I. Grammar Examples

1. Construct a grammar for the language

(i) $\{a^n c b^n \mid n > 0\}$

(ii) $\{a^n b^n c^m \mid n \geq 0, m > 0\}$

Answers :

$$
\begin{aligned}
S &\rightarrow a\ A\ b \\
A &\rightarrow a\ A\ b \\
&\quad \mid\ c
\end{aligned}
$$

$$
\begin{aligned}
S &\rightarrow S\ c \\
&\quad \mid\ A \\
A &\rightarrow a\ A\ b \\
&\quad \mid\ a\ b
\end{aligned}
$$

2. Consider the following grammar for generating expressions involving addition (+), subtraction (-), multiplication (*), and division (/). The symbol "*i* " represents a variable name.

$F \rightarrow i$
$F \rightarrow (E)$
$T \rightarrow F$
$T \rightarrow T * F$
$T \rightarrow T / F$
$E \rightarrow T$
$E \rightarrow E + T$
$E \rightarrow E - T$

Give the derivations for the following expressions:

$i + i$ $\qquad$ $i - i / i$ $\qquad$ $i * (i + i)$ $\qquad$ $i * i + i$

3. Show that the following grammar is ambiguous.

s → IF bexpr THEN s ELSE s
  | IF bexpr THEN s

Re-write this grammar so as to remove the ambiguity

Homework for Grammar :

1. Construct a context-free grammar for a number. Give a derivation and the corresponding syntax tree for each of the following:

a. 251 b. –61 c. –72.25 d. 1.73

II. Left Recursion Elimination

1. consider the following grammar with left-recusrion

$$
\begin{aligned}
E &\rightarrow E + T \\
&\mid T \\
T &\rightarrow T * P \\
&\mid P \\
P &\rightarrow (E) \\
&\mid V
\end{aligned}
$$

The left recursion free grammar is :

	<u>Transformation rules</u>
$E \rightarrow T$	$A_i \rightarrow \beta_1$
$\mid T\ E'$	$\mid \beta_1\ A_i'$
$E' \rightarrow + T$	$A_i' \rightarrow \alpha_1$
$\mid + T\ E'$	$\mid \alpha_1\ A_i'$
$T \rightarrow P$	
$\mid P\ T'$	
$T' \rightarrow * P$	
$\mid * P\ T'$	
$P \rightarrow (E)$	
$\mid V$	

III. LL(1) Parsing, First and Follow Examples

Example 1 : Computing First and Follow Sets for the given grammar

1.	$S \rightarrow A\ B\ e$	$FIRST_1(ABe) = \{ d, a, c \}$
2.	$A \rightarrow d\ B$	$FIRST_1(dB) = \{ d \}$
3.	$\mid a\ S$	$FIRST_1(aS) = \{ a \}$
4.	$\mid c$	$FIRST_1(c) = \{ c \}$

5.	$B \rightarrow A\ S$	$FIRST_1(AS) = \{ d, a, c \}$
6.	$\mid b$	$FIRST_1(b) = \{ b \}$

(A Little Revision of Concepts and Definitions)

Definition : A context free grammar that has no empty right hand sides and no left recursion is an LL(1) grammar if for all productions of the form

$$
\begin{aligned}
A &\rightarrow \alpha_1 \\
&\mid \alpha_2 \\
&\mid \ldots
\end{aligned}
$$

the sets $FIRST_1(\alpha_i) \cap FIRST_1(\alpha_j) = \phi$ for all i, j, $i \neq j$

The LL(1) parse table is defined as

$$
M(A, a) = \begin{cases} \mathbf{i, \alpha} \text{ if } a \in FIRST_1(\alpha) \text{ and } A \rightarrow \alpha \in P_i \\ \text{error otherwise} \end{cases}
$$

Example 2 : Compute the LL(1) parse table for the grammar shown above :

Answer :

	a	b	c	d	e
S	1, ABe	-	1, ABe	1, ABe	-
A	3, aS	-	4, c	2, dB	-
B	5, AS	6, b	5, AS	5, AS	-

Example 3 : Parse the following input string : **(adbbeccbee, S$, λ).**

(The S$ indicates the initial stack state and 3rd element of the triplet is the production number used.)

Answer :

The parse sequence is

(adbbeccbee, S$, λ)

(adbbeccbbee, ABe$, 1)

(adbbeccbbee, aSBe$, 3)

(dbbeccbee, Sbe$, -)

(dbbeccbee, ABeBe$, 1)

(dbbeccbee, dBBeBe$, 2)

(bbeccbee, BBeBe$, -)

(bbeccbee, bBeBe$, 6)

(beccbee, BeBe$, -)

(beccbee, beBe$, 6)

(eccbee, eBe$, -)

(ccbee, Be$, -)

(ccbee, ASe$, 5)

(ccbee, cSe$, 4)

(cbee, Se$, -)

(cbee, ABee$, 1)

(cbee, cBee$, 4)

(bee, Bee$, -)

(bee, bee$, 6)

(ee, ee$, -)

(e, e$, -)

(λ, $, -)

The parse sequence is 1, 3, 1, 2, 6, 6, 5, 4, 1, 4, 6

LL(1) Grammar with Empty RHS

Definition : $\text{FOLLOW}_1(A) = \{ a \mid S \Rightarrow^* \alpha A\gamma, a \in V_t, \text{ and } a \in \text{FIRST}_1(\gamma) \}$

Definition : A context free grammar is LL(1) if it is not left recursive and for each pair of productions of the form

$$\begin{aligned} A &\rightarrow \alpha \\ &\mid \beta \end{aligned}$$

$\text{FIRST}_1(\alpha\text{FOLLOW}_1(A)) \cap \text{FIRST}_1(\beta\text{FOLLOW}_1(A)) = \phi$

Note : If neither α nor β is λ, then we have

$\text{FIRST}_1(\alpha) \cap \text{FIRST}_1(\beta) = \phi$

If β is λ, then we have

$\text{FIRST}_1(\alpha) \cap \text{FOLLOW}_1(A) = \phi$

Because of λ is a RHS of some production, we first augment the grammar by adding a production of the form

$$S' \rightarrow S \ \#$$

where S' is a new non-terminal symbol and # is a new terminal symbol. Also, we will append a # to the end of any input string that we parse.

To compute the $\text{FOLLOW}_1(A)$, we apply the following rules until nothing can be added to the FOLLOW set.

1. Place # in $\text{FOLLOW}_1(S')$, where S' is the start symbol of the augmented grammar.
2. If $A \rightarrow \alpha B\beta \in P$, then everything in $\text{FIRST}_1(\beta)$ except for λ is placed in $\text{FOLLOW}_1(\beta)$.
3. If $A \rightarrow \alpha B \in P$ or $A \rightarrow \alpha B\beta \in P$ where $\lambda \in \text{FIRST}_1(\beta)$, then everything in $\text{FOLLOW}_1(A)$ is in $\text{FOLLOW}_1(B)$.

Example 4 : Consider the following grammar

0. $S' \rightarrow E \ \#$
1. $E \rightarrow T \ E'$
2. $E' \rightarrow + \ T \ E'$
3. $\quad \mid \lambda$
4. $T \rightarrow F \ T'$
5. $T' \rightarrow * \ F \ T'$
6. $\quad \mid \lambda$
7. $F \rightarrow (\ E \ )$
8. $\quad \mid V$

Compute the FOLLOW sets of S', E, E', T,T' and F

Answer :

$FOLLOW_1(S') = \{ \}$
$FOLLOW_1(E) = \{), \# \}$
$FOLLOW_1(E') = \{), \# \}$
$FOLLOW_1(T) = \{ +,), \# \}$
$FOLLOW_1(T') = \{ +,), \# \}$
$FOLLOW_1(F) = \{ +, *,), \# \}$