Section: Transforming grammars  
(Ch. 6)

Methods for Transforming Grammars

We will consider CFL without \( \lambda \). It would be easy to add \( \lambda \) to any grammar by adding a new start symbol \( S_0 \),

\[
S_0 \rightarrow S \mid \lambda
\]
Theorem (Substitution) Let G be a CFG. Suppose G contains

\[ A \rightarrow x_1Bx_2 \]

where A and B are different variables, and B has the productions

\[ B \rightarrow y_1 \mid y_2 \mid \ldots \mid y_n \]

Then can construct G’ from G by deleting

\[ A \rightarrow x_1Bx_2 \]

from P and adding to it

\[ A \rightarrow x_1y_1x_2 \mid x_1y_2x_2 \mid \ldots \mid x_1y_nx_2 \]

Then, \( L(G) = L(G') \).
Example:

\[ S \rightarrow aBa \quad \text{becomes} \]
\[ B \rightarrow aS \mid a \]

Definition: A production of the form \( A \rightarrow Ax, \ A \in V, \ x \in (V \cup T)^* \) is \textit{left recursive}. 
Example Previous expression grammar was left recursive.

\[
\begin{align*}
E & \rightarrow E+T \mid T \\
T & \rightarrow T*F \mid F \\
F & \rightarrow I \mid (E) \\
I & \rightarrow a \mid b
\end{align*}
\]

Derivation of \(a+b+a+a\) is:

\[
\begin{align*}
E \Rightarrow & \ E+T \Rightarrow \ E+T+T \Rightarrow \ E+T+T+T \\
\Rightarrow & \ a+T+T+T
\end{align*}
\]
Theorem (Removing Left recursion)
Let $G=(V, T, S, P)$ be a CFG. Divide productions for variable $A$ into
left-recursive and non left-recursive productions:

$$A \rightarrow A x_1 \mid A x_2 \mid \ldots \mid A x_n$$

$$A \rightarrow y_1 \mid y_2 \mid \ldots \mid y_m$$

where $x_i, y_i$ are in $(V \cup T)^*$. 

Then $G'=(V \cup \{Z\}, T, S, P')$ and $P'$
replaces rules of form above by

$$A \rightarrow y_i y_i Z, \ i=1,2,\ldots,m$$

$$Z \rightarrow x_i x_i Z, \ i=1,2,\ldots,n$$
Example:

E → E + T | T becomes

T → T * F | F becomes

Now, Derivation of a + b + a + a is:
Useless productions

S → aB | bA
A → aA
B → Sa
C → cBc | a

What can you say about this grammar?

Theorem (useless productions) Let G be a CFG. Then ∃ G’ that does not contain any useless variables or productions s.t. L(G)=L(G’).
To Remove Useless Productions:
Let $G = (V, T, S, P)$.

I. Compute $V_1 = \{\text{Variables that can derive strings of terminals}\}$

1. $V_1 = \emptyset$

2. Repeat until no more variables added
   - For every $A \in V$ with $A \rightarrow x_1 x_2 \ldots x_n$, $x_i \in (T^* \cup V_1)$, add $A$ to $V_1$

3. $P_1 = \text{all productions in } P \text{ with symbols in } (V_1 \cup T)^*$

Then $G_1 = (V_1, T, S, P_1)$ has no variables that can’t derive strings.
II. Draw Variable Dependency Graph

For $A \rightarrow xBy$, draw $A \rightarrow B$.

Remove productions for $V$ if there is no path from $S$ to $V$ in the dependency graph. Resulting Grammar $G'$ is s.t. $L(G)=L(G')$ and $G'$ has no useless productions.
Example:

S → aB | bA
A → aA
B → Sa | b
C → cBc | a
D → bCb
E → Aa | b
Theorem (remove $\lambda$ productions) Let $G$ be a CFG with $\lambda$ not in $L(G)$. Then $\exists$ a CFG $G'$ having no $\lambda$-productions s.t. $L(G) = L(G')$.

To Remove $\lambda$-productions

1. Let $V_n = \{A \mid \exists \text{ production } A \to \lambda \}$

2. Repeat until no more additions
   - if $B \to A_1 A_2 \ldots A_m$ and $A_i \in V_n$ for all $i$, then put $B$ in $V_n$

3. Construct $G'$ with productions $P'$ s.t.
   - If $A \to x_1 x_2 \ldots x_m \in P$, $m \geq 1$, then put all productions formed when $x_j$ is replaced by $\lambda$ (for all $x_j \in V_n$) s.t. $|\text{rhs}| \geq 1$ into $P'$. 

Example:

\[ S \rightarrow Ab \]
\[ A \rightarrow BCB \mid Aa \]
\[ B \rightarrow b \mid \lambda \]
\[ C \rightarrow cC \mid \lambda \]
Definition Unit Production

\[ A \rightarrow B \]

where \( A, B \in V \).

Consider removing unit productions:

Suppose we have
\[ A \rightarrow B \]
becomes
\[ B \rightarrow a \mid ab \]

But what if we have
\[ A \rightarrow B \]
becomes
\[ B \rightarrow C \]
\[ C \rightarrow A \]
Theorem (Remove unit productions)
Let $G=(V,T,S,P)$ be a CFG without $\lambda$-productions. Then $\exists$ CFG $G'=(V',T',S,P')$ that does not have any unit-productions and $L(G)=L(G')$.

To Remove Unit Productions:

1. Find for each $A$, all $B$ s.t. $A \Rightarrow B$ (Draw a dependency graph)
2. Construct $G'=(V',T',S,P')$ by
   (a) Put all non-unit productions in $P'$
   (b) For all $A \Rightarrow B$ s.t. $B \Rightarrow y_1 | y_2 | \ldots y_n \in P'$, put $A \Rightarrow y_1 | y_2 | \ldots y_n \in P'$
Example:

S → AB
A → B
B → C | Bb
C → A | c | Da
D → A
Theorem Let L be a CFL that does not contain \( \lambda \). Then \( \exists \) a CFG for L that does not have any useless productions, \( \lambda \)-productions, or unit-productions.

Proof

1. Remove \( \lambda \)-productions
2. Remove unit-productions
3. Remove useless productions

Note order is very important. Removing \( \lambda \)-productions can create unit-productions! QED.
Definition: A CFG is in Chomsky Normal Form (CNF) if all productions are of the form

$$A \rightarrow BC \quad \text{or} \quad A \rightarrow a$$

where $A,B,C \in V$ and $a \in T$.

Theorem: Any CFG $G$ with $\lambda$ not in $L(G)$ has an equivalent grammar in CNF.

Proof:

1. Remove $\lambda$-productions, unit productions, and useless productions.

2. For every rhs of length $> 1$, replace each terminal $x_i$ by a new variable $C_j$ and add the production $C_j \rightarrow x_i$.

3. Replace every rhs of length $> 2$ by a series of productions, each with rhs of length 2. QED.
Example:

\[ S \rightarrow CBcd \]
\[ B \rightarrow b \]
\[ C \rightarrow Cc \mid e \]
Definition: A CFG is in Greibach normal form (GNF) if all productions have the form

\[ A \rightarrow ax \]

where \( a \in T \) and \( x \in V^* \)

Theorem For every CFG G with \( \lambda \) not in \( L(G) \), \( \exists \) a grammar in GNF.

Proof:

1. Rewrite grammar in CNF.
2. Relabel Variables \( A_1, A_2, \ldots A_n \)
3. Eliminate left recursion and use substitution to get all productions into the form:

\[ A_i \rightarrow A_j x_j, \ j > i \]
\[ Z_i \rightarrow A_j x_j, \ j \leq n \]
\[ A_i \rightarrow a x_i \]

where \( a \in T \), \( x_i \in V^* \), and \( Z_i \) are new variables introduced for left recursion.

4. All productions with \( A_n \) are in the correct form, \( A_n \rightarrow a x_n \). Use these productions as substitutions to get \( A_{n-1} \) productions in the correct form. Repeat with \( A_{n-2}, A_{n-3} \), etc until all productions are in the correct form.