Université Paris Diderot
L2 Info et Math-Info

Analyse de Données Structurées
Année 2015-2016

Examen

Lundi, 23 mai 2016

Tout document papier est autorisé. Les ordinateurs, les téléphones portables, comme tout autre moyen de communication vers l'extérieur, doivent être éteints et rangés.

Le temps à disposition est de 2 heures. Cet énoncé a 2 pages.

Exercice 1 On suppose donnée la définition suivante du type Token :

```
public enum Token {TA, TB, TC;}
```

Questions :

1. On considère un fichier jflex qui contient les règles lexicales suivantes :

   ```
   ([a-z][a-z])+  {return new Token(TA);}
   [a-e]+         {return new Token(TB);}
   [a-z]+         {return new Token(TC);}
   [ ]            {}
   ```

   Donner la séquence de jetons produite par le flot d'entrée suivant :

   ```
   abcdef xyz cde bcde
   ```

2. Dans le fichier jflex donné au-dessus, est-ce que la quatrième ligne est utile ? Il est obligatore de justifier votre réponse (et pas seulement de répondre "oui" ou "non"). Quelques lignes doivent être suffisantes pour cette justification.

3. On cherche maintenant à modifier le fichier jflex de la question (1), afin que toutes les lettres majuscules (entre A et Z) soient traitées exactement de la même façon que les lettres minuscules correspondantes. Par exemple, le fichier jflex modifié doit produire pour le flot d'entrée

   ```
   aBcdEF xYz CdE bcDe
   ```

   exactement la même séquence de jetons que pour le flot d'entrée de la question (1). Donner le fichier jflex modifié.

4. On considère maintenant le fichier jflex suivant :

   ```
   [a-z]+         {return new Token(TC);}
   [a-e]+         {return new Token(TB);}
   ([a-z][a-z])+  {return new Token(TA);}
   [ ]            {}
   ```

   Est ce que ce fichier jflex est interchangeable avec le fichier jflex de la question (1), dans le sens que les deux fichiers, quand lancés sur le même flot d'entrée, produisent toujours le même flot de jetons ? Il est obligatoire de justifier votre réponse.

Exercice 2 On considère la grammaire $G = (V, \Sigma, S, P)$ qui représente des sequences d'expressions arithmétiques sur les nombres réels (avec point décimal), où

— $V = \{(,), ;, ., 0, +, -, \mathsf{succ}, \$, \mathsf{list}, \mathsf{number}, \mathsf{int}, \mathsf{constInt}, \mathsf{start}\}$
— $\Sigma = \{(,), ;, ., 0, +, -, \mathsf{succ}, \$\}$
— $S = \mathsf{start}$
— P consiste en les règles suivantes :

$$
\begin{array}{rcl}
\mathsf{start} & \to & \mathsf{list}\ \$ \\
\mathsf{list} & \to & \epsilon \mid \mathsf{number}\,;\mathsf{list} \\
\mathsf{number} & \to & \mathsf{int}\,.\,\mathsf{int} \mid \mathsf{number} + \mathsf{number} \mid \mathsf{number} - \mathsf{number} \\
\mathsf{int} & \to & \epsilon \mid \mathsf{constInt} \\
\mathsf{constInt} & \to & 0 \mid \mathsf{succ}\,(\mathsf{constInt})
\end{array}
$$

Questions :

1. Dessinez un arbre de dérivation pour la chaîne suivante :

```
succ(0).succ(succ(0)); (succ(0). + succ(0).); $
```

2. Quels sont les symboles effaçables (ensemble appelé en cours EPS) ?
3. Calculez $FIRST_1$, en suivant la méthode donnée en cours.
4. Quelles sont les valeurs de $FOLLOW_1(\mathsf{list})$ et de $FOLLOW_1(\mathsf{int})$? Il suffit de donner les deux ensembles.
5. Expliquez pourquoi la grammaire G n'est pas LL(1).
6. (Question Bonus) : Proposez une grammaire équivalente à G qui est LL(1).

Exercice 3 On considère maintenant le langage des polynômes à une seule variable x. On considère que l'on dispose d'un analyseur lexical et syntaxique qui revoit la syntaxe abstraite. L'ensemble P des polynômes en syntaxe abstraite est défini comme suit :

— tout $Int(n)$, où $n \in \mathbb{N}$, est un élément de P
— x est un élément de P
— si $e_1, e_2 \in P$, alors $Sum(e_1, e_2) \in P$
— si $e_1, e_2 \in P$, alors $Product(e_1, e_2) \in P$

Voici un exemple d'entrée correspondant à ce langage, en syntace concrète : $(3 + x) * (x + 2) + 1$

La syntaxe abstraite est représentée en Java par la classe `Expression` définie comme ceci :

```
abstract class Expression { }

// Represent variable "x"
class Var extends Expression { }

// Integer constants
class Int extends Expression {
  private int value;
}

// E + E'
class Sum extends Expression {
  private Expression left, right;
}

// E * E'
class Product extends Expression {
  private Expression left, right;
}
```

Questions :

1. Ajoutez une méthode `valeurZero` qui calcule la valeur du polynôme quand la variable x vaut zéro.
2. Ajoutez une méthode `valeur` avec le prototype suivant :

```
int valeur(int xv);
```

   Cette méthode doit calculer la valeur du polynôme quand la variable x vaut `xv`.
3. Ajoutez une méthode `degre` qui calcule le degré du polynôme fourni. On rappelle que le degré du polynôme est le plus haut degré de la variable. Par exemple $x + 1$ est de degré 1 et $(x + 1) * x$ est de degré 2.
4. (Bonus) Étendez la classe `Var` et la méthode `valeur` pour les polynômes à plusieurs variables.