LALR parser:-

In this type of parser the lookahead symbol is generated for each set of item. The table obtaining by this method are smaller in size than LR(K) parser.

In fact the state of SLR and LALR parsing are always same

Most of the programming languages uses LALR parsers.

Steps to construct LALR parsing technique

1. construction of canonical set of items along with the lookahead.
2. Build LALR parsing table
3. parsing the input string using canonical LR parsing table

Construction set of LR(1) items along with the Lookahead:-

The construction LR(1) items is same. But the only difference is that: in construction of LR parser, we have differed the two state if the second component is different but in this case we will merge the two state by merging of first and second component from both the state.

In previous example.

goto (I_0, a)

I_3: $C \to a \cdot C, a/d$
$C \to \cdot aC, a/d$
$C \to \cdot d, a/d$

goto (I_2, a)

I_6:-
$C \to a \cdot C, \$$
$C \to \cdot aC, \$$
$C \to \cdot d, \$$

I_{36}: goto (I_0, a)

$C \to a \cdot C, a/d/\$$
$C \to \cdot aC, a/d/\$$
$C \to \cdot d, a/d/\$$

Algorithm for construction of LALR parsing table

Step 1:- construct the LR(1) Set of items

Step 2:- Merge the two states I_i and I_j if the first component (i.e; production rules with dots) are matching and create a new state replacing one of the older state such as $I_{ij} = I_i \cup I_j$

Step 3:- The parsing actions are based on each item I_i.

(a) If $[A \to \alpha \cdot a\beta, b]$ is in I_i and $goto(I_i, a) = I_j$ then create an entry in the action table $action[I, a] = shift\ j$

(b) If there is a production $[A \to \alpha \cdot, a]$ in I_i then in the action table $action[I_i, a] = reduce\ by\ A \to \alpha$. Here A should not be S'

(c) If there is a production $S' \to S \cdot, \$$ in I_i then $action[i, \$] = accept$.

Step 4:- The goto part of the LR table can be filled as

The goto transitions for state i is considered for non-terminals only. If $goto(I_i, A) = I_j$ then $goto[I_i, A] = j$

Step 5:- If the parsing action conflict then the algorithm fails to produce LALR parser and grammar is not LALR(1)

All the entries not defined by rule 3 and 4 are considered to be "error".

$S \to CC$
$C \to aC$
$C \to d$

construct set of LR(1) items for LALR parser

I_0:- $S' \to \cdot S, \$$
$S \to \cdot CC, \$$
$C \to \cdot aC, a/d$
$C \to \cdot d, a/d$

I_1:- goto(I_0, S)
$S' \to S \cdot, \$$

I_2:- goto(I_0, C)
$S \to C \cdot C, \$$
$C \to \cdot aC, \$$
$C \to \cdot d, \$$

I_3:- goto(I_0, a)
$C \to a \cdot C, a/d$
$C \to \cdot aC, a/d$
$C \to \cdot d, a/d$

I_4:- goto(I_0, d)
$C \to d \cdot a/d$

I_5:- goto(I_2, C)
$S \to CC \cdot, \$$

I_6: goto(I_2, a)
$C \to a \cdot C, \$$
$C \to \cdot aC, \$$
$C \to \cdot d, \$$

I_7: goto(I_2, d)
$C \to d \cdot, \$$

I_8:- goto(I_3, C)
$C \to aC \cdot, a/d$

I_9: goto(I_6, C)
$C \to aC \cdot, \$$

I_{36}:- goto(I_0, a)
$C \to a \cdot C, a/d/\$$
$C \to \cdot aC, a/d/\$$
$C \to \cdot d, a/d/\$$

I_{47}: goto(I_0, d)
$C \to d \cdot a/d/\$$

I_{89}: goto(I_3, C)
$C \to aC \cdot a/d/\$$

we have merged two states I_3 and I_6 and made the second component as a or d or $
Similarly I_4 and I_7, I_5 & I_9 remain as it is

The set of items consists of states:

$\{I_0, I_1, I_2, I_{36}, I_{47}, I_5, I_{89}\}$

construction of LALR parsing table.

State	Action			goto	
	a	d	$	S	C
0	S36	S47		1	2
1			Acc		
2	S36	S47			5
36	S36	S47			89
47	γ_3	γ_3	γ_3		
5			γ_1		
89	γ_2	γ_2	γ_2		

[0,a] : S36
[0,d] = S47

I_{47}:-
$A \to \alpha \cdot, a.$
$C \to d \cdot, a/d/\$$

action [47, a] = reduce by $C \to d$ i.e. rule 3
action [47, d] = reduce by $C \to d$ i.e. rule 3
action [47, $] = reduce by $C \to d$ i.e. rule 3

$S' \to S \cdot, \$$ in I_1
so we will create action [1, $] = accept

parsing the i/p string using LALR parser.

Stack	input buffer	Action table	Goto table	parsing Action
$0	aadd$	action[0,a]=s36		shift
$0a36	add$	action[6,a]=s36		shift
$0a36a36	dd$	action[36,d]=s47		shift
$0a36a36d47	d$	action[47,d]=r3	[36,C]=89	reduce by C→id
$0a36a36C89	d$	action[89,d]=r2	[36,C]=89	reduce by C→ac
$0a36 ~~a36~~ C89	d$	action[89,d]=r2	[0,C]=2	reduce by C→ac
$0C2	d$	action[2,d]=s47		shift
$0C2d47	$	action[47,$]=r3	[2,C]=5	reduce by C→id
$0C2C5	$	action[5,$]=r1	[0,S]=1	reduce by S→C
$0S1	$	accept		