[CS 335] Compiler Design : Assignment 2

Aniket Sanghi (170110)
sanghi@iitk.ac.in

Due: Feb 12, 2020

1 Solution 1

Given Grammar,

$$S \to (L) \,|\, a$$
$$L \to L, S \,|\, LS \,|\, b$$

It has left recursion on L so introducing variable to correct it gives

$$S \to (L) \,|\, a$$
$$L \to bT$$
$$T \to, ST \,|\, ST \,|\, \epsilon$$

Using LL(1) predictive parser, FIRST and FOLLOW set of each are as follows

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

Considered follow set of only T since it has ϵ in its FIRST set, then added rules corresponding to the FIRST and FOLLOW(only in case of T here) set of the variable.

Following is the LL(1) predictive parsing table for the grammar

	(	)	a	b	,	$
S	$S \to (L)$		$S \to a$			
L				$L \to bT$		
T	$T \to ST$	$T \to \epsilon$	$T \to ST$		$T \to, ST$	

Table 1: Predictive Parsing table using LL(1)

2 Solution 2

Given Grammar (After including new start symbol),

$$S' \to S$$

$$S \to Lp \,|\, qLr \,|\, sr \,|\, qsp$$

$$L \to s$$

Processing it via SLR(1) Parser gives the following canonical collection and FIRST/FOLLOW sets

$$FIRST(S') = \{s, q\} \qquad FOLLOW(S') = \{\$\}$$

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

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

$I_0 = Closure(S' \to \bullet S) = \{$
$\quad S' \to \bullet S$
$\quad S \to \bullet Lp,$
$\quad S \to \bullet qLr,$
$\quad S \to \bullet sr,$
$\quad S \to \bullet qsp,$
$\quad L \to \bullet s$
$\}$

$I_1 = GOTO(I_0, S) = \{$
$\quad S' \to S\bullet$
$\}$

$I_2 = GOTO(I_0, q) = \{$
$\quad S \to q \bullet Lr,$
$\quad S \to q \bullet sp,$
$\quad L \to \bullet s$
$\}$

$I_3 = GOTO(I_0, L) = \{$
$\quad S \to L \bullet p$
$\}$

$I_4 = GOTO(I_0, s) = \{$
$\quad S \to s \bullet r$
$\quad L \to s\bullet$
$\}$

In Item I_4, $L \to s\bullet$ (final item) will be reduced on next symbol = r since r it is in the FOLLOW set of L, also there is a shift (GOTO) on r due to $S \to s \bullet r$. Hence there is a **shift/reduce** conflict.

Hence the grammar is **NOT SLR(1)**

Processing it via LALR(1) parser gives the following canonical collection

$I_0 = Closure(S' \to \bullet S) = \{$
$\quad S' \to \bullet S, \$,$
$\quad S \to \bullet Lp, \$,$
$\quad S \to \bullet qLr, \$,$
$\quad S \to \bullet sr, \$,$
$\quad S \to \bullet qsp, \$,$
$\quad L \to \bullet s, p$
$\}$

$I_4 = GOTO(I_0, s) = \{$
$\quad S \to s \bullet r, \$$
$\quad L \to s\bullet, p$
$\}$

$I_1 = GOTO(I_0, S) = \{$
$\quad S' \to S\bullet, \$$
$\}$

$I_2 = GOTO(I_0, L) = \{$
$\quad S \to L \bullet p, \$$
$\}$

$I_3 = GOTO(I_0, q) = \{$
$\quad S \to q \bullet Lr, \$,$
$\quad S \to q \bullet sp, \$,$
$\quad L \to \bullet s, r$
$\}$

$I_5 = GOTO(I_4, r) = \{$
$\quad S \to sr\bullet, \$$
$\}$

$I_6 = GOTO(I_2, p) = \{$
$\quad S \to Lp\bullet, \$$
$\}$

$I_7 = GOTO(I_3, L) = \{$
$\quad S \to qL \bullet r, \$$
$\}$

$I_8 = GOTO(I_7, r) = \{$
$S \to qLr\bullet, \$$
$\}$

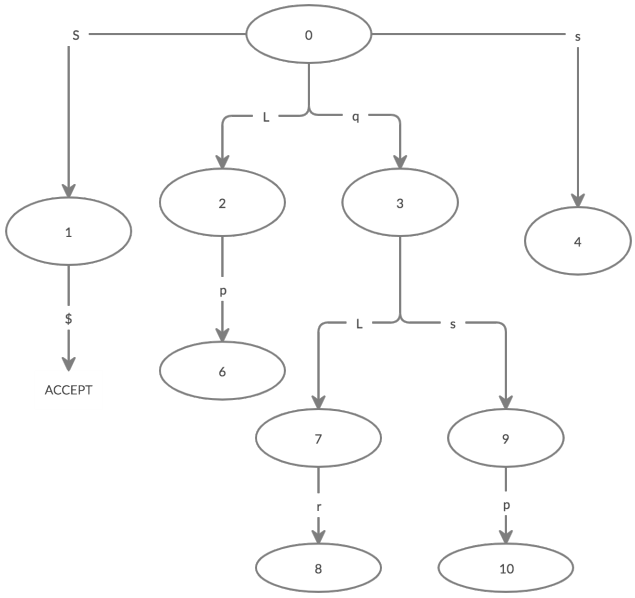
$I_9 = GOTO(I_3, s) = \{$
$S \to qs \bullet p, \$$
$L \to s\bullet, r$
$\}$

$I_{10} = GOTO(I_9, p) = \{$
$S \to qsp\bullet, \$$
$\}$

There are no two states with same LR(1) items so we don't need to merge any 2 states for LALR(1). Let's number the Production to be used in the parsing table

0. $S' \to S$
1. $S \to Lp$
2. $S \to qLr$
3. $S \to sr$
4. $S \to qsp$
5. $L \to s$

[Correction: In the Automaton there is one more state numbered 5 which has an incoming edge from state 4 on r]



	ACTION					GOTO		
	s	**q**	**p**	**r**	**$**	S'	S	L
0	s4	s3					s1	s2
1					accept			
2			s6					
3	s9							s7
4			r5	s5				
5					r3			
6					r1			
7				s8				
8					r2			
9			s10	r5				
10					r4			

Table 2: Parsing table using LALR(1)

Since there are no reduce/reduce or shift/reduce conflicts in the parsing table, the grammar is **LALR(1)**

3 Solution 3

Given Grammar after introducing new start symbol and numbering,

0. $R' \to R$
1. $R \to R|R$
2. $R \to RR$
3. $R \to R*$
4. $R \to (R)$
5. $R \to a$
6. $R \to b$

Applying SLR(1) parsing algorithm gives the following canonical structure of LR(0) items

$I_0 = Closure(R' \to \bullet R) = \{$
$R' \to \bullet R$
$R \to \bullet R|R$
$R \to \bullet RR$
$R \to \bullet R*$
$R \to \bullet(R)$
$R \to \bullet a$
$R \to \bullet b$
$\}$

$I_4 = GOTO(I_0, ( ) = \{$
$R \to (\bullet R)$
$R \to \bullet R|R$
$R \to \bullet RR$
$R \to \bullet R*$
$R \to \bullet(R)$
$R \to \bullet a$
$R \to \bullet b$
$\}$

$I_5 = GOTO(I_4, R) = \{$
$R \to (R\bullet)$
$R \to R \bullet |R$
$R \to R \bullet R$
$R \to R \bullet *$
$R \to \bullet R|R$
$R \to \bullet RR$
$R \to \bullet R*$
$R \to \bullet(R)$
$R \to \bullet a$
$R \to \bullet b$
$\}$

$I_6 = GOTO(I_5,)) = \{$
$R \to (R)\bullet$
$\}$

$I_1 = GOTO(I_0, R) = \{$
$R' \to R\bullet$
$R \to R \bullet |R$
$R \to R \bullet R$
$R \to R \bullet *$
$R \to \bullet R|R$
$R \to \bullet RR$
$R \to \bullet R*$
$R \to \bullet(R)$
$R \to \bullet a$
$R \to \bullet b$
$\}$

$I_2 = GOTO(I_0, a) = \{$
$R \to a\bullet$
$\}$

$I_3 = GOTO(I_0, b) = \{$
$R \to b\bullet$
$\}$

$I_8 = GOTO(I_5, *) = \{$
$R \to R * \bullet$
$\}$

$I_7 = GOTO(I_5, |) = \{$
$R \to R| \bullet R$
$R \to \bullet R|R$
$R \to \bullet RR$
$R \to \bullet R*$
$R \to \bullet(R)$
$R \to \bullet a$
$R \to \bullet b$
$\}$

$I_2 = GOTO(I_1, a)$
$I_3 = GOTO(I_1, b)$
$I_4 = GOTO(I_1, ( )$
$I_7 = GOTO(I_1, |)$
$I_8 = GOTO(I_1, *)$
$I_9 = GOTO(I_1, R)$

$I_4 = GOTO(I_4, ( )$
$I_2 = GOTO(I_4, a)$
$I_3 = GOTO(I_4, b)$

$I_2 = GOTO(I_5, a)$
$I_3 = GOTO(I_5, b)$
$I_4 = GOTO(I_5, ( )$

$I_2 = GOTO(I_7, a)$
$I_3 = GOTO(I_7, b)$
$I_4 = GOTO(I_7, ( )$

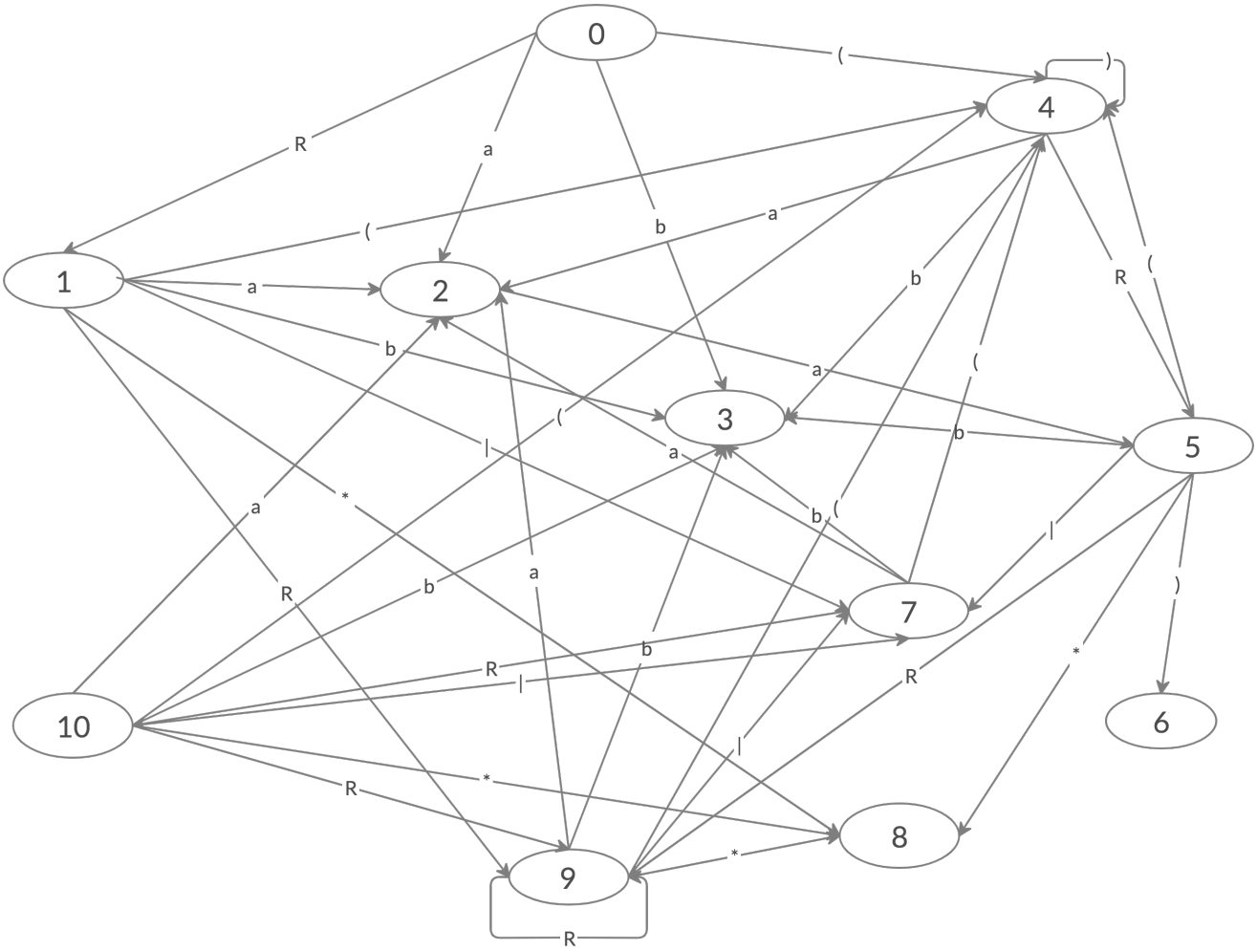
$I_2 = GOTO(I_9, a)$
$I_3 = GOTO(I_9, b)$
$I_4 = GOTO(I_9, ( )$
$I_7 = GOTO(I_9, |)$
$I_8 = GOTO(I_9, *)$
$I_9 = GOTO(I_9, R)$

$I_9 = GOTO(I_5, R) = \{$
$R \to RR\bullet$
$R \to R \bullet |R$
$R \to R \bullet R$
$R \to R \bullet *$
$R \to \bullet R|R$
$R \to \bullet RR$
$R \to \bullet R*$
$R \to \bullet(R)$
$R \to \bullet a$
$R \to \bullet b$
$\}$

$I_{10} = GOTO(I_7, R) = \{$
$R \to R|R\bullet$
$R \to R \bullet |R$
$R \to R \bullet R$
$R \to R \bullet *$
$R \to \bullet R|R$
$R \to \bullet RR$
$R \to \bullet R*$
$R \to \bullet(R)$
$R \to \bullet a$
$R \to \bullet b$
$\}$

$I_2 = GOTO(I_{10}, a)$
$I_3 = GOTO(I_{10}, b)$
$I_4 = GOTO(I_{10}, ( )$
$I_7 = GOTO(I_{10}, |)$
$I_8 = GOTO(I_{10}, *)$
$I_9 = GOTO(I_{10}, R)$

This gives us the following SLR(1) Automaton



FIRST and FOLLOW set of variables comes out to be

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

Using these we get the following parsing table.

	ACTION							GOTO
	|	*	(	)	**a**	b	$	R
0			s4		s2	s3		s1
1	s7	s8	s4		s2	s3	accept	s9
2	r5	r5	r5	r5	r5	r5	r5	
3	r6	r6	r6	r6	r6	r6	r6	
4			s4		s2	s3		s5
5	s7	s8	s4	s6	s2	s3		s9
6	r4	r4	r4	r4	r4	r4	r4	
7			s4		s2	s3		s10
8	r3	r3	r3	r3	r3	r3	r3	
9	~~s7~~/r2	s8/~~r2~~	s4/~~r2~~	r2	~~s2~~/r2	~~s3~~/r2	r2	s9
10	~~s7~~/r1	s8/~~r1~~	s4/~~r1~~	r1	s2/~~r1~~	s3/~~r1~~	r1	s9

Table 3: Parsing table using SLR(1)

We observe about 10 shift/reduce conflict in the parsing table which have been resolved by following the given rules

- $cell[9][|]$: We have RR on top of stack and the next input symbol is | so, giving concatenation (RR) a higher preference resolves the conflict (i.e. reduce in this case)
- $cell[10][|]$: We have $R|R$ on top of stack and the next input symbol is | so, considering | operation to to be left associative resolves the conflict (i.e. reduce in this case)
- $cell[9][*]$: We have RR on top of the stack and the next input symbol is $*$ so, giving $*$ a higher preference than concatenation resolves the conflict (i.e. shift in this case)
- $cell[10][*]$: We have $R|R$ on top of the stack and the next input symbol is $*$ so, giving $*$ a higher preference than | resolves the conflict (i.e. shift in this case)
- $cell[9][(]$: We have RR on top of stack and the next input symbol is (so, giving parenthesis () a higher preference resolves the conflict (i.e. shift in this case)
- $cell[10][(]$: We have $R|R$ on top of stack and the next input symbol is (so, giving parenthesis () a higher preference resolves the conflict (i.e. shift in this case)
- $cell[9][a]$: We have RR on top of the stack and the next input symbol is a so, considering concatenation operation to be left associative resolves the conflict (i.e. reduce in this case)

- $cell[10][a]$: We have $R|R$ on top of the stack and the next input symbol is a so, giving concatenation a higher preference than | resolves the conflict (i.e. shift in this case)
- $cell[9][b]$: We have RR on top of the stack and the next input symbol is b so, considering concatenation operation to be left associative resolves the conflict (i.e. reduce in this case)
- $cell[10][b]$: We have $R|R$ on top of the stack and the next input symbol is b so, giving concatenation a higher preference than | resolves the conflict (i.e. shift in this case)

Finally listing out all the disambiguate rules used (assuming it to be grammar for regex)

1. () has the highest precedence
2. RR follows parenthesis in precedence and is left associative
3. $R|R$ follows RR in precedence and is left associative

4 Solution 4

I have added the makefile to create the executable. Following are the steps to compile and execute

- make
- ./parser < inputfile > outputfile

In case the makefile doesn't work, here are the manual steps to compile and run

- bison -d parser.y
- flex lexer.l
- g++ -std=c++11 lex.yy.c parser.tab.c -o parser
- ./parser < inputfile > outputfile

Tools Used

- FLEX
- BISON