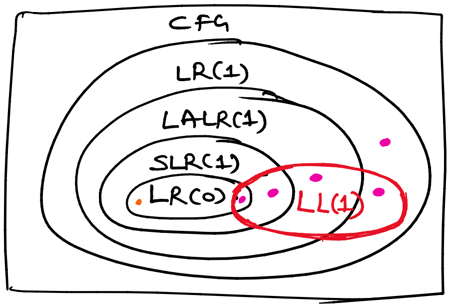
LR(K) parser

Tuesday, April 5, 2022 7:03 PM

LR(K) parser

- L → Left-to-right Scanning of i/p.
- R → Rightmost derivation in reverse.
- K → No of look ahead Symbols.

LR-Family LR(0) ⊆ SLR(1) ⊆ LALR(1) ⊆ CLR(1)/LR(1)



→ Every LR(0) is SLR(1) but every SLR(1) need not be LR(0).

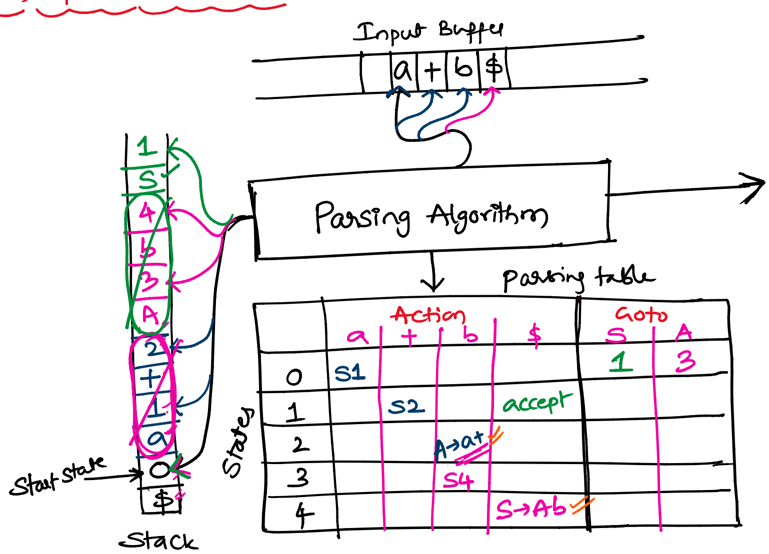
→ Every SLR(1) is LALR(1) but every LALR(1) need not be SLR(1)

→ Every LALR(1) is LR(1) but every LR(1) need not be LALR(1).

Note → Every LL(1) is also LR(1) but every LR(1) need not be LL(1).

Note → No ambiguous grammar can be LR(K)

LR(K) Parser Structure :-



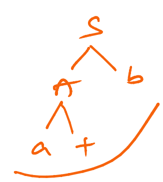
	Action a	Action +	Action b	Action $	Goto S	Goto A
0	S1				1	3
1		S2		accept		
2			A→a+			
3			S4			
4				S→Ab		

Initial Configuration

Stack	input
$0	w$

4 actions

1. Shift j ⟹ push i/p and state j
2. Reduce A→α ⟹ pop RHS, push LHS; push [Goto(current State, LHS)]
3. Accept ⟹ Successful completion of parsing.
4. Error ⟹ Unsuccessful parse.



LR(K) Family

- LR(0), SLR(1) — Uses LR(0) Items
- LALR(1), CLR(1)/LR(1) — Uses LR(1) Item

LR(0) Item's :- An LR(0) Item of a grammar 'G' to be a production of G with a dot at some position of the right side.

Ex:- A → XYZ generate 4 Items: A → •XYZ

A → X•YZ → Some part of input derived using X and remaining part is expecting to derive using YZ.

A → XY•Z

A → XYZ•

Note : A → ε generates only 1 Item i.e, A → •

Step1: Augument the grammar with production S' → S where 'S' is starting Nonterminal.

Step 1: Augment the grammar with production $S' \to S$ where 'S' is starting Nonterminal.

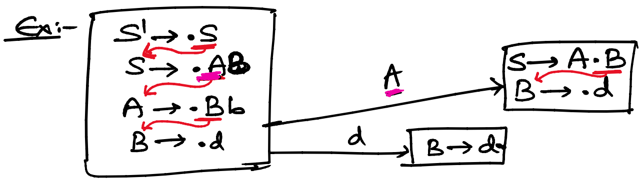
$S' \to .S$ ⇐ start.

$S' \to S.$ ⇐ end.

Step 2: Find the Closure (I).

If $A \to \alpha . B\beta$ is in state I and $B \to \gamma$ is a production then add $B \to .\gamma$ to I.

Ex:-

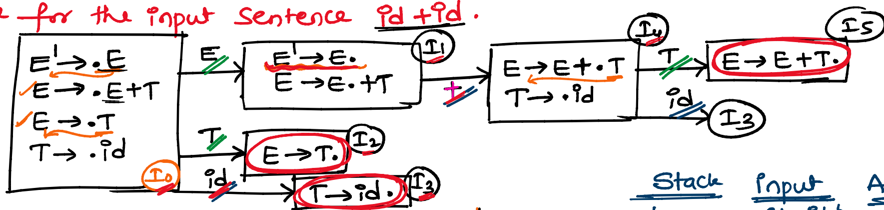


Step 3: Find Goto (I, β) where β is either terminal or nonterminal.

If $A \to \alpha . \beta\gamma$ is in state I, then Goto (I, β) = closure set of all items contains $A \to \alpha\beta . \gamma$

Q:- Find LR(0) States for the following grammar and Construct LR(0) parse table then construct a parse tree for the input sentence id + id.

$E \to E+T$
$E \to T$
$T \to id$



6-States

LR(0) Parse table

1. If $S' \to S.$ is in state I then
Set Action [I, \$] = Accept.

2. If $A \to \alpha . a\beta$ is in state I then
Set Action [I, a] = Shift J

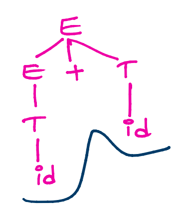
3. If $A \to \alpha . B\beta$ is in state I then
Set Goto (I, B) = J

4. If $A \to \alpha .$ is in state I then
Set Action [I, all terminals] = Reduce $A \to \alpha$.

	Action			Goto	
	id	+	\$	E	T
0	S3			1	2
1		S4	Accept		
2	E→T	E→T	E→T		
3	T→id	T→id	T→id		
4	S3				5
5	E→E+T	E→E+T	E→E+T		

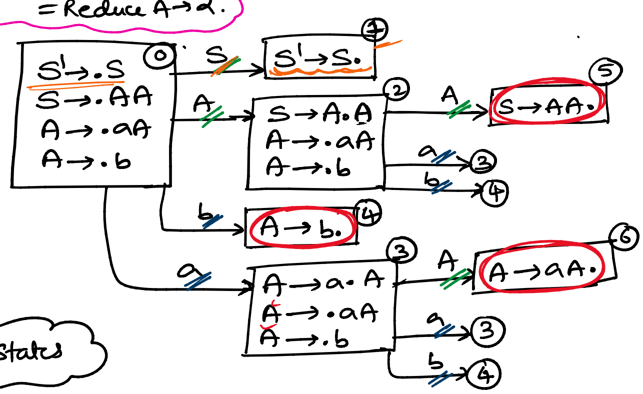
the grammar is LR(0)

Stack	Input	Actions
\$0	id+id\$	S3
\$0id3	+id\$	T→id
\$0T2	+id\$	E→T
\$0E1	+id\$	S4
\$0E1+4	id\$	S3
\$0E1+4id3	\$	T→id
\$0E1+4T5	\$	E→E+T
\$0E1	\$	Accept



Q:- $S \to AA$
$A \to aA$
$A \to b$

input: abb



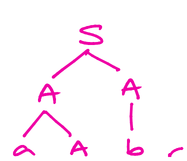
7-States

LR(0) parse table

	Action			Goto	
	a	b	\$	S	A
0	S3	S4		1	2
1			Accept		
2	S3	S4			5
3	S3	S4			6
4	A→b	A→b	A→b		
5	S→AA	S→AA	S→AA		
6	A→aA	A→aA	A→aA		

the grammar is LR(0)

Stack	Input	Actions
\$0	abb\$	S3
\$0a3	bb\$	S4
\$0a3b4	b\$	A→b
\$0a3A6	b\$	A→aA
\$0A2	b\$	S4
\$0A2b4	\$	A→b



(msQ)
$S \to AaBb$
$A \to dB$
$\to e$
b) d < e
d) a > d

Ex $S \to A$
$A \to dB$
$B \to e$

a) $a = b$, b) $d < e$
c) $e > b$ d) $a > d$

$\$0\boxed{a3\cancel{A6}}$	$b\$$	$A \to aA$ ✓
$\$0A2$	$b\$$	S4
$\$0A2\boxed{\cancel{b4}}$	$\$$	$A \to b$ ✓
$\$0\boxed{A2\cancel{A5}}$	$\$$	$S \to AA$ ✓
$\$0S1$	$\$$	Accept