Free University of Bozen-Bolzano

Faculty of Computer Science

Bachelor in Computer Science

Prof. Alessandro Artale

Formal Languages and Compilers – A.Y. 2018/2019 – 13.September.2019

Compiler Part

Time: 1^h45 minutes

This is a closed book exam. The use of Pencils is not allowed! Write clearly, in the sense of logic, language and legibility. The clarity of your explanations affects your grade. Write your name and ID on every solution sheet.

1 Exercise: Top-Down Parsing [10 Points]

Given the following grammar with terminals $VT = \{\textbf{type}, \textbf{ID}, :, \textbf{INT}, \textbf{REAL}, (,)\}$ and `TL` is the scope of the language:

$$\begin{array}{rcl} \texttt{Prog} & \rightarrow & \texttt{TL} \\ \texttt{TL} & \rightarrow & \texttt{T TL} \mid \epsilon \\ \texttt{T} & \rightarrow & \textbf{type ID} : \texttt{D} \\ \texttt{D} & \rightarrow & \textbf{REAL} \mid \textbf{INT} \mid (\texttt{DL}) \\ \texttt{DL} & \rightarrow & \texttt{D DL} \mid \epsilon \end{array}$$

1. Show the value of the function `FIRST` for all the non terminal symbols. [1 Point]
2. Show the value of the function `FOLLOW` for all the non terminal symbols. [1 Point]
3. Show the parsing table for the LL(1) Top Down Parser recognizing the grammar. [3 Points]
4. Show the stack and the moves of the LL(1) parser on input:
   "**type x** : (**REAL INT**)". [3 Points]
5. To use a predictive parser (e.g., LL(1)) it is `necessary` that

   $$\textsc{first}(\alpha) \cap \textsc{first}(\beta) = \emptyset,$$

   for all α, β right side of *alternative* productions—i.e., productions associated to the same non-terminal. Discuss why this condition is necessary. [2 Points]

2 Exercise: Bottom-Up Parsing [15 Points]

1. Consider the following grammar with terminals $VT = \{-, \texttt{num}\}$:

   $$\texttt{E} \rightarrow \texttt{E} - \texttt{E} \mid \texttt{num}$$

   Explain why the Grammar is not SLR by showing a state with a conflict. In particular, describe: [4 Points]

   - The kind of conflict;
   - The default decision taken by `YACC` when dealing with such a conflict;
   - A technique to eliminate the conflict.

Consider now the following grammar with terminals $VT = \{\textbf{num}, \textbf{id}, (,), ,\}$:

$$\begin{array}{rcl} \texttt{S} & \rightarrow & \texttt{A} \mid \texttt{L} \\ \texttt{A} & \rightarrow & \textbf{num} \mid \textbf{id} \\ \texttt{L} & \rightarrow & (\ \texttt{LS}\ ) \\ \texttt{LS} & \rightarrow & \texttt{LS}\ ,\ \texttt{S} \mid \texttt{S} \end{array}$$

Show the following:

2. The canonical SLR collection. [4 Points]
3. The transition diagram describing the automaton which recognizes handles at the top of the stack. [2 Points]
4. The parsing table for the SLR parser [2 Points].
5. The stack and the moves of the SLR parser on the input: "$(\texttt{8}, \texttt{x})$". [3 Points]

3 Exercise: Semantic Analysis [7 Points]

1. Complete the following syntax directed definition:

Production	Semantic Rules
`Decl` → `T` : `VL`	**?**
`T` → **int**	`T`.type = 'int'
`T` → **real**	`T`.type = **?**
`T` → **vect** [**num**] **of** $\texttt{T}_1$	`T`.type = vect(num.val, **?**)
`VL` → $\texttt{VL}_1$, **id**	$\texttt{VL}_1$.type = **?**
`VL` → **id**	*addtype*(id.ptr, **?**)

Where *addtype* is a function that adds to the symbol table entry id.ptr its type. [2 Points]

Given the following syntax directed definition:

Production	Semantic Rules
$Prog \rightarrow S$	$S.next := newlabel; Prog.code := S.code \parallel gen(S.next\ ':')$
$S \rightarrow S_1\ ;\ S_2$	$S_1.next := newlabel; S_2.next := S.next;$
	$S.code := S_1.code \parallel gen(S_1.next\ ':') \parallel S_2.code$
...	...

Where:

- The function *newlabel* generates new symbolic labels: $l_1, l_2, \ldots$
- The function *gen* generates strings such that everything in quotes is generated literally while the rest is evaluated.
- The symbol $\parallel$ means string concatenation.

Show the following:

2. Given the production: $S \rightarrow S_1\ ;\ S_2$, indicate what are the **Inherited** and the **Synthesized** attributes and show the corresponding **Translation Scheme**. [2 Points]

3. Give the definition of an **L-Attributed Syntax-Directed Definition** and discuss on how an L-Attributed Syntax-Directed Definition can be transformed into a **Translation Scheme**. [3 Points]