J. Varun Iyer 19115037

Compiler Design Assignment - 1

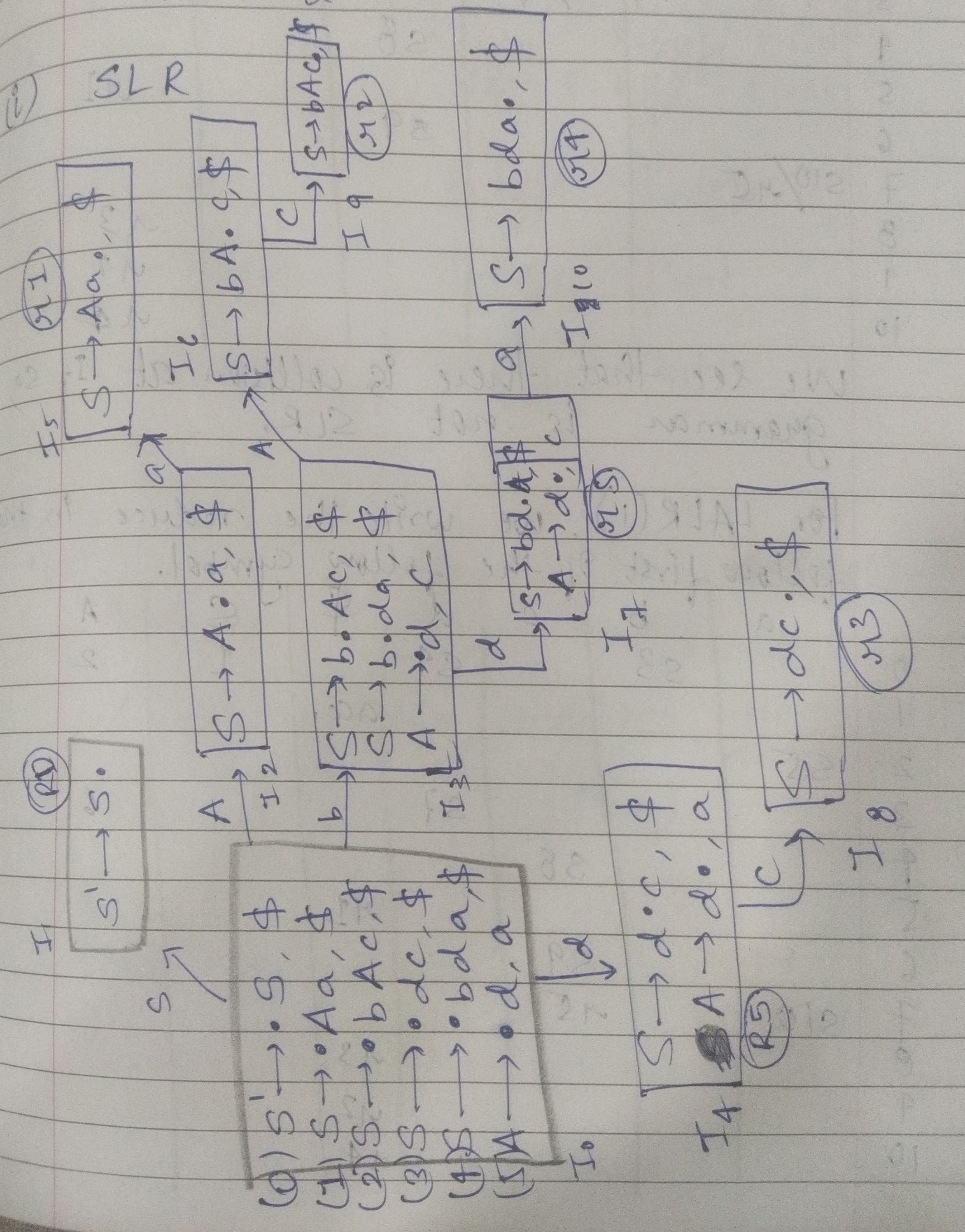
Q1.

→ Given grammar:

$S \rightarrow Aa \,/\, bAc \,/\, dc \,/\, bda$

$A \rightarrow d$

(i) SLR



For SLR, reduce is written in follow of grammar symbol.

	a	b	c	d	$
0		s3		s4	
1					accept
2	s5				
3				s7	
4			s8		
5					r1
6			s9		
7	s10/r5				
8					r3
9					r2
10					r4

We see that there is collision at I_7 so, grammar is not SLR.

For LALR(1), we write the reduce in the ~~follow~~ first of the follow symbol.

	a	b	c	d	$	S	A
0		s3		s4		1	2
1					acc.		
2	s5						
3				s7			6
4			s8				
5					r1		
6			s9				
7	s10		r5				
8					r3		
9					r2		
10					r4		

Here, since each cell only has one value, therefore the grammar is LALR(1).

Q3.

→

Production	Semantic rules.
$E \rightarrow E_1 + T$	if (E_1.type = int & T.type = int) then E.type = int else E.type = float
$E \rightarrow T$	E.type = T.type
$T \rightarrow num.num$	T.type = float
$T \rightarrow num$	T.type = int

Q4.2.

→ (a) In C- language, we have the following structure for the FOR loop.

for (initialization; conditions; increment/decrement)

for (i = 0; i <= 10; i++)

keyword : for

terminals : (, =, ;, <, =, >, ++, --

variables

So, we have the following CFG:

$P \rightarrow S$

$S \rightarrow XY$

$X \rightarrow$ for

$Y \rightarrow (A ; B ; C)$

$A \rightarrow V = N$

$B \rightarrow V == N / V <= N / V >= N$

$C \rightarrow V++ / V--$

$N \rightarrow 0-9$

$V \rightarrow W / a-z W$

$W \rightarrow W / a-z W / 0-9 W / \epsilon$

Example : for (i=0; i<=5; i++)

$P \rightarrow S$

$S \rightarrow XY$

for Y

for (A ; B ; C)

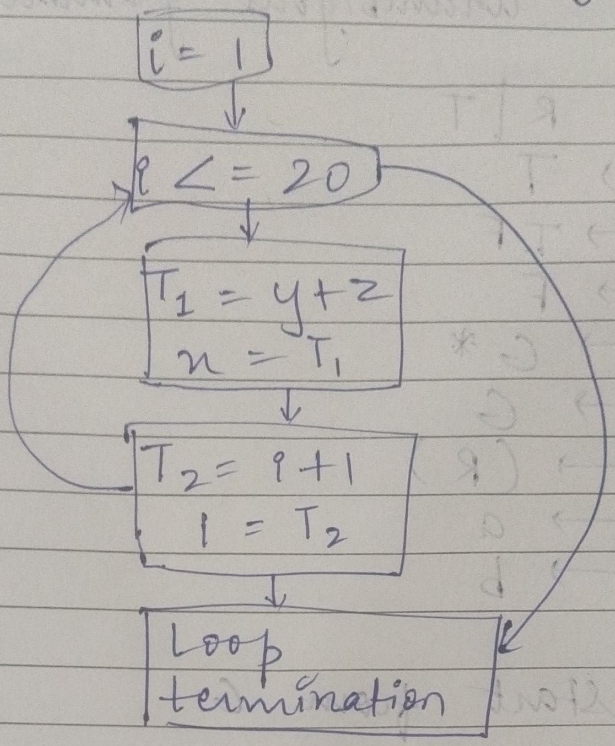
for (V=N; V<=N; V++)

for (i=0; i<=5; i++)

•) Three address code for FOR loop

1. V = N (V = variable, N = Number)
2. If condition goto 4
3. goto 8
4. Statement 1.
5. Statement 2.
6. V++ or V--
7. goto 2
8. exit

For loop & circuit diagram:



Q4.

→ (a) Given grammar:

$R \rightarrow R|R / RR / R* / (R) / a / b$

First (R) = { a, b, (}

Follow (R) = {), *,), a, b, (, $ }

(b) Parsing table

Non terminal	a	b	(	)	*		$
R	1,2,3,5	1,2,3,6	1,2,3,4				

As cells have multiple entries, the grammar is ambiguous.

(c) New unambiguous grammar:

$R \rightarrow R|T$
$R \rightarrow T$
$T \rightarrow TF$
$T \rightarrow F$
$F \rightarrow G*$
$F \rightarrow G$
$G \rightarrow (R)$
$G \rightarrow a$
$G \rightarrow b$

(d) We start from G.

FIRST (G) = {a, b, (}
FIRST (F) = FIRST (G) = {a, b, (}
FIRST (T) = FIRST (F) = {a, b, (}
FIRST (R) = FIRST (T) = {a, b, (}

FOLLOW (R) = {$, |,)}
FOLLOW (F) = {$, |,)}
FOLLOW (T) = {$, |,)}
FOLLOW (G) = {*, $,), |}

Parsing table:

Non-terminal	a	b	(	)	*	|	$
R	1, 2	1, 2	1, 2				
F	5, 6	5, 6	5, 6				
T	3, 4	3, 4	3, 4				
G	8	9	7				

Since, cells have multiple have entries, therefore, the grammar is not LL(1).