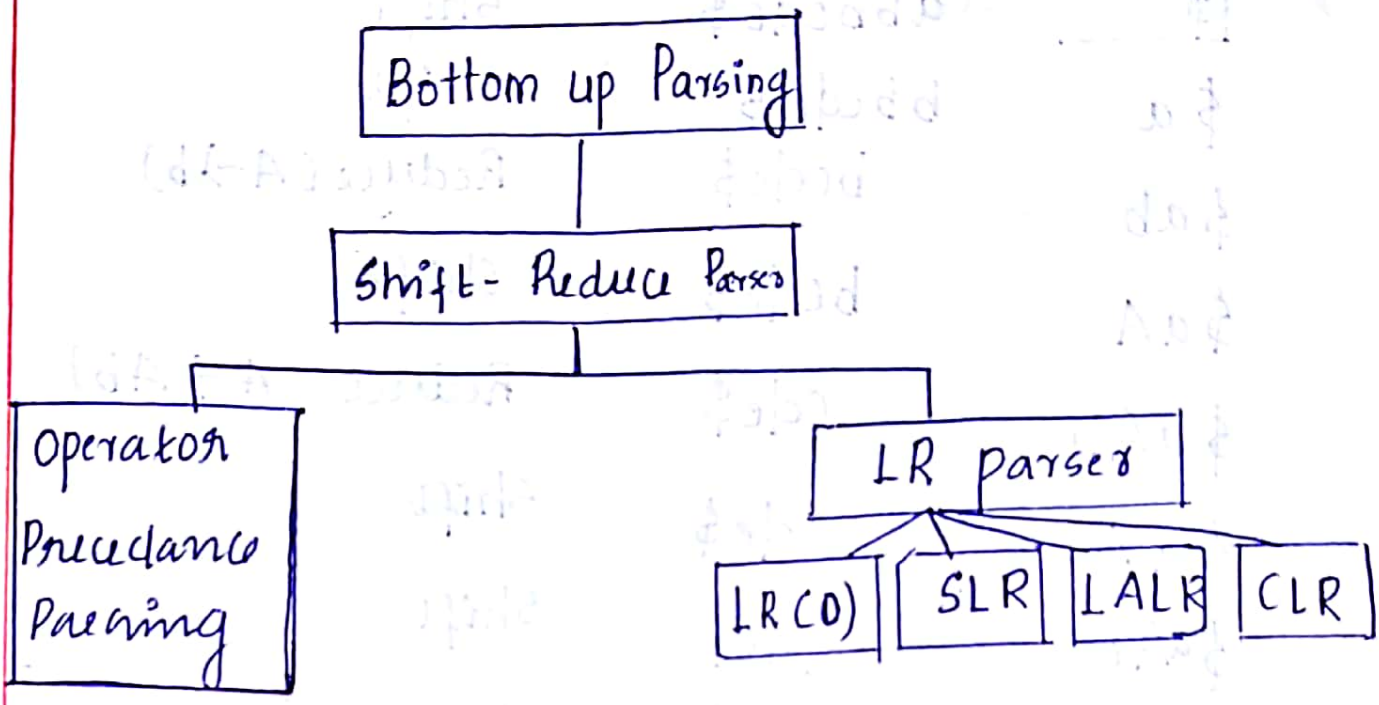
16/2/2018

Module - III



Shift Reduce Parsing:

eg: Consider the following set of production.

$S \rightarrow aAcBe$

$A \rightarrow Ab \mid b$

$B \rightarrow d$

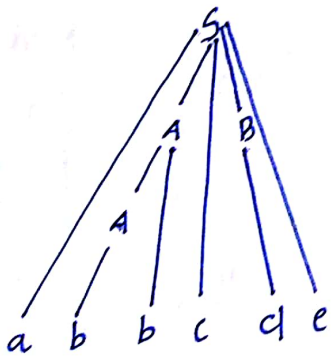
Shift reduce parsing of input is done as follows:

KTU STUDENTS

Scanned by CamScanner

Stack	Input buffer	Operation
$	abbcde$	Shift
$a	bbcde$	Shift
$ab	bcde$	Reduce (A→b)
$aA	cde$	Shift
$aAb	cde$	Reduce (A→Ab)
$aA	de$	Shift
$aAc	e$	Shift
$aAcd	e$	reduce (B→d)
$aAcB	$	reduce (A→aAcBe)
$aAcBe	$	accept string
$S		

Parse Tree:



Operator Precedence Parsing:

It is applied to small class of grammars or P

Operator grammar has two important characteristics:

1) There is no ε production in grammar.
2) Production should not have two adjacent variables.

eg: for operator grammar.

E → E+E
E → E*E
E → id

eg: fo non-operator grammar.

E → EPE
P → */+
E → id

Components Operator

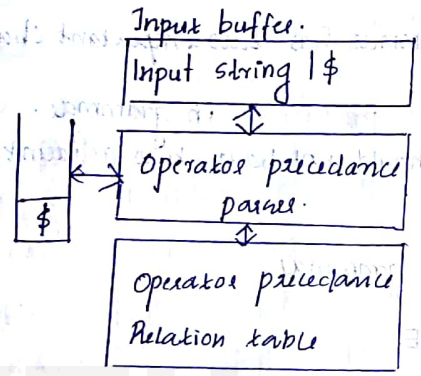
1) Input buffer.
2) Stack → Top element is '$'

KTU STUDENTS

Scanned by CamScanner

3) Operator precedance Parser:

4) Operator precedance Relation table



Precedance relation between a & b is given below

Relation	meaning
$a \gtrdot b$	a has higher precedance than b
$b \gtrdot a$	a has lower precedance than b.
$a = b$.	Equal precedance.

Q: operator precedance ~~oper~~ table for below grammar.

$E \rightarrow E+E$

$E \rightarrow E*E$

$E \rightarrow id$

Terminals - +, *, id, $

	id	+	*	$
id	–	$\gtrdot$	$\gtrdot$	$\gtrdot$
+	$\lessdot$	$\gtrdot$	$\lessdot$	$\gtrdot$
*	$\lessdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$
$	$\lessdot$	$\lessdot$	$\lessdot$	–

Parsing Algorithm:

if ($ S is on TOS and $ is ~~in~~ input
then accept
else
{ if (a is on TOS and b is an input)
if (a $\lessdot$ b or a = b)
then shift b onto TOS
else if (a $\gtrdot$ b)
then reduce.

KTU STUDENTS

Scanned by CamScanner

```
else
    print(error);
}
```

Parse the input string "id + id"

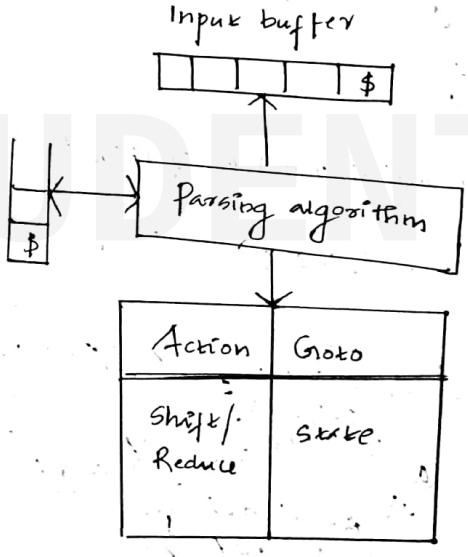
Stack	Input	Operation
$	id + id $	$ <· id, shift
$id	+ id $	id ·> +, reduce
$E	+ id $	$ <· +, shift
$E +	id $	+ <· id, shift
$E + id	$	id ·> $, reduce E → id
$E + E	$	+ ·> $, reduce E → E + E
$E	$	Accept

E → E + E
E → E * E
E → id

LR Parsing

LR(0) SLR(1) LALR(1) CLR(1)

LR –
SLR – Simple
LALR – Look ahead
CLR – Canonical



KTU STUDENTS

Scanned by CamScanner

SLR Parsing

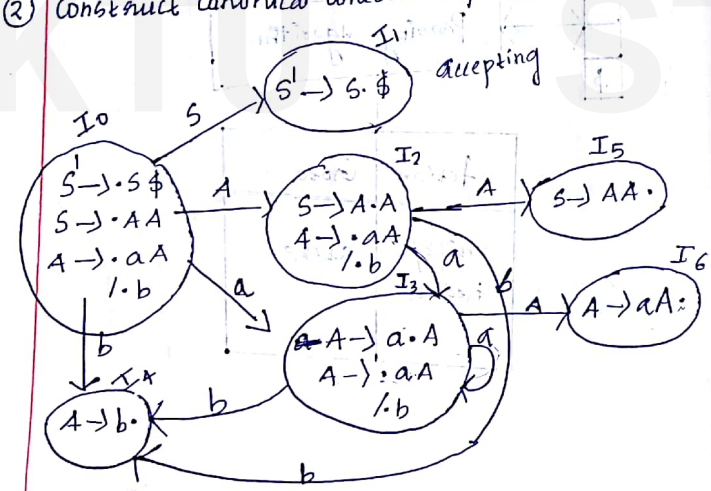
- Simple LR parsing
- Bottom up parsing technique

eg. $S \to AA$
$A \to aA$
$\quad | \; b$

① Construct augmented grammar.

$S' \to S\$$
$S \to AA$
$A \to aA$
$\quad | \; b$

② Construct canonical collection of LR(0) items.



Number the productions

$S \to AA$ — (r_1)
$A \to aA$ — (r_2)
$\quad | \; b$ — (r_3)

follow (S) = {$}
follow (A) = {a, b}

Now construct parsing table:

State	Action			Goto	
	a	b	$	S	A
0	s_3	s_4		1	2
1			Accept		
2	s_3	s_4			5
3	s_3	s_4			6
4	r_3	r_3			
5			r_1		
6	r_2	r_2			

KTU STUDENTS

Scanned by CamScanner

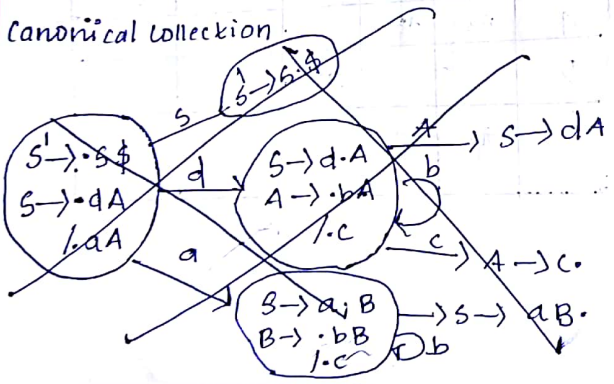
Q Construct SLR parsing table for below grammar.

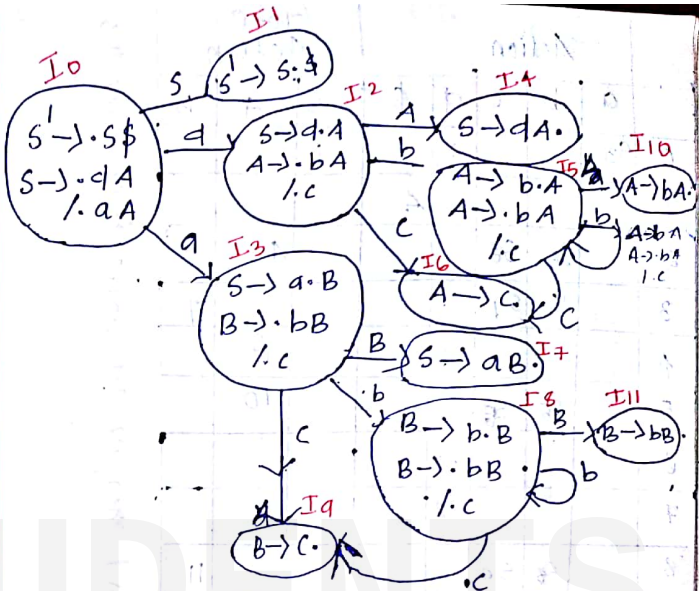
$S \to dA$ — ①
$\quad / aB$ — ②
$A \to bA$ — ③
$\quad / c$ — ④
$B \to bB$ — ⑤
$\quad / c$ — ⑥

A augmented grammar

$S' \to S\$$
$S \to dA / aB$
$A \to bA / c$
$B \to bB / c$

Canonical collection





follow (S) = {$}
follow (A) = {$}
follow (B) = {$}

KIU STUDENTS

Scanned by CamScanner

	Action					Goto		
	a	b	c	d	$	S	A	B
0	S_3			S_2		1		
1					A			
2		S_5	S_6				4	
3		S_8	S_9					7
4				r_1				
5		S_5	S_6				10	
6				r_4				
7				r_2				
8		S_8	S_9					11
9				r_6				
10				r_3				
11				r_5				

eg follow(A) → $ then we can write item 10 like r_3

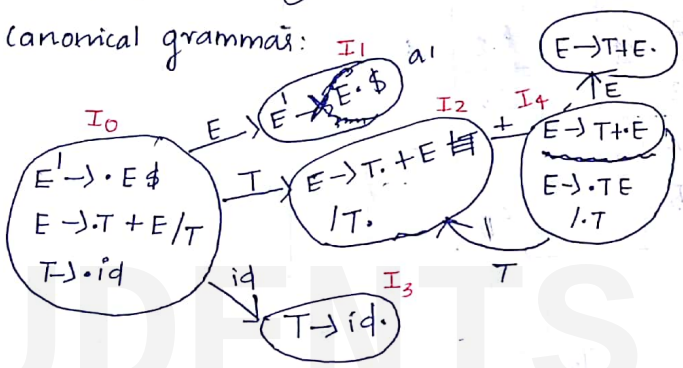
Q Construct SLR parsing table for below grammar.

E → T + E
/ T
T → id

augmented grammar

E' → E$ —
E → T + E —(1)
/ T —(2)
T → id —(3)

canonical grammar:



follow(E) = { $ }
follow(T) = { $, + }

KTU STUDENTS

Scanned by CamScanner

	Action			goto	
	+	id	$	E	T
0		s_3		1	2
1			ACCEPT		
2	s_4		r_2		
3			r_3		
4				5	2
5			r_1		

I_2

E → T. + E

/ T.

+ → s_4

T is reduced in (2)

	+
5	s/r

Shift Reduce conflict

Reduce Reduce conflict.

13/2018

Note

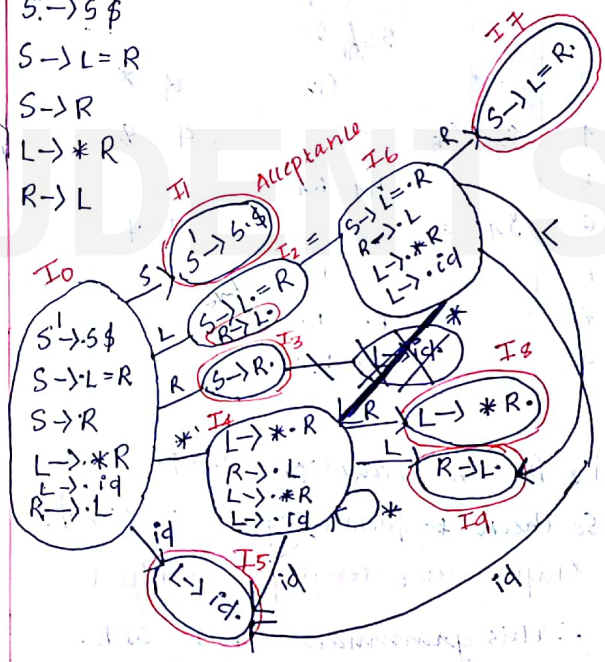
Every SLR(1) grammar is unambiguous. But there are many unambiguous grammar that are not SLR(1)

eg:

S → L = R — ①
S → R — ②
L → * R — ③
L → id — ④
R → L — ⑤

Soln

S' → S $
S → L = R
S → R
L → * R
R → L



KTU STUDENTS

Scanned by CamScanner

follow(S) = { $ }

follow(L) = { =, $ }

follow(R) = { =, $ }

Parsing table.

	ACTION				GOTO		
	*	id	=	$	S	L	R
0	S4	S5			1	2	3
1				Accept			
2			S6/R5	R5			
3				R2		~~4~~	~~8~~
4	S4	S5				9	8
5			R4	R4			
6	S4	S5				9	7
7				R1			
8			R3	R3			
9			R4	R4			

- I_2 has a reduction R -> L.

So check follow of R = { =, $ }

Shift reduce ~~conflict~~ conflict

∴ this grammar is not SLR.

- Reduce - Reduce Conflict

In

A -> α.

B -> β.

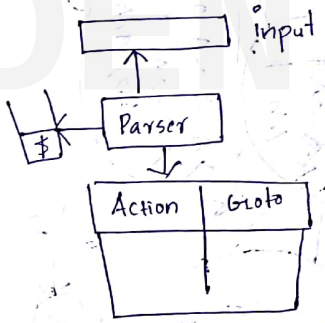
follow(A) ∩ follow(B) ≠ ∅

CLR(1) Parsing

- LR parsing technique

CLR - Canonical LR

Require canonical collection of LR(1) items



Parsing table.

KTU STUDENTS

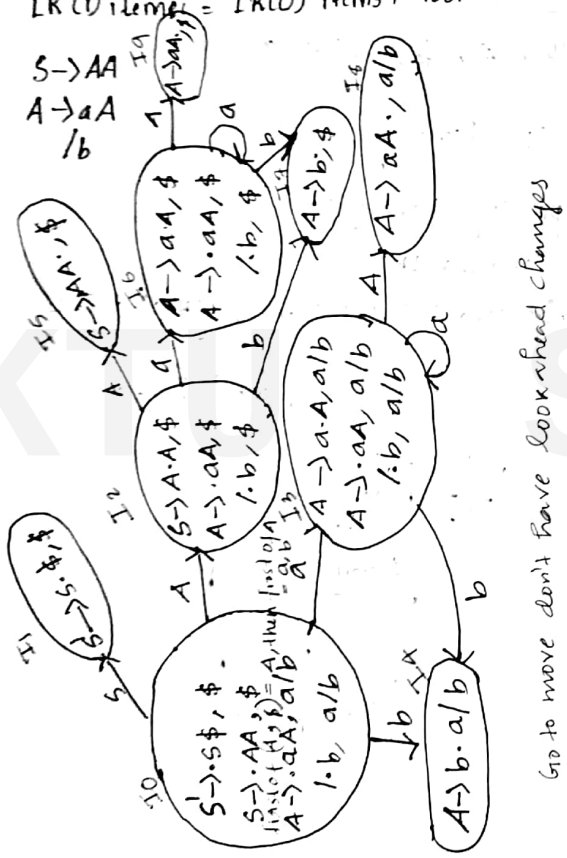
Scanned by CamScanner

LR(0) SLR(1) CLR LALR

need LR(0) items LR(1) items

LR(1) items = LR(0) items + look ahead

eg: S->AA
A->aA
/b



Go to move don't have lookahead changes

	Action			GoTo	
	a	b	$	S	A
0	s_3	s_4		1	2
1			Acc		
2	s_6	s_7			5
3	s_3	s_4			8
4	r_3	r_3			
5			r_1		
6	s_6	s_7			9
7			r_3		
8	r_2	r_2			
9			r_2		

S -> AA — (1)
A -> aA — (2)
A -> b — (3)

3/3/18

LALR(1) parsing table for above grammar

Identify same items with different look ahead if any such item sets are present form a new state by merging similar items.

$I_3, I_6 \rightarrow I_{36}$

$I_8, I_9 \rightarrow I_{89}$

$I_4\ I_7 \rightarrow I_{47}$

KTU STUDENTS

Scanned by CamScanner

	ACTION			GOTO	
	a	b	$	S	A
0	S_{36}	S_{47}		1	2
1			Accept		r
2	S_{36}	S_{47}			5
36 3	S_{36}	S_{47}			89
47 4	R_3	R_3	R_3		
5			R_1		
89 6	R_2	R_2	R_2		

14/3/18

LALR

- look ahead LR parser
- size of the CLR parsing table is large compared to other parsing technique
- LALR reduces the size of CLR parsing
  CLR is merged with s

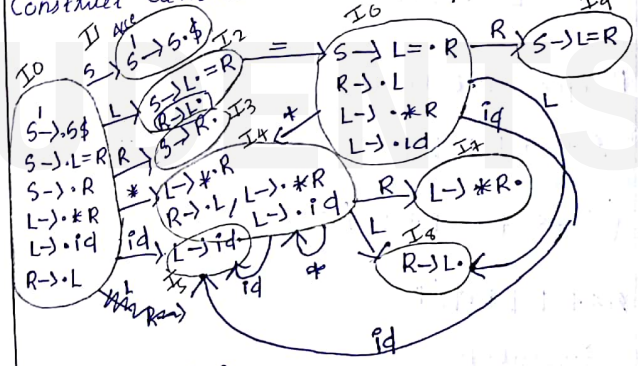
Q Construct SLR parsing table for below grammar.

S -> L = R / R
L -> * R / id
R -> L

Sol^n Construct augmented grammar.

S' -> S $
S -> * R / id
R -> L

Construct canonical collection of LR(0) items



follow(S) = {$}
follow(L) = {=, $}
follow(R) = {=, $}.

KTU STUDENTS

Scanned by CamScanner