CSCI 340: Computational Models

# Grammatical Format

Chapter 13

Department of Computer Science

## Regular Grammars

Some of the languages we’ve been defining with CFGs are *regular*

We have three possibilities:

1. All possible languages can be generated by CFGs
2. All regular languages can be generated by CFGs, and so can some nonregular languages but not all possible languages
3. Some regular languages can be generated by CFGs (and some cannot). Some nonregular languages can be generated by CFGs (and maybe some cannot)

Which one is true?

## Taking FAs and converting them to CFGs

**Definition**

A **semiword** is a string of terminals (maybe none) ending in a single non-terminal

(terminal) (terminal) (terminal) . . . (terminal)(Nonterminal)

**Theorem**

*Given any FA, there is a CFG that generates exactly the language accepted by the FA.*

*Alternatively, all regular languages are context-free languages*

## Proof by constructive algorithm

**Proof.**

1. The nonterminals in the CFG will be all the names of the states in the FA with the start state renamed to $S$
2. For every edge $Q_x \overset{a}{\rightarrow} Q_y$, create the production:
   $$Q_x \longrightarrow aQ_y$$
   Repeat for $b$ edges
3. For every final state $X$ create the production
   $$X \longrightarrow \Lambda$$

□

## Showing a CFG is regular

**Theorem**

*If all of the productions in a given CFG fits one of the two forms:*

$$\textit{Nonterminal} \longrightarrow \textit{semiword} \qquad \textit{or} \qquad \textit{Nonterminal} \longrightarrow \textit{word}$$

*where word may be* $\Lambda$*, then the language generated by this CFG is regular*

How can we prove this? Generate a transition graph!

**Example**

$$S \longrightarrow aaS \mid bbS \mid abX \mid baX \mid \Lambda$$
$$X \longrightarrow aaX \mid bbX \mid abS \mid baS$$

## Killing $\Lambda$-Productions

Quote from the book:

“We have not yet committed ourselves to a definite stand on the social acceptability of $\Lambda$-**productions** (productions of the form $N \to \Lambda$). We have employed them, but we do not pay them equal wages. They make our lives very difficult in later discussions, so we must ask ourselves, Do we need them at all?”

Answer: No

## Bar-Hillel, Perles, and Shamir Theorems

**Theorem**

*If L is a context-free language generated by a CFG that includes $\Lambda$-productions, then there is a different CFG that has no $\Lambda$-productions that generates either the whole language L (if L doesn't include $\lambda$) or generates all the words in L that are not $\lambda$)*

**Definition**

In a given CFG, we call a nonterminal $N$ **nullable** if

- There is a production $N \to \Lambda$ or
- There is a derivation that starts at $N$ and leads to $\Lambda$

$$N \Rightarrow \ldots \Rightarrow \Lambda$$

**Goal**

Fix all *nullable* nonterminals while removing all $\Lambda$-productions

## Replacement Rules and Example

1. Delete all $\Lambda$-productions
2. Add the following productions: For every production

   $$\text{X} \rightarrow \text{old string}$$

   add new productions of the form X $\rightarrow \dots$ where the right side will account for **any** modification of the old string that can be formed by deleting all possible subsets of nullable non-terminals.

**Example**

$$\begin{aligned} S &\rightarrow a \mid Xb \mid aYa \\ X &\rightarrow Y \mid \Lambda \\ Y &\rightarrow b \mid X \end{aligned}$$

## Example

$$S \to a \mid Xb \mid aYa$$
$$X \to Y \mid \Lambda$$
$$Y \to b \mid X$$

| **Old Production** | **Newly-formed Productions** |
|---|---|
| $X \to Y$ | Nothing |
| $X \to \Lambda$ | Nothing |
| $Y \to X$ | Nothing |
| $S \to Xb$ | $S \to b$ |
| $S \to aYa$ | $S \to aa$ |

**New CFG:**

$$S \to a \mid Xb \mid aYa \mid b \mid aa$$
$$X \to Y$$
$$Y \to b \mid X$$

## Chalkboard Examples

**Example**

| | |
|---|---|
| $S \longrightarrow Xa$ | $S \longrightarrow a$ |
| $X \longrightarrow aX \mid bX \mid \Lambda$ | $X \longrightarrow a \mid b$ |

**Example**

| | |
|---|---|
| $S \longrightarrow XY$ | |
| $X \longrightarrow Zb$ | $X \longrightarrow b$ |
| $Y \longrightarrow bW$ | $Y \longrightarrow b$ |
| $Z \longrightarrow AB$ | $Z \longrightarrow A \mid B$ |
| $W \longrightarrow Z$ | nothing new |
| $A \longrightarrow aA \mid bA \mid \Lambda$ | $A \longrightarrow a \mid b$ |
| $B \longrightarrow Ba \mid Bb \mid \Lambda$ | $B \longrightarrow a \mid b$ |

## Killing Unit Productions

**Theorem**

*If there is a CFG for the language L that has no $\Lambda$-productions, then there is also a CFG for L with no $\Lambda$-productions and no unit productions*

A **unit production** is a production of the form:

$$\text{Nonterminal} \rightarrow \text{one Nonterminal}$$

For every pair of nonterminals $A$ and $B$, *if* the CFG has a unit production $A \rightarrow B$ *or if* there is a chain of unit productions leading from $A$ to $B$, such as

$$A \Rightarrow X_1 \Rightarrow X_2 \Rightarrow \ldots \Rightarrow B$$

where $X_1 \ \ldots X_n$ are some nonterminals introduce new productions!

If the non unit productions (from $B$) are $B \rightarrow s_1 \mid s_2 \mid s_3 \mid \ldots$
create the productions $A \rightarrow s_1 \mid s_2 \mid s_3 \mid \ldots$

Do this for all pairs $A$ and $B$

## Example

$$
\begin{aligned}
S &\longrightarrow A \mid bb \\
A &\longrightarrow B \mid b \\
B &\longrightarrow S \mid a
\end{aligned}
$$

We can first split the productions into two groups:

| **Unit Productions** | **Decent Folks** |
|---|---|
| $S \longrightarrow A$ | $S \longrightarrow bb$ |
| $A \longrightarrow B$ | $A \longrightarrow b$ |
| $B \longrightarrow S$ | $B \longrightarrow a$ |

## Example (continued)

Now we can list all unit productions and sequences of unit productions

$$\begin{array}{lll}
S \to A & \text{gives} & S \to b \\
S \to A \to B & \text{gives} & S \to a \\
A \to B & \text{gives} & A \to a \\
A \to B \to S & \text{gives} & A \to bb \\
B \to S & \text{gives} & B \to bb \\
B \to S \to A & \text{gives} & B \to b
\end{array}$$

The new CFG for this language is:

$$S \to bb \mid b \mid a$$
$$A \to b \mid a \mid bb$$
$$B \to a \mid bb \mid b$$

## Chomsky Normal Form

Separating terminals and non-terminal production rules seemed nice!

**Theorem**

*If L is a language generated by some CFG, then there is another CFG that generates all the non-$\lambda$ words of L, all of whose productions are of one of two basic forms:*

1. *Nonterminal $\rightarrow$ string of only Nonterminals*
2. *Nonterminal $\rightarrow$ one terminal*

**Form 2:** Create a "capitalized" Nonterminal for each terminal:

- $A \rightarrow a$, $B \rightarrow b$

**Form 1:** Replace terminals with Nonterminal created for **Form 2** if they are on the right-hand side

- Before: $S \rightarrow aSa$
- After: $S \rightarrow ASA$

*Note: "If it ain't broke, don't fix it" (Forms 1 and 2 may already exist)*

## CNF Example

$$\begin{aligned} S &\rightarrow X \mid YaY \mid aSb \mid b \\ X &\rightarrow YY \mid b \\ Y &\rightarrow aY \mid aaX \end{aligned}$$

New Nonterminal states for terminals:

$$\begin{aligned} A &\rightarrow a \\ B &\rightarrow b \end{aligned}$$

After replacement of $a$ with $A$ and $b$ with $B$:

$$\begin{aligned} S &\rightarrow X \mid YAY \mid ASB \mid B \\ X &\rightarrow YY \mid B \\ Y &\rightarrow AY \mid AAX \\ A &\rightarrow a \\ B &\rightarrow b \end{aligned}$$

## Chomsky Normal Form

### Definition

If a CFG has only productions of the following two forms, it is said to be in **Chomsky Normal Form** or **CNF**

1. Nonterminal $\rightarrow$ string of exactly two Nonterminals
2. Nonterminal $\rightarrow$ one terminal

### Theorem

*For any context-free language L, the non-$\lambda$ words of L can be generated by a grammar in which all productions are in CNF.*

*Note: if L contained $\lambda$, then $\lambda$ will be "dropped" from the resulting CFL once converted to CNF*

## Chomsky Normal Form

**Proof.**

1. Assume we start with a CFG with no unit productions or $\Lambda$-productions
2. Create Non-terminals for each terminal and replace in the productions
   - $A \longrightarrow a$
   - $B \longrightarrow b$
3. Split all sequences of Non-terminals into productions with only two Non-terminals on the right-hand side
   - $S \longrightarrow ABCDE$
   - $S \longrightarrow AR_1$, $R_1 \longrightarrow BR_2$, $R_2 \longrightarrow CR_3$, $R_3 \longrightarrow DE$
   - Note that these $R$ states are only used internally
4. The effect of our productions are the same. All $R$-states are in CNF. All created Non-terminals from encapsulating terminals are in CNF. The new grammar generates the same language as the old grammar. $\square$

## Converting to CNF Example

**Non-null palindrome is defined as:**

$S \longrightarrow aSa \mid bSb \mid a \mid b \mid aa \mid bb$

**Separate Terminals from Nonterminals:**

$S \longrightarrow ASA \mid BSB \mid A \mid B \mid AA \mid BB$

$A \longrightarrow a$

$B \longrightarrow b$

**Introduce $R$ states:**

$S \longrightarrow AR_1 \mid BR_2 \mid A \mid B \mid AA \mid BB$

$R_1 \longrightarrow SA$

$R_2 \longrightarrow SB$

$A \longrightarrow a$

$B \longrightarrow b$