Langages Formels

Licence Informatique – ENS Paris-Saclay

Examen du 25 mai 2022

Toutes les réponses devront être correctement justifiées. Temps : 2h.
Le chiffre en regard d'une question est une indication sur sa difficulté ou sa longueur.

Rappels

Une grammaire algébrique G est donné par $\langle \Sigma, V, P, S \rangle$, pour un alphabet de *terminaux* Σ, un ensemble de variables V, des productions $P \subseteq V \times (V \cup \Sigma)^*$, tous fini, et $S \in V$ le symbole de départ. G est *linéaire* si $P \subseteq V \times (\Sigma^* \cup \Sigma^* V \Sigma^*)$ et *linéaire droite* si $P \subseteq V \times (\Sigma^* \cup \Sigma^* V)$. Un langage est linéaire s'il est engendré par une grammaire linéaire.

Un automate à pile (AAP) $\mathcal{A}$ est donné par $\langle Q, \Sigma, Z, T, q_0 z_0, F \rangle$, avec Q les états, Σ l'alphabet, Z les symboles de pile, $T \subseteq QZ \times (\Sigma \cup \{\varepsilon\}) \times QZ^*$ les transitions (tous fini), $q_0 z_0$ la configuration initiale, et $F \subseteq Q$. Un AAP est *simple* s'il n'a qu'un seul état et *temps-réel* s'il n'a pas de ε-transitions.

Un *alphabet visible* est un triplet $\bar{\Sigma} = \langle \Sigma_c, \Sigma_i, \Sigma_r \rangle$, où $\Sigma_c, \Sigma_i, \Sigma_r$ sont tous des alphabets finis disjoints. Un *automate à pile visible* (AAPV) est un AAP $\langle Q, \bar{\Sigma}, Z, T, q_0 z_0, F \rangle$ avec les restrictions suivantes sur les transitions :

— $T = \langle T_c, T_i, T_r \rangle$, où
— $T_c \subseteq Q \times \Sigma_c \times Q \times (Z \setminus \{z_0\})$;
— $T_i \subseteq Q \times \Sigma_i \times Q$;
— $T_r \subseteq Q \times \Sigma_r \times Z \times Q$.

Le système de transitions associé à un AAPV est défini sur QZ^* avec $q_0 z_0$ comme état initial et $qw \xrightarrow{a} q'w'$ si
— il existe $\langle q, a, q', z \rangle \in T_c$ et $w' = zw$;
— il existe $\langle q, a, q' \rangle \in T_i$ et $w' = w$;
— il existe $\langle q, a, z, q' \rangle \in T_r$ avec $z \neq z_0$ et $zw' = w$;
— il existe $\langle q, a, z_0, q' \rangle \in T_r$ et $w' = w = z_0$.

1 Calcul postfixe

On considère la grammaire G des expressions postfixes définie par

$$S \to a \mid SS+ \mid SS\times$$

sur l'alphabet terminal $\Sigma = \{a, +, \times\}$.

(a) Le langage engendré par la grammaire G est-il rationnel ? [2]

(b) La grammaire G est-elle ambiguë ? [2]

(c) Donner un AAP déterministe, temps-réel, simple qui reconnaît par sommet de pile le langage engendré par G. [3]

Solution :

(a) $\mathcal{L}(G)$ n'est pas rationnel :
- On remarque d'abord que G endendre $a^{n+1}+^n$ pour tout $n \geq 0$. En effet, c'est vrai pour $n = 0$ ($S \to a$), sinon on applique $S \to SS+ \to aS+$, et par récurrence $aS+ \to_G^* aa^{n+1}+^n+$.
- Deuxièmement, tout mot w engendré par G satisfait $|w|_a = |w|_+ + |w|_\times + 1$, ce qu'on prouve par récurrence sur la longueur de la dérivation : La dérivation est soit simplement $S \to a$ (et $w = a$ satisfait le lemme), soit commence par $S \to SS+$ (le cas avec $\times$ étant analogue). Dans le second cas, on obtient la propriété facilement par récurrence sur les mots engendrés par les deux S.
- Ensemble, on conclut que $\mathcal{L}(G) \cap a^*+^* = \{\, a^{n+1}+^n \mid n \geq 0 \,\}$, langage connu comme étant non-rationnel, alors que les rationnels sont clos par intersection.

(b) G n'est pas ambiguë. On observe que toute production génère une lettre différente à la queue du mot. Du coup, tout mot w engendré par G possède une dérivation droite unique car la dernière lettre d'un mot détermine la production à appliquer. Cette dérivation droite détermine un unique arbre de dérivation pour w.

(c) L'AAP suivant convient (afin de faciliter l'explication, le sommet de pile sera à droite) :
- $\langle \{q\}, \Sigma, \{\#, Z, S\}, T, q\#, \{Z\} \rangle$;
- T contient les transitions suivants (on omet q) :

$$\# \xrightarrow{a} Z \qquad Z \xrightarrow{a} ZS \qquad S \xrightarrow{a} SS \qquad S \xrightarrow{+} \varepsilon \qquad S \xrightarrow{\times} \varepsilon$$

L'automate fonctionne dans la manière d'un automate bottom-up. Au début, a doit être reconnu, du coup on remplace # par Z. Après, en consommant un a, on applique tout de suite la réduction $S \to a$ (ce qui revient à empiler un S). Lorsque S est au sommet, la pile possède alors la forme ZS^+. En consommant un $+$, on peut alors appliquer la réduction $S \to SS+$, ce qui revient à dépiler un S. Le cas avec $\times$ est analogue.

2 Grammaires et automates

Soit $\Sigma = \{u, r, d\}$. Un mot de Σ^* designe une trace dans deux dimensions qui commence à $\langle 0, 0\rangle$ et qui évolue selon les lettres. Si un mot w mène jusqu'à $\langle i, j\rangle$, alors le mot wa, $a \in \Sigma$ mène à : $\langle i, j+1\rangle$ si $a = u$ (up) ; $\langle i+1, j\rangle$ si $a = r$ (right) ; $\langle i, j-1\rangle$ si $a = d$ (down).

Un mot est un *skyline* si sa trace :
- ne visite aucun point deux fois ;
- ne visite aucun point $\langle i, j\rangle$ avec $j < 0$;
- se termine à un point $\langle i, 0\rangle$ avec $i > 0$.

P.ex. r et urd ou encore $ururdd$ sont des skyline, mais ε, u, ud ou dru ne le sont pas. L_s designe le langage de skylines.

(a) Donner une grammaire algébrique sur Σ qui génère L_s. [2]
Remarques : Il existe une grammaire avec quatre productions. Indiquez *brièvement* pourquoi votre solution fonctionne, mais inutile de donner une preuve complète.

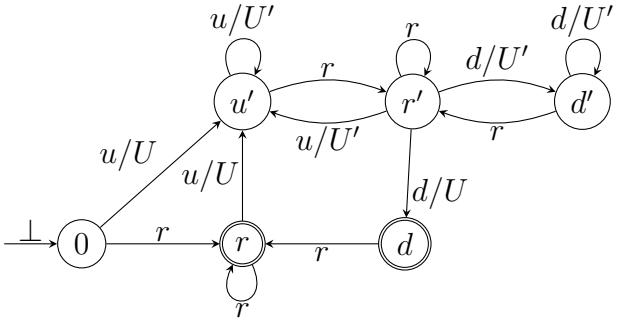
(b) Donner un AAPV pour L_s. [2]

Solution :

(a) La grammaire suivante convient :

$$S \to TrT \mid uSd \qquad T \to S \mid \varepsilon$$

(b) L'automate ci-dessous est déterministe ; ses états mémorisent la dernière lettre lue et si on se trouve on rez-de-chaussée ou par-dessus. Certains ont trouvé un automate non-déterministe avec cinq états en laissant u' et d' faire le travail de r'.



3 Déterminisation

Un AAPV sur $\bar{\Sigma}$ est dit *déterministe* si, pour tout $q \in Q$:
- pour tout $a \in \Sigma_c$ il existe au plus une transition $\langle q, a, q', z\rangle$;
- pour tout $a \in \Sigma_i$ il existe au plus une transition $\langle q, a, q'\rangle$;
- pour toute paire $a \in \Sigma_r, z \in Z$ il existe au plus une transition $\langle q, a, z, q'\rangle$.

Soit $\mathcal{A}$ un AAPV quelconque et w un mot et q, q' deux états. On appelle $\langle q, q'\rangle$ un *sommaire* pour w si $\mathcal{A}$ peut passer de la configuration qz_0 à $q'z_0$ avec w. En général, un mot peut posséder plusieurs sommaires.

(a) On considère l'AAPV [1]

$$\langle\{q_0, q_1, q_2\}, \langle\emptyset, \{a, b\}, \emptyset\rangle, \{z\}, T, q_0 z, \{q_2\}\rangle$$

avec $T_i = \{\langle q_0, a, q_0\rangle, \langle q_0, a, q_1\rangle, \langle q_0, b, q_0\rangle, \langle q_1, b, q_2\rangle\}$. Quels sont les sommaires du mot *aaba* ?

(b) Soit $\mathcal{A}$ un AAPV sur $\bar{\Sigma} = \langle\emptyset, \Sigma_i, \emptyset\rangle$, donc effectivement un automate fini. Expliquer comment construire un automate déterministe équivalent $\mathcal{A}$ qui, en plus de la construction habituelle, calcule l'ensemble des sommaires de son mot en entrée dans son état. Du coup, si Q sont les états de $\mathcal{A}$, l'automate déterministe possède des états $Q' := 2^Q \times 2^{Q\times Q}$. [2]

(c) Généraliser cette construction : étant donné un AAPV $\mathcal{A}$ pour un $\bar{\Sigma}$ quelconque, construire un AAPV $\mathcal{A}'$ déterministe équivalent. [4]

Indications : Dans $\mathcal{A}$, pour un état q et $a \in \Sigma_c$ donné, il peut y avoir plusieurs transitions de la forme $\langle q, a, q', z\rangle \in T_c$. Du coup, en consommant un mot $w \in \bar{\Sigma}^*$, $\mathcal{A}$ peut non seulement atteindre plusieurs états, mais aussi de différentes configurations de pile. Il ne suffit pas de garder 2^Z comme alphabet de pile pour $\mathcal{A}'$, il est plutôt recommandé d'utiliser un alphabet de pile $(Q' \times \Sigma_c) \cup \{z_0\}$.

Solution :

(a) $\langle q_0, q_0\rangle$ et $\langle q_0, q_1\rangle$

(b) Étant donné $\mathcal{A} := \langle Q, \bar{\Sigma}, T, q_0 z_0, F\rangle$ on construit $\mathcal{A}' := \langle Q, \bar{\Sigma}, T, q_0' z_0, F'\rangle$ avec
- $Q := 2^Q \times 2^{Q\times Q}$
- $q_0' := \langle\{q_0\}, \{\,\langle q, q\rangle \mid q \in Q\,\}\rangle$
- $F' := \{\,\langle R, S\rangle \mid R \cap F \neq \emptyset\,\}$
- $T_i' := \big\{\,\langle\langle R, S\rangle, a, \langle R', S'\rangle\rangle \mid R' := \{\,q' \mid q \in R, \langle q, a, q'\rangle \in T_i\,\}, S' := \{\,\langle q, q''\rangle \mid \exists q' : \langle q, q'\rangle \in S, \langle q', a, q''\rangle \in T_i\,\}\,\big\}$,
- $T_c' = T_r' = \emptyset$

Dans un état $\langle R, S\rangle$ de $\mathcal{A}'$, R est l'état habituel de l'automate déterministe pour $\mathcal{A}$, et S représente les sommaires.

(c) Principalement, il s'agit de calculer les sommaires à travers une paire call/return. $\mathcal{A}$ étant non-déterministe, un $a \in \Sigma_c$ peut permettre d'empiler de différents symboles dans $\mathcal{A}$. Dans notre automate déterministe, les règles pour les return doivent relier les dépilements avec les empilements précédents correspondants. Pour cela, il convient de mémoriser, sur la pile, la lettre lue lors du call ainsi que l'état à ce moment-là.
L'alphabet de pile sera donc $Z' := (Q' \times \Sigma_c) \cup \{z_0\}$ et

$$\begin{aligned} T'_c := \big\{\, \langle\langle R, S\rangle, a, \langle R', S'\rangle, \langle R, S, a\rangle\rangle \mid a \in \Sigma_c, \\ R' := \{q' \mid \exists q \in R, \langle q, a, q', z\rangle \in T_c\}, \\ S' := \{\, \langle q, q\rangle \mid q \in Q \,\}\big\} \end{aligned}$$

$$\begin{aligned} T'_r := \big\{\, \langle\langle R, S\rangle, a, \langle R', S', a'\rangle, \langle R'', S''\rangle\rangle \mid a \in \Sigma_r, \\ R'' := \{\, q' \mid \exists q \in R' : \langle q, q'\rangle \in U \,\} \\ S'' := \{\, \langle q, q'\rangle \mid \exists q'' : \langle q, q''\rangle \in S'', \langle q'', q'\rangle \in U \,\} \\ \text{avec } U := \{\, \langle q, q'\rangle \mid \exists q_1, q_2, z : \\ \langle q, a', q_1, z\rangle \in T_c, \langle q_1, q_2\rangle \in S, \langle q_2, a, z, q'\rangle \in T_r \,\}\,\big\} \\ \cup \big\{\, \langle\langle R, S\rangle, a, z_0, \langle R', S'\rangle\rangle \mid \\ R' := \{\, q' \mid \exists q \in R, \langle q, a, z_0, q'\rangle \in T_r \,\}, \\ S' := \{\, \langle q, q''\rangle \mid \exists q' : \langle q, q'\rangle \in S, \langle q', a, z_0, q''\rangle \in T_r \,\}\,\big\} \end{aligned}$$

Ici, U calcule les mouvements du call, du return et de toute étape entre les deux, pour toute paire de call et return utilisant le même symbole de pile et consommant les bonnes lettres de Σ_c et Σ_r. Le résultat de U est ensuite appliqué aux R', S' sauvegardés sur la pile lors du call. Les "dépilements" de z_0 se comportent comme les transitions de T_i qui sont données dans la partie (b).

4 Au pic

Rappel (TD) : Un *automate à un pic* est un AAP tel que dans tout calcul valide, la taille de la pile n'augmente plus une fois qu'elle a diminué. La taille de la pile peut donc augmenter (au sens large) pendant une première partie du calcul, puis elle ne fait que diminuer (au sens large). Un langage est *à un pic* s'il peut être accepté par pile vide par un automate à un pic.

Au TD, le résultat suivant a été montré :

Tout langage linéaire est un langage à un pic.

On va s'intéresser au sens invers de ce résultat.

D'ailleurs, le résultats suivant à été montré au TD :

Soit $G = \langle \Sigma, V_1 \uplus V_2, P_1 \uplus P_2, S \rangle$ une grammaire vérifiant

$$P_1 \subseteq V_1 \times (V_1\Sigma^* \cup \Sigma^*) \quad \text{et} \quad P_2 \subseteq V_2 \times (\Sigma^* V_2 V_1^* \cup \Sigma^*).$$

Pour tout $x \in V_1 \cup V_2$, $\mathcal{L}_G(x)$ est un langage linéaire.

Soit $\mathcal{A} = \langle Q, \Sigma, Z, T, q_0 z_0, F \rangle$ un AAP quelconque. Pour $p, q \in Q$ et $z \in Z$, on définit :

$$K_{pzq} := \{\, w \in \Sigma^* \mid pz \overset{w}{\to}{}^*_{\mathcal{A}}\, q \text{ avec un taille de pile toujours} \leq 1 \,\}$$

$$L_{pzq} := \{\, w \in \Sigma^* \mid pz \overset{w}{\to}{}^*_{\mathcal{A}}\, q \text{ calcul à un pic} \,\}$$

(a) Montrer que le langage K_{pzq} peut être engendré par une grammaire linéaire droite. [1]

(b) Construire une grammaire vérifiant les mêmes conditions que G et qui engendre les langages L_{pzq} avec V_2 et les langages K_{pzq} avec V_1. Montrer que tout langage à un pic est un langage linéaire. [3]

Solution :

(a) Un AAP avec taille de pile limitée n'a qu'un nombre fini de configurations, il est donc équivalent à un automate fini. Du coup K_{pzq} est reconnaissable, et tout langage reconnaissable est engendré par un grammaire linéaire droite.

(b) Définissons $J_{pzry} := \{\, w \in \Sigma^* \mid pz \overset{w}{\to}{}^*_{\mathcal{A}}\, ry \text{ avec un taille de pile toujours } 1 \,\}$.
Nous allons utiliser trois types de variables :
— $J := \{\, \langle pzry \rangle \mid p, r \in Q, y, z \in Z \,\}$, où $\langle pzry \rangle$ engendre J_{pzry} ;
— $K := \{\, \langle pzq \rangle \mid p, q \in Q, z \in Z \,\}$, où $\langle pzq \rangle$ engendre K_{pzq} ;
— $L := \{\, [pzq] \mid p, q \in Q, z \in Z \,\}$, où $[pzq]$ engendre L_{pzq} ;
Pour les L_{pzq}, la construction marche quasiment comme dans la construction d'une grammaire à partir d'un AAP. Pour K_{pzq} on dévine la transition $ry \overset{a}{\to} q$ qui fait le dépilement finale, puis on génère J_{pzry} de droite à gauche. Formellement, mettons $V_1 := J \cup K$, $V_2 := L$, $P_1 := P_J \cup P_K$, $P_2 := P_L$, où dans le suivant a peut être une lettre ou ε :

— $P_J = \bigcup_{p\in P, z\in Z}\{\, \langle pzry\rangle \to \langle pzqx\rangle a \mid qx \xrightarrow{a}_{\mathcal{A}} ry \,\} \cup \{\langle pzpz\rangle \to \varepsilon\}$;
— $P_K = \{\, \langle pzq\rangle \to \langle pzry\rangle a \mid ry \xrightarrow{a}_{\mathcal{A}} q \,\}$;
— $P_L = \{\, [pzq] \to a \mid pz \xrightarrow{a}_{\mathcal{A}} q \,\} \cup \{\, [pzq] \to a[ryq] \mid pz \xrightarrow{a}_{\mathcal{A}} ry \,\}$
$\cup \{\, [pzq] \to a[rz_1q_1]\langle q_1z_2q_2\rangle \cdots \langle q_{k-1}z_kq\rangle \mid pz \xrightarrow{a}_{\mathcal{A}} rz_1z_2\cdots z_k, q_1, \ldots q_{k-1} \in Q \,\}$.

Finalement $\mathcal{L}(\mathcal{A}) = \bigcup_{q\in Q} \mathcal{L}_G([q_0z_0q])$, donc une union de langages linéares.