Formal Languages

Simplifications of CFGs

Module – 3	
CONTEXT-FREE LANGUAGES AND SIMPLIFICATION OF CONTEXT-FREE GRAMMARS AND NORMAL FORMS: Context-Free grammars, Parsing and Ambiguity, Methods for Transforming Grammars, Two important Normal Forms. **Text Book 1:** Chapter 5: 5.1 -5.2, Chapter 6: 6.1 – 6.2	**05 Hours**

- Why Simplification….?
- By removing the productions of the form

  $A \rightarrow \lambda$

  $A \rightarrow B$

We can make the process easier

- In this chapter we study several transformations and substitutions that will be useful in subsequent discussions.

- We also investigate **normal forms** for context-free grammars.
- A normal form is a grammatical form, so that any Context Free grammar has an equivalent normal-form version. We introduce two of the most useful of these, the **Chomsky normal form** and the **Greibach normal form**.

A Useful Substitution Rule

Let $G = (V, T, S, P)$ be a context-free grammar. Suppose that P contains a production of the form

$$A \to x_1 B x_2.$$

Assume that A and B are different variables and that

$$B \to y_1 \,|y_2\,| \ldots |y_n$$

is the set of all productions in P that have B as the left side. Let $\widehat{G} = (V, T, S, \widehat{P})$ be the grammar in which $\widehat{P}$ is constructed by deleting

$$A \to x_1 B x_2 \qquad (6.1)$$

from P, and adding to it

$$A \to x_1 y_1 x_2 \,|x_1 y_2 x_2|\, \cdots \,|x_1 y_n x_2.$$

then $L\left(\widehat{G}\right) = L(G).$

Example 1:

Consider $G = (\{A, B\}, \{a,b,c\}, A, P)$ with productions

$$A \to a \,|aaA|\, abBc,$$
$$B \to abbA|b.$$

Using the suggested substitution for the variable B, we get the grammar $\hat{G}$ with productions

$$A \to a \,|aaA|\, ababbAc|abbc,$$
$$B \to abbA|b.$$

The new grammar $\hat{G}$ is equivalent to G. The string $aaabbc$ has the derivation

$$A \Rightarrow aaA \Rightarrow aaabBc \Rightarrow aaabbc$$

in G, and the corresponding derivation

$$A \Rightarrow aaA \Rightarrow aaabbc \quad \text{in } \hat{G}.$$

Theorem 6.5

Let L be a context-free language that does not contain λ. Then there exists a context-free grammar that generates L and that does not have any useless productions, λ-productions, or unit-productions.

We can remove all undesirable productions using the following sequence of steps:

1. Remove λ-productions.
2. Remove unit-productions.
3. Remove useless productions.

Removing Useless Productions

$$S \rightarrow aSb \,|\lambda|\, A,$$
$$A \rightarrow aA,$$

the production $S \rightarrow A$ clearly plays no role, as A cannot be transformed into a terminal string. While A can occur in a string derived from S, this can never lead to a sentence. Removing this production leaves the language unaffected and is a simplification by any definition.

Definition 6.1

Let $G = (V, T, S, P)$ be a context-free grammar. A variable $A \in V$ is said to be **useful** if and only if there is at least one $\omega \in L(G)$ such that

$$S \overset{*}{\Rightarrow} xAy \overset{*}{\Rightarrow} w,$$

with x, y in $(V \cup T)^*$. In words, a variable is useful if and only if it occurs in at least one derivation. A variable that is not useful is called **useless**. A production is useless if it involves any useless variable.

Exampe 2:

$$S \rightarrow A,$$
$$A \rightarrow aA|\lambda,$$
$$B \rightarrow bA,$$

the variable B is useless and so is the production $B \rightarrow bA$. Although B can derive a terminal string, there is no way we can achieve $S \stackrel{*}{\Rightarrow} xBy$.

The two reasons why a variable is useless:

→ It cannot be reached from the start symbol

→ It cannot derive a terminal string.

Example 3: Eliminating useless productions

Eliminate useless symbols and productions from $G = (V,T,S,P)$, where $V = \{S, A, B, C\}$ and $T = \{a, b\}$, with P consisting of

$$S \to aS \,|A|\, C,$$
$$A \to a,$$
$$B \to aa,$$
$$C \to aCb.$$

→ First, we identify the set of variables that can lead to a terminal string.

→ $A \to a$ and $B \to aa$, the variables A and B belong to this set. So does S, because $S \Rightarrow A \Rightarrow a$. All leads to terminals.

→ Here C is useless bcos it is not giving a sentence.

S → C C → aCb (remains in sentential form…)

Removing C and its corresponding productions, we are led to the grammar G_1 with variables $V_1 = \{S, A, B\}$, terminals $T = \{a\}$, and productions

$$S \to aS|A,$$
$$A \to a,$$
$$B \to aa.$$

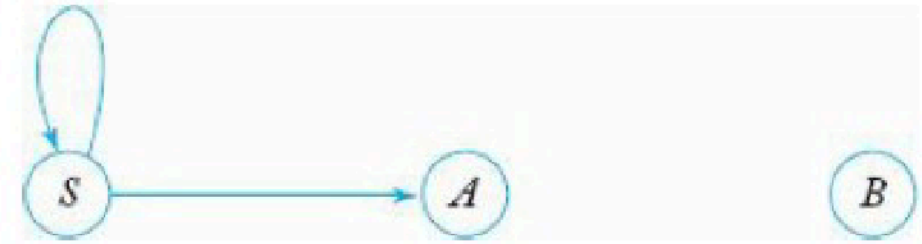
→ Next we want to eliminate the variables that cannot be reached from the start variable. For this,

→ We can draw a **dependency graph** for the variables.

→ A variable is useful only if there is a path from the vertex labeled S to the vertex labeled with that variable

Dependency graph for the production

$$S \to aS|A,$$
$$A \to a,$$
$$B \to aa.$$



So after removing the useless production we get

$\widehat{G} = \left(\widehat{V}, \widehat{T}, S, \widehat{P}\right)$ with $\widehat{V} = \{S, A\}, \widehat{T} = \{a\}$, and productions

$$S \to aS|A,$$
$$A \to a.$$

Removing λ-Productions

Definition 6.2

Any production of a context-free grammar of the form

$$A \rightarrow \lambda$$

is called a **λ-production**. Any variable A for which the derivation

$$A \stackrel{*}{\Rightarrow} \lambda$$

is possible is called **nullable**.

A grammar may generate a language not containing λ yet have some λ- productions or nullable variables. In such cases, the λ-productions can be removed.

Consider the grammar

$$S \rightarrow aS_1b,$$
$$S_1 \rightarrow aS_1b|\lambda,$$

with start variable S. This grammar generates the λ-free language $\{a^nb^n : n \geq 1\}$. The λ-production $S_1 \rightarrow \lambda$ can be removed after adding new productions obtained by substituting λ for S_1 where it occurs on the right. Doing this we get the grammar

$$S \rightarrow aS_1b|ab,$$
$$S_1 \rightarrow aS_1b|ab.$$

Example 4:

Find a context-free grammar without λ-productions equivalent to the grammar defined by

$$S \to ABaC,$$
$$A \to BC,$$
$$B \to b|\lambda,$$
$$C \to D|\lambda,$$
$$D \to d.$$

→ Find all nullable variables

→ Here we have B → λ, C → λ, A → λ

→ So A, B and C are nullable variables

After removing λ productions using substitution rule the new grammar is given as

$$
\begin{aligned}
S &\rightarrow ABaC\,|BaC|\,AaC\,|ABa|\,aC\,|Aa|\,Ba|a,\\
A &\rightarrow B\,|C|\,BC,\\
B &\rightarrow b,\\
C &\rightarrow D,\\
D &\rightarrow d.
\end{aligned}
$$

Removing Unit-Productions

Any production of a context-free grammar of the form

$$A \to B,$$

where $A, B \in V$, is called a **unit-production.**

Theorem 6.4

Let $G = (V, T, S, P)$ be any context-free grammar without λ-productions. Then there exists a context-free grammar $\hat{G} = (\hat{V}, \hat{T}, S, \hat{P})$ that does not have any unit-productions and that is equivalent to G.

Proof: Obviously, any unit-production of the form $A \rightarrow A$ can be removed from the grammar without effect, and we need only consider $A \rightarrow B$, where A and B are different variables.

→The new grammar is generated by first putting into all non-unit productions of P.

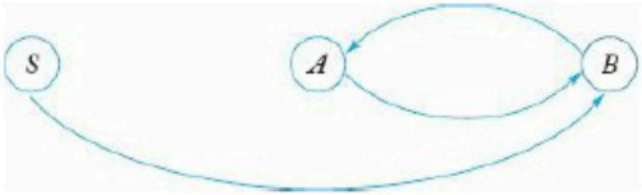
→T lency graph only with unit productions.

$$S \rightarrow Aa|B,$$
$$B \rightarrow A|bb,$$
$$A \rightarrow a|bc|B.$$

S → B

B → A

A → B



Example 5:

Remove all unit-productions from

$$S \to Aa|B,$$
$$B \to A|bb,$$
$$A \to a|bc|B.$$

The dependency graph for the unit-productions is given in Figure . we see from it that $S \overset{*}{\Rightarrow} A, S \overset{*}{\Rightarrow} B, B \overset{*}{\Rightarrow} A$, and $A \overset{*}{\Rightarrow} B$. Hence, we add to the original non-unit productions

$$S \to Aa,$$
$$A \to a|bc,$$
$$B \to bb,$$

After including non unit productions we get a new grammar

$$S \to a|bc|bb|Aa,$$
$$A \to a|bb|bc,$$
$$B \to a|bb|bc.$$

Theorem 6.5

Let L be a context-free language that does not contain λ. Then there exists a context-free grammar that generates L and that does not have any useless productions, λ-productions, or unit-productions.

We can remove all undesirable productions using the following sequence of steps:

1. Remove λ-productions.
2. Remove unit-productions.
3. Remove useless productions.

Example 6:

Eliminate useless productions from the following

1)

S → aA | bB
A → aA | a
B → bB
D → ab | Ea
E → aC | d

Productions Only with terminals

2) A → a
D → ab
E → d

S also derives terminal thru A and not thru B

3) S → aA (remove B)
A → aA |a
D → ab| E
E → aC |d

not reachable from A

4) S → aA
A → a | aA

Example 6:

Eliminate λ productions from the following

1)

S → BAAB
A → 0A2| 2A0 | λ
B → AB | 1B | λ

3)

S → BAAB | AAB| BAB| BAA|
AB| BB| BA| AA| B | A

A → 0A2| 2A0| 02| 20

B → AB| 1B| A| 1

2)

S → BAAB | AAB| BAB| BAA| AB| BB| BA| AA| B | A
A → 0A2| 2A0| 02| 20
B → AB| 1B| A|B| 1 can be removed

Example 7:

Eliminate unit productions from the following

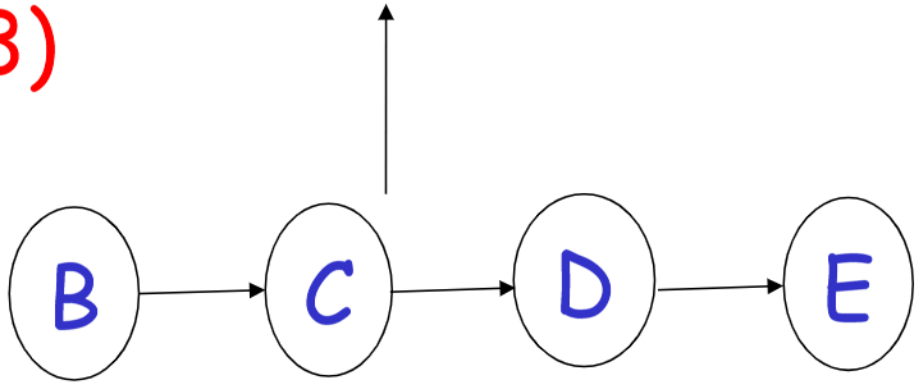
1)

S → AB
A → a
B → C | b
C → D
D → E | bC
E → d | Ab

2)

B → C
C → D
D → E

3)



4)

S→ AB
A→ a
B → b | bC | d | Ab
C → bC | d | Ab
D → d | Ab| bC
E → d | Ab

Remove all unit, useless and λ

1) $S \rightarrow aA \mid aBB$
$A \rightarrow aaA \mid \lambda$
$B \rightarrow bC \mid bbC$
$C \rightarrow B$

2) $S \rightarrow aA$
$A \rightarrow BB$
$B \rightarrow aBb \mid \lambda$

Two Important Normal Forms

- **Chomsky Normal Form**
- **Greibach Normal Form**

Chomsky Normal Form (CNF) and Greibach Normal Form (GNF) are two standardized forms for representing context-free grammars (CFGs), simplifying parsing and analysis in formal language theory.

CNF simplifies grammars by restricting the form of production rules, making them easier to analyze and parse. CNF is particularly useful for bottom-up parsing algorithm.

CNF is often used in the analysis of algorithms for parsing and language recognition, while GNF is used in the construction of parsers, especially simple top-down parsers for specific language processing tasks.

GNF is well-suited for top-down parsing algorithms and compiler design.

Chomsky Normal Form

- In this CNF , the number of symbols on the right of a production is strictly limited.
- The string on the right of a production consist of no more than two symbols

Definition 6.4

A context-free grammar is in Chomsky normal form if all productions are of the form

$$A \to BC$$

or

$$A \to a,$$

where A, B, C are in V, and a is in T.

Example 1:

The grammar

$$S \rightarrow AS|a,$$
$$A \rightarrow SA|b$$

is in Chomsky normal form. The grammar

$$S \rightarrow AS|AAS,$$
$$A \rightarrow SA|aa$$

is not; both productions $S \rightarrow AAS$ and $A \rightarrow aa$ violate the conditions of Definition

Any context-free grammar $G = (V, T, S, P)$ with $\lambda \notin L(G)$ has an equivalent grammar in Chomsky normal form.

Proof: Because of Theorem 6.5, we can assume without loss of generality that G has no λ- productions and no unit-productions. The construction of will be done in two steps.

Step 1: Construct a grammar $G_1 = (V_1, T, S, P_1)$ from G by considering all productions in P in the form

$$A \to x_1 x_2 \cdots x_n, \tag{6.5}$$

where each x_i is a symbol either in V or in T. If $n = 1$, then x_1 must be a terminal since we have no unit-productions. In this case, put the production into P_1. If $n \geq 2$, introduce new variables Ba for each $a \in T$. For each production of P in the form (6.5) we put into P_1 the production

$$A \to C_1 C_2 \ldots C_n,$$

where

$$C_i = x_i \text{ if } x_i \text{ is in } V,$$

and

$$C_i = B_a \text{ if } x_i = a.$$

For every B_a we also put into P_1 the production

$$B_a \to a.$$

→ This part of the algorithm removes all terminals from productions whose right side has length greater than one, replacing them with newly introduced variables.

→ At the end of this step we have a grammar $G1$ all of whose productions have the form

$$A \to a,$$

or

$$A \to C_1 C_2 \cdots C_n,$$

where $C_i \in V_1$.

Step 2: In the second step, we introduce additional variables to reduce the length of the right sides of the productions where necessary. First we put all productions of the form (6.6) as well as all the productions of the form (6.7) with $n = 2$ into $\hat{P}$. For $n \geq 2$, we introduce new variables D_1, D_2,...and put into $\hat{P}$ the productions

$$\begin{aligned} A &\rightarrow C_1 D_1, \\ D_1 &\rightarrow C_2 D_2, \\ &\vdots \\ D_{n-2} &\rightarrow C_{n-1} C_n. \end{aligned}$$

Obviously, the resulting grammar $\hat{G}$ is in Chomsky normal form. Repeated applications of Theorem 6.1 will show that $L(G_1) = L(\hat{G})$, so that

$$L(\hat{G}) = L(G).$$

Example 2:
Convert the grammar with productions

$$S \to ABa,$$
$$A \to aab,$$
$$B \to Ac$$

to Chomsky normal form.

In Step 1, we introduce new variables B_a, B_b, B_c and use the algorithm to get

$$S \to ABB_a,$$
$$A \to B_aB_aB_b,$$
$$B \to AB_c,$$
$$B_a \to a,$$
$$B_b \to b,$$
$$B_c \to c.$$

In the second step, we introduce additional variables to get the first two productions into normal form and we get the final result

$$
\begin{aligned}
S &\to AD_1, \\
D_1 &\to BB_a, \\
A &\to B_aD_2, \\
D_2 &\to B_aB_b, \\
B &\to AB_c, \\
B_a &\to a, \\
B_b &\to b, \\
B_c &\to c.
\end{aligned}
$$

Greibach Normal Form

Another useful grammatical form is the **Greibach normal form.** Here we put restrictions not on the length of the right sides of a production, but on the positions in which terminals and variables can appear.

Definition 6.5

A context-free grammar is said to be in Greibach normal form if all productions have the form $A \to ax$, where $a \in T$ and $x \in V^*$

The grammar

$$\begin{aligned} S &\to AB, \\ A &\to aA \,|bB|\, b, \\ B &\to b \end{aligned}$$

is not in Greibach normal form. However, using the substitution given by Theorem 6.1, we immediately get the equivalent grammar

$$\begin{aligned} S &\to aAB \,|bBB|\, bB, \\ A &\to aA \,|bB|\, b, \\ B &\to b, \end{aligned}$$

which is in Greibach normal form.

Example 3:

Convert the grammar

$$S \rightarrow abSb|aa$$

into Greibach normal form.

Here we can use a device similar to the one introduced in the construction of Chomsky normal form. We introduce new variables A and B that are essentially synonyms for a and b, respectively. Substituting for the terminals with their associated variables leads to the equivalent grammar

$$\begin{aligned} S &\rightarrow aBSB|aA, \\ A &\rightarrow a, \\ B &\rightarrow b, \end{aligned}$$

which is in Greibach normal form.

Convert the following to CNF

1] $S \rightarrow 0A \mid 1B$
$A \rightarrow 0AA \mid 1S \mid 1$
$B \rightarrow 1BB \mid 0S \mid 0$

2] $S \rightarrow AB \mid aB$
$A \rightarrow aab \mid \lambda$
$B \rightarrow bbA$

3] $S \rightarrow abAB$
$A \rightarrow bAB \mid \lambda$
$B \rightarrow Baa \mid A \mid \lambda$

Convert the following to GNF

1] $S \rightarrow aSb \mid bSa \mid a \mid b$

2] $S \rightarrow aSb \mid a \mid b$

3] $S \rightarrow ab \mid aS \mid aaS$

4] $S \rightarrow ABb \mid a$
$A \rightarrow aaA \mid B$
$B \rightarrow bAb$

Slides downloaded from peter linz
just go thru all the slides

A Substitution Rule

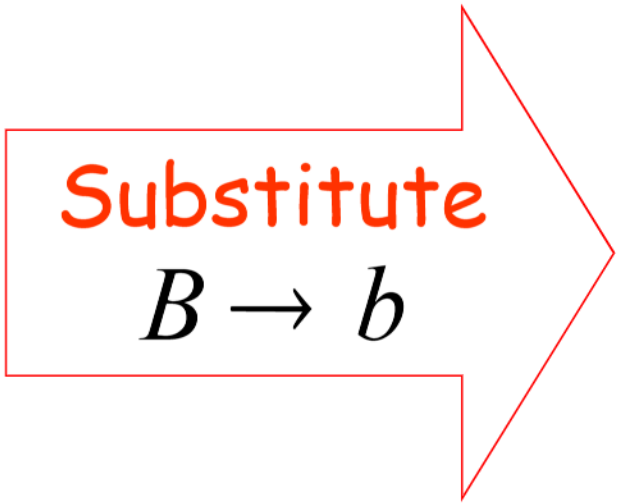
$S \rightarrow aB$

$A \rightarrow aaA$

$A \rightarrow abBc$

$B \rightarrow aA$

$B \rightarrow b$



Equivalent grammar

$S \rightarrow aB \mid ab$

$A \rightarrow aaA$

$A \rightarrow abBc \mid abbc$

$B \rightarrow aA$

A Substitution Rule

$$S \rightarrow aB \mid ab$$

$$A \rightarrow aaA$$

$$A \rightarrow abBc \mid abbc$$

$$B \rightarrow aA$$

Substitute

$$B \rightarrow aA$$

$$S \rightarrow \cancel{aB} \mid ab \mid aaA$$

$$A \rightarrow aaA$$

$$A \rightarrow \cancel{abBc} \mid abbc \mid abaAc$$

Equivalent grammar

In general:

$$A \rightarrow xBz$$

$$B \rightarrow y_1$$

Substitute

$$B \rightarrow y_1$$

$$A \rightarrow xBz \mid xy_1z$$

equivalent grammar

Nullable Variables

λ - production : $A \rightarrow \lambda$

Nullable Variable: $A \Rightarrow \ldots \Rightarrow \lambda$

Removing Nullable Variables

Example Grammar:

$$S \rightarrow aMb$$

$$M \rightarrow aMb$$

$$M \rightarrow \lambda$$

Nullable variable

Final Grammar

$S \rightarrow aMb$

$M \rightarrow aMb$

~~$M \rightarrow \lambda$~~

Substitute

$M \rightarrow \lambda$

$S \rightarrow aMb$

$S \rightarrow ab$

$M \rightarrow aMb$

$M \rightarrow ab$

Unit-Productions

Unit Production: $A \rightarrow B$

(single variables on both sides)

Removing Unit Productions

Observation:

$$A \rightarrow A$$

Is removed immediately

Example Grammar:

$$S \rightarrow aA$$

$$A \rightarrow a \mid B$$

$$B \rightarrow A \mid bb$$

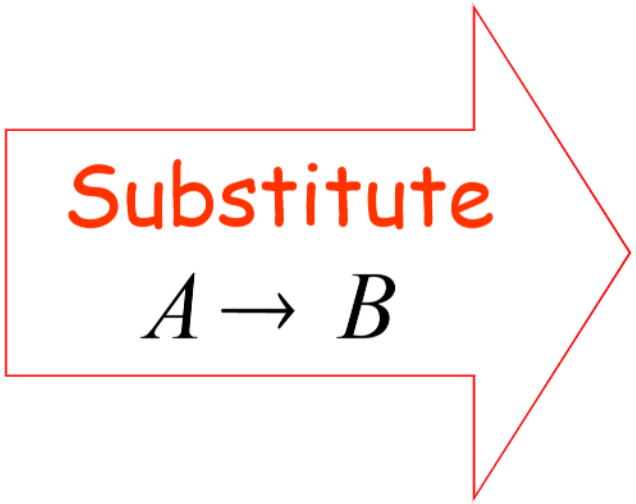
$S \to aA$

$A \to a$

~~$A \to B$~~

$B \to A$

$B \to bb$



$S \to aA$

$A \to a \mid bb$

$B \to A$

$B \to bb$

$S \rightarrow aA \mid aB$

$A \rightarrow a$

~~$B \rightarrow A$~~

$B \rightarrow bb$

Substitute

$B \rightarrow A$

$S \rightarrow aA$

$A \rightarrow a \mid bb$

$B \rightarrow a \mid bb$

Useless Productions

$$S \rightarrow aSb$$

$$S \rightarrow \lambda$$

$$S \rightarrow A$$

$$A \rightarrow aA$$ Useless Production

Some derivations never terminate...

$$S \Rightarrow A \Rightarrow aA \Rightarrow aaA \Rightarrow \ldots \Rightarrow aa\ldots aA \Rightarrow \ldots$$

Another grammar:

$$S \rightarrow A$$

$$A \rightarrow aA$$

$$A \rightarrow \lambda$$

$$B \rightarrow bA$$

Useless Production

Not reachable from S

In general:

contains only terminals

if $S \Rightarrow \ldots \Rightarrow xAy \Rightarrow \ldots \Rightarrow w$

$w \in L(G)$

then variable A is useful

otherwise, variable A is useless

A production $A \rightarrow x$ is useless
if any of its variables is useless

$S \rightarrow aSb$

$S \rightarrow \lambda$

Variables		Productions
	$S \rightarrow A$	useless
useless	$A \rightarrow aA$	useless
useless	$B \rightarrow C$	useless
useless	$C \rightarrow D$	useless

Removing Useless Productions

Example Grammar:

$$S \rightarrow aS \mid A \mid C$$

$$A \rightarrow a$$

$$B \rightarrow aa$$

$$C \rightarrow aCb$$

Remove useless productions

First: find all variables that can produce strings with only terminals

$S \rightarrow aS \mid A \mid C$

$A \rightarrow a$

$B \rightarrow aa$

$C \rightarrow aCb$

Round 1: $\{A, B\}$

$S \rightarrow A$

Round 2: $\{A, B, S\}$

Keep only the variables
that produce terminal symbols: $\{A, B, S\}$

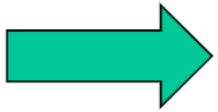
(other variables are useless)

$S \rightarrow aS \mid A \mid C$

$A \rightarrow a$

$B \rightarrow aa$

$C \rightarrow aCb$



$S \rightarrow aS \mid A$

$A \rightarrow a$

$B \rightarrow aa$

Remove useless productions

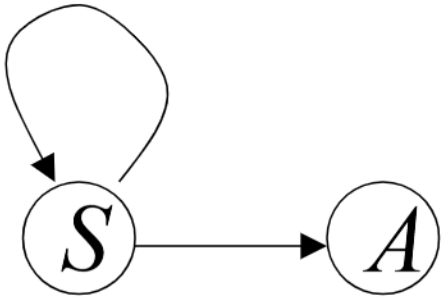
Second: Find all variables reachable from S

Use a Dependency Graph

$$S \rightarrow aS \mid A$$

$$A \rightarrow a$$

$$B \rightarrow aa$$



not reachable

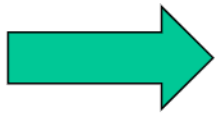
Keep only the variables reachable from S

(the other variables are useless)

$S \rightarrow aS \mid A$

$A \rightarrow a$

~~$B \rightarrow aa$~~



Final Grammar

$S \rightarrow aS \mid A$

$A \rightarrow a$

Remove useless productions

Removing All

Step 1: Remove Nullable Variables

Step 2: Remove Unit-Productions

Step 3: Remove Useless Variables

Normal Forms for Context-free Grammars

Chomsky Normal Form

Each production has form:

$A \to BC$ or $A \to a$

Examples:

$$S \rightarrow AS$$
$$S \rightarrow a$$
$$A \rightarrow SA$$
$$A \rightarrow b$$

Chomsky Normal Form

$$S \rightarrow AS$$
$$S \rightarrow AAS$$
$$A \rightarrow SA$$
$$A \rightarrow aa$$

Not Chomsky Normal Form

Conversion to Chomsky Normal Form

Example:

$$S \rightarrow ABa$$

$$A \rightarrow aab$$

$$B \rightarrow Ac$$

Not Chomsky Normal Form

Introduce variables for terminals: T_a, T_b, T_c

$$S \rightarrow ABa$$
$$A \rightarrow aab$$
$$B \rightarrow Ac$$

$$S \rightarrow ABT_a$$
$$A \rightarrow T_aT_aT_b$$
$$B \rightarrow AT_c$$
$$T_a \rightarrow a$$
$$T_b \rightarrow b$$
$$T_c \rightarrow c$$

Introduce intermediate variable: V_1

$$S \to ABT_a$$
$$A \to T_aT_aT_b$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

$$S \to AV_1$$
$$V_1 \to BT_a$$
$$A \to T_aT_aT_b$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

Introduce intermediate variable: V_2

$$S \rightarrow AV_1$$
$$V_1 \rightarrow BT_a$$
$$A \rightarrow T_aT_aT_b$$
$$B \rightarrow AT_c$$
$$T_a \rightarrow a$$
$$T_b \rightarrow b$$
$$T_c \rightarrow c$$

$$S \rightarrow AV_1$$
$$V_1 \rightarrow BT_a$$
$$A \rightarrow T_aV_2$$
$$V_2 \rightarrow T_aT_b$$
$$B \rightarrow AT_c$$
$$T_a \rightarrow a$$
$$T_b \rightarrow b$$
$$T_c \rightarrow c$$

Final grammar in Chomsky Normal Form:

Initial grammar

$$S \to ABa$$

$$A \to aab$$

$$B \to Ac$$

$$S \to AV_1$$

$$V_1 \to BT_a$$

$$A \to T_aV_2$$

$$V_2 \to T_aT_b$$

$$B \to AT_c$$

$$T_a \to a$$

$$T_b \to b$$

$$T_c \to c$$

In general:

From any context-free grammar
(which doesn't produce λ)
not in Chomsky Normal Form

we can obtain:

An equivalent grammar
in Chomsky Normal Form

The Procedure

First remove:

Nullable variables

Unit productions

Then, for every symbol a:

Add production $T_a \rightarrow a$

In productions: replace a with T_a

New variable: T_a

Replace any production $A \to C_1 C_2 \cdots C_n$

with

$$A \to C_1 V_1$$

$$V_1 \to C_2 V_2$$

$$\cdots$$

$$V_{n-2} \to C_{n-1} C_n$$

New intermediate variables: $V_1, V_2, \ldots, V_{n-2}$

Theorem: For any context-free grammar (which doesn't produce λ) there is an equivalent grammar in Chomsky Normal Form

Observations

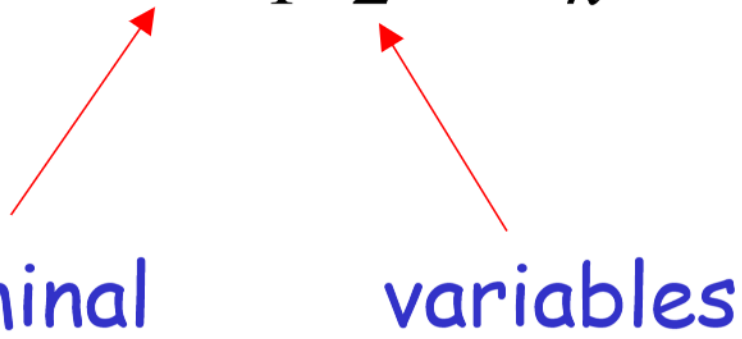
- Chomsky normal forms are good for parsing and proving theorems

- It is very easy to find the Chomsky normal form for any context-free grammar

Greibach Normal Form

All productions have form:

$$A \rightarrow a V_1 V_2 \cdots V_k \qquad k \geq 0$$



Examples:

$$S \rightarrow cAB$$

$$A \rightarrow aA \mid bB \mid b$$

$$B \rightarrow b$$

Greibach Normal Form

$$S \rightarrow abSb$$

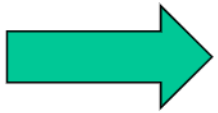
$$S \rightarrow aa$$

Not Greibach Normal Form

Conversion to Greibach Normal Form:

$$S \to abSb$$

$$S \to aa$$



$$S \to aT_bST_b$$

$$S \to aT_a$$

$$T_a \to a$$

$$T_b \to b$$

Greibach Normal Form

Theorem: For any context-free grammar (which doesn't produce λ) there is an equivalent grammar in Greibach Normal Form