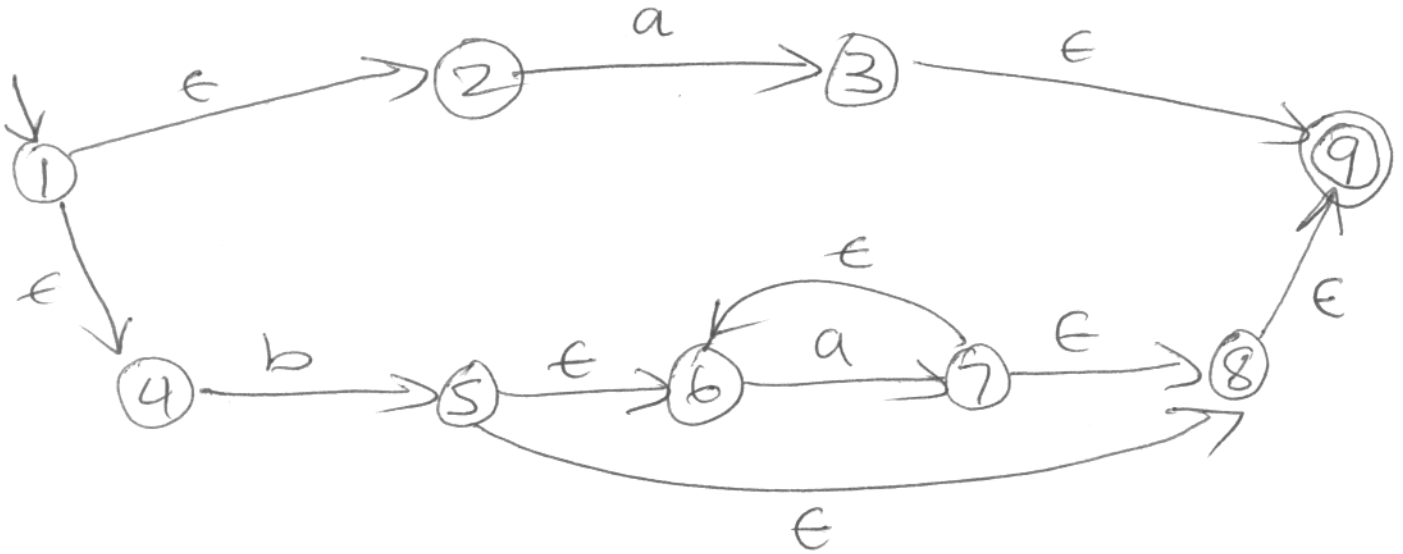


Problem 1 [15 pts.]

Part A. Using Thompson's Construction, build an NFA equivalent to the regular expression given below:

$a \mid ba^*$



Part B. Use subset construction to build a DFA equivalent to the NFA you gave in part A. Show your work.

$\epsilon\text{-cl}(1) = 124$

$m(124, a) = 3$

$\epsilon\text{-cl}(3) = 39$

$m(124, b) = 5$

$\epsilon\text{-cl}(5) = 5, 6, 8, 9$

$m(39, a) = -$

$m(39, b) = -$

$m(5689, a) = 7$

$\epsilon\text{-cl}(7) = 7, 6, 8, 9$
6789

$m(5689, b) = -$

$m(6789, a) = 7$

$m(6789, b) = -$

	a	b
<u>[124]</u>	[39]	[5689]
<u>[39]</u>	-	-
<u>[5689]</u>	[6789]	-
<u>[6789]</u>	[6789]	-

Problem 2 [10 pts.]

Is it possible to write a grammar that generates the language described by the regular expression below? If so, write one, if not, explain why not. Note: the alphabet is $\{a, b\}$

$$(ab^+ab^+)^*$$

$$\begin{array}{l} S \rightarrow \epsilon \\ \quad | ST \\ \\ T \rightarrow CC \\ \\ C \rightarrow aB \\ \\ B \rightarrow b \\ \quad | bB \end{array}$$

many correct
answers

Problem 3

The following grammar is a simplified version of lisp arithmetic expressions (so simplified that we only have one arithmetic operator in our version). The alphabet is $\{\text{num}, +, (,)\}$

$E \rightarrow \text{num}$
 $\quad | (+ A)$
 $A \rightarrow A A$
 $\quad | E$

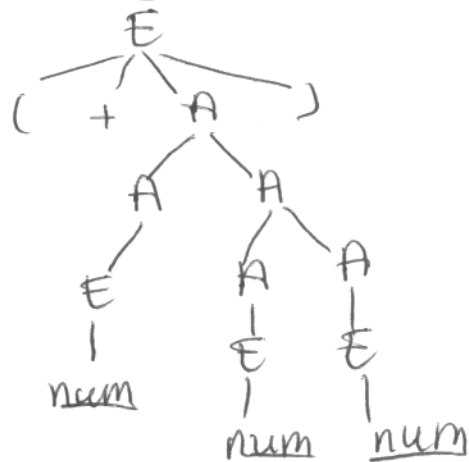
Note that lisp arithmetic expressions are allowed to take more than two operands. For example in $(+ (+ 2 3) 4 7)$ the outer plus has three operands.

In parts A through F that follow, we will be asking several questions related to this grammar.

Problem 3A [5 pts.]

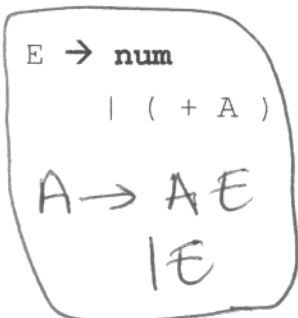
Show that the lisp grammar above is ambiguous.

String: $(+ \text{num } \text{num } \text{num})$



Problem 3B [5 pts.]

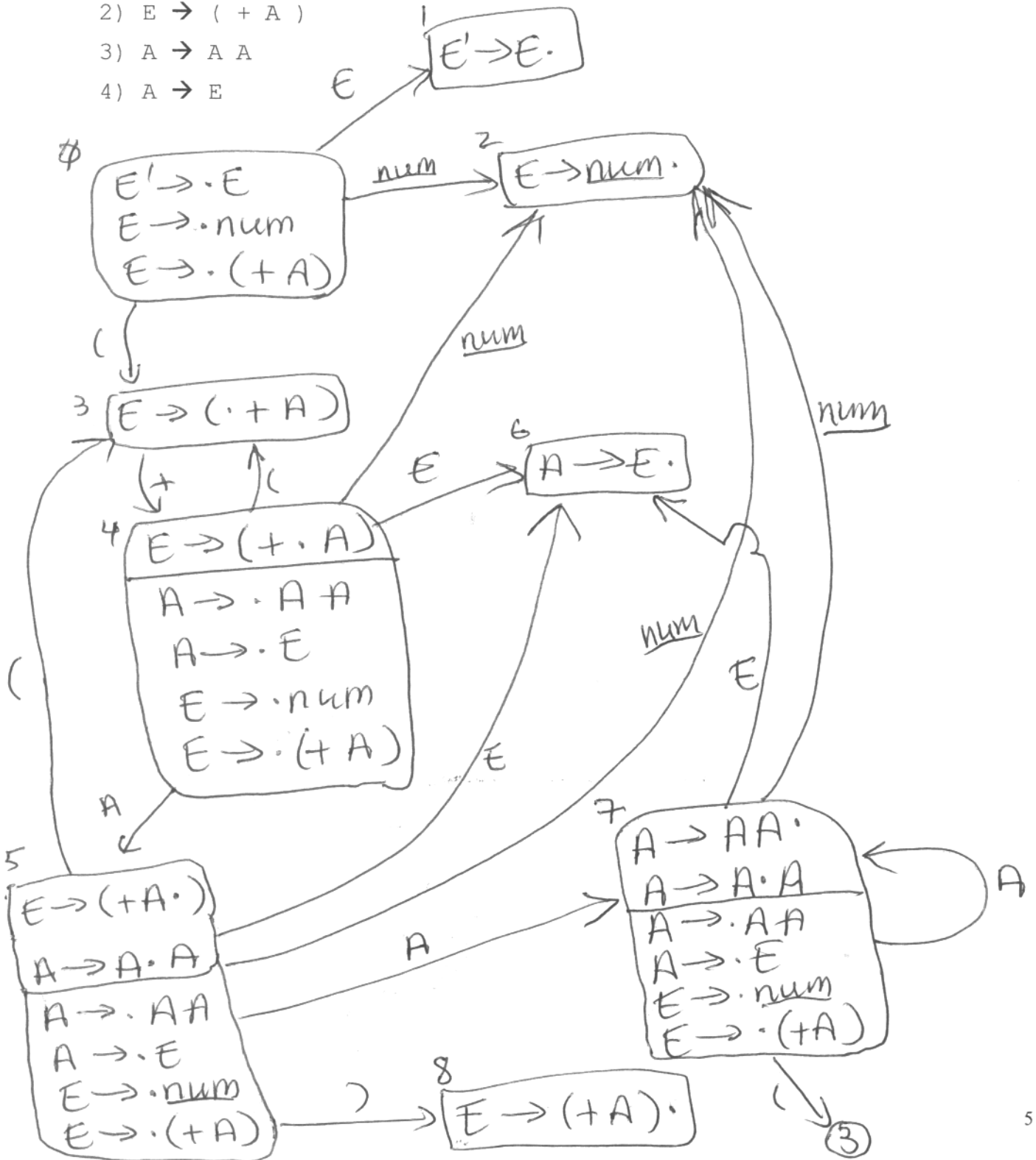
Write an unambiguous version of the grammar (i.e., your grammar must recognize the same language as the original grammar). Hint: The E-productions will not have to change, so they are already written in here for you.



Problem 3C [15 pts.]

Complete the DFA for recognizing viable prefixes for an SLR parser for the original grammar (shown again below). Show the set of items for each state. Note: the grammar has already been augmented for you.

- 0) $E' \rightarrow E$
- 1) $E \rightarrow \text{num}$
- 2) $E \rightarrow (+ A)$
- 3) $A \rightarrow A A$
- 4) $A \rightarrow E$



Problem 3D [10 pts.]

Give the FIRST and FOLLOW sets for each of the nonterminals in the grammar (shown again here).

- 0) $E' \rightarrow E$
- 1) $E \rightarrow \text{num}$
- 2) $E \rightarrow (+ A)$
- 3) $A \rightarrow A A$
- 4) $A \rightarrow E$

$$\begin{aligned}\text{FIRST}(E) &= \{(, \text{num}\} \\ \text{FIR}(E') &= \text{FIR}(E) = \{(, \text{num}\} \\ \text{FIR}(A) &= \text{FIR}(E) = \{(, \text{num}\}\end{aligned}$$

$$\begin{aligned}\text{FOLLOW}(E') &= \$ \\ \text{FOL}(E) &= \text{FOL}(E') \cup \text{FOL}(A) \\ &= \{ \$,), (, \text{num}\} \\ \text{FOL}(A) &= \{)\} \cup \text{FIR}(A) \\ &= \{), (, \text{num}\}\end{aligned}$$

Problem 3E [10 pts.]

If an SLR parser for the grammar would result in shift-reduce or reduce-reduce conflicts, show all such conflicts here (i.e., show the states, inputs, and relevant entries for the conflicts in the action table, based on your answers to the previous parts of the problem). If it has none, write NO CONFLICTS, and give your reasoning based on your answers to the previous parts of the problem. You are not required to build the whole action or goto table.

$$\text{action}[7, \text{num}] = s2 / r3 \quad (A \rightarrow AA)$$

$$\text{action}[7, (] = s3 / r3 \quad (A \rightarrow AA)$$

$$(\text{num} + '(' \text{ in } \text{FOLLOW}(A))$$

Problem 3F [10 pts.]

The the lisp grammar is shown again below. Write semantic rules to compute the integer semantic attribute $E.nd$, which is the deepest *nesting depth* (nd) of parentheses in the expression parsed. Here are some examples:

Expression	$E.nd$
(+ 3 2 9)	1
5	0
(+ (+ 3 4 (+ 7) 2) 5 (+ 7 4))	3

Note: the subscripts below are only to distinguish multiple instances of a non-terminal in a production. You may not use any global variables in your answer.

$$\begin{array}{l}
 E \rightarrow \text{num} \quad \{E.nd = \emptyset; \} \\
 \quad | (+ A) \quad \{E.nd = A.nd + 1; \} \\
 A \rightarrow A_1 A_2 \quad \{A.nd = (A_1.nd > A_2.nd) ? A_1.nd : A_2.nd; \} \\
 \quad | E \quad \{A.nd = E.nd \}
 \end{array}$$

or write w/ an if stmt.