CS 5900 Compiler
Final

Name: Mounika Duggula   ID: 700706184

3Q) Remove left recursion:
exp → exp addop term | term

Sol:
Rules: $A \to A\alpha \mid \beta$
$A \to \beta A'$
$A' \to \alpha A' \mid \varepsilon$

exp → term exp'
exp' → addop term exp' | ε

4Q) left factor the grammar
lexp → atom list | atom term1 term2

Sol:
Rules: $A \to \alpha\beta \mid \alpha\delta$
$A \to \alpha A'$
$A' \to \beta \mid \delta$

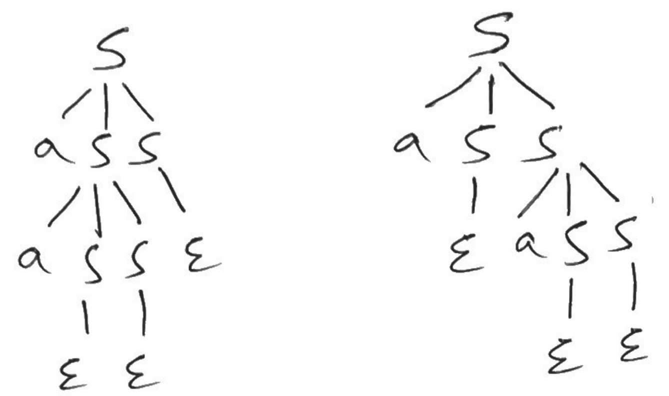
lexp → atom lexp'
lexp' → list | term1 term2

5Q) Given CF grammar S → aSS | ε

Sol:
1. left-most derivation for aa

S ⇒ aSS   (S → aSS)
⇒ aaSSS   (S → ε)
⇒ aaSS   (S → ε)
⇒ aaS   (S → ε)
⇒ aa

2. To show the grammar is ambiguous take one string aa and draw parse tree



as we got 2 distinct parse trees the given grammar is ambiguous.

6Q) LL(1) parsing for string ()()

M[N,T]	(	)	$
S	S→(S)S	S→ε	S→ε

Sol:

	Parsing stack	input	Action
1.	$S	()()$	S→(S)S
2.	$S)S(	()()$	match
3.	$S)S	)()$	S→ε
4.	$S)	)()$	match
5.	$S	()$	S→(S)S
6.	$S)S(	()$	match
7.	$S)S	)$	S→ε
8.	$S)	)$	match
9.	$S	$	S→ε
10	$	$	Accept

7Q) Remove left recursion:

$T \to Aa \mid b$

$A \to Ac \mid Td \mid \varepsilon$

Sol: Rules: $A \to A\alpha \mid \beta$

$A \to \beta A'$

$A' \to \alpha A' \mid \varepsilon$

As $T \to Aa \mid b$ is not in the rule no need to do the left recursion for $T \to Aa \mid b$

Substitute this $T \to Aa \mid b$ in $A \to Ac \mid Td \mid \varepsilon$ and apply rules to A

$A \to Ac \mid Aad \mid bd \mid \varepsilon$

apply rules to A

$A \to bdA' \mid \varepsilon A'$

$A' \to cA' \mid adA' \mid \varepsilon$

8Q) $S \to (AS) \mid \varepsilon$

$A \to S \mid \varepsilon$

1. LR(0) items

Sol:

Augmented grammar:

$S' \to S$ where S' - new start symbol

$S \to (AS) \mid \varepsilon$

$A \to S \mid \varepsilon$

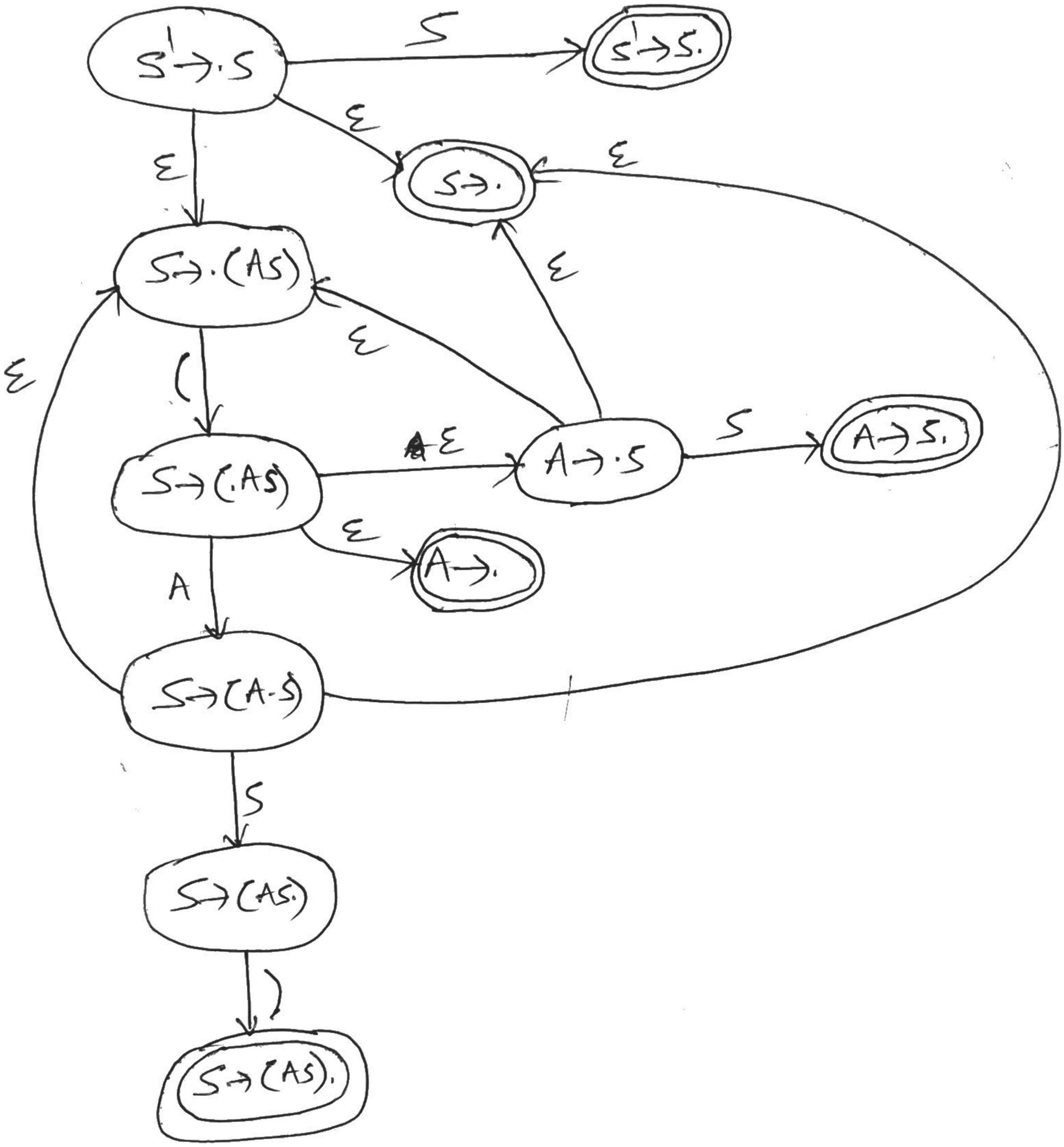
LR(0) items:

$S' \to .S$	$S \to .(AS)$	$S \to .$	$A \to .S$	$A \to .$
$S' \to S.$	$S \to (.AS)$		$A \to S.$	
	$S \to (A.S)$			
	$S \to (AS.)$			
	$S \to (AS).$			

11 LR(0) items are there so NFA will have 11 states.

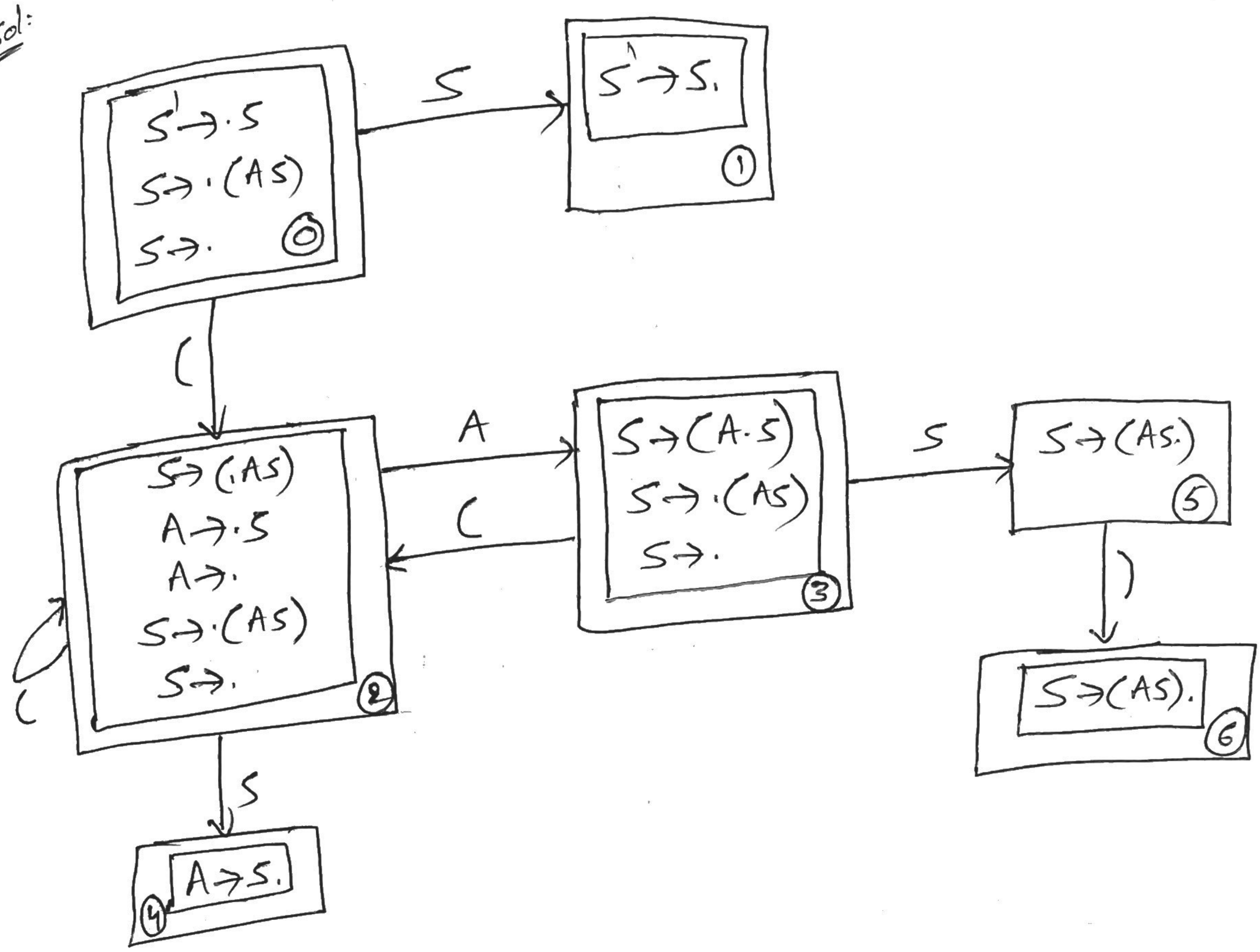
2. NFA of LR(0).

Sol:



3. DFA for the above NFA

Sol:



9Q) SLR(1) parser for the input (()())).

	Parsing stack	Input	Action
1.	\$0	(()()))\$	shift
2.	\$0(2	()()))\$	shift
3.	\$0(2(2	)()))\$	Reduce S→ε
4.	\$0(2(2S3	)()))\$	Reduce S→ε
5.	\$0(2(2S3S4	)()))\$	Shift
6.	\$0(2(2S3S4)5	()))\$	Reduce S→(SS)
7.	\$0(2S3	()))\$	shift
8.	\$0(2S3(2	)))\$	Reduce S→ε
9.	\$0(2S3(2S3	)))\$	Reduce S→ε
10.	\$0(2S3(2S3S4	)))\$	shift
11.	\$0(2S3(2S3S4)5	))\$	Reduce S→(SS)
12.	\$0(2S3S4	))\$	shift
13.	\$0(2S3S4)5	)\$	Reduce S→(SS)
14.	\$0S1	)\$	Reduce S'→S
15.	\$0S'	)\$	Not Accepted Error.

At step 15 we have reached to initial state 0 but the input is not empty and it doesn't accept) at state 0 so the given input string (()())) is not accepted.

1Q) CFG to regex

$S \rightarrow aSA \mid B$

$A \rightarrow c$

$B \rightarrow Bb \mid b$

Sol: The given grammar will accept the strings as

$\{b, bbb\ldots, abc, aaabc, aaabbbccc\ldots\}$

we can write the regular expression as $(a^+b^+c^+) \mid b^+$

2Q) Regex to CFG

$(a|b)(a|cb)^+(c|\varepsilon)$

Sol: $S \rightarrow ABC$

$A \rightarrow a|b$

$B \rightarrow DB|D$

$D \rightarrow a|cb$

$C \rightarrow c|\varepsilon$

where
$B \rightarrow (a|cb)^+$