Lecture 12 – Examples of Context-Free Grammars

COSE215: Theory of Computation

Jihyeok Park

PLRG

2025 Spring

Recall

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

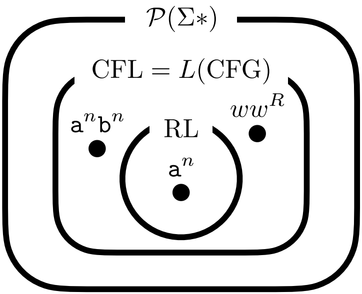
$$G = (V, \Sigma, S, R)$$

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

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

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

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



Contents

Contents

Regular Languages are Context-Free

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 ϵ-NFAs to equivalent CFGs

Regular Expressions to 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
$\varnothing$	$S \to S$
ϵ	$S \to \epsilon$
$a \in \Sigma$	$S \to a$
$R_1 \mid R_2$	$S \to S_1 \mid S_2$
$R_1 \cdot R_2$	$S \to S_1 S_2$
R_1^*	$S \to \epsilon \mid S_1 S$
(R_1)	$S \to S_1$

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.

Regular Expressions to CFGs – Examples

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

- $R = \epsilon|\mathtt{ab}|\mathtt{ba}$

$$S \rightarrow F \mid D \qquad A \rightarrow \mathtt{a} \qquad C \rightarrow AB \qquad E \rightarrow \epsilon$$
$$B \rightarrow \mathtt{b} \qquad D \rightarrow BA \qquad F \rightarrow E \mid C$$

  Its simplified version:

$$S \rightarrow \epsilon \mid \mathtt{ab} \mid \mathtt{ba}$$

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

$$S \rightarrow \epsilon \mid AS \qquad A \rightarrow \epsilon \mid \mathtt{a}$$

- $R = (\mathtt{0}|\mathtt{1}(\mathtt{0}\mathtt{1}^*\mathtt{0})^*\mathtt{1})^*$

$$S \rightarrow \epsilon \mid AS \qquad A \rightarrow \mathtt{0} \mid \mathtt{1}B\mathtt{1} \qquad C \rightarrow \mathtt{0}D\mathtt{0}$$
$$B \rightarrow \epsilon \mid CB \qquad D \rightarrow \epsilon \mid \mathtt{1}D$$

ϵ-NFAs to CFGs

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

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

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

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

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

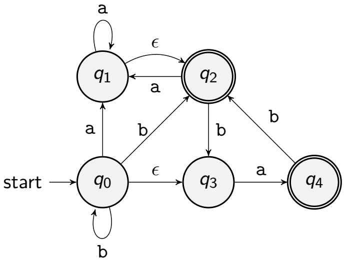
- For each final state $q \in F$ of N^ϵ, introduce a production rule:

$$A_q \to \epsilon$$

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

ϵ-NFAs to CFGs – Examples

$N^\epsilon =$



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

$$
\begin{aligned}
A_0 &\to \mathtt{b}A_0 \mid \mathtt{a}A_1 \mid \mathtt{b}A_2 \mid A_3 \\
A_1 &\to \mathtt{a}A_1 \mid A_2 \\
A_2 &\to \mathtt{a}A_1 \mid \mathtt{b}A_3 \mid \epsilon \\
A_3 &\to \mathtt{a}A_4 \\
A_4 &\to \mathtt{b}A_2 \mid \epsilon
\end{aligned}
$$

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

$$A_0 \Rightarrow \mathtt{b}A_0 \Rightarrow \mathtt{b}A_3 \Rightarrow \mathtt{ba}A_4 \Rightarrow \mathtt{ba}$$

Contents

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

Construct a CFG for the language:

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

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

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

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

Therefore, the following is a CFG for L:

$$\begin{aligned} S &\rightarrow A \mid \mathtt{b}S\mathtt{bb} \\ A &\rightarrow \epsilon \mid \mathtt{a}A \end{aligned}$$

Example 2: Well-Formed Brackets

Construct a CFG for the language:

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

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

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

Therefore, the following is a CFG for L:

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

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

Construct a CFG for the language:

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

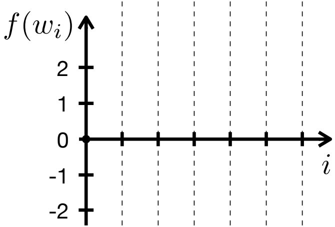
where $N_{\mathtt{a}}(w)$ and $N_{\mathtt{b}}(w)$ are the number of a's and b's in w, respectively.

Consider a function $f(w) = N_{\mathtt{a}}(w) - N_{\mathtt{b}}(w)$.

For example, if $w = \mathtt{abbaaa}$, then $f(w) = N_{\mathtt{a}}(w) - N_{\mathtt{b}}(w) = 4 - 2 = 2$.

If a word $w = a_1 a_2 \cdots a_n \in \{\mathtt{a}, \mathtt{b}\}^*$, 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

Construct a CFG for the language:

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

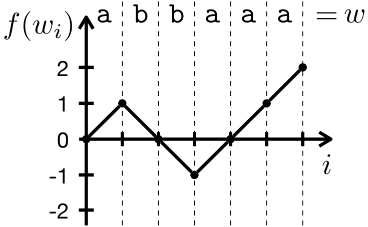
where $N_\mathtt{a}(w)$ and $N_\mathtt{b}(w)$ are the number of a's and b's in w, respectively.

Consider a function $f(w) = N_\mathtt{a}(w) - N_\mathtt{b}(w)$.

For example, if $w = \mathtt{abbaaa}$, then $f(w) = N_\mathtt{a}(w) - N_\mathtt{b}(w) = 4 - 2 = 2$.

If a word $w = a_1 a_2 \cdots a_n \in \{\mathtt{a}, \mathtt{b}\}^*$, 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 $0 \leq i \leq n$:

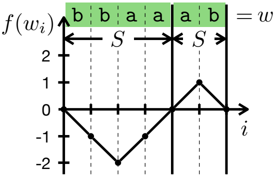


Example 3: Equal Number of a's and b's

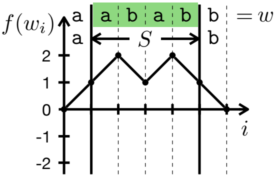
Let's split a word $w \in L$ using shorter words in L.

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

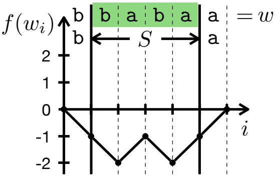
① $w = \epsilon$



② $w = w_1 w_2$



③ $w = \mathtt{a}w'\mathtt{b}$



④ $w = \mathtt{b}w'\mathtt{a}$

Therefore, the following is a CFG for L:

$$S \rightarrow \epsilon \mid SS \mid \mathtt{a}S\mathtt{b} \mid \mathtt{b}S\mathtt{a}$$

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

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

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

where $N_{\mathtt{a}}(w)$ and $N_{\mathtt{b}}(w)$ are the number of a's and b's in w, respectively.

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

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

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

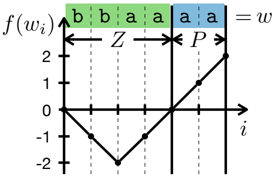
$$L = L_P \uplus L_N$$

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

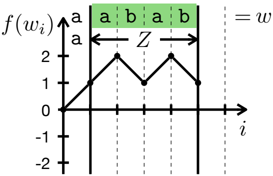
Let's split a word $w \in L_P$ using shorter words in L_Z, L_P, and L_N.

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



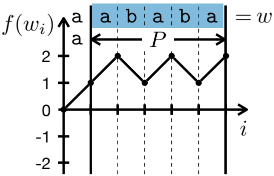
① $w = w_1 w_2$

$(w_1 \in L_Z, w_2 \in L_P)$



② $w = \mathtt{a}w'$

$(w' \in L_Z)$



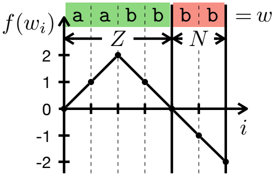
③ $w = \mathtt{a}w'$

$(w' \in L_P)$

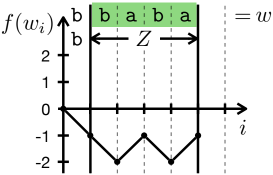
Therefore, the following is production rules for L_P:

$$P \rightarrow ZP \mid \mathtt{a}P \mid \mathtt{a}Z$$

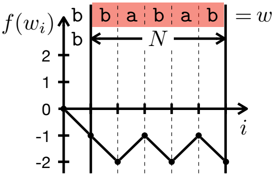
Example 4: Unequal Number of a's and b's



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



② $w = \mathtt{b}w'$
$(w' \in L_Z)$



③ $w = \mathtt{b}w'$
$(w' \in L_N)$

Similarly, the following is production rules for L_N:

$$N \rightarrow ZN \mid \mathtt{b}N \mid \mathtt{b}Z$$

Therefore, the CFG for L is:

$$\begin{aligned} S &\rightarrow P \mid N \\ P &\rightarrow ZP \mid \mathtt{a}P \mid \mathtt{a}Z \\ N &\rightarrow ZN \mid \mathtt{b}N \mid \mathtt{b}Z \\ Z &\rightarrow \epsilon \mid ZZ \mid \mathtt{a}Z\mathtt{b} \mid \mathtt{b}Z\mathtt{a} \end{aligned}$$

Example 5: Arithmetic Expressions

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

$$\begin{aligned} S &\to N \mid X \mid S\texttt{+}S \mid S\texttt{*}S \mid \texttt{(}S\texttt{)} \\ N &\to D \mid DN \\ D &\to \texttt{0} \mid \cdots \mid \texttt{9} \\ X &\to \texttt{a} \mid \cdots \mid \texttt{z} \end{aligned}$$

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

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

Example 6: Regular Expressions

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

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

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

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

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

$$S \rightarrow \varnothing \mid \varepsilon \mid \mathtt{a} \mid \mathtt{b} \mid S\mathtt{|}S \mid SS \mid S\mathtt{*} \mid \mathtt{(}S\mathtt{)}$$

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

$$\begin{array}{llllll} S & \Rightarrow & S\mathtt{*} & \Rightarrow & \mathtt{(}S\mathtt{)*} & \Rightarrow \; \mathtt{(}S\mathtt{|}S\mathtt{)*} \\ & \Rightarrow & \mathtt{(}S\mathtt{|}SS\mathtt{)*} & \Rightarrow & \mathtt{(}S\mathtt{|}S\mathtt{b)*} & \Rightarrow \; \mathtt{(}S\mathtt{|ab)*} \\ & \Rightarrow & \mathtt{(b|ab)*} & & & \end{array}$$

Example 7: Simplified Scala Syntax

We can define a CFG for a simplified version of Scala syntax[1]:

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

[1] 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:

$S \Rightarrow^*$ `def` X(P): T = E $\Rightarrow^*$ `def sum(` P `):` T = E

$\Rightarrow^*$ `def sum(` X: T `):` T = E $\Rightarrow^*$ `def sum(n: Int): Int =` E

$\Rightarrow^*$ `def sum(n: Int): Int =` E `match {` C `}`

$\Rightarrow^*$ `def sum(n: Int): Int = n match {` C `}`

$\Rightarrow^*$ `def sum(n: Int): Int = n match { case` E `=>` E `;` C `}`

$\Rightarrow^*$ `def sum(n: Int): Int = n match { case 0 => 0;` C `}`

$\Rightarrow^*$ `def sum(n: Int): Int = n match { case 0 => 0; case` E `=>` E `}`

$\Rightarrow^*$ `def sum(n: Int): Int = n match { case 0 => 0; case n =>` E `}`

$\Rightarrow^*$ `def sum(n: Int): Int = n match { case 0 => 0; case n =>` E + E `}`

$\Rightarrow^*$ `def sum(n: Int): Int = n match { case 0 => 0; case n => n +` E `}`

$\Rightarrow^*$ `def sum(n: Int): Int = n match { case 0 => 0; case n => n + sum(n - 1) }`

Summary

1. Regular Languages are Context-Free
   - Regular Expressions to CFGs
   - ϵ-NFA to CFG

2. Examples of Context-Free Grammars
   - Example 1: $\mathtt{b}^n\mathtt{a}^m\mathtt{b}^{2n}$
   - 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

Midterm Exam

- The midterm exam will be given in class.
- **Date:** 13:30-14:45 (1 hour 15 minutes), April 23 (Wed.).
- **Location:** 301, Aegineung (애기능생활관 301호)
- **Coverage:** Lectures 1 – 13
- **Format:** 7–9 questions with closed book and closed notes
  - Filling blanks in some tables, sentences, or expressions.
  - Construction of automata or grammars for given languages.
  - Proofs of given statements related to languages and automata.
  - Yes/No questions about concepts in the theory of computation.
  - etc.
- Note that there is **no class** on **April 28 (Mon.)**.
- Please refer to the **previous exams** in the course website:

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

Next Lecture

- Parse Trees and Ambiguity

Jihyeok Park
jihyeok_park@korea.ac.kr
https://plrg.korea.ac.kr