Overview of today’s lecture:
- Normal Forms for Context-Free Languages
- Pumping Lemma for Context-Free Languages

Useful, Useless, Generating and Reachable Symbols

Let \( G = (V, T, R, S) \) be a CFG.
Let \( X \in V \cup T \) and let \( \alpha, \beta \in (V \cup T)^* \).

**Definition:** The symbol \( X \) is **useful** if \( S \Rightarrow^* \alpha X \beta \Rightarrow^* w \) for some \( w \in T^* \).

**Definition:** \( X \) is **useless** iff it is not useful.

**Definition:** \( X \) is **generating** if \( X \Rightarrow^* w \) for some \( w \in T^* \).

**Definition:** \( X \) is **reachable** if \( S \Rightarrow^* \alpha X \beta \).

**Note:** A symbol that is useful should be generating and reachable.

We shall simplify the grammars by eliminating useless symbols.
**Eliminating Useless Symbols**

If we eliminate useless symbols we do not change the language generated by the grammar.

**Note:** It is important in which order we check these conditions.

**Example:** Consider the following grammar

\[
S \rightarrow AB \mid a \\
A \rightarrow b
\]

If we first check for generating symbols and then for reachability we get

\[
S \rightarrow a
\]

If we first check for reachability and then for generating we get

\[
S \rightarrow a \\
A \rightarrow b
\]

**Computing the Generating Symbols**

Let \( G = (V, T, R, S) \) be a CFG.

The following inductive procedure computes the generating symbols of \( G \):

- **Base Case:** All elements of \( T \) are generating.

- **Inductive Step:** If a production \( A \rightarrow \alpha \) is such that all symbols of \( \alpha \) are known to be generating, then \( A \) is also generating.
  
  Observe that \( \alpha \) could be \( \epsilon \).

**Theorem:** The procedure above finds all and only the generating symbols of a grammar.

**Proof:** See Theorem 7.4 in the book.
**Example: Generating Symbols**

Consider the grammar over \{a\} given by the rules:

\[
S \rightarrow aS \mid W \mid U \\
W \rightarrow aW \\
U \rightarrow a \\
V \rightarrow aa
\]

- \(a\) is generating.
- \(U\) and \(V\) are generating since \(U \rightarrow a\) and \(V \rightarrow aa\).
- \(S\) is generating since \(S \rightarrow U\).
- \(W\) is however not generating.

**Computing the Reachable Symbols**

Let \(G = (V, T, R, S)\) be a CFG. The following inductive procedure computes the reachable symbols of \(G\):

**Base Case:** The start variable \(S\) is reachable.

**Inductive Step:** If \(A\) is reachable and we have a production \(A \rightarrow \alpha\) then all symbols in \(\alpha\) are reachable.

**Theorem:** The procedure above finds all and only the reachable symbols of a grammar.

**Proof:** See Theorem 7.6 in the book.
Example: Reachable Symbols

Consider the grammar given by the rules:

\[
S \rightarrow aB \mid BC \\
A \rightarrow aA \mid c \mid aDb \\
B \rightarrow DB \mid C \\
C \rightarrow b \\
D \rightarrow B
\]

\S\ is reachable.
Hence \(a, B\) and \(C\) are reachable.
Then \(b\) and \(D\) are reachable.
However \(A\) and \(c\) are not reachable.

Eliminating Useless Symbols

\textbf{Theorem:} Let \(G = (V, T, R, S)\) be a CFG and let \(L(G) \neq \emptyset\). Let \(G' = (V', T', R', S)\) be constructed as follows:

- Eliminate all non-generating symbols and all productions involving one or more of those symbols;
- In the same way, eliminate now all symbols that are not reachable in the grammar.

Then \(G'\) has no useless symbols and \(L(G) = L(G')\).

\textbf{Proof:} See Theorem 7.2 in the book.
Example: Eliminating Useless Symbols

Consider the grammar given by the rules:

\[
\begin{align*}
S & \to gAe \mid aYB \mid CY \\
A & \to bBY \mid ooC \\
B & \to dd \mid D \\
C & \to jVB \mid gl \\
D & \to n \\
U & \to kW \\
V & \to baXXX \mid oV \\
W & \to c \\
X & \to fV \\
Y & \to Yhm \\
\end{align*}
\]

The simplified grammar is:

\[
\begin{align*}
S & \to gAe \\
A & \to ooC \\
C & \to gl \\
\end{align*}
\]

Nullable Variables

**Definition:** A variable \( A \) is **nullable** if \( A \Rightarrow^* \epsilon \).
Observing that only variables are nullable.

Let \( G = (V, T, R, S) \) be a CFG.
The following inductive procedure computes the nullable variables of \( G \):

**Base Case:** If \( A \to \epsilon \) is a production then \( A \) is nullable.

**Inductive Step:** If \( B \to X_1X_2 \ldots X_k \) is a production and all the \( X_i \) are nullable then \( B \) is also nullable.

**Theorem:** The procedure above finds all and only the nullable variables of a grammar.

**Proof:** See Theorem 7.7 in the book.
Eliminating ɛ-Productions

**Definition:** An ɛ-production is a production of the form $A \rightarrow ɛ$.

Let $G = (V, T, R, S)$ be a CFG.

The following procedure eliminates the ɛ-production of $G$:

1. Determine all nullable variables of $G$;
2. Build $P$ with all the productions of $R$ plus a rule $A \rightarrow \alpha\beta$ whenever we have $A \rightarrow \alpha B\beta$ and $B$ is nullable.
   
   Note: If $A \rightarrow X_1 X_2 \ldots X_k$ and all $X_i$ are nullable, we do not include the case where all the $X_i$ are absent;
3. Construct $G' = (V, T, R', S)$ where $R'$ contains all the productions in $P$ except for the ɛ-productions.

**Theorem:** The grammar $G'$ constructed from the grammar $G$ as above is such that $L(G') = L(G) - \{ɛ\}$.

**Proof:** See Theorem 7.9 in the book.

**Example:** Consider the grammar given by the rules:

$$S \rightarrow aSb \mid SS \mid ɛ$$

By eliminating ɛ-productions we obtain

$$S \rightarrow ab \mid aSb \mid S \mid SS$$

**Example:** Consider the grammar given by the rules:

$$S \rightarrow AB \quad A \rightarrow aAA \mid ɛ \quad B \rightarrow bBB \mid ɛ$$

By eliminating ɛ-productions we obtain

$$S \rightarrow A \mid B \mid AB \quad A \rightarrow a \mid aA \mid aAA \quad B \rightarrow b \mid bB \mid bBB$$
Eliminating Unit Productions

**Definition:** A *unit production* is a production of the form $A \rightarrow B$. This is similar to $\epsilon$-transitions in an $\epsilon$-NFA.

Let $G = (V, T, R, S)$ be a CFG. The following procedure eliminates the unit production of $G$:

1. Build $P$ with all the productions of $R$ plus a rule $A \rightarrow \alpha$ whenever we have $A \rightarrow B$ and $B \rightarrow \alpha$;
2. Construct $G' = (V, T, R', S)$ where $R'$ contains all the productions in $P$ except for the unit production.

**Theorem:** The grammar $G'$ constructed from the grammar $G$ as above is such that $L(G') = L(G)$.

**Proof:** See Theorem 7.13 in the book.

Example: Eliminating Unit Productions

Consider the grammar given by the rules:

$S \rightarrow CBh \mid D$

$A \rightarrow aaC$

$B \rightarrow Sf \mid ggg$

$C \rightarrow cA \mid d \mid C$

$D \rightarrow E \mid SABC$

$E \rightarrow be$

By eliminating unit productions we obtain:

$S \rightarrow CBh \mid be \mid SABC$

$A \rightarrow aaC$

$B \rightarrow Sf \mid ggg$

$C \rightarrow cA \mid d$

$D \rightarrow be \mid SABC$

$E \rightarrow be$
Simplification of a Grammar

**Theorem:** Let \( G = (V, T, R, S) \) be a CFG whose language contains at least one string other than \( \epsilon \). If we construct \( G' \) by

- Eliminating \( \epsilon \)-productions;
- Eliminating unit productions;
- Eliminating useless symbols;

using the procedures shown before then \( L(G') = L(G) - \{\epsilon\} \).

In addition, \( G' \) contains no \( \epsilon \)-productions, no unit productions and no useless symbols.

**Proof:** See Theorem 7.14 in the book.

**Note:** It is important to apply the steps in this order!

---

Chomsky Normal Form

**Definition:** A CFG is in **Chomsky Normal Form** (CNF) if \( G \) has no useless symbols and all the productions are of the form \( A \rightarrow BC \) or \( A \rightarrow a \).

Observe that a CFG that is in CNF has no unit or \( \epsilon \)-productions.

**Theorem:** For any CFG \( G \) whose language contains at least one string other than \( \epsilon \), there is a CFG \( G' \) that is in Chomsky Normal Form and such that \( L(G') = L(G) - \{\epsilon\} \).

**Proof:** See Theorem 7.16 in the book.
Constructing a Chomsky Normal Form

Let us assume $G$ has no $\epsilon$- or unit productions and no useless symbols. Then every production is of the form $A \rightarrow a$ or $A \rightarrow X_1 X_2 \ldots X_k$ for $k > 1$.

If $X_i$ is a terminal introduce a new variable $A_i$ and a new rule $A_i \rightarrow X_i$ (if no such rule exists for $X_i$). Use $A_i$ in place of $X_i$ in any rule whose body has length $> 1$.

Now, all rules are of the form $B \rightarrow b$ or $B \rightarrow C_1 C_2 \ldots C_k$ with all $C_j$ variables.

Introduce $k - 2$ new variables and break each rule $B \rightarrow C_1 C_2 \ldots C_k$ as

$$B \rightarrow C_1 D_1 \quad D_1 \rightarrow C_2 D_2 \quad \ldots \quad D_{k-2} \rightarrow C_{k-1} C_k$$

Example: Chomsky Normal Form

Consider the grammar given by the rules:

$$S \rightarrow aSb \mid SS \mid ab$$

We first obtain

$$S \rightarrow ASB \mid SS \mid AB \quad A \rightarrow a \quad B \rightarrow b$$

Then we build a grammar in Chomsky Normal Form

$$S \rightarrow AC \mid SS \mid AB$$

$$A \rightarrow a$$

$$B \rightarrow b$$

$$C \rightarrow SB$$
Pumping Lemma for Left Regular Languages

Let $G = (V, T, \mathcal{R}, S)$ be a left regular language and let $n = |V|$.

If $a_1a_2 \ldots a_m \in \mathcal{L}(G)$ and $m > n$, then any derivation

$$ S \Rightarrow a_1A_1 \Rightarrow a_1a_2A_2 \Rightarrow \ldots \Rightarrow a_1 \ldots \bar{a_i}A \Rightarrow \ldots \Rightarrow \bar{a_1} \ldots \bar{a_j}A \Rightarrow \ldots \Rightarrow a_1 \ldots a_m $$

has length $m$ and there is at least one variable $A$ which is used twice. (Pigeon-hole principle)

If $x = a_1 \ldots a_i$, $y = a_{i+1} \ldots a_j$ and $z = a_{j+1} \ldots a_m$, we have $|xy| \leq n$ and $x y^k z \in \mathcal{L}(G)$ for all $k$.

Pumping Lemma for Context-Free Languages

**Theorem:** Let $\mathcal{L}$ be a context-free language. Then, there exists a constant $n$ such that if $w \in \mathcal{L}$ with $|w| \geq n$, then we can write $w = xuyvz$ such that

- $|uyv| \leq n$;
- $uv \neq \epsilon$, that is, at least one of $u$ and $v$ is not empty;
- $\forall k \geq 0$, $xu^k yv^k z \in \mathcal{L}$.

**Proof:** (Sketch)

We can assume that the language is presented by a grammar in Chomsky Normal Form, working with $\mathcal{L} - \{\epsilon\}$.

Observe that parse trees for grammars in CNF have at most 2 children. A crucial remark is that if $m + 1$ is the height of a parse tree for $w$, then $|w| \leq 2^m$ (prove this as an exercise!).
Proof Sketch: Pumping Lemma for Context-Free Languages

Let $|V| = m > 0$. Take $n = 2^m$ and $w$ such that $|w| \geq 2^m$.

Any parse tree for $w$ has a path of length at least $m + 1$.

Let $A_0, A_1, \ldots, A_k$ be the variables in the path. We have $k \geq m$.

Then at least 2 of the last $m + 1$ variables should be the same, say $A_i$ and $A_j$.

Observe figures 7.6 and 7.7 in pages 282–283.

See Theorem 7.18 in the book for the complete proof.

Example: Pumping Lemma for Context-Free Languages

Consider the following grammar:

$$
S \rightarrow AC \mid AB \\
A \rightarrow a \\
B \rightarrow b \\
C \rightarrow SB
$$

Consider the derivation for the string $aaaabb bbb$

$$
S \Rightarrow AC \Rightarrow aC \Rightarrow aSB \Rightarrow aACB \Rightarrow aaCB \Rightarrow aaSBB \Rightarrow aaABBB \\
\Rightarrow aaaBBB \Rightarrow aaabBB \Rightarrow aaabbB \Rightarrow aaabbb
$$

Consider the parse tree and the last 2 occurrences of the symbol $S$.
Then we have $x = a$, $u = a$, $y = ab$, $v = b$, $z = b$. 
**Lemma:** The language $L = \{a^m b^m c^m \mid m > 0\}$ is not context-free.

**Proof:** Assume $L$ is context-free. Then we have $n$ as stated in the Pumping lemma.

Consider $w = a^n b^n c^n$. We have that $|w| \geq n$.

So we know that $w = xuyvz$ such that

$$|uyv| \leq n \quad |uv| > 0 \quad \forall k \geq 0, \ xu^k yv^k z \in L$$

Since $|uyv| \leq n$ there is one letter $d \in \{a, b, c\}$ that does not occur in $uyv$. Since $|uv| > 0$ there is another letter $e \in \{a, b, c\}, e \neq d$ that does occur in $uv$.

Then $e$ has more occurrences than $d$ in $xu^2 yv^2 z$ and this contradicts the fact that $xu^2 yv^2 z \in L$. 