Université Côte d'Azur — L3 Info, L3 MI, L3 ST
Automates & Langages — 2023–24

TD n° 1

Langages rationnels

Exercice 1) On se place sur l'alphabet $\Sigma = \{0, 1\}$. Parmi les expressions régulières suivantes, dites lesquelles décrivent le langage Σ^*. Vous donnerez tour à tour soit une explication, soit un contre-exemple :

1. $(0^*1)^* + (1^*0)^*$
2. $\emptyset + \varepsilon + 0(0+1)^* + 1(0+1)^*$
3. $0^* + 0^*(10^*)^+$
4. $(0^*1^*)^*$
5. $0^* + 0^*1(0+1)^*$
6. $0^*(1^+0^+)^*1^*$

Exercice 2) Donnez *si possible* une expression régulière pour décrire les langages suivants :

1. les représentations binaires des entiers sans 0 inutiles en tête ;
2. les identificateurs en PASCAL : suite alphanumérique commençant par une lettre (pas de limitation sur le nombre de caractères) ;
3. les réels en `C` : on suppose que chaque réel a un point (pas une virgule !) et contient au moins un chiffre avant et après le point.
   Exemples : `+1.0`, `12.34e-5`, `-0.7e07`, `12.001E+9`... ;
4. le langage $\{0^n1^n / n \leq 4\}$.

Exercice 3) Donnez une expression régulière dénotant les langages quotients $\{0\}^{-1}L$ et $\{1\}^{-1}L$ pour chacun des langages L décrits par les expressions régulières suivantes :

$E_1 : ((0+1)(0+1))^*$;
$E_2 : 0 + 1(0+1)^*$;
$E_3 : (0+1)^*(00+11)(0+1)^*$;
$E_4 : (0+\varepsilon)(10)^*(1+\varepsilon)$.

Exercice 4) On veut modéliser un digicode à trois touches A, B et C. La porte doit s'ouvrir seulement quand la séquence ABA est saisie. Toute saisie d'un mauvais code (séquence de trois touches distincte de ABA) a pour effet de faire revenir l'automate à son état initial. Trouvez un automate fini $\mathcal{A}$ déterministe pour modéliser ce digicode.

Exercice 5) Considérons l'automate fini $\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$ sur l'alphabet $\Sigma = \{0, 1\}$, avec $Q = \{q_0, q_1, q_2, q_3\}$ et $F = \{q_3\}$. La relation δ se déduit aisément de la figure suivante :

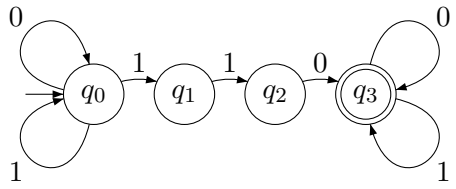

FIGURE 1 – L'automate fini $\mathcal{A}$

1. Donnez une expression régulière décrivant le langage L reconnu par l'automate $\mathcal{A}$.
2. Comment décririez-vous le langage L en français ? Et le langage L^* ?
3. Construisez puis dessinez l'automate $\mathcal{D}$, obtenu en déterminisant l'automate $\mathcal{A}$.
4. Tentez de trouver directement à partir de $\mathcal{A}$ un autre automate déterministe $\mathcal{D}'$, *plus simple* que l'automate $\mathcal{D}$.

Exercice 6) On définit le langage L comme étant l'union des mots binaires se terminant par 0 et des mots de parité paire (avec un nombre pair mais non nul de 1).

1. Donnez une expression régulière pour le langage L ;
2. Trouvez un automate fini pour chacun de ses deux sous-langages ;
3. Assemblez-les en un automate non-déterministe qui accepte les mots de L ;
4. Appliquez lui l'algorithme de déterminisation vu en cours.

Exercice 7) Considérons L un langage quelconque. On note L^+ la fermeture de Kleene d'un langage, privée de la puissance 0 de ce langage :

$$L^+ = \bigcup_{i>0} L^i$$

1. L^* est-il toujours un langage infini ?
2. L^+ est-il toujours un langage infini ?
3. $L^* \setminus L^+ = \{\varepsilon\}$ est-il toujours vrai ?

Exercices complémentaires

Exercice 8) Sur l'alphabet $\{0, 1\}$, quel est le langage dénoté par l'expression régulière E suivante ?

$$E : \ (0+1)^*(00+11)(0+1)^*$$

1. Quel est son complémentaire ? Trouvez une expression régulière qui le dénote.
2. Appliquez à E l'algorithme pour passer directement d'une expression régulière à un automate fini déterministe.

Exercice 9) Voici la définition inductive de l'ordre préfixe (noté $\leq_{préf}$) sur l'alphabet Σ quelconque :

Base : $\varepsilon \leq_{préf} \varepsilon$

Induction : $\forall u, v \in \Sigma^*$,
$\forall \alpha \in \Sigma$,
si $u \leq_{préf} v$ alors $u \leq_{préf} v\alpha$
$\alpha u \leq_{préf} \alpha v$

1. Dessinez le diagramme de Hasse de l'ordre préfixe pour les mots de longueur inférieure ou égale à 3 sur l'alphabet $\Sigma = \{0, 1\}$.
2. Donnez la définition inductive de l'ordre "facteur'.

Exercice 10) FLEX est un générateur automatique d'analyseurs lexicaux. Voici les conventions pour décrire une expression régulière (étendue) en FLEX :

`" "`	texte non interprété
`\`	caractère d'après interprété tel quel
`[ ]`	ensemble de caractères mais opérateurs ignorés sauf : – (intervalle) ^ (complémentaire) \ (échappement)
`?`	élément facultatif
`.`	n'importe quel caractère (sauf fin de ligne)
`*`	0 ou plusieurs fois
`+`	1 ou plusieurs fois
`|`	alternative
`( )`	pour grouper
...	...

Compte tenu de ces définitions, quel est le langage qui se cache sous l'expression régulière (étendue) FLEX suivante ?

```
"(*"[^*]*[*]+([^)*][^*]*[*]+)*")"
```