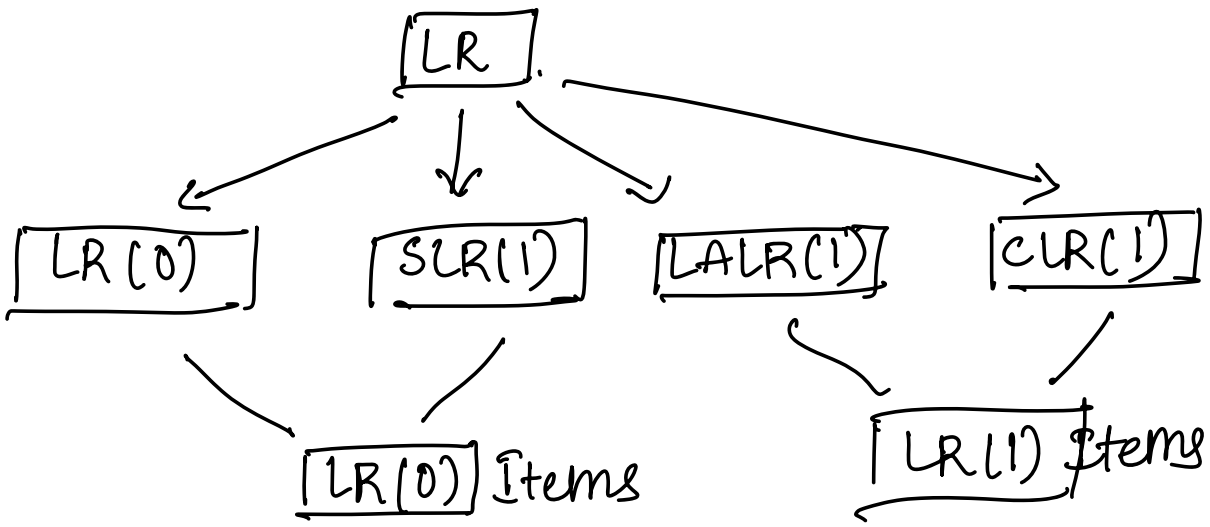
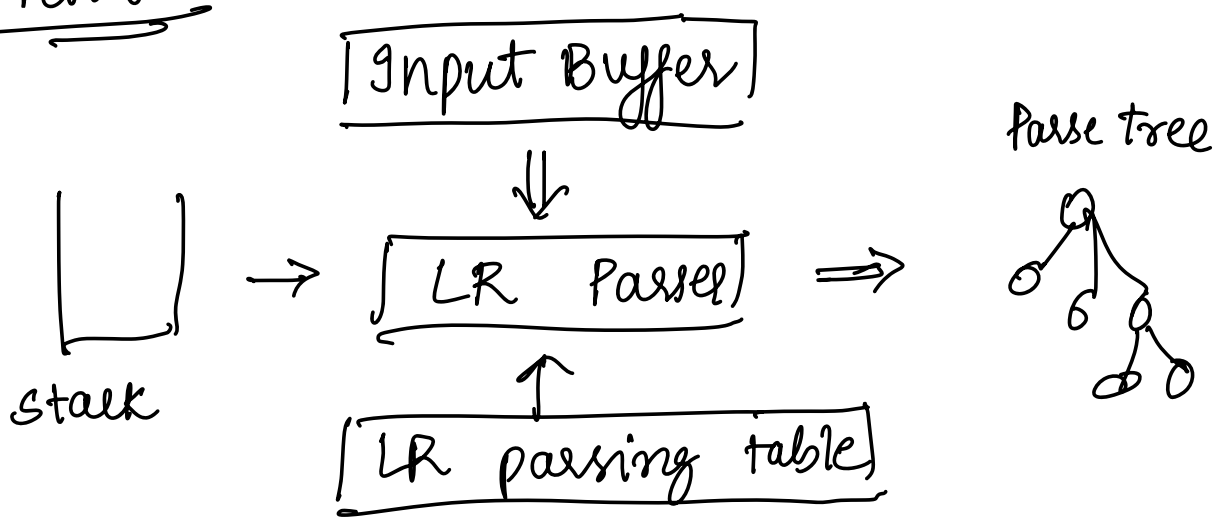
Bottom-up parsers:

- operator precedence
- LR Parsers
  - LR(0)
  - SLR(1)
  - LA LR(1)
  - CLR(1)



Architecture:



SLR (1) : Simple LR

LALR(1) : Look Ahead LR

CLR(1) : Canonical LR

(i) LR(0) Parser:

Steps:

1. Augment the given grammar
2. Draw the canonical collection of LR(0) items.
3. Number the productions.
4. Create the parsing Table
5. Stack Implementation
6. Build the parse tree.

Example: Given grammar is

$E \rightarrow BB$

$B \rightarrow cB \mid d$

Input string is "ccdd"

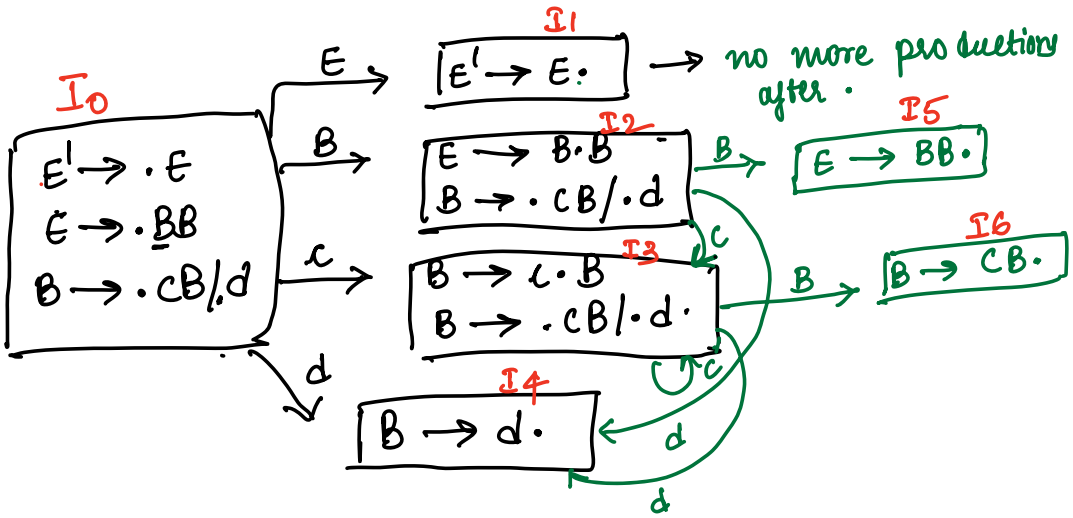
1. Augment the given grammar

Augmented grammar
$E' \rightarrow E$ (Add a new production by calling start symbol)
$E \rightarrow BB$
$B \rightarrow cB/d$

2. Draw canonical collection of LR(0) items

$E' \rightarrow E$
$E \rightarrow BB$
$B \rightarrow cB/d$

LR(0) Items: $E' \rightarrow \cdot E$ (Add a · to any production at starting of RHS → LR(0) items)
$E \rightarrow \cdot BB$
$B \rightarrow \cdot cB/\cdot d$



3. Number the productions:

$E' \rightarrow E$ → Augmented production

$E \rightarrow BB$ → (1)

$B \rightarrow cB \mid d$ → (3)

(2)

$B \rightarrow cB$ & $B \rightarrow d$

(2) (3)

4. Parsing Table:

Action ↑ → shift (S), Reduce (r)

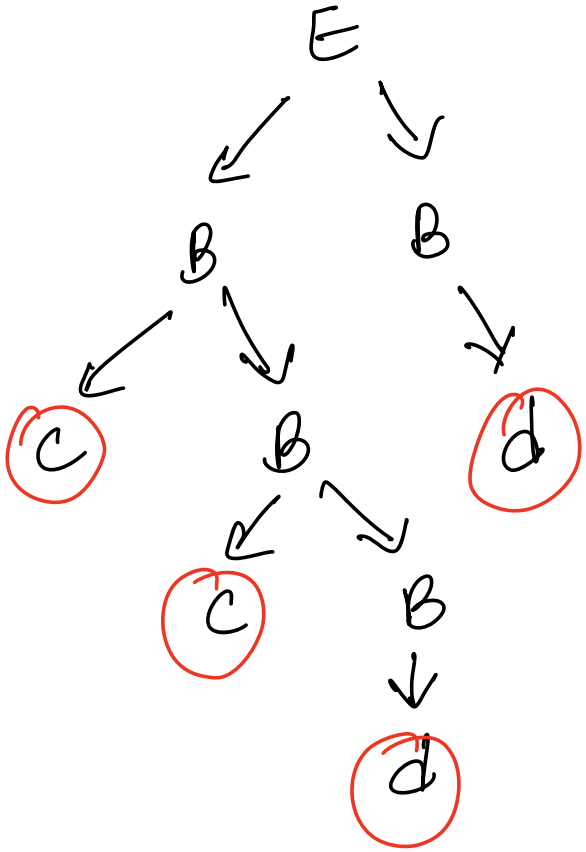
States	Action (Terminals)			GO TO (NT)	
All I values	c	d	$	E	B
0	S_3	S_4		1	2
1			Accept		
2	S_3	S_4			5
3	S_3	S_4			6
4	r_3	r_3	r_3		
5	r_1	r_1	r_1		
6	r_2	r_2	r_2		

5. Stack implementation:

stack	input	Actions
$ 0 'state 0'	c c d d $	s_3: shift c into stack and goto 3
$ 0 c 3	c d d $	s_3 : shift c into stack & goto 3
$ 0 c 3 c 3	d d $	s_4 : shift d & 4
$ 0 c 3 c 3 d 4	d $	r_3 : reduce 'd' by B → d
$ 0 c 3 c 3 B 6 (6 because 3 state and B column value is '6')	d $	r_2 : reduce by B → cB
$ 0 c 3 B 6	d $	r_2 : reduce by B → cB
$ 0 B 2	d $	s_4 : shift 'd' & goto 4

$ 0 B_2 d 4	$	r_3: reduce by production no: 3 $B \rightarrow d$
$ 0 B_2 B_5 Convert E to	$	r_1: reduce by $E \rightarrow BB$
$ 0 E 1	$	Accept

6. Parse tree



cddd.

NOTE:

If the passing table is having a dual value "rs or rr" (r: reduce s: shift) together in any of the box, then clearly. you can say it is not a LR(0)

eg: | r_1/r_2 | r/s | → X LR(0)