

CSCI565 – Compiler Design

Spring 2010

Homework 2

Due Date: February 10, 2010 in class

Problem 1 [10 points]: Predictive Top-Down Parsing

Explain why a left-recursive grammar cannot be parsed using the predictive top-down parsing algorithms.

Problem 2 [30 points]: Table-based LL(1) Predictive Top-Down Parsing

Consider the following CFG $G = (N = \{S, A, B, C, D\}, T = \{a, b, c, d\}, P, S)$ where the set of productions P is given below:

$S \rightarrow A$
 $A \rightarrow BC \mid DBC$
 $B \rightarrow Bb \mid \epsilon$
 $C \rightarrow c \mid \epsilon$
 $D \rightarrow a \mid d$

- Is this grammar suitable to be parsed using the recursive descent parsing method? Justify and modify the grammar if needed.
- Compute the FIRST and FOLLOW set of non-terminal symbols of the grammar resulting from your answer in a)
- Construct the corresponding parsing table using the predictive parsing LL method.
- Show the stack contents, the input and the rules used during parsing for the input $w = dbb$

Problem 3 [20 points]: LL parsing using Mutually Recursive Functions

Consider the following context-free grammar. (It corresponds roughly to the syntax of lists in the programming language LISP.) S is the start symbol, and the terminals are $a, (,)$.

$S \rightarrow ()$
 $S \rightarrow a$
 $S \rightarrow (A)$
 $A \rightarrow S$
 $A \rightarrow A , S$

- Show precisely why this grammar is not LL(1). (Hint: This will require computing some, but not all, of the FIRST and FOLLOW sets.)
 - Rewrite this grammar to make it suitable for recursive descent parsing.
 - On the basis of your revised grammar from part (b), write a recursive procedure S that parses this grammar by recursive descent. Your procedure may be written in C, C++, Java or pseudo-code. You may assume the existence of a global variable $token$ that holds the next input token, a function $advance()$ that reads the next token into $token$, and a function $error$ that may be called if the input is not in the language generated by the grammar.
-

Problem 4 [40 points]: LR Parsing Algorithm

Given the grammar below already augmented with the EOF production (0) answer the following questions:

- (0) $S \rightarrow \text{Stmts } \$$
- (1) $\text{Stmts} \rightarrow \text{Stmt}$
- (2) $\text{Stmts} \rightarrow \text{Stmts } ; \text{ Stmt}$
- (3) $\text{Stmt} \rightarrow \text{Var} = E$
- (4) $\text{Var} \rightarrow \text{id } [E]$
- (5) $\text{Var} \rightarrow \text{id}$
- (6) $E \rightarrow \text{id}$
- (7) $E \rightarrow (E)$

- a) Construct the set of LR(0) items and the DFA capable of recognizing it.
- b) Construct the LR(0) parsing table and determine if this grammar is LR(0). Justify.
- c) Is the SLR(0) DFA for this grammar the same as the LR(0) DFA? Why?
- d) Is this grammar SLR(0)? Justify by constructing its table.
- e) Construct the set of LR(1) items and the DFA capable of recognizing it.
- f) Construct the LR(1) parsing table and determine if this grammar is LR(1). Justify.
- g) How would you derive the LALR(1) parsing table from the table found in d) above?