Lecture: Chomsky Normal Form

Greibach Normal Form
1. **Chomsky Normal Form**

Chomsky Normal Form requires that each rule in the grammar is either

(C1) of the form $A \rightarrow BC$, where $A$, $B$, $C$ are all variables and neither $B$ nor $C$ is the start variable. (That is, a rule has exactly two variables on its right side.)

(C2) $A \rightarrow a$, where $A$ is a variable and $a$ is a terminal.

(A rule with terminals on its right side, has only a single character.)

(C3) $S \rightarrow \varepsilon$, where $S$ is the start symbol.

(The start variable can derive $\varepsilon$, but this is the only variable that can do so.)

Note, that rules of the form $A \rightarrow B$, $A \rightarrow BCD$ or $A \rightarrow aC$ are all illegal in a CNF. Also a grammar in CNF never has the start variable on the right side of a rule.

2. **Outline of conversion algorithm**

All context-free grammars can be converted to CNF. We did most of the steps already. Here is an outline of the procedure:

(i) Create a new start symbol $S_0$, with new rule $S_0 \rightarrow S$ mapping it to old start symbol (i.e., $S$).

(ii) Remove nullable variables (i.e., variables that can generate the empty string).

(iii) Remove unit rules (i.e., variables that can generate each other).

(iv) Restructure rules with long right-hand sides.

The only step we did not describe yet is the last one.

**Example**

Let us look at an example grammar with start symbol $S$.

$$
\Rightarrow \\
S \rightarrow ASA \mid aB \\
A \rightarrow B \mid S \\
B \rightarrow b \mid \varepsilon
$$
After adding the new start symbol $S_0$, we get the following grammar:

$$
\Rightarrow \quad S_0 \rightarrow S \\
S \rightarrow ASA \mid aB \\
A \rightarrow B \mid S \\
B \rightarrow b \mid \epsilon
$$

Removing nullable (epsilon) variables:

$$
\Rightarrow \quad S_0 \rightarrow S \\
S \rightarrow ASA \mid aB \mid a \mid SA \mid AS \\
A \rightarrow B \mid S \\
B \rightarrow b
$$

Removing unit rules:

$$
\Rightarrow \quad S_0 \rightarrow ASA \mid aB \mid a \mid SA \mid AS \\
S \rightarrow ASA \mid aB \mid a \mid SA \mid AS \\
A \rightarrow b \mid ASA \mid aB \mid a \mid SA \mid AS \\
B \rightarrow b
$$

Final restructuring. Now, we can directly patch any places where our grammar rules have the wrong form for CNF. First, if the rule has at least two symbols on its right-hand side but some of them are terminals, we introduce new variables which expand into these terminals.

$$
\Rightarrow \quad S_0 \rightarrow ASA \mid UB \mid a \mid SA \mid AS \\
S \rightarrow ASA \mid UB \mid a \mid SA \mid AS \\
A \rightarrow b \mid ASA \mid UB \mid a \mid SA \mid AS \\
B \rightarrow b \\
U \rightarrow a
$$
Then, if any rules have more than two variables on their right-hand side, we fix that with more new variables.

\[
\Rightarrow \quad S_0 \rightarrow A Z \mid U B \mid a \mid S A \mid A S \\
S \rightarrow A Z \mid U B \mid a \mid S A \mid A S \\
A \rightarrow b \mid A Z \mid U B \mid a \mid S A \mid A S \\
B \rightarrow b \\
U \rightarrow a \\
Z \rightarrow S A
\]

2. Greibach Normal Form (GNF)

GNF is form as:

\[
A \rightarrow a \alpha \quad , \quad A \in N \ , \ a \in (N \cup T)^*
\]

Theorem 1: substitution theory

If \( A \rightarrow \alpha_1 B \alpha_2 \) is production in CFG (G) And 

\( B \rightarrow B_1 | B_2 | \ldots | B_n \) are production which symbol B on left side, therefore we can substituted production \( A \rightarrow \alpha_1 B \alpha_2 \) by \( A \rightarrow \alpha_1 B_1 \alpha_2 | \alpha_1 B_2 \alpha_2 | \alpha_1 B_3 \alpha_2 | \ldots | \alpha_1 B_m \alpha_2 \) without affected on language.

Theorem 2: Left Recursive

If has

\[
A \rightarrow A \alpha_1 | A \alpha_2 | A \alpha_3 | \ldots | A \alpha_m \quad , \quad A \in N \ , \ \alpha \in (N \cup T)^*
\]

and

\[A \rightarrow B_1 | B_2 | \ldots | B_n\]

Can build new grammar by add new symbol non terminal (variable) such as \( A' \)

\[
A' \rightarrow B_1 | B_2 | \ldots | B_n \mid B_1 A' | B_2 A' | \ldots | B_n A'
\]

\[
A' \rightarrow \alpha_1 | \alpha_2 | \ldots | \alpha_m | \alpha_1 A' | \alpha_2 A' | \alpha_3 A' | \ldots | \alpha_m A'
\]

General steps convert CFG to GNF
1- Grammar convert to CNF

2-Non-terminal symbol (variable) change to A1, A2, A3, A4,…,Am whereas A1 is start symbol and N={A1, A2, A3, ....Am}

3- if grammar G from previous step has production form

   \[ A \rightarrow a\alpha, \ A \in N, \ a \in (N \cup T)^* \]

   or \[ A_i \rightarrow A_j\alpha, \ j > i \]

algorithm applied.

Begin

\[ i=0 \]

\textbf{while} \ i<>m \ \textbf{do}

\begin{verbatim}
  begin
  i=i+1
  \textbf{while} there exists a production of the form \( A_i \rightarrow A_j\alpha \) \( i<j \) \textbf{do} substitution for \( A_j \) according to theorem 1

  \textbf{if} there exist left–recursive production with \( A_i \) on the left hand side \textbf{then}

  introduce a new nonterminal \( A_j' \) and used theorem 2 to replace these left-recursive production by an equivalent set of production

  end

end

end

4- If grammar G from previous step has production form

\[ A_i \rightarrow A_j\alpha, \ j>i, \ \alpha \in (N \cup N' \cup T)^* \]

Algorithm applied
Begin

\[ i=m \]

while \( i<>1 \) do

begin

\[ i=i+1 \]

while there exist a production of form \( A_i \rightarrow A_j \alpha \ j>I \) do

substitution for \( A_j \) according to theorem 1

end

end

5- production has form \( A_i \rightarrow A_j \alpha \) are deleted by substituted due theorem 1

Example:

\[
S \rightarrow AS|AB
\]

\[
A \rightarrow BS| a
\]

\[
B \rightarrow AA| b
\]

1- Grammar convert to CNF, here grammar is written with CNF

2- Non-terminal symbol (variable) change

\[
A_1=S, A_2=A, A_3=B
\]

\[
A_1 \rightarrow A_2A_1|A_2A_3
\]

\[
A_2 \rightarrow A_3A_1| a
\]

\[
A_3 \rightarrow A_2A_2| b
\]

3- \( A_3 \rightarrow A_2A_2| b \) which violate form or \( A_i \rightarrow A_j \alpha \ j>i \), theorem 1 applied

\[
A_3 \rightarrow A_3A_1A_2| aA_2
\]

and substituted in grammar \( G \) (right hand side of rule \( A_2 \) )
A1 → A2A1|A2A3
A2 → A3A1| a
A3 → A3A1A2| aA2|b

We see A3 → A3A1A2| aA2|b which has left-recursive (fold nonterminal symbol) from left side. Theorem 2 resolve this state

A1 → A2A1|A2A3
A2 → A3A1| a
A3 → aA2|b|aA2 A3| b A3
A3 → A1A2|A1A2 A3

4- if has rule Ai → Ajα j>i , α ∈ (N U N’ U T)*

Applied theorem 1.

  aA2A3 ` A1A3|bA3 ` A1A3|aA3
A2 → aA2A1|bA1|aA2A3 ` A3|bA3 ` A1|a
A3 → aA2|b|aA2A3 `|bA3`
A3 ` → A1A2|A1A2A3`

5- production has form Ai ` → Ajα are deleted by substituted due theorem 1

A3 ` → aA2A1A1A2|bA1A1A2|aA2 A3 ` A1A1A2|b A3 ` A1A1A2|
aA1A2|aA2A1A3A2|bA1A3A2|aA2A3 ` A1A3A2|b A3 ` A1A3A2|
aA3A2|aA2A1A1A2A3 `|bA1A1A2 A3 `|aA2 A3 ` A1A1A2 A3 `|
b A3 ` A1A1A2 A3 `|aA1A2 A3 `|aA1A2 A3 `|aA2A1A3A2 A3 `|
bA1A3A2 A3 `|aA2 A3 ` A1A3A2 A3 `|b A3 ` A1A3A2 A3 `|aA3A2 A3 `