R. Natowicz

Esiee-Paris - majeure informatique - automates, langages et calculabilité
feuille d'exercices n°1 - 17 novembre 2011

Exercice 1. Les langages L_i suivants sont définis sur l'aphabet $\Sigma = \{0, 1\}$. Donner pour chacun d'eux un automate fini le reconnaissant et vérifier par simulation que chacun de vos automates reconnaît le langage correspondant. Pour cela, décrire votre automate par une base de faits en Prolog et demander, par exemple, la démonstration suivante : `between(0,5,K), length(C,K), accepte(C), write(C), fail.`

1. L_1 : ensemble des chaînes dont le nombre de 0 est multiple de 2 ou le nombre de 1 est multiple de 3 ;
2. $L_2 = \overline{L_1}$;
3. L_3 : ensemble des chaînes dont le nombre de 0 est multiple de 2 et le nombre de 1 est multiple de 3 ;
4. $L_4 = \overline{L_3}$;
5. L_5 : ensemble des chaînes composées d'une suite de 01 en nombre quelconque, suivie d'une suite de 10 en nombre quelconque, autrement dit $L_5 = (01)^*(10)^*$;
6. $L_6 = \overline{L_5}$;
7. L_7 : ensemble des chaînes composées d'une suite de 0110 en nombre quelconque, autrement dit $L_7 = (0110)^*$;
8. $L_8 = \overline{L_7}$;
9. L_9 : ensemble des chaînes ne contenant pas la chaîne 0110 ;
10. L_{10} : ensemble des chaînes commençant et finissant par le même symbole ($L_{10} = \epsilon \ \cup \ 0\Sigma^*0 \cup 1\Sigma^*1$) ;
11. L_{11} : ensemble des chaînes contenant le même nombre d'occurrences de la chaîne 01 et de la chaîne 10.

Exercice 2. Ecrire en Prolog les programmes suivants :

`longueur(L, N)` démontré si la liste `L` est de longueur `N` ; (prédéfini : `length(L, N)`)
`nieme(N, L, X)` démontré si le terme `X` est le N^e elément de la liste `L` ; (`nth1(N, L, X)`)
`entre(Inf, Sup, X)` démontré si `Inf` $\leq$ `X` $\leq$ `Sup` où `Inf`, `Sup` et `X` sont des entiers ; (`between(Inf, Sup, X)`)
`membreStar(X, L)` démontré si le terme `X` est dans la liste `L` quelque soit la profondeur à laquelle il apparaît ;
`longueurStar(L, N)` démontré si `N` est le nombre d'éléments de la liste `L`, quelque soit la profondeur à laquelle ils apparaîssent dans la liste ;
`concatenation(L1, L2, C)` démontré si la liste `C` est la concaténation des listes `L1` et `L2` ; (`append(L1, L2, C)`)
`applatie(L, A)` démontré si la liste `A` est la liste L applatie, c'est-à-dire dans laquelle tous les éléments sont au même niveau et toutes les listes vides sont supprimées (`flatten(L, F)`).
Exemple :

```
?- applatie([1, [], 2, [[3, [4, 5]], 6], [], [7, 8] ], A).
A = [1, 2, 3, 4, 5, 6, 7, 8].
```

Exercice 3. Le célèbre problème des « mutants. » On se donne une base de faits de noms d'animaux :

```
animal('elephant').
animal('antilope').
animal('perdrix').
```

Ecrire un programme `mutant(A1, A2, M)` démontré si `M` est un mutant de `A1` et `A2` au sens illustré ci-dessous :

```
?- mutant(A,B,M), write('A = '), write(A), write('\tB = '), write(B), write('\tM = '), write(M), nl, fail.
A = elephant B = antilope M = elephantilope
A = antilope B = elephant M = antilopelephant
A = antilope B = perdrix M = antiloperdrix
false.
```

Remarque : vous aurez besoin du prédicat prédéfini `string_to_list(S,L)`.
Exemple :

```
?- string_to_list('elephant', L), string_to_list(S, L), write('L = '), write(L),
   write('\t S = '), writeln(S), fail.
L = [101, 108, 101, 112, 104, 97, 110, 116]  S = elephant
false.
```

Exercice 4. On considère les expressions régulières $R_1 = 0^*$ et $R_2 = 00^*1^*$.

1. donner pour chacune d'elle un automate fini déterministe reconnaissant son langage et donner un automate fini déterministe reconnaissant l'union des deux langages ;
2. donner un automate fini non déterministe N à deux états reconnaissant l'union des deux langages ;
3. transformer cet automate N en un automate fini déterministe reconnaissant le même langage.

Exercice 5. On considère un système informatique dans lequel tout nom de fichier est de l'une des deux formes :

`nom simple` : lettre (`a, ..., z, A, ...,Z`) suivie d'une suite de longueur quelconque de lettres ou chiffres (`0, ..., 9`);
`nom composé` : suite non nulle de noms simples séparés par le symbole « `/` ».

Exemples : « `repertoire42` », « `fichier12ter` », « `repertoire42/fichier12ter` ».

1. Donner une expression régulière R caractérisant l'ensemble des noms de fichiers de ce système ;
2. donner un automate fini A reconnaissant le langage $L(A) = R$.

Exercice 6. Pour toute chaîne $\sigma = \sigma_1, ..., \sigma_n$, on note $\sigma^{\mathcal{R}}$ sa « renversée ». Ainsi : $\sigma^{\mathcal{R}} = \sigma_n, ..., \sigma_1$. Par extension, pour tout langage L, on définit $L^{\mathcal{R}} = \{\sigma \ ; \ \sigma^{\mathcal{R}} \in L\}$. Montrer que si le langage L est régulier, alors le langage $L^{\mathcal{R}}$ est régulier.

Exercice 7. Pour tout langage L, on note $\mathrm{pre}(L)$ le langage composé de l'ensemble de ses préfixes : $\mathrm{pre}(L) = \{\pi \in \Sigma^* \ ; \ \exists \sigma \in \Sigma^*, \pi.\sigma \in L\}$. Montrer que si le langage L est régulier, alors le langage $\mathrm{pre}(L)$ est régulier. De même, montrer que si le langage L est régulier, alors le langage $\mathrm{suf}(L)$, composé de l'ensemble de ses suffixes, est un langage régulier.

Exercice 8. Montrer que s'il existe un langage non régulier, alors il en existe une infinité.