Compiler - HW2

April 2023

1 Context-Free Grammars

Give context-free grammars that generate the following languages.

a) $\{w \in \{0,1\}^* \mid w \text{ contains at least three 1s}\}$

b) $\{w \in \{0,1\}^* \mid w \text{ is palindrom and } |w| \text{ is even}\}$

c) $\{w \in \{0,1\}^* \mid |w| \text{ is odd and the middle symbol of } w \text{ is } 0\}$

d) $\{a^i b^j c^k \mid i,j,k \geq 0, i = j \text{ or } i = k\}$

e) $\{a^i b^j c^k \mid i,j,k \geq 0, i + j = k\}$

f) $\{a^i b^j c^k \mid i,j,k \geq 0, i + k = j\}$

g) $\{a^i b^j c^k \mid i,j,k \geq 0, i + j > k\}$

2 Ambiguity

Consider the following Grammar:

$$\begin{aligned} S &\to AB \mid C \\ A &\to \boldsymbol{a}A\boldsymbol{b} \mid \boldsymbol{ab} \\ B &\to \boldsymbol{c}B\boldsymbol{d} \mid \boldsymbol{cd} \\ C &\to \boldsymbol{a}C\boldsymbol{d} \mid \boldsymbol{a}D\boldsymbol{d} \\ D &\to \boldsymbol{b}D\boldsymbol{c} \mid \boldsymbol{bc} \end{aligned}$$

Is the grammar as given ambiguous? If yes, give an example of an expression with two parse trees under this grammar. If not, explain why that is the case.

3 Predictive Recursive Descent

Consider the following CFG, where the set of terminals is $\{a, b, c, d, \#, ?\}$:

$$\begin{aligned} S &\to \#UT \mid T? \\ T &\to \boldsymbol{a}S \mid \boldsymbol{b}U\boldsymbol{c} \mid \epsilon \\ U &\to \boldsymbol{a}S\boldsymbol{c} \mid \boldsymbol{b}T\boldsymbol{d} \end{aligned}$$

a) Can this grammar be parsed by a predictive recursive descent parser? Write your reason why.

b) Write the appropriate sub-routine for each of the non-terminals in a recursive descent parser. (Match function isn't required)

4 Transition Diagram

Draw transition diagrams for the following CFG. Then, simplify diagrams as much as possible.

$$S' \rightarrow S \tag{1}$$
$$S \rightarrow E \tag{2}$$
$$E \rightarrow E\ \boldsymbol{or}\ T \tag{3}$$
$$E \rightarrow T \tag{4}$$
$$T \rightarrow T\ \boldsymbol{and}\ F \tag{5}$$
$$T \rightarrow F \tag{6}$$
$$F \rightarrow \boldsymbol{not}\ F \tag{7}$$
$$F \rightarrow (E) \tag{8}$$
$$F \rightarrow \boldsymbol{id} \tag{9}$$

5 LL(1) Parser

Consider the following CFG (the start symbol is P):

$\gamma_1 : E \rightarrow \boldsymbol{x}$
$\gamma_2 : E \rightarrow \boldsymbol{\lambda x} : E$
$\gamma_3 : E \rightarrow EE$
$\gamma_4 : E \rightarrow \oplus E$
$\gamma_5 : E \rightarrow E \oplus E$
$\gamma_6 : E \rightarrow (E)$
$\gamma_7 : A \rightarrow \boldsymbol{x} = E$
$\gamma_8 : S \rightarrow \epsilon$
$\gamma_9 : S \rightarrow$ let A
$\gamma_{10} : S \rightarrow$ let A where A
$\gamma_{11} : S \rightarrow \$$
$\gamma_{12} : S \rightarrow SP$

a) Give left-most and right-most derivation for the following string:
let $x = \oplus\, x \oplus x$ where $x = x \oplus x \oplus x$

b) Compute the First and Follow sets for this Grammar.

c) Explain First-First conflicts, and how you would compute them using your answer to part (b). List all First-First conflicts you find in above Grammar.

d) Explain First-Follow conflicts, and how you would compute them using your answer to part (b). List all First-Follow conflicts you find in above Grammar.

e) Demonstrate the correct use of Left-Factoring and Right-Recursion to derive an equivalent grammar that resolves all conflicts you found in parts (c) and (d). Briefly explain why your transformations do not change the language.

6 LR Parser

In this section you should use the previous part's Grammar for answering the following questions.

a) Explain shift-reduce and reduce-reduce conflicts (i.e. how they manifest themselves in an LR(1) parser).

b) An LR(0) parser is a special case of LR parsers characterized by a zero token look-ahead. Reduce actions occur purely based on the tokens that have already been pushed on to the stack. List all the conflicts that an LR(0) parser for the above Grammar would encounter.

c) An SLR parser is an LR(0) parser with one addition: we perform a reduce action only if the next token is in the Follow set of the nonterminal that the reduction would derive. List all the conflicts that an SLR parser for Λ would encounter. Also explain what you would do to get the right-most possible derivations for the above Grammar using an SLR parser if you had the ability to manually pick a state transition where there is a conflict.