Examen de compilation

Licence info, S5
UFR IEEA — 2nde session COMPIL -2h- juin 2009

Tous documents autorisés. Sujet : 2 pages

Exercice 1 : Grammaire attribuée

On donne la grammaire suivante G_N d'axiome *notes* et de terminaux `float`, `[`, `]` :

$$notes \rightarrow \texttt{float [} l \texttt{]}$$
$$l \rightarrow \texttt{float} \mid \texttt{float } l$$

Les mots de $L(G_N)$ s'interprétent ainsi :
- le non-terminal l engendre une liste de `float` telle que chaque `float` est interprété comme une note comprise entre 0 et 20, par exemple `12.5` ou `6` ;
- le terminal `float` qui commence les mots engendrés par *notes* représente une valeur d'arrondi valant `0`, `0.25` ou `0.5`.

Par exemple le mot $m =$ `float [ float float ]` tel que $\texttt{float}_1$ vaut[1] `0.5`, $\texttt{float}_2$ vaut `17.2` et $\texttt{float}_3$ vaut `13.6` représente les notes `17.2` et `13.6` à arrondir au demi-point supérieur.

Q 1.1 : Donner un arbre syntaxique pour le mot `float [ float float ]`. □

On veut calculer la moyenne des notes de la liste, chaque note étant arrondie individuellement. Par exemple, dans le cas du mot m, la moyenne est `(17.5 + 14)/2`. Pour ce faire on suppose donnée une fonction binaire *round* qui prend en premier argument une valeur d'arrondi, en second argument une valeur de note, et retourne la note arrondie correspondante. Par exemple `round(0.5, 17.2)` vaut `17.5` et `round(0.5, 13.6)` vaut `14`.

Q 1. 2 : Attribuer la grammaire pour associer à l'axiome la moyenne des notes arrondies individuellement. Indiquer quels attributs vous avez utilisé, et pour chaque attribut s'il est synthétisé ou hérité, à quel non-terminal il est attaché, s'il est fixé par l'analyseur lexical le cas échéant, ainsi que son type de données. □

Q 1.3 : Quel contrôle sémantique peut-on proposer pour cette grammaire ? □

Exercice 2 : Analyse descendante

Soit la grammaire $G_2 = (S, V_{T2}, V_{N2}, P_2)$ avec $V_{T2} = \{x, y, z\}$, $V_{N2} = \{S, A, B, C\}$ et l'ensemble des productions P_2 :

$$S \rightarrow AC \mid Bz\ ,\ A \rightarrow xA \mid \epsilon\ ,\ B \rightarrow xB \mid z\ ,\ C \rightarrow yC \mid \epsilon$$

Q 2.1 : Calculer les ensembles *Premier* et *Suivant* pour G_2. □

Q 2.2 : Donner la table d'analyse LL(1) de G_2. □

Q 2.3 : G_2 est-elle LL(1) ? Justifier à partir de la table d'analyse. □

On souhaite coder un analyseur syntaxique récursif descendant pour G_2, sans préjuger de son caractère LL(1). On suppose donné un type Java `TypeSymboles` définissant comme des constantes statiques les symboles terminaux de la grammaire `x`, `y`, `z`, plus le marqueur de fin de fichier `EOF`. On s'autorisera à abbrévier `TypeSymboles` en `TS`, on écrira par exemple `TS.x`. On utilisera la méthode `public void consommer(TypeSymboles t) throws ScannerException` et l'attribut `courant` de type `TypeSymboles` vus en cours.

Q 2. 4 : Donner la méthode Java (signature et corps) de l'analyseur récursif descendant qui reconnait le non-terminal A, avec levée d'une exception `ParserException` en cas d'erreur syntaxique et `ScannerException` en cas d'erreur lexicale. □

[1] $\texttt{float}_i$ désigne la ième occurrence de `float` dans le mot.

Exercice 3 : Analyse ascendante

Soit la grammaire $G_1 = (S, V_{T1}, V_{N1}, P_1)$ avec $V_{T1} = \{a, b, c, d\}$, $V_{N1} = \{S, A\}$ et $P_1 =$
$\{\ S \to Aa \mid bAc \mid dc \mid bda\ ,\ A \to d\ \}$

Q 3.1 : Construire l'automate LR-AFD de G_1 (11 états). □

Q 3.2 : Expliquer pourquoi G_1 n'est ni LR(0), ni SLR(1) en explicitant les conflits LR(0) et SLR(1). □