CS335: COMPILER DESIGN - Assignment 2

Harsh Bihany (210406)

Problem 1

Consider the following context-free grammar. The terminal symbols are **id** ( ) * **,** and \$. $Function$ is the start nonterminal.

$$\begin{aligned}
Function &\rightarrow Type\ \mathbf{id}\ (Arguments)\\
Type &\rightarrow \mathbf{id}\\
Type &\rightarrow Type*\\
Arguments &\rightarrow ArgList\\
Arguments &\rightarrow \epsilon\\
ArgList &\rightarrow Type\ \mathbf{id}\ \mathbf{,} ArgList\\
ArgList &\rightarrow Type\ \mathbf{id}
\end{aligned}$$

(i) Explain why the grammar is not LL(1)?

A grammar is *left recursive* if it has a non-terminal A such that there exists a derivation $A \xrightarrow{*} A\alpha$ for some string α. In the following grammar we have one such derivation $Type \longrightarrow Type*$.

Further a left recursive grammar cannot be LL(1). Hence the above grammar is not LL(1).

(ii) Perform required transformations on the grammar to make it LL(1). You may introduce new nonterminals. Show your transformed grammar.

To make the grammar LL(1), we need two things. Elimination of direct and indirect left recursion and left factoring to eliminate backtracking if needed.

Removing the direct left recursion mentioned earlier. We get

$$\begin{aligned}
Function &\rightarrow Type\ \mathbf{id}\ (Arguments)\\
Type &\rightarrow \mathbf{id}\ Type^{'}\\
Type^{'} &\rightarrow *Type^{'}\\
Type^{'} &\rightarrow \epsilon\\
Arguments &\rightarrow ArgList\\
Arguments &\rightarrow \epsilon\\
ArgList &\rightarrow Type\ \mathbf{id}\ \mathbf{,} ArgList\\
ArgList &\rightarrow Type\ \mathbf{id}
\end{aligned}$$

Now, after left factoring (required for the nonterminal $ArgList$) we get,

$$\begin{aligned}
\mathit{Function} &\to \mathit{Type}\ \mathbf{id}\ (\mathit{Arguments}) && (1)\\
\mathit{Type} &\to \mathbf{id}\ \mathit{Type}' && (2)\\
\mathit{Type}' &\to *\mathit{Type}' && (3)\\
\mathit{Type}' &\to \epsilon && (4)\\
\mathit{Arguments} &\to \mathit{ArgList} && (5)\\
\mathit{Arguments} &\to \epsilon && (6)\\
\mathit{ArgList} &\to \mathit{Type}\ \mathbf{id}\ \mathit{More} && (7)\\
\mathit{More} &\to \mathbf{,}\mathit{ArgList} && (8)\\
\mathit{More} &\to \epsilon && (9)
\end{aligned}$$

Remark. The productions are labeled so that they can be referred subsequently.

(iii) Show FIRST and FOLLOW set for the nonterminals in your transformed grammar.

$$\begin{aligned}
\text{FIRST}(\mathbf{T}) &= \{\mathbf{T}\} \quad \forall \mathbf{T} \in \{\mathbf{id}, (,), *, \mathbf{,}, \$, \epsilon\}\\
\text{FIRST}(\mathit{Function}) &= \{\mathbf{id}\}\\
\text{FIRST}(\mathit{Type}) &= \{\mathbf{id}\}\\
\text{FIRST}(\mathit{Type}') &= \{*, \epsilon\}\\
\text{FIRST}(\mathit{Arguments}) &= \{\mathbf{id}, \epsilon\}\\
\text{FIRST}(\mathit{ArgList}) &= \{\mathbf{id}\}\\
\text{FIRST}(\mathit{More}) &= \{\mathbf{,}, \epsilon\}
\end{aligned}$$

$$\begin{aligned}
\text{FOLLOW}(\mathit{Function}) &= \{\$\}\\
\text{FOLLOW}(\mathit{Type}) &= \{\mathbf{id}\}\\
\text{FOLLOW}(\mathit{Type}') &= \{\mathbf{id}\}\\
\text{FOLLOW}(\mathit{Arguments}) &= \{)\}\\
\text{FOLLOW}(\mathit{ArgList}) &= \{)\}\\
\text{FOLLOW}(\mathit{More}) &= \{)\}
\end{aligned}$$

(iv) Show the predictive LL(1) parsing table for your transformed grammar. You do not need to show the working of the parser with an example input string.

NON-TERMINALS	**INPUTS**					
	\$	(	)	**id**	*	**,**
Function				(1)		
Type				(2)		
Type′				(4)	(3)	
Arguments			(6)	(5)		
ArgList				(7)		
More			(9)			(8)

Table 1.1: Predictive LL(1) Parsing Table

Problem 2