INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Mid-Autumn Semester 2022-23

Date of Examination : 21-09-2022 Session : FN Duration : 2 hrs Full Marks : 60
Subject No. : CS31003 Subject : Compilers
Department/Center/School : Department of Computer Science and Engineering
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) Consider the following grammar with terminal alphabet {a, (,), +,*} and start symbol E. The production rules of the grammar are:

E → aA | (E)

A → +E | *E | ε

(i) Compute the FIRST and FOLLOW sets for the non-terminals E and A.

(ii) Complete the LL(1) parse table for the grammar.

(b) Let us assume that Prof Shanku proposed a new grammar LL_X(1) for which two of the production rules are A→ α | β, where α and β represent string of terminals and non-terminals. Also, LL_X(1) follows the following two rules.

Rule-1: For no terminal symbol **a**, do both α and β derive strings beginning with **a**
Rule-2: At most one of α and β can derive the empty string.
Illustrate with an example, the problem that you might face in designing predictive parsers for the proposed LL_X(1) grammar.

Marks: (6+4)+3=13

---

2. An expression grammar is given as $G_2=(\{S,E,F\},\{+,-,\text{id}\},P_2,E)$ where

P_2:

(1) $S \rightarrow S-E$
(2) $S \rightarrow E$
(3) $E \rightarrow E+F$
(4) $E \rightarrow F$
(5) $F \rightarrow (S)$
(6) $F \rightarrow$ id

(a) Draw the LR parser table for this grammar after augmenting the grammar as per requirement. Make sure that you are not generating redundant states. Use the following notations

Shift operation:	s_i where i is the stack state after shift.
Reduce operation:	r_j for reduction by production rule j.
Accept state:	acc
Error state:	Keep them blank

(b) Using the above table, parse the following input string ('$' is the end of input marker).

id+(id-id)$

Fill up the parsing trace in the format given below.

Line	Stack	Symbols	Input	Action
1	0	$	id+(id-id)$	

(c) Using the trace, write the right-most derivation of

id+(id-id)$

Marks: 12+8+5=25

3. (a) Prove or disprove with argument:
   (i) "The look ahead symbol in LR(1) parser is only useful for the reduction step, but does not play any role for the shift action."
   (ii) "If LALR parser suffers from shift reduce conflict, then the corresponding LR(1) parser should also suffer from shift reduce conflict." Is the statement true for reduce-reduce conflict too?

   (b) Prof Shanku claims that "The look ahead symbol **a** in item {A→α., a} for LR(1) parser is redundant and unnecessary for the reduction action A→ α, since SLR parser already handles this case by ensuring that input symbol **a** should be in Follow(A) for item {A→α.}". Do you agree with Prof. Shanku? Note: In item dot is at the end of the production rule.

   (c) Consider an item set I_i with state i of an LR(0) automaton. (i) Is it possible to have transitions to state i on multiple grammar symbols X & Y? (ii) Is it possible to have multiple item sets I_i and I_j (with state i and j respectively) with transitions on same grammar symbol X?

Marks: (2.5×2)+2.5+2.5=10

4. The production and the corresponding semantic rules are given below. Identify which one is synthesized and which one is inherited attribute.

PRODUCTION	SEMANTIC RULES
T → B C	T.a=C.a C.b=B.a
B → int	B.a=integer
B → float	B.a=float
C → [num] C_1	C.a=array(num.val,C_1.a) C_1.b=C.a
C → ε	C_1.a=C.b

(a) Draw an annotated parse tree for the following expression **float [4][3][2]**.
(b) Clearly indicate the synthesized and inherited attribute in each of the internal nodes (if present) of your annotated parse tree.
(c) Show the order of evaluation (as appeared in the dependency graph) of this parse tree for obtaining the expression **float [4][3][2]**

Marks: 4+2+6=12