Normal Forms

**Chomsky normal form**

All productions must be of one of the forms:
\[ S \rightarrow \epsilon \]
\[ X \rightarrow a \]
\[ X \rightarrow YZ \]

“S is the only variable that can be replaced by empty string; RHS of all rules is of length at most 2; in particular a variable can be replaced by either a letter or 2 consecutive variables”

Any CFG is equivalent to some CFG in Chomsky normal form.

Chomsky normal form is used for a version of the pumping lemma for CFLs.
(compare with pumping lemma for regular sets)

**Greibach normal form**

All productions must be of one of the forms:
\[ S \rightarrow \epsilon \]
\[ X \rightarrow a \]
\[ X \rightarrow aX_1X_2 \ldots X_r \]

“You can replace a variable with a single constant followed by a possibly empty sequence of variables. As before start symbol only may be replaced by empty string.”

Any CFG is equivalent to some CFG in Greibach normal form.

Greibach normal form is used in proof of equivalence of CFGs and pushdown automata (analogous to equivalence of regular grammars and finite automata)

**Conversion to Chomsky normal form**

First, get rid of empty productions.
(ie. rules that allow a variable other than the start symbol to be replaced with empty string.)
\[ S \rightarrow XaY \]
\[ X \rightarrow YcX \]
\[ X \rightarrow b \]
\[ X \rightarrow c \]
\[ X \rightarrow \epsilon \]
\[ Y \rightarrow \epsilon \]

Example:
\[ S \rightarrow XaY \]
\[ S \rightarrow aY \]
\[ X \rightarrow YcX \]
\[ X \rightarrow \epsilon \]
\[ X \rightarrow Yc \]

Replace \[ X \rightarrow YcX \] with \[ X \rightarrow Yc \]

Replace \[ X \rightarrow YcX \] with \[ X \rightarrow Yc \]

**Conversion to Chomsky normal form**

First, get rid of empty productions.
(ie. rules that allow a variable other than the start symbol to be replaced with empty string.)
\[ S \rightarrow XaY \]
\[ X \rightarrow YcX \]
\[ X \rightarrow b \]
\[ X \rightarrow c \]
\[ X \rightarrow \epsilon \]
\[ Y \rightarrow \epsilon \]

Example:
\[ S \rightarrow XaY \]
\[ S \rightarrow Xa \]
\[ S \rightarrow aY \]
\[ S \rightarrow a \]

Replace \[ X \rightarrow YcX \] with \[ X \rightarrow Yc \]

Replace \[ X \rightarrow YcX \] with \[ X \rightarrow Yc \]
General Rule

Any empty production \( X \rightarrow \epsilon \) (assuming \( X \) is not the start symbol) can be re-expressed by making copies of rules where \( X \) appears on RHS, replacing that \( X \) in the RHS by \( \epsilon \).

Note: if you have e.g.

\[
S \rightarrow XaXbX \\
X \rightarrow \epsilon
\]

then you need rules for all patterns of deletion of the \( X \)'s, i.e. replace with

\[
S \rightarrow XaXbX \\
S \rightarrow aXbX \\
S \rightarrow XabX \\
S \rightarrow XaXb \\
S \rightarrow Xab \mid aXb \mid abX \mid ab
\]

Next, get rid of variable changes.

Example:

\[
X \rightarrow W \\
X \rightarrow c \\
Y \rightarrow aa \\
Z \rightarrow a \\
W \rightarrow a
\]

We note: \( X \) can be replaced by \( Y \) or \( Z \) or \( W \), and \( Y \) can be replaced by \( Z \).

Replace

\[
S \rightarrow XaaX \\
S \rightarrow YaaX
\]

For each variable, identify which single variables may replace it using one or more variable change rules.

For each (non-variable change) rule with \( X \) on RHS, add rules with all possible replacements of \( X \)'s on RHS by \( X_1, \ldots, X_r \).

Then (the variable change rules that allow \( X \) to be replaced by other variables become redundant) delete the variable change rules with \( X \) on LHS.

Do the above for all variables.
Next, get rid of rules where RHS contains a constant together with other symbols

\[ S \rightarrow \epsilon \]

We have rules of the form \( X \rightarrow a \) where \( S \) is the start symbol and \( \alpha \) is a string of length \( \geq 2 \).

Given any rule of the form

\[ X \rightarrow \beta a\gamma \]

(where \( a \in A \)), replace with

\[ X \rightarrow \beta U\gamma \]
\[ U \rightarrow a \]

where \( U \) is a new variable that may occur only in these 2 rules.

Finally, replace all rules whose RHS is of length \( > 2 \).

By now all rules have the form

\[ S \rightarrow \epsilon \]
\[ X \rightarrow a \]
\[ X \rightarrow X_1X_2X_3 \ldots X_r \]

where \( S \) is the start symbol, \( X \) and the \( X_j \) are variables.

For \( r > 2 \) replace \( X \rightarrow X_1X_2X_3 \ldots X_r \) by

\[ X \rightarrow X_1X_2X_3 \ldots X_{r-2}U \]
\[ U \rightarrow X_{r-1}X_r \]

where as before \( U \) is unique to the above 2 rules.

Obviously the first new rule above may need to be replaced using the same trick. Keep going until all RHS’s have length \( \leq 2 \).

Recall the syntax of Greibach normal form: all rules are of the form:

\[ S \rightarrow \epsilon \]
\[ X \rightarrow a \]
\[ X \rightarrow aX_1X_2 \ldots X_r \]

“You can replace a variable with a single constant followed by a possibly empty sequence of variables. As before start symbol only may be replaced by empty string.”

bigskip Next, we see how to translate any CFG into an equivalent Greibach grammar...

Conversion to Greibach normal form

We can assume grammar is already in Chomsky NF.

We describe 2 useful tricks to help us convert from Chomsky NF to Greibach NF.
trick 1: Removing Directly Left Recursive Productions

If I have
\[ X \rightarrow X\alpha \]
\[ X \rightarrow \beta \]
I can replace with
\[ X \rightarrow \beta \]
\[ X \rightarrow \beta Y \]
\[ Y \rightarrow \alpha \]
\[ Y \rightarrow \alpha Y \]
Both sets of rules say \( X \) is a \( \beta \) followed by a string of \( \alpha \)'s.
Generally, if there is more than one \( \alpha \) or \( \beta \), add new rules of above form for all \( \alpha \)'s and \( \beta \)'s.

---

Converting a CFG to Greibach Normal Form

Start by converting to Chomsky Normal Form. Then suppose I have e.g.
(1) \( S \rightarrow XY \quad X \rightarrow a \)
(2) \( S \rightarrow XW \quad Y \rightarrow b \)
(3) \( X \rightarrow YZ \quad Z \rightarrow c \)
(4) \( Y \rightarrow WZ \quad W \rightarrow d \)
Rules (1-4) can be rewritten:
(4) can be written as \( Y \rightarrow dZ \)
Then (3) can be written as \( X \rightarrow dZZ \mid bZ \)
(2) can be written as \( S \rightarrow dZZW \mid bZW \mid aW \)
(1) can be written as \( S \rightarrow dZZY \mid bZY \mid aY \)
We have converted to Greibach normal form.

---

trick 2: Removing a Right-hand variable from a Production.

Given
\[ X \rightarrow \alpha Y \beta \]
\[ Y \rightarrow \gamma_1 \gamma_2 \ldots \gamma_n \]
(above are all rules with \( Y \) on LHS) we can replace \( X \rightarrow \alpha Y \beta \)
with
\[ X \rightarrow \alpha \gamma_1 \beta \mid \alpha \gamma_2 \beta \mid \ldots \mid \alpha \gamma_n \beta \]

We were able to arrange the variables in a sequence such that if a variable begins the RHS of a rule, it occurs later in the sequence than the LHS variable. Then we worked backwards through the sequence converting to Greibach normal form.
But what if rule (4) had been:
\[ Y \rightarrow XZ \]
then substituting for \( X \) in (4) would give us
\[ Y \rightarrow aZ \mid YZZ \]
Now what? Problem seems to be a directly left recursive production.
General Approach

Given a Chomsky normal form grammar with variables \{S, X_1, \ldots, X_n\}, add new variables \(Y_0, \ldots, Y_n\) that will allow conversion to rules of the form:

- \(S \rightarrow \epsilon | \langle \text{string of vars not starting with } S \rangle\)
- \(S \rightarrow a \langle \text{string of vars} \rangle\)
- \(X_i \rightarrow a \langle \text{string of vars} \rangle\)  \(i > 1\)
- \(Y_0 \rightarrow X_j \langle \text{string of vars} \rangle\)  \(j > 1\)

First use \(Y_0\) to get rid of any directly left recursive productions for \(S\).

Use substitution procedure to remove productions of the form \(X_i \rightarrow S \alpha\), where \(i > 1\).

Then \(Y_1\) for \(X_1\), etc.

General Approach (continued)

Continue forward through the sequence, introducing \(Y_j\) if needed for \(X_j\). When we finish, rules are of desired form. Then we substitute for variables backwards through the sequence.

- \(X_n \rightarrow \text{RHS is in Greibach normal form.}\)
- \(X_{n-1}/Y_{n-1} \rightarrow \text{RHS; Here RHS may start with } X_n, \text{if so we can substitute to get Greibach normal form.}\)
- \(X_{n-2} \rightarrow \text{RHS; substitute again (since RHS may start with } X_{n-1}, X_n, Y_{n-1} \text{ or } Y_n)\)

Example

(1) \(S \rightarrow XY\)  \(X \rightarrow a\)
(2) \(S \rightarrow XW\)  \(Y \rightarrow b\)
(3) \(X \rightarrow YZ\)  \(Z \rightarrow c\)
(4) \(Y \rightarrow XZ\)  \(W \rightarrow d\)

Arrange in order: \(S, X, Y, Z, W\)

(4) is unsatisfactory (w.r.t above ordering); replace with

\(Y \rightarrow aZ\)
\(Y \rightarrow YZZ\)

Replace 2nd of these new rules with

\((\text{using the trick for directly left recursive productions; note } Y \Rightarrow^* (aZ | b)(ZZ)^* \text{ and } Y' \Rightarrow^* ZZ^+)\)

Order: \(S, X, Y, Y', Z, W\)
Example (continued)

We have sequence: \( S, X, Y, Y', Z, W \)

(1) \( S \rightarrow XY \quad X \rightarrow a \)

(2) \( S \rightarrow XW \quad Y \rightarrow b \)

(3) \( X \rightarrow YZ \quad Z \rightarrow c \)

\( Y \rightarrow aZ \)

\( Y \rightarrow aZ'Y' \mid bY' \mid aZ \mid b \)

\( Y' \rightarrow ZZY' \mid ZZ \)

Don't need to substitute for \( W \).

bottom rule becomes (by substituting for \( Z \)):

\( Y' \rightarrow cZY' \mid cZ \)

(3) becomes (by substituting for \( Y' \)):

\( X \rightarrow aZZ \mid aZY'Z \mid bY'Z \mid bZ \)

(Noting we also have \( X \rightarrow a, \) (1) and (2) become:

\[
\begin{align*}
S & \rightarrow aZZY \mid aZY'ZY \mid bY'ZY \mid bZY \\
aZZW & \mid aZY'ZW \mid bY'ZW \mid bZW \\
aY & \mid aW
\end{align*}
\]