LR Parsers

Topics: 1) Introduction to LR parsers: Simple LR -(SLR)
- → LR(0) items, SLR parsing table.

2) More powerful parsers.
- → Canonical LR(1) items.
- → Canonical LR parsing table,
- → LALR parsing table.

LR parsers

Definition: → The LR parsers will parse the i/p string by applying rightmost derivation in reverse.

→ The LR parsers are bottom-up parsers that are capable of handling context free languages very efficiently where

* L stands for Left to right scan of the input symbol.
* R stands for Right most derivation in reverse.

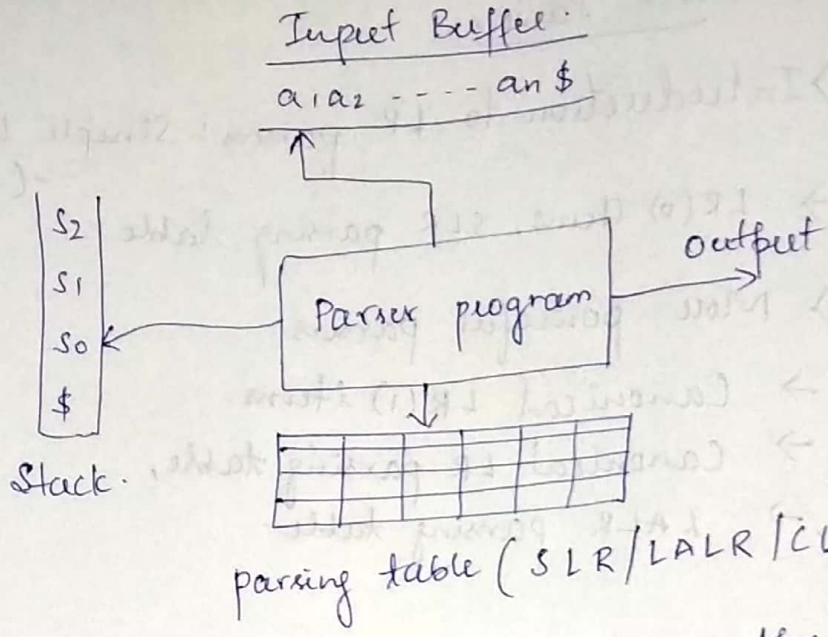
Application of LR parser:

→ used to recognize virtually all programming lang's constructs for which CFG exists.

Block Diagram: Working of LR Parser.

The working of LR parser consists of various components of the predictive parser. namely.

1) input
2) stack
3) parsing table
4) parsing program
5) output.



Input → Input buffer contains the string to be parsed and the i/p string ends with $.

Here, $ indicates end of i/p.

The i/p is read from the i/p buffer from left to right one symbol at a time.

Stack → Initially $ is placed on top of the stack indicating stack is empty. The stack contains states: $S_0\ S_1\ S_2 \dots S_n$.

S_0 → start state

S_n → top of stack.

Based on the current i/p symbol & top of the stack, parser may do the following task;

1. Based on top of stack & i/p symbol } Parser may shift a state on to the top of stack.

2) It may pop a no. of states from the stack that correspond to RHS of the production & pushes a state onto the stack corresponding to LHS of the production.

Parsing Table } → consists of two parts.

The first one → parsing action denoted by ACTION.

The second one → function GOTO.

Parser → The parser is also called driver routine. Based on top of stack & i/p symbol parser performs different actions. The actions are shift, reduce, accept and error.

Definitions of terminologies of LR parsers.

1) Item or LR(0) item.

An LR(0) item or item of a grammar G is a production of G with a (•) dot on the RHS of a production.

Eg:- $A \rightarrow \bullet BCD$.

$A \rightarrow B \bullet CD$

$A \rightarrow BC \bullet D$

$A \rightarrow BCD \bullet$

Note: For the prodn. of the form $A \rightarrow \epsilon$

item is $A \rightarrow \bullet$

An item always indicates how much of a production we have seen at a given point in the parsing process.

Using the above items called canonical LR(0) collection, we can construct a DFA w/c is used to used to make parsing decisions such as shifting and reducing.

The canonical LR(0) collection of items for a grammar can be obtained using

a) Augmented grammar
b) CLOSURE function
c) GOTO function
d) ITEMS function

Augmented Grammar.

Let $G = (V, T, P, S)$ be a grammar. An augmented grammar G' for the grammar G is defined as:

$$G' = (V', T, P', S').$$

where $V' = V \cup S'$

$P' = P \cup S' \rightarrow S$

S' = is the start symbol for the augmented grammar.

The main purpose the new starting prodn, is to indicate to the parser when it should stop parsing and announce acceptance of the input. That is, acceptance occurs whenever the parser is about to reduce using the prodn $S' \rightarrow S$.

1) Consider the grammar: $E \to E+T$, $E \to T$, $T \to T*F$, $T \to F$, $F \to (E)$, $F \to id$ — Obtain parsing table (SLR) for the grammar.

Soln:-

Given grammar

$E \to E+T$

$E \to T$

$T \to T*F$

$T \to F$

$F \to (E)$

$F \to id$

Augmented grammar

$E' \to E$

$E \to E+T$

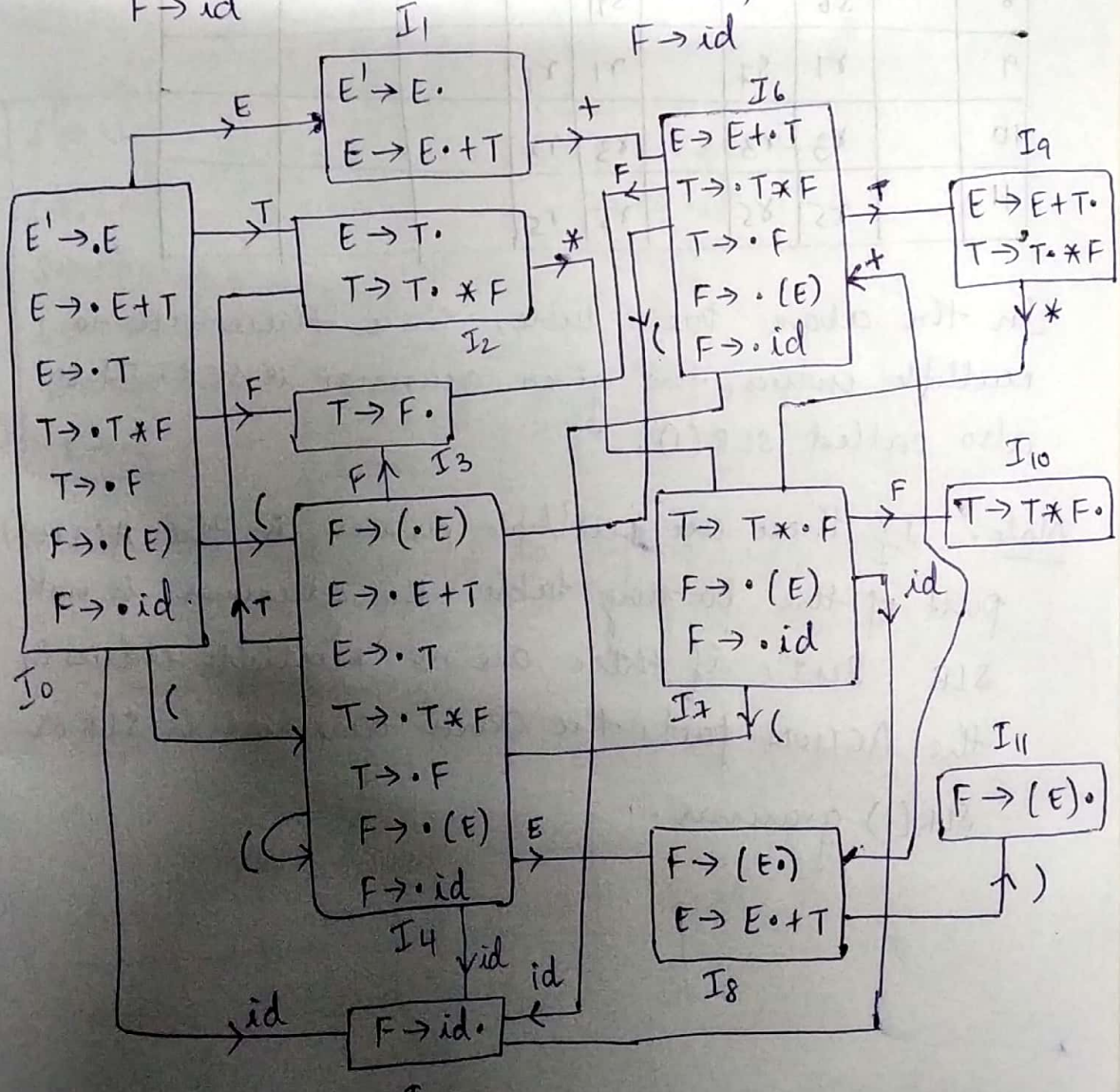
$E \to T$

$T \to T*F$

$T \to F$

$F \to (E)$

$F \to id$



	E	T	F
FIRST	(, id	(, id	(, id
FOLLOW	$, +,)	*, $, +,)	*, $, +,)

	Actions (terminals)						GOTO (variables)		
	id	+	*	(	)	$	E	T	F
0	s5			s4			1	2	3
1		s6				accept			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

In the above parse table, since there are no multiple entries, the given grammar is SLR. It is also called SLR(1).

Note: If there are multiple entries in the ACTION part of the parsing table, the grammar is not SLR. But, if there are no multiple entries in the ACTION part, the given grammar is SLR or SLR(1) grammar.

2) Consider the following grammar.

$S \to L = R \mid R$

$L \to *R \mid id$

$R \to L$

a) Obtain LR(0) items.

b) Compute FIRST & FOLLOW symbols.

c) obtain SLR parsing table.

d) check whether the given grammar is SLR or not?

Soln:

Given grammar:

$S \to L = R$

$S \to R$.

$L \to *R$

$L \to id$.

$R \to L$.

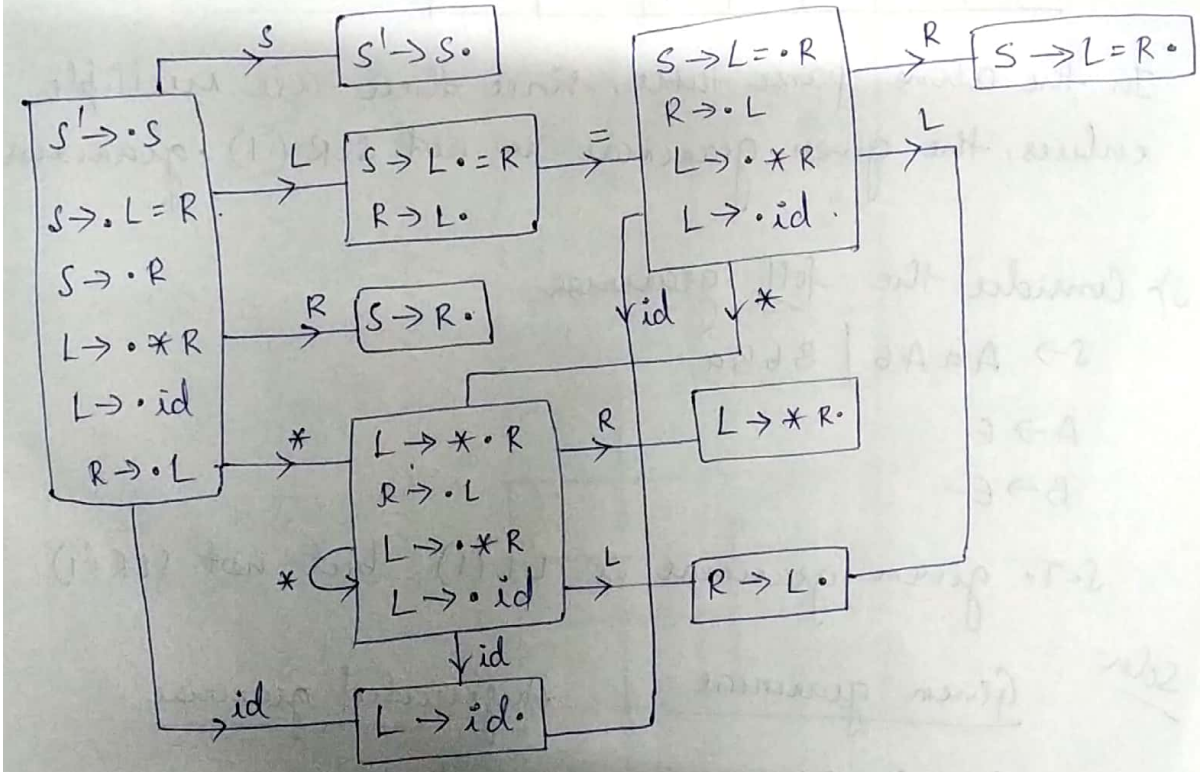
Augmented grammar.

$S' \to S$

$S \to L = R \mid R$.

$L \to *R \mid id$

$R \to L$



	S	L	R
FIRST	*, id	*, id	*, id
FOLLOW	$	$, =	$, =

	Action				Goto		
	id	=	*	$	S	L	R
0	s5		s4		1	2	3
1				accept			
2		s6, r5		r5			
3							
4	s5		s4			8	7
5		r4		r4			
6	s5		s4			8	9
7		r3		r3			
8		r5		r5			
9				r1			

In the above parse table, since there are multiple entries, the given grammar is not SLR (1) grammar.

3) Consider the foll grammar

$S \rightarrow AaAb \mid BbBa$

$A \rightarrow \epsilon$

$B \rightarrow \epsilon$

S.T. given grammar is LL(1) but not SLR(1).

Soln:

Given grammar

$S \rightarrow AaAb$

$S \rightarrow BbBa$

$A \rightarrow \epsilon$

$B \rightarrow \epsilon$

Augmented grammar

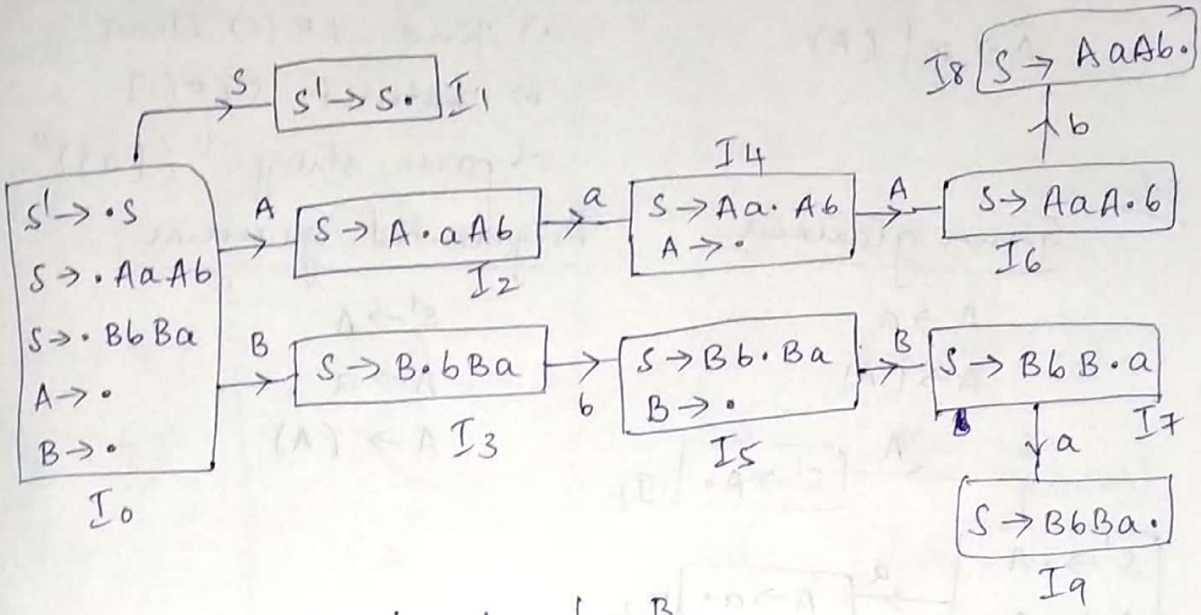
$S' \rightarrow .S$

$S \rightarrow .AaAb$

$S \rightarrow .BbBa$

$A \rightarrow .$

$B \rightarrow .$



	S	A	B
FIRST	a, b	∈	∈
FOLLOW	$	a, b	a, b.

	Action			Goto		
	a	b	$	S	A	B
0	r3, r4	r3, r4		1	2	3
1			accept			
2	S4					
3		S5				
4					6	
5						7
6		S8				
7	S9					
8			r1			
9			r2			

In above parse table, since there are multiple entries, the given grammar is not SLR(1).

4) Consider the foll grammar:

$A \to a \mid (A)$

a) find LR(0) items.
b) Construct SLR(1)
c) parse string "((a))".

Soln

Given grammar

$A \to a$

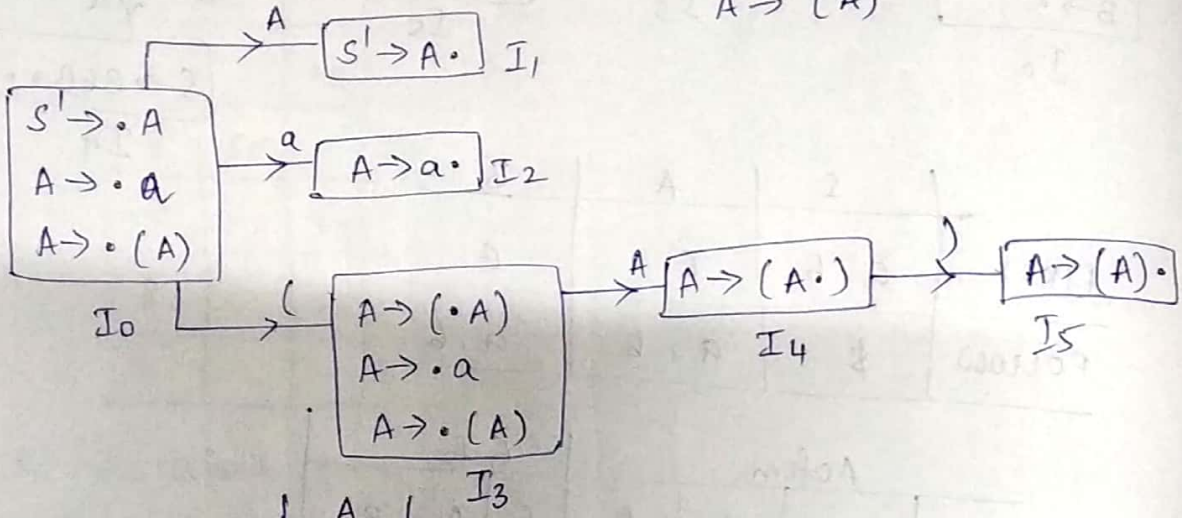
$A \to (A)$

Augmented grammar.

$S' \to A$

$A \to a$

$A \to (A)$



	A
FIRST	a, (
FOLLOW	$,)

	Action				Goto
	a	(	)	$	A
0	S2	S3			1
1				accept	
2			r1	r1	
3	S2	S3			4
4			S5		
5			r2	r2	

Sequence of moves:- "((a))"

Stack	Symbol	Input	Action
0		((a))$	shift 3 i.e.(

0 3.	(	(a))$	shift 3, (
0 3 3	((	a))$	shift 2, a
0 3 3 2	((a	))$	r1 (reduce A→a)
0 3 3	((A	))$	Goto 4
0 3 3 4.	((A	))$	shift 5,)
0 3 3 4 5	((A)	)$	r2. (A→(A))
0 3.	(A	)$	Goto 4
0 3 4.	(A	)$	shift 5,)
0 3 4 5	(A)	$	r2, A→(A)
0	A	$	Goto 1
0 1	A	$	accept (parsing is successfull).

5) Consider the foll. grammar.

$S \rightarrow SS+ \mid SS* \mid a$.

a) Obtain canonical collection of LR(0) items.

b) Construct SLR(1) parsing table. Is grammar SLR?

c) parsing string "aa*a+"

Soln

Given grammar

$S \rightarrow SS+$

$S \rightarrow SS*$

$S \rightarrow a$.

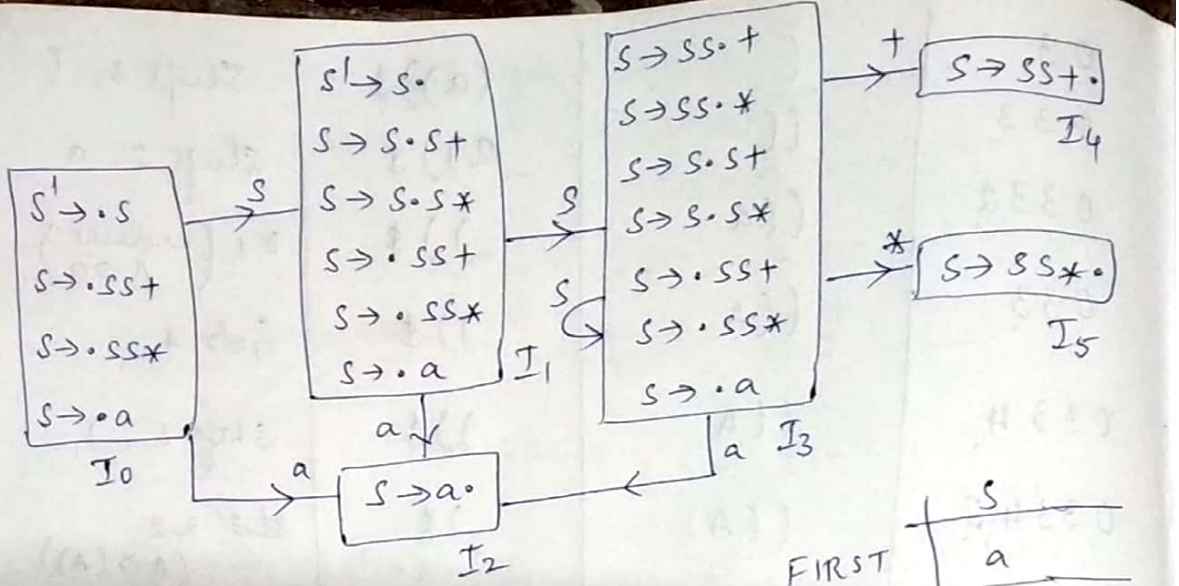
Augmented grammar

$S' \rightarrow S$

$S \rightarrow SS+$

$S \rightarrow SS*$

$S \rightarrow a$.



	S
FIRST	a
FOLLOW	*, +, a, $

	Action				Goto
	a	+	*	$	S
0	S2				1
1	S2			acc	3
2	r3	r3	r3	r3	
3	S2	S4	S5		3
4	r1	r1	r1	r1	
5	r2	r2	r2	r2	

Note: Grammar is SLR(1).

Stack	Symbol	Input	Action
0		aa*a+$	shift 2, a
0 2	a	a*a+$	reduce S→a
0 1	Sa	a*a+$	shift 2, a
0 1 2	Sa	*a+$	reduce S→a
0 1 3	SS*	*a+$	shift 5, *
0 1 3 5	S	a+$	reduce S→SS*
0 1	Sa	a+$	shift 2, a
0 1 2	SS	+$	reduce S→a
0 1 3	SS+	+$	shift 4, +
0 1 3 4	S	$	reduce S→SS+
0 1	S	$	accept

Assignment

$S \rightarrow CC$

$C \rightarrow cC$

$C \rightarrow d$.

a) Obtain LR(0)

b) Construct SLR(1) parsing table.

c) string to parse : cddddd.