INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Mid-Autumn Semester 2023-24

Date of Examination: 18-09-2023 Session: AN Duration: 2 hrs Full Marks: 60
Subject No.: CS31003 Subject: Compilers
Department/Center/School: Department of Computer Science and Engineering
Specific charts, graph paper, log book etc., required : None
Special Instructions (if any) : *(1) Answer all the questions. (2) In case of reasonable doubt, make practical assumptions and write that on your answer script. (3) The parts of each question must answered be together.*

1.

(a) Write L-attributed SDD for the following production, which represents a familiar flow-of-control construct, as in the programming language C. You may need to generate a particular label for the jump statement, in which case you should generate goto label.

$$S \rightarrow \textbf{do } S_1 \textbf{ while } (C)$$

Here, S is the nonterminal that generates all kinds of statements, and C stands for a conditional expression, a boolean expression that evaluates to true or false.

(b) Convert your above SDD to an SDT (Syntax Directed Translation).

Marks: 6+4=10

2.

(a) Consider a grammar G, which generates binary numbers with a "decimal" point. Prof X designed an L-attributed SDD for G, with synthesized attribute ***val***, defined for the non-terminals S, L (and L_1, L_2), B, and inherited attribute ***bit***, defined for the non-terminal L. The inherited attribute ***bit*** counts the number of bits presents in the binary string (before and after the decimal point, separately). The objective of this SDD is to compute **S.val**, which represents the decimal-number value of an input binary string. For example, the computation of string 101.101 should be the decimal number 5.625.

(i) Find the grammar G and the SDD proposed by Prof. X in the table below. Unfortunately, Prof. X missed out some of the semantic rules of the SDD. Complete the missing semantic rules, such that **S.val** can compute the decimal number value of the input binary string. **[Hint: $0.101=5/2^3$]**

PRODUCTION	SEMANTIC RULES
$S \rightarrow L_1 . L_2$	S.val =_____
$S \rightarrow L$	S.val =_____
$L \rightarrow L_1 B$	L.val= _____ L.bit=L.bit+1
$L \rightarrow B$	L.val=B,val L.bit=1
$B \rightarrow 0$	_____
$B \rightarrow 1$	_____

(ii) Now, given the input string 11.10, construct the dependency graph and annotate the parse tree, using the complete SDD.

(b) Consider the grammar below, which generates arithmetic expressions involving operator + and integer or floating-point operands. Floating-point numbers are distinguished by having a decimal point.

$$E \rightarrow E + T \mid T$$
$$T \rightarrow \textbf{num} . \textbf{num} \mid \textbf{num}$$

Focus on the data type of the arithmetic expression, generated by E. The data type of an arithmetic expression, involving addition (+ operator) of two integer operands becomes an integer expression, whereas data type of an arithmetic expression involving one integer and one floating point operand becomes a floating point expression. For instance, the data type of expression 7+4 is integer, whereas 8+3.4, 5.6+3, 7.5+2.9 are floating point expressions.

Consider a SDD, which determines the data type of the expression E. In this SDD, ***type*** is the only (synthesized) attribute, defined for the non-terminals E and T, where **E.type** stores the data type of the expression E. Write the complete SDD which computes **E.type** to determine the data type of the expression E.

Marks: (4+2)+4=10

3.

(a)

$$A \rightarrow BC\mathbf{d} \mid EC\mathbf{f}$$
$$B \rightarrow \mathbf{xy}$$
$$E \rightarrow \mathbf{xy}$$
$$C \rightarrow C\mathbf{c} \mid \mathbf{c}$$

Modify the abovementioned grammar G to an equivalent grammar G', such that minimum number of look-ahead symbol(s) are required to parse the G' using LR(k) parser ($k \geq 0$). Determine the value of your k. Justify your answer. No need to draw the parsing table.

(b) For the following grammar G, draw the LALR(1) parsing table (the states and the transitions must be clearly shown for drawing the parsing table). Mark on the table, if there is any conflict. Also mention the kind of conflict.

$$S \rightarrow \mathbf{a}A\mathbf{d} \mid \mathbf{b}B\mathbf{d} \mid \mathbf{a}B\mathbf{e} \mid \mathbf{b}A\mathbf{e}$$
$$A \rightarrow \mathbf{c}$$
$$B \rightarrow \mathbf{c}$$

Marks: 5+15=20

4.

(a) Consider the grammar G specified below (S, A are non-terminals; 0, 1 are terminal symbols). Apply the left recursion elimination algorithm to transform the grammar G into an equivalent non-left-recursive grammar. Write the final grammar after the elimination process.

$$S \rightarrow A\mathbf{1} \mid \mathbf{0} \mid \mathbf{1}$$
$$A \rightarrow A\mathbf{0} \mid S\mathbf{1} \mid \mathbf{1}$$

(b) Consider the grammar G, which is used to develop a predictive parser.

$$S \rightarrow A$$
$$A \rightarrow B\mathbf{b} \mid C\mathbf{d}$$
$$B \rightarrow \mathbf{a}B \mid \varepsilon$$
$$C \rightarrow \mathbf{c}C \mid \varepsilon$$

(i) Determine whether the grammar G is LL(1), by constructing the LL(1) parsing table for the grammar.

(ii) If possible, apply the predictive parser to parse the input string **aaabcc** using the LL(1) parsing table. Show the configuration of the parser after each step of parsing. Finally, determine if the string is accepted or rejected by the predictive parser.

Marks: 3+(4+3)=10

5.

Consider the grammar G

$$E \rightarrow T+E$$
$$E \rightarrow T$$
$$T \rightarrow \mathbf{id}$$

Justify the following statements by constructing the parsing tables

(a) G is an SLR grammar

(b) G is an LR(1) grammar

Marks: 5+5=10