Chomsky and Greibach Normal Forms
Simplifying a CFG

- It is often convenient to simplify CFG
Simplifying a CFG

- It is often convenient to simplify CFG
- One of the simplest and most useful simplified forms of CFG is called the Chomsky normal form
Simplifying a CFG

- It is often convenient to simplify CFG
- One of the simplest and most useful simplified forms of CFG is called the Chomsky normal form
- Another normal form usually used in algebraic specifications is Greibach normal form
Simplifying a CFG

- It is often convenient to simplify CFG
- One of the simplest and most useful simplified forms of CFG is called the Chomsky normal form
- Another normal form usually used in algebraic specifications is Greibach normal form

Note the difference between grammar cleaning and simplification
Normal forms are useful when more advanced topics in computation theory are approached, as we shall see further
Definition

A context-free grammar $G$ is in Chomsky normal form if every rule is of the form:

$$A \rightarrow BC$$
$$A \rightarrow a$$

where $a$ is a terminal, $A, B, C$ are nonterminals, and $B, C$ may not be the start variable (the axiom)
The rule \( S \rightarrow \epsilon \), where \( S \) is the start variable, is not excluded from a CFG in Chomsky normal form.
Theorem 2.9

Any context-free language is generated by a context-free grammar in Chomsky normal form.

Proof idea:
Show that any CFG can be converted into a CFG in Chomsky normal form.
Conversion procedure has several stages where the rules that violate Chomsky normal form conditions are replaced with equivalent rules that satisfy these conditions.
Order of transformations:
1. add a new start variable,
2. eliminate all -rules,
3. eliminate unit-rules,
4. convert other rules.
Check that the obtained CFG defines the same language.

Chomsky and Greibach Normal Forms – p. 6/24
Theorem 2.9

Any context-free language is generated by a context-free grammar in Chomsky normal form.

Proof idea:
Theorem 2.9

Any context-free language is generated by a context-free grammar in Chomsky normal form.

Proof idea:

- Show that any CFG $G$ can be converted into a CFG $G'$ in Chomsky normal form.
Theorem 2.9

Any context-free language is generated by a context-free grammar in Chomsky normal form.

Proof idea:

• Show that any CFG $G$ can be converted into a CFG $G'$ in Chomsky normal form

• Conversion procedure has several stages where the rules that violate Chomsky normal form conditions are replaced with equivalent rules that satisfy these conditions
Theorem 2.9

Any context-free language is generated by a context-free grammar in Chomsky normal form.

Proof idea:

• Show that any CFG $G$ can be converted into a CFG $G'$ in Chomsky normal form

• Conversion procedure has several stages where the rules that violate Chomsky normal form conditions are replaced with equivalent rules that satisfy these conditions

• Order of transformations: (1) add a new start variable, (2) eliminate all $\epsilon$-rules, (3) eliminate unit-rules, (4) convert other rules
Theorem 2.9

Any context-free language is generated by a context-free grammar in Chomsky normal form.

Proof idea:

- Show that any CFG $G$ can be converted into a CFG $G'$ in Chomsky normal form
- Conversion procedure has several stages where the rules that violate Chomsky normal form conditions are replaced with equivalent rules that satisfy these conditions
- Order of transformations: (1) add a new start variable, (2) eliminate all $\epsilon$-rules, (3) eliminate unit-rules, (4) convert other rules
- Check that the obtained CFG $G'$ defines the same language
Proof

Let $G = (N, T, R, S)$ be the original CFG.
Proof

Let $G = (N, T, R, S)$ be the original CFG.

Step 1: add a new start symbol $S_0$ to $N$, and the rule $S_0 \rightarrow S$ to $R$.
Proof

Let $G = (N, T, R, S)$ be the original CFG.

Step 1: add a new start symbol $S_0$ to $N$, and the rule $S_0 \longrightarrow S$ to $R$

Note: this change guarantees that the start symbol of $G'$ does not occur on the $rhs$ of any rule
Step 2: eliminate $\epsilon$-rules

Repeat

1. Eliminate the $\epsilon$ rule $A \rightarrow \epsilon$ from $R$ where $A$ is not the start symbol

2. For each occurrence of $A$ on the rhs of a rule, add a new rule to $R$ with that occurrence of $A$ deleted

   Example: replace $B \rightarrow uAv$ by $B \rightarrow uAv|uv$;
   replace $B \rightarrow uAvAw$ by $B \rightarrow uAvAw|uwA|aAvw|uvw$

3. Replace the rule $B \rightarrow A$, (if it is present) by $B \rightarrow A|\epsilon$ unless the rule $B \rightarrow \epsilon$ has been previously eliminated

until all $\epsilon$ rules are eliminated
Step 3: remove unit rules

Repeat

1. Remove a unit rule $A \rightarrow B \in R$

2. For each rule $B \rightarrow u \in R$, add the rule $A \rightarrow u$ to $R$, unless $B \rightarrow u$ was a unit rule previously removed

until all unit rules are eliminated

Note: $u$ is a string of variables and terminals
Convert all remaining rules

Repeat

1. Replace a rule $A \rightarrow u_1 u_2 \ldots u_k$, $k \geq 3$, where each $u_i$, $1 \leq i \leq k$, is a variable or a terminal, by:

   $A \rightarrow u_1 A_1$, $A_1 \rightarrow u_2 A_2$, $\ldots$, $A_{k-2} \rightarrow u_{k-1} u_k$

   where $A_1, A_2, \ldots, A_{k-2}$ are new variables.

2. If $k \geq 2$ replace any terminal $u_i$ with a new variable $U_i$ and add the rule $U_i \rightarrow u_i$

until no rules of the form $A \rightarrow u_1 u_2 \ldots u_k$ with $k \geq 3$ remain
Convert all remaining rules

Repeat

1. Replace a rule \( A \to u_1 u_2 \ldots u_k, \ k \geq 3 \), where each \( u_i, \ 1 \leq i \leq k \), is a variable or a terminal, by:
   \[
   A \to u_1 A_1, \ A_1 \to u_2 A_2, \ldots, \ A_{k-2} \to u_{k-1} u_k
   \]
   where \( A_1, A_2, \ldots, A_{k-2} \) are new variables

2. If \( k \geq 2 \) replace any terminal \( u_i \) with a new variable \( U_i \) and add the rule \( U_i \to u_i \)

until no rules of the form \( A \to u_1 u_2 \ldots u_k \) with \( k \geq 3 \) remain
Convert all remaining rules

Repeat

1. Replace a rule $A \rightarrow u_1 u_2 \ldots u_k$, $k \geq 3$, where each $u_i$, $1 \leq i \leq k$, is a variable or a terminal, by:
   
   $A \rightarrow u_1 A_1$, $A_1 \rightarrow u_2 A_2$, \ldots, $A_{k-2} \rightarrow u_{k-1} u_k$
   
   where $A_1, A_2, \ldots, A_{k-2}$ are new variables

2. If $k \geq 2$ replace any terminal $u_i$ with a new variable $U_i$ and add the rule $U_i \rightarrow u_i$

until no rules of the form $A \rightarrow u_1 u_2 \ldots u_k$ with $k \geq 3$ remain
Consider the grammar $G_6$ whose rules are:

\[
S \rightarrow ASA|aB \\
A \rightarrow B|S \\
B \rightarrow b|\epsilon
\]

**Notation:** symbols removed are green and those added are red.

After first step of transformation we get:

\[
S_0 \rightarrow S \\
S \rightarrow ASA|aB \\
A \rightarrow B|S \\
B \rightarrow b|\epsilon
\]
Example CFG conversion

Consider the grammar $G_6$ whose rules are:

\[
\begin{align*}
S & \rightarrow ASAaB \\
A & \rightarrow B|S \\
B & \rightarrow b|\epsilon
\end{align*}
\]

**Notation:** symbols removed are green and those added are red.

After first step of transformation we get:

\[
\begin{align*}
S_0 & \rightarrow S \\
S & \rightarrow ASAaB \\
A & \rightarrow B|S \\
B & \rightarrow b|\epsilon
\end{align*}
\]
Example CFG conversion

Consider the grammar $G_6$ whose rules are:

\[
S \rightarrow ASA|aB \\
A \rightarrow B|S \\
B \rightarrow b|\epsilon
\]

**Notation:** symbols removed are green and those added are red.

After first step of transformation we get:

\[
S_0 \rightarrow S \\
S \rightarrow ASA|aB \\
A \rightarrow B|S \\
B \rightarrow b|\epsilon
\]
Removing $\varepsilon$ rules

Removing $B \rightarrow \varepsilon$:

\[
\begin{align*}
S_0 & \rightarrow S \\
S & \rightarrow ASA|aB|a \\
A & \rightarrow B|S|\varepsilon \\
B & \rightarrow b|\varepsilon
\end{align*}
\]

Removing $A \rightarrow \varepsilon$:

\[
\begin{align*}
S_0 & \rightarrow S \\
S & \rightarrow ASA|aB|a|SA|AS|S \\
A & \rightarrow B|S|\varepsilon \\
B & \rightarrow b
\end{align*}
\]
Removing $\epsilon$ rules

Removing $B \rightarrow \epsilon$:

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

Removing $A \rightarrow \epsilon$:

\[
\begin{align*}
S_0 & \rightarrow S \\
S & \rightarrow ASA|aB|a|SA|AS|S \\
A & \rightarrow B|S|\epsilon \\
B & \rightarrow b
\end{align*}
\]
Removing $\varepsilon$ rules

Removing $B \rightarrow \varepsilon$:

\[
\begin{align*}
S_0 & \rightarrow S \\
S & \rightarrow ASA|aB|a \\
A & \rightarrow B|S|\varepsilon \\
B & \rightarrow b|\varepsilon
\end{align*}
\]

Removing $A \rightarrow \varepsilon$:

\[
\begin{align*}
S_0 & \rightarrow S \\
S & \rightarrow ASA|aB|a|SA|AS|S \\
A & \rightarrow B|S|\varepsilon \\
B & \rightarrow b
\end{align*}
\]
Removing unit rule

Removing $S \rightarrow S$:

$S_0 \rightarrow S$
$S \rightarrow ASA|aB|a|S|A|S|S$
$A \rightarrow B|S$
$B \rightarrow b$

Removing $S_0 \rightarrow S$:

$S_0 \rightarrow S|ASA|aB|a|S|A|S$
$S \rightarrow ASA|aB|a|S|A|S$
$A \rightarrow B|S$
$B \rightarrow b$
Removing unit rule

Removing $S \rightarrow S$:

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

Removing $S_0 \rightarrow S$:

\[
\begin{align*}
S_0 & \rightarrow S|ASA|aB|a|S A|AS \\
S & \rightarrow ASA|aB|a|S A|AS \\
A & \rightarrow B|S \\
B & \rightarrow b
\end{align*}
\]
Removing unit rule

Removing $S \rightarrow S$:

$S_0 \rightarrow S$
$S \rightarrow ASA|aB|a|SA|AS|S$
$A \rightarrow B|S$
$B \rightarrow b$

Removing $S_0 \rightarrow S$:

$S_0 \rightarrow S|ASA|aB|a|SA|AS$
$S \rightarrow ASA|aB|a|SA|AS$
$A \rightarrow B|S$
$B \rightarrow b$
More unit rules

Removing $A \rightarrow B$:

$S_0 \rightarrow ASA|aB|a|SA|AS$

$S \rightarrow ASA|aB|a|SA|AS$

$A \rightarrow B|S|b$

$B \rightarrow b$

Removing $A \rightarrow S$:

$S_0 \rightarrow ASA|aB|a|SA|AS$

$S \rightarrow ASA|aB|a|SA|AS$

$A \rightarrow S|b|ASA|aB|a|SA|AS$

$B \rightarrow b$
More unit rules

Removing $A \rightarrow B$:

\begin{align*}
S_0 & \rightarrow ASA|aB|a|SA|AS \\
S & \rightarrow ASA|aB|a|SA|AS \\
A & \rightarrow B|S|b \\
B & \rightarrow b
\end{align*}

Removing $A \rightarrow S$:

\begin{align*}
S_0 & \rightarrow ASA|aB|a|SA|AS \\
S & \rightarrow ASA|aB|a|SA|AS \\
A & \rightarrow S|b|ASA|aB|a|SA|AS \\
B & \rightarrow b
\end{align*}
More unit rules

Removing $A \rightarrow B$:

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

Removing $A \rightarrow S$:

\[
\begin{align*}
S_0 & \rightarrow ASA|aB|a|SA|AS \\
S & \rightarrow ASA|aB|a|SA|AS \\
A & \rightarrow S|b|ASA|aB|a|SA|AS \\
B & \rightarrow b
\end{align*}
\]
Converting remaining rules

\[
S_0 \rightarrow AA_1 | UB | a | SA | AS \\
S \rightarrow AA_1 | UB | a | SA | AS \\
A \rightarrow b | AA_1 | UB | a | SA | AS \\
A_1 \rightarrow SA \\
U \rightarrow a \\
B \rightarrow b
\]
Converting remaining rules

\[ S_0 \rightarrow AA_1|UB|a|SA|AS \]
\[ S \rightarrow AA_1|UB|a|SA|AS \]
\[ A \rightarrow b|AA_1|UB|a|SA|AS \]
\[ A_1 \rightarrow SA \]
\[ U \rightarrow a \]
\[ B \rightarrow b \]
Converting remaining rules

\[
\begin{align*}
S_0 & \rightarrow AA_1 | UB | a | SA | AS \\
S & \rightarrow AA_1 | UB | a | SA | AS \\
A & \rightarrow b | AA_1 | UB | a | SA | AS \\
A_1 & \rightarrow SA \\
U & \rightarrow a \\
B & \rightarrow b
\end{align*}
\]
The conversion procedure produces several variables $U_i$ along with several rules $U_i \rightarrow a$.

Since all these represent the same rule, we may simplify the result using a single variable $U$ and a single rule $U \rightarrow a$. 
The conversion procedure produces several variables $U_i$ along with several rules $U_i \rightarrow a$. Since all these represent the same rule, we may simplify the result using a single variable $U$ and a single rule $U \rightarrow a$. 
Note

- The conversion procedure produces several variables $U_i$ along with several rules $U_i \rightarrow a$.
- Since all these represent the same rule, we may simplify the result using a single variable $U$ and a single rule $U \rightarrow a$. 
A context-free grammar \( G = (V, \Sigma, R, S) \) is in Greibach normal form if each rule \( r \in R \) has the property: \( \text{lhs}(r) \in V, \text{rhs}(r) = a\alpha, a \in \Sigma \) and \( \alpha \in V^* \).

**Note:** Greibach normal form provides a justification of operator prefix notation usually employed in algebra.
A context-free grammar $G = (V, \Sigma, R, S)$ is in Greibach normal form if each rule $r \in R$ has the property: $lhs(r) \in V$, $rhs(r) = a\alpha$, $a \in \Sigma$ and $\alpha \in V^*$. 

**Note:** Greibach normal form provides a justification of operator prefix-notation usually employed in algebra.
A context-free grammar $G = (V, \Sigma, R, S)$ is in Greibach normal form if each rule $r \in R$ has the property: $lhs(r) \in V$, $rhs(r) = a\alpha$, $a \in \Sigma$ and $\alpha \in V^*$. 

Note: Greibach normal form provides a justification of operator prefix notation usually employed in algebra.
Greibach Theorem

Every CFL $L$ where $\epsilon \notin L$ can be generated by a CFG in Greibach normal form.

Proof idea: Let $G = (V, \Sigma, R, S)$ be a CFG generating $L$. Assume that $G$ is in Chomsky normal form

- Let $V = \{A_1, A_2, \ldots, A_m\}$ be an ordering of nonterminals.
- Construct the Greibach normal form from Chomsky normal form
Every CFL $L$ where $\epsilon \notin L$ can be generated by a CFG in Greibach normal form.

**Proof idea:** Let $G = (V, \Sigma, R, S)$ be a CFG generating $L$. Assume that $G$ is in Chomsky normal form

- Let $V = \{A_1, A_2, \ldots, A_m\}$ be an ordering of nonterminals.
- Construct the Greibach normal form from Chomsky normal form.
Greibach Theorem

Every CFL $L$ where $\epsilon \not\in L$ can be generated by a CFG in Greibach normal form.

Proof idea: Let $G = (V, \Sigma, R, S)$ be a CFG generating $L$. Assume that $G$ is in Chomsky normal form

- Let $V = \{A_1, A_2, \ldots, A_m\}$ be an ordering of nonterminals.
- Construct the Greibach normal form from Chomsky normal form
1. Modify the rules in $R$ so that if $A_i \rightarrow A_j \gamma \in R$ then $j > i$

2. Starting with $A_1$ and proceeding to $A_m$ this is done as follows:
   
   (a) Assume that productions have been modified so that for $1 \leq i \leq k$, $A_i \rightarrow A_j \gamma \in R$ only if $j > i$
   
   (b) If $A_k \rightarrow A_j \gamma$ is a production with $j < k$, generate a new set of productions substituting for the $A_j$ the rhs of each $A_j$ production
   
   (c) Repeating (b) at most $k - 1$ times we obtain rules of the form $A_k \rightarrow A_p \gamma$, $p \geq k$
   
   (d) Replace rules $A_k \rightarrow A_k \gamma$ by removing left-recursive rules
Construction

1. Modify the rules in $R$ so that if $A_i \rightarrow A_j \gamma \in R$ then $j > i$

2. Starting with $A_1$ and proceeding to $A_m$ this is done as follows:
   (a) Assume that productions have been modified so that for $1 \leq i \leq k$, $A_i \rightarrow A_j \gamma \in R$ only if $j > i$
   (b) If $A_k \rightarrow A_j \gamma$ is a production with $j < k$, generate a new set of productions substituting for the $A_j$ the rhs of each $A_j$ production
   (c) Repeating (b) at most $k - 1$ times we obtain rules of the form $A_k \rightarrow A_p \gamma$, $p \geq k$
   (d) Replace rules $A_k \rightarrow A_k \gamma$ by removing left-recursive rules
Construction

1. Modify the rules in $R$ so that if $A_i \rightarrow A_j \gamma \in R$ then $j > i$

2. Starting with $A_1$ and proceeding to $A_m$ this is done as follows:
   (a) Assume that productions have been modified so that for $1 \leq i \leq k$, $A_i \rightarrow A_j \gamma \in R$ only if $j > i$
   (b) If $A_k \rightarrow A_j \gamma$ is a production with $j < k$, generate a new set of productions substituting for the $A_j$ the rhs of each $A_j$ production
   (c) Repeating (b) at most $k - 1$ times we obtain rules of the form $A_k \rightarrow A_p \gamma$, $p \geq k$
   (d) Replace rules $A_k \rightarrow A_k \gamma$ by removing left-recursive rules
Removing left-recursion

Left-recursion can be eliminated by the following scheme:

• If $A \rightarrow A\alpha_1|A\alpha_2 \ldots |A\alpha_r$ are all $A$ left recursive rules, and $A \rightarrow \beta_1|\beta_2| \ldots |\beta_s$ are all remaining $A$-rules then chose a new nonterminal, say $B$

• Add the new $B$-rules $B \rightarrow \alpha_i|\alpha_iB, 1 \leq i \leq r$

• Replace the $A$-rules by $A \rightarrow \beta_i|\beta_iB, 1 \leq i \leq s$

This construction preserve the language $L$. 
Removing left-recursion

Left-recursion can be eliminated by the following scheme:

- If $A \to A\alpha_1 | A\alpha_2 \ldots | A\alpha_r$ are all $A$ left recursive rules, and $A \to \beta_1 | \beta_2 | \ldots | \beta_s$ are all remaining $A$-rules then chose a new nonterminal, say $B$
- Add the new $B$-rules $B \to \alpha_i | \alpha_i B$, $1 \leq i \leq r$
- Replace the $A$-rules by $A \to \beta_i | \beta_i B$, $1 \leq i \leq s$

This construction preserves the language $L$. 
Removing left-recursion

Left-recursion can be eliminated by the following scheme:

- If $A \rightarrow A\alpha_1 | A\alpha_2 \ldots | A\alpha_r$ are all $A$ left recursive rules, and $A \rightarrow \beta_1 | \beta_2 | \ldots | \beta_s$ are all remaining $A$-rules then chose a new nonterminal, say $B$
- Add the new $B$-rules $B \rightarrow \alpha_i | \alpha_i B$, $1 \leq i \leq r$
- Replace the $A$-rules by $A \rightarrow \beta_i | \beta_i B$, $1 \leq i \leq s$

This construction preserve the language $L$. 
More on Greibach NF

See Introduction to Automata Theory, Languages, and Computation, J.E, Hopcroft and J.D Ullman, Addison-Wesley 1979, p. 94–96
Example

Convert the CFG

\[ G = (\{A_1, A_2, A_3\}, \{a, b\}, R, A_1) \]

where

\[ R = \{A_1 \to A_2 A_3, A_2 \to A_3 A_1 | b, A_3 \to A_1 A_2 | a\} \]

into Greibach normal form.
Example

Convert the CFG

\[ G = (\{A_1, A_2, A_3\}, \{a, b\}, R, A_1) \]

where

\[ R = \{ A_1 \rightarrow A_2 A_3, A_2 \rightarrow A_3 A_1|b, A_3 \rightarrow A_1 A_2|a \} \]

into Greibach normal form.
Example

Convert the CFG

\[ G = (\{A_1, A_2, A_3\}, \{a, b\}, R, A_1) \]

where

\[ R = \{A_1 \rightarrow A_2 A_3, A_2 \rightarrow A_3 A_1|b, A_3 \rightarrow A_1 A_2|a\} \]

into Greibach normal form.
1. **Step 1**: ordering the rules: (Only $A_3$ rules violate ordering conditions, hence only $A_3$ rules need to be changed). Following the procedure we replace $A_3$ rules by:

   $A_3 \rightarrow A_3A_1A_3A_2|bA_3A_2|a$

2. Eliminating left-recursion we get: $A_3 \rightarrow bA_3A_2B_3|aB_3|bA_3A_2|a$, $B_3 \rightarrow A_1A_3A_2|A_1A_3A_2B_3$

3. All $A_3$ rules start with a terminal. We use them to replace $A_1 \rightarrow A_2A_3$. This introduces the rules $B_3 \rightarrow A_1A_3A_2|A_1A_3A_2B_3$

4. Use $A_1$ production to make them start with a terminal
1. **Step 1:** ordering the rules: (Only $A_3$ rules violate ordering conditions, hence only $A_3$ rules need to be changed). Following the procedure we replace $A_3$ rules by:

   \[ A_3 \rightarrow A_3A_1A_3A_2|bA_3A_2|a \]

2. Eliminating left-recursion we get: \[ A_3 \rightarrow bA_3A_2B_3|aB_3|bA_3A_2|a, \]

   \[ B_3 \rightarrow A_1A_3A_2|A_1A_3A_2B_3 \]

3. All $A_3$ rules start with a terminal. We use them to replace $A_1 \rightarrow A_2A_3$. This introduces the rules $B_3 \rightarrow A_1A_3A_2|A_1A_3A_2B_3$

4. Use $A_1$ production to make them start with a terminal.
Solution

1. **Step 1**: ordering the rules: (Only $A_3$ rules violate ordering conditions, hence only $A_3$ rules need to be changed).
   Following the procedure we replace $A_3$ rules by:
   
   $A_3 \rightarrow A_3A_1A_3A_2 | bA_3A_2 | a$

2. Eliminating left-recursion we get:
   
   $A_3 \rightarrow bA_3A_2B_3 | aB_3 | bA_3A_2 | a,$
   
   $B_3 \rightarrow A_1A_3A_2 | A_1A_3A_2B_3$

3. All $A_3$ rules start with a terminal. We use them to replace
   
   $A_1 \rightarrow A_2A_3$. This introduces the rules
   
   $B_3 \rightarrow A_1A_3A_2 | A_1A_3A_2B_3$

4. Use $A_1$ production to make them start with a terminal