

# Lecture 12 – Examples of Context-Free Grammars

## COSE215: Theory of Computation

Jihyeok Park



2026 Spring

- A **context-free grammar (CFG)**:

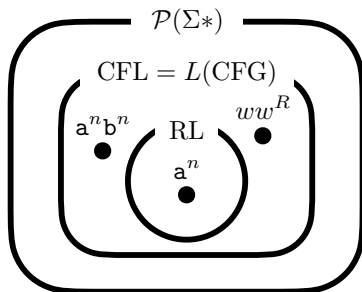
$$G = (V, \Sigma, S, R)$$

- The **language** of a CFG  $G$ :

$$L(G) = \{w \in \Sigma^* \mid S \Rightarrow^* w\}$$

- A language  $L$  is a **context-free language (CFL)**:

$$\exists \text{ CFG } G. L(G) = L$$



## 1. Regular Languages are Context-Free

Regular Expressions to CFGs

$\epsilon$ -NFA to CFG

## 2. Examples of Context-Free Grammars

Example 1:  $b^n a^m b^{2n}$

Example 2: Well-Formed Brackets

Example 3: Equal Number of a's and b's

Example 4: Unequal Number of a's and b's

Example 5: Arithmetic Expressions

Example 6: Regular Expressions

Example 7: Simplified Scala Syntax

## 1. Regular Languages are Context-Free

Regular Expressions to CFGs

$\epsilon$ -NFA to CFG

## 2. Examples of Context-Free Grammars

Example 1:  $b^n a^m b^{2n}$

Example 2: Well-Formed Brackets

Example 3: Equal Number of a's and b's

Example 4: Unequal Number of a's and b's

Example 5: Arithmetic Expressions

Example 6: Regular Expressions

Example 7: Simplified Scala Syntax

## Theorem (RLs are CFLs)

*All regular languages are context-free.*

**Proof)** There are two ways to prove this theorem:

- 1 Converting regular expressions to equivalent CFGs
- 2 Converting  $\epsilon$ -NFAs to equivalent CFGs

For a given regular language  $L$ , let's construct an equivalent CFG  $G$  using the equivalent regular expression  $R$ .  $L(G) = L(R)$ .

RE $R$	CFG $G$
$\emptyset$	$S \rightarrow S$
$\epsilon$	$S \rightarrow \epsilon$
$a \in \Sigma$	$S \rightarrow a$
$R_1 \mid R_2$	$S \rightarrow S_1 \mid S_2$
$R_1 \cdot R_2$	$S \rightarrow S_1 S_2$
$R_1^*$	$S \rightarrow \epsilon \mid S_1 S$
$(R_1)$	$S \rightarrow S_1$

where  $S_1$  and  $S_2$  are start variables of CFGs  $G_1$  and  $G_2$  such that  $L(G_1) = L(R_1)$  and  $L(G_2) = L(R_2)$ , respectively.

For a given RE  $R$ , construct a CFG  $G$  such that  $L(G) = L(R)$ .

- $R = \epsilon | ab | ba$

$$\begin{array}{llll}
 S \rightarrow F | D & A \rightarrow a & C \rightarrow AB & E \rightarrow \epsilon \\
 & B \rightarrow b & D \rightarrow BA & F \rightarrow E | C
 \end{array}$$

Its simplified version:

$$S \rightarrow \epsilon | ab | ba$$

- $R = (\epsilon | a)^*$

$$S \rightarrow \epsilon | AS \quad A \rightarrow \epsilon | a$$

- $R = (0 | 1(01^*0)^*1)^*$

$$\begin{array}{lll}
 S \rightarrow \epsilon | AS & A \rightarrow 0 | 1B1 & C \rightarrow 0D0 \\
 & B \rightarrow \epsilon | CB & D \rightarrow \epsilon | 1D
 \end{array}$$

For a given  $\epsilon$ -NFA  $N^\epsilon = (Q, \Sigma, \delta, q_0, F)$ , let's construct a CFG  $G$  as:

- For each state  $q \in Q$  of  $N^\epsilon$ , introduce a non-terminal  $A_q$ .
- For each transition  $q \xrightarrow{a} q'$  of  $N^\epsilon$ , introduce a production rule:

$$A_q \rightarrow aA_{q'}$$

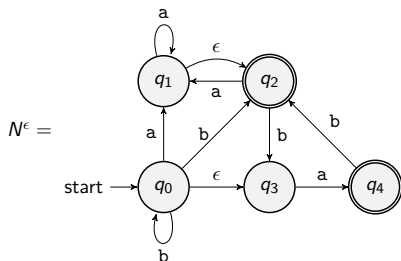
- For each  $\epsilon$ -transition  $q \xrightarrow{\epsilon} q'$  of  $N^\epsilon$ , introduce a production rule:

$$A_q \rightarrow A_{q'}$$

- For each final state  $q \in F$  of  $N^\epsilon$ , introduce a production rule:

$$A_q \rightarrow \epsilon$$

- The start variable of  $G$  is  $A_{q_0}$ .



We can construct a CFG  $G$  ( $A_0$  is the start variable) for  $N^\epsilon$ :

$$A_0 \rightarrow bA_0 \mid aA_1 \mid bA_2 \mid A_3$$

$$A_1 \rightarrow aA_1 \mid A_2$$

$$A_2 \rightarrow aA_1 \mid bA_3 \mid \epsilon$$

$$A_3 \rightarrow aA_4$$

$$A_4 \rightarrow bA_2 \mid \epsilon$$

For example, we can derive  $ba \in L(N^\epsilon)$  using  $G$ :

$$A_0 \Rightarrow bA_0 \Rightarrow bA_3 \Rightarrow baA_4 \Rightarrow ba$$

## 1. Regular Languages are Context-Free

Regular Expressions to CFGs

$\epsilon$ -NFA to CFG

## 2. Examples of Context-Free Grammars

Example 1:  $b^n a^m b^{2n}$

Example 2: Well-Formed Brackets

Example 3: Equal Number of a's and b's

Example 4: Unequal Number of a's and b's

Example 5: Arithmetic Expressions

Example 6: Regular Expressions

Example 7: Simplified Scala Syntax

## Example 1: $b^n a^m b^{2n}$

Construct a CFG for the language:

$$L = \{b^n a^m b^{2n} \mid n, m \geq 0\}$$

Let's split a word  $w \in L$  using shorter words in  $L$ .

$$\forall w \in L. w = \begin{cases} \textcircled{1} a^m & \text{for some } m \geq 0 \\ \textcircled{2} bw'bb & \text{for some } w' \in L \end{cases} \implies S \rightarrow A \mid bSbb$$

$$\forall m \geq 0. a^m = \begin{cases} \textcircled{1} \epsilon & \\ \textcircled{2} aa^{m-1} & \end{cases} \implies A \rightarrow \epsilon \mid aA$$

Therefore, the following is a CFG for  $L$ :

$$\begin{aligned} S &\rightarrow A \mid bSbb \\ A &\rightarrow \epsilon \mid aA \end{aligned}$$

## Example 2: Well-Formed Brackets

Construct a CFG for the language:

$$L = \{w \in \{ (, ), \{, \}, [, ] \}^* \mid w \text{ is well-formed} \}$$

Let's split a word  $w \in L$  using shorter words in  $L$ .

$$\forall w \in L. w = \begin{cases} \textcircled{1} \epsilon \\ \textcircled{2} (w') & \text{for some } w' \in L \\ \textcircled{3} \{w'\} & \text{for some } w' \in L \\ \textcircled{4} [w'] & \text{for some } w' \in L \\ \textcircled{5} w_1 w_2 & \text{for some } w_1, w_2 \in L \end{cases}$$

Therefore, the following is a CFG for  $L$ :

$$S \rightarrow \epsilon \mid (S) \mid \{S\} \mid [S] \mid SS$$

## Example 3: Equal Number of a's and b's

Construct a CFG for the language:

$$L = \{w \in \{a, b\}^* \mid N_a(w) = N_b(w)\}$$

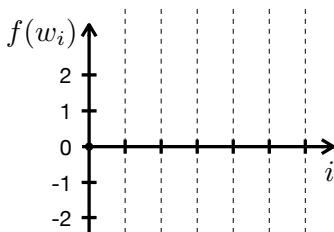
where  $N_a(w)$  and  $N_b(w)$  are the number of a's and b's in  $w$ , respectively.

Consider a function  $f(w) = N_a(w) - N_b(w)$ .

For example, if  $w = abbaaa$ , then  $f(w) = N_a(w) - N_b(w) = 4 - 2 = 2$ .

If a word  $w = a_1a_2 \cdots a_n \in \{a, b\}^*$ , let  $w_i = a_1a_2 \cdots a_i$  for  $0 \leq i \leq n$ .

Let's draw a graph for  $f(w_i)$  for  $0 \leq i \leq n$ :



## Example 3: Equal Number of a's and b's

Construct a CFG for the language:

$$L = \{w \in \{a, b\}^* \mid N_a(w) = N_b(w)\}$$

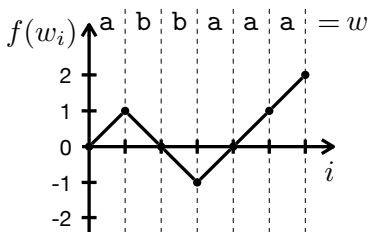
where  $N_a(w)$  and  $N_b(w)$  are the number of a's and b's in  $w$ , respectively.

Consider a function  $f(w) = N_a(w) - N_b(w)$ .

For example, if  $w = abbaaa$ , then  $f(w) = N_a(w) - N_b(w) = 4 - 2 = 2$ .

If a word  $w = a_1 a_2 \cdots a_n \in \{a, b\}^*$ , let  $w_i = a_1 a_2 \cdots a_i$  for  $0 \leq i \leq n$ .

Let's draw a graph for  $f(w_i)$  for  $0 \leq i \leq n$ :

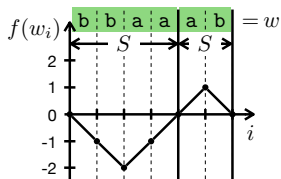


## Example 3: Equal Number of a's and b's

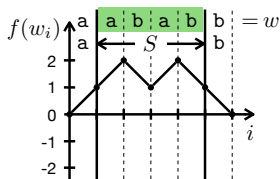
Let's split a word  $w \in L$  using shorter words in  $L$ .

For a given  $w \in L$ , there are four cases:

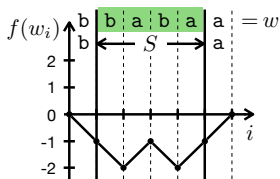
①  $w = \epsilon$



②  $w = w_1 w_2$



③  $w = a w' b$



④  $w = b w' a$

Therefore, the following is a CFG for  $L$ :

$$S \rightarrow \epsilon \mid SS \mid aSb \mid bSa$$

## Example 4: Unequal Number of a's and b's

Construct a CFG for the **complement** of the language in Example 3:

$$L = \{w \in \{a, b\}^* \mid N_a(w) \neq N_b(w)\}$$

where  $N_a(w)$  and  $N_b(w)$  are the number of a's and b's in  $w$ , respectively.

We can categorize  $w \in \{a, b\}^*$  into three cases using the function  $f$ :

- $L_Z = \{w \in \{a, b\}^* \mid f(w) = 0\}$  – equal number of a's and b's
- $L_P = \{w \in \{a, b\}^* \mid f(w) > 0\}$  – more a's than b's
- $L_N = \{w \in \{a, b\}^* \mid f(w) < 0\}$  – more b's than a's

The language  $L$  is the disjoint union of  $L_P$  and  $L_N$ :

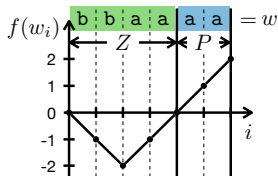
$$L = L_P \uplus L_N$$

Let's define production rules for  $L_P$  and  $L_N$  using graphs for  $f(w_i)$ .

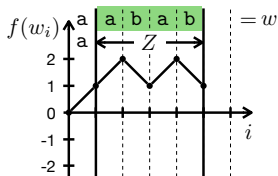
## Example 4: Unequal Number of a's and b's

Let's split a word  $w \in L_P$  using shorter words in  $L_Z$ ,  $L_P$ , and  $L_N$ .

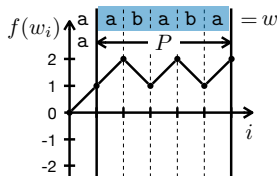
For a given  $w \in L_P$ , there are three cases:



①  $w = w_1 w_2$   
 ( $w_1 \in L_Z, w_2 \in L_P$ )



②  $w = a w'$   
 ( $w' \in L_Z$ )

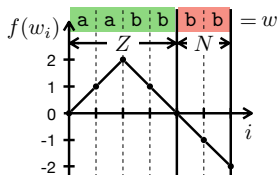


③  $w = a w'$   
 ( $w' \in L_P$ )

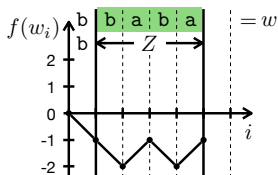
Therefore, the following is production rules for  $L_P$ :

$$P \rightarrow ZP \mid aP \mid aZ$$

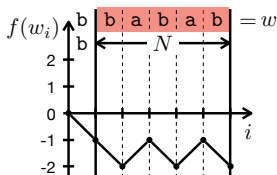
# Example 4: Unequal Number of a's and b's



①  $w = w_1 w_2$   
 $(w_1 \in L_Z, w_2 \in L_N)$



②  $w = b w'$   
 $(w' \in L_Z)$



③  $w = b w'$   
 $(w' \in L_N)$

Similarly, the following is production rules for  $L_N$ :

$$N \rightarrow ZN \mid bN \mid bZ$$

Therefore, the CFG for  $L$  is:

$$\begin{aligned} S &\rightarrow P \mid N \\ P &\rightarrow ZP \mid aP \mid aZ \\ N &\rightarrow ZN \mid bN \mid bZ \\ Z &\rightarrow \epsilon \mid ZZ \mid aZb \mid bZa \end{aligned}$$

## Example 5: Arithmetic Expressions

An **arithmetic expression** is defined with the following CFG:

$$S \rightarrow N \mid X \mid S+S \mid S*S \mid (S)$$

$$N \rightarrow D \mid DN$$

$$D \rightarrow 0 \mid \dots \mid 9$$

$$X \rightarrow a \mid \dots \mid z$$

We can **derive** an arithmetic expression  $13*(2+x)$  as follows:

$$\begin{aligned} S &\Rightarrow S*S && \Rightarrow N*S && \Rightarrow DN*S && \Rightarrow 1N*S \\ &\Rightarrow 1D*S && \Rightarrow 13*S && \Rightarrow 13*(S) && \Rightarrow 13*(S+S) \\ &\Rightarrow 13*(N+S) && \Rightarrow 13*(D+S) && \Rightarrow 13*(2+S) && \Rightarrow 13*(2+X) \\ &\Rightarrow 13*(2+x) \end{aligned}$$

## Example 6: Regular Expressions

Consider a language representing the **syntax of regular expressions**:

$$L = \{w \in \{\emptyset, \varepsilon, a, b, |, *, (, )\}^* \mid w \text{ is a regular expression over } \{a, b\}\}$$

Is this language  $L$  **regular**? or **context-free**?

We can prove that  $L$  is **not regular** using the pumping lemma.  
(Hint: consider a word  $(^n\varepsilon)^n$  for a given  $n > 0$ )

However, the language  $L$  is **context-free**:

$$S \rightarrow \emptyset \mid \varepsilon \mid a \mid b \mid S \mid S \mid SS \mid S^* \mid (S)$$

We can **derive** a regular expression  $(b|ab)^*$  as follows:

$$\begin{aligned} S &\Rightarrow S^* && \Rightarrow (S)^* && \Rightarrow (S|S)^* \\ &\Rightarrow (S|SS)^* && \Rightarrow (S|Sb)^* && \Rightarrow (S|ab)^* \\ &\Rightarrow (b|ab)^* \end{aligned}$$

## Example 7: Simplified Scala Syntax

We can define a CFG for a simplified version of Scala syntax<sup>1</sup>:

(Scala Program)	$S \rightarrow E \mid S ; E$
(Expressions)	$E \rightarrow N \mid X \mid E + E \mid E - E \mid E * E \mid E / E$   <code>val</code> $X : T = E$   <code>def</code> $X ( P ) : T = E$   $E ( A )$   <code>if</code> $( E ) E$ <code>else</code> $E$   <code>enum</code> $T \{ D \}$   $E$ <code>match</code> $\{ C \}$
(Numbers)	$N \rightarrow 0 \mid \dots \mid 9 \mid 0N \mid \dots \mid 9N$
(Variables)	$X \rightarrow Y \mid YX$ $Y \rightarrow \emptyset \mid a \mid \dots \mid z \mid A \mid \dots \mid Z$
(Types)	$T \rightarrow X \mid T [ T ] \mid T \Rightarrow T$
(Parameters)	$P \rightarrow \epsilon \mid X : T \mid P , X : T$
(Arguments)	$A \rightarrow \epsilon \mid E \mid A , E$
(Cases)	$C \rightarrow \text{case } E \Rightarrow E \mid C ; \text{case } E \Rightarrow E$
(Enum Cases)	$D \rightarrow \text{case } T ( P ) \mid D ; \text{case } T ( P )$

<sup>1</sup><https://docs.scala-lang.org/scala3/reference/syntax.html>

## Example 7: Simplified Scala Syntax

```
def sum(n: Int): Int = n match { case 0 => 0; case n => n + sum(n - 1) }
```

A derivation for this program:

$$\begin{aligned} S &\Rightarrow^* \text{def } X(P): T = E && \Rightarrow^* \text{def } \text{sum}(P): T = E \\ &\Rightarrow^* \text{def } \text{sum}(X: T): T = E && \Rightarrow^* \text{def } \text{sum}(n: \text{Int}): \text{Int} = E \\ &\Rightarrow^* \text{def } \text{sum}(n: \text{Int}): \text{Int} = E \text{ match } \{ C \} \\ &\Rightarrow^* \text{def } \text{sum}(n: \text{Int}): \text{Int} = n \text{ match } \{ C \} \\ &\Rightarrow^* \text{def } \text{sum}(n: \text{Int}): \text{Int} = n \text{ match } \{ \text{case } E \Rightarrow E ; C \} \\ &\Rightarrow^* \text{def } \text{sum}(n: \text{Int}): \text{Int} = n \text{ match } \{ \text{case } 0 \Rightarrow 0; C \} \\ &\Rightarrow^* \text{def } \text{sum}(n: \text{Int}): \text{Int} = n \text{ match } \{ \text{case } 0 \Rightarrow 0; \text{case } E \Rightarrow E \} \\ &\Rightarrow^* \text{def } \text{sum}(n: \text{Int}): \text{Int} = n \text{ match } \{ \text{case } 0 \Rightarrow 0; \text{case } n \Rightarrow E \} \\ &\Rightarrow^* \text{def } \text{sum}(n: \text{Int}): \text{Int} = n \text{ match } \{ \text{case } 0 \Rightarrow 0; \text{case } n \Rightarrow E + E \} \\ &\Rightarrow^* \text{def } \text{sum}(n: \text{Int}): \text{Int} = n \text{ match } \{ \text{case } 0 \Rightarrow 0; \text{case } n \Rightarrow n + E \} \\ &\Rightarrow^* \text{def } \text{sum}(n: \text{Int}): \text{Int} = n \text{ match } \{ \text{case } 0 \Rightarrow 0; \text{case } n \Rightarrow n + \text{sum}(n - 1) \} \end{aligned}$$

## 1. Regular Languages are Context-Free

Regular Expressions to CFGs

$\epsilon$ -NFA to CFG

## 2. Examples of Context-Free Grammars

Example 1:  $b^n a^m b^{2n}$

Example 2: Well-Formed Brackets

Example 3: Equal Number of a's and b's

Example 4: Unequal Number of a's and b's

Example 5: Arithmetic Expressions

Example 6: Regular Expressions

Example 7: Simplified Scala Syntax

- The midterm exam will be given in class.
- **Date:** 13:30-14:45 (1 hour 15 minutes), April 22 (Wed.).
- **Location:**
  - **IT-Education Hall B102:** Students with Student ID 2025XXXX
  - **IT-Education Hall 611:** All other students
- **Coverage:** Lectures 1 – 13 (No coding question)
- **Format:** 7–9 questions with closed book and closed notes
- Please refer to the **previous exams** in the course website:

<https://plrg.korea.ac.kr/courses/cose215/>

- **Apr. 27 (Mon.):** No class
- **Apr. 29 (Wed.):** Recorded video lecture (Online).

- Parse Trees and Ambiguity

Jihyeok Park

[jihyeok\\_park@korea.ac.kr](mailto:jihyeok_park@korea.ac.kr)

<https://plrg.korea.ac.kr>