1. **Universality of Leftmost Derivations.**

**Theorem.** For any context free grammar $G$, if there is a derivation of $w \in \Sigma^*$ from the start symbol in $n$ steps, then there is a leftmost derivation in the same number of steps.

**Key Observation:** The re-write or production rules for a CFG are commutative; that is, if $A \to \alpha$ and $B \to \beta$, then

$$
\gamma_1 A\gamma_2 B\gamma_3 \to \gamma_1 A\gamma_2 \beta\gamma_3 \to \gamma_1 \alpha\gamma_2 \beta\gamma_3 \quad \text{and} \quad \gamma_1 A\gamma_2 B\gamma_3 \to \gamma_1 \alpha\gamma_2 B\gamma_3 \to \gamma_1 \alpha\gamma_2 \beta\gamma_3
$$

have the same yields.

**Proof.** We consider $w \in L(G)$ and a derivation of the string: $S = w_0 \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \ldots \Rightarrow w_n = w \in \Sigma^*$. Let $k$ be the least integer such that $w_k \Rightarrow w_{k+1}$ is not leftmost. We write:

$$
w_k = \gamma_1 A\gamma_2 B\gamma_3 \quad \text{and} \quad w_{k+1} = \gamma_1 A\gamma_2 \beta\gamma_3
$$

where $B \to \beta$.

We shall define a new derivation for $w$:

$$
w = w_0 \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \ldots \Rightarrow w_k \Rightarrow w'_k \Rightarrow \ldots \Rightarrow w'_n = w
$$

such that the first $k$ terms agree with the previous derivation so that they are leftmost and $w_k \Rightarrow w'_{k+1}$ is also leftmost.

In the original derivation of $w$, the nonterminal $A$ must be expanded. So, let $\ell$ be the least integer greater than $k$ such that $w_\ell = \gamma_1 A\xi$ and $w_{\ell+1} = \gamma_1 \alpha\xi$, where $\xi \in V^*$, $\gamma \in \Sigma^*$, and $A \to \alpha$. 

To sum up, we now have:

\[ w = w_0 \xrightarrow{G} w_k = \gamma_1 A(\gamma_2 B\gamma_3), \text{ in } k \text{ steps}, \]

\[ \gamma_1 A(\gamma_2 B\gamma_3) \Rightarrow (\gamma_1 \alpha)(\gamma_2 B\gamma_3) \text{ in 1 step}; \]

\[ (\gamma_1 \alpha)(\gamma_2 B\gamma_3) \xrightarrow{G} (\gamma_1 \alpha)\xi \text{ in } \ell - k \text{ steps}; \]

and \( (\gamma_1 \alpha)\xi \xrightarrow{G} w_n = w, \text{ in } (n - \ell - 1) \text{ steps}. \)

Note: \( w_{k+1} = \gamma_1 \alpha\xi \xrightarrow{G} w_n = w \) in \( (n - \ell - 1) \) steps, since this is just the completion of the original derivation.

It can be convenient to modify a given context free grammar to eliminate certain “useless” productions and to put it into a standard form. The next several items discuss these procedures.

2. Let \( G \) be a CFG. We call a non-terminal \( A \in V \setminus \Sigma \) productive if \( A \xrightarrow{G} w \in \Sigma^* \).

Let \( P \) be the set of all productive non-terminals. They are be found inductively as follows:

(a) We define the increasing sequence of sets \( P_0, P_1, \ldots, P_n \subseteq V \setminus \Sigma \), where \( n = |V \setminus \Sigma| \), by:

1. \( P_0 = \emptyset \),

2. \( P_{j+1} = \{ A \in V \setminus \Sigma : A \in P_j \) or \( A \rightarrow w \in R, \) for \( w \in (\Sigma \cup P_j)^* \} \),

Then: \( P_n = P \).

3. We compute the non-terminal symbols derivable from the start symbol \( S \).

Let \( X_0 = \{ S \} \), where \( S \) is the start symbol. We want to find all \( A \in V \setminus \Sigma \) such that \( S \xrightarrow{G} \alpha A\beta \), where \( \alpha, \beta \in V^* \).

Let \( X_{j+1} = \{ A \in (V \setminus \Sigma)^* : B \rightarrow A \) and \( B \in X_j \} \).

Then: \( X_0 \subseteq X_1 \subseteq X_2 \subseteq \ldots \) and eventually this sequence terminates. Further, if \( S \xrightarrow{G} A \), then \( A \) is a member of some \( X_j \).
4. We call a nonterminal $A$ **nullable** if $A \xrightarrow{\epsilon} G$. 

Computation of finding all nullable nonterminals:

$$X_0 = \{A \in (V \setminus \Sigma) : A \rightarrow \epsilon\},$$

$$X_{j+1} = \{A \in (V \setminus \Sigma) : A \rightarrow w, \text{ where } w \in X_j^*\}, \text{ for } j = 0, 1, 2, \ldots$$

5. In item 4, we indicated how to identify the nullable non-terminals. Now, we describe how to remove the $\epsilon$-rules that result from their presence. The following procedure removes the $\epsilon$-rules while finding the nullable non-terminals.

Let $R$ be the set of productions from the grammar.

We repeat the following until there are no rules of the form $A \rightarrow \epsilon$ in $R$.

Choose $A \in V \setminus \Sigma$ if $A \rightarrow \epsilon$.

Then for each rule in $R$, say, $B \rightarrow X_1X_2\ldots X_n$,

we create new rules $B \rightarrow Y_1Y_2\ldots Y_n$, such that $Y_i = X_i$, if $X_i \neq A$, and $Y_i = \epsilon$, if $X_i = A$, and add these rules to $R$.

Note: there is an exceptional case when all the $Y_i$'s are $\epsilon$. We do not add the rule $B \rightarrow \epsilon$, if $B \rightarrow \epsilon$, is a rule that has already been eliminated from $R$.

Since $\epsilon \in L(G)$, we let $S_0$ be a new start state and declare the rules $S_0 \rightarrow S$ and $S_0 \rightarrow \epsilon$ to be added to $R$.

6. **Unit Rules or Chain Rules**.

A unit rule or chain rule has the form: $A \xrightarrow{\epsilon} B$, where $A$ and $B$ are non-terminals.

We can eliminate such rules as follows provided the grammar has no $\epsilon$-rules.

We define: $\text{Unit}(A) = \{B \in V \setminus \Sigma : A \xrightarrow{\epsilon} B\}$.

Note: $A \in \text{Unit}(A)$, always.

We can compute $\text{Unit}(A)$ with the usual technique. Set
Week Seven ................................................. Normal Forms; Pumping Lemma

\[
X_0 = \{A\}.
\]
\[
X_{j+1} = \{B \in V \setminus \Sigma : C \rightarrow B, \text{ for some } C \in X_j\}.
\]

When these sets stabilize, they form Unit(A).

7. We can find an equivalent CFG with no chain rules.

Let \( G = (V, \Sigma, R, S) \) be given, then we find \( G' = (V, \Sigma, R', S) \) where

1. \( R' = R; \)
   2. for each \( A \in V \setminus \Sigma \) with Unit\((A) \neq \{A\}, \)
      for each \( B \in \text{Unit}(A), \)
         for each non-unit production \( B \rightarrow w \) in \( R, \)
            add the rule \( A \rightarrow w \) to \( R'; \)
   3. remove all unit productions from \( R'. \)

8. Chomsky Normal Form.

Let \( G = (V, \Sigma, R, S) \) be a context free grammar, with no \( \epsilon \)-productions. Then there is an equivalent CFG \( G' \) such that its production rules have the form:

1. \( A \rightarrow \sigma, \) for \( \sigma \in \Sigma, \) or
2. \( A \rightarrow BC, \) where \( B, C \in V \setminus \Sigma.\)

To convert \( G \) to Chomsky Normal Form, we assume \( \epsilon \notin L(G). \) Note: any non-empty string \( w \in L(G) \) has a derivation that does not use \( \epsilon \)-rules and unit "chain" rules. To see this, choose a minimal length derivation for \( w. \) Then it is easy to check that such a derivation will not contain \( \epsilon \)-rules and unit productions. We create a new start symbol \( S_0 \) together with the production \( S_0 \rightarrow S, \) so the start symbol cannot appear on the right hand side of any production rule. Then we

1. eliminate all \( \epsilon \)-productions,
2. eliminate all unit productions.
Now, if $A \to w$ and $|w| = 1$, then $w = \sigma \in \Sigma$, since $G$ has no unit productions.

Next, we consider a production $A \to w$, with $|w| \geq 2$. We write: $A \to X_1X_2 \ldots X_n$.

We show how to convert $A \to w$ to guarantee $w$ consists only of non-terminals.

If $X_j = \sigma$, create a new non-terminal $C_\sigma$ and the new rule $C_\sigma \to \sigma$ and $A \to X_1 \ldots X_{j-1}C_\sigma X_{j+1} \ldots X_n$.

Without loss of generality, we assume $A \to X_1X_2 \ldots X_n$, where $X_j \in V \setminus \Sigma$. We give the productions:

$A \to X_1 D_1, \ D_1 \to X_2 D_2, \ D_2 \to X_3 D_3, \ldots, \ D_{n-1} \to X_{n-1} X_n$.

**Example.** We assume we are given the following grammar $G$:

\[
\begin{align*}
S & \to ASA | aB \\
A & \to B | S \\
B & \to b | \epsilon
\end{align*}
\]

We first introduce a new start symbol $S_0$ with production $S_0 \to S$.

\[
\begin{align*}
S_0 & \to S \\
S & \to ASA | aB \\
A & \to B | S \\
B & \to b | \epsilon
\end{align*}
\]

**Eliminate all \(\epsilon\)-productions:**

\[
\begin{align*}
S_0 & \to S \\
S & \to ASA | aB | a \\
A & \to B | S | \epsilon \\
B & \to b \\
S_0 & \to S \\
S & \to ASA | aB | a | SA | AS | S
\end{align*}
\]

Remove the rule $B \to \epsilon$.

We now remove the unit rules.
Normal Forms; Pumping Lemma

At this point, all $\epsilon$-moves and all unit rules have been eliminated to form a new equivalent grammar.

We note that there are three production rules whose right hand side has three symbols and there are also three rules whose right hand side contains both a terminal symbol and a non-terminal symbol.

Therefore, we must introduce four new non-terminals: $D_1, D_2, D_3$, and $C_a$ with production rule $C_a \to a$.

$S_0 \to ASA$ is replaced by: $S_0 \to AD_1, D_1 \to SA$.

$S \to ASA$ is replaced by: $S \to AD_2, D_2 \to SA$.

$A \to ASA$ is replaced by: $A \to AD_3, D_3 \to SA$.

Next, replace $S_0 \to aB$ with $S_0 \to C_a B$.

Replace $S \to aB$ with $S \to C_a B$.

Replace $A \to aB$ with $A \to C_a B$.

9. Pumping Lemma. Let $G$ be a context free grammar, $L = L(G)$. Then there is a constant $k$ such that if $z \in L$ has length greater than $k$, we can write

$z = uvwx$, where

\[(1) \text{length} (vx) \geq 1; \text{ that is, either } v \neq \epsilon \text{ or } x \neq \epsilon.\]
(2) length \((vxy) \leq k\);

(3) \(uv^iwx^iy \in L\), for \(i = 0, 1, 2, \ldots\).

**Proof.** For regular languages, “pumping” was proven easily by using the finite number of states in an accepting DFA. For context free languages, we use the fact that there are only finitely many non-terminal symbols and combinatorial properties of the parse tree. Some authors simplify the proof by assuming the grammar is in Chomsky normal form, but this is not necessary.

Let \(G = (V, \Sigma, R, S)\) be the context free grammar,

let \(m = |V \setminus \Sigma|\), number of nonterminals in \(G\),

and let \(p = \max\{\alpha : A \rightarrow \alpha \in R\}\), which is the length of the longest production in \(G\).

Set \(k = p^{m+1}\), which we find is the pumping constant.

Consider any string \(z \in L\) whose length \(|z| \geq k\). Among all derivations of \(z\) (there are only finitely many), choose the shortest one, with parse tree \(T\).

*Observation:* The height of \(T\) is greater than or equal \(m + 1\), which we can verify by induction on the height of the tree.

In particular, in the derivation tree \(T\), there is at least one path \(\gamma\) of length greater than or equal \(m + 1\). Such a path must contain repeated non-terminal symbols since the path will contain \(m + 2\) nodes of which only the last can be a terminal symbol. Traversing the path \(\gamma\) from leaf (bottom) towards the root, let \(A\) be the first repeated nonterminal.

Note: by the choice of \(A\), there are no repeated nonterminals on \(\gamma\) below the upper occurrence of \(A\).

By construction, \(S \xrightarrow{\gamma} z\) is a derivation of \(z\) of minimal length. In particular,

(1) \(S \xrightarrow{\gamma} uAy\), where \(u, y \in \Sigma^*\) and \(A\) is the upper occurrence in \(\gamma\);

(2) \(A \xrightarrow{\gamma} vAx\), where \(v, x \in \Sigma^*\) and \(A\) is the lower occurrence in \(\gamma\);
(3) \( A \xrightarrow{G} w \), where \( w \in \Sigma^* \) and \( A \) is the lower occurrence in \( \gamma \).

To obtain step (1), we expand all the non-terminals that occur either to the left or right of the first occurrence of \( A \) in the derivation of \( z \). In step (2), we isolate the contribution of the expansion of the non-terminal \( A \) in going from the “upper/first” occurrence to the “lower/last” occurrence. Since \( A \xrightarrow{G} vAx \), we must have \(|vx| \geq 1\). Otherwise, \( v = z = \epsilon \), so (2) becomes \( A \xrightarrow{G} A \), a “chain rule.” This would contradict the minimality of the derivation.

Further, the length \(|vwx| \leq k\), because the height of the subtree \( T' \) whose root is the upper occurrence of \( A \) is less than or equal to \( m + 1 \) (the subtree \( T' \) has no repeated nonterminals except for \( A \)).

Finally, by omitting step (2), we find:

\[
S \xrightarrow{G} uAy \xrightarrow{G} uwy;
\]

by repeating step (2), we have:

\[
S \xrightarrow{G} uAy \xrightarrow{k}{G} uv^kAx^ky \xrightarrow{G} uv^kwx^ky, \text{ for } k = 1, 2, 3, \ldots
\]

10. Examples of the Pumping Lemma.

Example 1. \( L_1 = \{a^n b^n c^n : n \geq 1\} \) is not context free.

We assume that \( L_1 \) is context free. Let \( p \) be the pumping constant, and let \( s = a^p b^p c^p \).
Write \( s = uvxyz \), where either \( u \neq \epsilon \) or \( y \neq \epsilon \). We consider two cases based on whether \( v \) and \( y \) contain more than one type of alphabet symbol.

Case 1: When both \( v \) and \( y \) contain only one type of alphabet symbol, \( v \) does not contain both \( a \)'s and \( b \)'s or both \( b \)'s and \( c \)'s, and the same holds for \( y \). So, the string \( uv^2xy^2z \) cannot contain an equal number of \( a \)'s, \( b \)'s, and \( c \)'s. So, it cannot lie in \( L_1 \).

Case 2: When either \( v \) or \( y \) contain more than one type of symbol \( uv^2xy^2z \) may contain equal numbers of the three alphabet symbols but will not contain them in correct order. Hence, it does not lie in \( L_1 \).

Example 2. \( L_2 = \{a^i b^j c^k : 0 \leq i \leq j \leq k\} \) is not context free.
Let $p$ be the pumping constant, and let $s = a^p b^p c^p$. Then $s = uvxyz$.

Case 1: When both $v$ and $y$ contain only one type of alphabet symbol, $v$ does not contain both $a$’s and $b$’s or both $b$’s and $c$’s, and the same holds for $y$. Observe that because $v$ and $y$ contain only on type of alphabet symbol, one of the symbols $a, b, or c$ will not appear in either $x$ or $y$. We need to divide this case further according to which symbol does not appear.

(i) The $a$’s do not appear. Then we pump as $uv^0xy^0z = uxz$. This will contain the same number of $a$’s as $s$ does, but it has fewer $b$’s or fewer $c$’s. Hence, it is not in $L_2$.

(ii) The $b$’s do not appear. Then either the $a$’s or $c$’s must appear in $v$ or $y$ because both cannot be the empty string. If the $a$’s appear, the string $uv^2xy^2z$ contains more $a$’s than $b$’s, so it is not in $L_2$. If the $c$’s appear, the string $uv^0xy^0z$ contains more $b$’s than $c$’s, so it too cannot be in $L_2$.

(iii) The $c$’s do not appear. Then the string $uv^2xy^2z$ contains more $a$’s or more $b$’s than $c$’s, so cannot be in $L_2$.

Case 2: When either $v$ or $y$ contains more than one type of symbol, $uv^2xy^2z$ will not contain the symbols in the correct order.

Example 3. $L_3 = \{ww : w \in \{a, b\}^*\}$ is not context free.

Suppose $L_3$ is context free, and let $p$ be the pumping constant. We shall select a special string for pumping: $s = 0^p 1^p 0^p 1^p$. Note: $s = uvxyz$, where $|vxy| \leq p$.

We first observe that the substring $vxy$ must straddle the midpoint of $s$. Otherwise, if the substring occurs only in the first half of $s$, (that is, $vxy$ is a substring of $0^p 1^p$) pumping $s$ to $uv^2xy^2z$ moves the symbol ‘1’ into the first position of the second half, and so it cannot be of the form: $ww$. Similarly, if $vxy$ occurs in the second half of $s$, pumping $s$ to $uv^2xy^2z$ moves the symbol ‘0’ into the last position of the second half, and so it cannot be of the form $ww$. 

9
But if the substring $vx \ y$ straddles the midpoint of $s$, when we pump $s$ down to $uxz$, it forms $0^i1^01^j$, where $i$ and $j$ cannot both be $p$. This string does not have the form $ww$, so it is not in $L_3$. We conclude that $L_3$ cannot be context free.

11. **Closure Properties of Context Free Languages.**

**Theorem.** The class of context free languages are closed under the operations of union, concatenation, and Kleene star. In general, the complement of a context free language is not context free, and the intersection of two context free languages need not be context free.

**Theorem.** The class of context free languages are closed under homomorphism, inverse-image of a homomorphism, and reversal.

**Theorem.** The class of context free languages are NOT closed under intersection and complementation.

The language $L = \{a, b\}^* \setminus \{ww : w \in \{a, b\}^*\}$ is context free.

We can give a grammar for $L$:

- $S \rightarrow AB \mid BA \mid A \mid B$,
- $A \rightarrow CAC \mid a$,
- $B \rightarrow CBC \mid b$,
- $C \rightarrow a \mid b$

This grammar generates all strings of odd length through the initial productions $S \rightarrow A$ and $S \rightarrow B$. Next observe that the production $A \rightarrow CAC \mid a$ generates all strings of the form $xay$, where $|x| = |y|$. Similarly, the production $B \rightarrow CBC \mid b$ generates all strings of the form $ubv$ where $|u| = |v|$. 

10
Further, the rules $S \rightarrow AB \mid BA$ will generate strings of the form $w = xayuvb$ or $w = ubvxy$, where $m = |x| = |y|$ and $n = |u| = |v|$, so $|w| = 2(m + n) + 2$. In particular, the occurrences of $a$ and $b$ are exactly $m + n$ spaces apart. Hence, $w$ cannot be written in the form $w'w'$ for some string $w'$. (To verify this, we need to consider modular arithmetic on the indices of the letters of $w'$ relative to base $m + n + 1$.)

We conclude that this context free grammar generates the desired set complement.