Examen : Traduction des langages

1h45 avec documents

Remarques : Le barème est donné à titre indicatif.

Attention : Les exercices 3 et 4 sont liés.

Exercice 1 : Automates et grammaires formelles (4pt)

L'ensemble des langages rationnels (les langages reconnus par des automates finis / expressions régulières) est un sous-ensemble des langages algébriques (langages engendrés par une grammaire algébrique). Pour chaque langage rationnel, il existe donc une grammaire algébrique qui l'engendre. Le but de l'exercice est de trouver comment construire cette grammaire à partir d'un automate fini déterministe qui reconnaît le langage rationnel.

1. Soit $X_1 = \{a, b\}$ et L_1 le langage $(a + b)^*$ sur X_1^*. Donner un automate fini déterministe reconnaissant L_1 et une grammaire algébrique engendrant L_1.
2. Soit $X_2 = \{a, b, c, d, e, f\}$ et L_2 le langage $ab(c^*) + de(f^*)$ sur X_2^*. Donner un automate fini déterministe reconnaissant L_2 et une grammaire algébrique engendrant L_2.
3. Soit $A = (Q, X, \delta, q_I, F)$ un automate fini déterministe, donner une grammaire algébrique engendrant $L(A)$.

Exercice 2 : Manipulation de grammaire (4pt)

Dans cet exercice, on travaille sur l'alphabet $\Sigma = \{[,], entier, chaine, ?, id, .\}$.
Soit G la grammaire des tuples suivante :

1. $A \rightarrow T\ \$$
2. $T \rightarrow [S]$
3. $S \rightarrow \Lambda$
4. $S \rightarrow V\ S$
5. $V \rightarrow entier$
6. $V \rightarrow chaine$
7. $V \rightarrow T$
8. $V \rightarrow ?\ C$
9. $C \rightarrow id$
10. $C \rightarrow C\ .\ id$

Questions :

1. Calculer les symboles directeurs de toutes les règles de production.
2. La grammaire est-elle LL(1) ?
3. Quel est l'intéret des grammaires LL(1) ?
4. Si la grammaire n'est pas LL(1), en proposer une équivalente qui est LL(1).
5. Donner un arbre de dérivation pour [10 [ ] "$toto$"]$.

Exercice 3 : Couple et type nommé (5pt)

Nous souhaitons ajouter au langage RAT :
— les couples ;
— la possibilité de nommer des types.

1. Quels terminaux et quelles règles de production faut-il ajouter à la grammaire du cours ? On doit, par exemple, pouvoir écrire :

```
progTypeNomme{
  <int,bool> c1 = <0,true>;
  <<int,int>,bool> c2 = <<0,1>,true>;

  typedef <int,int> Coordonnees;
  Coordonnees p1 = <1,3>;
}
```

2. Quelles informations doivent être associées à un identifiant de type dans la TDS ?
3. Donner les règles de typage à ajouter au système de type du langage RAT. Au besoin, vous pouvez modifier la forme des jugements de typage.
4. Le système proposé est-il un typage par nom ou par structure ?

```
progTypeNomme2{
  typedef <int,int> Coordonnees;
  Coordonnees p1 = <1,3>;
  <int,int> p2 = <2,3>;
  p1 = p2;
}
```

Dans le cas d'un typage par nom, le programme précédent lèvera une exception de typage, alors que dans le cas d'un typage par structure il sera correctement typé.

5. Donner les modifications des règles de typage nécessaires pour réaliser l'autre forme de typage.

Exercice 4 : Enregistrement (7pt)

Nous souhaitons maintenant compléter notre langage avec les enregistrements.
La définition puis l'accès aux éléments d'un enregistrement se fait de la façon suivante :

```
progEnregistrement{
  typedef struct {int x; int y} Point;
  typedef struct {Point ext1; Point ext2} Segment;

  Segment s = {{0,0},{1,1}};
  Point p1 = s.ext1;
  s.ext1.y = 2;
}
```

1. Quels terminaux et quelles règles de production faut-il ajouter à la grammaire du cours ?
2. Donner le placement mémoire de chaque variable.
3. Donner le code TAM du programme `progEnregistrement`.
4. Donner la structure de l'arbre abstrait et les actions sémantiques (en pseudo code ou en code Java correct) à réaliser pour traiter les enregistrements. On doit traiter :
   (a) la table des symboles ;
   (b) le typage ;
   (c) le placement mémoire ;
   (d) la génération de code.