Maîtrise d'Informatique **12 Décembre 95**
Compilation (tronc commun) **Devoir à la maison**

1 Introduction

Soit r une expression rationnelle, construite à partir des lettres de l'alphabet, de l'opérateur d'union (noté $|$), de l'opérateur de concaténation, de l'opérateur étoile (noté $*$), et de parenthèses ; les opérateurs sont munis des priorités habituelles. A chaque expression r correspond un unique *arbre syntaxique* : chaque noeud interne est étiqueté par un opérateur, et le sous-arbre issu de ce noeud définit une *sous-expression* de r.

On numérote, de 1 à n, les occurrences des *lettres* dans r, on appelle *positions* ces numéros, et on note x_i la lettre en position i ; par exemple l'expresssion $r = (a|ab|ba)^*$ possède 5 positions, et :

$$x_1 = a,\ x_2 = a,\ x_3 = b,\ x_4 = b,\ x_5 = a\ .$$

Soit L le langage dénoté par r ; un mot w dans L correspond à une suite de positions (en général cette suite n'est pas unique) ; par exemple, ici, le mot aba correspond aux suites de positions $1, 4, 5$ ou $2, 3, 1$. N'importe quelle suite de positions n'est pas admissible pour un mot de L : par exemple, ici, la position 2 doit toujours être suivie de 3, et la position 3 ne peut être suivie que par 1, 2 ou 4.

Le but de ce devoir est de construire la fonction *suivant*, qui caractérise, pour chaque position i, la liste des positions qui peuvent suivre i dans un mot de L ; on obtient ainsi un automate non déterministe, dont les états sont les positions, et qui reconnaît L : cet automate comporte une transition $i \to j$, étiquetée x_j, pour chaque position j dans *suivant(i)* (la description de cet automate est volontairement incomplète).

Note : les algorithmes qui suivent sont décrits dans le "dragon", section 3.9 ; le devoir est rédigé de façon à vous éviter la consultation du livre, mais cette dernière n'est évidemment pas interdite ; attention cependant : le "dragon" suppose l'expresssion terminée par un marqueur, ce qui n'est pas le cas ici.

2 Algorithmes

Le calcul de *suivant* passe par le calcul de trois autres fonctions, définies pour toute sous-expression de r (et donc pour tout noeud interne de l'arbre syntaxique) :

- *vide*, qui indique si le langage dénoté par une expression, contient le mot vide ;
- *premier*, qui donne la liste des positions initiales admissibles pour les mots du langage dénoté par une expression ; pour l'exemple introductif, *premier* $(r) = \{1, 2, 4\}$.
- *dernier*, qui donne la liste des positions finales admissibles pour les mots du langage dénoté par une expression ; pour l'exemple introductif, *dernier* $(r) = \{1, 3, 5\}$.

Ces fonctions peuvent être calculées récursivement, à partir de leurs valeurs évidentes sur les lettres de l'alphabet, comme suit (les formules pour *dernier* sont symétriques de celles pour *premier*) :

1. *vide* $(r_1|r_2)$ = *vide* (r_1) OU *vide* (r_2);

   premier $(r_1|r_2)$ = *premier* (r_1) UNION *premier* (r_2);

2. *vide* $(r_1 r_2)$ = *vide* (r_1) ET *vide* (r_2);

   premier $(r_1 r_2)$ = SI *vide* (r_1) ALORS *premier* (r_1) UNION *premier* (r_2) SINON *premier* (r_1) ;

3. *vide* (r^*) = VRAI;

   premier (r^*) = *premier* (r) .

L'algorithme de calcul de *suivant* peut alors se formuler ainsi :

- pour toute sous-expression de la forme $r_1 r_2$, pour toute position i dans *dernier* (r_1), et pour toute position j dans *premier* (r_2), j appartient à *suivant* (i);

- pour toute sous-expression de la forme r_1^*, pour toute position i dans *dernier* (r_1) et pour toute position j dans *premier* (r_1), j appartient à *suivant* (i) .

3 Devoir

1. Construire un exemple d'expression rationnelle, son arbre syntaxique décoré par les attributs *vide*, *premier*, *dernier*, et la table de la fonction *suivant*; l'exemple devra être choisi pour illustrer toutes les règles de construction.

2. Ecrire une grammaire des expressions rationnelles, et un analyseur dans le format reconnu par Yacc, qui calcule les attributs *vide*, *premier*, *dernier*, et la table de la fonction *suivant* .

   L'analyseur lexical sera construit avec Lex, et l'on supposera que l'alphabet sur lequel sont construites les expressions rationnelles, est constitué (en dehors des opérateurs et des parenthèses), des lettres minuscules de l'alphabet ordinaire.

3. Compléter le calcul précédent par celui de l'automate non déterministe qui reconnaît L, basé sur la fonction *suivant* : préciser les états initiaux et terminaux; on sera probablement conduit à ajouter, dans les algorithmes, une position fictive initiale (numérotée 0) et une position fictive finale (numérotée $n+1$).

4. (Facultatif) Déterminiser et minimiser cet automate; étendre la grammaire aux opérateurs + et ?.

4 Recommandations

1. Votre note dépendra essentiellement de la clarté, de la simplicité et de l'efficacité des solutions proposées.

2. L'usage de l'allocation dynamique de mémoire n'est pas obligatoire : ne gaspillez pas votre énergie et votre temps à "chasser les pointeurs fous" !

3. L'analyseur doit, autant que possible, effectuer tous les calculs en une seule passe, en utilisant le mécanisme des attributs synthétisés ; toute dérogation injustifiée à cette règle sera sanctionnée.

4. Décrivez clairement, dans votre copie, les structures de données utilisées, et présentez un jeu d'essais commenté.

5. L'analyseur devra afficher la table de l'automate, sous un format simple mais facilement lisible ; une option doit permettre des affichages intermédiaires plus détaillés.

6. Vous pouvez convenir avec votre enseignant de TD de rendre les fichiers Makefile, Lex, et Yacc par courrier électronique. Cela ne vous dispense pas de rédiger, sur une copie, le reste du devoir : présentation d'un exemple, explication des structures de données choisies, etc...