3. CONTEXT FREE GRAMMAR

Based on the Chomsky classification, context free grammar can be presented as (type 2)

$$A \to \beta,$$

where $A \in V_N, \beta \in (V_N \cup V_T)^*$

The context free grammar is a real model for representation the program languages. The grammars are not always optimized that means the grammar may consist of some extra symbols (non-terminal, epsilon productions) and having the extra symbols, unnecessary increase the length of grammar. Simplification of grammar means reduction of grammar by removing useless symbols as: ε – productions, unit productions, inaccessible symbols, non-productive symbols.

One of the simplest and most useful simplified forms of context free grammar is called **Chomsky Normal Form**. Another normal form usually used in algebraic specifications is the **Greibach Normal Form.**

3.1Elimination of ε – productions

If a context free grammar include ε productions as

$$A \to \varepsilon,$$

where $A \in V_N$.

In this case the context free grammar can be transformed to the equivalent grammar without ε productions.

Algorithm for determination the Nε set:

Step I. $N\varepsilon=\{A|A \to \varepsilon\}$, productions from P.

Step II. For all productions $B \to \alpha$, for which $\alpha \in V_N$ and $\alpha \in N_\varepsilon^*$ we have $N\varepsilon = N\varepsilon \cup \{B\}$.

Step III. The 2nd step is repeating until there are some changes in the set $N\varepsilon$.

Step IV. The algorithm is stopped.

Algorithm for removing the ε - productions

It is given context free grammar $G=(V_N, V_T, P, S)$:

Step I. It is determinate the set N_ε

$$P=\{A \mid A \to \varepsilon \in P\}$$

Step II. For all productions by type $A \to \alpha_1\beta_1\ \alpha_2\beta_2 \ldots \alpha_n\beta_n \in P$, where α_i is a some set, $\alpha_i \in ((V_N \cup V_T) \setminus N_\varepsilon)^*$, $\beta_i \in N_\varepsilon$ it is added to P' all productions that were obtained by the following way $A \to \alpha_1 x_1 \alpha_2 x_2 \ldots \alpha_n x_n \in P$, where $x_i = \beta_{i,}\ x_i = \varepsilon$

Step III. It is obtained $G'=(V_N, V_T, P', S)$.

Example:

$G = (V_N, V_T, S, P)$:

$V_N = \{ S, A, B\}$;

$V_T = \{a, b\}$,

$P = \{ S \to ACD$

$A \to a$

$B \to \varepsilon$

$C \to ED \mid \varepsilon$

$D \to BC \mid b$

$E \to b$

$\}$

According to the exposed algorithm:

$$N\varepsilon=\{B,C, D\}$$

Removing these productions there are obtained:

$P' = \{ S \to ACD|AD|AC|A$

$A \to a$

$C \to ED|E$

$D \to BC \mid b|B|C$

$E \to b$

$\}$

Practical Tasks

Remove the ε – productions for the given grammars:

1. $G = (V_N, V_T, S, P)$:
   $V_N = \{ S, A \}$;
   $V_T = \{a, z\}$;
   $P = \{ S \rightarrow AzA$
   $A \rightarrow a|\varepsilon \}$
2. $G = (V_N, V_T, S, P)$:
   $V_N = \{ S, A, B, C\}$;
   $V_T = \{0, 1\}$;
   $P = \{ S \rightarrow S0$
   $S \rightarrow 1$
   $S \rightarrow AB$
   $B \rightarrow AC1$
   $A \rightarrow \varepsilon$
   $C \rightarrow \varepsilon \}$
3. $G = (V_N, V_T, S, P)$:
   $V_N = \{ S, A, B, C\}$;
   $V_T = \{a, b, c\}$;
   $P = \{ S \rightarrow ABAC$
   $A \rightarrow aA|\varepsilon$
   $B \rightarrow bB|\varepsilon$
   $C \rightarrow c \}$

3.2 Elimination of the Unit Productions

Production in form of $A \rightarrow B$, where $A, B \in V_N$ is called unit - production.

Algorithm for removing unit productions:

Step I. For all productions $A \rightarrow B$, it is added the new production $A \rightarrow x$, where $B \rightarrow x$ and $x \in (V_N \cup V_T)$.

Step II. Production $A \rightarrow B$ is removed.

Step III. The 2nd step is repeated until all unit productions will be removed.

Example:

$G = (V_N, V_T, S, P)$:
$V_N = \{ S, A, B, C, D, E\}$;
$V_T = \{a, b\}$,
$P = \{ S \to ACD|AD|AC|A$
$A \to a$
$C \to ED|E$
$D \to BC \mid b|B|C$
$E \to b$
$B \to a$
$\}$

Solution:

The unit productions from P are $S \to A$, $C \to E$, $D \to B$, $D \to C$.
After elimination of this production therea are obtained:
$P' = \{ S \to ACD|AD|AC|a$
$A \to a$
$C \to ED|b$
$D \to BC \mid b|a|ED|b$
$E \to b$
$B \to a$
$\}$

Practical Tasks

Remove the unit productions for the given grammars:

1. $G = (V_N, V_T, S, P)$:
   $V_N = \{ S, A, B, C, D, E \}$;
   $V_T = \{a, b\}$;
   $P = \{ S \to AB$
   $A \to a$

$B \rightarrow C|b$
$C \rightarrow D$
$D \rightarrow E$
$E \rightarrow a$
}

2. $G = (V_N, V_T, S, P)$:
$V_N = \{ S, A, B\}$;
$V_T = \{a, b\}$;
$P = \{ S \rightarrow A|bb$
$A \rightarrow B|b$
$B \rightarrow S|a$
}

3. $G = (V_N, V_T, S, P)$:
$V_N = \{ S, T, F \}$;
$V_T = \{+, *, (,), a\}$;
$P = \{ S \rightarrow S+T|T$
$T \rightarrow T*F|F$
$F \rightarrow (S)|a$
}

3.3 Elimination of the Inaccessible Symbols

Symbol $x \in (V_N \cup V_T)$ is called inaccessible, if it doesn't exist $S \rightarrow \alpha_1 x \alpha_2$, namely x doesn't appear in any deviation from the start symbol.

Algorithm for removing inaccessible symbols

Step I. It is given A_c set of accessible symbols. From start

$$A_c = \{S\}.$$

Step II. For all non-terminal symbols $\beta \in A_c$ and all productions $\beta \rightarrow x_1, x_2, x_3 \ldots x_n$ the set A_c is changed

$$A_c = \{A_c \cup \{ x_1, x_2, x_3 \ldots x_n \}\}$$

Step III. If at the 2nd step was some changes in A_c when the 2nd step is repeating, otherwise move to step IV.

Step IV. It is build the set of inaccessible symbols

$$I=(V_N \cup V_T) \setminus A_c.$$

From productions P, there are removed all productions that contain at least one inaccessible symbol.

Example:

$G = (V_N, V_T, S, P)$:

$V_N = \{ S, A, B, C, D, E\}$;

$V_T = \{a, b\}$,

$P = \{ S \to AC$

$A \to a$

$B \to b$

$C \to Ea$

$D \to BC \mid b$

$E \to b$

$\}$

Solution:

Step I. $A_c = \{S\}$.

Step II. $A_c = A_c \cup \{A, C, D\}=\{S, A,C\}$.

Step III. $A_c = A_c \cup \{a, b, C, E\}=\{S, A,C, E, a, b\}$.

Step IV. $I=(V_N \cup V_T) \setminus A_c=(\{ S, A, B, C, D, E \} \cup \{a, b\}) \setminus \{ S, A,C, E, a, b \}= \{D\}$.

In this case the inaccesible symbol is D and after elimination of this production therea are obtained:

$V_N = \{ S, A, B, C, E \}$, $V_T = \{a, b\}$,

$P'' = \{ S \to AC$

$A \to a$

$B \to b$

$C \to Ea$

$E \to b$

$\}$

Practical Tasks

Remove the inaccessible symbols for the given grammars:

1. $G = (V_N, V_T, S, P)$:
   $V_N = \{ S, A, B, C\}$;
   $V_T = \{a, b\}$;
   $P = \{ S \to A|bb$
   $A \to B|b$
   $B \to S|a$
   $C \to a|b$

2. $G = (V_N, V_T, S, P)$:
   $V_N = \{ S, A, B, C, D, E \}$;
   $V_T = \{a, b\}$;
   $P = \{ S \to AB$
   $A \to a$
   $B \to C|b$
   $C \to ab$
   $D \to E$
   $E \to a$
   $\}$

3.3 Elimination of the Non-Productive Symbols

Non-terminal symbols $A \in V_N$ are called non-productive, if it doesn't exist $A \to y$, $y \in V_T^*$.

Algorithm for removing non-productive sysmbols

Step I. From start $P_r = \varnothing$.
Step II.
a) For all productions $A \to \alpha$, where $\alpha \in V_T^*$ we change the set P_r:

$$P_r = P_r \cup \{A\}$$

b) For all productions $B \to \beta$, where $\beta \in (V_T \cup P_r)$, we change P_r:

$$P_r = P_r \cup \{B\}$$

Step II. For all time there are some changes in the set P_r 2nd step is repeating.

Example:

$G = (V_N, V_T, S, P)$:

$V_N = \{ S, A, B\}$;

$V_T = \{a, b\}$,

$P = \{ S \to ACD$

$A \to a$

$C \to ED$

$D \to BC \mid b$

$E \to b$

$\}$

Step I. $Pr = \{\varnothing\}$.

Step II. $Pr = P_r \cup \{A, E, D\} = \{A, E, D\}$.

Step III. $Pr = P_r \cup \{C, S\} = \{A, E, D, C, S\}$.

In this case there are no non-productive symbols.

Practical Tasks

I. Remove the non-productive symbols for the given grammars:

1. $G = (V_N, V_T, S, P)$:

   $V_N = \{ S, A, B, D\}$;

   $V_T = \{a, b, d\}$;

   $P = \{ S \to AB$

   $A \to a$

   $B \to b$

$A \to ABD$

$D \to d$

}

2. $G = (V_N, V_T, S, P)$:

$V_N = \{ S, A, B, D\}$;

$V_T = \{a, b\}$;

$P = \{ S \to AB$

$A \to a$

$B \to b$

$A \to ABD$

$D \to aBD$

}

3. $G = (V_N, V_T, S, P)$:

$V_N = \{ S, A, B, D\}$;

$V_T = \{a, b\}$;

$P = \{ S \to AB$

$A \to a$

$B \to b$

$A \to ABD$

$D \to BA$

}

4. $G = (V_N, V_T, S, P)$:

$V_N = \{ S, A, B, D, E\}$;

$V_T = \{a, b\}$;

$P = \{ S \to AB$

$A \to a$

$B \to b$

$A \to ABD$

$D \to BDA$

$E \to A\}$

II. For the given Grammar remove the inacesible symbols and non-productive symbols:

$G = (V_N, V_T, S, P)$:

$V_N = \{ S, A, B, H\}$;

$V_T = \{0, 1\}$;

$P = \{ S \rightarrow 0S1|0SH|0|1B0$

$B \rightarrow 1H0|SH$

$H \rightarrow 1AB$

$A \rightarrow SB$

$\}$

3.4 Chomsky Normal Form

The Context free grammar G is said to be in **Chomsky normal form** if it doesn't contain :

1. ε – productions.
2. Unit productions.
3. Inaccessible Symbols.
4. Non-productive symbols.

And all of its productions rule are of the following form:

$$A \rightarrow BC, \quad \text{or}$$
$$A \rightarrow a, \quad \text{or} \quad S \rightarrow \varepsilon.$$

Algorithm to convert Context Free Grammar into Chomsky Normal Form

Step 1 − If the start symbol S occurs on some right side, it is created a new start symbol S' and is added a new production

$$S' \rightarrow S.$$

Step 2 – It shoul be removed ε – productions.

Step 3 − It shoul be removed the unit productions.

Step 4 − It shoul be removed the inaccessible symbols.

Step 5 − It shoul be removed the non-productive symbols.

Step 6 – The each production $A \rightarrow B_1...B_n$ is replaced with $A \rightarrow B_1C$ where $C \rightarrow B_2 ...B_n$.

This step is repeated for all productions that having two or more symbols in the right side.

Step 7– If the right side of any production is in the form $A \rightarrow aB$, where a is a terminal and A, B are non-terminal, then the production is replaced by $A \rightarrow XB$ and $X \rightarrow a$.

This step is repeated for every production which is in the form $A \rightarrow aB$.

Example:

$G = (V_N, V_T, S, P)$:
$V_N = \{ S, A, B, C, D\}$;
$V_T = \{a, b\}$,
$P=\{ \quad S \rightarrow aBbAC| \ AB$
$A \rightarrow a| \ ABBa$
$B \rightarrow \varepsilon| \ a$
$C \rightarrow aA$
$D \rightarrow ab$
$\}$

Solution:

1. **Removing ε productions:**

a) $N\varepsilon=\varnothing$.
b) For the production $B \rightarrow \varepsilon$, $\quad N\varepsilon=\varnothing \cup \{B\}$,
$N\varepsilon=\{B\}$.

Removing this production there are obtained:

$P'=\{ \quad S \rightarrow aBbAC|abAC| \ AB|A$
$A \rightarrow a|ABBa|ABa|Aa$
$B \rightarrow a$
$C \rightarrow aA$

$D \rightarrow ab$

}

2. Removing of the unit productions:

The unit production from P' is $S \rightarrow A$.

After elimination of this production therea are obtained:

$P'' = \{ \quad S \rightarrow aBbAC|abAC|AB|a|ABBa|ABa|Aa$

$A \rightarrow a|ABBa|ABa|Aa$

$B \rightarrow a$

$C \rightarrow aA$

$D \rightarrow ab$

}

3. Elimination of nonproductive symbols.

Step I. $Pr = \{\varnothing\}$.

Step II. $Pr = P_r \cup \{S, A, B\} = \{S, A, B\}$.

Step III. $Pr = P_r \cup \{C, D\} = \{ S, A, B, C, D \}$.

4. Elimination of the inaccesibile symbols.

Step I. $A_c = \{S\}$.

Step II. $A_c = A_c \cup \{A,B,C,a\} = \{S, A,B,C,a\}$.

Step III. $A_c = \{S, A,B,C,a\}$.

Step IV. $I = (V_N \cup V_T) \setminus A_c = (\{S, A, B, C, D\} \cup \{a, b\}) \setminus$
$\{S, A,B,C,a\} = \{D\}$.

In this case the inaccesible symbol is D and after elimination of this production therea are obtained:

$V_N = \{ S, A, B, C \}$, $V_T = \{a, b\}$,

$P''' = \{ \quad S \rightarrow aBbAC|abAC|AB|a|ABBa|ABa|Aa$

$A \rightarrow a|ABBa|ABa|Aa$

$B \rightarrow a$

$C \rightarrow aA$

}

5. The Chomsky Normal Form

P^{IV}= { $S \to X_4X_7|X_8X_9|AB|a|X_2X_3|A\ X_3|A\ X_1$

$A \to a|\ X_2X_3|\ X_2\ X_1|A\ X_1$

$B \to a$

$C \to X_1A$

$X_1 \to a$

$X_2 \to AB$

$X_3 \to BX_1$

$X_4 \to X_1B$

$X_5 \to AC$

$X_6 \to b$

$X_7 \to X_6\ X_5$

$X_8 \to X_1\ X_6$

$X_9 \to AC$

}

$V_T = \{a, b\}$,

V_N={ $S, A, B, C, X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9$ }.

Practical Tasks

Convert the given context free grammar into Greibach normal form:

1. $G = (V_N, V_T, S, P)$:

$V_N = \{ S, A, C, D, E\}$;

$V_T = \{a, b\}$;

P = { $S \to aAa$

$A \to Sb|bCC|DaA|\varepsilon$

$C \to abb|DD|\varepsilon$

$E \to aC$

$D \to aDa$

}

2. $G = (V_N, V_T, S, P)$:

$V_N = \{ S, A, B, C\};$
$V_T = \{0, 1\};$
$P = \{ S \to S0|1|AB$
$B \to AC$
$A \to \varepsilon$
$C \to \varepsilon$
$\}$

3. $G = (V_N, V_T, S, P)$:
$V_N = \{ S, A, B, C, D\};$
$V_T = \{a, b\};$
$P = \{ S \to aB|bA|A$
$B \to b|bS|aD|\varepsilon$
$A \to B|AS|bBAB|b$
$C \to Ba$
$D \to AA$
$\}$

4. $G = (V_N, V_T, S, P)$:
$V_N = \{ S, A, B, C\};$
$V_T = \{a, b\};$
$P = \{ S \to aABC$
$A \to AB|\varepsilon$
$B \to CA|a$
$C \to AA|b$
$\}$

5. $G = (V_N, V_T, S, P)$:
$V_N = \{ S, A, B, D\};$
$V_T = \{a, b\};$
$P = \{ S \to aB|DA$
$A \to a|BD|bDAB$
$B \to b|BA$
$D \to BA|\varepsilon$
$\}$

6. $G = (V_N, V_T, S, P)$:

$V_N = \{ S, A, B, D\}$;
$V_T = \{a, b\}$;
$P = \{ S \rightarrow aB|AC$
$A \rightarrow a|ASC|BC$
$B \rightarrow b|bS$
$C \rightarrow BA|\varepsilon$
$\}$

7. $G = (V_N, V_T, S, P)$:
$V_N = \{ S, A, B, D\}$;
$V_T = \{a, b\}$;
$P = \{ S \rightarrow aASAb|aB|b$
$A \rightarrow B$
$B \rightarrow b|\varepsilon$

$\}$

8. $G = (V_N, V_T, S, P)$:
$V_N = \{ S, A, B, C, D\}$;
$V_T = \{a, b\}$;
$P = \{ S \rightarrow aBbAC|aAB|abAC|aA$
$A \rightarrow B|ABBa|a|\varepsilon$
$B \rightarrow a$
$C \rightarrow aA$
$D \rightarrow ab$
$\}$

9. $G = (V_N, V_T, S, P)$:
$V_N = \{ S, A, B, C\}$;
$V_T = \{0, 1\}$;
$P = \{ S \rightarrow S0$
$S \rightarrow 1$
$S \rightarrow AB$
$B \rightarrow AC1$
$A \rightarrow \varepsilon$

$C \to \varepsilon$ }

10. $G = (V_N, V_T, S, P)$:

$V_N = \{ S, A, B, C\}$;

$V_T = \{a, b, c\}$;

$P = \{ S \to ABAC$

$A \to aA|\varepsilon$

$B \to bB|\varepsilon$

$C \to c$ }

3.5 Left recursion

Direct recursion

Let G be a context-free grammar and a production of G is said **left recursion**, if it has the form

$$A \to A\alpha_1\alpha_2 \ldots \alpha_n, \text{where } A \in V_N,$$
$$\alpha_i \in (V_N \cup V_T)^*.$$

Example:

$G = (V_N, V_T, S, P)$:

$V_N = \{ S, A, B\}$;

$V_T = \{a, b\}$,

$P = \{ S \to ACD$

$A \to a$

$C \to ED$

$D \to DC \mid b$

$E \to b$

}

In this case the left recursion is given by the production $D \to DC$.

Algorithm for removing the left recursion

It is supposed given the context free grammar that contains the following productions:

- 1) $A \rightarrow A\alpha_1, A \rightarrow A\alpha_2, \ldots, A \rightarrow A\alpha_n.$
- 2) $A \rightarrow \beta_1, A \rightarrow \beta_2, \ldots, A \rightarrow \beta_m.$

where $\alpha_i, \beta_j \in (V_N \cup V_T)^*$, $i = \overline{1,n}; j = \overline{1,m}.$

There are two methods for removing the left recutrsion:

1st Method

In this case it is introduced the new non-terminal symbol Y and there are obtained the following productions:

I $A \rightarrow \beta_1 Y, A \rightarrow \beta_2 Y, \ldots, A \rightarrow \beta_m Y$

II $Y \rightarrow \alpha_1; Y \rightarrow \alpha_2; \ldots Y \rightarrow \alpha_n;$

III $Y \rightarrow \alpha_1 Y; Y \rightarrow \alpha_2 Y; \ldots Y \rightarrow \alpha_n Y;$

IV $A \rightarrow \beta_1; \ldots; A \rightarrow \beta_m.$

Example:

$G=(V_N, V_T, S, P)$ $V_N=\{E, T\}$ $V_T=\{a,+\}$

$P=\{ E \rightarrow E+T|T$

$T \rightarrow a\}$

In this case the left recursion is given by the production $E \rightarrow E+T$.

Solution:

Applying the 1st method there are obtained:

$V_N=\{E, E', T\}$ $V_T=\{a,+\}$

$P=\{ E \rightarrow TE'$

$E \rightarrow T$

$E' \rightarrow +TE'$

$E' \rightarrow +T$

$T \rightarrow a$

$\}$

2nd Method:

The productions (1) and (2) can be presented:

1. $A \rightarrow \beta_1 Y \ldots\ldots A \rightarrow \beta_m Y;$

2. $Y \to \alpha_1 Y$ ….. $Y \to \alpha_n Y$;
3. $Y \to \varepsilon$.

Where Y is new non-terminal symbol.

Example:
$G=(V_N, V_T, S, P)$ $V_N=\{E, T\}$ $V_T=\{a,+\}$
$P=\{ E \to E+T|T$
$T \to a\}$
In this case the left recursion is given by the production $E \to E+T$.

Solution:
Applying the 2nd method there are obtained:
$V_N=\{E, E', T\}$ $V_T=\{a,+\}$
$P=\{ E \to TE'$
$E' \to +TE'$
$E' \to \varepsilon$
$\}$

Indirect recursion

A grammar is said to posess indirect left recursion if it is possible, starting from any symbol of the grammar, to derive a string whose head is that symbol.

Example:
$G = (V_N, V_T, S, P)$:
$V_N = \{ S, A, B, C\}$;
$V_T = \{e, f\}$,
$S=\{A\}$,
$P = \{ A \to Cd$
$B \to Ce$
$C \to A \mid B|f$

}

$C \rightarrow A \rightarrow Cd$ $\Rightarrow$ in this case we obtain the derivation $C \rightarrow Cd$, that represents the indirect recursion.

$C \rightarrow B \rightarrow Ce$ $\Rightarrow$ in this case we obtain the derivation $C \rightarrow Ce$, that represents the indirect recursion.

In this way the contect free grammar can be rewrited in the following way:

$P' = \{ A \rightarrow Cd$

$B \rightarrow Ce$

$C \rightarrow Cd \mid Ce \mid f$

}

In this case it is given the direct left recursion that can be removed in the following way:

$P'' = \{ A \rightarrow Cd$

$B \rightarrow Ce$

$C \rightarrow fC'$

$C' \rightarrow dC' \mid eC' \mid \varepsilon$

}

Removing indirect left recursion

Let any ordering of the nonterminals of the given context free grammar be

$$A_1, \ldots, A_m,$$

It will be removed the indirect left recursion by constructing an equivalant grammar G' such that

If $A_i \rightarrow A_j\,\alpha$ is any production of G', then $i < j$.

Example:

$G = (V_N, V_T, S, P)$:

$V_N = \{ S, A, B, C\}$;

$V_T = \{a, b\}$,

$S = \{A\}$,

$P = \{ A \rightarrow BC$

$B \rightarrow CA|b$

$C \rightarrow AA \mid a$

$\}$

There are introduced the notation:

$A_1 = A$

$A_2 = B$

$A_3 = C$

Ant the equivalent grammar can be rewrited:

$P' = \{ A_1 \rightarrow A_2 A_3$

$A_2 \rightarrow A_3 A_1 \mid b$

$A_3 \rightarrow A_1 A_1 \mid a$

$\}$

It is replaced $A_3 \rightarrow A_1 A_1$ by $A_3 \rightarrow A_2 A_3 A_1$ and then replace this by

$$A_3 \rightarrow A_3 A_1 A_3 A_1 \text{ and } A_3 \rightarrow b A_3 A_1$$

Eliminating direct left recursion in the above,

gives: $A_3 \rightarrow a \mid b A_3 A_1 \mid aA_3' \mid b A_3 A_1 A_3'$

$A_3' \rightarrow A_1 A_3 A_1 \mid A_1 A_3 A_1 A_3'$

The resulting grammar is then:

$P'' = \{ A_1 \rightarrow A_2 A_3$

$A_2 \rightarrow A_3 A_1 \mid b$

$A_3 \rightarrow a \mid b A_3 A_1 \mid aA_3' \mid b A_3 A_1 A_3'$

$A_3' \rightarrow A A_3 A \mid A_1 A_3 A_1 A_3'$

$\}$

Or

$P'' = \{ A \rightarrow B C$

$B \rightarrow C A \mid b$

$C \rightarrow a \mid b C A \mid aA_3' \mid b C A A_3'$

$A_3' \rightarrow A C A \mid A C A A_3'$

$\}$

Practical Tasks

Remove left recursion for the given grammars and use the both methods:

1. $G = (V_N, V_T, S, P)$:
   $V_N = \{ S, A, B, D\}$;
   $V_T = \{a, b, d\}$;
   $P = \{ S \to AB$
   $A \to a$
   $B \to b$
   $A \to ABD$
   $D \to d$
   $\}$
2. $G = (V_N, V_T, S, P)$:
   $V_N = \{ S, A, B, D\}$;
   $V_T = \{a, b\}$;
   $P = \{ S \to AB$
   $A \to a$
   $B \to b$
   $A \to AB|AD|A$
   $D \to aBD$
   $\}$
3. $G = (V_N, V_T, S, P)$:
   $V_N = \{ S, A, B, D\}$;
   $V_T = \{a, b\}$;
   $P = \{ S \to AB$
   $A \to a|b|D$
   $B \to b$
   $A \to ABD$
   $D \to BA$
   $\}$
4. $G = (V_N, V_T, S, P)$:

$V_N = \{ S, A, B, D, E\};$
$V_T = \{a, b\};$
$P = \{ S \to AB$
$A \to a$
$B \to b$
$A \to ABD$
$D \to DA$
$D \to a\}$

5. $G = (V_N, V_T, S, P)$:
$V_N = \{ S, A \};$
$V_T = \{a, b\};$
$P = \{ S \to Aa|b$
$A \to Sb$

$\}$

6. $G = (V_N, V_T, S, P)$:
$V_N = \{ S, A, B, C\};$
$V_T = \{a, b\};$
$P = \{ S \to BC$
$B \to CA$
$C \to S|a$
$A \to b$
$\}$
7. $G = (V_N, V_T, S, P)$:
$V_N = \{ S, A, B, X\};$
$V_T = \{a, b\};$
$P = \{ S \to XA|BB$
$B \to b|SB$
$X \to b$
$A \to a$
$\}$

8. $G = (V_N, V_T, S, P)$:

$V_N = \{ S, L, R, N\}$;

$V_T = \{+, i, p\}$;

$P = \{ S \rightarrow L$

$L \rightarrow N|NRL$

$R \rightarrow +|bR|Rb$

$N \rightarrow i|p|+$

$\}$

9. $G = (V_N, V_T, S, P)$:

$V_N = \{ S, X\}$;

$V_T = \{a\}$;

$P = \{ S \rightarrow SX|SSb\backslash XS|a$

$X \rightarrow Sa|Xb|b$

$\}$

10. $G = (V_N, V_T, S, P)$:

$V_N = \{ S, B, C\}$;

$V_T = \{a, b\}$;

$P = \{ S \rightarrow AB$

$B \rightarrow c$

$C \rightarrow aA|bB$

$C \rightarrow b|SC$

$\}$

11. $G = (V_N, V_T, S, P)$:

$V_N = \{ S, A, B, C\}$;

$V_T = \{a, b\}$;

$P = \{ S \rightarrow BC$

$B \rightarrow CA$

$C \rightarrow S|a$

$A \rightarrow b$

$\}$

3.6 Greibach Normal Form

The context free grammar is in Greibach normal form (GNF) if all production have the form:

$$A \to a\alpha,$$

where $A \in V_N$, $a \in V_T$, $\alpha \in V_N^*$.

There are two way of conversion the CFG into Greibach Normal Formal:

I Algorithm

Step 1. CFG is converted the grammar into Chomsky Normal Form.
Step 2. It is eliminated the left recursion from grammar if it exists.
Step 3. It is converted the production rules into Greibach Normal Form form.

Example:
$G = (V_N, V_T, S, P)$:
$V_N = \{ S, A\}$;
$V_T = \{a, b\}$,
$P = \{ S \to Ab$
$\quad A \to aS|Ab|a$
$\}$
Step 1. Conversion into Chomsky Normal Form.
$V_N = \{ S, A, X, Y\}$;
$V_T = \{a, b\}$,
$P' = \{ S \to AY$
$\quad A \to XS|AY|a$
$\quad X \to a$
$\quad Y \to b$
$\}$
Step 2. Removing of the left recursion from grammar.

In this case we have the left recursion that should be removed for productions $A \rightarrow XS|AY|a$ and there are obtained the following productions:

$V_N = \{ S, A, A', X, Y\}$;

$V_T = \{a, b\}$,

$P' = \{ S \rightarrow AY$

$A \rightarrow XSA'|XS|aA'|a$

$A' \rightarrow YA'|b$

$X \rightarrow a$

$Y \rightarrow b$

$\}$

Step 3. Conversion into Greibach Normal Form form.

$P' = \{ S \rightarrow aSA'Y|aSY|aA'Y|aY$

$A \rightarrow aSA'|aS|aA'|a$

$A' \rightarrow bA'|b$

$X \rightarrow a$

$Y \rightarrow b$

$\}$

II Algorithm

Step 1. CFG is converted the grammar into Chomsky Normal Form.

Step 2. All non-terminal symbols are renamed in $A_1, A_2, \ldots A_n$.

Step 3. It should be modified the rule productions so that

$$A_i \rightarrow A_j\alpha, \text{ then } j>i.$$

Step 4. If $A_i \rightarrow A_j\alpha$ and $j<i$, it is generated a new set of productions with substitution A_j and is obtained

$$A_k \rightarrow A_p\alpha, \text{ then } p \geq k.$$

Step 5. It is eliminated the left recursion from grammar if it exists.

Step 6. It is converted the production rules into Greibach Normal Form form.

Example:

$G = (V_N, V_T, S, P)$:

$V_N = \{ S, A\}$;

$V_T = \{a, b\}$,

$P = \{ S \rightarrow Aa$

$A \rightarrow aS|a$

$\}$

Step 1. Conversion into Chomsky Normal Form.

$V_N = \{ S, A, X \}$;

$V_T = \{a, b\}$,

$P' = \{ S \rightarrow AX$

$A \rightarrow XS|a$

$X \rightarrow a$

$\}$

Step 2. All non-terminal symbols are renamed in $A_1, A_2,\ldots A_n$.

$A_1=S$;

$A_2 =A$;

$A_3=X$.

$P'' = \{ A_1 \rightarrow A_2 A_3$

$A_2 \rightarrow A_3A_1|a$

$A_3 \rightarrow a$

$\}$

Step 3. It is verified all indexes and there are 1<2, 2<3 and there is no left recursion. In this case the CFG can be converted in Greibach Normal form.

$P''' = \{ A_1 \rightarrow a\, A_1\, A_3|a\, A_3$

$A_2 \rightarrow a\, A_1|a$

$A_3 \rightarrow a$

$\}$

$P^{IV} = \{ S \rightarrow a\, S\, X|a\, X$

$A \to a\, S|a$
$X \to a$
}

Practical Tasks

Convert the given context free grammar into Greibach normal form:

1. $G = (V_N, V_T, S, P)$:
   $V_N = \{ S, A, B, C\}$;
   $V_T = \{a, b\}$;
   $P = \{ S \to BC$
   $B \to CA$
   $C \to S|a$
   $A \to b$
   }
2. $G = (V_N, V_T, S, P)$:
   $V_N = \{ E, Y, T, F, Z\}$;
   $V_T = \{+, *, (,), a\}$;
   $P = \{ E \to T|TY|+TY$
   $Y \to +T$
   $T \to F|FZ|*FZ$
   $Z \to *F$
   $F \to a|(E)$
   }

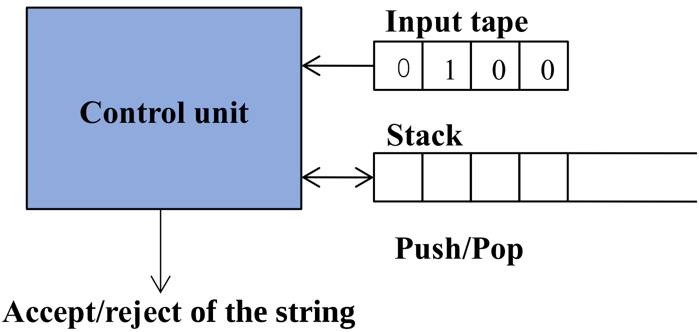
3.7 Pushdown Automata

A pushdown automaton (PDA) is a is a finite automata with extra memory which is called stack, that permits to recognize a context-free language in a similar way, as DFA or NFA recognize a regular language. A DFA can remember a finite amount of information, but a PDA can remember an infinite amount of information and consists of −

"Finite state machine" + "a stack".

A pushdown automaton has three components −

- an input tape;
- a control unit;
- a stack with infinite size.



The stack head scans the top symbol of the stack.

A stack does two operations −

- **Push** − a new symbol is added at the top.
- **Pop** − the top symbol is read and removed.
-

A pushdown automaton is $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where:

- Q is a finite set of states;
- Σ is the input alphabet;
- Γ is the stack alphabet
- q_0 in Q is the initial state;
- $F \subseteq Q$ is a set of final states;
- δ is the transition function

$$\delta: Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \rightarrow \text{subsets of } Q \times (\Gamma \cup \{\varepsilon\}).$$

A PDA is nondeterministic and the language of a PDA is the set of all strings in Σ^* that can lead the PDA to an accepting state.

Instantaneous Description

The instantaneous description (ID) of a PDA is represented by a triplet (q, w, s) where

- q is the state;
- w is input string;
- s is the stack contents.

The Transition Function

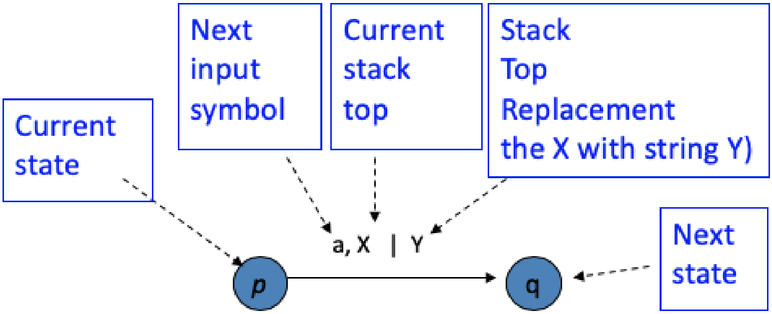
This function takes three arguments:

1. A state, in Q.
2. An input, which is either a symbol in Σ or ε.
3. A stack symbol in Γ.

$$\delta(p,a,X) = \{(q,Y)\}$$

This describes the trasnition from p state to q state, reading the input symbol a, where the pop symbol from stack is X and the push symbol in stack is Y.

The following diagram shows a transition in a PDA from a state p to state q.



Turnstile Notation

The "turnstile" notation is used for connecting pairs of ID's that represent one or many moves of a PDA. The process of transition is denoted by the turnstile symbol "⊢".

Consider a given PDA, a transition can be mathematically represented by the following turnstile notation

$$(p, wa, \beta) \vdash (q, a, \alpha)$$

This implies that while taking a transition from state *p* to state *q*, the input symbol *w* is consumed, and the top of the stack β is replaced by a new string α.

There are three forms of representation oft he PDA:

- Table representation.
- Analytical form.
- Graphical representation.

Acceptance of the string

There are two different ways to define PDA acceptability:

1. Final State Acceptability

For the given PDA P, the language accepted by P, is denoted by $L(P)$ by *final state*, is:

$$\{w \mid (q_0,w,Z_0) \vdash (q,\varepsilon,\varepsilon)\}, \text{ s.t., } q \in F$$

2. Empty Stack Acceptability

For a PDA P, the language accepted by P, denoted by $N(P)$ by *empty stack*, is:

$$\{w \mid (q_0,w,Z_0) \vdash (q,\varepsilon,\varepsilon)\}, \text{ for any } q \in Q.$$

Example:

For the given PDA in the analytical form:

- present table and graph representation.
- analyze the word *aaaa*.

$M=(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

$Q=\{q_0,q_1\}$, $\Sigma=\{a,b\}$, $\Gamma=\{a,b,A,B\}$, $q_0=\{q_0\}$, $Z_0=\{\lambda\}$, $F=\{q_1\}$

$\delta(q_0,\mathrm{a},\varepsilon) = \{(q_0,\mathrm{A})\}$

$\delta(q_0,\mathrm{b},\varepsilon) = \{(q_0,\mathrm{B})\}$

$\delta(q_0,\varepsilon,\varepsilon) = \{(q_1,\varepsilon)\}$

$\delta(q_1,\mathrm{a},\mathrm{A}) = \{(q_1,\varepsilon)\}$

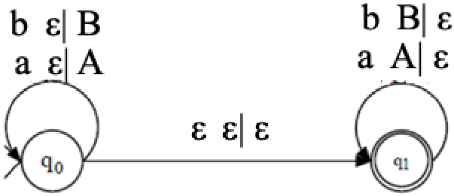
$\delta(q_1,\mathrm{b},\mathrm{B}) = \{(q_1,\varepsilon)\}$

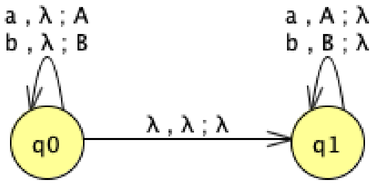
Solution:

- Tabel representation

δ	a	b	ε
$\rightarrow q_0$	q_0 ε| A	q_0 ε| B	q_1 ε| ε
* q_1	q_0 A| ε	q_0 B| ε	-

- Graphical representation





Curent state	Input	Pop string	Push String	New state
q0	ε	ε	ε	q1
q0	a	ε	A	q0
q0	ε	ε	ε	q1
q1	a	A	ε	q1
q1	a	A	ε	q1

$[q_0, aaaa, \varepsilon] \vdash [q_0, aaa, A] \vdash [q_0, aa, AA] \vdash [q_1, aa, AA] \vdash [q_1, a, A] \vdash [q_1, \varepsilon, \varepsilon]$

Convertion a Context free grammar into a PDA

Every CFG can be converted to an equivalent PDA. The constructed PDA will perform a leftmost derivation of any string accepted by the CFG.

It is given a CFG and let V_T and V_N be its sets of terminal and non-terminal symbols respectively. Let S be the start symbol. Then the steps for the obtaining the PDA are the following:

- The input alphabet oft he PDA is V_T.
- There are only three states, that are denoted with q_1, q_2, q_3, where q_1 is the start state, q_3 is the only one final state, in this way $Q = \{q_1, q_2, q_3\}$, $S=\{q_1\}$, $F=\{q_3\}$.
- The stack alphabet is $\Gamma = V_T \cup V_N$.
- There are add the transitions as follows:
  - $\delta(q_1, \varepsilon, \varepsilon) = \{(q_2, \$S)\}$
  - For each grammar rule of the form $A \rightarrow \alpha$, there is a
    $$\delta(q_2, \varepsilon, A) = \{(q_2, \alpha)\}$$
    where $\alpha \in (V_T \cup V_N)^*$.
  - For each terminal symbol a from V_T, it add the transition
    $$\delta(q_2, a, a) = \{(q_2, \varepsilon)\}$$
  - In final it is add the transition
    $$\delta(q_2, \$, \varepsilon) = \{(q_3, \varepsilon)\}$$
    where \$ is a marker, which means end and start of the string.

Example:

It is givem the context free grammar and it is necessary to convert this grammar to the PDA.

$G = (V_N, V_T, S, P)$:

$V_N = \{A, B\}$;

$V_T = \{0,1,\#\}$;

$S=\{A\}$.

$P = \{ \ A \rightarrow 0A1$

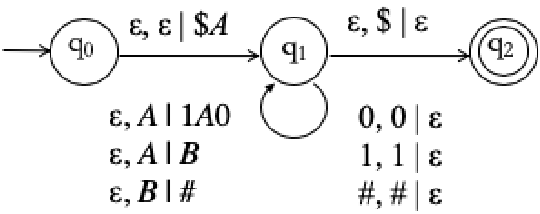
$\quad A \rightarrow B$

$\quad B \rightarrow \#$

$\}$

Solution:

$M=(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$
$Q=\{q_0, q_1, q_2\}$, $\Sigma=\{0,1, \#\}$, $\Gamma=\{0,1,\#, A, B\}$, $q_0=\{q_0\}$, $Z_0=\{\lambda\}$, $F=\{q_2\}$
$\delta(q_0,\varepsilon,\varepsilon) = \{(q_0, \$A)\}$
$\delta(q_1,\varepsilon,A) = \{(q_1,1A0)\}$
$\delta(q_1,\varepsilon,A) = \{(q_1,B)\}$
$\delta(q_1,\varepsilon,B) = \{(q_1,\#)\}$
$\delta(q_1,0,0) = \{(q_1, \varepsilon)\}$
$\delta(q_1,1,1) = \{(q_1, \varepsilon)\}$
$\delta(q_1,\#,\#) = \{(q_1, \varepsilon)\}$
$\delta(q_1,\varepsilon,\$) = \{(q_2, \varepsilon)\}$



Practical Tasks

1. Construct pushdown automata and present the analysis of the word for the following languages :

a) $L=\{a^n c b^n | n \in N, a, b, c \in \Sigma\}$
b) $L=\{a^n c b^m | n, m \in N, a, b, c \in \Sigma\}$
c) $L=\{a c^m b | n, m \in N, a, b, c \in \Sigma\}$
d) $L=\{a^n c b^{n+1} | n \in N, a, b, c \in \Sigma\}$
e) $L=\{a^{n-1} c^n b | n \in N, a, b, c \in \Sigma\}$
f) $L=\{a^{n+1} c b^{n+1} | n \in N, a, b, c \in \Sigma\}$
g) $L=\{a^n c^{n+1} b | n \in N, a, b, c \in \Sigma\}$
h) $L=\{a^m c^n b^{n+1} | n \in N, a, b, c \in \Sigma\}$

i) $L=\{da^ncb^{n+1}d \mid n \in N,\ a, b, c, d \in \Sigma\}$

j) $L=\{a^nc^mb \mid n \in N,\ a, b, c \in \Sigma\}$

2. For the given Push Down Automaton $M=(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, $Q=\{q_0,q_1, q_2\}$, $\Sigma=\{a,b\}$, $\Gamma=\{a,b, A,B\}$, $q_0=\{q_0\}$, $Z_0=\{\varepsilon\}$, $F=\{q_2\}$

$\delta(q_0,a,\varepsilon) = \{(q_1,A)\}$ $\delta(q_1,b, \varepsilon) = \{(q_1, B)\}$

$\delta(q_1, \varepsilon, \varepsilon) = \{(q_2, \varepsilon)\}$ $\delta(q_2,a,A) = \{(q_2, \varepsilon)\}$

$\delta(q_2,b,B) = \{(q_2, \varepsilon)\}$ $\delta(q_2,a, \varepsilon) = \{(q_2, \varepsilon)\}$

a) Present the PDA in the graph form. Present the PDA in the graph form.

b) Present the PDA in the table form.

c) Analyze the word: *abbababba*.

3. Convert the given CFG tp PDA

a) $G= (V_N, V_T, S, P)$, $V_N = \{S, B, C\}$, $V_T = \{a, b\}$,
$P=\{S\rightarrow aSBC;\ BC\rightarrow B;\ C\rightarrow a;\ B\rightarrow b\}$.

b) $G= (V_N, V_T, S, P)$, $V_N = \{S, C, D\}$, $V_T = \{0,1\}$,
$P=\{S\rightarrow CD;\ C\rightarrow 0C|0;\ D\rightarrow 1D|1\}$.

c) $G= (V_N, V_T, S, P)$, $V_N = \{S, B, C\}$, $V_T = \{a, b\}$,
$P=\{S\rightarrow aSBC;\ BC\rightarrow Bab;\ C\rightarrow a;\ B\rightarrow b|\ \varepsilon\}$.