IFIPS – Cycle Ingénieur 2^{e} année Informatique
Compilation 2006–2007

Travaux Dirigés n^{o}1
Analyse lexicale

1 Expressions régulières et automates

Donner une expression régulière et un automate fini pour chacun des langages suivants :
- le langage sur l'alphabet $\{a, b\}$ des mots contenant un nombre pair de a ;
- le langage des constantes entières du C, où les nombres commençant par 0 sont des constantes *octales* (base 8) et les autres des constantes décimales ;
- les commentaires (non-imbriqués) de la forme `/*...*/`. Pour ce dernier cas, on utilisera sans le démontrer le résultat suivant : le langage L solution de l'équation $L = \alpha L|\beta$, avec α et β deux mots, est $L = \alpha^* \beta$.

2 Reconnaissance d'une chaîne de caractères par une expression régulière

Dans cet exercice, on se propose d'écrire une fonction de test d'appartenance d'un mot au langage $L(r)$ défini par une expression régulière r. On adopte les notations suivantes pour les expressions régulières :

$\emptyset$	Langage vide
ϵ	Mot vide
c	Le caractère c
$r_1 \| r_2$	Union
$r_1 r_2$	Produit
r^*	Étoile

1. Définir une fonction `contient_epsilon` qui détermine si le mot vide ϵ appartient au langage défini par une expression régulière donnée.
2. On définit le *résidu* d'une expression régulière r par rapport à un caractère c de la manière suivante :
$$Res(r, c) = \{ \ w \mid cw \in L(r) \ \}$$
   (a) Calculer $Res(r, c)$ dans le cas où $r = (a|b)^*a$ et $c \in \{a, b\}$.
   (b) Le langage $Res(r, c)$ peut lui-même être décrit par une expression régulière. Écrire une fonction `residu` calculant à partir de r et de c une expression régulière r' telle que $L(r') = Res(r, c)$.

   (c) Calculer `residu`(r, c) dans le cas où $r = (a|b)^*a$ et $c \in \{a, b\}$.

3. En déduire une fonction `reconnait` qui détermine si une liste de caractères appartient au langage défini par une expression régulière donnée.
4. Appliquer cette fonction à la reconnaissance du mot *aba* par l'expression régulière $r = (a|b) * a$

3 Construction d'automates déterministes à partir d'expressions régulières

On remarque tout d'abord que si un automate fini reconnaît le langage correspondant à l'expression régulière r alors à chaque lettre lue sur l'entrée on peut faire correspondre une occurrence d'une lettre dans l'expression r.
On indexe chaque occurrence d'une lettre dans l'expression régulière par un entier de manière à distinguer les différentes occurrences d'une même lettre.

On prend comme exemple l'expression régulière $(a|b)^*a(a|b)$. Après avoir indicé les caractères, on obtient l'expression : $(a_1|b_1)^*a_2(a_3|b_2)$. Si l'entrée est le mot *aabaab*, alors il correspond à l'expression régulière de la manière suivante :

$$a_1a_1b_1a_1a_2b_2$$

L'idée est de construire un automate dont les états sont des ensembles de lettres indexées correspondant aux occurrences qui peuvent être lues à un instant. Au départ on regarde quelles sont les premières lettres possibles. Dans notre exemple, il s'agit de a_1, b_1, a_2.

Pour construire les transitions, il suffit de pouvoir calculer pour chaque occurrence d'une lettre, les occurrences qui peuvent la suivre. Par exemple si on vient de lire a_1 alors les caractères possibles suivants sont a_1, b_1, a_2, si on vient de lire a_2 alors les caractères possibles suivants sont a_3, b_2.

Si on est dans l'état initial (a_1, b_1, a_2) et qu'on lit un a alors c'est soit a_1 soit a_2 et donc les caractères qui peuvent être lus ensuite sont a_1, b_1, a_2, a_3, b_2.

On va montrer comment calculer systématiquement l'ensemble des caractères suivants.

1. Définir par récurrence sur la structure d'une expression régulière une fonction booléenne *null* qui dit si le langage reconnu contient ϵ.
2. Définir par récurrence sur la structure d'une expression régulière une fonction *début* (resp. *fin*) qui renvoie l'ensemble des premiers (resp. derniers) caractères des mots reconnus.
3. Les caractères suivants c dans une expression régulière r sont caractérisés par la propriété suivante :
   $d \in \mathit{suivant}(c, r)$ ssi il existe r_1, r_2 des expressions régulières telles que r_1r_2 soit une sous-expression de r et $d \in \mathit{début}(r_2)$ et $c \in \mathit{fin}(r_1)$
   ou bien
   il existe r_1 une expression régulière telle que r_1^* soit une sous-expression de r et $d \in \mathit{début}(r_1)$ et $c \in \mathit{fin}(r_1)$

À partir de cette caractérisation définir par récurrence une fonction $suivant(c_i, r)$.

4. Pour construire l'automate déterministe, on procède de la manière suivante sur l'expression régulière indéxée r :
   (a) Ajouter un nouveau caractère $\sharp$ à la fin de l'expression.
   (b) Calculer la fonction *suivant* pour chaque caractère indéxé.
   (c) Construire l'automate dont les états sont des ensembles de caractères indéxés, l'état initial est $début(r)$ les états d'acceptation sont tous ceux qui contiennent $\sharp$. On a une transition de q vers q' à la lecture du caractère c si

$$q' = \bigcup_{c_i \in q} suivant(c_i, r).$$

   Utiliser cet algorithme pour construire des automates déterministes reconnaissant les expressions :
   - $((\epsilon|a)b^*)^*$
   - $(a_1|b_1)^* a_2 (a_3|b_2)(a_4|b_3)$