SLR (1) : Simple LR

It is an advanced version of LR(0) and also uses LR(0) items only.

→ same 6 steps as the LR(0) parser only at the 4th step

→ create the parsing table, there is a slight difference

considering the same example:

$E \rightarrow BB$

$B \rightarrow cB/d$

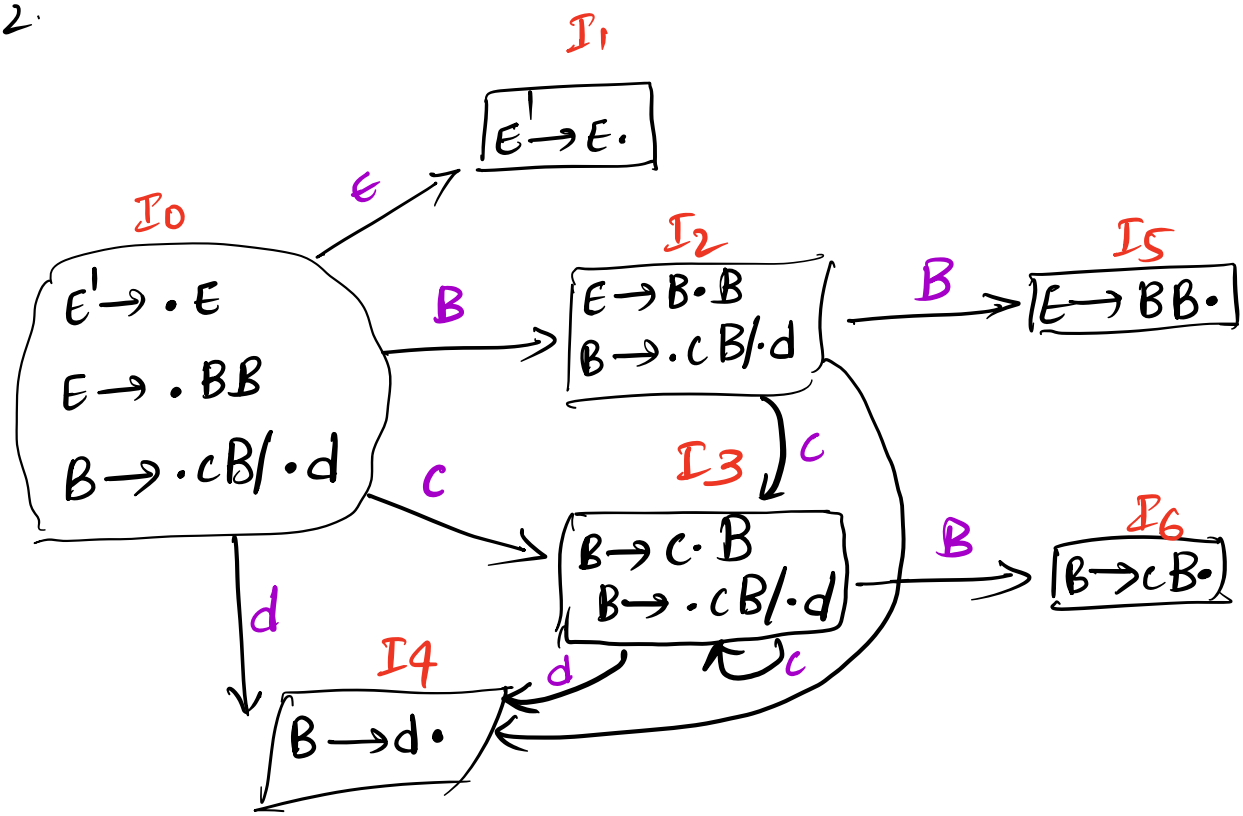
1. After Augmenting the grammar

$E' \rightarrow E$

$E \rightarrow BB$

$B \rightarrow cB/d$.

2.



3. Number the production

$E' \rightarrow E$

$E \rightarrow BB$ — (1)

$B \rightarrow cB \mid d$ (2) (3)

4. parsing Table

state	Action			Goto	
	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		
6	r_2	r_2	r_2		

for 1st state i.e I_1 as it is Augmented, we write Accept.

for 4th state, in LR(0) for whole row, we write reduction (r) in full row.

But for SLR, it is different

for I_4 i.e 4th state

$B \rightarrow d \cdot$

so, this matches 3rd production

$B \rightarrow d$

so, now we have to see the follow of 'B'.

follow of B = {c, d, $}

So, under c, d & $, we write r_3.

for 5th state, $E \rightarrow BB$.

So, $E \rightarrow BB$ is 1st production so (r_1)

find the follow of E

follow(E) = {$}

So, r_1 in $

for 6th state, $B \rightarrow cB$

follow(B) = {c, d, $}

& it is 2nd production. So r_2. So, SLR is considered to be a optimized version.

5. Stack Implementation:

Stack	Input	Action
$0	ccdd$	shift c → 3
$0c3	cdd$	shift c → 3
$0c3c3	dd$	shift d → 4
$0c3c3d4	d$	reduce d by B → d
$0c3c3B6	d$	reduce B by B → cB
$0c3B6	d$	reduce B B → cB
$0B2	d$	shift d → 4
$0B2d4	$	reduce B → d
$0B2B5	$	reduce E → BB
$0E1	$	Accept.

6. parse tree.