

LALR Parsing Handout 3

We use “\$” as both the bottom of stack symbol and the end of file symbol. The instantaneous description, **id**, is a string consisting of the stack, from bottom to top, followed by the current (remaining) input file starting with the next symbol, followed by the current output file. The symbols in the stack above the bottom are alternating stack states and grammar symbols, where the stack states are written as subscripts for clarity. The last symbol in the input file will be \$.

In all of the LALR parsers given below, there will be two special stack states, 0, the state of the empty stack, and 1, the state when the start symbol is just above the bottom. The stack is initially $\$_0$, and the last configuration of the stack is always $\$_0 S_1$, where S is the start symbol. We give several examples of simple LALR parsers. When we write a grammar, we include stack states as subscripts.

Example 1: Dangling else

The following LALR parser demonstrates how the “dangling else” can be resolved. Let L be the language generated by the ambiguous CF grammar below, with start symbol S .

1. $S \rightarrow a_2$
2. $S \rightarrow w_3 S_4$
3. $S \rightarrow i_5 S_6$
4. $S \rightarrow i_5 S_6 e_7 S_8$

Here are the ACTION and GOTO tables.

	a	w	i	e	$\$$	E
0	$s2$	$s3$	$s5$			1
1					halt	
2	$r1$	$r1$	$r1$	$r1$	$r1$	
3	$s2$	$s3$	$s5$			4
4				$r2$	$r2$	
5	$s2$	$s3$	$s5$			6
6				$s7$	$r3$	
7	$s2$	$s3$	$s5$			8
8				$r4$	$r4$	

Problem 1. Which entry of the ACTION table resolves the dangling else problem?

Row 6, column “e.” The grammar permits the entry $r2$, but the conventional resolution of the dangling else problem requires the entry $s7$.

We now show the action of our parser on the input string $iiwaea$.

STACK	INPUT	OUTPUT	ACTION
$\$0$	$iiwaea\$$		
$\$0i_5$	$iwaea\$$		$s5$
$\$0i_5i_5$	$waea\$$		$s5$
$\$0i_5i_5w_3$	$aea\$$		$s3$
$\$0i_5i_5w_3a_2$	$ea\$$		$s2$
$\$0i_5i_5w_3S_4$	$ea\$$	1	$r1$
$\$0i_5i_5S_6$	$ea\$$	12	$r2$
$\$0i_5i_5S_6e_7$	$a\$$	12	$s7$
$\$0i_5i_5S_6e_7a_2$	$\$$	12	$s2$
$\$0i_5i_5S_6e_7S_8$	$\$$	121	$r1$
$\$0i_5S_6$	$\$$	1214	$r4$
$\$0S_1$	$\$$	12143	$r3$
$\$0S_1$	$\$$	12143	HALT

Example 2: Allowing a List of Statements

The body of a while statement, or the scope of an if-condition or else could be a statement, but it could also be a list of statements enclosed in delimiters, such as braces, such as given in the following grammar.

1. $S \rightarrow a_2$
2. $S \rightarrow w_3S_4$
3. $S \rightarrow i_5S_6$
4. $S \rightarrow i_5S_6e_7S_8$
5. $S \rightarrow \{9L_{10}\}_{11}$
6. $L \rightarrow L_{10}S_{12}$
7. $L \rightarrow \lambda$

	a	w	i	e	$\{$	$\}$	$\$$	S	L
0	$s2$	$s3$	$s5$		$s9$				
1							HALT		
2	$r1$	$r1$	$r1$	$r1$	$r1$	$r1$	$r1$		
3	$s2$	$s3$	$s5$		$s9$			4	
4	$r2$	$r2$	$r2$	$r2$	$r2$	$r2$	$r2$		
5	$s2$	$s3$	$s5$		$s9$			6	
6	$r3$	$r3$	$r3$	$s7$	$r3$	$r3$			
7	$s2$	$s3$	$s5$		$s9$			8	
8	$r4$	$r4$	$r4$	$r4$	$r4$	$r4$	$r4$		
9	$r7$	$r7$	$r7$		$r7$	$r7$			10
10	$s2$	$s3$	$s5$		$s9$	$s11$		12	
11	$r5$	$r5$	$r5$	$r5$	$r5$	$r5$	$r5$		
12	$r6$	$r6$	$r6$	$r6$	$r6$	$r6$			

Problem 2. Fill the missing entries in rows 0 through 8.

The rest of our examples are based on algebra. We use just one identifier, x , to represent all identifiers,

just to keep the tables shorter. In Example 8, we ask how we would modify Examples 2. through 8. to allow infinitely many identifiers.

Example 3: Left Associativity of an Operator

The following grammar generates an algebraic language with one operator, subtraction, and one variable, x . We use E (for expression) as the start symbol. Subtraction is left-associative. For example, $8 - 4 - 2$ is 2, not 6.

1. $E \rightarrow x_2$
2. $E \rightarrow E -_3 E_4$

Here are the ACTION and GOTO tables.

	x	$-$	$\$$	E
0	$s2$			1
1		$s3$	halt	
2		$r1$	$r1$	
3	$s2$			4
4		$r2$	$r2$	

Problem 3. Which entry of the ACTION table guarantees that subtraction is left-associative?

Row 4, column “ $-$ ”

Example 4: Binary and Unary Minus Sign

In computer languages, $--4$ means 4, although your algebra teacher would not like it. How does an LALR parser distinguish between the two operators, and enforce the priority of the unary operator?

1. $E \rightarrow x_2$
2. $E \rightarrow E -_3 E_4$
3. $E \rightarrow -_5 E_6$

Here are the ACTION and GOTO tables.

	x	$-$	$\$$	S
0	$s2$	$s5$		1
1		$s3$	halt	
2		$r1$	$r1$	
3	$s2$	$s5$		4
4		$r2$	$r2$	
5	$s2$	$s5$		6
6		$r3$	$r3$	

Problem 4. Walk through the steps of the parser with the input string $x - -x$.

STACK	INPUT	OUTPUT	ACTION
$\$0$	$x - -x\$$		
$\$0x_2$	$- -x\$$		
$\$0E_1$	$- -x\$$	1	$r1$
$\$0E_1-3$	$-x\$$	1	
$\$0E_1-3-5$	$x\$$	1	
$\$0E_1-3-5x_2$	$\$$	1	
$\$0E_1-3-5E_4$	$\$$	11	$r1$
$\$0E_1-3E_4$	$\$$	113	$r3$
$\$0E_1$	$\$$	1132	$r2$
$\$0E_1$	$\$$	HALT	

Example 5: Right Associativity of an Operator

Exponentiation is right associative. For example, $2^{3^2} = 512$, not 64. We'll use “ $\wedge$ ” for exponentiation.

1. $E \rightarrow x_2$
2. $E \rightarrow E \wedge_3 E_4$

	x	$\wedge$	$\$$	E
0	$s2$			1
1		$s3$	halt	
2		$r1$	$r1$	
3	$s2$			4
4		$s3$	$r2$	

Problem 5. Which entry of the ACTION table guarantees that exponentiation is right-associative?

Row 4, column “ $\wedge$ ”

Example 6: Precedence of Operators

Multiplication has precedence over subtraction. For example, $7 - 3 * 2$ is 1, not 8. Consider the language generated by the CF grammar:

1. $E \rightarrow x_2$
2. $E \rightarrow E -_3 E_4$
3. $E \rightarrow E *_5 E_6$

	x	$-$	$*$	$\$$	E
0	$s2$	$s3$			1
1		$s3$	$s5$	halt	
2		$r1$	$r1$	$r1$	
3	$s2$				4
4		$r2$	$s5$	$r2$	
5	$s2$				6
6		$r3$	$r3$	$r3$	

Problem 6. Which two entries guarantee that multiplication has precedence over subtraction?

Row 4, column “ $*$ ” and row 6, column “ $-$ ”

Example 7: Parentheses

1. $E \rightarrow x_2$
2. $E \rightarrow E -_3 E_4$
3. $E \rightarrow ({}_5 E_6)_7$

	x	$-$	$($	$)$	$\$$	E
0	$s2$		$s5$			1
1		$s3$			halt	
2		$r1$		$r1$	$r1$	
3	$s2$		$s5$			4
4		$r2$		$r2$	$r2$	
5	$s2$		$s5$			6
6		$s3$		$s7$		
7		$r3$		$r3$	$r3$	

Unlike the previous examples, this grammar is unambiguous, so there is no ambiguity to resolve

Example 8: Combining Examples 3, 4, 5, 6, and 7.

Problem 7. Design an LALR parser for a grammar which has all of the above operators. Let’s throw in addition, just for fun!

The operators are addition, subtraction, multiplication, exponentiation, and negation. Negation has the highest precedence, followed by exponentiation, followed by multiplication. Addition and subtraction are of equal and lowest precedence. Addition, subtraction, and multiplication are left associative, while exponentiation is right-associative.

1. $E \rightarrow E +_2 E_3$
2. $E \rightarrow E -_4 E_5$
3. $E \rightarrow E *_6 E_7$

4. $E \rightarrow E \wedge_8 E_9$
5. $E \rightarrow -_{10} E_{11}$
6. $E \rightarrow (_{12} E_{13})_{14}$
7. $E \rightarrow x_{15}$

	x	$+$	$-$	$*$	$\wedge$	$($	$)$	$\$$	E
0	s_{15}		s_{10}			s_{12}			1
1		s_2	s_4	s_6				HALT	
2	s_{15}		s_{10}			s_{12}			3
3		r_1	r_1	s_6			r_1	r_1	
4	s_{15}		s_{10}			s_{12}			5
5		r_2	r_2	s_6	s_8		r_2	r_2	
6	s_{15}		s_{10}			s_{12}			7
7		r_3	r_3	r_3	s_8		r_3	r_3	
8	s_{15}		s_{10}			s_{12}			9
9		r_4	r_4	r_4	s_8		r_4	r_4	
10	s_{15}		s_{10}			s_{12}			11
11		r_5	r_5	s_6	s_8		r_5	r_5	
12	s_{15}		s_{10}			s_{12}			13
13		s_2	s_4	s_6	s_8		s_{14}		
14		r_6	r_6	r_6	r_6		r_6	r_6	
15		r_7	r_7	r_7	r_7		r_7	r_7	

Example 9: Generating Identifiers

Suppose an identifier must start with a letter and can contain any combination of letters and numerals. Let's say identifiers are case-insensitive. Note that the language of all identifiers is regular.

Problem 8. Modify the grammar given in Example 7 to allow for any identifiers. Your grammar should generate strings such as

x4

george

tom3-(sam52-x0a43z2)

Hint: Your grammar needs more identifiers. I did not write stack states for this grammar.

1. $E \rightarrow V$
2. $E \rightarrow E - E$
3. $E \rightarrow (E)$
4. $V \rightarrow LM$
5. $L \rightarrow a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z$
6. $M \rightarrow \lambda$
7. $M \rightarrow DM$
8. $M \rightarrow LM$
9. $D \rightarrow 0|1|2|3|4|5|6|7|8|9$