Chapitre 10

Grammaires algébriques

Le lemme de l'étoile montre que de nombreux langages ne sont pas reconnaissables par automates finis en particulier dès que ces langages utilisent un parenthésage bien formé. Il est donc nécessaire de dépasser le formalisme des expressions régulières pour analyser des langages plus complexes, par exemple pour analyser les expressions arithmétiques, pour compiler des langages de programmation, pour faire l'analyse syntaxique de certaines structures biologiques (génomes) ... Afin d'étudier les structures grammaticales des langues naturels, le linguiste Noam Chomsky a dégagé la notion de grammaire formelle et a proposé une classification de ces grammaires. Cette théorie s'est avérée ensuite très utile en informatique pour la conception de compilateur. En particulier la théorie des grammaires dites algébriques (ou hors-coontextes) qui engendrent exactement les langages reconnus par les automates à pile, y est centrale.

10.1. Grammaires formelles

10.1.1 Définition. Une *grammaire formelle* est un système de réécriture $\mathcal{G} = (\Sigma, V, S, R)$ tel que :
- Σ est un ensemble fini de *terminaux* (l'alphabet du langage) ;
- V est un ensemble fini de *variables* disjoint de Σ ;
- S un symbole distingué de V appelé *axiome* ;
- R est une partie finie $(\Sigma \cup V)^* V (\Sigma \cup V)^* \times (\Sigma \cup V)^*$ appelé ensemble des *productions* de la grammaire.

En pratique on notera une production $(x, y) \in R$ sous la forme $x \xrightarrow{\mathcal{G}} y$ (ou encore $x \to y$ lorsque la grammaire est implicite). De plus quand $\mathcal{G}$ contient des productions partant du même mot x, on utilisera la notaion plus concise $x \to y_1|y_2|y_3|...$ pour $x \to y_1$, $x \to y_2$, $x \to y_3$ ...

10.1.2. Dérivations. On appelle $\mathcal{G}$*-réécriture* tout couple (u, v) de mots de $(\Sigma \cup V)^*$, tel qu'il existe une production $x \to y$ de $\mathcal{G}$ et des mots w_1 et w_2 vérifiant $u = w_1 x w_2$ et $v = w_1 y w_2$. On note également une réécriture sous la forme $u \xrightarrow{\mathcal{G}} v$ ou simplement $u \to v$.

On appelle *dérivation* de $\mathcal{G}$ toute suite de réécritures

$$u \to u_1 \to u_2 \to \ldots \to u_n \to v.$$

Une dérivation est notée $u \xrightarrow{*} v$. Un mot w de $(\Sigma \cup V)^*$ est dit $\mathcal{G}$*-dérivable* à partir de u si on a une dérivation $u \xrightarrow{*} w$. On dit que w est $\mathcal{G}$-dérivable à partir de u en n-étapes s'il existe une dérivation $u \xrightarrow{*} w$ constituée de n réécritures.

On appelle langage engendré par $\mathcal{G}$, noté $L(\mathcal{G})$, le langage

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

10.1.3. Exemple. On considère les expressions arithmétiques sur l'alphabet

$$\Sigma = \{a, b, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -, *, (,)\}.$$

Voici un exemple d'expression arithmétique sur cet alphabet sous forme d'un code informatique :

```
(a+b)*(b-32*a*(a-b)).
```

Afin d'engendrer ces expressions, on peut considérer la grammaire définie de la manière suivante :
- Les terminaux sont les éléments de Σ ;
- L'ensemble des variables $V = \{E, N\}$ (E pour expression, N pour nombre) ;
- L'axiome est E ;
- Les productions sont :
  1. $E \to E * E|(E + E)|(E - E)$,
  2. $E \to N|a|b$,
  3. $N \to cN|c$ pour c parcourant les chiffres $0, 1, ..., 9$.

Voici une dérivation engendrant le mot $(a + b) * (b - 32 * a * (a - b))$:

$$E \to E * E \to (E + E) * E \to (a + E) * E \to (a + E) * (E - E) \to (a + b) * (E - E)$$
$$\to (a + b) * (E - E * E) \to (a + b) * (E - E * E * E) \to (a + b) * (E - N * E * E)$$
$$\to (a + b) * (E - N * E * (E - E)) \to (a + b) * (E - 3N * E * (E - E))$$
$$\to (a + b) * (E - 32 * E * (E - E)) \to (a + b) * (b - 32 * E * (E - E)) \to$$
$$(a + b) * (b - 32 * a * (E - E)) \to (a + b) * (b - 32 * a * (a - E)) \to (a + b) * (b - 32 * a * (a - b))$$

On remarque facilement sur cet exemple que plusieurs dérivations sont possibles pour engendrer le même mot.

10.1.4. Classification de Chomsky.
- **Grammaires de type 0** : toutes les grammaires formelles sans aucune restriction sur les règles de récritures. Les langages engendrés correspondent aux langages récursivement énumérables (ou semi-décidables, c'est-à-dire les langages reconnaissables par une machine de Turing, cf chapitre 14).
- **Grammaires de type 1** : les *grammaires contextuelles*. Une grammaire est contextuelle si toutes ses productions $u \to v$ vérifient $|v| \geq |u|$. Les langages engendrés correspondent aux langages reconnus par des machine de Turing totalement bornées.
- **Grammaires de type 2** : les *grammaires algébriques* (ou *hors-contextes*). Une grammaire est algébrique si toutes ses productions sont du type

  $$T \to u \text{ avec } T \in V.$$

  On appelle *langage algébriques* tout langage qui est engendré par une grammaire algébrique. Les langages algébriques correspondent aux langages reconnus par les automates à piles (voir plus loin dans ce chapitre).
  Exemple : La grammaire avec pour alphabet $\Sigma = \{a, b\}$, pour seule variable S et pour productions $S \to aSb|\varepsilon$. Le langage engendré est $\{a^n b^n : n \in \mathbb{N}\}$.

– **Grammaires de type 3** : les *grammaires linéaires à gauche (respectivement à droite)*. Une grammaire est linéaire à gauche (respectivement à droite) si toutes ses productions sont du type

$$T_1 \to u \text{ ou } T_1 \to uT_2 \text{ (respectivement } T_1 \to T_2u) \text{ avec } T_1, T_2 \in V \text{ et } u \in \Sigma^*$$

Les langages engendrés correspondent aux langages rationnels (voir section 10.2).
Exemple : La grammaire avec pour alphabet $\Sigma = \{a\}$, pour seule variable S et pour productions $S \to aaS|\varepsilon$. Le langage engendré est $\{a^{2n} : n \in \mathbb{N}\}$.

10.2. GRAMMAIRES LINÉAIRES ET LANGAGES RATIONNELS

10.2.1 Proposition. *Tout langage rationnel est engendré par une grammaire linéaire à gauche.*

Démonstration. Soit L un langage rationnel. Soit $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ un automate le reconnaissant. On considère alors la grammaire $\mathcal{G} = (\Sigma, Q, R, q_o)$ où R est l'ensemble des productions de la forme $p \to xq$ pour chaque transition $p \xrightarrow{x} q$ de $\mathcal{A}$, augmenté des productions $p \to \varepsilon$ pour tout état final p. On montre alors par récurrence sur n que les mots $\mathcal{G}$-dérivables en n-étapes à partir de q_0 sont :
- les mots wq où w est de longueur n et $q_0 \xrightarrow{w} q$, et
- les mots w de longueur $n-1$ acceptés par $\mathcal{A}$.

On en déduit que $L = L(\mathcal{G})$. □

10.2.2 Proposition. *Tout langage engendré par une grammaire linéaire à gauche est rationnel.*

Démonstration. Soit L un langage engendré par une grammaire linéaire à gauche $\mathcal{G} = (\Sigma, V, R, S)$. On construit alors un automate fini $\mathcal{A}$ (non nécessairement déterministe) sur l'alphabet Σ qui va reconnaître L. L'ensemble des états de $\mathcal{A}$ est construit à partir de l'ensemble V et complété de la façon suivante :
- On ajoute un état q_f qui sera l'unique état final de $\mathcal{A}$;
- Pour toute production $T_1 \to a_1....a_nT_2$ on ajoute $n-1$ états $p_1, ..., p_{n-1}$ et on ajoute à $\mathcal{A}$ le chemin de transitions
$$T_1 \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2 \cdots p_{n-1} \xrightarrow{a_n} T_2 ;$$
- Pour toute production $T_1 \to a_1....a_n$ on ajoute $n-1$ états $p_1, ..., p_{n-1}$ et on ajoute à $\mathcal{A}$ le chemin de transitions
$$T_1 \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2 \cdots p_{n-1} \xrightarrow{a_n} p_f .$$

Enfin $\mathcal{A}$ a pour unique état initial S.

On vérifie alors facilement que $L(\mathcal{A}) = L(\mathcal{G})$.

□

10.2.3. Equations linéaires. Soit $\mathcal{G}$ une grammaire linéaire à gauche. Pour toute variable $T \in V$, considérons le langage L_T consistant en les mots de Σ^* dérivant de T. On remarque que si la liste des productions de $\mathcal{G}$ partant de T est de la forme

$$T \to u_1T_1|u_2T_2|...|u_nT_n|v_1|...|v_m$$

alors l'équation suivante est vérifiée :

$$L_T = \{u_1\}L_{T_1} \cup \{u_2\}L_{T_2} \cup \ldots \cup \{u_n\}L_{T_n} \cup \{v_1, ..., v_m\}.$$

On note souvent ce type d'équation sous la forme

$$L_T = u_1L_{T_1} + u_2L_{T_2} + ... + u_nL_{T_n} + v_1 + ... + v_m.$$

On peut donc associer à $\mathcal{G}$ un système d'équations linéaires à gauche. Chacune des équations de ce système provient de la liste de productions partant d'un variable et est de la forme

$$T = T_1 + u_2T_2 + ... + u_nT_n + v_1 + ... + v_m.$$

où T et les T_i sont des variables, et les u_i, v_j des scalaires (mots de terminaux). On peut alors vérifier que la famille des langages L_T est la solution minimale (pour l'inclusion des langages) de ce système d'équations.

De manière analogue, on peut associer à une grammaire linéaire à droite un système d'équations linéaires à droite et à une grammaire algébrique un système d'équations algébriques.

10.3. Arbres de dérivation d'une grammaire algébrique

Fixons pour cette section une grammaire algébrique $\mathcal{G}$.

10.3.1 Définition. On appelle *arbre de dérivation* de $\mathcal{G}$ un arbre étiqueté ayant les propriétés suivantes :

1. chaque noeud est étiqueté par une variable, un terminal ou le mot vide ;
2. un noeud interne est étiqueté par une variable ; de plus si T est cette variable et u est le mot constitué par la suite des étiquettes (lues de gauche à droite) des successeurs du noeud considéré, alors $T \to u$ est une production de $\mathcal{G}$.
3. Si un noeud est étiquetté par le mot vide alors le prédecesseur de ce noeud n'a pas d'autre successeur. (En particulier dans ce cas, si T est l'étiquette du prédesseur alors $T \to \varepsilon$ est une production de $\mathcal{G}$).

10.3.2. Exemple. L'arbre suivant est un arbre de dérivation de la grammaire de l'exemple 10.1.3.

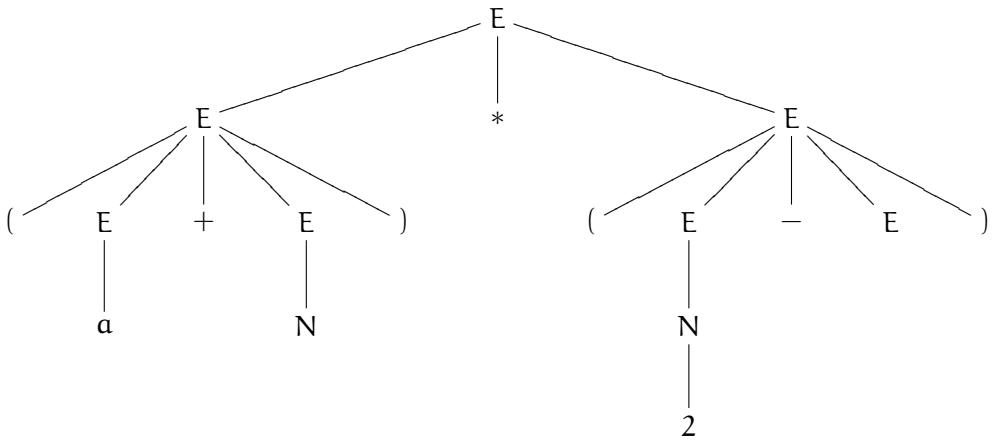

Fig. 10.1 – Un arbre de dérivation

A tout arbre de dérivation, on associe le mot formé des étiquettes de ses feuilles lues de gauche à droite. Le mot associé à l'exemple ci-dessus est $(a + N) * (2 - E)$.

On vérifie facilement par récurrence la proposition suivante :

10.3.3 Proposition. *Pour toute variable* T *de* $\mathcal{G}$ *et tout mot* u *de variables et terminaux,* u *est dérivable de* T *si et seulement si* u *est associé à un arbre de dérivation de racine* T.

Par conséquent le langage $L(\mathcal{G})$ correspond à l'ensemble des mots associés aux arbres de dérivations dont la racine est étiquétée par l'axiome S et les feuilles par des terminaux. Etant donné un tel arbre T , le mot associé $u \in L(\mathcal{G})$ est en général dérivable de S de plusieurs manières. On appelle *dérivation gauche* de u, l'unique dérivation obtenue à partir de T en effectuant à chaque étape la réécriture la plus à gauche dans l'arbre. On obtient ainsi une dérivation formée d'une suite de réécritures où l'on réécrit systématiquement la variable la plus à gauche. De manière analogue, on définit la notion de *dérivation droite*.

Remarque. Les grammaires algébriques s'appellent également grammaires hors-contextes car pour chaque noeud d'un arbre de dérivation, les sous-arbres de racines les successeurs de ce noeud sont indépendants les uns des autres.

10.3.4 Définition. La grammaire $\mathcal{G}$ est dite *ambiguë* s'il existe un mot $u \in L(\mathcal{G})$ qui est associé à deux arbres de dérivations disctincts de racine S.

10.3.5. Exemple. On considère la grammaire qui a pour terminaux les caractères $a, b, c, +, *$, pour unique variable S (qui est également l'axiome) et pour productions

$$S \to S + S|S * S|a|b|c.$$

Alors le mot $a + b * c$ est associé à deux arbres dérivations :

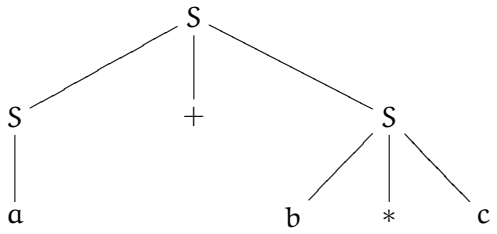

FIG. 10.2 – Premier arbre de dérivation pour $a + b * c$

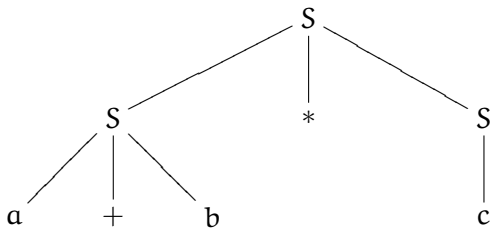

FIG. 10.3 – Second arbre de dérivation pour $a + b * c$

L'ambiguïté pose problème pour l'interprétation, en particulier dans le cas des langages de programmation. En effet un programme doit être interprété et executé d'une seule façon. Nous verrons plus loin des sous-classes de langages algébriques non ambiguës.

10.4. GRAMMAIRES ALGÉBRIQUES ET AUTOMATES À PILE NON DÉTERMINISTES

10.4.1 Proposition. *Tout langage algébrique est reconnaissable par un automate à pile.*

Démonstration. Soit $\mathcal{G} = (\Sigma, V, S, R)$ une grammaire algébrique. Considérons l'automate à pile $\mathcal{A} = (\{q_0\}, \Sigma, \Pi, S, \delta, \tau, q_0, \emptyset)$ défini de la manière suivante :

- $\mathcal{A}$ n'a qu'un seul état q_0 qui est l'état initial ;
- l'alphabet de pile $\Pi = \Sigma \cup S$;
- $\mathcal{A}$ a pour transitions non instantannées, les transitions $(q_0, a) \xrightarrow{a} (q_0, \varepsilon)$ pour toute lettre $a \in \Sigma$;
- $\mathcal{A}$ a pour transitions instantanées, les transitions $(q_0, T) \xrightarrow{\varepsilon} (q_0, u)$ pour toute production $T \to u$ de $\mathcal{G}$.

Nous allons montrer que cet automate reconnaît $L(\mathcal{G})$ par pile vide. Vérifions pour cela que si $\mathcal{A}$ arrive à une configuration (q_0, w_2, u_2) à partir de la configuration (q_0, w_1w_2, u_1) alors le mot w_1u_2 est $\mathcal{G}$ dérivable à partir de u_1 . C'est-à-dire,

$$\text{si } (q_0, w_1w_2, u_1) \overset{*}{\longmapsto} (q_0, w_2, u_2) \text{ alors } u_1 \xrightarrow{\mathcal{G},*} w_1u_2 .$$

Pour cela vérifions le dans la cas d'un chemin constitué d'une seule transition

$$(q_0, w_1w_2, u_1) \longmapsto (q_0, w_2, u_2) .$$

Dans ce cas

- ou bien il s'agit d'une transition par une lettre $a \in \Sigma$ et alors l'automate dépile a, c'est-à-dire $w_1 = a$ et $u_1 = au_2$, d'où $w_1u_2 = u_1$;
- ou bien il s'agit d'une transition instantanée et dans ce cas l'automate dépile une variable T et empile u tel que $T \to u$ est une production de $\mathcal{G}$. Ainsi, on a $w_1 = \varepsilon$ et $u_1 = Tu_3 \to uu_3 = u_2$.

On obtient alors le résultat par récurrence sur la longueur du chemin. En effet, si le chemin

$$(q_0, w_1w_2w_3, u_1) \overset{*}{\longmapsto} (q_0, w_3, u_3)$$

se décompose en

$$(q_0, w_1w_2w_3, u_1) \longmapsto (q_0, w_2w_3, u_2) \overset{*}{\longmapsto} (q_0, w_3, u_3)$$

tel que $u_1 \xrightarrow{\mathcal{G}} w_1u_2$ et $u_2 \xrightarrow{\mathcal{G},*} w_2u_3$, alors on a $u_1 \xrightarrow{\mathcal{G},*} w_1w_2u_3$.

On en déduit que si w est un mot reconnu par pile vide (i.e. $(q_0, w, S) \overset{*}{\longmapsto} (q_0, \varepsilon, \varepsilon)$) alors w est $\mathcal{G}$-dérivable à partir de S. Donc $L(\mathcal{A}) \subset L(\mathcal{G})$.

Réciproquement, considérons un mot $w \in L(\mathcal{G})$. Alors il existe une dérivation gauche $u_0 = S \to u_1 \to u_2 \to \ldots \to u_m = w$. Pour chaque $i \geq m$, on a $u_i = v_iT_iu_i'$ tel que la première variable de u_i est T_i et $w = v_iw_i'$. Alors dans l'automate $\mathcal{A}$ on a pour chaque $i < m$, un chemin de la configuration (q_0, w_i', T_iu_i') à la configuration $(q_0, w_{i+1}', T_{i+1}u_{i+1}')$. Il suffit pour cela de remarquer que l'on doit avoir une réécriture du type $u_i = v_iT_iu_i' \to v_iv_i'T_{i+1}u_{i+1}' = u_{i+1}$ avec $w_i' = v_iw_{i+1}'$. D'où le passage de la configuration (q_0, w_i', T_iu_i') à la configuration $(q_0, w_i', v_iv_i'T_{i+1}u_{i+1}')$ puis, par dépilement, le passage de la configuration $(q_0, v_iw_{i+1}', v_iv_i'T_{i+1}u_{i+1}')$ à la configuration $(q_0, w_{i+1}', v_i'T_{i+1}u_{i+1}')$. Il suit que w est reconnu par $\mathcal{A}$ par pile vide.. □

10.4.2 Proposition. *Tout langage reconnu par un automate à pile est algébrique.*

Démonstration. Soit L un langage reconnu par un automate $\mathcal{A}$ par pile vide. L'objectif est d'associer une grammaire algébrique dont les suites de dérivations gauche correspondent aux chemins dans $\mathcal{A}$. Dans le cas d'un langage régulier, on a associé une grammaire dont les variables étaient les états de l'automate. Ici pour suivre le chemin on aura de plus besoin de considérer comme paramètre le haut de pile. En fait il faudra également ajouter l'état cible du chemin envisagé.

Pour construire cette grammaire il est utile d'avoir des hypothèses supplémentaires sur $\mathcal{A}$. En reprenant la construction de la proposition 8.5.2, on peut supposer que $\mathcal{A}$ a un seul état initial q_0 et a un seul état q_ε où la pile peut être vide. On remarque de plus que par cette construction, l'automate n'arrive en q_ε qu'avec une pile vide et que le symbole de fond de pile S reste en fond de pile dans tous les états intermédiaires et ne se trouve qu'en fond de pile. Afin de simplifier les productions, on peut supposer en rajoutant des états intermédiaires pour chaque transtion, que $\mathcal{A}$ ne possède que des transitions rudimentaires de l'un des trois types suivants :

1. Transition ne modifiant pas la pile $(p,x) \xrightarrow{a} (q,x)$ (où $a \in \Sigma \cup \{\varepsilon\}$).
2. Empilement strict d'un symbole $(p,x) \xrightarrow{\varepsilon} (q,yx)$ (où $x,y \in \Pi$ et $y \neq S$).
3. Dépilement strict $(p,x) \xrightarrow{\varepsilon} (q,\varepsilon)$.

On considère une grammaire $\mathcal{G}$ dont les variables sont les triplets (p,x,q) où p et q sont des états et x un symbole de pile. Pour une variable (p,x,q) les paramètres p et x représenteront l'état courant et le symbole en haut de la pile courante. Le paramètre q est une balise pour garder en mémoire l'objectif à atteindre ultérieurement. On fournit donc à $\mathcal{G}$ comme axiome le triplet (q_0,S,q_ε).

Maintenant pour chaque type de transition, on met les productions suvantes dans $\mathcal{G}$:

1. À une transition $(p,x) \xrightarrow{a} (q,x)$ est associée pour chaque $r \in Q$ la production
$$(p,x,r) \to a(q,x,r).$$
Ici la balise visée reste inchangée.
2. À une transition $(p,x) \xrightarrow{\varepsilon} (q,yx)$ est associée pour chaque $r \in Q$ la liste des productions
$$(p,x,r) \to (q,y,r')(r',x,r)$$
pour tout $r' \in Q$. Une nouvelle balise r' doit être atteinte avant r..
3. À une transition $(p,x) \xrightarrow{\varepsilon} (q,\varepsilon)$ est associée la production
$$(p,x,q) \to \varepsilon.$$
La balise est atteinte et donc supprimée.

Notons que quand on fait un empilement strict par un symbole y on ne sait pas dans quel état se trouvera l'automate quand il dépilera cet y. C'est pour cela que l'on met dans les productions toutes les balises intermédiaires possibles.

Par récurrence sur n (laissée au lecteur), il est facile de vérifier qu'un mot u non terminal qui dérive de l'axiome par une dérivation gauche constituée de n réécritures est de la forme
$$u = w(q_1,x_1,q_2)(q_2,x_2,q_3)...(q_{m-1},x_{m-1},q_m)(q_m,S,q_\varepsilon)$$
où $w \in \Sigma^*$. On peut de plus vérifier que cette dérivation correspond à un chemin de n-transitions dans $\mathcal{A}$ étiqueté par w et allant de l'état q_0 avec la pile intiale S à l'état q_1 avec pour pile $x_1x_2....x_{m-1}S$. Il suit que tout mot terminal produit par $\mathcal{G}$ est reconnu par pile vide par $\mathcal{A}$.

Réciproquement, on peut vérifier par récurrence sur la longueur des chemins de $\mathcal{A}$ (laissée au lecteur), que pour tout chemin de $\mathcal{A}$ étiqueté par w et allant de l'état q_0 avec la pile intiale S à l'état q_1 avec pour pile $x_1x_2....x_{m-1}S$ (pile de taille m) alors, pour tout $q_2, ..., q_m$, il existe une dérivation gauche conduisant à partir de l'axiome au mot $w(q_1, x_1, q_2)(q_2, x_2, q_3)...(q_{m-1}, x_{m-1}, q_m)(q_m, S, \varepsilon)$. Il suit que tout mot reconnu par pile vide par $\mathcal{A}$ est dans $L(\mathcal{G})$.

□

10.5. DÉCIDABILITÉ D'UN LANGAGE ALGÉBRIQUE

Dans un automate à pile, il peut exister des chemins infinis de transitions instantanées. Il n'est donc pas évident qu'un langage reconnu par automate à pile est décidable par un algorithme. De manière analogue, si une grammaire algébrique a des ε-productions ou productions vides ($T \to \varepsilon$, alors un mot fixé pourrait-être dérivable de l'axiome par une dérivation de longueur arbitraire et donc il pourrait y avoir une infinité de dérivations à tester.

10.5.1 Proposition. *Soit* L *un langage algébrique alors* $L \setminus \{\varepsilon\}$ *peut être engendré par une grammaire algébrique sans* ε*-production.*

Démonstration. Soit $\mathcal{G}$ une grammaire engendrant L. On commence par déterminer l'ensemble des variables effaçables, c'est-à-dire l'ensemble

$$V_\varepsilon = \{T \in V : \ T \xrightarrow{*} \varepsilon \ \}.$$

Ceci peut se faire par un algorithme récursif. On forme alors la grammaire $\mathcal{G}'$ à partir de $\mathcal{G}$ en supprimant toutes les ε-productions et en ajoutant, pour chaque production $T \to u$ de $\mathcal{G}$ toutes les productions $X \to u'$ om u' est un mot non vide obtenu à partir de u en supprimant zéro, une ou plusieurs occurrences des variables effaçables dans u.

Il est évident alors que $L(\mathcal{G}') \subset L(\mathcal{G})$ (il suffit de remarquer que toute production de $\mathcal{G}'$ correspond à une dérivation de $\mathcal{G}$). Réciproquement, le fait que $L(\mathcal{G}) \setminus \{\varepsilon\} \subset L(\mathcal{G}')$ se montre par récurrence sur la longueur des dérivations. □

Remarquons qu'une grammaire algébrique est sans ε-productions si et seulement si elle est de type 1.

10.5.2 Proposition. *Tout langage engendré par une grammaire de type 1 est décidable.*

Démonstration. Une grammaire $\mathcal{G}$ de type 1 n'a que des productions $u \to v$ vérifiant $|v| \geq |u|$. Remarquons tout d'abord qu'une dérivation sur une telle grammaire est constituée d'une suite de réécritures croissantes. Notons M le cardinal de $\Sigma \cup V$. On remarque si u_2 est dérivable de u_1 avec $|u_2| = |u_1| = n$ alors il existe une telle dérivation de longueur au plus M^n (le nombre de mots longueurs n sur $\Sigma \cup V$). Donc pour tout mot de $w \in L(\mathcal{G})$, le mot w est dérivable de l'axiome par une dérivation de longueur au plus $M + M^2 + ... + M^n$ où n est la longueur de w. Maintenant, pour chaque réécriture d'un mot de longueur n il y a au plus C^{n^2} choix possibles où C est le nombre de productions de $\mathcal{G}$. Donc le nombre de dérivations de longueur l de S à un mot de longueur n est alors bornée par $C^{n^2 l}$. Par conséquent, si w est un mot de longueur n, on peut tester son appartenance à $L(\mathcal{G})$ en énumérant toutes les dérivations de longueurs inférieures à $M + M^2 + ... + M^n = \dfrac{M^{n+1} - M}{M - 1}$ qui sont en nombre fini. On remarque que le nombre de test est ici doublement exponentiel en n. □

10.5.3 Corollaire. *Tout langage algébrique est décidable.*

Remarque. Pour un langage algébrique L on peut assez facilement utiliser un algorithme de complexité bien inférieure à celui de la proposition 10.5.2. Pour cela, on peut montrer facilement qu'il existe une grammaire $\mathcal{G}$ engendrant $L \setminus \{\varepsilon\}$ qui ne possède ni production vide, ni production du type $T_1 \to T_2$ où T_1 et T_2 sont des variables. Avec un telle grammaire, on montre facilement que tout mot $w \in L(\mathcal{G})$ est dérivable de l'axiome par une dérivation de longueur au plus $2n - 1$ où $n = |w|$. Alors pour tester l'appartenance d'un mot w de longueur n, il suffit d'énumérer l'ensemble des dérivations gauche de longueur au plus $2n - 1$, ce qui nous donne une complexité de l'ordre de C^{2n} où C est le nombre de productions de la grammaire.

Une analyse plus fine des langages algébriques par passage à une grammaire sous *forme normale de Chomsky* et l'utilisation de la *programmation dynamique*, permettent d'obtenir un algorithme qui teste l'appartenance d'un mot de longueur n à un langage algébrique en temps $O(n^3)$. Par conséquent dans le cas d'un langage algébrique non ambigüe, il existe un algorithme permettant d'obtenir l'arbre de dérivation d'un mot (et donc son analyse syntaxique) en un temps de l'ordre de n^3. Dans le chapitre suivant, nous verrons des classes de langages algébriques pour lesquelles l'analyse syntaxique est possible en temps linéaire.

10.6. OPÉRATIONS ENSEMBLISTES

Dans cette partie, nous vérifions que la classe des langages algébriques est closes par certaines opérations ensemblistes (union, produit et opération étoile)

10.6.1 Proposition. *Soient $\mathcal{G}_1 = (\Sigma, V_1, S_1, R_1)$ et $\mathcal{G}_2 = (\Sigma, V_2, S_2, R_2)$ deux grammaires telles que V_1 et V_2 sont disjoints. Alors la grammaire $\mathcal{G} = (\Sigma, V_1 \cup V_2 \cup \{S\}, S, R_1 \cup R_2 \cup \{S \to S_1 | S_2\}$ engendre le langage $L(\mathcal{G}_1) \cup L(\mathcal{G}_2)$. De plus si $\mathcal{G}_1$ et G_2 sont algébriques alors $\mathcal{G}$ est également algébrique.*

Démonstration. Soit $w \in \Sigma^*$. Supposons que $S_1 \xrightarrow{\mathcal{G}_1,*} w$. On remarque que toute $\mathcal{G}_1$-dérivation est également une $\mathcal{G}$-dérivation. On a donc $S_1 \xrightarrow{\mathcal{G},*} w$ .et comme $S \to S_1$ est une production de $\mathcal{G}$, on obtient $S \xrightarrow{\mathcal{G},*} w$. De même si $S_2 \xrightarrow{\mathcal{G}_2,*} w$ alors $S \xrightarrow{\mathcal{G},*} w$.

Inversement, supposons que $S \xrightarrow{\mathcal{G},*} w$. Comme tout $\mathcal{G}$-dérivation à partir de S commence par $S \xrightarrow{\mathcal{G}} S_1$ ou $S \xrightarrow{\mathcal{G}} S_2$, on a nécessairement $S_1 \xrightarrow{\mathcal{G},*} w$ ou $S_2 \xrightarrow{\mathcal{G},*} w$. Dans le premier cas, une $\mathcal{G}$-dérivation de w à partir de S_1 n'utilise que des variables de $\mathcal{G}_1$ et des productions de $\mathcal{G}_1$, c'est donc également une $\mathcal{G}_1$-dérivation. De même dans le second cas avec $\mathcal{G}_2$. Il suit que l'on a $S_1 \xrightarrow{\mathcal{G}_1,*} w$ ou $S_2 \xrightarrow{\mathcal{G}_2,*} w$ et donc $w \in L(\mathcal{G}_1) \cup L(\mathcal{G}_2)$.

□

10.6.2 Proposition. *Soient $\mathcal{G}_1 = (\Sigma, V_1, S_1, R_1)$ et $\mathcal{G}_2 = (\Sigma, V_2, S_2, R_2)$ deux grammaires telles que V_1 et V_2 sont disjoints. Alors la grammaire $\mathcal{G} = (\Sigma, V_1 \cup V_2 \cup \{S\}, S, R_1 \cup R_2 \cup \{S \to S_1 S_2\}$ engendre le langage $L(\mathcal{G}_1)L(\mathcal{G}_2)$. De plus si $\mathcal{G}_1$ et G_2 sont algébriques alors $\mathcal{G}$ est également algébrique.*

Démonstration. Démonstration laissée en exercice. □

10.6.3 Proposition. *Soient $\mathcal{G} = (\Sigma, V, S, R)$ une grammaire. Alors la grammaire $\mathcal{G}' = (\Sigma, V \cup \{S'\}, S', R \cup \{S' \to SS' | \varepsilon\}$ engendre le langage $L(\mathcal{G})^*$. De plus si $\mathcal{G}$ est algébrique alors $\mathcal{G}'$ est également algébrique.*

Démonstration. Démonstration laissée en exercice. □

Nous allons voir dans la section suivante que l'intersection de deux langages algébriques n'est pas nécessairement algébrique. Par contre, la classe des langages algébriques est stable par intersection avec un langage rationnel.

10.6.4 Proposition. *L'intersection d'un langage algébrique L_1 avec un langage rationnel L_2 est algébrique.*

Démonstration. La preuve est analogue à celle de l'intersection de deux langages rationnels (voir 2.3.14). Considérons un automate à pile $\mathcal{A}_1$ reconnaissant L_1 par état final et un automate fini déterministe $\mathcal{A}_2$ reconnaissant L_2. On construit alors un automate à pile $\mathcal{A}$ qui va simuler en même temps $\mathcal{A}_1$ et $\mathcal{A}_2$.

Ses états sont des couples d'états formés d'un état de $\mathcal{A}_1$ et d'un état de $\mathcal{A}_2$. Un état de $\mathcal{A}$ est initial (respectivement final) si c'est le couple d'un état initial (respectivement final) de $\mathcal{A}_1$ avec l'état initial (respectivement un état final) de $\mathcal{A}_2$. L'alphabet de pile est le même que celui de $\mathcal{A}_1$ et la pile initilae également.

Aux transitions $(p_1, x) \xrightarrow{a} (q_1, v)$ dans $\mathcal{A}_1$ et $p_2 \xrightarrow{a} q_2$ dans $\mathcal{A}_2$, on associe la transition $((p_1, p_2), x) \xrightarrow{a} ((q_1, q_2), v)$ dans $\mathcal{A}$.

A tout état p_2 de $\mathcal{A}_2$ et toute transtion $(p_1, x) \xrightarrow{\varepsilon} (q_1, v)$ dans $\mathcal{A}_1$, on associe la transition $((p_1, p_2), x) \xrightarrow{\varepsilon} ((q_1, p_2), v)$ dans $\mathcal{A}$.

On vériife alors facilement que tout chemin étiqueté par un mot w dans $\mathcal{A}$ correspond à deux chemins simultanés étiquetés par w dans $\mathcal{A}_1$ et $\mathcal{A}_2$ et donc qu'un mot est reconnue par $\mathcal{A}$ si et seulement s'il est reconnu à la fois par $\mathcal{A}_1$ et $\mathcal{A}_2$. □

10.7. Propriété d'itération et propriétés de non-clôture

10.7.1 Théorème (Propriété d'itération)**.** *Soit* L *un langage algébrique. Alors il existe un entier* N *telle que tout mot* $w \in L$ *de longueur supérieure ou égale à* N *se décompose sous la forme* $w = uxzyv$ *où*

- *Les mots* x *et* y*ne sont pas tous deux vides,*
- $|xzy| \leq N$,
- $ux^n zy^n v \in L$ *pour tout* $n \geq 0$.

Démonstration. On peut supposer que $\varepsilon \notin L$. Ainsi L est engendré par une grammaire $\mathcal{G} = (\Sigma, V, S, R)$ sans production vide. Soit m la longueur maximale des productions de $\mathcal{G}$ ($m = \max\{|u| : T \to u\}$). Alors un arbre de dérivation de dérivation de $\mathcal{G}$ qui a une hauteur h a au plus m^h feuilles.

Posons $N = m^{|V|+1}$. Soit $w \in L$ tel que $|w| \geq N$. Alors w est associé à un arbre de dérivation. On peut supposer que cet arbre ne contient pas de tronçon correspondant à une suite de réécritures de la forme $T_0 \to T_1 \to ... \to T_0$ (sinon on les supprime). Comme $|w| \geq N$, cet arbre a une hauteur $h > |V|$ et contient donc nécessairement une branche avec deux noeuds étiquetés par une même variable. On considère les deux noeuds les plus bas dans cette branche vérifiant cette propriété. Notons T la variable qui étiquette ces noeuds. Il suit alors que l'on a une $\mathcal{G}$-dérivation de la forme

$$S \to \cdots \to uTv \to \cdots \to uxTyv \to \cdots \to uxzyv = w$$

telle que $T \to \cdots \to xTy$ et $T \to \cdots \to z$. Donc pour tout $n \geq 0$, on a $S \to \cdots \to ux^n zy^n v$. Par ailleurs le mot xzy est associé à un sous-arbre de hauteur inférieure ou égale à $|V| + 1$ (car on a pris les noeuds les plus bas de la branche). Donc $|xzy| \leq N$.

Enfin, comme on a supprimé de l'arbre les réécritures de la forme $T_0 \to T_1 \to ... \to T_0$, les mots x et y ne peuvent être tous deux vides. □

10.7.2. Exemple. Le langage $L = \{a^n b^n c^n : n \geq 0\}$ n'est pas algébrique. Soit $w = uxzyv \in L$ avec x ou y non vide. Si x ou y contient deux lettres différentes alors $ux^2zy^2v \notin L$. Sinon, x et y ne sont écrits chacun qu'avec une seule lettre. Il y a donc une troisième lettre qui n'est ni dans x, ni dans y. Alors dans le mot ux^2zy^2v il n'y a pas les nombres d'occurrences de a, de b et de c ne sont pas tous identiques.

10.7.3 Corollaire. *La classe des langages algébriques n'est pas close par intersection.*

Démonstration. Les langages $L_1 = \{a^n b^n c^k : n \geq 0, k \geq 0\}$ et $L_2 = \{a^n b^k c^k : n \geq 0, k \geq 0\}$ sont algébriques. Mais le langage $L_1 \cap L_2 = \{a^n b^n c^n : n \geq 0\}$ ne l'est pas. □

10.7.4 Corollaire. *La classe des langages algébriques n'est pas close par complémentaire.*

Démonstration. La classe des langages algébriques est close par union. Donc si elle était close par complémentaire, elle le serait également par intersection. □

10.7.5 Corollaire. *Il existe un langage algébrique qui n'est reconnaissable par aucun automate à pile déterministe. En particulier, on ne peut pas déterminiser tous les automates à pile.*

Démonstration. La classe des langages reconus par automate à pile déterministe est une sous-classe de langages algébriques qui est close par complémentaire (voir 8.6.7). □

EXERCICES

10.1 Exercice. Vérifier qu'un langage est rationnel si et seulement s'il est engendré par une grammaire linéaire à droite.

10.2 Exercice.
1. Soit la grammaire $\mathcal{G} = (\{a, b\}, \{S\}, S, \{S \to SaS|b\})$. Montrer que $\mathcal{G}$ est ambigüe.
2. Soit la grammaire $\mathcal{G}' = (\{a, b\}, \{S, T\}, S, \{S \to Tb, T \to Tba|\varepsilon\})$. Montrer que $\mathcal{G}$ et $\mathcal{G}'$ engendrent le même langage, mais que $\mathcal{G}'$ n'est pas ambigüe.

10.3 Exercice.
1. Montrer les propositions 10.6.2 et 10.6.3.
2. Soient deux automates à piles $\mathcal{A}_1$ et $\mathcal{A}_2$ reconnaissant L_1 et L_2 par états finaux (respectivement par pile vide). Donner des automates reconnaissant $L_1 \cup L_2$, L_1L_2, L_1^* par états finaux (respectivement par pile vide).

10.4 Exercice. Donner une grammaire contextuelle (c'est-à-dire de type 1) qui engendre le langage $L = \{a^n b^n c^n : n > 0\}$.

10.5 Exercice. Montrer que le langage $L = \{a^i b^j c^k : \ i, j, k \geq 0, i \neq j \text{ ou } j \neq k\}$ est algébrique. (On pourra décomposer L en l'union de quatre langages algébriques). Que peut-on dire du complémentaire de L ?

10.6 Exercice. Montrer que les langages suivants ne sont pas algébriques :
1. $L_1 = \{a^n b^n a^n : n \geq 0\}$,
2. $L_2 = \{a^m b^n a^m : m \geq n \geq 0\}$,
3. $L_3 = \{uu : u \in \{a, b\}^*\}$,
4. $L_4 = \{a^{2^n} : n \geq 0\}$.

10.7 Exercice. Soit L le langage $\{a^n b^m a^s b^t : n + s = m + t\}$.
1. Donner une grammaire algébrique engendrant L.
2. Construire un automate à pile déterministe reconnaissant L par pile vide.