

CS 6353 Compiler Construction, Homework #2

1. Eliminate left recursions and left factors for the following grammar.

$$\begin{aligned} S &\rightarrow A E B \\ A &\rightarrow Ax \mid Ay \mid Ba \mid a \\ E &\rightarrow = \mid \neq \\ B &\rightarrow Ab \mid b \end{aligned}$$

$$\begin{aligned} A &\rightarrow Ax \mid Ay \mid Ba \mid a \implies \\ A &\rightarrow BaA' \mid aA' \\ A' &\rightarrow xA' \mid yA' \mid \varepsilon \end{aligned}$$

The hidden left recursion: $A \rightarrow BaA'$ and $B \rightarrow Ab \implies$
 $B \rightarrow BaA'b \mid aA'b \mid b$

Remove left recursion again,
 $B \rightarrow aA'bB' \mid bB'$
 $B' \rightarrow aA'b \mid \varepsilon$

2. Consider the following grammar. Note that `id`, `+`, `[`, `]`, and `“,”` are terminals.

$$\begin{aligned} E &\rightarrow E + T \mid T \\ T &\rightarrow \text{id} \mid \text{id}[] \mid \text{id}[X] \\ X &\rightarrow E, E \mid E \end{aligned}$$

- (a) Eliminate left recursion in the grammar.

Index non-terminals as X, E and T
 $X \rightarrow E, E \mid E$
 $E \rightarrow T E'$
 $E' \rightarrow + TE' \mid \varepsilon$
 $T \rightarrow \text{id} \mid \text{id}[] \mid \text{id}[X]$

- (b) Perform left factoring for the grammar.

$$\begin{aligned} X &\rightarrow EX' \\ X' &\rightarrow , E \mid \varepsilon \\ E &\rightarrow T E' \\ E' &\rightarrow + TE' \mid \varepsilon \\ T &\rightarrow \text{id}T' \\ T' &\rightarrow \varepsilon \mid [T'' \\ T'' &\rightarrow] \mid X \end{aligned}$$

- (c) Compute the First set for all symbols in the grammar.

$$\begin{aligned} \text{First}(X) &= \text{First}(E) = \text{First}(T) = \{\text{id}\} \\ \text{First}(X') &= \{, , \varepsilon\} \\ \text{First}(E') &= \{+, \varepsilon\} \\ \text{First}(T') &= \{\varepsilon, [\} \\ \text{First}(T'') &= \{], \text{id}\} \end{aligned}$$

$\text{First}(+) = \{+\}$
 $\text{First}([\] = \{[\]\}$
 $\text{First}(\) = \{\ \}$
 $\text{First}(,) = \{,\}$

(d) Compute the Follow set for all non-terminals in the grammar.

$\text{Follow}(X) = \{\$, \]\}$
 $\text{Follow}(X') = \{\$, \]\}$
 $\text{Follow}(E) = \text{First}(X') - \{\epsilon\} \cup \text{Follow}(X') = \{., \$, \]\}$
 $\text{Follow}(E') = \text{Follow}(E)$
 $\text{Follow}(T) = \text{First}(E') - \{\epsilon\} \cup \text{Follow}(E) = \{\$, +, ., \]\}$
 $\text{Follow}(T') = \text{Follow}(T)$
 $\text{Follow}(T'') = \text{Follow}(T')$

(e) Build an LL(1) parser for the grammar.

	Id	+	[]	,	\$
X	$X \rightarrow EX'$					
X'				$X' \rightarrow \epsilon$	$X' \rightarrow , E$	$X' \rightarrow \epsilon$
E	$E \rightarrow TE'$					
E'		$E' \rightarrow + TE'$		$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow idT'$					
T'		$T' \rightarrow \epsilon$	$T' \rightarrow [T''$	$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
T''	$T'' \rightarrow X]$			$T'' \rightarrow]$		

(f) Parse the string $id + id[id+id, id[]]$. Show the stack, the input, and the action taken.

Stack	Input	Action
E \$	id + id[id+id, id[]]\$	$E \rightarrow TE'$
TE' \$	id + id[id+id, id[]]\$	$T \rightarrow idT'$
T'E' \$	+ id[id+id, id[]]\$	$T' \rightarrow \epsilon$
E' \$	+ id[id+id, id[]]\$	$E' \rightarrow + TE'$
TE' \$	id[id+id, id[]]\$	$T \rightarrow idT'$
T'E' \$	[id+id, id[]]\$	$T' \rightarrow [T''$
T''E' \$	id+id, id[]]\$	$T'' \rightarrow X]$
X]E' \$	id+id, id[]]\$	$X \rightarrow EX'$
EX']E' \$	id+id, id[]]\$	$E \rightarrow TE'$
TE'X']E' \$	id+id, id[]]\$	$T \rightarrow idT'$
T'E'X']E' \$	+id, id[]]\$	$T' \rightarrow \epsilon$
E'X']E' \$	+id, id[]]\$	$E' \rightarrow + TE'$
TE'X']E' \$	id, id[]]\$	$T \rightarrow idT'$
T'E'X']E' \$, id[]]\$	$T' \rightarrow \epsilon$
E'X']E' \$, id[]]\$	$E' \rightarrow \epsilon$
X']E' \$, id[]]\$	$X' \rightarrow , E$
E]E' \$	id[]]\$	$E \rightarrow TE'$
TE']E' \$	id[]]\$	$T \rightarrow idT'$
T'E']E' \$	[]]\$	$T' \rightarrow [T''$
T''E']E' \$] \$	$T'' \rightarrow]$
E']E' \$] \$	$E' \rightarrow \epsilon$

$]E'S$	$]S$	$]$
$E'S$	S	$E' \rightarrow \epsilon$

(g) Build the parse tree while you are parsing. Show your parse tree.

3. Consider the following grammar.

- $S \rightarrow As$
- $A \rightarrow BCA$
- $A \rightarrow BCa$
- $B \rightarrow b$
- $C \rightarrow c$

(a) Show that the grammar is not LL(1).

The rules $A \rightarrow BCA$ and $A \rightarrow BCa$, has common left factors.

(b) Is the grammar LL(k)? If so, give the k value and show the parsing table.

$\text{First}_3(C) = \{c\epsilon\}$ $\text{First}_3(B) = \{b\epsilon\}$ $\text{First}_3(A) = \{bca, bcb\}$ $\text{First}(S) = \{bca, bcb\}$
 $\text{Follow}_3(S) = \{\$ \}$ $\text{Follow}_3(A) = \{s\$ \}$ $\text{Follow}_3(B) = \{cbc, cas\}$ $\text{Follow}_3(C) = \{bca, bcb, as, \$ \}$

Yes, it is LL(3).

In the table, we only list the useful columns, not all combinations of the input.

For S and A, only bca and bcb are useful

For B: we have bε, so look at the follow set of B and we have bcb and bca also.

For C: we have bε, with the follow(C), we have cbc and cas

Thus, the table is:

	bca	bcb	cas	cbc
S	$S \rightarrow As$	$S \rightarrow As$		
A	$A \rightarrow BCa$	$A \rightarrow BCA$		
B	$B \rightarrow b$	$B \rightarrow b$		
C			$C \rightarrow c$	$C \rightarrow c$