LALR Parsing Handout 2

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. In most of the grammars in this handout, we use E (for expression) as the start symbol, hence the last configuration of the stack is $\$_0 E_1$. We give several examples of simple LALR parsers. When we write a grammar, we include stack states, written as subcripts.

Example 1: 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.

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

Example 2: 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 2. Write the steps of the parser with the input string $x - -x$.

Example 3: Right Associativity of an Operator

Exponentiation is right associative. For example, $2^{3^2} = 512$, not 64. We'll use "$\wedge$" for exponentiation. This operator has higher precedence than multiplication, but lower than negation.

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

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

Problem 3. Which entry of the ACTION table guarantees that exponentiation is right-associative?

Example 4: 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. $E \to x_2$
2. $E \to E -_3 E_4$
3. $E \to E *_5 E_6$

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

Example 5: Parentheses

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

	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 6: Combining Examples 1, 2, 3, 4, and 5.

Problem 5. Design an LALR parser for a grammar which has all of the above operators, and which allows parentheses. 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. For simplicity, let x represent any variable. Here is the grammar with annotated stack states.

1. $E \to E +_2 E_3$
2. $E \to E -_4 E_5$
3. $E \to E *_6 E_7$
4. $E \to E \wedge_8 E_9$
5. $E \to -_{10} E_{11}$

6. $E \to ({}_{12}E_{13})_{14}$
7. $E \to x_{15}$

Example 7: 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, as given in the following grammar.

1. $S \to a_2$
2. $S \to w_3 S_4$
3. $S \to i_5 S_6$
4. $S \to i_5 S_6 e_7 S_8$
5. $S \to \{_9 L_{10}\}_{11}$
6. $L \to L_{10} S_{12}$
7. $L \to \lambda$

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

Problem 6. I have written the entries in rows 0 through 8, as well as those in columns "{" and "}". Fill in the others.

The rest of our examples are based on algebra. We use x to represent any identifier.