CS164 Programming Languages and Compilers **Spring 2024**

Written Assignment 2

Assigned: February 8 **Due:** February 23 at 11:59pm

Instructions: This assignment asks you to prepare written answers to questions on LL and LR parsers. Each of the questions has a short answer. You may discuss this assignment with other students and work on the problems together. However, your write-up should be your own individual work.

Please write your name, email address, and discussion section on your homework. ***Please start each question on a new page. All written assignments must be submitted as a PDF via Gradescope:*** ***`https://gradescope.com`.*** Instructions for how to submit assignments to Gradescope can be found at the following links: `https://gradescope.com/get_started#student-submission`

1. Use left-factoring and/or elimination of left recursion to convert the following grammars into LL(1) grammars. You may assume that these grammars are unambiguous. In all grammars, we will treat any symbol (such as , * +) or any word starting with a lower-case character (such as bool, int) as a terminal. We will treat any upper-case character or any word starting with upper-case letter as a non-terminal.

   (a)

$$\begin{aligned} S &\rightarrow T \mid B \\ T &\rightarrow Sa \mid \epsilon \\ B &\rightarrow BxC \mid C \\ C &\rightarrow c \mid \epsilon \end{aligned}$$

   (b)

$$L \rightarrow int \mid int\ +\ L \mid int\ +\ *\ L \mid (L)$$

   (c)

$$A \rightarrow A\ bool \mid A\ and\ bool \mid \epsilon$$

2. Consider the following grammar describing a certain sort of expression language:

$$\begin{array}{rcl} A & \rightarrow & A \ * \ B \mid C \\ B & \rightarrow & B \ + \ C \mid C \\ C & \rightarrow & (\ A\ ) \mid int \end{array}$$

The nonterminals are A, B, and C, while the terminals are $*$, $+$, (,), and int.

(a) Please eliminate left recursion.

(b) Give the First and Follow sets for each nonterminal in the grammar obtained in part (a).

(c) Using this information, construct an LL parsing table for the grammar obtained in part (a).

(d) Suppose we generated an LL parser for the grammar using the table you constructed. What would go wrong if the parser tried to parse the following input string?

$$+int * int * (int + int)$$

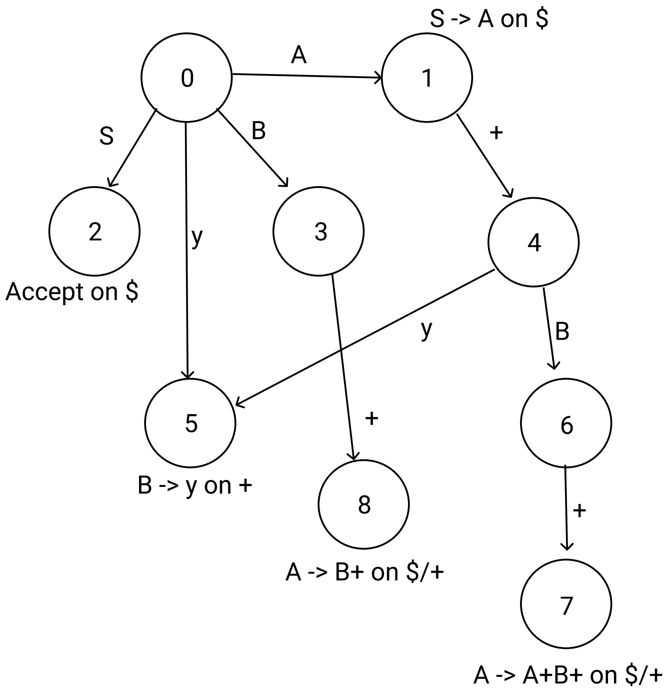
(That is, when we get an error, how much of the input string has been consumed, and what is the parser trying to do?)

3. Consider the following LR(1) grammar:

$$\begin{array}{lcl} S & \rightarrow & A \\ A & \rightarrow & A + B + \ \mid \ B + \\ B & \rightarrow & y \end{array}$$

(Each '+' is a separate token.)

and its corresponding DFA:



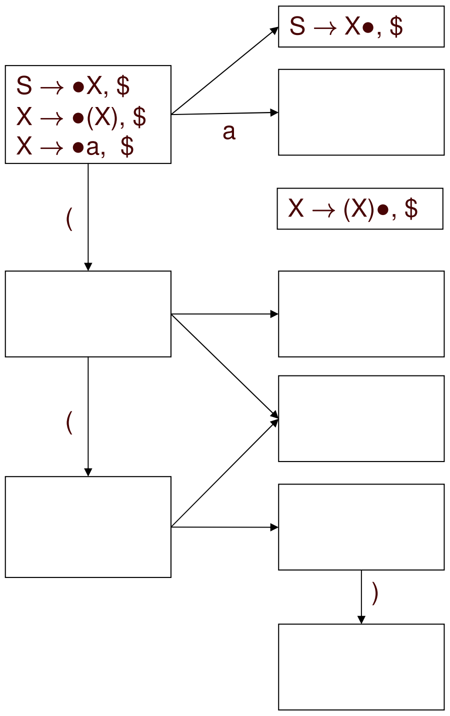
Complete the table below showing the trace of an LR(1) parser (which uses the DFA above) on the input provided. The "Stack" column must show the stack (with the top at right), the "Input" column shows the not-yet-processed input terminals, and the "Action" column must show whether the parser performs a shift action or a reduce action or accepts the input. In the case of a reduce action, please indicate which production is used.

Stack (with top at right)	Input	Action
	► y + + y + $	shift
y		

4. Consider the following grammar with start symbol S:

$$\begin{aligned} S &\rightarrow X \\ X &\rightarrow (X) \mid a \end{aligned}$$

The following figure shows a skeleton of the LR(1) parsing DFA for this grammar.



(a) Complete the DFA skeleton. You need to fill in the LR(1) items for each state of the DFA, add new transitions, and label each transition. (You should not add any new states though.) Hint: One of the states has a self-loop.

(b) Is the grammar LR(1)? Is the grammar LR(2)? Is the grammar LL(1)? Why or why not?

(c) **Optional challenge question - WILL NOT BE GRADED:** Use your DFA to parse $((((a))))$. Show the sequence of shift/reduce steps (by filling a table like that in question 3).

5. Is the following grammar ambiguous? Is it LR(1)? Explain both of your answers.

$$\begin{array}{lcl} S & \rightarrow & X\ Y\ t \\ S & \rightarrow & A\ Z\ n \\ X & \rightarrow & b \\ Y & \rightarrow & L \\ Z & \rightarrow & L \\ L & \rightarrow & a \\ A & \rightarrow & m \end{array}$$

6. In each of the following cases, give as simple a grammar as you can.

   (a) Give an LR(1) grammar that is not LL(1). Explain why your grammar is LR(1) and not LL(1).

   (b) Give an unambiguous grammar that is not LR(1). Explain why your grammar is unambiguous and not LR(1).

   Hint: In each of the above cases, there exists a grammar that generates a language with only two strings.