LL(1) parser

CFG	First	Follow
S → L=R | abR | ε	{*, id, a, ε}	{$}
L → *R | id	{*, id}	{$, =}
R → L | ε	{*, id, ε}	{$, =}

Parsing table:

	=	a	b	*	id	$
S		S → abR		S → L=R	S → L=R	S → ε
L				L → *R	L → id	
R	R → ε			R → L	R → L	R → ε

There is no multiple entry in a single cell so, the CFG is LL(1).

Given input string = *id = id $

Stack	Input	Action
$S	*id = id $	S → L=R
$ R=L	*id = id $	L → *R
$ R=R*	*id = id $	Match & pop
$ R=R	id = id $	R → L
$ R=L	id = id $	L → id
$ R=id	id = id $	Match & pop
$ R=	= id $	Match & pop
$ R	id $	R → L
$ L	id $	L → id
$ id	id $	Match & pop
$	$	Accept

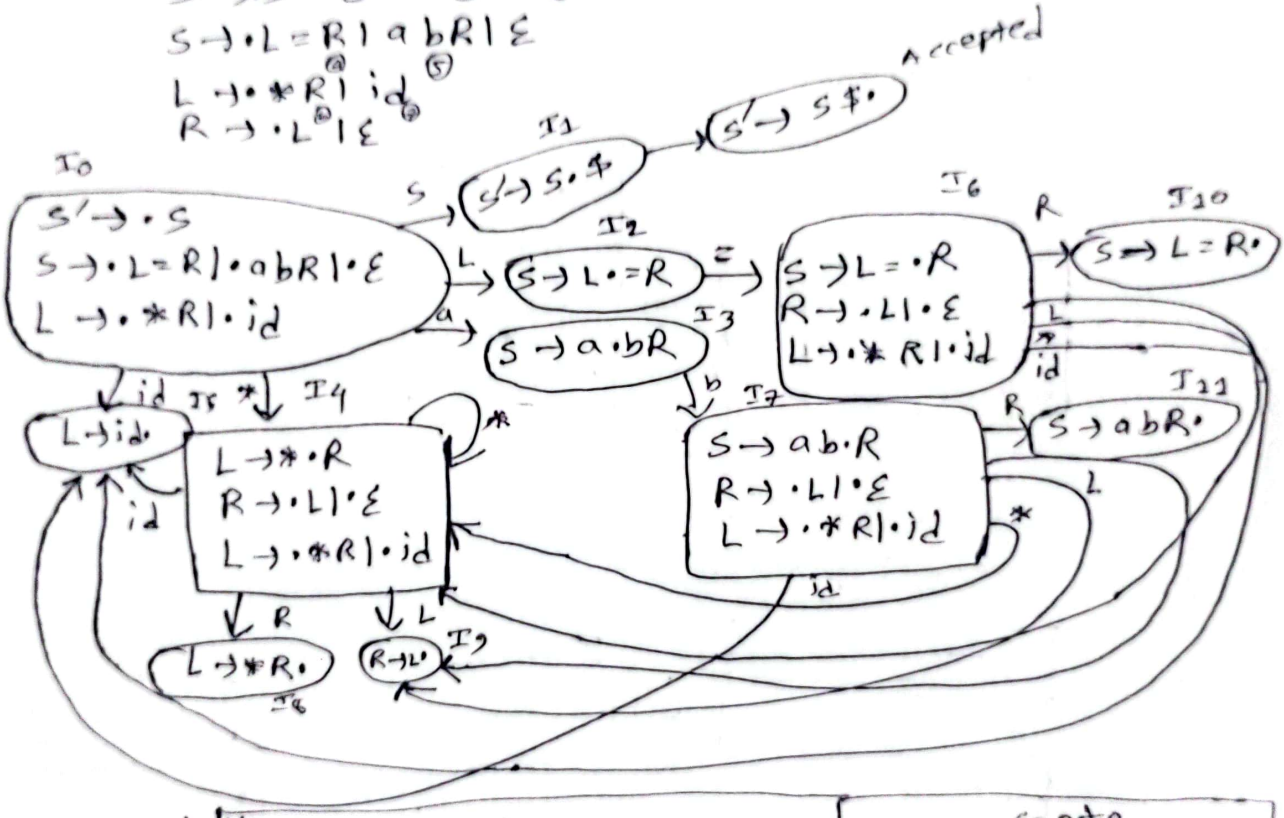
so the input string is parsed.

SLR(1) parser

CFG	First	Follow
S → L=R | abR | ε	{*, id, a, ε}	{$}
L → *R | id	{*, id}	{$, =}
R → L | ε	{*, id, ε}	{$, =}

Here,

S' → .S ① ② ③
S → .L=R | a bR | ε
L → .*R ④ | id ⑤
R → .L ⑥ | ε



Parsing table

State	=	a	b	*	id	$	S	L	R
0		S3		S4	S5	r3	1	2	
1						Accepted			
2	S6								
3			S7						
4	r7			S4	S5	r7		9	8
5	r5					r5			
6	r7			S4	S5	r7		9	10
7	r7			S4	S5	r7		9	11
8	r4					r4			
9	r6					r6			
10						r1			
11						r2			

Action | Goto

There is no conflict because there is no multiple entries in single cell. So it is SLR(1).

Input string → *id=id

Stack	Input	Action
0	*id=id$	shift 4
0*4	id=id$	shift 5
0*4id5	=id$	Reduce L→id
0*4L9	=id$	Reduce R→L
0*4R8	=id$	Reduce L→*R
0L2	=id$	shift 6
0L2=6	id$	shift 5
0L2=6id5	$	Reduce L→id
0L2=6L9	$	Reduce R→L
0L2=6R10	$	Reduce S→L=R
0S1	$	Accept

So, the input string is parsed

CLR(1) Parser

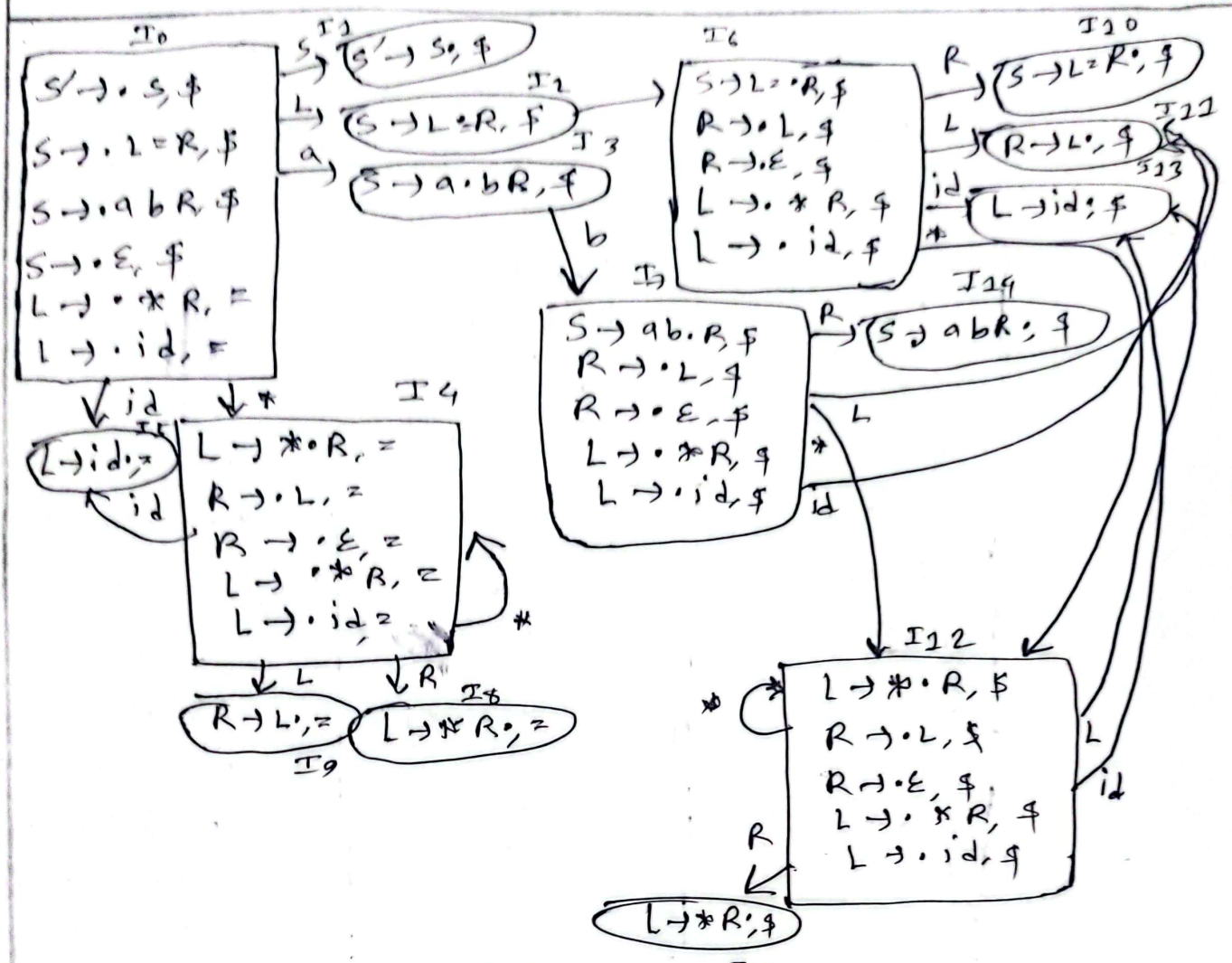
CFG	First	Follow
S→L=R|abR|ε	{*, id, a, ε}	{$}
L→*R|id	{*, id}	{$, =}
R→L|ε	{*, id, ε}	{$, =}

S'→S ① ② ③

S → L=R | abR | ε

L → *R | id ④ ⑤

R → L ⑥ | ε ⑦



Parsing table of CLR(1) Parser:

State	Action						Goto		
	=	a	b	*	id	$	S	L	R
0		S3		S4	S5	r3	1	2	
1						Accept			
2	S6								
3			S7						
4	r7			S4	S5			9	8
5	r5								
6				S12	S13	r7		11	10
7				S12	S13			11	14
8	r4								
9	r6								
10						r1			
11						r6			
12				S12	S13	r7		11	15
13						r5			
14						r2			
15						r4			

There is no conflict in the parsing table because there is no multiple entries in a single cell of the table. So it is CLR(1).

input string → *id=id,

Stack	Input	Action
0	*id=id$	shift 4
0*4	id=id$	shift 5
0*4id5	=id$	reduce L→id
0*4L9	=id$	reduce R→L
0*4R8	=id$	reduce L→*R
0L2	=id$	shift 6
0L2=6	id$	shift 13
0L2=6id13	$	reduce L→id
0L2=6L12	$	reduce R→L
0L2=6R10	$	reduce S→L=R
0S1	$	Accept

The input string is parsed with CLR(1) parser.

LALR(1) parser

S → .L=R | abR | ε
L → .*R | id
R → .L | ε

Here, we can observe (I_4, I_{12}), (I_5, I_{13}), (I_7, I_{11}), (I_8, I_{15}) has same production

so, we merge those states.

Parsing table of LALR(1):

State	Action						Goto		
	=	a	b	*	id	$	S	L	R
0		S3		S412	S513	r3	1	2	
1						Accept			
2	S6								
3			S7						
412	r7			S412	S513	r7		711	815
513	r5					r5			
6				S412	S513	r7		911	10
7				S412	S513			911	14
815	r4					r4			
911	r6					r6			
10						r1			
14						r2			

No, multiple entries, so the CFG is LALR(1)

Input string: * id = id

stack	Input	Action
0	* id = id $	Shift 412
0 * 412	id = id $	Shift 513
0 * 412 id 513	= id $	Reduce L → id
0 * 412 L 911	= id $	Reduce R → L
0 * 412 R 815	= id $	Reduce L → * R
0 L 2	= id $	Shift 6
0 L 2 = 6	id $	Shift 513
0 L 2 = 6 id 513	$	Reduce L → id
0 L 2 = 6 L 911	$	Reduce R → L
0 L 2 = 6 R 10	$	Reduce S → L = R
0 S 1	$	Accept

So, it can be said that input string * id = id is parsed with LALR(1) parser.