

## Solutions to Written Assignment 3

1. Use left-factoring and/or elimination of left recursion to convert the following grammars into LL grammars. You may assume that these grammars are unambiguous.

(a)

$$\begin{aligned} E &\rightarrow E + T \mid T \\ T &\rightarrow int \mid (E) \end{aligned}$$

**Solution: eliminate left recursion:**

$$\begin{aligned} E &\rightarrow TE' \\ E' &\rightarrow +TE' \mid \epsilon \\ T &\rightarrow int \mid (E) \end{aligned}$$

(b)

$$L \rightarrow int \mid int, L \mid (L)$$

**Solution: left-factor the grammar:**

$$\begin{aligned} L &\rightarrow int L' \mid (L) \\ L' &\rightarrow \epsilon \mid ,L \end{aligned}$$

(c)

$$A \rightarrow int \mid int + A \mid int - A \mid A - (A)$$

**Solution: there are two ways to do this. Way 1: First left-factor...**

$$\begin{aligned} A &\rightarrow int A' \mid A - (A) \\ A' &\rightarrow \epsilon \mid + A \mid - A \end{aligned}$$

**...and then eliminate left-recursion:**

$$\begin{aligned} A &\rightarrow int A' A'' \\ A' &\rightarrow \epsilon \mid + A \mid - A \\ A'' &\rightarrow -A(A'') \mid \epsilon \end{aligned}$$

**Way 2: Eliminate left-recursion first...**

$$\begin{aligned}
 A &\rightarrow \text{int } AA' \mid \text{int} + AA' \mid \text{int} - AA' \\
 A' &\rightarrow -(A)A' \mid \epsilon
 \end{aligned}$$

...and then left-factor

$$\begin{aligned}
 A &\rightarrow \text{int } A'' \\
 A' &\rightarrow -(A)A' \mid \epsilon \\
 A'' &\rightarrow AA' \mid +AA' \mid -AA'
 \end{aligned}$$

2. Consider the following LL(1) grammar describing a certain sort of nested lists:

$$\begin{aligned}
 S &\rightarrow T ; S \mid \epsilon \\
 T &\rightarrow U \bullet T \mid U \\
 U &\rightarrow x \mid y \mid [S]
 \end{aligned}$$

(a) Left-factor this grammar.

**Solution:**

$$\begin{aligned}
 S &\rightarrow T ; S \mid \epsilon \\
 T &\rightarrow UR \\
 R &\rightarrow \bullet T \mid \epsilon \\
 U &\rightarrow x \mid y \mid [S]
 \end{aligned}$$

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

**Solution:**

	FIRST	FOLLOW
S	$x, y, [, \epsilon$	$;, ]$
T	$x, y, [$	$;$
R	$\bullet, \epsilon$	$;$
U	$x, y, [$	$\bullet, ;$

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

**Solution:**

	$;$	$\bullet$	$x$	$y$	$[$	$]$	$\$$
S			$T ; S$	$T ; S$	$T ; S$	$\epsilon$	$\epsilon$
T			$UR$	$UR$	$UR$		
R	$\epsilon$	$\bullet T$					
U			$x$	$y$	$[S]$		

- (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?

$[ x ; y ] \bullet [ ;$

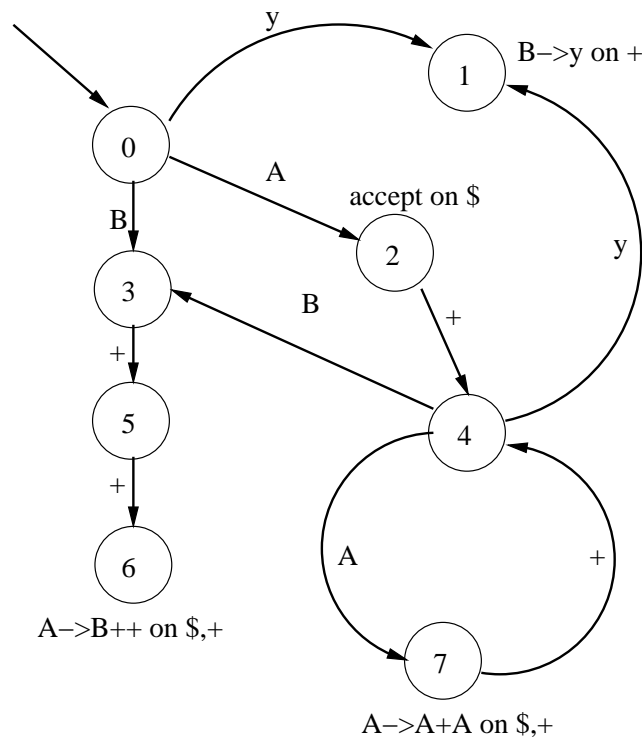
(That is, when we get an error, how much of the input string has been consumed, and what is the parser trying to do?)

**Solution:** The LL parser will successfully consume the first four tokens of input (  $[$  ,  $x$  ,  $;$  , and  $y$  ) but at that point it will try to parse an  $R$  with  $]$  as lookahead, and notice there is no corresponding table entry. (The full contents of the stack at that point are:  $R; S]R; S$ ).

3. Consider the following LR(1) grammar:

$S \rightarrow A$   
 $A \rightarrow A + A \mid B + +$  (Each '+' is a separate token.)  
 $B \rightarrow y$

and its corresponding DFA:



Complete the table like the one below showing the trace of an LR(1) parser (which uses the DFA above) on the above input. 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 say which production is used.

**Solution (with table filled in):**

Stack(with top at right)	Input	Action
	▶ y + + + y + + \$	shift
y	▶ + + + y + + \$	reduce $B \rightarrow y$
B	▶ + + + y + + \$	shift
B +	▶ + + y + + \$	shift
B + +	▶ + y + + \$	reduce $A \rightarrow B + +$
A	▶ + y + + \$	shift
A +	▶ y + + \$	shift
A + y	▶ + + \$	reduce $B \rightarrow y$
A + B	▶ + + \$	shift
A + B +	▶ + \$	shift
A + B + +	▶ \$	reduce $A \rightarrow B + +$
A + A	▶ \$	reduce $A \rightarrow A + A$
A	▶ \$	reduce $S \rightarrow A$
S	▶ \$	accept