CS335
Compiler Design
Indian Institute of Technology, Kanpur

Naman SIngla (200619)

Assignment II

Submission Deadline:
Feb 14, 2023, 23:55hrs

Question 1

For the following grammar, design a predictive parser and show the predictive parsing table. Perform desired processing like removing left-recursion and left-factoring on the grammar if required.

$$S \longrightarrow (L)|a$$

$$L \longrightarrow L,S|LS|b$$

Solution

To design a predictive parser, first we need to remove left-recursion from the grammar:

$$S \longrightarrow (L)|a$$
$$L \longrightarrow bL'$$
$$L' \longrightarrow ,SL'|SL'|\epsilon$$

Following are the FIRST, FOLLOW sets of Non - Terminals:

$$FOLLOW(S) = \{\$\,,\,,\,(\,,a\,,)\} \qquad FIRST(S) = \{\,(\,,a\,\}$$
$$FOLLOW(L') = \{\,)\,\} \qquad FIRST(L') = \{\,b\,\}$$
$$FOLLOW(L) = \{\,)\,\} \qquad FIRST(L) = \{\,,\,,\,(\,,a\,,\epsilon\,\}$$

Now we will create the parsing table for our predictive parser.

Non - Terminal	(	)	a	b	,	$
S	S $\longrightarrow$ (L)		S $\longrightarrow$ a			
L		L' $\longrightarrow \epsilon$		L' $\longrightarrow$ b		
L'	L' $\longrightarrow$ SL'		L' $\longrightarrow$ SL		L' $\longrightarrow$,SL'	

Question 2

Show that the following grammar is LALR(1) but not SLR(1).

$$S \longrightarrow Lp \mid qLr \mid sr \mid qsp$$
$$L \longrightarrow s$$

Solution

Let's first construct the FIRST, FOLLOW sets

$$FOLLOW(S) = \{\$\} \qquad FIRST(S) = \{s, q\}$$
$$FOLLOW(L) = \{p, r\} \qquad FIRST(L) = \{s\}$$

SLR(1) Parser :

Consider following states for our parser and S' as starting state

- I_0 =Closure([S' $\longrightarrow \cdot$ S])
- $I_1 = Goto(I_0, S)$
- $I_2 = Goto(I_0, q)$
- $I_3 = Goto(I_0, s)$
- $I_4 = Goto(I_0, L)$
- $I_5 = Goto(I_2, L)$
- $I_6 = Goto(I_2, s)$
- $I_7 = Goto(I_3, r)$
- $I_8 = Goto(I_4, p)$
- $I_9 = Goto(I_5, r)$
- $I_{10} = Goto(I_6, p)$

Following are the corresponding productions for our states.

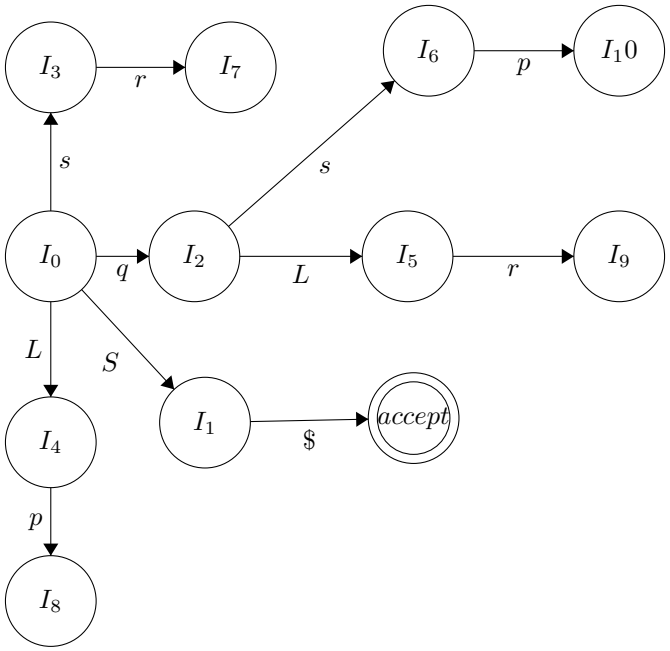
States	I_0	I_1	I_2	I_3	I_4	I_5	I_6
Productions	S' $\longrightarrow \cdot$S S $\longrightarrow \cdot$Lp S $\longrightarrow \cdot$qLr S $\longrightarrow \cdot$sr S $\longrightarrow \cdot$qsp L $\longrightarrow \cdot$s	S' $\longrightarrow$ S$\cdot$	S $\longrightarrow$ q $\cdot$ Lr S $\longrightarrow$ q $\cdot$ sp L $\longrightarrow \cdot$s	S $\longrightarrow$ s $\cdot$ r L $\longrightarrow$ s $\cdot$	S $\longrightarrow$ L $\cdot$ p	S $\longrightarrow$ qL $\cdot$ r	S $\longrightarrow$ qs $\cdot$ p L $\longrightarrow$ s $\cdot$

States	I_7	I_8	I_9	I_{10}
Productions	S $\longrightarrow$ sr $\cdot$	S $\longrightarrow$ Lp $\cdot$	S $\longrightarrow$ qLr $\cdot$	S $\longrightarrow$ qsp $\cdot$

Following is the SLR(1) parse table for the given grammar:

States	p	q	r	s	$	S	L
I_0		s2		s3		1	4
I_1					accept		
I_2				s6			5
I_3	r5		**s7** **r5**				
I_4	s8						
I_5			s9				
I_6	**s10** **r5**		r5				
I_7					r3		
I_8					r1		
I_9					r2		
I_{10}					r4		

We can easily see that there are shift/reduce conflict in some(Highlighted). Hence it is not SLR(1).
Following is the Transition diagram for the parser.



LALR(1):

Consider following states for our parser and S' as starting state

- I_0 =Closure([S' $\longrightarrow \cdot$ S])
- $I_1 = Goto(I_0, S)$
- $I_2 = Goto(I_0, L)$
- $I_3 = Goto(I_0, q)$
- $I_4 = Goto(I_0, s)$
- $I_5 = Goto(I_2, p)$
- $I_6 = Goto(I_3, L)$
- $I_7 = Goto(I_3, s)$
- $I_8 = Goto(I_4, r)$
- $I_9 = Goto(I_6, r)$
- $I_{10} = Goto(I_7, p)$

Following are the corresponding productions for our states.

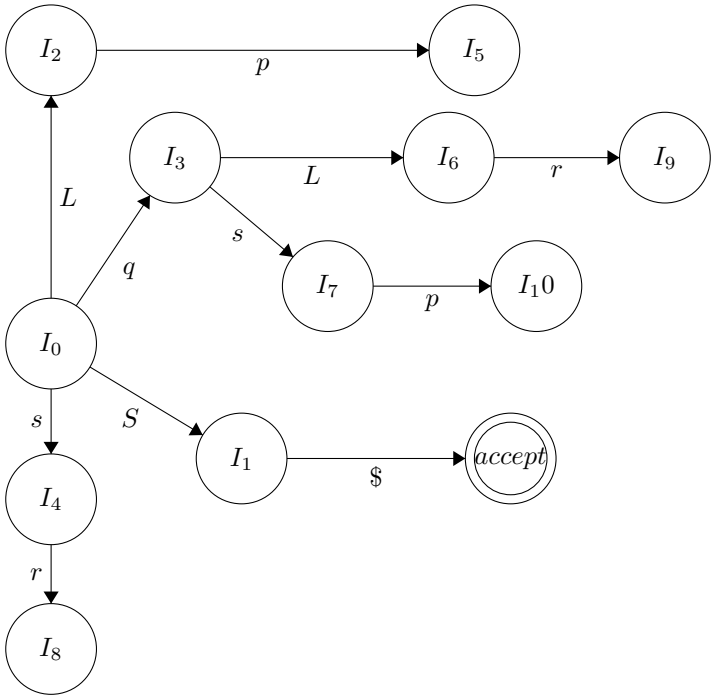
States	I_0	I_1	I_2	I_3	I_4
Productions	S' $\longrightarrow \cdot$S,$ S $\longrightarrow \cdot$qLr,$ S $\longrightarrow \cdot$sr,$ S $\longrightarrow \cdot$Lp,$ S $\longrightarrow \cdot$qsp,$ L $\longrightarrow \cdot$s,$	S' $\longrightarrow$ S$\cdot$, $	S $\longrightarrow$ L $\cdot$ p,$	S $\longrightarrow$ q $\cdot$ Lr,$ S $\longrightarrow$ q $\cdot$ sp,$ L $\longrightarrow \cdot$s,r	S $\longrightarrow$ s $\cdot$ r,$ L $\longrightarrow$ s $\cdot$,p

States	I_5	I_6	I_7	I_8	I_9	I_{10}
Productions	S $\longrightarrow$ Lp $\cdot$,$	S $\longrightarrow$ qL $\cdot$ r,$	S $\longrightarrow$ qs $\cdot$ p,$ L $\longrightarrow$ s $\cdot$,r	S $\longrightarrow$ sr $\cdot$,$	S $\longrightarrow$ qLr $\cdot$,$	S $\longrightarrow$ qsp $\cdot$,$

Following is the parse table for LALR(1) grammar:

States	p	q	r	s	$	S	L
I_0		s3		s4		1	2
I_1					accept		
I_2	s5						
I_3				s7			6
I_4	r5		s8				
I_5					r1		
I_6			s9				
I_7	s10		r5				
I_8					r3		
I_9					r2		
I_{10}					r4		

Following is the transition diagram for our LALR(1) parser:



Every cell of the LALR(1) parse table contains only one shift or reduce action. So, there is no conflict in parsing. Hence, we can say that given grammar is LALR(1).

Question 3

Construct an SLR parsing table for the following grammar. Show the canonical set of states and the transition diagram.

$$R \longrightarrow R'|'R$$
$$R \longrightarrow RR$$
$$R \longrightarrow R*$$
$$R \longrightarrow (R)$$
$$R \longrightarrow a|b$$

Note that the vertical bar in the first production is the "or" symbol (i.e., terminal), and is not a separator between alternations. Resolve the parsing action conflicts in such a way that regular expressions will be parsed normally. Include your disambiguation rules in the PDF file, and show the final parsing table.

Solution

Following are the FIRST, FOLLOW sets of R:

$$FOLLOW(R) = \{\$\,,\,*\,,\,(\,,\,)\,,\,'|'\,,\,a\,,\,b\} \qquad FIRST(R) = \{(\,,a\,,\,b\}$$

Consider following states for our parser SLR(1) and S as starting state

- I_0 =Closure([S $\longrightarrow \cdot$ R])
- $I_1 = Goto(I_0, R)|$
- $I_2 = Goto(I_0, ()|Goto(I_1, ()|Goto(I_2, ()|Goto(I_5, ()|Goto(I_6, ()|Goto(I_8, ()|Goto(I_9, ()$
- $I_3 = Goto(I_0, a)|Goto(I_1, a)|Goto(I_2, a)|Goto(I_5, a)|Goto(I_6, a)|Goto(I_8, a)|Goto(I_9, a)$
- $I_4 = Goto(I_0, b)|Goto(I_1, b)|Goto(I_2, b)|Goto(I_5, b)|Goto(I_6, b)|Goto(I_8, b)|Goto(I_9, b)$
- $I_5 = Goto(I_1,'|')|Goto(I_6,'|')|Goto(I_8,'|')|Goto(I_9,'|')$
- $I_6 = Goto(I_1, R)|Goto(I_6, R)|Goto(I_8, R)|Goto(I_9, R)$
- $I_7 = Goto(I_1, *)|Goto(I_6, *)|Goto(I_8, *)|Goto(I_9, *)$
- $I_8 = Goto(I_2, R)$
- $I_9 = Goto(I_5, R)$
- $I_{10} = Goto(I_8,))$

Following are the corresponding productions for our states.

States	I_0	I_1	I_2	I_3	I_4	I_5
Productions	S $\longrightarrow$ ·R R $\longrightarrow$ ·R'|'R R $\longrightarrow$ ·RR R $\longrightarrow$ ·R* R $\longrightarrow$ ·(R) R $\longrightarrow$ ·a R $\longrightarrow$ ·b	S $\longrightarrow$ R · R $\longrightarrow$ R ·'|'R R $\longrightarrow$ R·R R $\longrightarrow$R·* R $\longrightarrow$ ·R'|'R R $\longrightarrow$ ·RR R $\longrightarrow$ ·R* R $\longrightarrow$ ·(R) R $\longrightarrow$ ·a R $\longrightarrow$ ·b	R $\longrightarrow$ (·R) R $\longrightarrow$ ·R'|'R R $\longrightarrow$ ·RR R $\longrightarrow$ ·R* R $\longrightarrow$ ·(R) R $\longrightarrow$ ·a R $\longrightarrow$ ·b	R $\longrightarrow$ a ·	R $\longrightarrow$ b ·	R $\longrightarrow$ R'|'·R R $\longrightarrow$ ·R'|'R R $\longrightarrow$ ·RR R $\longrightarrow$ ·R* R $\longrightarrow$ ·(R) R $\longrightarrow$ ·a R $\longrightarrow$ ·b

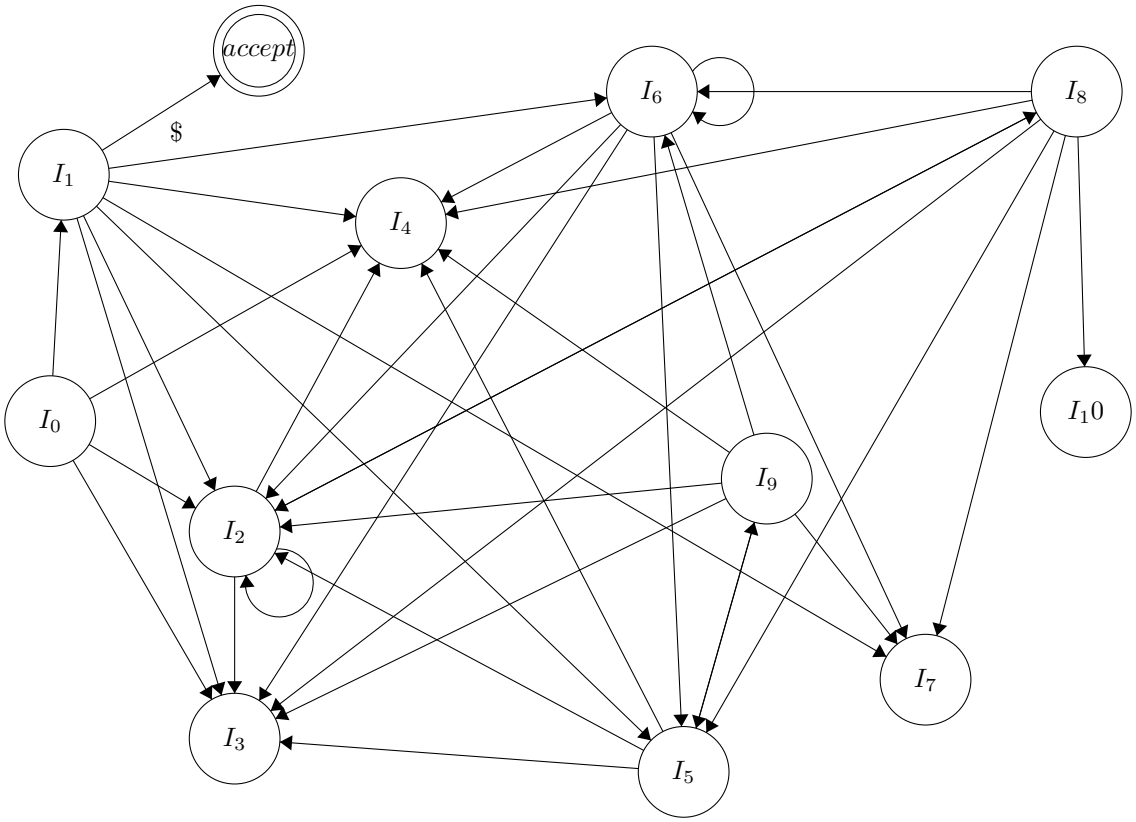
States	I_6	I_7	I_8	I_9	I_{10}
Productions	R $\longrightarrow$ R R · R $\longrightarrow$ R ·'|'R R $\longrightarrow$ R·R R $\longrightarrow$ R·* R $\longrightarrow$ ·R'|'R R $\longrightarrow$ ·RR R $\longrightarrow$ ·R* R $\longrightarrow$ ·(R) R $\longrightarrow$ ·a R $\longrightarrow$ ·b	R $\longrightarrow$ R*·	R $\longrightarrow$ (R ·) R $\longrightarrow$ R ·'|'R R $\longrightarrow$ R·R R $\longrightarrow$ R·l R $\longrightarrow$ ·R'|'R R $\longrightarrow$ ·RR R $\longrightarrow$ ·R* R $\longrightarrow$ ·(R) R $\longrightarrow$ ·a R $\longrightarrow$ ·b	R $\longrightarrow$ R'|'R· R $\longrightarrow$ R ·'|'R R $\longrightarrow$ R·R R $\longrightarrow$ R·* R $\longrightarrow$ ·R'|'R R $\longrightarrow$ ·RR R $\longrightarrow$ ·R* R $\longrightarrow$ ·(R) R $\longrightarrow$ ·a R $\longrightarrow$ ·b	R $\longrightarrow$ (R) ·

I have left some productions as they will start repeating. Let's construct the parse table for above

States	R	(	a	b	$	'|'	*	)
I_0	1	s2	s3	s4				
I_1	6	s2	s3	s4	accept	s5	s7	
I_2	8	s2	s3	s4				
I_3		r5	r5	r5	r5	r5	r5	r5
I_4		r6	r6	r6	r6	r6	r6	r6
I_5	9	s2	s3	s4				
I_6	6	**s2** **r2**	**s3** **r2**	**s4** **r2**	r2	**s5** **r2**	**s7** **r2**	r2
I_7		r3	r3	r3	r3	r3	r3	r3
I_8	6	s2	s3	s4		s5	s7	s10
I_9	6	**s2** **r1**	**s3** **r1**	**s4** **r1**	r1	**s5** **r1**	**s7** **r1**	r1
I_{10}		r4	r4	r4	r4	r4	r4	r4

As we can see there are shift-reduce conflict in the parse table. To resolve this conflict we can do following : If the current character is of lower precedence as compared to look-ahead then we can shift else reduction will be performed. This will resolve the conflict and we will get SLR(1) grammar.

Following is the automata of SLR(1) parser for the given language



Question 4

Coding

Solution

To run the solution. Run "run.sh"
To run for file say "file.txt", Replace "sample.txt" in script with "file.txt"
Tools used : Flex, Bison in C++
NOTE: You need to have flex, bison, g++, printf, cat commands installed.