Methods for Transforming Grammars

We will consider CFL without $\epsilon$. It would be easy to add $\epsilon$ to any grammar by adding a new start symbol $S_0$.

$$S_0 \rightarrow S | \epsilon$$

**Definition:** A production of the form $A \rightarrow Ax$, $A \in V$, $x \in (V \cup T)^*$ is *left recursive*.

**Example** Previous expression grammar was left recursive.

$$E \rightarrow E + T | T$$
$$T \rightarrow T * F | F$$
$$F \rightarrow I | (E)$$
$$I \rightarrow a | b$$

A top-down parser would want to derive the leftmost terminal as soon as possible. But in the left recursive grammar above, in order to derive a sentential form that has the leftmost terminal, we have to derive a sentential form that has other terminals in it.

Derivation of $a + b + a + a$ is:

$$E \Rightarrow E + T \Rightarrow E + T + T \Rightarrow E + T + T + T \Rightarrow a + T + T + T$$

We will eliminate the left recursion so that we can derive a sentential form with the leftmost terminal and no other terminals.

**Theorem (Removing Left recursion)** Let $G = (V,T,R,S)$ be a CFG. Divide productions for variable $A$ into left-recursive and non left-recursive productions:

$$A \rightarrow Ax_1 | Ax_2 | \ldots | Ax_n$$
$$A \rightarrow y_1 y_2 | \ldots | y_m$$

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

Then $G' = (V \cup \{Z\}, T, R', S)$ and $R'$ 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:**

$$E \rightarrow E + T | T$$  becomes
$$T \rightarrow T * F | F$$  becomes

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

S → aB | bA
A → aA
B → Sa | a
C → cBc | a

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:


I. Compute V₁=\{Variables that can derive strings of terminals\}

1. V₁=∅
2. Repeat until no more variables added
   - For every A∈V with A→x₁x₂...xₙ, xᵢ∈(T* ∪ V₁), add A to V₁
3. R₁ = all productions in R with symbols in (V₁ ∪ T)*

Then G₁=(V₁,T,R₁,S) has no variables that can’t derive strings.

II. Draw Variable Dependency Graph

For A → xBy, draw A→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.

Theorem (remove ε productions) Let G be a CFG with ε not in L(G). Then ∃ a CFG G’ having no ε-productions s.t. L(G)=L(G’).

To Remove ε-productions

1. Let Vₙ = \{A | ∃ production A→ε \}
2. Repeat until no more additions
   - if B→A₁A₂...Aₘ and Aᵢ∈Vₙ for all i, then put B in Vₙ
3. Construct G’ with productions R’ s.t.
   - If A→x₁x₂...xₘ∈R, m ≥ 1, then put all productions formed when xⱼ is replaced by ε (for all xⱼ ∈ Vₙ) s.t. |rhs| ≥ 1 into R’.

Example:

S → Ab
A → BC | Aa
B → b | ε
C → cC | ε
**Definition** Unit Production

\[ A \to B \]

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

**Consider removing unit productions:**

Suppose we have

\[
A \to B \quad \text{becomes} \quad B \to a | ab
\]

But what if we have

\[
A \to B \quad \text{becomes} \quad B \to C \\
C \to A
\]

**Theorem** (Remove unit productions) Let \( G = (V, T, R, S) \) be a CFG without \( \epsilon \)-productions. Then \( \exists \) CFG \( G' = (V', T', R', S') \) 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 \( G' = (V', T', R', S') \) by
   (a) Put all non-unit productions in \( R' \)
   (b) For all \( A \Rightarrow B \) s.t. \( B \Rightarrow y_1y_2 \ldots y_n \in R' \), put \( A \Rightarrow y_1y_2 \ldots y_n \in R' \)

**Theorem** Let \( L \) be a CFL that does not contain \( \epsilon \). Then \( \exists \) a CFG for \( L \) that does not have any useless productions, \( \epsilon \)-productions, or unit-productions.

**Proof**

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

Note order is very important. Removing \( \epsilon \)-productions can create unit-productions! QED.

**Definition:** A CFG is in Chomsky Normal Form (CNF) if all productions are of the form

\[ A \to BC \quad \text{or} \quad A \to a \]

where \( A, B, C \in V \) and \( a \in T \).
**Theorem:** Any CFG $G$ with $\epsilon$ not in $L(G)$ has an equivalent grammar in CNF.

**Proof:**

1. Remove $\epsilon$-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:**

$S \rightarrow CBcd$

$B \rightarrow b$

$C \rightarrow Cc | \epsilon$

**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 $\epsilon$ not in $L(G)$, $\exists$ a grammar in GNF.