TD langages rationnels et automates

Exercice 1 Expressions régulières

Décrire aussi simplement que possible les langages définis par les expressions régulières suivantes sur l'alphabet $\{a, b\}$:

1. $a(a|b)^*$
2. $a^*|b^*$
3. $(b|ab)^*(a|\varepsilon)$
4. $(aa|b)^*$
5. $(ab^*a|b)^*$

Correction

1. *mots commençant par a*
2. *mots n'ayant que des a ou que des b*
3. *mots n'ayant pas deux a consécutifs*
4. *mots avec des blocs de a de longueur paire*
5. *mots ayant un nombre pair de a*

Exercice 2 Expression régulières

Donner des expressions régulières caractérisant les langages suivants sur l'alphabet $\{a, b\}$:

1. mots de longueur au plus deux
2. mots de longueur paire
3. mots contenant la séquence *ab* mais pas la séquence *ba*
4. mots contenant au plus l'une des deux séquences *ab* ou *ba*
5. mots contenant la séquence *ab* mais pas la séquence *aa*

Correction

1. $(\varepsilon|a|b)(\varepsilon|a|b)$
2. $((a|b)(a|b))^*$
3. a^*abb^*
4. $a^*b^*|b^*a^*$
5. $b^*ab(ab|b)^*$

Exercice 3 Automates

Donner de automates reconnaissant les langages suivants :

1. commentaires de la forme `/* ... */` (sans imbrication)
2. nombres en écriture décimale, positifs ou négatifs, avec éventuelle virgule et pouvant faire intervenir la notation scientifique (comme par exemple : `1.02e-5`)

Exercice 4 Lemme de l'étoile

Démontrer que les langages suivants ne sont pas reconnaissables par automate fini :

1. $\{ a^n b^m \mid n < m \}$
2. ensemble des palindromes sur un alphabet avec au moins deux lettres
3. $\{ baba^2ba^3 ... ba^n \mid n \geq 0 \}$
4. $\{ a^n \mid n$ premier $\}$

Correction

Dans tous les cas, on note N l'entier donné par le lemme de l'étoile, et L le langage étudié.

1. *Appliquer lemme sur le début de $a^N b^{N+1}$.*
2. *Appliquer lemme sur le début de $a^N ba^N$.*
3. *Appliquer lemme sur la fin de $baba^2 ... ba^N$.*
4. *Prendre un n premier $\geq N$. On décompose $a^n = a^{n_1} a^{n_2} a^{n_3}$ avec $n_2 \neq 0$ et $n_1 + n_2 + n_3 = n$ et $\forall k, a^{n_1} a^{kn_2} a^{n_3} \in L$. On a $a^{n_1} a^{kn_2} a^{n_3} = a^{n_1+n_3} a^{kn_2} = a^{n-n_2} a^{kn_2} = a^{n+(k-1)n_2}$. En prenant $k = n + 1$ on obtient $a^{(n+1)n_2} \in L$. Or $(n + 1)n_2$ n'est pas premier.*

Exercice 5 Réductions

Preuves de non-reconnaissabilité par réduction à des langages non reconnaissables déjà connus.

1. Démontrer que si L_1 et L_2 sont deux langages reconnaissables, alors $L_1 \cap L_2$ est encore reconnaissable.
2. On note $|m|_c$ le nombre d'occurrence de la lettre c dans le mot m. On définit un langage $L = \{ m \mid |m|_a = |m|_b \}$ sur l'alphabet $A = \{a, b, c\}$. Trouver un L' tel que $L \cap L' = \{ a^n b^n \mid n \geq 0 \}$. En déduire que L n'est pas reconnaissable.
3. Pour A et B deux alphabets, une fonction $f : A \longrightarrow B^*$ peut être étendue en un morphisme $A^* \longrightarrow B^*$ en posant $f(a_1 a_2 \ldots a_n) = f(a_1) f(a_2) \ldots f(a_n)$. Démontrer que si $L_B \subseteq B^*$ est un langage reconnaissable, alors $L_A = f^{-1}(L_B)$ est un langage reconnaissable sur A.
4. En déduire que les commentaires imbriqués de caml de la forme

   ```
   (* ... (* ... *) ... (* ... (* ... *) ... *) ... *)
   ```

   ne peuvent pas être décrits par une expression rationnelle.

Correction

1. *On démontre d'abord que les langages reconnaissables sont stables par complément, en remplaçant F par $Q \setminus F$ dans un automate déterministe complet (Q, T, I, F) reconnaissant ce langage.*

   En outre, $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$. L'union de deux langages reconnaissables est trivialement reconnaissable (correspondant à $e_1 \mid e_2$ dans les expressions rationnelles), d'où conclusion en combinant les compléments.
2. *Prendre $L' = a^* b^*$. Supposons L reconnaissable. Alors $L \cap L'$ serait reconnaissable. Or $\{ a^n b^n \mid n \geq 0 \}$ n'est pas reconnaissable (cours), d'où contradiction.*
3. *On prend un automate (Q, T, I, F) pour L_B et on en déduit un automate (Q, T', I, F) pour L_A en posant $T' = \{ (p, a, q) \mid p \xrightarrow{f(a)} q \text{ chemin dans } T \}$.*
4. *Prendre l'intersection avec $(\texttt{(*})^*(\texttt{*)})^*$ pour se ramener à $\{ (\texttt{(*})^n(\texttt{*)})^n \mid n \geq 0 \}$, puis le morphisme inverse par $[a \mapsto \texttt{(*}, b \mapsto \texttt{*)}]$ pour se ramener à $\{ a^n b^n \mid n \geq 0 \}$.*

Exercice 6 Résiduels

1. On note L le langage défini par l'expression régulière $a^*b|ba^*$. Donner les résiduels de L après a, après ab, après ba, après aba.
2. Donner l'ensemble des langages résiduels du langage $L = \{ a^n b^n \mid n \geq 0 \}$. Qu'en déduire ?

Correction

1. $a^{-1}L = a^*b$, $(ab)^{-1}L = \varepsilon$, $(ba)^{-1}L = a^*$, $(aba)^{-1}L = \emptyset$.
2. *Résiduels : $\emptyset$ (par exemple pour ba), $\{a^n b^{n+k} | n \geq 0\}$ (pour chaque a^k), $\{b^k | k \geq 0\}$ (pour par exemple $a^{k+1}b$).*

   On a une infinité de langages résiduels différents, ce qui donne une nouvelle preuve du fait que ce langage n'est pas rationnel (le nombre de résiduels étant une borne inférieure pour le nombre d'états d'un automate déterministe complet).

Exercice 7 Construction directe d'un automate déterministe pour une expression régulière

Remarquons que chaque lettre d'un mot m reconnu par une expression régulière e peut être mise en relation avec une des lettres de e. Ainsi par exemple, le mot $aabaab$ est reconnu par $(a|b)^*a(a|b)$ de la façon suivante :

$a \quad (\underline{a}|b)^*a(a|b)$
$a \quad (\underline{a}|b)^*a(a|b)$
$b \quad (a|\underline{b})^*a(a|b)$
$a \quad (\underline{a}|b)^*a(a|b)$
$a \quad (a|b)^*\underline{a}(a|b)$
$b \quad (a|b)^*a(a|\underline{b})$

Nous allons distinguer les différentes lettres d'une expression pour faciliter ce suivi. Notre expression régulière devient donc

$(a_1|b_2)^*a_3(a_4|b_5)$

On veut alors construire un automate dont les états sont des ensembles de lettres (par exemple : $\{a_1, b_2, a_3\}$). L'état q va reconnaitre les mots dont la première lettre appartient à q (donc dans notre exemple, un a correspondant à a_1 ou a_3, ou un b correspondant à b_2).

Rappelons que l'on dispose déjà d'une fonction null(e) qui renvoie vrai si l'expression e reconnaît le mode vide ε.

$$\begin{array}{rcl}
\mathsf{null}(\emptyset) & = & \mathsf{false} \\
\mathsf{null}(\varepsilon) & = & \mathsf{true} \\
\mathsf{null}(a) & = & \mathsf{false} \\
\mathsf{null}(e_1e_2) & = & \mathsf{null}(e_1) \wedge \mathsf{null}(e_2) \\
\mathsf{null}(e_1 \mid e_2) & = & \mathsf{null}(e_1) \vee \mathsf{null}(e_2) \\
\mathsf{null}(e^*) & = & \mathsf{true}
\end{array}$$

1. Définir deux fonctions first et last renvoyant respectivement l'ensemble des premières lettres et des dernières lettres possibles d'un mot reconnu par une expression régulière. On doit donc avoir

$$\begin{array}{rcl}
\mathsf{first}((a_1|b_2)^*a_3(a_4|b_5)) & = & \{a_1, b_2, a_3\} \\
\mathsf{last}((a_1|b_2)^*a_3(a_4|b_5)) & = & \{a_4, b_5\}
\end{array}$$

2. Définir une fonction follow qui prend en entrée une lettre a et une expression e, et qui renvoie l'ensemble des lettres qui peuvent suivrent a dans un mot reconnu par e. On doit donc avoir

$$\mathsf{follow}(a_1, (a_1|b_2)^*a_3(a_4|b_5)) \quad = \quad \{a_1, b_2, a_3\}$$

3. On propose alors de construire un automate déterministe pour une expression régulière e en se basant sur les fonctions précédentes, appliquées à l'expression $e\#$ obtenu en ajoutant une lettre spéciale à la fin de e.
Préciser ce que seront l'état initial, les états acceptants, et les transitions. Appliquer cette construction à notre exemple $(a_1|b_2)^*a_3(a_4|b_5)\#$.
Indication. L'état initial doit être $\{a_1, b_2, a_3\}$, et il en part les deux transitions
$(\{a_1, b_2, a_3\}, a, \{a_1, b_2, a_3, a_4, b_5\})$ et $(\{a_1, b_2, a_3\}, b, \{a_1, b_2, a_3\})$.

Remarque : et ça se code en caml.

Correction

1.

$$\begin{array}{rcl}
\mathit{first}(\emptyset) & = & \emptyset \\
\mathit{first}(\varepsilon) & = & \emptyset \\
\mathit{first}(a) & = & \{a\} \\
\mathit{first}(e_1e_2) & = & \mathit{first}(e_1) \cup \mathit{first}(e_2) \text{ si } \mathit{null}(e_1) \\
\mathit{first}(e_1e_2) & = & \mathit{first}(e_1) \text{ sinon} \\
\mathit{first}(e_1 \mid e_2) & = & \mathit{first}(e_1) \cup \mathit{first}(e_2) \\
\mathit{first}(e^*) & = & \mathit{first}(e)
\end{array}$$

$$\begin{array}{rcl}
\mathit{last}(\emptyset) & = & \emptyset \\
\mathit{last}(\varepsilon) & = & \emptyset \\
\mathit{last}(a) & = & \{a\} \\
\mathit{last}(e_1e_2) & = & \mathit{last}(e_1) \cup \mathit{last}(e_2) \text{ si } \mathit{null}(e_2) \\
\mathit{last}(e_1e_2) & = & \mathit{last}(e_2) \text{ sinon} \\
\mathit{last}(e_1 \mid e_2) & = & \mathit{last}(e_1) \cup \mathit{last}(e_2) \\
\mathit{last}(e^*) & = & \mathit{last}(e)
\end{array}$$

2.

$$\begin{array}{rcl}
\mathit{follow}(c, \emptyset) & = & \emptyset \\
\mathit{follow}(c, \varepsilon) & = & \emptyset \\
\mathit{follow}(c, a) & = & \emptyset \\
\mathit{follow}(c, e_1e_2) & = & \mathit{follow}(c, e_1) \cup \mathit{follow}(c, e_2) \cup \mathit{first}(e_2) \text{ si } c \in \mathit{last}(e_1) \\
\mathit{follow}(c, e_1e_2) & = & \mathit{follow}(c, e_1) \cup \mathit{follow}(c, e_2) \text{ sinon} \\
\mathit{follow}(c, e_1 \mid e_2) & = & \mathit{follow}(c, e_1) \cup \mathit{follow}(c, e_2) \\
\mathit{follow}(c, e^*) & = & \mathit{follow}(c, e) \cup \mathit{first}(e) \text{ si } c \in \mathit{last}(e) \\
\mathit{follow}(c, e^*) & = & \mathit{follow}(c, e) \text{ sinon}
\end{array}$$

3. — *État initial : $first(e\#)$*
   — *Transitions : $s.c = \bigcup_{c_i \in s} follow(c_i, e\#)$*
   — *États acceptants : états contenant #*

Sur l'exemple :

num	*état*
1	$\{a_1, b_2, a_3\}$
2	$\{a_1, b_2, a_3, a_4, b_5\}$
3	$\{a_1, b_2, a_3, a_4, b_5, \#\}$
4	$\{a_1, b_2, a_3, \#\}$

	a	*b*
1	2	1
2	3	4
3	3	4
4	2	1