{\vdash}} uA\gamma$ *existe.*

PREUVE :

- : $\Leftarrow$ Si G n'est pas $LL(k)$, d'après la définition, il existe $\alpha \neq \beta$, ...Alors $k : w$ appartient aux deux ensembles $Premier_k(\alpha\theta)$ et $Premier_k(\beta\theta)$ et donc l'intersection est non vide.
- : $\Rightarrow$ Si la condition de droite n'est pas satisfaite, alors il existe un mot u dans l'intersection et à l'aide de ce mot on obtient L qui n'est pas $LL(k)$.

■

COROLLAIRE 6.2.1 *Les grammaires* $LL(k)$ *ne sont pas récursives à gauche.*

PREUVE : Supposons que G soit $LL(k)$ pour un certain k et récursive à gauche. Alors $\exists A \to A\alpha$ et $A \to \beta$ dans P (on suppose toujours qu'aucun symbole n'est inutile). Alors considérons les deux dérivations $S \overset{*}{\underset{g}{\longmapsto}} uA\theta \overset{+}{\underset{g}{\longmapsto}} uA\alpha^n\theta \longmapsto uA\alpha^{n+1}\theta$ et $S \overset{*}{\underset{g}{\longmapsto}} uA\theta \overset{+}{\underset{g}{\longmapsto}} uA\alpha^n\theta \longmapsto u\beta\alpha^n\theta$. Comme G est $LL(k)$, les ensembles $Premier_k(A\alpha^{n+1}\theta)$ et $Premier_k(\beta\alpha^n\theta)$ sont d'intersection nulle. Comme $A \to \beta \in P$, $Premier_k(\beta\alpha^{n+1}\theta) \subseteq Premier_k(A\alpha^{n+1}\theta)$. Donc :

$$Premier_k(\beta\alpha^{n+1}\theta) \cap Premier_k(\beta\alpha^n\theta) = \emptyset.$$

Mais alors, si $\alpha \longmapsto \varepsilon$ et on a une contraction avec l'égalité précédente. Sinon, on considère $n > k$ pour avoir encore une contradiction.

■

Et encore une caractérisation, mais cette fois pour $k = 1$:

PROPOSITION 6.2.2 *Une grammaire est $LL(1)$ ssi pour toutes productions distinctes $A \to \alpha$ et $A \to \beta$, on a :*

$$Premier_1\big(\alpha Suivant_1(A)\big) \cap Premier_1\big(\beta Suivant_1(A)\big) = \emptyset.$$

PREUVE : On utilise le fait que :

$$Premier_1(\alpha Suivant_1(A)) = \begin{cases} Premier_1(\alpha) \setminus \{\varepsilon\} \cup Suivant_1(A) & \text{si } \alpha \overset{*}{\underset{g}{\longmapsto}} \varepsilon \\ Premier_1(\alpha) & \text{sinon.} \end{cases}.$$

- $\Rightarrow$: Soit I l'intersection des deux ensembles. Si I est non vide, alors il existe $a \in I$ et alors :
  - Si α et β ne dérivent pas ε, alors a appartient à l'intersection des ensembles $Premier_1(\alpha)$ et $Premier_1(\beta)$. Donc $\alpha \overset{*}{\underset{g}{\longmapsto}} aw$ et $\beta \overset{*}{\underset{g}{\longmapsto}} aw'$.

    Et on peut exhiber deux dérivations à gauche : $S \overset{*}{\underset{g}{\longmapsto}} uA\theta \longmapsto u\alpha\theta \overset{*}{\underset{g}{\longmapsto}} uawt$ et $S \overset{*}{\underset{g}{\longmapsto}} uA\theta \longmapsto u\beta\theta \overset{*}{\underset{g}{\longmapsto}} uaw't'$ avec $1 : awt = 1 : aw't'$ ce qui est absurde car G est $LL(1)$.
  - autres cas similaires.
- $\Leftarrow$: C'est immédiat en utilisant la définition.

■

On en déduit une autre caractérisation des grammaires $LL(1)$:

PROPOSITION 6.2.3 *G est $LL(1)$ ssi pour toutes productions distinctes $A \to \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_m$, on a :*

1. *Les ensembles $Premier_1(\alpha_i)$ sont deux à deux disjoints,*
2. *De $\alpha_i \overset{*}{\underset{g}{\longmapsto}} \varepsilon$ il résulte que $Premier_1(\alpha_j) \cap Suivant_1(\alpha_i)$ pour tout $j \neq i$.*

A ce stade, on a tout ce qu'il faut pour montrer que si G est $LL(k)$, alors $L(G)\$$ est reconnu par un automate à pile déterministe par pile vide.

PROPOSITION 6.2.4 *Soit G une grammaire algébrique $LL(1)$. Alors $L(G)\$$, où $\$$ est un caractère n'appartenant pas à $(\Sigma \cup N)$, est reconnu par APD par pile vide.*

PREUVE : On construit à l'aide de G l'automate

$$\mathcal{M} = (\Sigma \cup \{q_0, \$\}, \Sigma \cup \{\$\}, N, \delta, q_0, S)$$

avec δ définie par :
- $\delta(q_0, \$, \$) = (\$, \varepsilon)$ si $\varepsilon \in L(G)$.
- Pour $a \in \Sigma$, $\delta(q_0, a, S) = (a, \alpha)$ si $S \to \alpha \in P$ et $a \in Premier_1(Suivant_1(S))$.
- Pour $a \in \Sigma$, $\delta(a, \varepsilon, A) = (a, \alpha)$ si $A \to \alpha \in P$ et $a \in Premier_1(Suivant_1(A))$.
- Pour $a \in \Sigma$ et $b \in (\Sigma \cup \$)$, $\delta(a, b, a) = (b, \varepsilon)$.

Montrons alors que $L(G)\$ = L_P(G)$. Pour cela, prouvons :

$$A\beta \underset{g}{\overset{*}{\longmapsto}} a_1 \dots a_p \text{ ssi } (a_1, a_2 \dots a_p\$, A\beta) \underset{\mathcal{M}}{\overset{*}{\longmapsto}} (\$, \varepsilon, \varepsilon)$$

Chaque implication sera prouvée par récurrence :

- $\Rightarrow HR_n$= "$\forall k \leq n, A\beta \underset{g}{\overset{n}{\longmapsto}} a_1 \dots a_p \Rightarrow (a_1, a_2 \dots a_p\$, A\beta \underset{\mathcal{M}}{\overset{*}{\longmapsto}} (\$, \varepsilon, \varepsilon)$" :
  - si $n = 1$ Supposons donc que $A\beta \underset{g}{\overset{1}{\longmapsto}} a_1 \dots a_p$. Alors il existe $\alpha, A \to \alpha \in P$ et $A\beta \longmapsto \alpha\beta = a_1 \dots a_p$. Trois cas sont alors possibles :
  - Si $\alpha = \varepsilon$ et $\beta = a_1 \dots a_p$, alors :

    $$(a_1, a_2 \dots a_p\$, A\beta \underset{\mathcal{M}}{\overset{1}{\longmapsto}} (a_1, a_2 \dots a_p\$, \beta) = (a_1, a_2 \dots a_p\$, a_1 \dots a_p) \underset{\mathcal{M}}{\overset{*}{\longmapsto}} (\$, \varepsilon, \varepsilon),$$

    la deuxième dérivation étant bien définie car $a_1 \in Premier_1(\alpha Suivant_1(A))$.
  - Si $\alpha = a_1 \dots a_l$ et $\beta = a_{l+1} \dots a_p$, cela se traite de façon identique.
  - Si $\alpha = a_1 \dots a_p$ et $\beta = \varepsilon$, idem.
  - soit n quelconque. On va découper la dérivation de taille n dans G en une première dérivation $A\beta \underset{G}{\overset{1}{\longmapsto}} \alpha\beta$ et les $n-1$ autres : $\alpha\beta \underset{G}{\overset{n-1}{\longmapsto}} a_1 \dots a_p$. On a alors $A \to \beta \in P$, et en écrivant $\alpha = u\beta\alpha'$, où $u \in \Sigma^*$, ou bien $u = \varepsilon$ et on écrit $A\beta \underset{G}{\overset{1}{\longmapsto}} B\alpha'\beta' \underset{G}{\overset{n-1}{\longmapsto}}$, ou bien $u = a_1 \dots a_l$ et on obtient $B\alpha'\beta \underset{G}{\overset{m}{\longmapsto}} a_{l+1}a_p$ en appliquant le lemme fondamental.

    En appliquant l'hypothèse de récurrence sur la dérivation plus petite (taille $n-1$ ou m), on obtient bien le résultat voulu.

- $\Leftarrow$ On va montrer que $\forall n, (a_1, a_2 \dots a_p\$, A\beta) \underset{\mathcal{M}}{\overset{n}{\longmapsto}} (\$, \varepsilon, \varepsilon) \Rightarrow A\beta \underset{G}{\overset{*}{\longmapsto}} a_1 \dots a_p$.
  - si $n = 2$ Si on a $(a_1, a_2 \dots a_p\$, A\beta) \underset{\mathcal{M}}{\overset{1}{\longmapsto}} (a_1, a_2 \dots a_p\$, \alpha\beta) \underset{\mathcal{M}}{\overset{1}{\longmapsto}} (\$\varepsilon, \varepsilon)$ c'est que $A \to \alpha \in P$ et $a_1 \in Premier_1(\alpha Suivant_1(A))$. Donc ou bien $\alpha = a_1$ et $\beta = \varepsilon$, ou bien $\alpha = \varepsilon$ et $\beta = a_1$, ce qui entraîne dans chaque cas $A\beta \underset{G}{\overset{*}{\longmapsto}} a_1$.
  - soit n quelconque. On découpe là aussi la dérivation en un premier pas, puis les $n-1$ autres à qui on peu appliquer l'HR. Cela se passe bien.

■

REMARQUE : C'est ce dernier automate que l'on a construit en cours de compilation, sauf que les états sont utilisés pour la construction de la table.

6.3 Grammaires $LR(k)$

EXEMPLE 6.3.1 *Considérons la grammaire définie par les productions suivantes :*

$$\begin{cases} S' \to Sc \\ S \to A \mid SA \\ A \to aSb \mid ab \end{cases}$$

DÉFINITION 6.3.1 (FSD)
On appelera **forme syntaxique droite** tout mot $\gamma \in (\Sigma \cup N)^*$ qui se dérive le plus à droite à partir de S. Dans l'exemple précédent, $Saabbc$ est une FSD.

6.3.1 Notion de manche

DÉFINITION 6.3.2
Soit γ une FSD. Un couple $(A \to \beta, i)$ où $A \to \beta \in P$ et $i \geq 0$ est dite une **poignée** pour γ si $S \overset{*}{\underset{d}{\longmapsto}} \alpha A w \longmapsto \alpha\beta w$ avec $\alpha\beta w = \gamma$ et $i = |\alpha\beta|$.

EXEMPLE 6.3.2

- *Soit G définie par les productions suivantes :*

$$\begin{cases} S \to aB \mid ab \\ B \to b \end{cases}$$

  Alors pour la FSD ab, $(S \to ab, 2)$ et $(B \to b, 2)$ sont des poignées. En effet, on a $S \longmapsto ab$ et $S \longmapsto aB \longmapsto ab$.

- *Soit G' définie par les productions suivantes :*

$$\begin{cases} S \to aA \mid Aa \\ A \to a \end{cases}$$

  Alors $(A \to a, 1)$ et $(A \to a, 2)$ sont des poignées pour $\gamma = aa$.

6.3.2 Définition de $LR(k)$

DÉFINITION 6.3.3
Soit G une grammaire algébrique réduite (*i.e.* sans symbole inutile) telle que S ne dérive pas S. Alors G est $LR(k)$ pour $k \geq 0$ si la condition suivante est vérifiée :

$$\begin{cases} (1) \quad S \overset{*}{\underset{d}{\longmapsto}} \alpha A w \longmapsto \alpha\beta w \\ (2) \quad S \overset{*}{\underset{d}{\longmapsto}} \alpha' A' w' \longmapsto \alpha'\beta' w' = \alpha\beta w'' \\ (3) \quad k : w = k : w' \end{cases} \Rightarrow \big(A \to \beta, |\alpha\beta|\big) = \big(A' \to \beta', |\alpha'\beta'|\big),$$

c'est-à-dire $A = A'$, $B = B'$, $|\alpha\beta| = |\alpha'\beta'|$, donc $w' = w''$ et $\alpha' = \alpha$, $\beta' = \beta$.

EXEMPLE 6.3.3
- *Soit G définie par les productions suivantes :*
$$\begin{cases} S \to aAc \\ A \to Abb \mid b \end{cases}$$
est $LR(0)$. En effet, les formes syntaxiques droites que l'on peut obtenir sont :
  - *aAc de poignée $(S \to aAc, 3)$*
  - *$aAb^{2n}c$ de poignée $(A \to Abb, 4)$*
  - *$ab^{2n+1}c$ de poignée $(A \to b, 2)$.*
- *Attention, soit G' définie par les productions suivantes :*
$$\begin{cases} S \to aB \mid ab \\ A \to a \end{cases}$$
Alors cette grammaire engendre le même langage que G mais elle n'est pas $LR(0)$. Il faut considérer $S \underset{d}{\overset{1}{\longmapsto}} aAc \underset{d}{\overset{1}{\longmapsto}} abc$ et $S \longmapsto aAc \longmapsto abAbc \longmapsto abbbc$, on obtient les manches pour ab : $(A \to b, 2)$ et $(A \to b, 3)$.

6.3.3 Non-ambiguité de $LR(k)$

LEMME 6.3.1 *Soit G une grammaire algébrique réduite. Supposons que l'on ait les dérivations : $S \underset{d}{\overset{*}{\longmapsto}} \alpha Aw \longmapsto \alpha\beta w$ et $S \underset{d}{\overset{*}{\longmapsto}} \alpha' A' w' \longmapsto \alpha'\beta' w' = \alpha\beta w$. Alors on a l'équivalence :*
$$\alpha Aw = \alpha' A' w' \text{ ssi } \big(A \to \beta, |\alpha\beta|\big) = \big(A' \to \beta', |\alpha'\beta'|\big)$$

PREUVE : Facile et laissée au lecteur. ■

LEMME 6.3.2 *Soit G une grammaire algébrique réduite. Alors G est non ambiguë si et seulement si toute forme algébrique droite a exactement une poignée à l'exception de S qui n'en a aucune.*

PREUVE :
- $\Rightarrow$: Supposons G non ambiguë :
  - S n'a pas de poignée. Sinon, il existerait une poignée $(A \to \beta, i)$ telle que $S \underset{d}{\overset{*}{\longmapsto}} \alpha Aw \longmapsto \alpha\beta w = S$, $i = |\alpha\beta|$ donc pour tout u mot sur Σ^*, on aurait : $S \underset{d}{\overset{+}{\longmapsto}} S \underset{d}{\overset{*}{\longmapsto}} u$ et $S \longmapsto u$, ce qui contredit l'hypothèse.
  - Toute forme syntaxique droite différente de S possède une poignée puisque G est réduite.
  - Montrons qu'il n'y en a qu'une par FSD. Supposons que $(A \to \beta, i)$ et $(A' \to \beta', i')$ soient des poignées pour γ une FSD. En appliquant la définition des poignées, on obtient l'existance des dérivations :
$$S \longmapsto \alpha Aw \longmapsto \alpha\beta w \underset{d}{\overset{*}{\longmapsto}} u \text{ et } S \longmapsto \alpha' A' w' \longmapsto \alpha'\beta' w' = \gamma = \alpha\beta w \underset{d}{\overset{*}{\longmapsto}} u$$

, ce qui contredit le fait que G est supposée non ambiguë.

- $\Leftarrow$: Réciproquement, montrons que G n'est pas ambiguë. Supposons le contraire. Alors il existe un mot u qui admet deux dérivations droites distinctes :

  $$\alpha_0 = S \longmapsto \alpha_1 \longmapsto \ \ldots \ \longmapsto \alpha_m = u \text{ et } \alpha'_0 = S \longmapsto \alpha'_1 \longmapsto \ \ldots \ \longmapsto \alpha_{m'} = u$$

  Alors il existe certainement i, $\alpha_{m-i} \neq \alpha'_{m'-i}$ (distinguer les cas $m = m'$ et $m \neq m'$). Soit alors j le plus petit entier vérifiant $\alpha_{m-j} \neq \alpha'_{m'-j}$. Alors on obtient $\alpha_{m-j+1} = \alpha'_{m'-j+1}$. Comme toute forme syntaxique a exactement une poignée, on déduit que $\alpha_{m-j} = \alpha_{m'_j}$. Ce qui est contradictoire avec j minimal.

■

On en déduit immédiatement le corollaire à l'aide du lemme précédent :

COROLLAIRE 6.3.1 *Si G est une grammaire LL(k), alors elle n'est pas ambiguë.*

6.3.4 L'automate caractéristique d'une grammaire algébrique

DÉFINITION 6.3.4
On appelle **préfixe viable** de G tout mot $\mu \in (\Sigma \cup N)^*$ tel que

$$S \overset{*}{\underset{d}{\longmapsto}} \alpha A w \longmapsto \alpha\beta w, \text{ et } \mu \text{ préfixe de } \alpha\beta.$$

DÉFINITION 6.3.5
On appelle **item** de G tout objet $[A \to \beta_1 \bullet \beta_2]$ tel que $A \to \beta_1\beta_2 \in P$.

DÉFINITION 6.3.6
On appelle **item complet** un item $[A \to \bullet]$ ou $[A \to \beta\bullet]$.

DÉFINITION 6.3.7
Un item $[A \to \beta_1 \bullet \beta_2]$ est **valide** pour un préfixe viable μ de la grammaire si il existe une dérivation de la forme :

$$S \overset{*}{\underset{d}{\longmapsto}} \alpha A w \longmapsto \alpha\beta_1\beta_2 w,$$

telle que $\mu = \alpha\beta_1$.

On a alors la proposition suivante :

PROPOSITION 6.3.1 *Pour tout préfixe viable μ, il existe un item valide pour μ.*

PREUVE : facile mais à faire! ■

DÉFINITION 6.3.8 (AUTOMATE CARACTÉRISTIQUE)
L'automate caractéristique $\mathcal{C}_G$ d'une grammaire G a pour états les items de G ($Q_\mathcal{C}$), son alphabet est $\Sigma \cup N$. Son état initial est $[S' \to \bullet S] = q_{0c}$ et les états d'acceptation sont $F_\mathcal{C}\{[A \to B\bullet] \mid A \to B \in P\}$. Enfin, la fonction de transition $\delta_\mathcal{C}$ est donnée par :

– $\delta([A \to \alpha \bullet X\beta], X) = \{[A \to \alpha X \bullet \beta]\}$ quand $A \to \alpha X \beta$ avec $X \in \Sigma \cup N$.
– $\delta([A \to \alpha \bullet B\beta], \varepsilon) = \{[B \to \bullet u], B \to u \in P\}$ pour $B \in N$.

PROPOSITION 6.3.2 *Soient $u \in (\sigma \cup N)^*$ et $q \in Q_{\mathcal{C}}$. Alors $q \in \delta_{\mathcal{C}}(q_{0c}, u)$ ssi μ est un préfixe viable de G et q est un item valide pour μ.*

PREUVE :

- $\Rightarrow$. Supposons que $q \in \delta_{\mathcal{C}}(q_{0c}, \mu)$. Il existe donc un chemin de longueur n menant de q_{0c} à q dans l'automate. On va montrer par récurrence sur n la longueur du chemin la proposition.
  – si $n = 1$ Alors $q_{0c} = [S' \to \bullet S] \to_{\mu} [S \to \bullet\gamma]$ et nécessairement $\mu = eps$ et $S \to \gamma \in P$. Alors $S \overset{1}{\underset{\mathcal{C}}{\longmapsto}} \gamma$ et $[S \to \bullet\gamma]$ est valide pour ε.
  – On suppose que c'est vrai pour les chemins de longueur inférieur strict à n. On va montrer que c'est vrai pour n. Si on prend un chemin de longueur n, il se segmente en un chemin de longueur $n-1$ et se termine sur une transition sur ε ou une transition à l'aide d'un élément de $\Sigma \cup N$:
    – Si le chemin est de la forme $q_{0c} \to_{\mu} q' \to_{\varepsilon} q$, alors par HR, q' est un item valide pour μ. On note $q' = [A \to \beta_1 \bullet \beta_2]$ et on a alors $S \overset{*}{\underset{d}{\longmapsto}} \alpha A w \longmapsto \alpha\beta_1\beta_2 w$ avec $\mu = \alpha\beta_1$. Comme dans l'automate la transition (q, q') est étiquetée par ε, on a appliqué la seconde règle de la définition pour construire cette transition. Par conséquent, β_2 commence par une variable B et donc s'écrit $\beta_2 = B\beta_2'$ et aussi on a $q = [B \to \bullet\gamma]$. Alors les dérivations $S \overset{*}{\underset{d}{\longmapsto}} \alpha A w \longmapsto \alpha\beta_1 B\beta_2' \longmapsto \alpha\beta_1\gamma\beta_2' w$ prouvent que q est valide pour μ.
- $\Leftarrow$. On suppose que q est un item valide pour le préfixe viable μ. On va montrer par récurrence la longueur de μ que $q \in \delta_{\mathcal{C}}(q_{0c}, \mu)$
  – si $n = 0$: $|\mu| = 0$ signifie $\mu = \varepsilon$, et $q = [A \to \beta_1 \bullet \beta_2$ est un item valide pour ε, ce qui signifie qu'il existe une dérivation la plus à droite du type : $S \overset{*}{\underset{d}{\longmapsto}} \alpha A w \longmapsto \alpha\beta_1\beta_2 w$ tel que $\alpha\beta_1 = \varepsilon = \alpha = \beta_1$. Alors en réécrivant cette dernière dérivation : $S \overset{*}{\underset{d}{\longmapsto}} Aw$ et $A \to \beta_2 \in P$. Cela implique que $S \to X\alpha_1 w \overset{*}{\underset{d}{\longmapsto}} Xw' \longmapsto A\beta' w'$ (X dernière variable avant A). D'où $S \to X\alpha_1 w_1 \in P$ et $X \to a\beta' \in P$. Alors dans l'automate $[S' \to S] \to_{\varepsilon} [S \to \bullet X\alpha_1 w_1] \to_{\varepsilon} [X \to \bullet A\beta'] \to_{\varepsilon} [A \to \bullet\beta_2]$, donc $q_{oc} \to_{\varepsilon} q$ et c'est gagné.
  – On suppose que c'est vrai pour les mots de longueur inférieur strict à n. On va montrer que c'est vrai pour n. Si μ est un préfixe viable et $q = [A \to \beta_1 \bullet \beta_2]$ est un préfixe viable pour μ, alors par définition on a la dérivation : $S \overset{*}{\underset{d}{\longmapsto}} \alpha A w \longmapsto \alpha\beta_1\beta_2 w$ et $\mu = \alpha\beta_1$. On écrit alors $\mu = \mu' X$ avec $|\mu'| = |mu| - 1 = n - 1$.
    – Si $\beta_1 \neq \varepsilon$, alors $\beta_1 = \beta_1' X$ et par HR $[A \to \beta_1' \bullet X\beta_2]$ est un item valide pour μ'. donc on a :

$$q_{0c} \to_{\mu'} [A \to \beta_1' \bullet X\beta_2] \to_X [A \to \beta_1' X \bullet \beta_2] = [A \to \beta_1 \bullet \beta_2]$$

    et c'est bien ce qu'on veut.

– Si $\beta_1 = \varepsilon$, alors $q = [A \to \bullet\beta_2]$, $\alpha = \mu' X$, $S \underset{d}{\overset{*}{\vdash}} \alpha A w \underset{\beta_2}{\overset{\alpha}{\vdash}} w = \mu' X \beta_2 w$. Donc cela s'écrit encore :

$$S \underset{d}{\overset{*}{\vdash}} \theta' B w' \longmapsto \theta\nu X \rho w' \underset{d}{\overset{*}{\vdash}} \theta\nu X A w \longmapsto \theta\nu X \beta_2 w = \mu' X \beta_2 w$$

Alors ρ s'écrit forcément $C\rho'$ donc $\rho w' = C\rho' w' \underset{d}{\overset{*}{\vdash}} A\nu' w'$, avec $B \to \nu X \rho \in P$. Alors $[B \to \nu \bullet X\rho]$ est valide pour $\theta\nu = \mu'$:

$$q_{0c} \to_{\mu'} [B \to \nu \bullet X\rho] \to_{\alpha} [B \to \nu X \bullet \rho] \to_{\varepsilon} [C \to \bullet A\nu'] \to_{\varepsilon} [A \to \bullet\beta_2]$$

■

On déterminise ensuite l'automate $\mathcal{C}_G$ en $\mathcal{D}_G$. Alors on obtient une caractérisation des grammaires algébriques $LR(0)$:

Proposition 6.3.3 *Une grammaire algébrique réduite (i.e. sans caractères inutiles) est $LR(0)$ ssi :*

1. *S ne dérive pas S à droite,*
2. *$\mathcal{D}_G$ ne contient pas d'état impropre (i.e. il ne contient pas plusieurs items complets distincts ni d'item complet accompagné d'items incomplets)*

Preuve : La preuve, similaire à la précédente, est laissée au lecteur. ■

6.3.5 Automate reeconnaisseur de $LR(0)$

On va,dans cette section, décrire un automate à pile reconnaissant le langage engendré par une grammaire algébrique $LR(0)$.

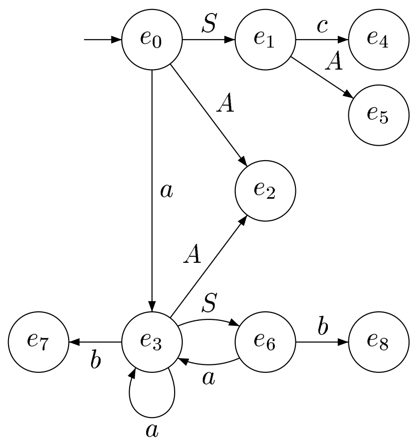
Sa fonction de transition sera notée $\tau : Q \times \gamma \times \Sigma \cup \{\varepsilon\} \to Q \times \Gamma^+$. De plus, dans le mot de pile, la lettre la plus à droite sera ici le sommet de pile. L'alphabet de pile est $Q_{\mathcal{D}} \cup N \cup \Sigma$, où $Q_{\mathcal{D}} = \{e_0, e_1, \dots e_k\}$ sont les états de l'automate déterminisé précédent (e_0 était l'état initial). e_0 sera le symbole de fin de pile. L'état initial de cet automate à pile sera q_0. Construisons maintenant la fonction de transition :

- $\tau(q_0, e_0, a) = (q_0, e_0 a \delta_{\mathcal{D}}(e_0, a))$.
- $\tau(q_0, e, a) = (q_0, e, a) = (q_0, e_0 a \delta_{\mathcal{D}}(e_0, a))$ si e ne contient pas d'item complet, et $a \in \Sigma$.
- Si e contient un item complet $[A \to \beta_1 \dots \beta_n \bullet]$, chaque $\beta_i \in \Sigma \cup N$, le mot de sommet de pile est alors de la forme $e'\beta_1 e_1 \beta_2 e_2 \dots \beta_n e_n$. Alors on va le manger, d'où les règles suivantes :
  - $\tau(q_0, e_n, \varepsilon) = (q_1, \varepsilon)$ (changement d'état “dépile”)
  - $\tau(q_1, \beta_n, \varepsilon) = (q_2, \varepsilon)$
  - ...
  - $\tau(q_1, \beta_1, \varepsilon) = (q_{2n}, \varepsilon)$
  - $\tau(q_{2n}, e', \varepsilon) = \begin{cases} (q_0, e' A \delta_{\mathcal{D}}(e', A) & \text{si } A \neq S' \\ (q_0, e' S') & \text{si } A = S' \end{cases}$
- $\tau(q_0, S', \varepsilon) = (q^*, \varepsilon)$ (c'est fini)
- $\tau(q^*, e_0, \varepsilon) = (q^*, \varepsilon)$.

EXEMPLE 6.3.4 *Pour la grammaire :*

$$G = \begin{cases} S' \to Sc \\ S \to A \mid SA \\ A \to aSb \mid ab \end{cases},$$

on a obtenu l'automate suivant :



En notant les états :

- $e_0 = \{S' \to \bullet Sc, S \to \bullet SA, S \to \bullet A, A \to \bullet aSb, A \to \bullet ab\}$
- $e_1 = \{S' \to S \bullet c, S \to S \bullet A, A \to \bullet aSb, A \to \bullet ab\}$
- $e_2 = \{S \to A\bullet\}$
- $e_3 = \{A \to a \bullet Sb, A \to a \bullet b, A \to \bullet aSb, S \to \bullet A, S \to \bullet SA, A \to \bullet ab\}$
- $e_4 = \{S' \to Sc\bullet\}$
- $e_5 = \{S \to SA\bullet\}$
- $e_6 = \{S \to S \bullet A, A \to aS \bullet b, A \to \bullet ab, A \to \bullet aSb\}$
- $e_7 = \{A \to ab\bullet\}$
- $e_8 = \{A \to aSb\bullet\}$

Alors pour le mot abaabc, on a les mots de pile successifs : $e_0ae_3be_7$*, puis comme* e_7 *contient un item "Reduce" on obtient* e_0Ae_2*, puis* $e_0Se_1ae_3a_3ba_7$*, etc.*

6.3.6 Grammaires et langages $LR(1)$

On admettra ici que tout langage $LR(1)$ est algébrique déterministe.

THÉORÈME 6.3.1 *Tout langage algébrique déterministe est engendré par une grammaire* $LR(1)$.

Le cheminement de la preuve est le suivant :

- Soit L un langage algébrique déterministe. Alors il est reconnu par un APD par état final. Alors $L\$$ est reconnu par un APD par pile vide. $L\$$ a aussi la propriété du préfixe.
- Notion de grammaire algébrique stricte :

  DÉFINITION 6.3.9
  Une grammaire algébrique est dite déterministe **stricte** si il existe une partition Π de $N \cup \Sigma$ telle que :

1. $\Sigma \in \Pi$
2. pour tous A, A' de N, pour tous $\alpha, \beta, \beta' \in (\Sigma \cup N)^*$, si $A \to \alpha\beta \in P$, $\alpha \to \alpha\beta' \in P$) et $A \equiv A'[\Pi]$, alors
   – ou bien β et β' sont distincts de ε et $1 : \beta \equiv 1 : \beta'[\Pi]$
   – ou bien $\beta = \beta' = \varepsilon$ et $A = A'$.

- On montre le théorème suivant :

  Théorème 6.3.2 *La famille des langages algébriques ayant la propriété su préfixe est celle des langages engendrés par les grammaires algébriques déterministes strictes.*

- Avec la définition suivante :
- Il existe un algorithme qui décide si une grammaire algébrique donnée est déterministe stricte ou non.
- Toute grammaire algébrique stricte est $LR(0)$. Donc $L\$$ est engendré par une grammaire $LR(0)$, et qui plus est par une grammaire $LR(0)$ sans ε-transition.
- On utilise enfin(!) le lemme :

  Lemme 6.3.3 *Soit G une grammaire $LR(0)$ de la forme $(N \cup \{\$\}, \Sigma \cup \{\$\}, P, S)$ où $P \subseteq N \times \big((N \cup \Sigma)^* \times (n \cup \Sigma)^*\$\big)$ et $L(G) \subseteq \Sigma^*\$$ et telle qu'il n'existe pas de dérivation pouvant mener de S à $S\$$ dans G. Alors la grammaire $G' = (N, \Sigma, P', S)$ avec $P' = P_1 \cup P_2$, $P_1 = \{A \to \beta \mid A \to \beta \in P, \beta \in (N \cup \Sigma)^*\}$ et $P_2 = \{A \to \beta \mid A \to \beta\$ \in P\}$ est $LR(1)$ et $L(G')\$ = L(G)$.*

Théorème 6.3.3 *Tout langage engendré par une grammaire $LR(k)$ est algébrique déterministe.*

On va maintenant construire un analyseur $LR(k)$, c'est-à-dire un automate déterministe pour une grammaire $LR(k)$.

6.3.7 Analyseur $LR(k)$

Définition 6.3.10 ($LR(k)$-item)
Un $LR(k)$**-item** pour une grammaire algébrique est $[A \to \beta_1 \bullet \beta_2, u]$, où $u \in Suivant_k(A)$ et $A \to \beta_1\beta_2 \in P$.

Remarque : En cours de compilation, on a noté cela $[A \to \beta_1 \bullet \beta_2]\{u\}$.

Définition 6.3.11
L'item $[A \to \beta_1 \bullet \beta_2, u]$ est dit **valide** pour un préfixe viable μ si

$$S \overset{*}{\underset{d}{\longmapsto}} \alpha A w \longmapsto \alpha\beta_1\beta_2 w \text{ avec } \mu = \alpha\beta_1 \text{ et } u = k : w$$

Il existe un algorithme qui permet de calculer les ensembles d'items valides pour les préfixes viables de la grammaire.

On note $\mathcal{S}_G$ l'ensembles des ensembles de $LR(k)$-items valides pour les préfixes viables de G.

Définition 6.3.12
Un ensemble $\mathcal{S}$ d'ensembles de $LR(k)$-items valides est dit **consistant** si pour tout ensemble I de $\mathcal{S}$, $\forall A, A' \in N$, $\forall \beta, \beta_1, \beta_2 \in (\Sigma \cup N)^*$ vérifiant $1 : \beta_2 \notin N$ et pour u, v mots de Σ^* de taille $\leq k$, $u \in Premier_k(\beta_2 v)$, on a :

$$\Big([A \to \beta \bullet, u] \in I, [A \to \beta_1 \bullet \beta_2] \in P\Big) \Rightarrow \Big([A \to \beta \bullet, u] = [A' \to \beta_1 \bullet \beta_2, v]\Big)$$

On a alors :

Théorème 6.3.4 *G est $LR(k)$ ssi $\mathcal{S}_{k,G}$ est consistant.*

Tous ces résultats nous permettent de construire l'analyseur $LR(k)$ cherché :

On commence par définir deux "fonctions" f et g dites d'action et de succession. Soit T un ensemble de $LR(k)$-items de G et $u \in \Sigma^{\leq k}$. On définit :

1. $f(T, u)$: **décaler** si il existe $\beta_1 \in (N \cup \Sigma)^*$, $\beta_1 \in \Sigma(\mathbb{N} \cup \Sigma)^*$, $v \in \Sigma^{\leq k}$ et $A \in N$ vérifiant :
$$[A \to \beta_1 \bullet \beta_2] \in T \text{ avec } u \in Premier_k(\beta_2 v)$$
2. $f(T, u)$: **réduire** la production $A \to \beta$ si $[A \to \beta \bullet, u] \in T$.
3. $f(T, u)$: **erreur** partout ailleurs
4. si on note $T = V_k(\gamma)$ (*i.e.* γ préfixe viable et $V_k(\gamma)$ valide pour γ) et $X \in (N \cup \Sigma)$:
   - $g(V_k(\gamma), X) = V_k(\gamma X)$ si $V_k(\gamma) \neq \emptyset$
   - $g(V_k(\gamma), X)$: **erreur** sinon.

Proposition 6.3.4 *Si l'ensemble d'ensemble d'items T est consistant, alors f et g sont vraiment des fonctions.*

On numérote les productions de la grammaire. Soit $(\gamma T, z, \rho)$ une configuration, où γT désigne le contenu de la pile (T est au dessus), z l'entrée, ρ la sortie. Quelle est la configuration suivante ?

1. Si $f(T, k : z)$=décaler alors
   - si $z = \varepsilon$ retourner ERREUR
   - si $z \neq \varepsilon$, notons $z = bz', b \in \Sigma$:
     - si $g(T, b)$=erreur alors retourner ERREUR
     - sinon $(\gamma, T, bz', \rho) \longmapsto (\gamma T b g(T), z', \rho)$
2. Si $f(T, k : z)$=réduire(i) alors on veut dépiler $2\,|\beta|$ symboles si la production numéro i est $A \to \beta$. Donc si $\gamma T = \gamma T \gamma''$ avec $|\gamma''| = 2\,|\beta|$, on a deux cas :
   - si $T' = T_0$ (fonds de pile), $S = A$, $z = \varepsilon$, alors $(\gamma T, z, rho) \longmapsto (T_0, \varepsilon, \rho i)$ ACCEPTE
   - si $T \neq T_0$ alors :
     - si $g(T', A)$=erreur alors retourner ERREUR
     - sinon la nouvelle configuration est $(\gamma T' A g(T', A), z, \rho i)$.
3. si $f(T, k : z)$=erreur alors retourner ERREUR

Chapitre 7

Langages rationnels de mots infinis

7.1 Premières définitions

On prend dans toute la suite Σ un alphabet **fini**. On note $\Sigma^\infty = \Sigma^* \cup \Sigma^{\mathbb{N}}$, où $\Sigma^{\mathbb{N}}$ est l'ensemble des mots "infinis" sur Σ, *i.e.* les suites d'éléments pris dans Σ. On notera souvent $\Sigma^\omega = \Sigma^{\mathbb{N}}$. On va comme d'habitude définir des opérations sur ces objets.

La concaténation "à gauche" : si $u \in \Sigma^*$, $v \in \Sigma^\omega$, alors $uv \in \Sigma^\omega$ défini intuitivement en "collant" le mot fini u à gauche du mot infini v.

L'opération puissance : pour $X \subseteq \Sigma^*$, on note $X^\omega = \{(x_i)_{i\geq 0}, x_i \in X, x_i \neq \varepsilon\}$. Ceci est cohérent avec la notation Σ^ω. Pour $X = \{\varepsilon\}$, $X^\omega = \emptyset$.

7.2 Langages

On peut énoncer quelques résultats élémentaires sur le comportement de ces opérateurs afin d'étendre ceux déjà énoncés dans le cas des mots finis :

PROPOSITION 7.2.1 *Soient X et Y des langages sur Σ contenus dans Σ^*. On a alors*

1. $(X+Y)^\omega = (X^*Y)^\omega + (X+Y)^*X^\omega$
2. $(XY)^\omega = X(YX)^\omega$
3. *pour tout entier n,* $(X^n)^\omega = (X^+)^\omega = X^\omega$
4. $XX^\omega = X^+X^\omega = X^\omega$

Ces résultats sont plutôt intuitifs et se prouvent sans difficulté.

On va à présent étendre la notion de langage rationnel au cas des mots infinis. Commençons par donner une définition formelle.

DÉFINITION 7.2.1
La classe $Rat(\Sigma^\infty)$ est la plus petite classe $\mathcal{R}$ de langages sur Σ qui vérifie

- $\emptyset \in \mathcal{R}$ et pour tout élément a de Σ, $\{a\} \in \mathcal{R}$;

– $\mathcal{R}$ est stable par union finie ;
– si X et Y sont deux éléments de $\mathcal{R}$ contenus dans Σ^+ et Σ^∞ respectivement, alors leur concaténation XY est élément de $\mathcal{R}$;
– pour tout X de $\mathcal{R}$ inclus dans Σ^*, X^* et X^ω sont éléments de $\mathcal{R}$.

Dans la suite, on s'intéresse plus particulièrement aux mots de $Rat(\Sigma^\infty)$ inclus dans Σ^ω, langages qu'on appelle ω-rationnels.

On a facilement la

PROPOSITION 7.2.2 *Soit $X \in Rat(\Sigma^\infty)$. Alors*

1. *$X \cap \Sigma^*$ est un langage rationnel sur Σ*
2. *$X \cap \Sigma^\omega \in Rat(\Sigma^\infty)$.*

PREUVE : Soit $\mathcal{E}$ la classe des langages de Σ^∞ qui satisfont les propriétés 1 et 2. Alors :

- $\emptyset \in \mathcal{E}$, $\{a\} \in \mathcal{E}$,
- $\mathcal{E}$ est close par union finie,
- $\mathcal{E}$ est close par produit. Soient $X \subseteq \Sigma^+$, $Y \in \Sigma^\omega$ et $X, Y \in \mathcal{E}$, alors on a $XY \cap \Sigma^* = X(Y \cap \Sigma^*)$ et $XY \cap \Sigma^\omega = X(Y \cap \Sigma^\omega)$. Donc $XY \in \mathcal{E}$.
- $\mathcal{E}$ est close par itération infinie.

Donc d'après la définition de $Rat(\Sigma^\infty)$, $Rat(\Sigma^\infty) \subseteq \mathcal{E}$. C'est ce qu'on voulait. ∎

Ce qui nous amène à la première caractérisation suivante :

PROPOSITION 7.2.3 *Une partie de Σ^ω est un langage ω-rationnel si et seulement si c'est une union finie de langages du type XY^ω avec X et Y rationnels (donc composés de mots finis).*

PREUVE : La preuve, facile, est laissée au lecteur. ∎

7.3 Automates de Büchi

On va maintenant chercher comme précédemment à caractériser cette nouvelle classe de langages à l'aide d'un modèle de calcul adapté. Comme la définition employée est proche de celle des langages rationnels, on s'intéresse à une caractérisation par des automates finis.

C'est ici qu'un nouveau problème apparaît, car il n'y a pas de définition de l'acceptation qui s'impose d'elle-même pour des mots infinis. En effet, on ne peut plus considérer d'état final puisque la lecture d'un mot est précisément infinie. C'est donc sur la définition de la condition d'acceptation que vont différer les différents modèles étudiés.

Le premier modèle d'automate reconnaissant des mots infinis est celui des automates de Büchi, qui est peut-être l'extension la plus naturelle du modèle fini :

DÉFINITION 7.3.1 (AUTOMATE DE BÜCHI)
Un automate de Büchi est un quintuplet (Q, Σ, E, I, F) où

– Q est l'ensemble des états ;
– Σ est l'alphabet ;

- E, partie de $Q \times \Sigma \times Q$, est la table de transition ;
- $I \subseteq Q$ est l'ensemble des états initiaux ;
- $F \subseteq Q$ est l'ensemble d'acceptation.

Définition 7.3.2 (Chemin)

Avec cette définition, un chemin est une suite $(q_i, a_i, q'_i)_{i \geq 1}$. Un tel chemin est dit "réussi" si il est cohérent, *i.e.* quel que soit i, on a $q_{i+1} = q'_i$ et il commence par $q_0 \in I$. Un mot infini **étiquette** un chemin si sa suite de lettres (a_i) est une suite de a_i apparaissant dans un chemin réussi.

Définition 7.3.3 (Inf)

Pour un mot infini u donné sur Σ, on obtient alors naturellement un ensemble de chemins $c \in Q^{\mathbb{N}}$ associés à u. On note ensuite $\mathrm{Inf}(c)$ l'ensemble des états rencontrés une infinité de fois par un chemin c. Le chemin c est alors acceptant si $\mathrm{Inf}(c)$ rencontre F, c'est-à-dire si $\mathrm{Inf}(c) \cap F \neq \emptyset$.

Définition 7.3.4 (Langage)

La langage des mots reconnus par $\mathcal{M}$, automate de Büchi, est l'ensemble des mots infinis étiquetant un chemin réussi dans $\mathcal{M}$ tel que $\mathrm{Inf}(c) \cap F \neq \emptyset$. Comme F est fini, cela se traduite par l'existance d'un élément de F qui apparaît une infinité de fois sur le chemin c.

On note en général :

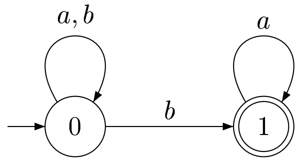
- $\mathcal{L}^+(\mathcal{M})$ l'ensemble des mots finis reconnus par $\mathcal{M}$, à l'exception de ε.
- $\mathcal{L}^\omega(\mathcal{M})$ l'ensemble des mots infinis reconnus par $\mathcal{M}$.

Une question naturelle alors est de se poser la question

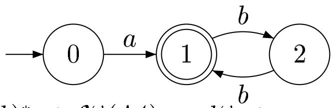
$$\mathcal{L}^\omega(\mathcal{M}) = (\mathcal{L}^+(\mathcal{M}))^\omega ?$$

La réponse est non :

- $(\mathcal{L}^+(\mathcal{M}))^\omega \not\subseteq \mathcal{L}^\omega(\mathcal{M})$, en considérant l'automate suivant :



  Alors, $\mathcal{L}^\omega(\mathcal{M}) = (a+b)^* a^\omega$ et $\mathcal{L}^+(\mathcal{M}) = (a+b)^* ba^*$, $b \in \mathcal{L}^+(\mathcal{M})$ alors que $b^\omega \notin \mathcal{L}^\omega(\mathcal{M})$.
- $\mathcal{L}^\omega(\mathcal{M}) \not\subseteq (\mathcal{L}^+(\mathcal{M}))^\omega$, en considérant l'automate suivant :



  Alors, $\mathcal{L}^+(\mathcal{M}) = a(bb)^*$ et $\mathcal{L}^\omega(\mathcal{M}) = ab^\omega$ et ce qu'on veut.

Remarque : On a donc une forme d'automate non déterministe. On peut en déduire une notion d'automate de Büchi déterministe en imposant à E d'être une fonction et en imposant également que I contienne exactement un élément. On verra plus tard des propriétés correspondant à cette forme d'automate.

Cette définition, dans le cas non déterministe, aboutit à la caractérisation voulue :

PROPOSITION 7.3.1 *Une partie X de Σ^ω est reconnaissable par un automate de Büchi non déterministe si et seulement si elle est ω-rationnelle.*

PREUVE :

- Supposons X reconnu par un Büchi M non déterministe. On va prouver que X est ω-rationnel. Il apparaît que pour tout mot infini u de X, il existe un chemin acceptant de M correspondant à u, et ce chemin rencontre au moins un état d'acceptation f une infinité de fois. Le chemin est alors de la forme
$$i \to \cdots \to f \to \cdots \to f \to \cdots \to f \to \cdots$$
donc u est élément du langage
$$\bigcup_{i \in I} \bigcup_{f \in F} L^*(i, E, f)(L^+(f, E, f))^\omega$$
et il est facile de voir que le langage ainsi défini ne contient pas de mot qui ne soit pas accepté par l'automate M, donc X est ω-rationnel d'après la première caractérisation énoncée, puisque les L^* et L^ω sont de la forme voulue.
- Réciproquement, si X est supposé ω-rationnel, il peut s'écrire sous la forme $\bigcup_{i=1}^n X_i Y_i^\omega$ et on en déduit un automate non déterministe à de Büchi qui reconnaît X à partir des automates finis reconnaissant les X_i et les Y_i.

■

On a donc caractérisé la classe des langages ω-rationnels à l'aide d'un modèle d'automates non déterministes. Cela dit, on aimerait avoir une caractérisation par des automates déterministes. Sur ce point, les langages de mots infinis diffèrent des langages de mots finis, en effet **les modèles déterministe et non déterministe des automates de Büchi ne caractérisent pas la même classe de langages** [1].

Afin de caractériser la classe des langages reconnus par les automates de Büchi déterministes, introduisons la notation suivante : pour un langage L de mots finis sur Σ, on pose
$$\overrightarrow{L} = u \mid u \in \Sigma^\omega, u \text{ a une infinité de préfixes dans } L$$
Par exemple, si $L = a^*b$, le langage $\overrightarrow{L}$ est vide car tout mot a au plus un préfixe dans L. Par contre, si $L = (ab)^+$, le langage $\overrightarrow{L}$ est $(ab)^\omega$. De même, si $L = (a^*b)^+$, le langage $\overrightarrow{L}$ est $(a^*b)^\omega$.

REMARQUE : Tous les langages de mots infinis ne sont naturellement pas de la forme $\overrightarrow{L}$, ni même tous les langages ω-rationnels. Considérons par exemple le langage $X = (a+b)^*a^\omega$. Si on avait $X = \overrightarrow{L}$ pour un langage L donné,

- comme $ba^\omega \in X$, L contiendrait un ba^{n_1}
- comme $ba^{n_1}ba^\omega \in X$, L contiendrait un $ba^{n_1}ba^{n_2}$
- et ainsi de suite …

On aurait donc une suite (n_k) telle que pour tout entier k le mot $ba^{n_1} \dots ba^{n_k}$ soit élément de L, par conséquent la concaténation de la suite des ba^{n_k} pour k parcourant $\mathbb{N}$ serait dans $\overrightarrow{L}$ et donc dans X, ce qui est absurde puisque ce mot contient une infinité de fois la lettre b.

[1] Loupé !

Cet opérateur permet cependant de définir exactement la classe des langages reconnus par automates de Büchi déterministes :

PROPOSITION 7.3.2 *Soit M un automate de Büchi déterministe. On a*

$$L^{\omega}(M) = \overrightarrow{L^{+}(M)}$$

On peut à présent énoncer une caractérisation satisfaisante des langages reconnus par automates de Büchi déterministes :

PROPOSITION 7.3.3 *Soit X un langage de mots infinis sur un alphabet Σ. X est reconnu par un automate de Büchi déterministe si et seulement s'il existe un langage L rationnel sur Σ tel que X soit le langage $\overrightarrow{L}$.*

PREUVE : Laissée en exercice. ■

7.4 Automates de Muller

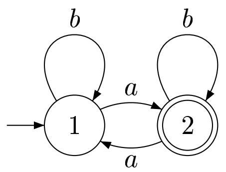
Le modèle précédent n'est donc pas totalement satisfaisant car il ne fournit pas de caractérisation des langages ω-rationnels par un modèle d'automate déterministe. C'est pour cette raison qu'ont été introduits les automates de Muller.

DÉFINITION 7.4.1 (AUTOMATE DE MULLER)
Un automate de Muller est un quintuplet $(Q, \Sigma, \delta, q_0, \mathcal{L})$ où

- Q est l'ensemble fini des états ;
- Σ est l'alphabet ;
- δ est la fonction de transition, de $Q \times \Sigma$ dans Q ;
- $q_0 \in Q$ est l'état initial ;
- $\mathcal{L}$ est une partie de $\mathcal{P}(Q)$.

Dans cette définition, $\mathcal{L}$ est donc l'ensemble d'acceptation de l'automate. On dira alors qu'un mot u infini est accepté par l'automate si il existe un chemin c qui lui est associé vérifie $\mathrm{Inf}(c) \in \mathcal{L}$, *i.e.* il existe un chemin c, tel que $\exists T \in \mathcal{L}$, $\mathrm{Inf}(c) = T$.

EXEMPLE 7.4.1 *Pour l'automate suivant :*



avec $\mathcal{L} = \{\{2\}\}$, on obtient le langage des mots ne contenant que des b à partir d'un certain rang, et avec un nombre impair de a (on s'interdit de passer une infinité de fois par l'état 1).

On a obtient à nouveau avec ce modèle des langages ω-rationnels :

PROPOSITION 7.4.1 *Soit $X \subseteq \Sigma^{\omega}$ un langage reconnu par un automate de Muller. Alors X est ω-rationnel.*

PREUVE : On note toujours $L^\omega(M)$ le langage de mots infinis reconnu par l'automate M dans le cas d'automates de Muller, lorsqu'il n'y a pas d'ambiguïté. Pour un automate $M = (Q, \Sigma, \delta, q_0, \mathcal{L})$, on a alors clairement

$$L^\omega(M) = \bigcup_{T \in \mathcal{L}} L^\omega(Q, \Sigma, \delta, q_0, \{T\})$$

Comme la classe des langages ω-rationnels est stable par réunion finie, on peut donc se limiter à considérer le cas où $\mathcal{L}$ est réduit à une seule partie T de Q, puisque $\mathcal{L}$ est forcément fini.

Soit alors u un mot de $L^\omega(M)$ et soit c le chemin de M associé à u. On sait donc que $\mathrm{Inf}(c) = T$, donc il existe un rang à partir duquel c est entièrement contenu dans T et chaque élément de T y apparaît une infinité de fois. On peut donc décomposer c sous la forme

$$\underbrace{q_0 \longrightarrow^*}_{X} \underbrace{t_0 \longrightarrow^*}_{Y_0} \underbrace{t_1 \longrightarrow^*}_{Y_1} \cdots \underbrace{t_m \longrightarrow^*}_{Y_m} t_0 \longrightarrow^* \cdots \longrightarrow^* t_m \longrightarrow^* t_0 \longrightarrow^* \cdots$$

Ceci induit une décomposition de u en mots des langages
- $X = L^+(Q, \Sigma, \delta, q_0, \{t_0\})$
- $Y_i = L^+(Q, \Sigma, \delta, t_i, \{t_{i+1}\})$ pour $0 \le i < m$
- $Y_m = L^+(Q, \Sigma, \delta, t_m, \{t_0\})$

comme l'indique le schéma. Comme les états initiaux et terminaux dans les définitions des Y_i sont toujours distincts, on sait de plus que les Y_i ne contiennent jamais ε, et par conséquent on a bien

$$L^\omega(M) = X(Y_0 Y_1 \ldots Y_m)^\omega$$

ce qui prouve que $L^\omega(M)$ est ω-rationnel. ■

REMARQUE : On a donc Müller $\Rightarrow$ Büchi à ce stade.

On va à présent chercher à minimiser les automates de Muller afin de prouver de nouvelles caractérisations. On introduit pour cela les notions suivantes :
- **un état accessible** d'un automate de Muller est un état atteignable au sens des automates finis ;
- **un état co-accessible** d'un automate de Muller d'ensemble d'acceptation $\mathcal{L}$ est un état q tel qu'il existe dans M un chemin c d'origine q tel que $\mathrm{Inf}(c) \in \mathcal{L}$;
- **une partie admissible** de Q est une partie T telle qu'il existe dans M un chemin c partant de l'état initial de M et telle que $\mathrm{Inf}(c) = T$.

On a alors les résultats suivants :

PROPOSITION 7.4.2 *Tout automate de Muller est équivalent à un automate de Muller où tous les états sont accessibles et où tout ensemble d'états est admissible.*

PROPOSITION 7.4.3 *Tout automate de Muller est équivalent à un automate de Muller complet, c'est-à-dire dans lequel la fonction de transition est définie partout.*

PROPOSITION 7.4.4 *Soit $M = (Q, \Sigma, \delta, q_0, \mathcal{L})$ un automate de Muller complet. Alors le langage $\Sigma^\omega \backslash L^\omega(M)$ est reconnu par l'automate $M = (Q, \Sigma, \delta, q_0, \mathcal{P}(Q) \backslash \mathcal{L})$.*

PROPOSITION 7.4.5 *Soient* $M_1 = (Q_1, \Sigma, \delta_1, q_1, \mathcal{L}_1)$ *et* $M_2 = (Q_2, \Sigma, \delta_2, q_2, \mathcal{L}_2)$ *deux automates de Muller, alors l'automate de Muller*

$$M = \big(Q_1 \times Q_2, \Sigma, \delta, (q_1, q_2), \mathcal{L}\big)$$

avec

$$\mathcal{L} = \{S \mid S \subseteq Q_1 \times Q_2,\ \pi_1(S) \in \mathcal{L}_1,\ \pi_2(S) \in \mathcal{L}_2\}$$

$$\delta : ((q, q'), a) \longmapsto \big(\delta_1(q, a), \delta_2(q, a)\big)$$

où π_1 *et* π_2 *sont les projections canoniques de* $Q_1 \times Q_2$ *sur* Q_1 *et* Q_2 *respectivement, reconnaît l'intersection* $L^\omega(M_1) \cap L^\omega(M_2)$.

On montre de la même façon que l'ensemble des langages reconnus par automates de Muller est stable par réunion finie, ce qui donne toutes les opérations booléennes.

PROPOSITION 7.4.6 *Pour toute partie* X *de* Σ^ω*, les assertions suivantes sont équivalentes :*

(i) *X est reconnue par un automate de Muller*

(ii) *X est réunion finie de langages de la forme $U \setminus V$ où U et V sont des ω-langages reconnus par des automates de Büchi déterministes*

(iii) *X est une combinaison booléenne finie de langages de Σ^ω reconnus par des automates de Büchi déterministes.*

PREUVE : Dans les démos, on suppose que $|\mathcal{L}| = 1$, soit $\mathcal{L} = \{T\}$

- (i) $\Rightarrow$ (ii)
  Supposons X reconnu par un automate de Muller. Comme précédemment, on peut supposer que cet automate est de la forme $(Q, \Sigma, \delta, q_0, \{T\})$ puis procéder par réunion finie. On peut alors exprimer X sous la forme

  $$\bigcap_{t \in T} L^\omega\big(Q, \Sigma, \delta, q_0, \{t\}\big) \Big\backslash \bigcup_{t \notin T} L^\omega\big(Q, \Sigma, \delta, q_0, \{t\}\big)$$

  où les L^ω s'entendent au sens de Büchi (déterministes), car cette formule exprime qu'un chemin c est acceptant si et seulement si l'ensemble $\mathrm{Inf}(c)$ rencontre chacun des éléments de T et aucun des éléments de $Q \setminus T$.
- (ii) $\Rightarrow$ (iii)
  Cette implication est triviale.
- (iii) $\Rightarrow$ (i)
  Il est facile de voir que tout langage reconnu par un automate de Büchi déterministe est reconnu par un automate de Muller. En effet, si l'ensemble d'acceptation est $T \subseteq Q$ dans le cas de Büchi, il suffit de prendre le même automate avec pour ensemble d'acceptation $\{U \mid U \subseteq Q,\ U \cap T \neq \emptyset\}$ pour obtenir un automate de Muller équivalent. Par suite, comme l'ensemble des langages reconnus par automates de Muller est stable par toutes les opérations booléennes d'après les résultats précédents, l'implication est démontrée.

■

On peut enfin se demander quelle condition est nécessaire pour obtenir une équivalence entre automates de Muller et automates de Büchi déterministe. La condition donnant l'équivalence est en fait naturelle :

PROPOSITION 7.4.7 *Un automate de Muller $(Q, \Sigma, \delta, q_0, \mathcal{L})$ est équivalent à un automate de Büchi déterministe si et seulement si son ensemble d'acceptation $\mathcal{L}$ est plein, c'est-à-dire si pour toutes parties admissibles T et T' se Q telles que $T \subseteq T'$, $T \in \mathcal{L}$ entraîne $T' \in \mathcal{L}$.*

7.5 Automates de Rabin

Comme le modèle des automates de Muller n'a pas fourni une caractérisation des langages ω-rationnels, on cherche donc un nouveau modèle d'automate déterministe. C'est la raison d'être du modèle des automates de Rabin.

DÉFINITION 7.5.1 (AUTOMATE DE RABIN)
Un automate de Rabin est un quintuplet $(Q, \Sigma, \delta, q_0, R)$ où
- Q est l'ensemble fini des états ;
- Σ est l'alphabet ;
- δ est la fonction de transition, de $Q \times \Sigma$ dans Q ;
- $q_0 \in Q$ est l'état initial ;
- R est l'ensemble d'acceptation, de la forme $\{(L_j, U_j) \mid j \in J\}$ $(L_j \subseteq Q, U_j \subseteq Q)$

Un chemin acceptant (partant de q_0) est alors un chemin c tel qu'il existe un élément j de J pour lequel on ait

$$\mathrm{Inf}(c) \cap L_j = \emptyset \qquad \text{et} \qquad \mathrm{Inf}(c) \cap U_j \neq \emptyset$$

REMARQUE : Un automate de Rabin déterministe est un automate de Rabin avec $R = \{(\emptyset, F)\}$.

Ce modèle est visiblement très proche de celui des automates de Muller, et on a en effet la propriété suivante :

PROPOSITION 7.5.1 *Tout automate de Rabin est équivalent à un automate de Muller. De plus, une partie X de Σ^ω est reconnue par un automate de Rabin à n paires si et seulement si elle est une réunion finie de n différences de ω-langages reconnus par automates de Büchi déterministes.*

PREUVE : Soit $M = (Q, \Sigma, \delta, q_0, R)$ un automate de Rabin. L'automate de Muller $(Q, \Sigma, \delta, q_0, \mathcal{L})$ défini par

$$\mathcal{L} = \{T \mid T \subseteq Q,\ \exists (L, U) \in R, T \cap L = \emptyset, T \cap U \neq \emptyset\}$$

reconnaît clairement le langage $L^\omega(M)$.

Soit à présent M' un automate de Muller. D'après une caractérisation précédente, on sait que le langage $L(M')$ est une réunion finie de différence de langages reconnus par des automates de Büchi déterministes. D'autre part, il est clair qu'un chemin est acceptant un automate de Rabin pour le couple (L, U) si et seulement s'il est acceptant pour l'automate de Büchi $(Q, \Sigma, \delta, q_0, U)$ et non acceptant pour l'automate de Büchi $(Q, \Sigma, \delta, q_0, L)$, donc on a bien la réciproque.

La caractérisation du nombre de différences de langages reconnus par automates de Büchi découle de la remarque précédente. *En fait c'est plus compliqué mais j'ai la flemme de le rédiger.* ■

COROLLAIRE 7.5.1 *La classe des langages reconnus par automate de Müller est exactement la classe des langages reconnus par automate de Rabin.*

A ce stade, on a montré les deux résultats :
- Müller $\Rightarrow$ Büchi
- Rabin $\Leftrightarrow$ Müller

Dans la section suivante, on va montrer que Büchi $\Rightarrow$ Rabin, donc en particulier puisque l'on sait prendre le complémentaire d'un automate de Müller, on pourra affirmer :

THÉORÈME 7.5.1 *La classe des langages ω-rationnels est stable par complémentaire.*

7.6 Algorithme de Safra

Le formalisme des automates de Rabin caractérise en fait la classe des langages ω-rationnels. L'algorithme de Safra constitue une preuve constructive de cette caractérisation, en construisant à partir d'un automate de Büchi (non déterministe) un automate de Rabin équivalent.

On considère donc initialement un automate de Büchi non déterministe $M = (Q, \Sigma, \delta, I, F)$, et on construit un automate de Rabin $\mathcal{D}$ reconnaissant le même langage. Notons

$$Q = \{1, 2, \dots, n\}$$

et posons $V = \{1, 2, \dots, 2n\}$.

7.6.1 États de $\mathcal{D}$

Les états de l'automate $\mathcal{D}$ sont des quadruplets (T, f, e, M) où
- T est un arbre dont l'ensemble des nœuds est une partie de V ;
- f est une application de T dans T^* ; f ordonne l'arbre. Le mot $f(v)$ est le mot formé par les fils du nœud v.
- e est une application qui à chaque nœud v de T associe une partie non vide de Q ;
- M est un marquage de T, c'est-à-dire une partie de l'ensemble des nœuds de T.

On impose de plus à l'arbre T de satisfaire les contraintes suivantes :
- la racine de T est 1 ;
- les nœuds marqués sont des feuilles ;
- pour tout nœud v, on a l'inclusion stricte

$$\bigcup_{w \text{ fils de } v} e(w) \subsetneq e(v)$$

- si pour deux nœuds v et v', v n'est par ancêtre de v' et v' n'est pas ancêtre de v, alors $e(v)$ et $e(v')$ sont disjoints.

FAIT 7.6.1 *Dans un tel arbre, le nombre de nœuds est au plus n.*

PREUVE : Posons r la fonction qui à un nœud donné associé l'ensemble des éléments qui appartiennent à son image par e et à celle d'aucun autre nœud, c'est-à-dire

$$v \mapsto r(v) = e(v) \Big\backslash \bigcup_{w \text{ fils de } v} e(w)$$

Il est facile de voir que cette définition rend bien tous les $r(v)$ disjoints. D'autre part, les hypothèses imposent que $e(v)$ contienne toujours strictement plus d'éléments que les images des fils de v réunies, donc que les $r(v)$ contiennent tous au moins un élément. On a donc

$$|T| \quad = \quad \sum_{v \in T} 1 \quad \leq \quad \sum_{v \in T} \big|r(v)\big| \quad \leq \quad |Q| \quad = \quad n$$

car les images des nœuds de T par r sont des parties de Q disjointes, ce qui prouve bien que T a au plus n nœuds. ■

7.6.2 Fonction de transition

La fonction de transition Δ de $\mathcal{D}$ sera définie par un calcul en plusieurs étapes. Considérons le calcul de $\Delta\big((T, f, e, M), a)\big)$:

1. on applique la fonction de transition δ de M à toutes les étiquettes de T avec a et on supprime toutes les marques pour obtenir un état (T, f, e_1, M_1) :

   $$e_1(v) = \delta\big(e(v), a\big) \qquad M_1 = \emptyset$$

2. à chaque nœud v on ajoute un fils placé à droite des autres, étiqueté par $e_1(v) \cap F$ et marqué, avec un nom pris dans $V \setminus T$, ce qui définit T_2, e_2 et M_2. Pour définir f_2, posons $\bar{v}$ le nom du fils qui a été ajouté à v. On a donc $\bar{v} \in T$ moins ce qui a déjà été affecté (parmi les possibles, on choisit le minimum), pour tout v de T. On définit alors f_2 par

   $$\forall v \in T,\ f_2(v) = f(v)\bar{v} \qquad \forall v \in \bar{T},\ f_2(v) = \varepsilon$$

   L'image d'un nœud v par f (et donc f_2) se comprend donc comme la suite des fils de v.

3. pour chaque nœud v, on supprime dans l'étiquette $e_2(v)$ les éléments appartenant à l'étiquette de l'un des noeuds de T_2 situés à gauche de v, c'est-à-dire

   $$e_3(v) = e_2(v) \Big\backslash \bigcup_{w \text{ à gauche de } v} e_2(w)$$

   T_2, f_2 et M_2 restent inchangés.

4. On définit T_4 en supprimant les nœuds de T_3 ayant une étiquette vide, et on restreint f_3, e_3 et M_3 au nouvel arbre.

5. Pour chaque nœud v de T_4, si on a

   $$e_4(v) = \bigcup_{w \text{ descendant de } v} e_4(w)$$

   alors on marque v et on supprime tous les descendants de v. L'iéde est que l'on supprime tous les fils car l'information est toute contenu dans le seul nœud père.

On définit donc $\Delta(T, f, e, M)$ comme le quadruplet (T_5, f_5, e_5, M_5) obtenu après le calcul précédent.

7.6.3 État initial

L'état initial de $\mathcal{D}$ est défini en fonction de I et F :
- si I et F sont disjoints, il est réduit à un nœud non marqué et étiqueté par I ;
- si I est inclus dans F, il est réduit à un nœud marqué et étiqueté par I ;
- sinon il contient un nœud non marqué ayant un fils marqué et étiqueté par $I \cap F$.

7.6.4 Ensemble d'acceptation

L'ensemble d'acceptation $\mathcal{R}$ est défini par

$$\mathcal{R} = \{(L_v, U_v) \mid v \in V\} \quad \text{avec} \quad \begin{array}{l} L_v = \{\mathcal{R} \mid v \text{ n'est pas un nœud de } \mathcal{R}\} \\ U_v = \{\mathcal{R} \mid v \text{ est un nœud marqué de } \mathcal{R}\} \end{array}$$

REMARQUE : Les $\mathcal{R}$ sont les états de l'automate de Rabin construit.

7.6.5 Théorème

THÉORÈME 7.6.1 *Ces deux automates reconnaissent le même langage.*

PREUVE : Cette démonstration est trop longue mais il faut retenir qu'elle utilise le lemme de König dans un sens. ■