Introduction à la compilation – TD 3 : Menhir

Université Paris Diderot – Licence 3

(2011-2012)

La documentation de `Menhir` se trouve ici :

`http://cristal.inria.fr/~fpottier/menhir/manual.pdf`

Exercice 1 (Expressions booléennes) *Soit l'alphabet* $T = \{$***true***, ***false***, ***id***, (,), $\wedge$, $\vee\}$. *Voici la grammaire des expressions booléennes :*

$$\begin{array}{rcl} e & ::= & \textbf{\textit{true}} \\ & | & \textbf{\textit{false}} \\ & | & \textbf{\textit{id}} \\ & | & e \wedge e \\ & | & e \vee e \\ & | & (e) \end{array}$$

1. *Spécifiez l'ensemble des terminaux à l'aide de la directive* `%token` *dans le fichier* `parser.mly`.
2. *Modifiez le fichier* `lexer.mll` *pour reconnaître l'ensemble de ces terminaux lors de l'analyse lexicale. Pour reconnaître* $\vee$*, on pourra prendre* `\/`*. Pour reconnaître* ***id****, on pourrait utiliser l'expression rationnelle* `['a'-'z']+`*. N'oubliez pas d'ignorer les espaces et les sauts de ligne.*
3. *Modifiez le fichier* `parser.mly` *pour mimer la spécification de grammaire donnée plus haut.*
4. *Observez le fichier* `parser.conflicts`*. Affectez des priorités raisonnables aux règles de production pour résoudre ces conflits. Pour cela, utilisez les directives* `%left`, `%right` *ou* `%nonassoc`.

□

Exercice 2 (Un conflit classique) *On augmente l'alphabet* T *par l'ensemble de terminaux* $\{$***if***, ***then***, ***else***, =$\}$.
On définit la syntaxe des instructions d'un langage très simple :

$$\begin{array}{rcl} i & ::= & \textbf{\textit{if}}\ e\ \textbf{\textit{then}}\ i \\ & | & \textbf{\textit{if}}\ e\ \textbf{\textit{then}}\ i\ \textbf{\textit{else}}\ i \\ & | & \textbf{\textit{id}} = e \\ & | & (i) \end{array}$$

1. *Modifiez le fichier* `parser.mly` *pour y inclure ces nouveaux terminaux.*
2. *Modifiez le fichier* `lexer.mll` *pour les reconnaître lors de l'analyse lexicale.*
3. *Modifiez les fichiers* `ast.ml` *et* `main.ml` *pour qu'ils prennent en compte les nouvelles constructions de la grammaire.*
4. *Rajoutez la définition du non terminal* i *dans la grammaire du fichier* `parser.mly`.
5. *Analysez le conflit détecté par* `Menhir` *en donnant une entrée qui peut être reconnue de deux façons différentes.*
6. *Implémentez deux spécifications de la grammaire correspondant à ces deux variantes.*

□

Exercice 3 (Une grammaire LR(2)) *On étend maintenant la grammaire de la façon suivante :*

$$
\begin{array}{rcl}
i & ::= & \ldots \\
 & | & ma \\
 & & \\
ma & ::= & a \\
 & | & a \wedge ma \\
 & & \\
a & ::= & \textbf{\textit{id}} = e
\end{array}
$$

1. *Modifiez les fichiers pour prendre en compte cette extension telle quelle.*
2. *Observez le conflit généré par* `Menhir`*.*
3. *Modifiez la structure de la grammaire pour qu'elle soit LR(1) et donc adaptée à* `Menhir`*.*
4. *Revenez à la définition de la grammaire que vous aviez en question 2 et utilisez la directive* `%inline` *pour obtenir le même résultat qu'à la question 3.*

□