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 (Expressions 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. $b^*ab(ab|b)^*$

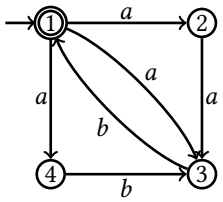
□

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`)

□

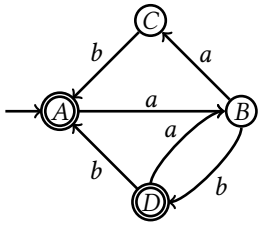
Exercice 4 (Déterminisation) On considère l'automate suivant sur l'alphabet $\{a, b\}$.



1. Déterminiser l'automate.
2. Donner une expression régulière décrivant le langage reconnu par l'automate.

Correction :

1. *À gauche, l'automate déterminisé, à droite la correspondance entre les états de l'automate déterminisé et des ensembles d'états de l'automate de départ.*



A	$\{1\}$
B	$\{2, 3, 4\}$
C	$\{3\}$
D	$\{1, 3\}$

2. *L'automate reconnaît n'importe quelle répétition des motifs ab, aab et abb. D'où l'expression* $(ab|aab|abb)^*$.

□

Exercice 5 (Langages non reconnaissables) On veut montrer que le langage P formé par l'ensemble des palindromes sur l'alphabet $\{0, 1\}$ ne peut pas être décrit par une expression régulière. Pour ceci, on commence le raisonnement suivant : *« Supposons que P peut être décrit par une expression régulière. Alors il est également reconnaissable par un automate fini A. Notons N le nombre d'états de A, et considèrons le mot* $m = 0^{N+1}10^{N+1}$*. Le mot m étant un palindrome, il doit donc être reconnu : il existe dans A un chemin acceptant étiqueté par m. »*

1. Montrer que les N premières transitions du chemin acceptant m comportent au moins un cycle.
2. En déduire l'existence d'un mot qui est accepté par A alors qu'il n'est pas un palindrome.
3. Compléter la preuve du fait que P n'est pas reconnaissable.
4. Avec la même technique, démontrer que le langage B formé des mots dont la longueur est une puissance de 2, n'est pas non plus reconnaissable (*ou en utilisant directement le lemme de l'étoile*).

Correction :

1. *Ces N premières transitions visitent* $N + 1$ *états. Comme il n'y a que N états différents dans l'automate, au moins un état est visité deux fois. La portion de chemin comprise entre les deux premières visites de cet état forme un cycle de longueur* $k \geq 1$.
2. *On obtient encore un chemin acceptant en retirant un passage dans le cycle identifié ci-dessus. Ce chemin raccourci est étiqueté par le mot* $0^{N+1-k}10^{N+1}$*, qui est alors reconnu sans être un palindrome.*
3. *Structure complète : preuve par l'absurde. On suppose que P est reconnaissable, et par les deux points suivants on en déduit une contradiction (la déclaration que* $0^{N+1-k}10^{N+1}$ *appartient à P). Donc P ne pouvait pas être reconnaissable.*
4. *Note : pour cet autre langage, il n'y a pas besoin de choisir une zone particulière du mot où chercher une boucle (i.e. la version faible du lemme de l'étoile suffit).*

□

Exercice 6 (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 :

$$\begin{array}{ll} a & (\underline{a}|b)^*a(a|b) \\ a & (\underline{a}|b)^*a(a|b) \\ b & (a|\underline{b})^*a(a|b) \\ a & (\underline{a}|b)^*a(a|b) \\ a & (a|b)^*\underline{a}(a|b) \\ b & (a|b)^*a(a|\underline{b}) \end{array}$$

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

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

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

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

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.

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

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

2.

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

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

□