Automates avancés – Partiel (durée : 1h50)

Tous les documents sont autorisés.

Le barème est donné à titre indicatif et peut être modifié.

Quelques rappels de définitions (déjà toutes vues en TD) : Une *grammaire hors-contexte* est un quadruplet $G = (V, T, P, S)$, où V et T sont deux ensembles finis disjoints (appelés respectivement variables (ou non-terminaux) et terminaux), P est un ensemble de productions de la forme $A \to \alpha$, où $A \in V$ et $\alpha \in (V \cup T)^*$ et S est le symbole de départ. Soit $\Sigma = V \cup T$.

Nous définissons deux relations $\Rightarrow$ et $\Rightarrow_G^*$ entre suites de Σ^* comme suit : Si $A \to \beta$ est une production de P et $\alpha, \gamma \in \Sigma^*$, $\alpha A \gamma \Rightarrow \alpha\beta\gamma$. La suite $\alpha A \gamma$ est un *prédécesseur immédiat* de $\alpha\beta\gamma$ et $\alpha\beta\gamma$ est un *successeur immédiat* de $\alpha A\gamma$. La relation $\Rightarrow^*$ est alors la fermeture réflexive et transitive de $\Rightarrow$. Si $\alpha \Rightarrow^* \beta$, alors α est un prédécesseur de β et β est un successeur de α. Étant donné $L \subseteq \Sigma^*$, nous définissons $pre(L) = \{\alpha \in \Sigma^* \mid \exists \beta \in L \text{ avec } \alpha \Rightarrow \beta\}$ et $post(L) = \{\beta \in \Sigma^* \mid \exists \alpha \in L \text{ avec } \alpha \Rightarrow \beta\}$. $pre^i(L)$ est inductivement défini par $pre^0(L) = L$ et $pre^{i+1}(L) = pre(pre^i(L))$. Enfin, nous définissons $pre^*(L) = \bigcup_{i \geq 0} pre^i(L)$ ce qui est le même que $\{\alpha \in \Sigma^* \mid \exists \beta \in L \text{ avec } \alpha \Rightarrow^* \beta\}$. De même, $post^*(L) = \{\beta \in \Sigma^* \mid \exists \alpha \in L \text{ avec } \alpha \Rightarrow^* \beta\}$. Le langage généré par G est donné par $L(G) = post^*(\{S\}) \cap T^*$.

Un algorithme simple pour calculer $pre^*(L)$ d'un langage régulier donné par un automate M est donné ci-dessous. Il est basé sur la saturation de la relation de transition d'un automate non-déterministe $M = (Q, \Sigma, \delta, q_0, F)$, où $\delta \subseteq Q \times \Sigma \times Q$. Nous définissons la relation de transition étendue $\widehat{\delta} : (Q \times \Sigma^*) \to 2^Q$ par : $\widehat{\delta}(q, \epsilon) = \{q\}$, $\widehat{\delta}(q, a) = \{q' \mid (q, a, q') \in \delta\}$ et $\widehat{\delta}(q, wa) = \{p \mid p \in \widehat{\delta}(r, a) \text{ pour un } r \in \widehat{\delta}(q, w)\}$.

Entrée : $G = (V, T, P, S), M = (Q, \Sigma, \delta, q_0, F)$, **Sortie** : δ_{pre^*}
$rel \leftarrow \delta$
Répéter
  pour $q, q' \in Q, A \to \beta \in P$ **faire**
    si $q' \in \widehat{rel}(q, \beta)$ **alors** $rel \leftarrow rel \cup \{(q, A, q')\}$
jusqu'à ce que rel ne change plus
retourner rel

Exercice 1 [Grammaires hors-contexte et automates à pile, 8 points]
Considérez la grammaire G suivante :

$$\begin{aligned} S &\to AB \mid BB \\ A &\to CC \mid a \\ B &\to BB \mid CA \mid b \\ C &\to BA \mid AA \mid b \end{aligned}$$

- Donnez un automate à pile qui réconnait le langage $L(G)$ généré par G.
- Est-ce que $abaa$ est généré par la grammaire G ? Calculez pour cela $pre^*(\{abaa\})$ de la grammaire.

- On peut se poser la question suivante : Est-ce que bba ou $bbaa$ sont générés par la grammaire G ? Comment répondre à cette question en appliquant **une seule fois** l'algorithme pour pre^* ? Il n'est pas demandé d'appliquer l'algorithme pour pre^*.
- Comment étendre l'algorithme du pre^* pour une grammaire pour récupérer une dérivation d'un mot w (si elle existe) à partir du calcul de $pre^*(\{w\})$? Esquissez un algorithme et illustrez pour le mot $abba$.
- On considère l'automate à pile suivant : $M = (Q, \Sigma, \Gamma, q_0, Z_0, \emptyset, \delta)$ avec $Q = \{q_0, q_1\}$, $\Sigma = \emptyset$, $\Gamma = \{A, B, C, Z_0\}$ et δ est donnée par les quatres règles
  $\langle q_0, Z_0\rangle \overset{\epsilon}{\hookrightarrow} \langle q_1, BZ_0\rangle \quad \langle q_1, B\rangle \overset{\epsilon}{\hookrightarrow} \langle q_1, AB\rangle \quad \langle q_1, A\rangle \overset{\epsilon}{\hookrightarrow} \langle q_1, BA\rangle \quad \langle q_1, B\rangle \overset{\epsilon}{\hookrightarrow} \langle q_0, C\rangle$
  Calculez $post^*(\{(q_0, Z_0)\})$.

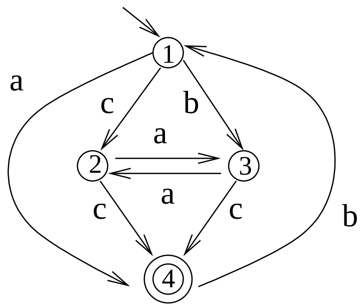
Exercice 2 [Image de Parikh, 7 points]

On considère la formule

$$\phi(x_a, x_b, x_c) \equiv (x_a + x_b = 2x_c)$$

- Donnez un automate fini sur l'alphabet $\{a, b, c\}$ dont l'image de Parikh de son langage est donné par $\phi(x_a, x_b, x_c)$ où x_a, x_b et x_c représentent respectivement le nombre de a, b et c. Indication : un automate qui a comme langage $(abc)^*$ n'est pas suffisant.
- Donnez une grammaire hors contexte dont l'image de Parikh de son langage est donné par $\phi(x_a, x_b, x_c)$ **et qui n'est pas régulier**. Il n'est pas demandé de démontrer que le langage n'est pas régulier.

Considérez l'automate fini suivant :



- Construisez d'abord **directement** à partir de l'automate une formule arithmétique pour caractériser l'image de Parikh du langage de l'automate.
- Pourquoi l'automate n'est pas minimale ?
- Est-ce qu'on peut utiliser la minimisation pour faciliter la construction d'une formule pour l'image de Parikh ? Illustrez.

Exercice 3 [Langages hors-contexte, 6 points]

Rappel du lemme d'itération pour les langages hors-contexte : Si L est un langage hors-contexte (algébrique) alors il existe un entier $N > 0$ tel que pour tout mot w de L avec $|w| \geq N$ il existe une factorisation $w = uvzxy$ où : (1) $|vx| \geq 1$ (2) $|vzx| \leq N$ (3) Pour tout entier $i \geq 0$ on a $uv^izx^iy \in L$.

- Donnez une grammaire hors contexte (ou un automate à pile) qui génère le langage $\{a^ib^jc^k \mid j = 2 * i + k\}$. Indication : On peut découper chaque mot en deux parties.
- Montrez que le langage $\{a^ib^jc^k \mid j = i * k\}$ n'est pas hors-contexte.