Written Assignment 3

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 print your name and email address on your homework! We need this information so that we can give you credit for the assignment and so that we can return it to you.

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.

   (a)

$$\begin{array}{rcl} E & \rightarrow & E + T \mid T \mid E! \\ T & \rightarrow & \mathit{int} \mid (E) \end{array}$$

   (b)

$$\begin{array}{rcl} L & \rightarrow & X \mid L, X \\ X & \rightarrow & \mathit{int} \mid \mathit{string} \mid (L) \end{array}$$

   (c)

$$\begin{array}{rcl} P & \rightarrow & P\ H\ 4\ U \mid p \\ H & \rightarrow & h \\ U & \rightarrow & u \mid u\ P \end{array}$$

2. Consider the following grammar fragment for a familiar syntax:

$$\begin{array}{rcl}
URL & \rightarrow & PROTO\ :\ /\ /\ HOST\ /\ FILE \\
PROTO & \rightarrow & http \\
PROTO & \rightarrow & ftp \\
HOST & \rightarrow & id\ I \\
I & \rightarrow & \epsilon \\
I & \rightarrow & .\ HOST \\
FILE & \rightarrow & id\ G \\
G & \rightarrow & \epsilon \\
G & \rightarrow & .\ FILE \\
G & \rightarrow & /\ FILE
\end{array}$$

The nonterminals are URL, $PROTO$, $HOST$, I, $FILE$ and G.

(a) Left-factor this grammar.

(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 it tried to parse the following input string?

$$ftp : //id.id/id./id..id$$

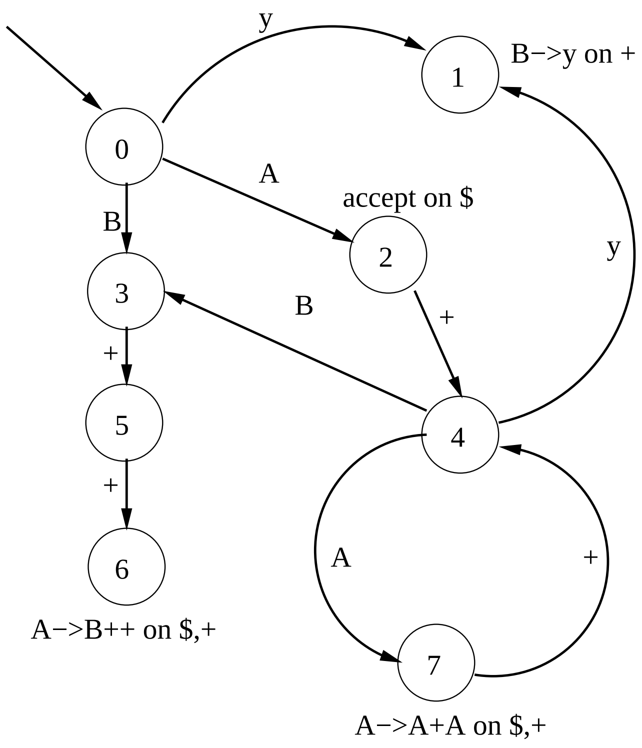
(That is, when we get an error, how much of the input string has been consumed, and what is the parser trying to do?)

3. Consider the following LR(1) grammar:

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

(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		