Mémoire
Langages algébriques. Exemples et applications.

Xavier Montillet

17 mai 2016

Résumé

Les langages algébriques sont assez complexes pour pouvoir représenter la plupart des langages de programmations actuels tout en étant assez simples pour que certaines propriétés restent décidables. Ils ont de nombreuses applications en compilation.

Dans cette leçon, nous présentons d'abord les représentations usuelles des langages algébriques et quelques-unes de leurs propriétés. Nous nous intéressons ensuite aux problèmes de décision sur les langages algébriques. Enfin, nous décrivons leurs applications à la compilation et certaines sous-classes utiles dans ce cadre.

Table des matières

1 Représentations

Dans cette section, nous donnons deux définitions équivalentes des langages algébriques : une par des grammaires et une par des automates.

1.1 Grammaires algébriques

Définition 1 (Grammaire algébrique)**.** Une *grammaire algébrique* est un quadruplet (Σ, V, R, S) où Σ et V sont des ensembles finis disjoints, $R \subseteq V \times (V \sqcup \Sigma)^*$ et $S \in V$. Les éléments de Σ sont appelés *terminaux* et ceux de V sont appelés *variables* ou *non terminaux*. Les éléments de R sont appelés *règles*. Le non terminal S est appelé *axiome*. On note $X \to u_1 + \cdots + u_n$ pour $(X, u_1), \ldots, (X, u_n) \in R$.

Exemple 2. $G_1 = (\Sigma_1,\ V_1, R_1)$, $\Sigma_1 = \{\ a,\ b\ \}$, $V_1 = \{\ S\ \}$,
$R_1 = \{\ S \to aSb,\ S \to \varepsilon\ \}$

Remarque 3. On se contentera souvent de donner R en utilisant la convention que les lettres minuscules sont des symboles terminaux et que les lettres majuscules sont des symboles non terminaux.

Exemple 4. $G_1 = \{S \to aSb + \varepsilon\}$,$G_2 = \left\{ \begin{array}{c} P \to aI + \varepsilon \\ I \to aP \end{array} \right\}$

$$G_{3,n} = \left\{ \begin{array}{c} S \to ST + \varepsilon \\ T \to a_1 S b_1 S \ + \ \ldots \ + \ a_n S b_n S \ + \ \varepsilon \end{array} \right\}$$

Définition 5 (Dérivation)**.** On étend $\to$ à $(V \sqcup \Sigma)^*$ par $\alpha X \beta \to \alpha u \beta$ si $X \to u$. On note $\to^*$ la clôture réflexive transitive de $\to$. Si $u \to v$, on dit que u se *dérive directement* en v.

Exemple 6. $S \to aSb \to aaSbb \to aabb$ dans G_1

Définition 7 (Langage engendré)**.** Pour tout $X \in V$, on pose $\widehat{\mathcal{L}_G}(u) \coloneqq \{\ v \in (V \sqcup \Sigma)^* : u \to^* v\ \}$ et $\mathcal{L}_G(u) \coloneqq \widehat{\mathcal{L}_G}(u) \cap \Sigma^*$. On appelle $\mathcal{L}_G(u)$ *langage engendré* par u dans G. On appelle *langage engendré* par la grammaire et note $\mathcal{L}(G)$ le langage $\mathcal{L}_G(S)$ engendré par l'axiom.

Exemple 8. $\mathcal{L}_{G_1}(S) = \{\ a^n b^n \ \mid\ n \in \mathbb{N}\ \}$, $\mathcal{L}_{G_2}(P) = \left\{\ a^{2n} \ \middle|\ n \in \mathbb{N}\ \right\}$,
$\mathcal{L}_{G_2}(I) = \left\{\ a^{2n+1} \ \middle|\ n \in \mathbb{N}\ \right\}$, $D_n^* \coloneqq \mathcal{L}_{G_{3,n}}(S)$ est appelé *langage de Dyck*. C'est le langage des mots bien parenthésés (si l'on considère a_i comme une parenthèse ouvrante et b_i comme la parenthèse fermante correspondante).

Définition 9 (Langage algébrique)**.** Un *langage algébrique* est un langage engendré par une grammaire algébrique.

Lemme 10 (Fondamental)**.** *Soit $G = (\Sigma, V, R, S)$ une grammaire algébrique et u et v deux mots de $(V \sqcup \Sigma)^*$. On suppose que u se factorise $u = u_1 u_2$.*

Alors il existe une dérivation $u \to^k v$ de longueur k si et seulement si v se factorise $v = v_1 v_2$ et s'il existe deux dérivations $u_1 \to^{k_1} v_1$ et $u_2 \to^{k_2} v_2$ où $k = k_1 + k_2$.

Définition 11 (Grammaire réduite)**.** Une grammaire $G = (\Sigma, V, R, S)$ est dite *réduite* si

— pour tout $X \in V$, $\mathcal{L}_G(X) \neq \emptyset$,

— pour tout $X \in V$, il existe $u, v \in (\Sigma \sqcup V)^*$ tels que $S \to^* uXv$.

Proposition 12 (Grammaire réduite)**.** *Pour toute grammaire algébrique G, $\mathcal{L}_G$ est engendré par une grammaire réduite de taille $O(|G|)$ calculable en temps $O(|G|)$.*

Définition 13 (Grammaire propre)**.** Une grammaire $G = (\Sigma, V, R, S)$ est dite *propre* si elle ne contient aucune règle de la forme $X \to \varepsilon$ ou de la forme $X \to Y$ pour $X, Y \in V$.

Proposition 14 (Grammaire propre)**.** *Pour toute grammaire algébrique G, $\mathcal{L}_G \setminus \{ \varepsilon \}$ est engendré par une grammaire propre de taille $O\left(|G|^2\right)$ calculable en temps $O\left(|G|^2\right)$.*

Définition 15 (Forme normale quadratique / de Chomsky)**.** Une grammaire $G = (\Sigma, V, R, S)$ est dite en *forme normale quadratique* si toutes ses règles sont de la forme $X \to YZ$ ou $X \to a$ où $X, Y, Z \in V$ et $a \in \Sigma$.

Proposition 16 (Forme normale quadratique / de Chomsky)**.** *Pour toute grammaire algébrique G, $\mathcal{L}_G \setminus \{ \varepsilon \}$ est engendré par une grammaire en forme normale quadratique de taille $O\left(|G|^2\right)$ calculable en temps $O\left(|G|^2\right)$.*

1.2 Automates à pile

Définition 17 (Automate à pile)**.** Un *automate à pile* est un septuplet $\mathcal{A} = (Q, \Sigma, \Gamma, \Delta, q_0, \gamma_0, F)$ où Q est un ensemble fini d'états, Σ est l'*alphabet d'entrée*, Γ est l'*alphabet de pile*, $\Delta \subseteq (Q \times (\Sigma \sqcup \{\varepsilon\}) \times (\Gamma \sqcup \{\varepsilon\})) \times (Q \times \Gamma^*)$ décrit les transitions, q_0 est l'*état initial*, γ_0 est le *symbole de pile initial* et $F \subseteq Q \times \Gamma^*$ est l'ensemble des *configurations acceptantes* (qui sont d'une forme particulière décrite plus bas). On note $(q, \gamma) \overset{u}{\rightarrowtail} (q', \alpha)$ pour $((q, u, \gamma), (q', \alpha)) \in \Delta$.

Définition 18 (Calcul d'un automate à pile)**.** On appelle *configuration* un élément de $Q \times \Gamma^*$. Une *étape de calcul* est une paire de configurations (C, C') telles que $C = (q, \gamma\beta)$, $C' = (q', \alpha\beta)$ et $(q, \gamma) \overset{u}{\rightarrowtail} (q', \alpha)$. On la note $C \overset{u}{\rightsquigarrow} C'$. Un *calcul* est une suite d'étapes de calcul consécutives $C_0 \overset{u_1}{\rightsquigarrow} C_1 \overset{u_2}{\rightsquigarrow} \dots \overset{u_n}{\rightsquigarrow} C_n$. Le mot $u_1 \dots u_n$ est l'*étiquette* du calcul.

Définition 19 (Modes d'acceptation d'un automate à pile)**.** Il existe plusieurs *modes d'acceptation* pour définir F.

— Pile vide : $(q, \alpha) \in F \iff \alpha = \varepsilon$.

— État final : $(q, \alpha) \in F \iff q \in Q_F$ où $Q_F \subseteq Q$ est un ensemble d'états dits finaux.

Définition 20 (Langage accepté)**.** Un mot u est *accepté* par un automate $\mathcal{A}$ à pile s'il existe un calcul $(q_0, \gamma_0) \overset{u}{\rightsquigarrow}{}^* C$ de la configuration initiale (q_0, γ_0) à une configuration finale $C \in F$. Le langage $\mathcal{L}(\mathcal{A})$ *accepté* par un automate $\mathcal{A}$ est l'ensemble des mots qu'il accepte.

Proposition 21 (Équivalence des modes d'acceptation)**.** *Les différents modes d'acceptation sont équivalents dans la mesure où ils permettent tous d'accepter exactement les mêmes langages.*

Théorème 22 (Équivalence grammaires algébriques / automates à piles)**.** *Un langage $L \subseteq \Sigma^*$ est engendré par une grammaire algébrique si et seulement si il est accepté par un automate à pile.*

2 Propriétés

2.1 Lemme d'itération

Dans cette sous-section, nous nous intéressons à l'équivalent du lemme de l'étoile pour les langages algébriques : le lemme d'Ogden. Il permet, entre autres, de démontrer que certains langages ne sont pas algébriques.

Lemme 23 (Ogden)**.** *Pour tout grammaire $G = (\Sigma, V, R, S)$ et toute variable $X \in V$, il existe un entier K tel que tout mot $f \in \widehat{\mathcal{L}_G(X)}$ ayant au moins K lettres distinguées se factorise en $f = \alpha u \beta v \gamma$, où $\alpha, u, \beta, v, \gamma \in (\Sigma \sqcup V)^*$, avec*

— *$S \to^* \alpha T \gamma$ et $T \to^* uTv + \beta$*

— *soit α, u, β, soit β, v, γ contiennent des lettres distinguées.*

— *$u\beta v$ contient moins de K lettres distinguées.*

Corollaire 24 (Théorème de Bar-Hillel, Perles et Shamir)**.** *Pour tout langage algébrique L, il existe $N \geq 0$ tel que pour tout mot $f \in L$, si $|f| \geq N$ alors on peut trouver une factorisation $f = \alpha u \beta v \gamma$ telle que $|uv| > 0$, $|u\beta v| < N$ et $\alpha u^n \beta v^n \gamma \in L$ pour tout $n \geq 0$.*

Application 25 (de 24). Le langage $\{\, a^n b^n c^n \mid n \in \mathbb{N} \,\}$ n'est pas algébrique.

Exemple 26. Ce langage pourrait représenter de la mise en forme en mode texte.

```
+-------+
| titre |
+-------+
```

Application 27 (de 23). Le langage $\{ a^m b^n c^m d^n \mid n, m \in \mathbb{N} \}$ n'est pas algébrique.

Ce langage pourrait représenter le fait qu'on a déclaré deux procédures à m et n arguments et qu'on les a ensuite utilisées.

2.2 Propriétés de clôture

Dans cette sous-section, nous nous intéressons aux propriétés de clôture des langages algébriques qui permettent de construire des langages algébriques à partir d'autres langages algébriques (a priori plus simples).

Proposition 28 (Opérations rationnelles)**.** *L'ensemble des langages algébrique est clos par union, concaténation et étoile.*

Proposition 29. *Tout langage rationnel est algébrique et l'inclusion est stricte.*

Proposition 30. *L'ensemble des langages algébrique est clos par intersection avec un rationnel mais ni par complémentation, ni par intersection.*

Définition 31 (Morphisme)**.** Un *morphisme* de X^* dans Y^* est une fonction $\varphi : X^* \to Y^*$ telle que $\varphi(\varepsilon) = \varepsilon$ et $\varphi(uv) = \varphi(u)\varphi(v)$ pour tous $u, v \in X^*$. L'*image* de $L \subseteq X^*$ par φ est $\{ f(u) : u \in L \}$ et l'*image inverse* de $L' \subseteq Y^*$ par φ est $\{ u \in X^* : \varphi(u) \in L' \}$.

Proposition 32. *L'ensemble des langages algébrique est clos par morphisme et morphisme inverse.*

Définition 33 (Substitution algébrique)**.** Une *substitution algébrique* est une fonction $\sigma : X^* \to \mathcal{P}(Y^*)$ telle que $\sigma(\varepsilon) = \{ \varepsilon \}$, $\sigma(uv) = \sigma(u)\sigma(v)$ pour tous $u, v \in X^*$, et pour tout $a \in X$, $\sigma(a)$ est algébrique. L'*image* de $L \subseteq X^*$ par σ est $\bigcup_{u \in L} \sigma(u)$.

Proposition 34. *L'ensemble des langages algébrique est clos par substitution algébrique.*

2.3 Théorème de Chomsky et Schützenberger

Nous avons vu à l'exemple 8 que les langages de Dyck sont algébriques. Le théorème de Chomsky et Schützenberger affirme, en un certain sens, que les langages de Dyck contiennent l'essence des langages algébriques.

Définition 35 (Morphisme alphabétique)**.** Un morphisme $\varphi : X^* \to Y^*$ est dit *alphabétique* si pour tout $u \in X^*$, $|\varphi(u)| \leq 1$.

Théorème 36 (Chomsky et Schützenberger)**.** *Un langage L est algébrique si et seulement si $L = \varphi(D_n^* \cap K)$ pour un entier n, un langage rationnel K et un morphisme alphabétique φ.*

Lemme 37. *Pour tout $n \geq 0$, il existe un morphisme $\psi : A_n^* \to A_2^*$ tel que $D_n^* = \psi^{-1}(D_2^*)$.*

Corollaire 38. *Tout langage algébrique s'écrit $\varphi\left(\psi^{-1}(D_2^* \cap K)\right)$ pour des morphismes φ et ψ et un langage rationnel K.*

Remarque 39. Une fonction $X \mapsto \varphi\left(\psi^{-1}(X \cap K)\right)$ s'appelle une *transduction rationnelle*. Ce sont des transformations très naturelles qui peuvent être réalisées avec des automates à deux bandes (une pour l'entrée et une pour la sortie).

3 Problèmes de décision

Le gain d'expressivité par rapport aux langages rationnels rend certains problèmes de décision plus difficiles voire indécidables.

3.1 Problèmes décidables

Théorème 40. *La vacuité d'un langage algébrique (représenté par une grammaire G) est décidable en temps $O(|G|)$.*

Théorème 41. *L'appartenance d'un mot u à un langage algébrique (représenté par une grammaire) est décidable en temps $O\left(|G|\,|u|^3\right)$.*

Remarque 42. La forme normale quadratique donne un algorithme exponentiel en $|u|$ car on peut borner la longueur d'une dérivation et donc tester toutes les dérivations possibles.

3.2 Problèmes indécidables

Proposition 43 (Développement 1, partie 1/2)**.** *Les problèmes suivants sont indécidable :*

1. *Pour deux grammaires, l'intersection des langages engendrés est-elle vide ?*
2. *Pour deux grammaires, les langages engendrés sont-ils égaux ?*
3. *Pour une grammaire, engendre-t-elle le langage de tous les mots sur l'alphabet ?*

Pour prouver le théorème on va utiliser l'indécidabilité du problème de correspondance de Post (PCP). On commence donc par définir des langages associés à une instance de PCP.

Soit $(u_1, v_1), \ldots, (u_m, v_m)$ une instance de PCP sur un alphabet Σ. On pose $A = \{a_1, \ldots, a_m\}$ où $a_1, \ldots, a_m$ sont des lettres n'appartenant pas à Σ. On définit deux langages sur $\Sigma \sqcup A$:

$$L_u = \{\, u_{i_1} \ldots u_{i_n} a_{i_n} \ldots a_{i_1} : n \geq 0 \text{ et } 1 \leq i_k \leq m \,\} \setminus \{\, \varepsilon \,\}$$

$$L'_u = \{\, wa_{i_n} \dots a_{i_1} : n \geq 0, w \in \Sigma^* \text{ et } w \neq u_{i_1} \dots u_{i_n} \,\} \cup \{\, \varepsilon \,\}$$

On note $\varphi_u(a_{i_n} \dots a_{i_1})$ pour $u_{i_1} \dots u_{i_n}$. On a alors $L_u = \{\, \varphi_u(x)x : x \in A^* \,\} \setminus \{\, \varepsilon \,\}$ et $L'_u = \{\, wx : x \in A^* \text{ et } w \neq \varphi_u(x) \,\} \cup \{\, \varepsilon \,\}$.

On remarque que $L_u \sqcup L'_u = \Sigma^* A^*$.

On définit de même L_v et L'_v en remplaçant u_i par v_i.

Lemme 44. *L_u est algébrique.*

Démonstration. La grammaire $G_u = \left\{ S \to \sum_{i=1}^{m} u_i S a_i + \sum_{i=1}^{m} u_i a_i \right\}$ convient. □

Lemme 45. *L'_u est algébrique.*

Démonstration. On pose la grammaire G'_u suivante d'axiome S_0 :

$$S_0 \to S + \varepsilon$$
$$S \to \sum_{i=1}^{m} u_i S a_i + \sum_{\substack{1 \leq i \leq m \\ x \in \Sigma^* \\ |x| = |u_i| \\ x \neq u_i}} x R a_i + \sum_{\substack{1 \leq i \leq m \\ x \in \Sigma^* \\ |x| < |u_i|}} x T a_i + \sum_{b \in \Sigma} bV$$
$$R \to \sum_{i=1}^{m} R a_i + \sum_{b \in \Sigma} bR + \varepsilon$$
$$T \to \sum_{i=1}^{m} T a_i + \varepsilon$$
$$V \to \sum_{b \in \Sigma} bV + \varepsilon$$

On commence par remarquer que $\mathcal{L}_{G'_u}(R) = \Sigma^* A^*$, $\mathcal{L}_{G'_u}(T) = A^*$ et $\mathcal{L}_{G'_u}(V) = \Sigma^*$.

On montre l'égalité $\mathcal{L}_{G'_u}(S) = \{\, wx : x \in A^* \text{ et } w \neq \varphi_u(x) \,\}$ par double inclusion. L'égalité $\mathcal{L}(G'_u) = L'_u$ en découle immédiatement.

«$\subseteq$» : Soit $x \in \mathcal{L}_{G'_u}(S)$. Une dérivation $S \to^* x$ est nécessairement de la forme

$$S \to u_{i_1} S a_{i_1} \to S \to u_{i_1} u_{i_2} S a_{i_2} a_{i_1} \to^* u_{i_1} \dots u_{i_j} S a_{i_j} \dots a_{i_1} \to y \to^* x$$

où S n'apparaît pas dans y. On va noter $\alpha = a_{i_j} \dots a_{i_1}$. On a donc $S \to^* \varphi_u(\alpha) S \alpha \to y$ et S n'apparaît pas dans y. On a alors trois cas selon le non terminal présent dans y :

— $\varphi_u(\alpha) S \alpha \to \varphi_u(\alpha) z R a_i \alpha \to \varphi_u(\alpha) z \beta \gamma a_i \alpha$ avec $\beta \in \Sigma^*$, $\gamma \in A^*$, $1 \leq i \leq m$, $z \in \Sigma^*$, $|z| = |u_i|$ et $z \neq u_i$. On a donc bien $\varphi_u(\gamma a_i \alpha) = \varphi_u(\alpha)\varphi_u(a_i)\varphi_u(\gamma) = \varphi_u(\alpha) u_i \varphi_u(\gamma) \neq \varphi_u(\alpha) u \beta$.

— $\varphi_u(\alpha)S\alpha \to \varphi_u(\alpha)zTa_i\alpha \to \varphi_u(\alpha)z\gamma a_i\alpha$ avec $\gamma \in A^*$, $1 \leq i \leq m$, $z \in \Sigma^*$ et $|z| < |u_i|$. On a donc bien $\varphi_u(\gamma a_i\alpha) = \varphi_u(\alpha)\varphi_u(a_i)\varphi_u(\gamma) = \varphi_u(\alpha)u_i\varphi_u(\gamma) \neq \varphi_u(\alpha)u$.
— $\varphi_u(\alpha)S\alpha \to \varphi_u(\alpha)bV\alpha \to \varphi_u(\alpha)b\beta\alpha$ avec $\beta \in \Sigma^*$. On a donc bien $\varphi_u(\alpha) \neq \varphi_u(\alpha)b\beta$.

«$\supseteq$» : On le prouve par récurrence sur le nombre de symboles de A.
— Initialisation : Soit $x \in L'_u \cap \Sigma^* A^0$. On a

$$S \to x_1V \to x_1x_2V \to \cdots \to x_1 \dots x_{|x|}V = xV \to x$$

Donc on a bien $x \in \mathcal{L}(G'_u)$.
— Récurrence : Supposons que pour tout $x \in L'_u \cap \Sigma^* A^k$, $x \in \mathcal{L}(G'_u)$, pour un certain $k \geq 0$. Soit $x \in L'_u \cap \Sigma^* A^{k+1}$. $x = w\alpha a_i$ avec $w \in \Sigma^*$, $\alpha \in A^*$, $a_i \in A$ et $\varphi_u(\alpha a_i) \neq w$. On a donc $u_i\varphi_u(\alpha) \neq w$.
— Si $|w| < |u_i|$, on a $S \to wTa_i \to^* w\alpha a_i = x$. Donc $x \in \mathcal{L}(G'_u)$.
— Si $|w| \geq |u_i|$, on factorise w en $w = w'w''$ avec $|w'| = |u_i|$. On a donc $u_i\varphi_u(\alpha) \neq w'w''$.
— Si $w' = u_i$, alors $\varphi_u(\alpha) \neq w''$ donc $w''\alpha \in L'_u$. Par hypothèse de récurrence, on a donc $S \to^* w''\alpha$. Donc $S \to u_iSa_i \to^* u_iw''\alpha a_i = w'w''\alpha a_i = w\alpha a_i = x$. Donc $x \in \mathcal{L}(G'_u)$.
— Si $w' \neq u_i$, $S \to w'Ra_i \to^* w'w''\alpha a_i = w\alpha a_i = x$. Donc $x \in \mathcal{L}(G'_u)$.

□

Lemme 46. *Une instance de PCP est positive si et seulement si $L_u \cap L_v \neq \emptyset$.*

Démonstration. «$\Rightarrow$» : Si $u_{i_1} \dots u_{i_n} = v_{i_1} \dots v_{i_n}$ pour $n \geq 1$ alors $L_u \ni u_{i_1} \dots u_{i_n}a_{i_n} \dots a_{i_1} = v_{i_1} \dots v_{i_n}a_{i_n} \dots a_{i_1} \in L_v$ donc $L_u \cap L_v \neq \emptyset$.

«$\Leftarrow$» : Si $L_u \cap L_v \neq \emptyset$, soit $x \in L_u \cap L_v$. $x = \varphi_u(\alpha)\alpha$ pour un certain $\alpha \in A^+$ et $x = \varphi_v(\alpha')\alpha'$ pour un certain $\alpha' \in A^*$. Comme Σ et A sont disjoints, $\varphi_u(\alpha) = \varphi_v(\alpha')$ et $\alpha = \alpha'$. On en déduit $\varphi_u(\alpha) = \varphi_v(\alpha)$. On factorise α en mots de taille 1 : $\alpha = a_{i_n} \dots a_{i_1}$ avec $n \geq 1$. On a alors $u_{i_1} \dots u_{i_n} = \varphi_u(\alpha) = \varphi_v(\alpha) = v_{i_1} \dots v_{i_n}$ donc l'instance de PCP est positive.

□

Démonstration de 43.

1' On pose $1'$ la restriction du problème 1 aux langages de la forme L_u. Par le lemme précédent, le calcul de L_u et L_v (qui peut effectivement se faire) est une réduction de PCP à $1'$. Comme PCP est indécidable, cela implique de $1'$ l'est.

1. L'inclusion des instance de 1' dans celles de 1 est une réduction de $1'$ vers 1. Or $1'$ est indécidable donc 1 l'est également.

3. $(L_u \cap L_v)^c = (\Sigma + A)^* \setminus (L_u \cap L_v) = ((\Sigma + A)^* \setminus L_u) \cup ((\Sigma + A)^* \setminus L_v)$ donc $L_u \cap L_v = \emptyset \iff ((\Sigma + A)^* \setminus L_u) \cup ((\Sigma + A)^* \setminus L_v) = (\Sigma + A)^*$. Or

comme $L_u \sqcup L'_u = \Sigma^* A^*$, on a $(\Sigma + A)^* \setminus L_u = L'_u \cup ((\Sigma + A)^* \setminus \Sigma^* A^*)$ qui est donc algébrique. Le calcul de $((\Sigma + A)^* \setminus L_u) \cup ((\Sigma + A)^* \setminus L_v)$ est donc une réduction de $1'$ à 3. Or $1'$ est indécidable donc 3 l'est aussi.

2. 3 se réduit à 2 qui est donc également indécidable. □

4 Application à la compilation

4.1 Arbres de dérivation et ambiguïté

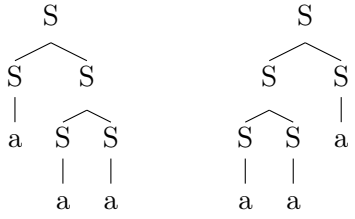
Lors de la compilation, on cherche à donner une structure d'arbre à un programme (c'est-à-dire un mot du langage de programmation). Il est donc naturel de s'assurer que pour chaque mot, un tel arbre est unique. Les langages vérifiant cette propriété sont dits non ambigus.

Définition 47 (Arbre de dérivation)**.** Soit $G = (\Sigma, V, R, S)$ une grammaire algébrique. Un *arbre de dérivation* est un arbre fini dont les nœuds sont étiquetés par $V \sqcup \Sigma \sqcup \{ \varepsilon \}$ vérifiant la propriété suivante. Si X est l'étiquette d'un nœud interne et $\alpha_1, \ldots, \alpha_n$ sont les étiquettes de ses fils alors $S \to \alpha_1 \ldots \alpha_n$.

Définition 48 (Frontière)**.** La *frontière* d'un arbre de dérivation est le mot obtenu par concaténation des étiquettes de ses feuilles de gauche à droite.

Proposition 49. *Pour tout non-terminal $X \in V$, le langage $\mathcal{L}_G(X)$ (resp. $\widehat{\mathcal{L}_G(X)}$) est l'ensemble des mots $u \in \Sigma^*$ (resp. $u \in (V \sqcup \Sigma)^*$) tels qu'il existe un arbre de dérivation de racine X et de frontière u.*

Exemple 50. Les deux arbres suivants sont des arbres de dérivation (de frontière aaa) de la grammaire $S \to SS + a$.



Définition 51 (Grammaire ambiguë)**.** Une grammaire algébrique G est dite *ambiguë* s'il existe un mot ayant deux arbres de dérivation distincts dont les racines ont la même étiquette.

Exemple 52. $S \to SS + a$ est une grammaire algébrique ambiguë.

Remarque 53. $S \to aS + a$ est une grammaire algébrique non ambiguë générant le même langage.

Proposition 54 (Développement 1, partie 2/2)**.** *L'ambiguïté d'une grammaire algébrique est indécidable.*

Démonstration. On utilise les constructions de la preuve de la proposition 43.

On pose $G_{u,v}$ la grammaire suivante.

$$\begin{aligned} S &\to S_1 + S_2 \\ S_1 &\to \sum_{i=1}^{m} u_i S_1 a_i + \sum_{i=1}^{m} u_i a_i \\ S_2 &\to \sum_{i=1}^{m} v_i S_2 a_i + \sum_{i=1}^{m} v_i a_i \end{aligned}$$

$G_{u,v}$ engendre $L_u \cup L_v$.

Les a_i permettent de savoir quelles règles $S_1 \to u_i s_1 a_i$ ou $S_1 \to u_i a_i$ ont été utilisées pour engendrer le mot donc les grammaires
$\{ S_1 \to \sum_{i=1}^{m} u_i S_1 a_i + \sum_{i=1}^{m} u_i a_i \}$ et $\{ S_2 \to \sum_{i=1}^{m} v_i S_2 a_i + \sum_{i=1}^{m} v_i a_i \}$ sont non ambiguës La grammaire $G_{u,v}$ est donc ambiguë si et seulement si $\mathcal{L}_{G_{u,v}}(S_1) \cap \mathcal{L}_{G_{u,v}}(S_2) \neq \emptyset$, c'est-à-dire si $L_u \cap L_v \neq \emptyset$. On a donc une réduction de $1'$ au problème de décision de l'ambiguïté d'une grammaire algébrique. $1'$ étant indécidable, l'ambiguïté d'une grammaire algébrique l'est également. □

Définition 55 (Langage ambigu)**.** Un langage algébrique est dit *non ambigu* s'il existe une grammaire algébrique non ambiguë qui l'engendre et *inhéremment ambigu* si toute grammaire algébrique qui l'engendre est ambiguë.

Proposition 56. *L'ensemble des langages algébriques non ambigus est clos par union disjointe, par intersection avec un rationnel et par morphisme inverse mais ni par union, ni par morphisme, ni par substitution.*

Proposition 57. *L'inclusion des langages non ambigus dans les langages algébriques est stricte.*

4.2 Langages déterministes

Ne pouvant pas décider l'ambiguïté d'une grammaire algébrique, on cherche une propriété plus forte que l'on sait décider. On étend donc la notion d'automate déterministe aux automates à piles.

Définition 58 (Automate à pile déterministe)**.** Un automate à pile $\mathcal{A} = (Q, \Sigma, \Gamma, \Delta, q_0, \gamma_0, F)$ est *déterministe* si pour toute configuration C,

- soit il existe une unique transition sortante $C \overset{\varepsilon}{\rightarrowtail} C'$,
- soit il n'existe par de transition sortante étiquetée par ε et pour toute $a \in \Sigma$, il existe au plus une transition sortante de la forme $C \overset{a}{\rightarrowtail} C'$.

Remarque 59. Il n'y a plus équivalence entre les différents modes d'acceptation.

Définition 60 (Langage algébrique déterministe(-préfixe))**.** Un langage algébrique est dit *déterministe* (resp. *déterministe-préfixe*) s'il est accepté par un automate à pile déterministe par état final (resp. par pile vide).

Proposition 61. *Tout langage rationnel est algébrique déterministe et l'inclusion est stricte.*

Définition 62 (Langage préfixe)**.** Un langage L est dit *préfixe* si pour tout mot $u \in L$ et tout préfixe strict v de u (c'est-à-dire $u = vw$ et $|v| < |u|$), $v \notin L$.

Remarque 63. Si L est un langage quelconque et \$ n'apparaît pas dans L, alors $L\$$ est un langage préfixe.

Proposition 64. *Un langage est déterministe-préfixe si et seulement si il est déterministe et préfixe.*

Proposition 65. *L'ensemble des langages algébriques déterministes est stable par complémentation, intersection avec un rationnel et morphisme inverse, mais ni par union (même disjointe), ni par intersection, ni par morphisme, ni par substitution.*

Proposition 66. *Tout langage algébrique déterministe est non ambigu et l'inclusion est stricte.*

4.3 Automate des items et analyses LL(k) et LR(k)

L'automate des items associé à une grammaire algébrique accepte le langage engendré par la grammaire, et peut donc permettre de montrer un sens de l'équivalence 22. Il permet également de décrire et motiver les analyses LL(k) et LR(k) introduites dans la suite.

Soit une grammaire algébrique $G = (\Sigma,\ V,\ R, S)$ à laquelle on ajoute une nouvelle variable S' et une nouvelle règle $S' \to S$.

Définition 67 (Item)**.** À $A \to \alpha\beta \in R$, on associe $[A \to \alpha \bullet \beta]$ que l'on appelle un *item*. On note It l'ensemble des items associés à G.

Intuition : $[A \to \alpha \bullet \beta]$ représente le fait que l'on cherche à reconnaître $u \in \Sigma^*$ tel que $A \to^* \alpha\beta \to^* u$. Par le lemme fondamental, on a donc $v, w \in \Sigma^*$ tel que $u = vw$, $\alpha \to^* v$ et $\beta \to^* w$. La position de $\bullet$ indique que l'on a déjà lu v et que l'on veut maintenant lire w.

Définition 68 (Automate des items)**.** L'automate des items A_G associé à G (et S) est l'automate à pile (généralisé [1]) à un seul état dont l'alphabet de pile est It, qui commence avec une pile contenant $[S' \to \bullet S]$, dont la pile « acceptrice » est $[S' \to S\bullet]$, et dont les transitions sont toutes celles de la forme (où l'état est omis car l'automate n'a qu'un état) :

1. On s'autorise à lire plusieurs symboles sur la pile pour faire une transition. On peut émuler ce comportement avec un automate à pile normal qui retient des symboles lus dans ses états.

(E) « Expansion » : $[X \to \alpha \bullet B\gamma] \overset{\varepsilon}{\rightarrowtail} [X \to \alpha \bullet B\gamma]\,[B \to \bullet\beta]$

(L) « Lecture » : $[X \to \alpha \bullet b\gamma] \overset{b}{\rightarrowtail} [X \to \alpha b \bullet \gamma]$

(R) « Réduction » : $[X \to \alpha \bullet B\gamma]\,[B \to \beta\bullet] \overset{\varepsilon}{\rightarrowtail} [X \to \alpha B \bullet \gamma]$

Définition 69 (Histoire)**.** L'histoire d'une suite d'items (qui peut apparaître dans la pile) $\rho = [X_1 \to \alpha_1 \bullet \beta_1] \dots [X_n \to \alpha_n \bullet \beta_n]$ est $\text{hist}(\rho) \coloneqq \alpha_1 \dots \alpha_n$ (qui représente ce qui a déjà été lu).

Lemme 70 (Développement 2, partie 1/3)**.** *Si* $[S' \to \bullet S] \overset{u}{\rightsquigarrow}{}^* \rho$ *alors* $\text{hist}(\rho) \to^* u$.

Démonstration. On le prouve par récurrence sur la longueur du calcul dans l'automate.

— Pour un calcul $[S' \to \bullet S] \overset{u}{\rightsquigarrow}{}^0 \rho$ de longueur 0, on a $u = \varepsilon$ et $\rho = [S' \to \bullet S]$ et donc $\text{hist}(\rho) = \varepsilon \to^* \varepsilon = u$.

— Pour un calcul $[S' \to \bullet S] \overset{u}{\rightsquigarrow}{}^{n+1} \rho$ de longueur $n+1$, on a ρ' tel que $[S' \to \bullet S] \overset{v}{\rightsquigarrow}{}^n \rho' \overset{w}{\rightsquigarrow}{}^1 \rho$ avec $u = vw$. Par récurrence, $\text{hist}(\rho') \to^* v$. On a trois cas suivant si $\rho' \overset{w}{\rightsquigarrow} \rho$ est une transition (E), (L) ou (R) :
(E) $\rho = \rho'\,[B \to \bullet\beta]$ et $w = \varepsilon$ donc

$$\text{hist}(\rho) = \text{hist}\left(\rho'\,[B \to \bullet\beta]\right) = \text{hist}\left(\rho'\right) \to^* v = u$$

(L) $\rho' = \rho''\,[X \to \alpha \bullet b\gamma]$, $\rho = \rho''\,[X \to \alpha b \bullet \gamma]$ et $w = b$ donc

$$\text{hist}(\rho) = \text{hist}\left(\rho''\,[X \to \alpha b \bullet \gamma]\right) = \text{hist}\left(\rho''\right)\alpha b = \text{hist}\left(\rho'\right) b \to^* vb = u$$

(R) $\rho' = \rho''\,[X \to \alpha \bullet B\gamma]\,[B \to \beta\bullet]$, $\rho = \rho''\,[X \to \alpha B \bullet \gamma]$ et $w = \varepsilon$ donc

$$\text{hist}\left(\rho''\right)\alpha\beta = \text{hist}\left(\rho'\right) \to^* v$$

Or comme $[B \to \beta\bullet] \in \text{It}$, on a aussi $B \to \beta \in R$. D'où

$$\text{hist}(\rho) = \text{hist}\left(\rho''\right)\alpha B \to \text{hist}\left(\rho''\right)\alpha\beta \to^* v = u$$

□

Lemme 71 (Développement 2, partie 2/3)**.** *Pour tout* $n \geq 1$, $A \in V$ *et* $u \in \Sigma^*$, *si* $A \to^n u$ *alors il existe* $A \to \alpha \in R$ *tel que* $[A \to \bullet\alpha] \overset{u}{\rightsquigarrow}{}^* [A \to \alpha\bullet]$.

Démonstration. On le prouve par récurrence forte sur n. □

— Pour une dérivation de longueur 1, $A \to^1 u$ donc $A \to u \in R$. On en déduit (en notant $a_1, \dots, a_k$ les lettres de u)

$$[A \to \bullet u] = [A \to \bullet a_1 a_2 \dots a_k] \underset{(L)}{\overset{a_1}{\rightsquigarrow}} [A \to a_1 \bullet a_2 \dots a_k] \underset{(L)}{\overset{a_2}{\rightsquigarrow}} \dots$$

$$\overset{a_k}{\underset{(L)}{\rightsquigarrow}} [A \to a_1 a_2 \dots a_k \bullet] = [A \to u \bullet]$$

Et donc $[A \to \bullet u] \overset{u}{\rightsquigarrow}^* [A \to u\bullet]$.

— Pour une dérivation de longueur $n+1$, on a $A \to^1 \alpha \to^n u$. On note $\alpha_1, \dots, \alpha_k$ les lettres de α. Par le lemme fondamental, on a $u_1, \dots, u_k \in \Sigma^*$ tel que pour chaque $i \in [\![1,\ k]\!]$, $\alpha_i \to^{n_i} u_i$ avec $\sum_{i=1}^{k} n_i = n$. En particulier, pour chaque $i \in [\![1,\ k]\!]$, $n_i \leq n$. Pour chaque $i \in [\![1,\ k]\!]$, il y a deux cas :

— $\alpha_i \in \Sigma$ et alors $[A \to \alpha_1 \dots \alpha_{i-1} \bullet \alpha_i \alpha_{i+1} \dots \alpha_k] \overset{\alpha_i}{\underset{(L)}{\rightsquigarrow}} [A \to \alpha_1 \dots \alpha_{i-1}\alpha_i \bullet \alpha_{i+1} \dots \alpha_k]$

— $\alpha_i \in V$ et alors, par récurrence, on a $\alpha_i \to \beta \in P$ tel que $[\alpha_i \to \bullet\beta] \overset{u_i}{\rightsquigarrow}^* [\alpha_i \to \beta\bullet]$. On a donc

$$[A \to \alpha_1 \dots \alpha_{i-1} \bullet \alpha_i \alpha_{i+1} \dots \alpha_k] \overset{\varepsilon}{\underset{(E)}{\rightsquigarrow}} [A \to \alpha_1 \dots \alpha_{i-1} \bullet \alpha_i \alpha_{i+1} \dots \alpha_k] [\alpha_i \to \bullet\beta]$$

$$\overset{u_i}{\rightsquigarrow}^* [A \to \alpha_1 \dots \alpha_{i-1} \bullet \alpha_i \alpha_{i+1} \dots \alpha_k] [\alpha_i \to \beta\bullet] \overset{\varepsilon}{\underset{(R)}{\rightsquigarrow}} [A \to \alpha_1 \dots \alpha_{i-1}\alpha_i \bullet \alpha_{i+1} \dots \alpha_k]$$

En combinant les exécutions on obtient

$$[A \to \bullet\alpha_1\alpha_2 \dots \alpha_k] \overset{u_1}{\rightsquigarrow}^* [A \to \alpha_1 \bullet \alpha_2 \dots \alpha_k] \overset{u_2}{\rightsquigarrow}^* \dots \overset{u_n}{\rightsquigarrow}^* [A \to \alpha_1\alpha_2 \dots \alpha_k\bullet]$$

Et donc on a bien $[A \to \bullet\alpha] \overset{u}{\rightsquigarrow}^* [A \to \alpha\bullet]$, c'est-à-dire $A \to \alpha \in R$ convient.

Théorème 72 (Développement 2, partie 3/3)**.** $\mathcal{L}(A_G) = \mathcal{L}(G)$

Démonstration. On prouve les deux inclusions.

« $\subseteq$ » Soit $u \in \mathcal{L}(A_G)$. Par définition de $\mathcal{L}(A_G)$, $[S' \to \bullet S] \overset{u}{\rightsquigarrow}^0 [S' \to S\bullet]$. Par le lemme 70, on a donc $S = \text{hist}([S' \to S\bullet]) \to^* u$, c'est-à-dire $u \in \mathcal{L}(G)$.

« $\supseteq$ » Soit $u \in \mathcal{L}(G)$. Par définition de $\mathcal{L}(G)$, on a $S' \to^* u$. Par le lemme 71, on a donc $S' \to \alpha \in P$ tel que $[S' \to \bullet\alpha] \overset{u}{\rightsquigarrow}^* [S' \to \alpha\bullet]$. Or la seule règle de la forme $S' \to \alpha$ est $S' \to S$. On a donc $[S' \to \bullet S] \overset{u}{\rightsquigarrow}^* [S' \to S\bullet]$, c'est-à-dire $u \in \mathcal{L}(A_G)$.

□

L'automate des items est (en général) non-déterministe car si une grammaire contient deux règles $X \to \beta$ et $X \to \beta'$, alors il y aura deux transitions (E) : $[Y \to \alpha \bullet B\gamma] \overset{\varepsilon}{\rightarrowtail} [X \to \alpha \bullet B\gamma]\,[B \to \bullet\beta]$ et $[Y \to \alpha \bullet B\gamma] \overset{\varepsilon}{\rightarrowtail} [X \to \alpha \bullet B\gamma]\,[B \to \bullet\beta']$. On définit donc certains classes de grammaires qui permettent de déterminiser, en un certain sens, l'automate des items.

Définition 73. Une grammaire algébrique est dite $LL(k)$ si la connaissance des k symboles suivants du mot permet de choisir la transition (E).

Remarque 74. On peut alors déterminiser l'automate si l'on remplace son unique état par $(\Sigma \sqcup \{ \# \})^k$ (où $\#$ signifie que le mot est fini, et au lieu du mot u, l'automate accepte $u\#^k$) et si l'on se sert des états pour stocker les k symboles suivants avant de les traiter.

Proposition 75. *Une grammaire $LL(k)$ est non ambiguë.*

Définition 76. Une grammaire algébrique est dite $LR(k)$ si, en acceptant temporairement le non-déterminisme causé par les transition (E), la connaissance des k symboles suivants du mot permet de choisir parmi les transitions (L) et (R) possibles.

Remarque 77. Le choix de la règle (R) détermine, a posteriori, la règle (E) choisie. L'analyse LR retarde le plus possible le choix de la règle (E) : jusqu'à la règle (R) correspondante.

Proposition 78. *Une grammaire $LR(k)$ est non ambiguë.*

Références

[1] Jean-Michel Autebert. *Théorie des langages et des automates*. Masson, 1994. isbn : 2-225-84001-6.

[2] Olivier Carton. *Langages formels. Calculabilité et complexité*. Vuibert, 2014. isbn : 978-2-311-01400-6.

[3] Reinhard Wilhelm et Dieter Maurer. *Les compilateurs. théorie. construction. génération.* Masson, 1994. isbn : 2-225-84615-4.

[4] Romain Legendre et François Schwarzentruber. *Compilation : analyse lexicale et syntaxique. Du texte à sa structure en informatique*. Ellipses, 2015. isbn : 9782340-003668.

[5] Rajeev Motwani et Jeffrey D. Ullman John E. Hopcroft. *Introduction to Automata Theory, Languages, and Computation*. Pearson, 2003. isbn : 0-321-21029-8.

[6] Robert W. Floyd et Richard Beigel. *The language of machines. An Introduction to Computability and Formal Languages*. Computer Science Press, 1994. isbn : 0-7167-8266-9.

[7] Michael Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 2013. isbn : 978-81-315-2529-6.