ASU 4.39

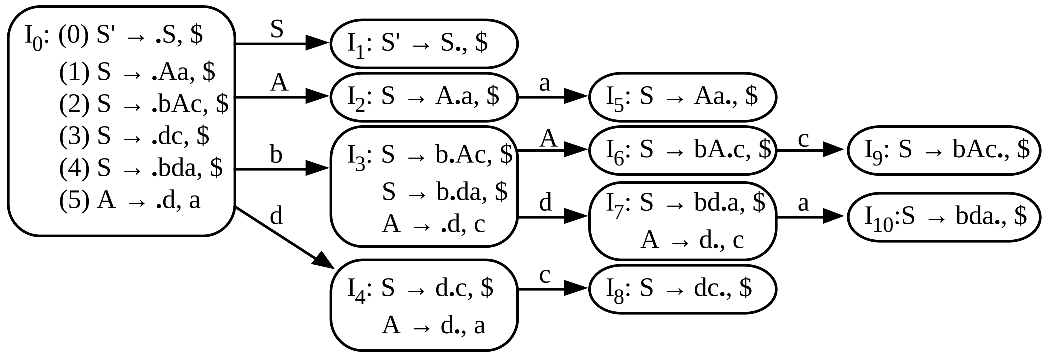
Show that the following grammar

$S \rightarrow Aa \mid bAc \mid dc \mid bda$

$A \rightarrow a$

is LALR(1) but not SLR(1).

Answer: In addition to the rules given above, one extra rule $S' \rightarrow S$ as the initial item. Following the procedures for constructing the LR(1) parser, here is the initial state and the resulting state diagram by taking closure:

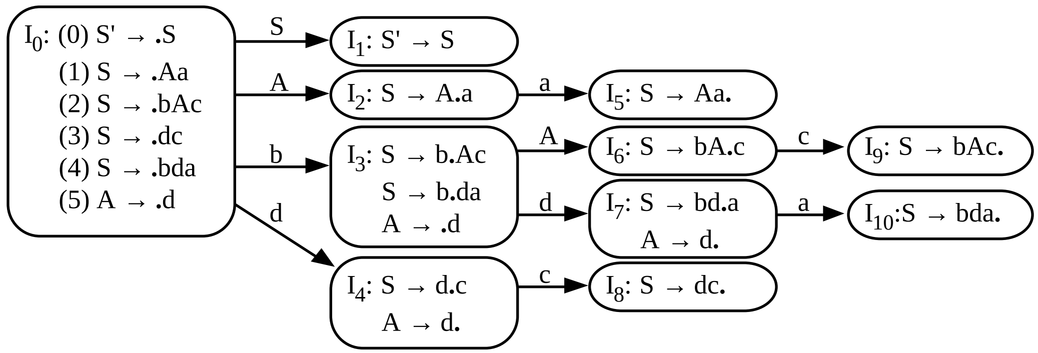


Based on the state diagram, we derive the LR(1) parsing table as follows:

State	Action					Goto	
	a	b	c	d	$	S	A
0		s3		s4		1	2
1					acc		
2	s5						
3				s7			6
4	r5		s8				
5							
6			s9				
7	s10		r5				
8					r3		
9					r2		
10					r4		

Then, the LALR(1) parsing table can be obtained by merging items with common first components, In this problem, no merging occurs. That is, the final LALR(1) parsing table is the same as the LR(1) one. Thus, the given grammar is LALR(1).

Next, following the similar procedures for taking closure, but without including the lookahead in items, we obtain the state diagram as follows:



Let's assume that the parser is in state I_7, and the next symbol is a, since a Follows(A)={a,c}, it causes a shift-reduce conflict. Same problem also happens to state I_4. Thus, the given grammar is not SLR(1).

ASU 4.40

Show that the following grammar

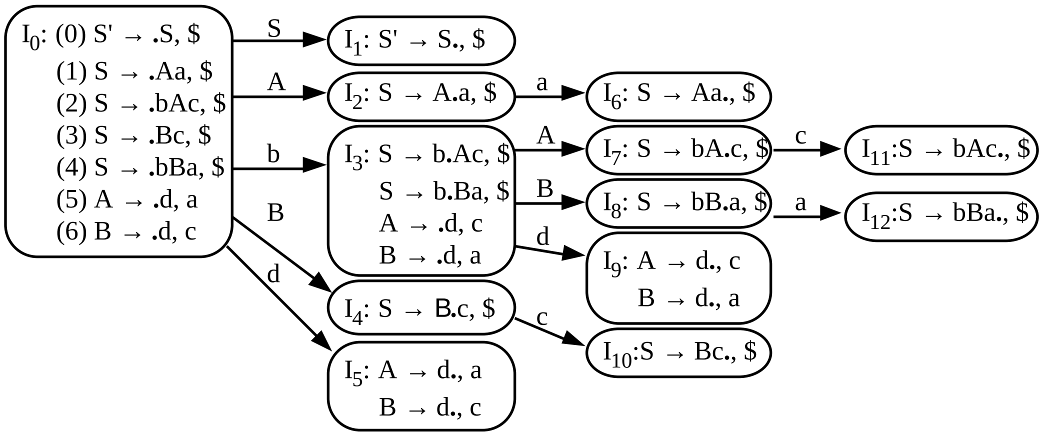
$S \rightarrow Aa \mid bAc \mid Bc \mid bBa$

$A \rightarrow d$

$B \rightarrow d$

is LR(1) but not LALR(1).

Answer: In addition to the rules given above, one extra rule $S' \rightarrow S$ as the initial item. Following the procedures for constructing the LR(1) parser, here is the resulting state diagram:



Based on the state diagram, we derive the LR(1) parsing table as follows:

State	Action					Goto		
	a	b	c	d	$	S	A	B
0		s3		s4		1	2	4
1					acc			
2	s6							
3				s9			7	8
4			s10					
5	r5		r6					
6								
7			s11					
8	s10							
9	r6		r5					
10					r3			
11					r2			
12					r4			

Since there are no mutiple actions in any entry, the given grammar is LR(1). However, when obtaining the LALR(1) parsing table by merging states, we will merge states I_5 and I_9, and the resulting state will be as follows:

$$I_{5+9}: \begin{array}{l} A \rightarrow d., \ a/c \\ B \rightarrow d., \ a/c \end{array}$$

It is basically a reduce-reduce conflict. So, the given grammar is not LALR(1).

ASU 4.44

Construct an SLR parsing table for the following grammar:

$R \rightarrow R \mid R$

$R \rightarrow RR$

$R \rightarrow R^*$

$R \rightarrow (R)$

$R \rightarrow a$

$R \rightarrow b$

Resolve the parsing action conflicts in such a way that regular expression will be parsed normally.

Answer: In addition to the rules given above, one extra rule $R' \rightarrow R$ as the initial item. Following the procedures for constructing the LR(1) parser, here is the resulting state transition:

The initial state for the SLR parser is:

I_0: (0) $R' \rightarrow .R$
(1) $R \rightarrow .R \mid R$
(2) $R \rightarrow .RR$
(3) $R \rightarrow .R^*$
(4) $R \rightarrow .(R)$
(5) $R \rightarrow .a$
(6) $R \rightarrow .b$

goto(I_0,R)= I_1: $R' \rightarrow R.$
$R \rightarrow R. \mid R$
$R \rightarrow R.R$
$R \rightarrow R.^*$
$R \rightarrow .R \mid R$
$R \rightarrow .RR$
$R \rightarrow .R^*$
$R \rightarrow .(R)$
$R \rightarrow .a$
$R \rightarrow .b$

goto(I_0,( )= I_2: $R \rightarrow (.R)$
$R \rightarrow .R \mid R$
$R \rightarrow .RR$
$R \rightarrow .R^*$
$R \rightarrow .(R)$
$R \rightarrow .a$
$R \rightarrow .b$

goto(I_0,a)= I_3: $R \rightarrow a.$

goto(I_0,b)= I_4: $R \rightarrow b.$

goto(I_1,|)= I_5: $R \rightarrow R \mid . R$
$R \rightarrow .R \mid R$
$R \rightarrow .RR$
$R \rightarrow .R^*$
$R \rightarrow .(R)$
$R \rightarrow .a$
$R \rightarrow .b$

goto(I_1,R)= I_6: $R \rightarrow RR.$
$R \rightarrow R. \mid R$
$R \rightarrow R.R$
$R \rightarrow R.^*$
$R \rightarrow .R \mid R$
$R \rightarrow .RR$
$R \rightarrow .R^*$
$R \rightarrow .(R)$
$R \rightarrow .a$
$R \rightarrow .b$

goto(I_1,*)= I_7: $R \rightarrow R^*.$

goto(I_1,( )= I_2

goto(I_1,a)= I_3

goto(I_1,b)= I_4

goto(I_2,R)= I_8: R $\rightarrow$ (R.)
R $\rightarrow$ R. | R
R $\rightarrow$ R.R
R $\rightarrow$ R.*
R $\rightarrow$ .R | R
R $\rightarrow$ .RR
R $\rightarrow$ .R*
R $\rightarrow$.(R)
R $\rightarrow$ .a
R $\rightarrow$ .b
goto(I_2,( )= I_2
goto(I_2,a)= I_3
goto(I_2,b)= I_4

goto(I_5,R)= I_9: R $\rightarrow$ R | R.
R $\rightarrow$ R. | R
R $\rightarrow$ R.R
R $\rightarrow$ R.*
R $\rightarrow$ .R | R
R $\rightarrow$ .RR
R $\rightarrow$ .R*
R $\rightarrow$.(R)
R $\rightarrow$ .a
R $\rightarrow$ .b
goto(I_5,( )= I_2
goto(I_5,a)= I_3
goto(I_5,b)= I_4

goto(I_6,|)= I_5
goto(I_6,R)= I_6
goto(I_6,*)= I_7
goto(I_6,( )= I_2
goto(I_6,a)= I_3
goto(I_6,b)= I_4

goto(I_8,))= I_{10}: R $\rightarrow$ (R).
goto(I_8,|)= I_5
goto(I_8,R)= I_6
goto(I_8,*)= I_7
goto(I_8,( )= I_2
goto(I_8,a)= I_3
goto(I_8,b)= I_4

goto(I_9,|)= I_5
goto(I_9,R)= I_6
goto(I_9,*)= I_7
goto(I_9,( )= I_2
goto(I_9,a)= I_3
goto(I_9,b)= I_4

From the grammar, we have computed Follow(R)={ |, *, (,), a, b, $ }. In the sets of items mentioned above, we can easily find shift-reduce conflicts, e.g. states I_6 and I_9, but we can use the operator precedence and associativity mentioned in Section 3.3 to resolve it. Here is the operator precedence:

( ) > * > catenate[†] > |

And all these operators are left-associative. Based on this extra information, we construct the parssing table as follows:

State	Action							Goto
	|	*	(	)	a	b	$	R
0			s2		s3	s4		1
1	s5	s7	s2		s3	s4	acc	6
2			s2		s3	s4		8
3	r5	r5	r5	r5	r5	r5	r5	
4	r6	r6	r6	r6	r6	r6	r6	
5			s2		s3	s4		9
6	r2	s7	r2	r2	r2	r2	r2	6
7	r3	r3	r3	r3	r3	r3	r3	
8	s5	s7	s2	s10	s3	s4		6
9	r1	s7	s2	r1	s3	s4	r1	6
10	r4	r4	r4	r4	r4	r4	r4	

† The catenate operator is implicit, and always exists between two consecutive R nonterminals.