Langages Formels

Licence Informatique – ENS Cachan

Examen du 26 mai 2016

durée 3 heures

Document autorisé : polycopié du cours.
Toutes les réponses devront être correctement justifiées.
Le chiffre en regard d'une question est une indication sur sa difficulté ou sa longueur.

1 Langages

On fixe un alphabet Σ. On note $\mathsf{Pref}(w)$ l'ensemble des préfixes d'un mot $w \in \Sigma^*$. Etant donné un langage $L \subseteq \Sigma^*$, on définit $\Phi(L) = \{w \in \Sigma^* \mid \mathsf{Pref}(w) \subseteq L\}$. On souhaite étudier la nature du langage $\Phi(L)$ en fonction de la nature de L.

[1] **a)** On suppose que $\Sigma = \{a\}$ ne contient qu'une lettre. Caractériser l'ensemble des langages $\Phi(L)$ que l'on peut obtenir.

[3] **b)** Montrer que
- si L est reconnaissable, alors $\Phi(L)$ est reconnaissable.
- si L est apériodique, alors $\Phi(L)$ est apériodique.

[2] **c)** Montrer que le langage $K = \{a^n b a^n b a^n b \mid n > 0\}$ n'est pas algébrique.

[3] **d)** Montrer que si L est algébrique, alors $\Phi(L)$ n'est pas nécessairement algébrique.
Indication : Construire un langage algébrique L tel que $\Phi(L) \cap a^+ba^+ba^+b = K$.

2 Logique

On fixe un alphabet Σ. Etant donné un langage $L \subseteq \Sigma^*$, on définit l'ensemble $\Psi(L)$ des mots $w \in \Sigma^*$ tels que pour toute factorisation $w = v_1 w'$ il existe une factorisation $w' = v_2 v_3$ telle que $v_3 v_2 v_1 \in L$.

[4] **a)** Soit $\alpha \in \mathrm{MSO}(\Sigma, <)$ une formule close. Construire une formule close $\beta \in \mathrm{MSO}(\Sigma, <)$ telle que $\mathcal{L}(\beta) = \Psi(\mathcal{L}(\alpha))$. En déduire que si L est reconnaissable (resp. apériodique) alors $\Psi(L)$ est reconnaissable (resp. apériodique).

3 Fonctions séquentielles

Soient $\mathcal{A}$ et $\mathcal{B}$ deux automates séquentiels définissant les fonctions $f = [\![\mathcal{A}]\!] \colon A^* \to B^*$ et $g = [\![\mathcal{B}]\!] \colon A^* \to B^*$.

[2] **a)** Soit $X = \mathrm{dom}(f) \cap \mathrm{dom}(g)$. Montrer que l'on peut décider si f et g coincident sur X, i.e., $f(u) = g(u)$ pour tout $u \in X$.

[2] **b)** On suppose que $X = \mathrm{dom}(f) \cap \mathrm{dom}(g) = \emptyset$. On note h la fonction "union" définie sur $\mathrm{dom}(f) \cup \mathrm{dom}(g)$ par $h(u) = f(u)$ si $u \in \mathrm{dom}(f)$ et $h(u) = g(u)$ si $u \in \mathrm{dom}(g)$. Montrer que h n'est pas nécessairement séquentielle.

[1] **c)** Montrer que l'on peut décider si la fonction f définie par $\mathcal{A}$ est surjective.

[4] **d)** Soit $Y \subseteq B^*$ l'ensemble des mots $v \in B^*$ qui ont au moins deux antécédents par f : il existe $u_1, u_2 \in A^*$ tels que $u_1 \neq u_2$ et $f(u_1) = v = f(u_2)$. En utilisant les propriétés des fonctions séquentielles et des langages reconnaissables (et donc sans construire un automate), montrer que le langage Y est reconnaissable. En déduire que l'on peut décider si la fonction f définie par $\mathcal{A}$ est injective.

Langages Formels

Licence Informatique – ENS Cachan

Partiel du 24 mars 2016

durée 2 heures

Document autorisé : polycopié du cours.
Toutes les réponses devront être correctement justifiées.
Le chiffre en regard d'une question est une indication sur sa difficulté ou sa longueur.

1 Relations reconnaissables

Un entier est codé en binaire en commençant par le bit de poids faible. Soit $B = \{0, 1\}$ l'alphabet binaire. On note $\overline{w}^2$ l'entier codé par le mot $w \in B^*$. Par exemple, $\overline{1011}^2 = 13$ (en base 10), $\overline{01100}^2 = 6$ et $\overline{\varepsilon}^2 = 0$. On code aussi un k-uple d'entiers $(n_1, \ldots, n_k)$ par un mot W sur l'alphabet B^k : l'entier n_i est codé par le mot w_i obtenu en projetant W sur la i-ème composante. On note $\overline{W}^2 = (n_1, \ldots, n_k)$. Par exemple, le triplet $(5, 2, 16)$ est codé par le mot $W = (1,0,0)(0,1,0)(1,0,0)(0,0,0)(0,0,1)$ sur l'alphabet B^3, puisque $\overline{10100}^2 = 5$, $\overline{01000}^2 = 2$ et $\overline{00001}^2 = 16$.

[1] **a)** Montrer que le langage $L_1 = \{W \in (B^2)^* \mid \overline{W}^2 = (n_1, n_2) \text{ avec } n_1 \leq n_2\}$ est reconnaissable en donnant une expression rationnelle pour L_1.

[1] **b)** Montrer que le langage $L_2 = \{W \in (B^2)^* \mid \overline{W}^2 = (n_1, n_2) \text{ avec } 3n_1 = n_2\}$ est reconnaissable en donnant un automate déterministe ayant au plus 3 états pour $L2$.

[1] **c)** Montrer que le langage $L_3 = \{W \in (B^2)^* \mid \overline{W}^2 = (n_1, n_2) \text{ avec } 3n_1 \leq n_2\}$ est reconnaissable en utilisant des propriétés de clôture des langages reconnaissables. On *ne* donnera *pas* explicitement un automate ou une expression rationnelle pour L_3.

2 Grammaires

On considère la grammaire $G = (\Sigma, V, P, S)$ avec $\Sigma = \{a, b\}$, $V = \{S\}$ et les trois règles suivantes : $S \to Sa + SS + b$.

[1] **a)** Construire une grammaire $G' = (\Sigma, V', P', S)$ équivalente à G, i.e., telle que $\mathcal{L}_G(S) = \mathcal{L}_{G'}(S)$, et telle que $P' \subseteq V' \times \Sigma(\Sigma \cup V')^*$.

3 Automate à pile

On considère une extension des automates à pile dans laquelle les transitions sont conditionnées par des contraintes rationnelles sur le contenu de pile.

Un automate à pile généralisé est un tuple $\mathcal{A} = (Q, \Sigma, Z, \texttt{Push}, \texttt{Pop}, q_0, F)$ avec Q un ensemble fini d'états, Σ l'alphabet d'entrée, Z l'alphabet de pile, $q_0 \in Q$ l'état initial, $F \subseteq Q$ l'ensemble des états acceptants et les ensembles *finis* de transitions

$$\begin{aligned} \texttt{Push} &\subseteq Q \times \mathrm{Rat}(Z^*) \times (\Sigma \cup \{\varepsilon\}) \times Z \times Q \\ \texttt{Pop} &\subseteq Q \times Z \times \mathrm{Rat}(Z^*) \times (\Sigma \cup \{\varepsilon\}) \times Q \end{aligned}$$

Comme pour les automates à pile classiques, la sémantique est définie par un système de transitions infini $\mathcal{T}$ dont les configurations sont de la forme $qh \in QZ^*$ avec $q \in Q$ et $h \in Z^*$. La configuration initiale est q_0. L'ensemble des configurations acceptantes est FZ^*. Les transitions de $\mathcal{T}$ sont définies par :

— $qh \xrightarrow{a} q'zh$ s'il existe $(q, K, a, z, q') \in \texttt{Push}$ avec $h \in K$,
— $qzh \xrightarrow{a} q'h$ s'il existe $(q, z, K, a, q') \in \texttt{Pop}$ avec $zh \in K$.

L'objectif est de montrer que les automates à pile généralisés ont le même pouvoir d'expression que les automates à pile classiques.

[1] **a)** Etant donnés des langages rationnels $K_1, \ldots, K_n$ sur l'alphabet Z, montrer que l'on peut construire un automate fini déterministe et complet $\mathcal{B} = (P, Z, \delta, p_0)$ et des ensembles d'états $P_1, \ldots, P_n \subseteq P$ tels que pour tout $1 \leq i \leq n$, le langage K_i est reconnu par $\mathcal{B}$ en utilisant $P_i \subseteq P$ comme ensemble d'états acceptants :

$$K_i = \{h \in Z^* \mid \delta(p_0, h) \in P_i\}\,.$$

[1] **b)** Etant donné un automate à pile généralisé $\mathcal{A}$, construire un automate à pile classique $\mathcal{A}'$ équivalent, i.e., tel que $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.
Il faut bien sûr prouver que $\mathcal{A}$ et $\mathcal{A}'$ acceptent le même langage.

[1] **c)** Expliquer comment généraliser à des transitions qui peuvent empiler ou dépiler un nombre arbitraire de symboles :

$$\begin{aligned} \texttt{Push} &\subseteq Q \times \mathrm{Rat}(Z^*) \times (\Sigma \cup \{\varepsilon\}) \times Z^* \times Q \\ \texttt{Pop} &\subseteq Q \times Z^* \times \mathrm{Rat}(Z^*) \times (\Sigma \cup \{\varepsilon\}) \times Q \end{aligned}$$

Les ensembles de transitions `Push` et `Pop` sont bien sûr toujours finis.
Pour cette question, on ne demande pas une construction détaillée.

Langages Formels

Licence Informatique – ENS Cachan

Examen du 26 mai 2015

durée 3 heures

Document autorisé : polycopié du cours.
Toutes les réponses devront être correctement justifiées.
Le chiffre en regard d'une question est une indication sur sa difficulté ou sa longueur.

1 Mots et arbres

Soit Σ un alphabet et $L \subseteq \Sigma^*$ un langage. La *racine* de L est le langage $\sqrt{L} = \{v \in \Sigma^* \mid vv \in L\}$.

[1] **a)** Montrer que si L est un langage rationnel alors $\sqrt{L}$ est aussi rationnel.

[3] **b)** Montrer que $L = \{a^n b^p a^{2p} b^q a^n \mid n, p, q \geq 1\}$ est un langage linéaire.
Montrer que $\sqrt{L}$ n'est pas algébrique.

[4] **c)** Soit Σ un alphabet et $L \subseteq T_p(\Sigma)$ un langage d'arbres. La *racine* de L est le langage

$$\sqrt{L} = \{t \in T_p(\Sigma) \mid \exists s \in T_{p,\Box}(\Sigma), \exists a \in \Sigma \text{ tels que } t = s \cdot a \text{ et } s \cdot s \cdot a \in L\}\,.$$

Montrer que si L est reconnaissable alors $\sqrt{L}$ est aussi reconnaissable.

[3] **d)** Soit $w = a_1 \cdots a_n \in \Sigma^*$ un mot avec $a_k \in \Sigma$ pour $1 \leq k \leq n$. On note $w[i,j] = a_i \cdots a_j$ le facteur de w défini par $1 \leq i \leq j \leq n$. Dans la suite, $u \in \Sigma^*$ et $L \subseteq \Sigma^*$ est un langage *fini*.
Définir une formule du premier ordre $\varphi_u(x,y) \in \mathrm{FO}(\Sigma, <)$ ayant deux variables libres x et y, telle que pour tout $w \in \Sigma^*$ et $1 \leq i \leq j \leq |w|$ on a

$$w[i,j] = u \qquad \Longleftrightarrow \qquad w, x \mapsto i, y \mapsto j \models \varphi_u(x,y)\,.$$

Définir une formule du premier ordre $\varphi_L(x,y) \in \mathrm{FO}(\Sigma, <)$ ayant deux variables libres x et y, telle que pour tout $w \in \Sigma^*$ et $1 \leq i \leq j \leq |w|$ on a

$$w[i,j] \in L \qquad \Longleftrightarrow \qquad w, x \mapsto i, y \mapsto j \models \varphi_L(x,y)\,.$$

Définir une formule close $\varphi_{L^*} \in \mathrm{MSO}(\Sigma, <)$ telle que pour tout $w \in \Sigma^*$ on a

$$w \in L^* \qquad \Longleftrightarrow \qquad w \models \varphi_{L^*}\,.$$

2 Langages linéaires et automates à un pic

Un *automate à un pic* est un automate à pile 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.

[3] **a)** Montrer que le langage $L = \{a^n b^n \mid n > 0\} \cup \{a^n b^{2n} \mid n > 0\}$ est un langage linéaire.
Montrer que L est un langage à un pic.
Montrer que le langage $K = \{ba^{i_1}ba^{i_2}b \cdots ba^{i_n}b \mid n > 0 \text{ et } \exists j, i_j \neq j\}$ est un langage à un pic.

[3] **b)** Montrer que tout langage linéaire est un langage à un pic.

[3] **c)** Soit $G = (\Sigma, V_1 \cup V_2, P_1 \cup P_2)$ une grammaire vérifiant :

$$\begin{aligned} &V_1 \cap V_2 = \emptyset \\ &P_1 \subseteq V_1 \times (V_1\Sigma^* \cup \Sigma^*) \\ &P_2 \subseteq V_2 \times (\Sigma^* V_2 V_1^* \cup \Sigma^*) \end{aligned}$$

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

Soit $\mathcal{A} = (Q, \Sigma, Z, T, (q_0, z_0))$ un automate à pile arbitraire.
Pour $p, q \in Q$ et $z \in Z$ on définit les langages

$$\begin{aligned} K_{zp,q} &= \{w \in \Sigma^* \mid \exists\ zp \xrightarrow{w} q \text{ calcul dans } \mathcal{A} \text{ avec une pile de taille toujours } \leq 1\}, \\ L_{zp,q} &= \{w \in \Sigma^* \mid \exists\ zp \xrightarrow{w} q \text{ calcul à un pic dans } \mathcal{A}\}. \end{aligned}$$

[2] **d)** Montrer que le langage $K_{zp,q}$ peut être engendré par une grammaire linéaire *droite*.

[3] **e)** Construire une grammaire vérifiant les conditions de la question **(c)** et qui engendre les langages $L_{zp,q}$ avec les variables de V_2 et les langages $K_{zp,q}$ avec les variables de V_1.
Montrer que tout langage à un pic est un langage linéaire.

Langages Formels

Licence Informatique – ENS Cachan

Partiel du 26 mars 2015

durée 3 heures

Document autorisé : polycopié du cours.
Toutes les réponses devront être correctement justifiées.
Le chiffre en regard d'une question est une indication sur sa difficulté ou sa longueur.

1 Mots

Soient A et B deux alphabets et $M = A^* \times B^*$. On définit un produit binaire sur M par $(u_1, u_2) \cdot (v_1, v_2) = (u_1 v_1, u_2 v_2)$ pour $(u_1, u_2), (v_1, v_2) \in M$.

[1] **a)** Montrer que $(M, \cdot, 1_M)$ est un monoïde avec un élément neutre 1_M que l'on précisera. Montrer que M est engendré par $X = (A \times \{\varepsilon\}) \cup (\{\varepsilon\} \times B)$.

Un M-automate est un quadruplet $\mathcal{A} = (Q, T, I, F)$ avec Q l'ensemble fini d'états, $I, F \subseteq Q$, et $T \subseteq Q \times X \times Q$ l'ensemble des transitions. Un couple $(u, v) \in M$ est accepté par $\mathcal{A}$ s'il existe un calcul $q_0 \xrightarrow{x_1} q_1 \xrightarrow{x_2} q_2 \cdots \xrightarrow{x_n} q_n$ avec $q_0 \in I$, $q_n \in F$ et $(u, v) = x_1 \cdot x_2 \cdots x_n$. On note $\mathcal{R}(\mathcal{A}) \subseteq M$ l'ensemble des couples acceptés par $\mathcal{A}$.

Une relation $R \subseteq M$ est rationnelle s'il existe un M-automate $\mathcal{A}$ tel que $R = \mathcal{R}(\mathcal{A})$. On note $\mathsf{Rat}(M)$ l'ensemble des relations rationnelles sur M.

[3] **b)** On suppose $a, c \in A$ et $b, d \in B$.
Montrer que la relation $R_1 = \{(a^n, b^n) \mid n \geq 0\}$ est rationnelle.
Montrer que la relation $R_2 = \{(a^n c^m, b^m d^n) \mid n, m \geq 0\}$ n'est pas rationnelle.

Le domaine d'une relation $R \subseteq M$ est $\mathrm{dom}(R) = \{u \in A^* \mid \exists v \in B^*,\ (u, v) \in R\} \subseteq A^*$.

[2] **c)** Montrer que le domaine d'une relation rationnelle est un langage rationnel : si $R \in \mathsf{Rat}(M)$ alors $\mathrm{dom}(R) \in \mathsf{Rat}(A^*)$.

L'image d'un langage $L \subseteq A^*$ par une relation $R \subseteq M$ est $R(L) = \{v \in B^* \mid \exists u \in L,\ (u, v) \in R\}$.

[3] **d)** Montrer que l'image d'un langage rationnel par une relation rationnelle est un langage rationnel : si $R \in \mathsf{Rat}(M)$ et $L \in \mathsf{Rat}(A^*)$ alors $R(L) \in \mathsf{Rat}(B^*)$.

[5] **e)** Soient C un nouvel alphabet, $\varphi\colon C^* \to A^*$ et $\psi\colon C^* \to B^*$ deux morphismes. Pour $K \subseteq C^*$ on définit $(\varphi, \psi)(K) = \{(\varphi(w), \psi(w)) \mid w \in K\} \subseteq M$.
Montrer que si $K \in \mathsf{Rat}(C^*)$ alors $(\varphi, \psi)(K) \in \mathsf{Rat}(M)$.
Soit $R \in \mathsf{Rat}(M)$. Montrer qu'il existe un alphabet C, deux morphismes $\varphi\colon C^* \to A^*$ et $\psi\colon C^* \to B^*$, et un langage $K \in \mathsf{Rat}(C^*)$ tels que $R = (\varphi, \psi)(K)$.

[5] **f)** Soit $R \subseteq M$ une relation. On définit le langage $\theta(R) = \{u\tilde{v} \mid (u,v) \in R\} \subseteq A^*B^*$ (où $\tilde{v}$ dénote le miroir du mot v).
Soit $R \in \mathsf{Rat}(M)$. Montrer que $\theta(R)$ peut être engendré par une grammaire linaire.
On suppose que $A = B$. Soit $L \subseteq A^*$ un langage linéaire. Montrer qu'il existe une relation rationnelle $R \subseteq M$ telle que $L = \theta(R)$.
On suppose que $A \cap B = \emptyset$. Montrer qu'il existe un langage linéaire $L \subseteq A^*B^*$ tel que pour toute relation rationnelle $R \subseteq M$, on a $L \neq \theta(R)$.

2 Arbres

Soit Σ un alphabet, $\# \notin \Sigma$ et $\Sigma_\# = \Sigma \cup \{\#\}$.

Pour un arbre t' sur l'alphabet $\Sigma_\#$ on définit les propriétés suivantes :

(P-1) les nœuds internes de t' sont étiquetés $\#$.

(P-2) t' n'a pas de nœud interne unaire.

(P-3) tous les nœuds internes de t' sont binaires.

Un langage d'arbres L satisfait une propriété si tous les arbres de L satisfont cette propriété.

Pour un langage d'arbres L, on note $\mathsf{Fr}(L)$ l'ensemble des frontières des arbres de L.

[6] **a)** Soit $L \subseteq T_p(\Sigma)$ un langage d'arbres reconnaissable.
Montrer qu'il existe un langage d'arbres $L' \subseteq T_p(\Sigma_\#)$ reconnaissable tel que $\mathsf{Fr}(L') = \mathsf{Fr}(L)$ et vérifiant la propriété (P-1).
Montrer qu'il existe un langage d'arbres $L' \subseteq T_p(\Sigma_\#)$ reconnaissable tel que $\mathsf{Fr}(L') = \mathsf{Fr}(L)$ et vérifiant les propriétés (P-1) et (P-2).
Montrer qu'il existe un langage d'arbres $L' \subseteq T_2(\Sigma_\#)$ reconnaissable tel que $\mathsf{Fr}(L') = \mathsf{Fr}(L)$ et vérifiant les propriétés (P-1) et (P-3).

Langages Formels

Licence Informatique – ENS Cachan

Examen du 26 mai 2014
durée 3 heures

Document autorisé : polycopié du cours.
Toutes les réponses devront être correctement justifiées.
Le chiffre en regard d'une question est une indication sur sa difficulté ou sa longueur.

1 Mélanges de mots

Soit Σ un alphabet et $u, v \in \Sigma^*$ deux mots. Le mélange de u et v est le langage

$$u \sqcup\!\sqcup v = \{u_1 v_1 \cdots u_n v_n \mid n \geq 0,\ u_i, v_i \in \Sigma \cup \{\varepsilon\},\ u = u_1 \cdots u_n \text{ et } v = v_1 \cdots v_n\}.$$

Par exemple, $ab \sqcup\!\sqcup ca = \{abca, acba, caba, acab, caab\}$.

Cette opération est étendue aux langages en posant $K \sqcup\!\sqcup L = \bigcup_{u \in K, v \in L} u \sqcup\!\sqcup v$ pour $K, L \subseteq \Sigma^*$.

[2] **a)** Soit $\Sigma = \{a, b, c\}$ un alphabet. Construire l'automate minimal du langage $\Sigma^* ab \Sigma^* \sqcup\!\sqcup \Sigma^* b \Sigma^*$.

Dans la suite, l'alphabet Σ est de nouveau arbitraire. On introduit une copie $\overline{\Sigma}$ de l'alphabet Σ et on note $\overline{a} \in \overline{\Sigma}$ la copie de la lettre $a \in \Sigma$. On considère les morphismes Π, Π_1 et Π_2 de $(\Sigma \cup \overline{\Sigma})^*$ dans Σ^* définis pour $a \in \Sigma$ par $\Pi(a) = \Pi(\overline{a}) = a$, $\Pi_1(a) = \Pi_2(\overline{a}) = a$, et $\Pi_1(\overline{a}) = \Pi_2(a) = \varepsilon$.

[3] **b)** Montrer que $K \sqcup\!\sqcup L = \Pi(\Pi_1^{-1}(K) \cap \Pi_2^{-1}(L))$.
En déduire que si K est rationnel (resp. linéaire ou algébrique) et que L est rationnel alors $K \sqcup\!\sqcup L$ est aussi rationnel (resp. linéaire ou algébrique).

[2] **c)** Le langage $\{a^n b^n \mid n \geq 0\} \sqcup\!\sqcup \{c^p d^p \mid p \geq 0\}$ est-il algébrique ? Justifier la réponse.

On suppose dans la suite que Σ est partitionné en $\Sigma = \Sigma_1 \cup \Sigma_2$ avec $\Sigma_1 \cap \Sigma_2 = \emptyset$. On note π_1 et π_2 les projections de Σ^* sur Σ_1^* et Σ_2^* respectivement. On considère deux langages $L_1 \subseteq \Sigma_1^*$ et $L_2 \subseteq \Sigma_2^*$ et on note $L = L_1 \sqcup\!\sqcup L_2$.

[2] **d)** Montrer que $L = \{u \in \Sigma^* \mid \pi_1(u) \in L_1 \wedge \pi_2(u) \in L_2\}$.
Pour $i \in \{1, 2\}$, on suppose que L_i est reconnu par un morphisme $\varphi_i : \Sigma_i^* \to M_i$ où M_i est un monoide fini. Construire un monoide fini M et un morphisme $\varphi : \Sigma^* \to M$ qui reconnaît L.

[2] **e)** Soit $u \in \Sigma^*$. On note $u_1 = \pi_1(u)$ et $u_2 = \pi_2(u)$. Montrer que $u^{-1}L = u_1^{-1}L_1 \sqcup\!\sqcup u_2^{-1}L_2$. Exprimer le nombre n de résiduels de L en fonction des nombres n_1 et n_2 de résiduels de L_1 et L_2. Justifier votre réponse.
Attention : ne pas oublier que L_1 et L_2 peuvent avoir un résiduel vide.

2 Analyse LL

On considère la grammaire G définie par

$$\begin{aligned} \textbf{Stmt} &\to \texttt{id} \mid \textbf{Var} := \textbf{Exp} \\ \textbf{Var} &\to \texttt{id} \mid \textbf{Var}[\textbf{Exp}] \\ \textbf{Exp} &\to \texttt{num} \mid \textbf{Var} \end{aligned}$$

sur l'alphabet terminal $\Sigma = \{\texttt{id}, \texttt{num}, :=, [,]\}$ et l'alphabet non terminal $V = \{\textbf{Stmt}, \textbf{Var}, \textbf{Exp}\}$.

[5] **a)** Pour chaque variable $x \in V$, calculer $\mathrm{First}_2(x)$ et $\mathrm{Follow}_2(x)$.
Montrer que G n'est pas fortement LL(2).
Existe-t-il k tel que G soit une grammaire LL(k) ?

3 Fonctions séquentielles

On considère les mots sur l'alphabet binaire $\Sigma = \{0, 1\}$ qui codent des entiers en binaire en commençant par le bit de poids faible : le mot $u = a_0 a_1 \cdots a_n$ code l'entier $\overline{u}^2 = \sum_{i=0}^{n} a_i 2^i$.

[3] **a)** On considère la fonction $g\colon \Sigma^* \to \Sigma^*$ qui associe à un mot $u \in \Sigma^*$ le mot $v \in \Sigma^*$ tel que $|v| = |u|$ et $\overline{v}^2 = \overline{u}^2 \,\texttt{div}\, 3$.
Montrer que la fonction g n'est pas séquentielle.

[3] **b)** On considère la fonction $h\colon \Sigma^* \to \Sigma^*$ qui coïncide avec g sur les mots $u \in \Sigma^*$ tels que $\overline{u}^2 \,\texttt{mod}\, 3 = 0$, et qui n'est pas définie sur les mots $u \in \Sigma^*$ tels que $\overline{u}^2 \,\texttt{mod}\, 3 \neq 0$.
Montrer que la fonction h est séquentielle. Construire son automate séquentiel minimal.

4 Logique et automates d'arbres

On considère la logique du premier ordre $\mathrm{FO}(\Sigma, <, \downarrow_1, \downarrow_2)$ sur les arbres binaires étiquetés par l'alphabet Σ ($x < y$ signifie que le nœud x est un ancêtre du nœud y).

[2] **a)** Donner des formules closes du premier ordre pour les propriétés suivantes :

ψ_1 la concaténation des étiquettes de chaque branche est un mot de $(ab)^+$,

ψ_2 chaque branche issue d'un nœud étiqueté a comporte au moins un nœud étiqueté b.

[4] **b)** On considère la formule φ

$$\forall x\, \forall z \left[\big(P_a(x) \wedge x < z \wedge \neg \exists y\, z < y\big) \to \exists y \left(x < y \leq z \wedge P_c(y) \wedge \forall z\, (x < z < y \to P_b(z)) \right) \right]$$

Donner un automate d'arbres $\mathcal{A}$ déterministe (ascendant) et complet qui reconnaît $\mathcal{L}(\varphi)$.
Il faut bien sûr prouver que $\mathcal{A}$ reconnaît $\mathcal{L}(\varphi)$.

Langages Formels

Licence Informatique – ENS Cachan

Partiel du 27 mars 2014
durée 2 heures

Document autorisé : polycopié du cours.
Toutes les réponses devront être correctement justifiées.
Le chiffre en regard d'une question est une indication sur sa difficulté ou sa longueur.

1 Mots

Soit Σ un alphabet et $L \subseteq \Sigma^*$ un langage. On définit le *tiers médian* de L par

$$\texttt{TM}(L) = \{v \in \Sigma^* \mid \exists u, w \in \Sigma^* \text{ tels que } uvw \in L \text{ et } |u| = |v| = |w|\}\,.$$

[3] **a)** On suppose que L est reconnu par un automate (non déterministe) $\mathcal{A} = (Q, \Sigma, T, I, F)$. Montrer que $\texttt{TM}(L)$ peut être reconnu par un automate (non déterministe) $\mathcal{A}' = (Q', \Sigma, T', I', F')$ tel que $|Q'| \leq |Q|^3$. Il faut bien sûr définir l'automate $\mathcal{A}'$ et prouver qu'il reconnaît bien $\texttt{TM}(L)$.

[3] **b)** On suppose maintenant que L est reconnu par un morphisme $h\colon \Sigma^* \to M$ où M est un monoïde fini. Montrer que $\texttt{TM}(L)$ peut être reconnu par un morphisme $h'\colon \Sigma^* \to M'$ où M' est un monoïde tel que $|M'| = 2^{\mathcal{O}(|M|)}$. Il faut bien sûr définir le monoïde, le morphisme et prouver qu'il reconnaît bien $\texttt{TM}(L)$.

On considère maintenant l'opération inverse :

$$\texttt{TM}^{-1}(L) = \{uvw \in \Sigma^* \mid v \in L \text{ et } |u| = |v| = |w|\}\,.$$

est l'ensemble des mots dont le tiers médian est dans L.

[2] **c)** On suppose que $L \subseteq \Sigma^*$ est un langage reconnaissable.
Le langage $\texttt{TM}^{-1}(L)$ est-il nécessairement reconnaissable ?
Le langage $\texttt{TM}^{-1}(L)$ est-il nécessairement algébrique ?
Mêmes questions en supposant que $|\Sigma| = 1$.

[3] **d)** Soient $\varphi_1, \varphi_2 \in \text{MSO}(\Sigma, <)$ deux formules closes qui définissent des langages $L_i = \mathcal{L}(\varphi_i) \in \Sigma^+$ $(i = 1, 2)$. Construire à partir de φ_1 et φ_2 une formule φ qui définit la concaténation $L_1 \cdot L_2$.

[3] **e)** Les langages $L_1 = \{a^n b^n a^n \mid n \geq 0\}$ et $L_2 = \{a^p b^n a^q \mid n = p+q \geq 0\}$ sont-ils reconnaissables ? Sont-ils algébriques ?

[4] **f)** Donner une grammaire algébrique qui engendre le langage $L_3 = \{w \in \{a, b\}^* \mid |w|_a = 2|w|_b\}$. Prouver que votre grammaire engendre bien le langage L_3.

2 Arbres

On considère l'ensemble $T(\mathcal{F})$ des termes construits avec un symbole binaire $\mathcal{F}_2 = \{c\}$ et deux constantes $\mathcal{F}_0 = \{a, b\}$.

On note $L = \{w \in \{a, b\}^+ \mid |w|_a = |w|_b\}$.

[2] **a)** L'ensemble $\{t \in T(\mathcal{F}) \mid \mathrm{Fr}(t) \in L\}$ des termes dont la frontière est dans le langage L est-il reconnaissable ?

[4] **b)** Construire un automate d'arbres $\mathcal{A}$ tel que $\mathcal{L}(\mathcal{A}) \subseteq T(\mathcal{F})$ et $\mathrm{Fr}(\mathcal{L}(\mathcal{A})) = L$.

Langages Formels

Licence Informatique – ENS Cachan

Examen du 27 mai 2013

durée 3 heures

Document autorisé : polycopié du cours.
Toutes les réponses devront être correctement justifiées.
Le chiffre en regard d'une question est une indication sur sa difficulté ou sa longueur.

1 Logique et automates d'arbres

On fixe $A_p = \{d_1, \ldots, d_p\}$ un alphabet de directions et Σ un alphabet. On considère la logique du premier ordre $\mathrm{FO}(\Sigma, <, \downarrow_1, \ldots, \downarrow_p)$ sur les arbres ($x < y$ signifie que le nœud x est un ancêtre du nœud y).

[2] **a)** Donner des formules du premier ordre ayant exactement une variable libre pour les propriétés suivantes :

$\psi_1(x)$ tous les fils de x sont étiquetés par la lettre a,

$\psi_2(x)$ il existe une branche issue de x dont la suite des étiquettes (à partir de x) est dans $a^*b\Sigma^*$,

$\psi_3(x)$ toutes les branches issues de x comportent au moins un nœud étiqueté b,

$\psi_4(x)$ il existe une branche issue de x telle que tous les nœuds de cette branche qui sont étiquetés a satisfont ψ_3.

On considère l'alphabet $\Sigma' = \Sigma \times \{0,1\}$. Pour un arbre $t \in T_p(\Sigma')$ on notera $t_1 \in T_p(\Sigma)$ sa première projection et $t_2 \in T_p(\{0,1\})$ sa deuxième projection. On remarque que $\mathrm{dom}(t) = \mathrm{dom}(t_1) = \mathrm{dom}(t_2)$.

Étant donné une formule $\varphi(x) \in \mathrm{FO}(\Sigma, <, \downarrow_1, \ldots, \downarrow_p)$ ayant exactement une variable libre x, on note

$$\mathcal{L}(\varphi) = \{t \in T_p(\Sigma') \mid \forall u \in \mathrm{dom}(t),\ t_2(u) = 1 \text{ ssi } t_1, x \mapsto u \models \varphi\}\,.$$

[4] **b)** On considère la formule

$$\varphi_1(x) = \exists y \left(P_a(y) \wedge \bigvee_{n=1}^{p} x \downarrow_n y \right).$$

Donner un automate d'arbres $\mathcal{A}_1$ déterministe et complet (ascendant) qui reconnaît $\mathcal{L}(\varphi_1)$.
Il faut bien sûr prouver que $\mathcal{A}_1$ reconnaît $\mathcal{L}(\varphi_1)$.

[4] **c)** On considère la formule

$$\varphi_2(x) = \forall z \left[(x \leq z \wedge \mathrm{feuille}(z)) \rightarrow \exists y \left(x \leq y \leq z \wedge P_b(y) \wedge \forall z\, (x \leq z < y \rightarrow P_a(z)) \right) \right]$$

dans laquelle $\mathrm{feuille}(z)$ est une macro pour $\neg \exists y\, z < y$.
Donner un automate d'arbres $\mathcal{A}_2$ déterministe et complet (ascendant) qui reconnaît $\mathcal{L}(\varphi_2)$.
Il faut bien sûr prouver que $\mathcal{A}_2$ reconnaît $\mathcal{L}(\varphi_2)$.

2 Fonctions séquentielles

Le filtre à rebonds est l'application $f : \{0,1\}^* \to \{0,1\}^*$ définie par $f(u_1 \cdots u_n) = v_1 \cdots v_n$ avec

$$v_i = \begin{cases} 0 & \text{si } u_{i-1} = u_{i+1} = 0 \\ 1 & \text{si } u_{i-1} = u_{i+1} = 1 \\ u_i & \text{sinon} \end{cases}$$

en posant $u_0 = u_{n+1} = 0$. Par exemple, $f(0101101) = 0011110$.

[4] **a)** Donner (dessiner) un automate séquentiel ayant au plus 5 états qui réalise la fonction f.
Normaliser cet automate et dessiner l'automate obtenu.
Minimiser l'automate précédent et dessiner l'automate minimal du filtre à rebonds.

Remarque : Un filtre à rebonds peut être utilisé pour " lisser " une suite de 0 et de 1 par exemple pour réduire les imperfections d'une image.

3 Grammaires

On considère la grammaire G des expressions postfixes définie par

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

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

Sur le même alphabet, on considère aussi la grammaire G' définie par les règles :

$$\begin{aligned} S &\to aS_1 \\ S_1 &\to \varepsilon \mid aS_1S_2 \\ S_2 &\to +S_1 \mid \times S_1 \end{aligned}$$

[2] **a)** Montrer que la grammaire G n'est pas LL.
Calculer First_1 et Follow_1 pour toutes les variables de la grammaire G'.
En déduire que G' est LL(1).
Construire la table d'analyse fortement LL(1) de la grammaire G'.

[2] **b)** Construire une grammaire G'' en forme normale de Greibach qui soit équivalente à G (si vous n'utilisez pas la construction du cours, il faudra prouver que G'' est équivalente à G).
Montrer que les grammaires G' et G'' sont équivalentes.

[4] **c)** Montrer que la grammaire G engendre le langage

$L = \{w \in \Sigma^* \mid |w|_a = |w|_+ + |w|_\times + 1 \text{ et } |u|_a > |u|_+ + |u|_\times \text{ pour tout préfixe propre } \varepsilon < u < w\}$

[2] **d)** Le langage engendré par la grammaire G est-il rationnel ?
La grammaire G est-elle ambiguë ?

[3] **e)** Donner un automate à pile déterministe, temps-réel, simple qui reconnaît par sommet de pile le langage engendré par la grammaire G.
Il faut bien sûr prouver que l'automate à pile reconnaît L.

Langages Formels

Licence Informatique – ENS Cachan

Partiel du 21 mars 2013
durée 3 heures

Document autorisé : polycopié du cours.
Toutes les réponses devront être correctement justifiées.
Le chiffre en regard d'une question est une indication sur sa difficulté ou sa longueur.

1 Automates et logique

On fixe un alphabet Σ contenant au moins les deux lettres a et b. On s'intéresse à la formule

$$\varphi_1(x) = \exists z\,(z \leq x \wedge P_b(z) \wedge \forall y\,(z < y \leq x \to P_a(y)))\ .$$

On note $\Sigma' = \Sigma \times \{0,1\}$, $\Sigma_0 = \Sigma \times \{0\}$ et $\Sigma_1 = \Sigma \times \{1\}$. On rappelle qu'un mot $W = (a_1, b_1)\cdots(a_n, b_n) \in \Sigma_0^*\Sigma_1\Sigma_0^*$ code le mot $w = a_1 \cdots a_n \in \Sigma^*$ et la valuation σ vérifiant $\sigma(x) = i$ si et seulement si $b_i = 1$. On écrit $W = (w, \sigma)$ pour indiquer que W code le mot w et la valuation σ. Le langage défini par φ_1 est

$$L_1 = \{W = (w, \sigma) \in \Sigma_0^*\Sigma_1\Sigma_0^* \mid w, \sigma \models \varphi_1\}\ .$$

[1] **a)** Donner un automate qui reconnaît L_1.

[1] **b)** Donner une expression sans étoile pour le langage L_1.

Pour tout mot $a_1 \cdots a_n \in \Sigma^+$ et toute position $1 \leq i \leq n$, on note

$$(a_1 \cdots a_n, i) = (a_1, 0)\cdots(a_{i-1}, 0)(a_i, 1)(a_{i+1}, 0)\cdots(a_n, 0) \in \Sigma_0^*\Sigma_1\Sigma_0^*\ .$$

Pour tout langage $L \subseteq \Sigma_0^*\Sigma_1\Sigma_0^*$, on définit le langage

$$L' = \{(a_1, b_1)\cdots(a_n, b_n) \in (\Sigma')^* \mid \forall\ 1 \leq\ i \leq n,\ b_i = 1 \iff (a_1 \cdots a_n, i) \in L\}\ .$$

[4] **c)** Montrer que si L est reconnaissable alors L' est aussi reconnaissable.
Indication : Si $\mathcal{A} = (Q, \Sigma', \delta, q_0, F)$ est un automate déterministe complet reconnaissant L, construire un automate $\mathcal{A}' = (Q', \Sigma', \delta', q_0', F')$ avec $Q' = Q \times 2^Q \times 2^Q$, $q_0' = (q_0, \emptyset, \emptyset)$ et $F' = Q \times 2^F \times 2^{Q \setminus F}$.

2 Automates d'arbres

Soit $A_2 = \{d_1, d_2\}$ un alphabet de directions et Σ un alphabet. Soit $t : A_2^* \to \Sigma$ un arbre binaire. Un ensemble $X \subseteq \mathrm{dom}(t)$ est un *préfixe* de t si $X \neq \emptyset$ et X est fermé par préfixe : $uv \in X$ implique $u \in X$ pour tous $u, v \in A_2^*$.

Pour un arbre t, un préfixe X de t et un nœud $u \in X$, on définit la frontière $\mathrm{Fr}(u, X, t)$ inductivement par :

— si u est une feuille de X, i.e., $uA_2 \cap X = \emptyset$, alors $\mathrm{Fr}(u, X, t) = t(u) \in \Sigma$,
— si u est d'arité 1 dans X, i.e., $uA_2 \cap X = \{ud\}$, alors $\mathrm{Fr}(u, X, t) = \mathrm{Fr}(ud, X, t)$,
— si u est d'arité 2 dans X, i.e., $uA_2 \cap X = \{ud_1, ud_2\}$, alors $\mathrm{Fr}(u, X, t) = \mathrm{Fr}(ud_1, X, t)\mathrm{Fr}(ud_2, X, t)$.

Soit $L \subseteq \Sigma^*$ un langage de mots. On note $L' \subseteq T_2(\Sigma)$ l'ensemble des arbres $t\colon A_2^* \to \Sigma$ tels qu'il existe un préfixe X de t avec $\mathrm{Fr}(\varepsilon, X, t) \in L$.

[4] **a)** Montrer que si $L \subseteq \Sigma^*$ est un langage reconnaissable de mots, alors $L' \subseteq T_2(\Sigma)$ est un langage d'arbres reconnaissable.

3 Grammaires

Pour cet exercice, on rappelle que le langage de Dyck

$$D_1^* = \{v \in \{a, b\}^* \mid |v|_a = |v|_b \text{ et } |u|_a \geq |u|_b \text{ pour tous préfixes } u \text{ de } v\}$$

est engendré par la grammaire non ambiguë $S \to aSbS + \varepsilon$.

[2] **a)** Montrer que le langage $L_1 = \{v \in \{a, b\}^* \mid |u|_a = |u|_b\}$ n'est pas rationnel.
Donner une grammaire algébrique G_1 qui engendre le langage L_1.
Il faut bien sûr prouver que G_1 engendre exactement le langage L_1.

[2] **b)** On considère le langage $L_1' = \{v \in L_1 \mid \forall u \in L_1,\ u \leq v \Longrightarrow (u = \varepsilon \vee u = v)\}$.
Donner une grammaire algébrique *non ambiguë* G_1' qui engendre le langage L_1'.
Il faut bien sûr prouver que G_1' est non ambiguë et engendre L_1'.

[1] **c)** Montrer que le langage $L_2 = \{a^i b^j c^k \mid k = \max(i, j)\}$ n'est pas algébrique.

[2] **d)** Soit $L_3 = \{v \in \{a, b, c\}^* \mid |v|_a = |v|_b = |v|_c > 0\}$.
Donner une grammaire contextuelle G_3 qui engendre L_3.
Il faut bien sûr prouver que G_3 engendre exactement le langage L_3.

On note $\mathrm{pref}(L)$ l'ensemble des préfixes des mots d'un langage L.

[2] **e)** Montrer que si L est un langage algébrique alors $\mathrm{pref}(L)$ est aussi algébrique.

Soit $L \subseteq \Sigma^*$ un langage sur l'alphabet Σ. On note $\frac{1}{2}(L)$ l'ensemble des mots $u \in \Sigma^*$ tels que $u\Sigma^{|u|} \cap L \neq \emptyset$, i.e., l'ensemble des premières moitiés des mots de L.

[1] **f)** Montrer que $\frac{1}{2}(D_1^*) = \mathrm{pref}(D_1^*)$.

[4] **g)** Montrer que la moitié d'un langage rationnel est un langage rationnel : si $L \subseteq \Sigma^*$ est un langage rationnel alors $\frac{1}{2}(L)$ est aussi rationnel.

[**] **h)** Question subsidiaire : Montrer que la moitié d'un langage algébrique n'est pas nécessairement algébrique.

Langages Formels

Licence Informatique – ENS Cachan

Examen du 24 mai 2012
durée 3 heures

Document autorisé : polycopié du cours.
Toutes les réponses devront être correctement justifiées.
Le chiffre en regard d'une question est une indication sur sa difficulté ou sa longueur.

1 Automates d'arbres

[1] **a)** Un arbre binaire est équilibré si pour chaque nœud, la différence de hauteur entre les sous-arbres gauche et droit est au plus 1. L'ensemble des arbres binaires équilibrés avec étiquettes dans l'alphabet $\Sigma = \{a, b, c\}$ est-il reconnaissable ?

[3] **b)** On considère l'ensemble $T_p(\Sigma)$ des Σ-arbres d'arité au plus p avec l'alphabet de directions $A_p = \{d_1, \ldots, d_p\}$. Pour $a \in \Sigma$ et $L \subseteq T_p(\Sigma)$ on définit

$$L_a = \{u \in A_p^* \mid \exists t \in L \text{ tel que } u \in \mathrm{dom}(t) \text{ et } \ t(u) = a\}\,.$$

Soit $\mathcal{A} = (Q, \Sigma, \delta, F)$ un automate d'arbres reconnaissant $L \subseteq T_p(\Sigma)$. Le langage $L_a \subseteq A_p^*$ est-il reconnaissable ?

[4] **c)** On considère l'automate d'arbre $\mathcal{A} = (Q, \Sigma, \delta, F)$ sur l'alphabet $\Sigma = \{a, b, f, g\}$ et l'ensemble d'états $Q = \{q_0, q_1, \ldots, q_5\}$, avec $F = \{q_4, q_5\}$ et ayant pour transitions

$$\xrightarrow{a} q_1 \qquad q_1 \xrightarrow{g} q_3 \qquad q_1, q_3 \xrightarrow{f} q_4$$
$$\xrightarrow{b} q_2 \qquad q_2 \xrightarrow{g} q_4 \qquad q_2, q_4 \xrightarrow{f} q_5$$
$$q_2, q_5 \xrightarrow{f} q_4$$

et toutes les transitions non spécifiées ci-dessus mènent vers l'état q_0.

Montrer que $\mathcal{A}$ accepte les arbres définis par l'expression $t_1^* t_2$ où $t_1 = f(b, \Box)$ et $t_2 = g(b)$.
Caractériser le langage d'arbres $\mathcal{L}(\mathcal{A})$ au moyen d'expressions similaires.
Minimiser l'automate $\mathcal{A}$.

2 Fonctions séquentielles

On considère les mots sur l'alphabet binaire $\Sigma = \{0, 1\}$ qui codent des entiers en binaire en commençant par le bit de poids faible : le mot $u = a_0 a_1 \cdots a_n$ code l'entier $\overline{u}^2 = \sum_{i=0}^{n} a_i 2^i$.

[1] **a)** On considère l'ensemble $L_1 \subseteq \Sigma^*$ des mots qui codent des entiers congrus à 1 modulo 3. L'ensemble L_1 est-il reconnaissable ? Si oui, construire son automate minimal.

[1] **b)** On considère la fonction $f\colon \Sigma^* \to \Sigma^*$ qui associe à un mot $u \in \Sigma^*$ le mot $v \in \Sigma^*$ tel que

$$\begin{cases} |v| = |u| \text{ et } \overline{v}^2 = \overline{u}^2 \,\texttt{div}\, 2 & \text{si } \overline{u}^2 \,\texttt{mod}\, 2 = 0 \\ |v| = |u| + 1 \text{ et } \overline{v}^2 = 2\overline{u}^2 & \text{si } \overline{u}^2 \,\texttt{mod}\, 2 = 1 \end{cases}$$

La fonction f est-elle séquentielle ? Si oui, construire son automate séquentiel minimal.

[3] **c)** On considère la fonction $g\colon \Sigma^* \to \Sigma^*$ qui associe à un mot $u \in \Sigma^*$ le mot $v \in \Sigma^*$ tel que $|v| = |u|$ et $\overline{v}^2 = \overline{u}^2 \,\texttt{div}\, 3$.
La fonction g est-elle séquentielle ? Si oui, construire son automate séquentiel minimal.

[3] **d)** On considère la fonction $h\colon \Sigma^* \to \Sigma^*$ qui associe à un mot $u \in \Sigma^*$ tel que $\overline{u}^2 \,\texttt{mod}\, 3 = 0$ le mot $v \in \Sigma^*$ tel que $|v| = |u|$ et $\overline{v}^2 = \overline{u}^2 \,\texttt{div}\, 3$. La fonction h n'est pas définie sur les mots $u \in \Sigma^*$ tels que $\overline{u}^2 \,\texttt{mod}\, 3 \neq 0$.
La fonction h est-elle séquentielle ? Si oui, construire son automate séquentiel minimal.

3 Grammaires, automates à pile et analyse syntaxique

[2] **a)** Construire un automate à pile pour le langage

$$L = \{w \in a^*b^*c^* \mid w \text{ n'est pas de la forme } a^n b^n c^n \text{ avec } n \geq 0\}.$$

[4] **b)** Étant donnés deux langages $K, L \subseteq \Sigma^*$ on définit

$$\mathrm{C}(K, L) = \{u \in \Sigma^* \mid \forall v \in L,\ uv \in K\}.$$

En supposant K rationnel et L algébrique, le langage $\mathrm{C}(K, L)$ est-il rationnel ? algébrique ?
En supposant K algébrique et L rationnel, le langage $\mathrm{C}(K, L)$ est-il rationnel ? algébrique ?

[4] **c)** On considère la grammaire augmentée G définie par les règles :

$$0 : S' \to S \qquad 1 : S \to S\_S \qquad 2 : S \to S\,\hat{}\,S \qquad 3 : S \to \{S\} \qquad 4 : S \to a$$

Calculer $\mathrm{Follow}_1(S')$ et $\mathrm{Follow}_1(S)$.
Calculer l'automate $\mathcal{C}_0$ des contextes SLR de G.
Donner la table (`action` et `goto`) de l'analyseur SLR de G.
La grammaire G est-elle SLR ? ambiguë ?

Langages Formels

Licence Informatique – ENS Cachan

Partiel du 29 mars 2012
durée 3 heures

Document autorisé : polycopié du cours.
Toutes les réponses devront être correctement justifiées.
Le chiffre en regard d'une question est une indication sur sa difficulté ou sa longueur.

1 Langages commutatifs

Soit $\Sigma = \{a_1, \ldots, a_k\}$ un alphabet de taille $k \geq 1$. *L'image de Parikh* d'un mot $v \in \Sigma^*$ est le vecteur d'entiers $\psi(v) = (|v|_{a_1}, \ldots, |v|_{a_k}) \in \mathbb{N}^k$. La définition s'étend aux langages $L \subseteq \Sigma^*$ par $\psi(L) = \{\psi(v) \mid v \in L\}$. Par exemple, si $k = 3$ alors $\psi(a_1a_2a_1) = (2, 1, 0)$ et $\psi((a_1a_3)^*) = \{(n, 0, n) \mid n \geq 0\}$.

L'application de Parikh $\psi : \Sigma^* \to \mathbb{N}^k$ est un morphisme du monoïde libre $(\Sigma^*, \cdot, \varepsilon)$ dans le monoïde commutatif $(\mathbb{N}^k, +, \bar{0})$, où $\bar{0} = (0, \ldots, 0)$ et l'addition est composante par composante. Pour $a \in \mathbb{N}$ et $u = (i_1, \ldots, i_k) \in \mathbb{N}^k$, on utilisera aussi le produit externe $au = (ai_1, \ldots, ai_k)$. Pour $u_1, \ldots, u_m \in \mathbb{N}^k$, on note $\langle u_1, \ldots, u_m \rangle$ le sous-monoïde de $\mathbb{N}^k$ engendré par $u_1, \ldots, u_m$:

$$\langle u_1, \ldots, u_m \rangle = \{a_1u_1 + \cdots + a_mu_m \mid a_1, \ldots, a_m \in \mathbb{N}\}.$$

Par exemple, $\psi((a_1a_3)^*) = \langle (1, 0, 1) \rangle$. Un sous-ensemble de $\mathbb{N}^k$ est *linéaire* s'il est de la forme

$$u_0 + \langle u_1, \ldots, u_m \rangle = \{u_0 + a_1u_1 + \cdots + a_mu_m \mid a_1, \ldots, a_m \in \mathbb{N}\}$$

avec $u_0, u_1, \ldots, u_m \in \mathbb{N}^k$. Finalement, un sous-ensemble de $\mathbb{N}^k$ est *semi-linéaire* si c'est une union finie d'ensembles linéaires.

[1] **a)** Montrer que tout ensemble semi-linéaire est l'image de Parikh d'un langage rationnel.

[1] **b)** Montrer que la somme de deux ensembles linéaires est encore linéaire.
Montrer que la somme de deux ensembles semi-linéaires est encore semi-linéaire.

Le sous-monoïde engendré par un sous-ensemble $K \subseteq \mathbb{N}^k$ est

$$\langle K \rangle = \{u_1 + \cdots + u_m \mid m \geq 0 \text{ et } u_1, \ldots, u_m \in K\}.$$

[5] **c)** Montrer que le sous-monoïde engendré par un ensemble *linéaire* est semi-linéaire mais pas forcément linéaire.
Montrer que $\langle K \cup K' \rangle = \langle K \rangle + \langle K' \rangle$.
Montrer que le sous-monoïde engendré par un ensemble *semi-linéaire* est semi-linéaire.
En déduire que l'image de Parikh d'un langage rationnel est semi-linéaire.

[1] **d)** Montrer qu'il existe des langages non algébriques dont l'image de Parikh est semi-linéaire.

Nous allons dans la suite prouver le théorème de Parikh : l'image de Parikh d'un langage algébrique est semi-linéaire.

Soit $G = (\Sigma, V, P, S)$ une grammaire algébrique. Pour un arbre de dérivation s de G, on note $\mathrm{rac}(s)$ le symbole à sa racine et $\mathrm{Fr}(s) \in (\Sigma \cup V)^*$ sa *frontière*. On note $h(s)$ la hauteur de l'arbre s (la hauteur d'un arbre trivial réduit à sa racine est 0). Finalement, on note $\mathrm{var}(s)$ l'ensemble des variables de V qui apparaissent dans l'arbre s.

Une *pompe* est un arbre de dérivation s de G qui est non trivial (n'est pas réduit à la racine) et dont la frontière est dans $\Sigma^* \cdot \mathrm{rac}(s) \cdot \Sigma^*$. Une pompe s de racine x correspond donc à une dérivation (non triviale) $x \xrightarrow{+} uxv$ de G avec $u, v \in \Sigma^*$.

On note $r \lhd t$ si r et t sont deux arbres de dérivation de G et que t peut s'obtenir à partir de r par insertion d'une pompe s à un nœud de r étiqueté par $\mathrm{rac}(s)$. En termes de dérivations, si la pompe s correspond à $x \xrightarrow{+} uxv$ alors il existe deux dérivations $\mathrm{rac}(r) \xrightarrow{*} \alpha x \gamma$ et $x \xrightarrow{*} \beta$ telles que r correspond à la dérivation $\mathrm{rac}(r) \xrightarrow{*} \alpha x \gamma \xrightarrow{*} \alpha\beta\gamma = \mathrm{Fr}(r)$ et t correspond à la dérivation $\mathrm{rac}(t) = \mathrm{rac}(r) \xrightarrow{*} \alpha x \gamma \xrightarrow{+} \alpha u x v \gamma \xrightarrow{*} \alpha u \beta v \gamma = \mathrm{Fr}(t)$.

Si $r \lhd t$ par insertion de la pompe s, on dit que s est contenue dans t. De plus le nombre de nœuds de r est strictement inférieur à celui de t puisque la pompe s est un arbre non trivial.

Une pompe *minimale* est une pompe s qui est minimale parmi les pompes pour la relation $\lhd$: si $r \lhd s$ alors r n'est pas une pompe.

[1] **e)** Montrer que si $r \lhd s$ et s est une pompe alors $\mathrm{Fr}(r) \in \Sigma^* \cdot \mathrm{rac}(r) \cdot \Sigma^*$.
Montrer que si $r \lhd s$ et s est une pompe minimale alors r est l'arbre trivial réduit à $\mathrm{rac}(s)$.

[1] **f)** Montrer que toute pompe contient une pompe minimale, i.e., si s est une pompe alors il existe un arbre de dérivation r et une pompe minimale s' tels que s s'obtient à partir de r par insertion de s'.

[4] **g)** Montrer que si s est une pompe minimale, alors $h(s) \leq 2|V|$.
Montrer qu'une grammaire contient un nombre fini de pompes minimales.
Donner une grammaire G et une pompe minimale s de G telle que $h(s) = 2|V|$.

On définit maintenant une relation d'ordre partiel sur les arbres de dérivation : $r \leq t$ si t peut s'obtenir à partir de r par une suite d'insertions de pompes minimales s telles que $\mathrm{var}(s) \subseteq \mathrm{var}(r)$. Donc si $r \leq t$ alors $\mathrm{var}(r) = \mathrm{var}(t)$ (ce qui n'est pas forcément le cas si $r \lhd t$).

[2] **h)** Soit r un arbre de dérivation de G tel que $\mathrm{Fr}(r) \in \Sigma^*$. Montrer que l'image de Parikh $\psi(\{\mathrm{Fr}(t) \mid r \leq t\})$ est un ensemble linéaire.

[5] **i)** Soit t un arbre de dérivation de G tel que $\mathrm{Fr}(t) \in \Sigma^*$. Montrer que si t contient une branche ayant au moins $|V| + 1$ occurrences d'une même variable $x \in V$ alors t n'est pas $<$-minimal, i.e., il existe $r \neq t$ tel que $r \leq t$.
Montrer qu'une grammaire G admet un nombre fini d'arbres de dérivation qui sont $<$-minimaux et dont la frontière est dans Σ^*.

[1] **j)** Montrer que l'image de Parikh d'un langage algébrique est semi-linéaire.

2 Automate déterministe descendant d'arbres

Soit $A_p = \{d_1, \ldots, d_p\}$ un alphabet de directions et Σ un alphabet disjoint de A_p. Soit $t : A_p^* \to \Sigma$ un arbre. Si $v \in \mathrm{dom}(t)$ est un nœud de t, on note $\mathrm{ar}_t(v)$ l'arité du nœud v dans l'arbre t. Soit $\Gamma = \Sigma \times \{0, \ldots, p\} \times (A_p \cup \{\bot\})$. Le chemin d'un arbre $t : A_p^* \to \Sigma$ associé à une feuille $u = u_1 u_2 \cdots u_{n-1} u_n \in \mathrm{feuille}(t)$ est le mot de Γ^+

$$c(t,u) = (t(\varepsilon), \mathrm{ar}_t(\varepsilon), u_1)(t(u_1), \mathrm{ar}_t(u_1), u_2) \cdots (t(u_1 \ldots u_{n-1}), \mathrm{ar}_t(u_1 \ldots u_{n-1}), u_n)(t(u), 0, \bot)\,.$$

On note $\mathcal{C}(t) = \{c(t,u) \mid u \in \mathrm{feuille}(t)\} \subseteq \Gamma^+$ l'ensemble des chemins de t. Si $L \subseteq T_p(\Sigma)$ est un langage d'arbres, on note $\mathcal{C}(L) = \bigcup_{t \in L} C(t) \subseteq \Gamma^+$ l'ensemble des chemins des arbres de L.

[2] **a)** Montrer que si $L \subseteq T_p(\Sigma)$ est un langage d'arbres reconnaissable alors $\mathcal{C}(L) \subseteq \Gamma^+$ est un langage reconnaissable de mots.

Si $L \subseteq T_p(\Sigma)$ est un langage d'arbres, on définit la *clôture par chemins* de L comme l'ensemble $\mathcal{CC}(L) \subseteq T_p(\Sigma)$ des arbres t tels que $\mathcal{C}(t) \subseteq \mathcal{C}(L)$.

[2] **b)** Montrer que si $L \subseteq T_p(\Sigma)$ est un langage d'arbres reconnaissable alors $\mathcal{CC}(L)$ aussi.

[3] **c)** Montrer qu'un langage d'arbres reconnaissable $L \subseteq T_p(\Sigma)$ vérifie $L = \mathcal{CC}(L)$ si et seulement si il peut être reconnu par un automate déterministe descendant.

[1] **d)** Peut-on décider si un langage d'arbres reconnaissable $L \subseteq T_p(\Sigma)$ est reconnaissable par un automate déterministe descendant ?

Langages Formels

Licence Informatique – ENS Cachan

Examen du 26 mai 2011
durée 3 heures

Document autorisé : polycopié du cours.
Toutes les réponses devront être correctement justifiées.
Le chiffre en regard d'une question est une indication sur sa difficulté ou sa longueur.

1 Langages linéaires et automates à un pic

Un *automate à un pic* est un automate à pile 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.

[3] **a)** Montrer que le langage $L = \{a^n b^n \mid n > 0\} \cup \{a^n b^{2n} \mid n > 0\}$ est un langage linéaire.
Montrer que L est un langage à un pic.
Montrer que le langage $K = \{ba^{i_1}ba^{i_2}b \cdots ba^{i_n}b \mid n > 0 \text{ et } \exists j, i_j \neq j\}$ est un langage à un pic.

[2] **b)** Montrer que tout langage linéaire est un langage à un pic.

[3] **c)** Soit $G = (\Sigma, V_1 \cup V_2, P_1 \cup P_2)$ une grammaire vérifiant :

$$\begin{aligned} &V_1 \cap V_2 = \emptyset \\ &P_1 \subseteq V_1 \times (V_1\Sigma^* \cup \Sigma^*) \\ &P_2 \subseteq V_2 \times (\Sigma^* V_2 V_1^* \cup \Sigma^*) \end{aligned}$$

Montrer que pour tout $x \in V$, $\mathcal{L}_G(x)$ est un langage linéaire.

Soit $\mathcal{A} = (Q, \Sigma, Z, T, (q_0, z_0))$ un automate à pile arbitraire.
Pour $p, q \in Q$ et $z \in Z$ on définit les langages

$$\begin{aligned} K_{zp,q} &= \{w \in \Sigma^* \mid \exists\ zp \xrightarrow{w} q \text{ calcul dans } \mathcal{A} \text{ avec une pile de taille toujours } \leq 1\}, \\ L_{zp,q} &= \{w \in \Sigma^* \mid \exists\ zp \xrightarrow{w} q \text{ calcul à un pic dans } \mathcal{A}\}. \end{aligned}$$

[2] **d)** Montrer que le langage $K_{zp,q}$ peut être engendré par une grammaire linéaire *droite*.

[2] **e)** Construire une grammaire vérifiant les conditions de la question **(c)** et qui engendre les langages $L_{zp,q}$ avec les variables de V_2 et les langages $K_{zp,q}$ avec les variables de V_1.
Montrer que tout langage à un pic est un langage linéaire.

2 Fonctions séquentielles et logique temporelle

On s'intéresse ici à une logique utilisant des modalités temporelles utiles pour spécifier les propriétés de certains systèmes. On fixe un ensemble fini AP de propositions atomiques. Un état du système est abstrait en l'ensemble $s \subseteq \mathrm{AP}$ des propositions atomiques qui sont vraies dans cet état. Une exécution du système sera donc décrite par un mot $\sigma = s_1 s_2 \cdots \in \Sigma^*$ où $\Sigma = 2^{\mathrm{AP}}$. On note $|\sigma|$ la longueur du mot σ.

Les formules de la logique temporelle $\mathrm{LTL}(\mathrm{AP}, \mathsf{Y}, \mathsf{S})$ sont obtenues à partir des propositions atomiques en utilisant les connecteurs booléens, la modalité unaire Y (Yesterday) et la modalité binaire S (Since) :

$$\alpha ::= p \in \mathrm{AP} \mid \neg\alpha \mid \alpha \vee \alpha \mid \mathsf{Y}\,\alpha \mid \alpha\,\mathsf{S}\,\alpha$$

Soit $\alpha \in \mathrm{LTL}(\mathrm{AP}, \mathsf{Y}, \mathsf{S})$ une formule, $\sigma = s_1 s_2 \cdots \in \Sigma^*$ un mot et i une position vérifiant $1 \leq i \leq |\sigma|$. On note $\sigma, i \models \alpha$ lorsque le mot σ à la position i satisfait la formule α. Cette sémantique se définit inductivement comme suit :

$$\begin{array}{lll}
\sigma, i \models p & \text{si} & p \in s_i \\
\sigma, i \models \neg\alpha & \text{si} & \sigma, i \not\models \alpha \\
\sigma, i \models \alpha \vee \beta & \text{si} & \sigma, i \models \alpha \text{ ou } \sigma, i \models \beta \\
\sigma, i \models \mathsf{Y}\,\alpha & \text{si} & i > 1 \text{ et } \sigma, i-1 \models \alpha \\
\sigma, i \models \alpha\,\mathsf{S}\,\beta & \text{si} & \exists j \in \{1, \ldots, i\},\ \sigma, j \models \beta \text{ et } \forall k \in \{j+1, \ldots, i\},\ \sigma, k \models \alpha
\end{array}$$

Par exemple, soit $\mathrm{AP} = \{p, q\}$ et soit $\sigma = \{p\}\{p,q\}\{q\}\{q\}\{p,q\}\{\}\{\}\{p\}\{q\} \in \Sigma^*$. On a $\sigma, 1 \models p \wedge \neg q$, $\sigma, 3 \models \mathsf{Y}\,p$, $\sigma, 4 \models q\,\mathsf{S}\,p$ et $\sigma, 8 \models \mathsf{Y}(\neg p\,\mathsf{S}\,q)$.

À chaque formule $\alpha \in \mathrm{LTL}(\mathrm{AP}, \mathsf{Y}, \mathsf{S})$ on associe une fonction $[\![\alpha]\!] : \Sigma^* \to \{0,1\}^*$ définie pour $\sigma \in \Sigma^*$ par $[\![\alpha]\!](\sigma) = c_1 \cdots c_{|\sigma|}$ avec pour tout $1 \leq i \leq |\sigma|$, $c_i = 1$ si et seulement si $\sigma, i \models \alpha$.

En reprenant le mot σ de l'exemple ci-dessus on a $[\![q]\!](\sigma) = 011110001$, $[\![\mathsf{Y}\,p]\!](\sigma) = 011001001$ et $[\![q\,\mathsf{S}\,p]\!](\sigma) = 111110011$.

[3] **a)** Soit $\mathrm{AP} = \{p, q\}$. Montrer que les fonctions suivantes sont séquentielles pures. Donner à chaque fois l'automate séquentiel pur (si possible minimal) qui calcule la fonction.

1. $[\![p]\!]$
2. $[\![p \to \mathsf{Y}\,q]\!]$
3. $[\![p\,\mathsf{S}\,q]\!]$
4. $p \to \mathsf{Y}((\neg p)\,\mathsf{S}\,q)$

À chaque opérateur op d'arité k on associe une fonction $f_{\mathrm{op}} : (\{0,1\}^k)^* \to \{0,1\}^*$ qui décrit sa sémantique. Par exemple,

— $f_\neg(a_1 \cdots a_n) = c_1 \cdots c_n$ avec pour tout $1 \leq i \leq n$, $c_i = 1 - a_1$,
— $f_{\mathsf{Y}}(a_1 \cdots a_n) = c_1 \cdots c_n$ avec $c_1 = 0$ et pour tout $1 < i \leq n$, $c_i = a_{i-1}$
— $f_{\mathsf{S}}((a_1, b_1) \cdots (a_n, b_n)) = c_1 \cdots c_n$ avec pour tout $1 \leq i \leq n$,
$c_i = 1$ si et seulement si $\exists j \in \{1, \ldots, i\}$ tel que $b_j = 1$ et $\forall k \in \{j+1, \ldots, i\}$, $a_k = 1$.

[2] **b)** Montrer que les fonctions $f_\neg$, $f_\vee$, f_{Y} et f_{S} sont séquentielles pures. En déduire que pour chaque formule $\alpha \in \mathrm{LTL}(\mathrm{AP}, \mathsf{Y}, \mathsf{S})$, la fonction $[\![\alpha]\!]$ est séquentielle pure.

Le problème SAT pour les formules LTL est le suivant :

Donnée Une formule $\alpha \in \mathrm{LTL}(\mathrm{AP}, \mathsf{Y}, \mathsf{S})$,

Question Existe-t-il un mot $\sigma \in \Sigma^+$ et une position $1 \leq\ i \leq |\sigma|$ tels que $\sigma, i \models \alpha$?

Le problème Rec-MC pour les formules LTL est le suivant :

Donnée Une formule $\alpha \in \mathrm{LTL}(\mathrm{AP}, \mathsf{Y}, \mathsf{S})$ et un langage reconnaissable $M \subseteq \Sigma^+$,

Question A-t-on $\sigma, |\sigma| \models \alpha$ pour tout $\sigma \in M$?

[2] **c)** Montrer que les problèmes SAT et Rec-MC sont décidables.
Facultatif : Montrer que ces problèmes sont dans PSPACE.

On introduit maintenant deux modalités "futur" : X (neXt) et U (Until) dont la sémantique est définie (en utilisant les notations précédentes) par :

$$\begin{array}{lll} \sigma, i \models \mathsf{X}\,\alpha & \text{si} & i < |\sigma| \text{ et } \sigma, i+1 \models \alpha \\ \sigma, i \models \alpha\,\mathsf{U}\,\beta & \text{si} & \exists\, j \in \{i, \ldots, |\sigma|\},\ \sigma, j \models \beta \text{ et } \forall k \in \{i, \ldots, j-1\},\ \sigma, k \models \alpha \end{array}$$

[2] **d)** Montrer que la fonction f_{X} est séquentielle. Est-elle séquentielle pure ? Justifier.
La fonction f_{U} est-elle séquentielle ? Justifier.

La restriction aux entrées d'un automate séquentiel doit être déterministe. Pour un automate fonctionnel non ambigu $\mathcal{A} = (Q, A, I, T, F, B, \varphi)$, on assouplit cette condition en demandant seulement que la restriction aux entrées soit non ambigu : Q est l'ensemble fini d'états, $I, F \subseteq Q$ définissent les états initiaux et finaux, $T \subseteq Q \times A \times Q$ est la relation de transitions et $\varphi : T \to B^*$ la fonction de sortie. Chaque mot $u \in A^*$ doit admettre au plus un calcul acceptant dans l'automate réduit aux entrées (Q, A, I, T, F) et la fonction calculée $[\![\mathcal{A}]\!](u)$ est la concaténation des sorties de ce calcul.

[2] **e)** Montrer que la fonction f_{U} peut être calculée par un automate fonctionnel non ambigu.

Langages Formels

Licence Informatique – ENS Cachan

Partiel du 31 mars 2011

durée 3 heures

Document autorisé : polycopié du cours.
Toutes les réponses devront être correctement justifiées.
Le chiffre en regard d'une question est une indication sur sa difficulté ou sa longueur.

1 Arithmétique de Presburger

On s'intéresse ici à la théorie logique du premier ordre des entiers munis de l'addition (mais pas de la multiplication). Plus précisément, on fixe un ensemble infini $\mathcal{X}$ de variables. On définit l'arithmétique de PRESBURGER, comme le plus petit ensemble $\mathcal{P}$ de formules logiques tel que

— si $x, y, z \in \mathcal{X}$ alors $x = 0$ et $x + y = z$ sont des formules de $\mathcal{P}$,
— si $x \in \mathcal{X}$ et $\varphi, \psi \in \mathcal{P}$, alors $\varphi \wedge \psi$, $\varphi \vee \psi$, $\neg\varphi$, $\forall x\ \varphi$ et $\exists x\ \varphi$ sont des formules de $\mathcal{P}$.

Dans la suite, $\mathcal{V} \subseteq \mathcal{X}$ dénotera un ensemble fini de variables. Une $\mathcal{V}$-assignation est un tuple $\sigma \in \mathbb{N}^{\mathcal{V}}$. On note $\mathrm{Free}(\varphi)$ l'ensemble des variables libres d'une formule φ. Si $\mathrm{Free}(\varphi) \subseteq \mathcal{V}$ et $\sigma \in \mathbb{N}^{\mathcal{V}}$, on note $\sigma \models \varphi$ si la formule φ est vraie pour l'assignation σ.

[2] **a)** Donner des formules de Presburger équivalentes à $x = y$, à $x < y$, à "x est pair", et à $x = 1$.

On choisit de coder les entiers en binaire, avec bit de poids fort à gauche. On définit une fonction de décodage $\nu : \{0,1\}^* \longrightarrow \mathbb{N}$ par

$$\nu(\varepsilon) = 0 \qquad \nu(w0) = 2\nu(w) \qquad \nu(w1) = 1 + 2\nu(w)$$

Remarquons que cette fonction est surjective, totale, mais pas injective.

Une assignation $\sigma \in \mathbb{N}^{\mathcal{V}}$ sera codée par un mot sur l'alphabet $\Sigma_{\mathcal{V}} = \{0,1\}^{\mathcal{V}}$. Soit $w \in (\Sigma_{\mathcal{V}})^*$ un mot et $x \in \mathcal{V}$ une variable, on note w_x la projection du mot w sur la composante x de l'alphabet $\Sigma_{\mathcal{V}}$. On note $\overline{\nu}(w) = (\nu(w_x))_{x \in \mathcal{V}} \in \mathbb{N}^{\mathcal{V}}$ l'assignation codée par le mot w. Par exemple, $\overline{\nu}\left(\begin{smallmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{smallmatrix}\right) = \left(\begin{smallmatrix} 18 \\ 7 \\ 6 \end{smallmatrix}\right)$.

Remarquons que si $\mathcal{V} = \emptyset$, alors $\Sigma_{\mathcal{V}} = \emptyset$ et donc $(\Sigma_{\mathcal{V}})^* = \{\varepsilon\}$ est réduit au mot vide.

Finalement, pour $\varphi \in \mathcal{P}$ et $\mathcal{V}$ contenant $\mathrm{Free}(\varphi)$, on note $[\![\varphi]\!]_{\mathcal{V}} = \{w \in (\Sigma_{\mathcal{V}})^* \mid \overline{\nu}(w) \models \varphi\}$.

[2] **b)** Donner des expressions rationnelles pour les langages $[\![x = 0]\!]_{\{x\}}$ et $[\![x + y = z]\!]_{\{x,y,z\}}$.

[4] **c)** Montrer que pour toute formule $\varphi \in \mathcal{P}$ et pour tout $\mathcal{V}$ contenant $\mathrm{Free}(\varphi)$, on peut effectivement construire un automate fini reconnaissant le langage $[\![\varphi]\!]_{\mathcal{V}}$.
Indication : on pourra utiliser des projections $(\Sigma_{\mathcal{W}})^* \to (\Sigma_{\mathcal{U}})^*$ pour $\mathcal{U} \subseteq \mathcal{W}$.

[2] **d)** En déduire que l'arithmétique de Presburger est décidable, i.e., qu'on peut décider si une formule close est valide. Quelle est la complexité de cette procédure de décision ?

2 Expressions préfixes

Dans les expressions arithmétiques en notation préfixe (ou polonaise inverse), l'opérateur s'écrit avant les opérandes et on n'utilise pas de parenthèses. Par exemple, l'expression infixe parenthésée $((a+b)\times(b+c))$ s'écrit $\times + a\ b + b\ c$ en notation préfixe.

[3] Donner une grammaire algébrique G qui engendre les expressions arithmétiques en notation préfixe sur l'alphabet $\Sigma = \{+, \times, a, b, c\}$, i.e., avec les opérateurs binaires $+$ et $\times$ et les constantes a, b et c.
Donner l'arbre de dérivation de $\times + a\ b + b\ c$.
Construire un automate d'arbres déterministe ascendant qui reconnaît les arbres de dérivation de la grammaire G.

3 Rationnels, Linéaires et Algébriques

Soit Σ un alphabet, $\# \in \Sigma$ une lettre de Σ et $A = \Sigma \setminus \{\#\}$. Pour $K \subseteq A^*$, on définit le langage $L_K = \{u\#v \in K\#A^* \mid |u| = |v|\}$.

[4] **a)** Montrer que K est rationnel si et seulement si L_K est linéaire.
Indication : considérer (ou construire) une grammaire linéaire droite pour le langage K.

[3] **b)** Montrer que si L_K est algébrique alors L_K est linéaire.

4 Mélanges de mots

Soit Σ un alphabet et $u, v \in \Sigma^*$ deux mots. Le mélange de u et v est le langage

$$u \sqcup\!\sqcup v = \{u_1 v_1 \cdots u_n v_n \mid n \geq 0,\ u_i, v_i \in \Sigma^*,\ u = u_1 \cdots u_n \text{ et } v = v_1 \cdots v_n\}.$$

Cette opération est étendue aux langages en posant $K \sqcup\!\sqcup L = \bigcup_{u \in K, v \in L} u \sqcup\!\sqcup v$ pour $K, L \subseteq \Sigma^*$.

[2] **a)** Soit $\Sigma = \{a, b, c\}$ un alphabet. Construire l'automate minimal du langage $\Sigma^* ab \Sigma^* \sqcup\!\sqcup \Sigma^* b \Sigma^*$.

Dans la suite, l'alphabet Σ est de nouveau arbitraire. On introduit une copie $\overline{\Sigma}$ de l'alphabet Σ et on note $\overline{a} \in \overline{\Sigma}$ la copie de la lettre $a \in \Sigma$. On considère les morphismes Π, Π_1 et Π_2 de $(\Sigma \cup \overline{\Sigma})^*$ dans Σ^* définis pour $a \in \Sigma$ par $\Pi(a) = \Pi(\overline{a}) = a$, $\Pi_1(a) = \Pi_2(\overline{a}) = a$, et $\Pi_1(\overline{a}) = \Pi_2(a) = \varepsilon$.

[3] **b)** Montrer que $K \sqcup\!\sqcup L = \Pi(\Pi_1^{-1}(K) \cap \Pi_2^{-1}(L))$.
En déduire que si K est rationnel (resp. linéaire ou algébrique) et que L est rationnel alors $K \sqcup\!\sqcup L$ est aussi rationnel (resp. linéaire ou algébrique).

[2] **c)** Le langage $\{a^n b^n \mid n \geq 0\} \sqcup\!\sqcup \{c^p d^p \mid p \geq 0\}$ est-il algébrique ? Justifier la réponse.

Langages Formels

Licence Informatique – ENS Cachan

Examen du 27 mai 2010

durée 3 heures

Document autorisé : polycopié du cours.
Toutes les réponses devront être correctement justifiées.
Le chiffre en regard d'une question est une indication sur sa difficulté ou sa longueur.

1 Indices et exposants

On s'intéresse dans cette partie aux expressions LaTeX utilisant des indices et des exposants comme par exemple :

`x_i`	x_i	`x_{i+1}`	x_{i+1}	`x^n`	x^n	`x^{n+1}`	x^{n+1}
`x_i^2`	x_i^2	`x^2_i`	x_i^2	`x^{n^2}`	x^{n^2}	`x^{n_i}`	x^{n_i}
`x_{i_j}`	x_{i_j}	`x_{i^2}`	x_{i^2}	`{x^2}^2`	x^{2^2}	`x^{2^2}`	x^{2^2}
`x_{i_i}`	x_{i_i}	`{x_i}_i`	x_{ii}	`x_i_j`	erreur	`x_i^2_j`	erreur

On considère la grammaire G_1 définie par

$$S \to a \mid \{S\} \mid S\,\hat{}\,S \mid S\_S$$

ainsi que la grammaire G_2 définie par

$$\begin{array}{llll} 1 : S \to T & 2 : S \to T\,\hat{}\,T & 4 : S \to T\,\hat{}\,T\_T & 6 : T \to a \\ & 3 : S \to T\_T & 5 : S \to T\_T\,\hat{}\,T & 7 : T \to \{S\} \end{array}$$

On note G_2' la grammaire G_2 augmentée par la règle $0 :\ S' \to S$.

[2] **a)** Montrer que la grammaire G_1 est ambiguë et montrer qu'elle engendre des expressions non autorisées en LaTeX.
Montrer que G_2 engendre `a^{a_a}_a` et dessiner un arbre de dérivation pour ce mot.
Montrer que G_2 n'engendre pas le mot `a^a_a_a`.

[2] **b)** Montrer que la grammaire G_2 n'est pas LL.
Calculer First_1 et Follow_1 pour les variables S', S et T. Détailler les calculs.

[6] **c)** Calculer l'automate $\mathcal{C}_0$ des contextes pour les 0-items.
On se restreindra bien sûr aux états accessibles. On pourra représenter un état de façon concise sous la forme $\text{clot}(W)$ en donnant seulement les éléments de W et pas tous les éléments de la clôture.
Donner la table d'analyse SLR de la grammaire G_2'.
Y a-t-il des conflits ? La grammaire est-elle ambiguë ?

On note Σ l'alphabet terminal des grammaires ci-dessus. Soit A l'alphabet réduit aux deux accolades { et } et π_A la projection de Σ sur A.

[2] **d)** Donner une grammaire G_3 qui engendre $\pi_A(\mathcal{L}_{G_2}(S))$.
Il faut bien sûr prouver que G_3 engendre bien ce langage.

[2] **e)** Montrer que tout mot engendré par G_2 est bien "parenthésé". Formellement, il faut prouver que $\pi_A(\mathcal{L}_{G_2}(S))$ est inclus dans le langage de Dyck D_1^* sur l'alphabet A.
Montrer que l'inclusion est stricte.

2 Distance d'édition

On définit 3 *opérations d'édition* sur les mots : insertion d'une lettre, suppression d'une lettre, modification d'une lettre. Par exemple, on peut passer du mot $abacacb$ au mot $aabaab$ au moyen de 3 opérations : insertion d'un a au début et suppression des deux c, ou encore suppression du premier b, modification des deux c en b et a respectivement.

La distance d'édition $d(u, v)$ entre deux mots u et v est le nombre minimal d'opérations d'édition qui permettent de passer de u à v. Par exemple, $d((abc)^3, (bca)^3) = 2$.

Si $L \subseteq \Sigma^*$ est un langage et $k \in \mathbb{N}$ un entier, on note $L_k = \{v \in \Sigma^* \mid \exists u \in L,\ d(u, v) \leq k\}$.

[4] **a)** Soit $\Sigma = \{a, b, c\}$ et $L = \Sigma^* ab \Sigma^* ab \Sigma^*$.
Montrer que $L_1 = \Sigma^* ab \Sigma^* (a + b) \Sigma^* \cup \Sigma^* (a + b) \Sigma^* ab \Sigma^*$.
Calculer l'automate minimal de L_1.

[2] **b)** Montrer que si $L \subseteq \Sigma^*$ est reconnu par un automate (déterministe ou non) ayant n états, alors le langage L_1 peut être reconnu par un automate non déterministe ayant $\mathcal{O}(n)$ états.
En déduire que l'on peut construire un automate déterministe pour L_1 ayant au plus $2^{\mathcal{O}(n)}$ états à partir d'un automate (déterministe ou non) pour L ayant n états.

[3] **c)** Soit $\Sigma = \{a, b\}$ et $L = \Sigma^* aa \Sigma^n$.
Donner un automate non déterministe reconnaissant L avec $\mathcal{O}(n)$ états.
Montrer que $L_1 = \Sigma^* a \Sigma^n \cup \Sigma^* a \Sigma^{n+1}$.
Montrer que l'automate minimal de L_1 a au moins $2^{\frac{n}{2}}$ états.

[3] **d)** Soit $L \subseteq \Sigma^*$ un langage rationnel, on définit la fonction partielle $f : \Sigma^* \to \Sigma^*$ dont le domaine est L_1 et qui à un mot $v \in L_1$ associe le plus petit mot dans l'ordre lexicographique de $\{u \in L \mid d(u, v) \leq 1\}$. Montrer que la fonction f n'est pas forcément séquentielle.

Langages Formels

Licence Informatique – ENS Cachan

Partiel du 18 mars 2010

durée 3 heures

Document autorisé : polycopié du cours.
Toutes les réponses devront être correctement justifiées.
Le chiffre en regard d'une question est une indication sur sa difficulté ou sa longueur.

1 Automates

Remarque : Toutes les questions du type "Donner un automate qui ..." doivent être justifiées, *i.e.*, il faut donner l'automate mais aussi prouver qu'il vérifie bien la propriété requise.

On considère l'alphabet $\Sigma = \{a, b, c\}$ et le langage $L = \Sigma^* ab \Sigma^* ab \Sigma^*$ des mots contenant au moins 2 occurrences du facteur ab.

[1] **a)** Donner un automate non déterministe avec 5 états qui reconnaît le langage L.
Montrer que l'on ne peut pas reconnaître le langage L avec un automate, déterministe ou non, ayant au plus 4 états.

[1] **b)** Donner un automate déterministe pour le langage L.

[1] **c)** Donner l'automate minimal du langage L.

On considère maintenant l'alphabet $\Sigma = \{a, d, c, d\}$ et l'ensemble T des arbres étiquetés par Σ dont les nœuds internes sont binaires et étiquetés par d et la frontière est dans le langage L défini ci-dessus.

[3] **d)** Donner un automate d'arbres déterministe ascendant qui reconnaît le langage T.

[2] **e)** Montrer que le langage T ne peut pas être reconnu par un automate d'arbres déterministe descendant.

2 Langages de Dyck

Soit $\Sigma_n = \{a_1, \ldots, a_n\} \cup \{\bar{a}_1, \ldots, \bar{a}_n\}$ l'alphabet formé de n paires de parenthèses.
Soit $G_n = (\Sigma_n, \{S\}, P_n, S)$ la grammaire dont les règles sont

$$S \to a_1 S \bar{a}_1 S + \cdots + a_n S \bar{a}_n S + \varepsilon \ .$$

Le langage $D_n^* = \mathcal{L}_{G_n}(S)$ est appelé langage de Dyck sur n paires de parenthèses

[2] **a)** Montrer que $D_n^* = \mathcal{L}_{G'_n}(S)$ où la grammaire $G'_n = (\Sigma_n, \{S\}, P'_n, S)$ a pour règles

$$S \to SS + a_1 S \bar{a}_1 + \cdots + a_n S \bar{a}_n + \varepsilon \ .$$

[2] **b)** Montrer que $D_1^* = \{w \in \Sigma_1^* \mid |w|_{a_1} = |w|_{\bar{a}_1}$ et $|v|_{a_1} \geq |v|_{\bar{a}_1}$ pour tous préfixes $v \leq w\}$.

On considère le système de réécriture (type 0) $G''_n = (\Sigma_n, P''_n)$ dont les règles sont

$$P''_n = \{a_i \bar{a}_i \to \varepsilon \mid 1 \leq i \leq n\} \ .$$

[3] **c)** Montrer que $D_n^* = \{w \in \Sigma_n^* \mid w \xrightarrow{*} \varepsilon$ dans $G''_n\}$.

[1] **d)** Soit $\mathcal{A} = (Q, \Sigma_n, T, I, F)$ un automate fini. Étant donné un couple $(p, q) \in Q^2$, montrer que l'on peut décider s'il existe un mot de Dyck $w \in D_n^*$ qui est l'étiquette d'un chemin de p à q dans $\mathcal{A} : p \xrightarrow{w} q$.

Soit Γ un alphabet disjoint de Σ_n, $\Sigma = \Sigma_n \cup \Gamma$ et $L \subseteq \Gamma^*$ un langage.
On définit la clôture $\mathrm{clot}(L) = \{v \in \Gamma^* \mid \exists w \in L$ tel que $w \xrightarrow{*} v$ dans $G''_n\}$.

[3] **e)** Montrer que si L est reconnaissable, alors $\mathrm{clot}(L)$ aussi.

On définit la réduction $\mathrm{red}(L) = \{v \in \mathrm{clot}(L) \mid v \not\to$ dans $G''_n\}$.

[1] **f)** Montrer que si L est reconnaissable, alors $\mathrm{red}(L)$ aussi.

[4] **g)** Soit $\mathcal{A} = (Q, \Sigma_n, T, I, F)$ un automate fini. On définit pour $k \geq 0$ les ensembles

$$\begin{aligned}
R &= \{(p, q) \in Q^2 \mid \exists w \in D_n^* \text{ tel que } p \xrightarrow{w} q \text{ dans } \mathcal{A}\} \\
R_0 &= \{(p, p) \mid p \in Q\} \\
R_{k+1} &= \{(p, q) \mid \exists r \in Q \text{ tel que } (p, r) \in R_k \wedge (r, q) \in R_k\} \cup \\
&\qquad \{(p, q) \mid \exists (p', q') \in R_k,\ \exists i \in \{1, \ldots, n\} \text{ tels que } p \xrightarrow{a_i} p' \wedge q' \xrightarrow{\bar{a}_i} q \text{ dans } \mathcal{A}\}
\end{aligned}$$

Montrer que la suite $(R_k)_{k \geq 0}$ est croissante et que $R = \bigcup_{k \geq 0} R_k$.
En déduire un algorithme pour calculer R et donner sa complexité en fonction de la taille de $\mathcal{A}$.

Langages Formels

Licence Informatique – ENS Cachan

Examen du 27 mai 2009

durée 3 heures

Document autorisé : polycopié du cours.

Toutes les réponses devront être correctement justifiées.

Le chiffre en regard d'une question est une indication sur sa difficulté ou sa longueur.

1 Addition d'Avizienis

Soit $k > 1$ un entier. Un système d'Avizienis est un procédé de numération en base k qui utilise des chiffres positifs et négatifs. De façon générale, si $D \subseteq \mathbb{Z}$ est un ensemble fini de chiffres, à chaque mot $u = a_n \cdots a_1 a_0 \in D^*$ (avec $a_i \in D$) on associe sa *valeur*, i.e., l'entier représenté par u en base k :

$$\mathrm{val}_k(u) = \sum_{i=0}^{n} a_i k^i \ .$$

Pour plus de lisibilité dans l'écriture d'un mot on utilisera de préférence $\bar{a}$ à la place du chiffre $-a$. Par exemple, $\mathrm{val}_2(10\bar{1}) = 3$ et $\mathrm{val}_{10}(3\bar{3}) = 27$.

Avec $A = \{0, \ldots, k-1\}$, les mots de $X = (A \setminus \{0\})A^* \cup \{\varepsilon\}$ correspondent à l'écriture usuelle des entiers en base k en commençant par le chiffre de poids fort. Cette écriture étant unique, l'application $\mathrm{val} : X \to \mathbb{N}$ est une bijection.

Soit $h = \lfloor \frac{k+1}{2} \rfloor$ et $B = \{-h, \ldots, 0, \ldots, h\}$. Le système de numération d'Avizienis correspond aux mots de B^* (on peut éventuellement se restreindre à ceux qui ne commencent pas par 0). Dans ce système, la représentation d'un entier n'est pas unique. Par exemple, $\mathrm{val}_4(12\bar{2}) = \mathrm{val}_4(112)$.

[3] **a)** Montrer qu'il existe une fonction séquentielle $f : A^* \to B^*$ qui préserve les valeurs, i.e., telle que $\mathrm{val}_k(u) = \mathrm{val}_k(f(u))$ pour tout $u \in A^*$.

[1] **b)** Soit $g : B^* \to A^*$ la fonction qui traduit une représentation d'Avizienis en la représentation usuelle : $g(B^*) = X$ et pour tout $u \in B^*$ on a $\mathrm{val}_k(u) = \mathrm{val}_k(g(u))$.
La fonction g est-elle séquentielle ?

[3] **c)** On considère maintenant l'alphabet $C = \{-2h, \ldots, 0, \ldots, 2h\}$. Montrer qu'il existe une fonction séquentielle $f : C^* \to B^*$ qui préserve les valeurs, i.e., telle que $\mathrm{val}_k(u) = \mathrm{val}_k(f(u))$ pour tout $u \in C^*$.
Indication : On pourra prendre comme ensemble d'états $Q = B^2 \setminus \{\bar{h}\bar{h}, hh\}$.

[1] **d)** Existe-t-il une fonction séquentielle $g : (B \times B)^* \to B^*$ qui réalise l'addition dans le système d'Avizienis en commençant par le bit de poids fort, i.e., telle que pour tout $w = (a_n, b_n) \cdots (a_0, b_0) \in (B \times B)^*$ on a $\mathrm{val}_k(g(w)) = \mathrm{val}_k(a_n \cdots a_0) + \mathrm{val}_k(b_n \cdots b_0)$?

2 Unaire ou binaire

Soit $u \in \{0,1\}^*$. On note $\overline{u}^2 \in \mathbb{N}$ l'entier qui s'écrit u en binaire avec le bit de poids fort à gauche. Par exemple, $\overline{1100}^2 = 12$. Pour $K \subseteq \mathbb{N}$, on note

$$\begin{aligned} \text{unary}(K) &= \{v \in \{1\}^* \mid |v| \in K\} \\ \text{binary}(K) &= \{u \in \{0,1\}^* \mid \overline{u}^2 \in K\} \ . \end{aligned}$$

[1] **a)** La proposition suivante est-elle vraie ou fausse ?

(i) Pour tout $K \subseteq \mathbb{N}$, si binary(K) est reconnaissable alors unary(K) est reconnaissable.

[3] **b)** La proposition suivante est-elle vraie ou fausse ?

(ii) Pour tout $K \subseteq \mathbb{N}$, si unary(K) est reconnaissable alors binary(K) est reconnaissable.

3 Permutation et Conjugaison

Soit Σ un alphabet et $L \subseteq \Sigma^*$ un langage. On définit

$$\begin{aligned} \text{permut}(L) &= \{v \in \Sigma^* \mid \exists u \in L,\ \forall a \in \Sigma,\ |v|_a = |u|_a\} \\ \text{conjug}(L) &= \{w \in \Sigma^* \mid \exists u, v \in \Sigma^*,\ uv \in L \text{ et } w = vu\} \ . \end{aligned}$$

[1] **a)** Les propositions suivantes sont-elles vraies ou fausses ?

(i) Pour tout $L \subseteq \Sigma^*$, si L est reconnaissable alors permut(L) est reconnaissable.

(ii) Pour tout $L \subseteq \Sigma^*$, si L est algébrique alors permut(L) est algébrique.

[3] **b)** Montrer que pour tout $L \subseteq \Sigma^*$, si L est reconnaissable alors conjug(L) est reconnaissable.

[2] **c)** Soit $L = \{a^n b^n c^p d^p \mid n, p \geq 0\}$. Montrer que conjug$(L)$ est algébrique.

[3] **d)** Soit $L = \{a^n b^n \mid n \geq 0\}$. Le langage conjug$(L)$ est-il linéaire ?

[5] **e)** La proposition suivante est-elle vraie ou fausse ?

(iii) Pour tout $L \subseteq \Sigma^*$, si L est algébrique alors conjug(L) est algébrique.

Langages Formels

Licence Informatique – ENS Cachan

Partiel du 25 mars 2009

durée 3 heures

Document autorisé : polycopié du cours.

Toutes les réponses devront être correctement justifiées.

Le chiffre en regard d'une question est une indication sur sa difficulté ou sa longueur.

1 Rationnels, algébriques et automates à pile

Le but de l'exercice est de prouver que les langages des mots de pile d'un automate à pile sont reconnaissables.

On considère un alphabet $A = \{a_1, \bar{a}_1, \ldots, a_n, \bar{a}_n\}$ et le langage de Dyck $D_n^* \subseteq A^*$ engendré par la grammaire

$$S \to \varepsilon + \sum_{i=1}^{n} a_i S \bar{a}_i S$$

[1] **a)** Montrer que $D_n^* \cdot D_n^* = D_n^*$.

On définit la relation de *réduction*

$$\Longrightarrow = \{(ua_i\bar{a}_i v\,,\, uv) \mid u, v \in A^* \text{ et } 1 \leq i \leq n\}$$

et on note $\overset{*}{\Longrightarrow}$ la clôture réflexive et transitive de $\Longrightarrow$.

[3] **b)** Montrer que $D_n^* = \{w \in A^* \mid w \overset{*}{\Longrightarrow} \varepsilon\}$.

On considère un automate fini $\mathcal{A} = (Q, A, T, I, F)$ avec $T \subseteq Q \times A \times Q$. Pour $p, q \in Q$ on définit $\mathcal{L}_{p,q}(\mathcal{A})$ comme l'ensemble des mots $w \in A^*$ acceptés par $\mathcal{A}$ si on prend p comme unique état initial et q comme unique état final. Enfin, on définit

$$X = \{(p, q) \in Q^2 \mid \mathcal{L}_{p,q}(\mathcal{A}) \cap D_n^* \neq \emptyset\}$$

[1] **c)** Montrer que la relation X est réflexive, transitive et qu'elle est calculable, i.e., étant donné $(p, q) \in Q^2$, on peut décider si $(p, q) \in X$

On définit par récurrence des relations $X_k \subseteq Q^2$ par

$$\begin{aligned} X_0 &= \{(p, p) \mid p \in Q\} \\ X_{k+1} &= X_k \cup \{(p, q) \mid \exists\, p', q', r' \in Q,\ \exists\, 1 \leq i \leq n \text{ avec} \\ &\qquad (p, a_i, p') \in T,\ (p', q') \in X_k,\ (q', \bar{a}_i, r') \in T \text{ et } (r', q) \in X_k\} \end{aligned}$$

[2] **d)** Montrer que $X = \bigcup_{k \geq 0} X_k$.

On définit l'automate $\mathcal{A}' = (Q, A, T', I, F)$ en ajoutant des ε-transitions à l'automate $\mathcal{A}$:

$$T' = T \cup \{(p, \varepsilon, q) \mid (p, q) \in X\}$$

[1] **e)** Montrer que $\mathcal{L}(\mathcal{A}')$ est fermé par réduction, i.e., si le mot $ua_i\bar{a}_iv$ est accepté par $\mathcal{A}'$ alors il en est de même du mot uv.

[2] **f)** Montrer que $\mathcal{L}(\mathcal{A}') = \{v \in A^* \mid \exists\, w \in \mathcal{L}(\mathcal{A}) \text{ avec } w \overset{*}{\Longrightarrow} v\}$.

[1] **g)** Soient $u, v, v' \in A^*$ trois mots. Montrer que si $u \Longrightarrow v$ et $u \Longrightarrow v'$ alors $v = v'$ ou il existe $w \in A^*$ tel que $v \Longrightarrow w$ et $v' \Longrightarrow w$.

[3] **h)** En déduire que la relation de réduction est confluente, i.e., pour tous $u, v, v' \in A^*$, si $u \overset{*}{\Longrightarrow} v$ et $u \overset{*}{\Longrightarrow} v'$ alors il existe $w \in A^*$ tel que $v \overset{*}{\Longrightarrow} w$ et $v' \overset{*}{\Longrightarrow} w$.

Un mot $v \in A^*$ est irréductible s'il n'existe pas de mot $w \in A^*$ tel que $v \Longrightarrow w$.

[1] **i)** Montrer que pout tout $v \in A^*$ il existe un unique $w \in A^*$ irréductible tel que $v \overset{*}{\Longrightarrow} w$.

On note $\rho : A^* \to A^*$ l'application qui associe à un mot $v \in A^*$ l'unique mot irréductible $\rho(v) \in A^*$ tel que $v \overset{*}{\Longrightarrow} \rho(v)$.

[2] **j)** Montrer que si $L \subseteq A^*$ est un langage rationnel, alors $\rho(L)$ est aussi rationnel.

Dans la suite, $\mathcal{A} = (Q, \Sigma, Z, T)$ sera un automate à pile avec l'ensemble fini de transitions $T \subseteq Q \times Z \times (\Sigma \cup \{\varepsilon\}) \times Q \times Z^*$. On considère T comme un alphabet et on introduit pour $q \in Q$ les sous-alphabets

$$G_q = T \cap (\{q\} \times Z \times (\Sigma \cup \{\varepsilon\}) \times Q \times Z^*)$$
$$D_q = T \cap (Q \times Z \times (\Sigma \cup \{\varepsilon\}) \times \{q\} \times Z^*)$$

Pour $p, q \in Q$, on définit le langage $L_{p,q} \subseteq T^*$ par $\varepsilon \in L_{p,q}$ si $p = q$ et

$$L_{p,q} \cap T^+ = (G_p \cdot T^* \cap T^* \cdot D_q) \setminus \bigcup_{r \neq s} T^* \cdot D_r \cdot G_s \cdot T^*$$

On suppose que l'alphabet de pile est $Z = \{a_1, a_2, \ldots, a_n\}$ et on définit le morphisme $h : T^* \to A^*$ par $h((p, z, x, q, u)) = \bar{z}u$ (on rappelle que $A = \{a_1, \bar{a}_1, \ldots, a_n, \bar{a}_n\}$).

Finalement, pour $(p, z, q) \in Q \times Z \times Q$, on définit

$$K_{p,z,q} = \rho(z \cdot h(L_{p,q})) \cap Z^*$$

[1] **k)** Montrer que $K_{p,z,q}$ est un langage rationnel.

[2] **l)** Soit $\tau \in L_{p,q}$ tel que $u = \rho(zh(\tau)) \in Z^*$. Montrer qu'il existe un calcul $(p, z) \overset{*}{\to} (q, u)$ dans l'automate à pile $\mathcal{A}$.

[3] **m)**Montrer que $K_{p,z,q} = \{u \in Z^* \mid \exists\, (p, z) \overset{*}{\to} (q, u) \text{ dans } \mathcal{A}\}$.

Langages Formels

Licence Informatique – ENS Cachan

Examen du 28 mai 2008

durée 3 heures

Document autorisé : polycopié du cours.
Les exercices sont indépendants.
Toutes les réponses devront être correctement justifiées.

1 Fonctions séquentielles

On appelle machine de Moore un automate déterministe dont les états génèrent des sorties. Formellement, une machine de Moore est un tuple $\mathcal{B} = (Q, A, B, \delta, q_0, F, \lambda)$ où (Q, A, δ, q_0, F) est un automate déterministe et $\lambda : Q \to B^*$ est la fonction (totale) de sortie. La sémantique de $\mathcal{B}$ est une fonction partielle $[\![\mathcal{B}]\!] : A^* \to B^*$ définie sur un mot $u = a_1 a_2 \dots a_{|u|} \in A^*$ si $\delta(q_0, u) \in F$ et dans ce cas $[\![\mathcal{B}]\!](u) = \lambda(q_0)\lambda(q_1)\dots\lambda(q_{|u|})$ où $q_i = \delta(q_0, a_1 \dots a_i)$ pour tout $1 \leq i \leq |u|$.

a) Soient $A = \{0,1\}^2$, $B = \{0,1\}$ et $f : A^* \to B^*$ la soustraction. Formellement, une donnée $(a_1, b_1)\dots(a_p, b_p)$ est interprétée comme un couple (n, m) d'entiers de représentations binaires (avec bit de poids faible en premier) $a_1 \dots a_p$ et $b_1 \dots b_p$. La fonction f est définie sur la donnée (n, m) si $n \geq m$ et le résultat est alors $n - m$ toujours représenté en binaire avec bit de poids faible en premier.
Donner une machine de Moore qui réalise cette soustraction, i.e., qui représente la fonction f.

b) On considère le codage défini par $f(x) = a$ et $f(y) = aba$. Peut-on réaliser le codage $f : \{x, y\}^* \to \{a, b\}^*$ avec une machine de Moore ? Peut-on réaliser le décodage f^{-1} avec une machine de Moore ?

c) Montrer que toute fonction séquentielle pure peut être réalisée par une machine de Moore. Qu'en est-il de la réciproque ?

d) Montrer que toute fonction réalisable par une machine de Moore est séquentielle. Qu'en est-il de la réciproque ?

2 Analyse LL

On considère la grammaire G sur l'alphabet $\Sigma = \{\texttt{id}, \texttt{num}, :=, [,]\}$ définie par

$$\begin{aligned} \textbf{Stmt} &\to \texttt{id} \mid \textbf{Var} := \textbf{Exp} \\ \textbf{Var} &\to \texttt{id} \mid \textbf{Var}[\textbf{Exp}] \\ \textbf{Exp} &\to \texttt{num} \mid \textbf{Var} \end{aligned}$$

a) Pour chaque variable x, calculer $\text{First}_2(x)$ et $\text{Follow}_2(x)$.

b) Montrer que G n'est pas fortement LL(2).

c) Existe-t-il k tel que G soit une grammaire LL(k) ?

3 Automates d'arbres

Soit $A_p = \{1, \ldots, p\}$, Σ un alphabet et $t : A_p^* \to \Sigma$ un arbre. Une feuille de t est un mot $u \in A_p^*$ tel que $u \in \mathrm{dom}(t)$ et $u \cdot 1 \notin \mathrm{dom}(t)$. Un chemin de t est un mot

$$t(\varepsilon) \cdot i_1 \cdot t(i_1) \cdot i_2 \cdot t(i_1 i_2) \cdots i_n \cdot t(i_1 i_2 \ldots i_n)$$

tel que $i_1 i_2 \ldots i_n$ est une feuille de t. On note $\mathcal{C}(t) \subseteq \Sigma^*$ l'ensemble des chemins de t. Si $L \subseteq T_p(\Sigma)$ est un langage d'arbres, on note $\mathcal{C}(L) \subseteq \Sigma^*$ l'ensemble des chemins des arbres de L.

a) Montrer que si $L \subseteq T_p(\Sigma)$ est un langage d'arbres reconnaissable alors $\mathcal{C}(L) \subseteq \Sigma^*$ est un langage reconnaissable de mots.

Si L est un langage d'arbres, on définit la *clôture par chemins* de L comme l'ensemble $\mathcal{CC}(L) \subseteq T_p(\Sigma)$ des arbres t tels que $\mathcal{C}(t) \subseteq \mathcal{C}(L)$.

b) Montrer que si $L \subseteq T_p(\Sigma)$ est un langage d'arbres reconnaissable alors $\mathcal{CC}(L)$ aussi.

c) Montrer qu'un langage d'arbres $L \subseteq T_p(\Sigma)$ est reconnaissable par un automate déterministe descendant si et seulement si $L = \mathcal{CC}(L)$.

d) Peut-on décider si un langage d'arbres reconnaissable $L \subseteq T_p(\Sigma)$ est clos par chemins, i.e., si $L = \mathcal{CC}(L)$?

4 Automates à pile

On s'intéresse ici à des automates dont l'alphabet de pile Γ est un singleton $\{z\}$.

a) Montrer que le langage $L = \{a^n b^m c \mid n \geq m \geq 1\}$ peut être accepté *par pile vide et état final* par un automate dont l'alphabet de pile est un singleton.

b) Montrer que le langage $L = \{a^n b^m c \mid n \geq m \geq 1\}$ ne peut pas être accepté *par pile vide* par un automate dont l'alphabet de pile est un singleton.

Langages Formels

Licence Informatique – ENS Cachan

Partiel du 19 mars 2008

durée 2 heures

Document autorisé : polycopié du cours.
Les exercices sont indépendants.
Toutes les réponses devront être correctement justifiées.

1 Automates à pile

a) Donner un automate à pile déterministe et temps réel pour le langage

$$L = \{w \in \{a, b\}^* \mid |w|_a = 2|w|_b\} .$$

2 Grammaires

Soit $G = (\Sigma, V, P, S)$ une grammaire algébrique. Une variable $x \in V$ est *strictement récursive* (SR) s'il existe une dérivation $x \xrightarrow{*} uxv$ dans G avec $u, v \in \Sigma^+$.

L'objectif est de montrer qu'une grammaire algébrique sans variable SR engendre un langage rationnel.

a) Montrer que l'on peut décider si une grammaire algébrique contient des variables SR.
Indication : Étant données une grammaire algébrique $G = (\Sigma, V, P, S)$ et une variable $x \in V$, montrer que l'on peut effectivement calculer l'ensemble

$$Z = \{y \in V \mid \exists\, x \xrightarrow{*} uyv \text{ avec } u, v \in \Sigma^+\} .$$

b) Soit $G = (\Sigma, V, P, S)$ une grammaire algébrique sans variable SR. Montrer que l'on peut effectivement construire une grammaire algébrique $G' = (\Sigma, V', P', S')$ réduite, propre, sans variable SR et telle que $\mathcal{L}_{G'}(S') = \mathcal{L}_G(S) \setminus \{\varepsilon\}$.

c) Soit $G = (\Sigma, V, P, S)$ une grammaire algébrique réduite, en forme normale de Greibach et sans variable SR. Montrer qu'il existe une constante $K \in \mathbb{N}$ telle que pour toute dérivation *gauche* $x \xrightarrow{*} u\alpha$ avec $u \in \Sigma^*$ et $\alpha \in V^*$ on a $|\alpha| \leq K$.

d) Soit $G = (\Sigma, V, P, S)$ une grammaire algébrique réduite, en forme normale de Greibach et sans variable SR. Montrer que le langage engendré $\mathcal{L}_G(S)$ est rationnel.

Soit $G = (\Sigma, V, P, S)$ une grammaire algébrique *propre*. Soit $x \in V$ et soient

$$A = \{\alpha \in (\Sigma \cup V)^* \mid x \to x\alpha \in P\}$$
$$B = \{\beta \in (\Sigma \cup V \setminus \{x\})(\Sigma \cup V)^* \mid x \to \beta \in P\}$$

On définit la grammaire $G' = (\Sigma, V', P', S)$ en ajoutant une variable x' ($V' = V \uplus \{x'\}$) et en remplaçant les règles de G issues de x par les règles $x \to \beta + \beta x'$ pour $\beta \in B$ et $x' \to \alpha + \alpha x'$ pour $\alpha \in A$.

e) Montrer que si $y \xrightarrow{*} uzv$ dans G' avec $y, z \in V$ et $u, v \in \Sigma^*$, alors $y \xrightarrow{*} uzv$ dans G.

f) Montrer que si G n'a pas de variable SR alors il en est de même pour G'.

g) Soit $G = (\Sigma, V, P, S)$ une grammaire algébrique sans variable SR. Montrer que l'on peut effectivement construire une grammaire algébrique $G' = (\Sigma, V', P', S')$ réduite, en forme normale de Greibach, sans variable SR et telle que $\mathcal{L}_{G'}(S') = \mathcal{L}_G(S) \setminus \{\varepsilon\}$.

h) En déduire qu'un langage est rationnel si et seulement si il peut être engendré par une grammaire algébrique sans variable SR.

3 Grammaires contextuelles

a) Le langage $L = \{a^p \mid p \text{ est premier}\}$ peut-il être engendré par une grammaire contextuelle ?

Théorie des Langages

Magistère STIC

Examen du 31 mai 2007

durée 3 heures

Document autorisé : polycopié du cours.
Les exercices sont indépendants.
Toutes les réponses devront être correctement justifiées.

1 Indices et exposants

On s'intéresse dans cette partie aux expressions LaTeX utilisant des indices et des exposants comme par exemple :

`x_i`	x_i	`x_{i+1}`	x_{i+1}	`x^n`	x^n	`x^{n+1}`	x^{n+1}
`x_i^2`	x_i^2	`x^2_i`	x_i^2	`x^{n^2}`	x^{n^2}	`x^{n_i}`	x^{n_i}
`x_{i_j}`	x_{i_j}	`x_{i^2}`	x_{i^2}	`{x^2}^2`	x^{2^2}	`x^{2^2}`	x^{2^2}
`x_{i_i}`	x_{i_i}	`{x_i}_i`	x_{ii}	`x_i_j`	erreur	`x_i^2_j`	erreur

On considère la grammaire G_1 définie par $S \to a \mid \{S\} \mid S\,\hat{}\,S \mid S\_S$.

a) Montrer que la grammaire G_1 est ambiguë et montrer qu'elle engendre des expressions non autorisées en LaTeX.

On considère la grammaire G_2 définie par

$$S \to T \mid T\,\hat{}\,T \mid T\_T \mid T\,\hat{}\,T\_T \mid T\_T\,\hat{}\,T$$
$$T \to a \mid \{S\}$$

On note G_2' la grammaire G_2 augmentée par la règle $S' \to S$.

b) Montrer que G_2 engendre `a^{a_a}_a` et dessiner un arbre de dérivation pour ce mot. Montrer que G_2 n'engendre pas le mot `a^a_a_a`.

c) Montrer que la grammaire G_2 n'est pas LL.

d) Calculer Follow_1 pour les variables S', S et T.

e) Calculer l'automate $\mathcal{C}_0$ des contextes pour les 0-items.
On se restreindra bien sûr aux états accessibles. On pourra représenter un état de façon concise sous la forme clot(W) en donnant seulement les éléments de W et pas tous les éléments de la clôture.

f) Donner la table d'analyse SLR de la grammaire G_2'. Y a-t-il des conflits ? La grammaire est-elle ambiguë ?

g) Montrer que tout mot engendré par G_2 est bien "parenthésé". Formellement, on note A l'alphabet réduit aux deux accolades { et }. Montrer que la projection de $\mathcal{L}_{G_2}(S)$ sur A est contenue dans le langage de Dyck D_1^* sur l'alphabet A. Montrer que l'inclusion est stricte.

h) (Facultatif) Montrer que G_2 n'engendre aucun mot de la forme $u\,\hat{}\,v\,\hat{}\,w$ avec $v \in D_1^*$.

2 Automates à un pic

Un *automate à un pic* est un automate à pile 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 à un pic* est un langage qui peut être accepté *par pile vide* par un automate à un pic.

a) Montrer que le langage $L = \{a^n b^n \mid n > 0\} \cup \{a^n b^{2n} \mid n > 0\}$ est un langage linéaire.

b) Montrer que L est un langage à un pic.

c) Montrer que tout langage linéaire est un langage à un pic.

d) Montrer que le langage $K = \{a^{i_1} b a^{i_2} b \cdots a^{i_n} b \mid n > 0 \text{ et } \exists j, i_j \neq j\}$ est un langage à un pic.

e) (Facultatif) Trouver une grammaire linéaire qui engendre K.

f) Soit $G = (\Sigma, V, P)$ une grammaire vérifiant :

1. $V = V_1 \uplus V_2$ avec $V_1 \cap V_2 = \emptyset$.
2. $P_1 = P \cap (V_1 \times (\Sigma \cup V)^*)$ satisfait $P_1 \subseteq V_1 \times (V_1\Sigma^* \cup \Sigma^*)$, i.e., les règles issues des variables de V_1 sont linéaires gauches et ne font pas intervenir les variables de V_2.
3. $P_2 = P \cap (V_2 \times (\Sigma \cup V)^*)$ satisfait $P_2 \subseteq V_2 \times (\Sigma^* V_2 V_1 \cup \Sigma^*(V_1 \cup V_2)\Sigma^* \cup \Sigma^*)$.

Montrer que pour tout $x \in V$, $\mathcal{L}_G(x)$ est un langage linéaire.

g) Même question en remplaçant la condition 3 ci-dessus par

$$P_2 \subseteq V_2 \times (\Sigma^* V_2 V_1^* \cup \Sigma^*(V_1 \cup V_2)\Sigma^* \cup \Sigma^*).$$

Soit $\mathcal{A} = (Q, \Sigma, Z, T, (q_0, z_0))$ un automate à pile arbitraire.

Pour $p, q \in Q$ et $z \in Z$ on définit $K_{p,z,q} \subseteq \Sigma^*$ comme l'ensemble des mots $w \in \Sigma^*$ tels qu'il existe un calcul $p, z \xrightarrow{w} q, \varepsilon$ dans $\mathcal{A}$ dans lequel la taille de la pile ne dépasse pas 1.

h) Montrer que $K_{p,z,q}$ est un langage rationnel.

Pour $p, q \in Q$ et $z \in Z$ on définit $L_{p,z,q} \subseteq \Sigma^*$ comme l'ensemble des mots $w \in \Sigma^*$ tels qu'il existe un calcul $p, z \xrightarrow{w} q, \varepsilon$ dans $\mathcal{A}$ qui soit *à un pic*, i.e., durant ce calcul la pile n'augmente plus une fois qu'elle a diminué.

i) Construire une grammaire vérifiant les conditions de la question **(g)** et dans laquelle les variables de V_2 engendrent les langages du type $L_{p,z,q}$ et les variables de V_1 engendrent les langages du type $K_{p,z,q}$.

j) Montrer que tout langage à un pic est un langage linéaire.

Théorie des Langages

Magistère STIC

Partiel du 28 mars 2007

durée 2 heures

Document autorisé : polycopié du cours.
Les exercices sont indépendants.
Toutes les réponses devront être correctement justifiées.

1 Langages locaux, algorithme de Glushkov

Un langage L sur l'alphabet A est dit *local* s'il existe des parties P et S de A et une partie N de A^2 telles que $L \setminus \{\varepsilon\} = (PA^* \cap A^*S) \setminus A^*NA^*$.

a) Montrer que tout langage local est reconnu par un automate déterministe ayant au plus $|A| + 1$ états.

b) Montrer qu'il existe des langages reconnaissables qui ne sont pas locaux.

c) Montrer que tout langage reconnaissable est l'image par un morphisme strictement alphabétique (c'est-à-dire que l'image d'une lettre est une lettre) d'un langage local.

d) Soient L_1 et L_2 deux langages locaux sur des alphabets A_1 et A_2 disjoints ($A_1 \cap A_2 = \emptyset$). Montrer que $L_1 \cup L_2$ et $L_1 \cdot L_2$ sont des langages locaux.

e) Soit L un langage local sur l'alphabet A. Montrer que L^* est un langage local.

f) Une expression rationnelle est dite *linéaire* si chaque lettre a au plus une occurrence dans l'expression. Montrer qu'une expression linéaire représente un langage local.

g) Montrer que tout langage rationnel est l'image par un morphisme strictement alphabétique d'un langage représenté par une expression rationnelle linéaire.

Soit L un langage sur l'alphabet A, on définit les ensembles

$$\begin{aligned} P(L) &= \{a \in A \mid aA^* \cap L \neq \emptyset\}, \\ S(L) &= \{a \in A \mid A^*a \cap L \neq \emptyset\}, \\ F(L) &= \{u \in A^2 \mid A^*uA^* \cap L \neq \emptyset\}. \end{aligned}$$

h) Montrer que si L est un langage local alors

$$L \setminus \{\varepsilon\} = (P(L)A^* \cap A^*S(L)) \setminus A^*(A^2 \setminus F(L))A^*.$$

i) Donner un algorithme qui, étant donnée une expression rationnelle représentant un langage L, détermine si $\varepsilon \in L$ et calcule les ensembles $P(L)$, $S(L)$ et $F(L)$.

j) En utilisant les questions précédentes, donner un algorithme pour construire un automate (non déterministe) à partir d'une expression rationnelle arbitraire. Exprimer le nombre d'états de l'automate en fonction de l'expression rationnelle.

k) Appliquer cet algorithme à l'expression rationnelle $(a + ba)^*b(ba + b)^*$.

2 Fonctions séquentielles

Un filtre à rebonds est une application $f : \{0,1\}^* \to \{0,1\}^*$ qui, dans un mot d'entrée, remplace un 0 par un 1 s'il est encadré par deux 1, et remplace un 1 par un 0 s'il est encadré par deux 0. On supposera que le mot d'entrée est encadré à gauche et à droite par deux 0 fictifs. Ainsi, le mot d'entrée 0101101 est transformé en 0011110 (le dernier 0 de la sortie s'explique par le fait que le dernier 1 de l'entrée est encadré par un 0 à gauche et par le 0 fictif à droite).

a) Donner (dessiner) un automate séquentiel qui réalise le filtre à rebonds f.

b) Normaliser cet automate et dessiner l'automate obtenu.

c) Calculer par raffinements successifs l'équivalence de Nerode associée à l'automate normalisé. En déduire l'automate minimal du filtre à rebonds et dessiner cet automate.

Remarque : Un filtre à rebonds peut être utilisé pour " lisser " une suite de 0 et de 1 par exemple pour réduire les imperfections d'une image.