

# Lecture 13 – Parse Trees and Ambiguity

## COSE215: Theory of Computation

Jihyeok Park



2025 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)**:

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

- For a given word  $w \in L(G)$ , a **derivation** for  $w$  is  $S \Rightarrow^* w$
- A sequence  $\alpha \in (V \cup \Sigma)^*$  is a **sentential form** if  $S \Rightarrow^* \alpha$ .

## 1. Parse Trees

- Definition

- Yields

- Relationship between Parse Trees and Derivations

## 2. Ambiguity

- Ambiguous Grammars

- Eliminating Ambiguity

- Inherent Ambiguity

## 1. Parse Trees

- Definition

- Yields

- Relationship between Parse Trees and Derivations

## 2. Ambiguity

- Ambiguous Grammars

- Eliminating Ambiguity

- Inherent Ambiguity

Consider the following CFG for balanced parentheses:

$$S \rightarrow \epsilon \mid (S) \mid SS$$

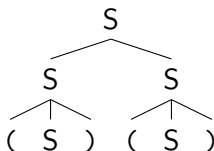
There are two different derivations for the sentential form  $(S)(S)$ :

$$\textcircled{1} \quad S \Rightarrow_L SS \Rightarrow_L (S)S \Rightarrow (S)(S)$$

$$\textcircled{2} \quad S \Rightarrow_R SS \Rightarrow_R S(S) \Rightarrow (S)(S)$$

However, **parse trees** focus on the structure of the derivations instead of considering the order of the derivation steps.

For example, the above two derivations have the same parse tree:



## Definition (Parse Trees)

For a given CFG  $G = (V, \Sigma, S, R)$ , **parse trees** are trees satisfying:

- ① The **root node** is labeled with the **start variable**  $S$ .
- ② Each **internal node** is labeled with a **variable**  $A \in V$ .

If its children are labeled with:

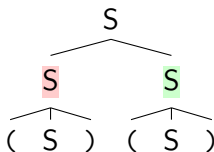
$$X_1, X_2, \dots, X_k$$

from the left to the right, then  $A \rightarrow X_1 X_2 \dots X_k \in R$ .

- ③ Each **leaf node** is labeled with a variable, symbol, or  $\epsilon$ . However, if a leaf node is labeled with  $\epsilon$ , it must be the only child of its parent.

$$S \rightarrow \epsilon \mid (S) \mid SS$$

A parse tree for  $(S)(S)$ :

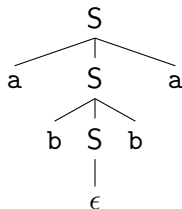


$$\textcircled{1} \quad S \Rightarrow_L \textcolor{red}{S} \textcolor{green}{S} \Rightarrow_L (S)S \Rightarrow (S)(S)$$

$$\textcircled{2} \quad S \Rightarrow_R \textcolor{red}{S} \textcolor{green}{S} \Rightarrow_R S(S) \Rightarrow (S)(S)$$

$$S \rightarrow \epsilon \mid aSa \mid bSb$$

A parse tree for abba:





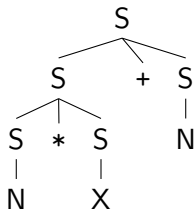
$$S \rightarrow N \mid X \mid S+S \mid S*S \mid (S)$$

$$N \rightarrow D \mid DN$$

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

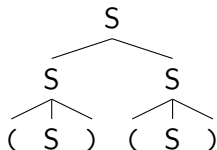
$$X \rightarrow a \mid \dots \mid z$$

A parse tree for  $N*X+N$ :

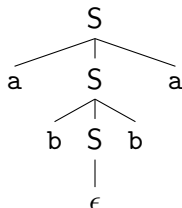


## Definition (Yields)

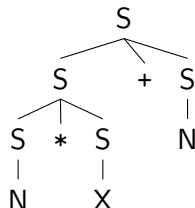
The sequence obtained by concatenating the labels (without  $\epsilon$ ) of the leaf nodes of a parse tree from left to right is called the **yield** of the parse tree.



Its yield is  $(S)(S)$ .



Its yield is  $abba$ .



Its yield is  $N * X + N$ .

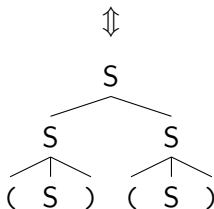
## Theorem (Parse Trees and Derivations)

For a given CFG  $G = (V, \Sigma, S, R)$ , for any sequence  $\alpha \in (V \cup \Sigma)^*$ :

$$S \Rightarrow^* \alpha \iff \exists \text{ parse tree } T. \text{ s.t. } T \text{ yields } \alpha$$

For example, consider the sequence  $(S)(S)$ :

$$S \Rightarrow SS \Rightarrow (S)S \Rightarrow (S)(S)$$



## 1. Parse Trees

Definition

Yields

Relationship between Parse Trees and Derivations

## 2. Ambiguity

Ambiguous Grammars

Eliminating Ambiguity

Inherent Ambiguity

Is there always a **unique** parse tree for a given sentential form?

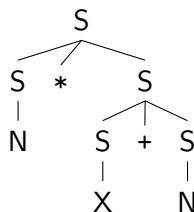
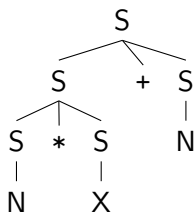
$$S \rightarrow N \mid X \mid S+S \mid S*S \mid (S)$$

$$N \rightarrow D \mid DN$$

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

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

For example, consider the sentential form  $N*X+N$ :



Actually, there are **two** parse trees for  $N*X+N$ .

## Definition (Ambiguous Grammar)

A context-free grammar  $G = (V, \Sigma, S, R)$  is **ambiguous** if there exist two distinct parse trees for a word  $w \in \Sigma^*$ . If not,  $G$  is **unambiguous**.

## Theorem

*Let  $G = (V, \Sigma, S, R)$  be a CFG. Then, the following numbers are equal for any sequence of variables or symbols  $w \in (V \cup \Sigma)^*$ :*

- 1 *The number of parse trees whose yields are  $w$ .*
- 2 *The number of left-most derivations for  $w$ .*
- 3 *The number of right-most derivations for  $w$ .*

**Proof)** We can convert a left-most (or right-most) derivation for a word  $w$  into the corresponding parse tree for  $w$  and vice versa.

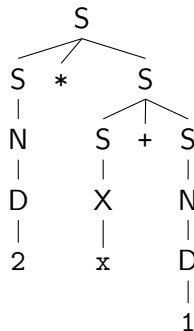
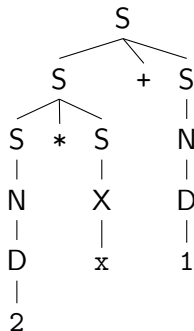
$$S \rightarrow N \mid X \mid S+S \mid S*S \mid (S)$$

$$N \rightarrow D \mid DN$$

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

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

This grammar is **ambiguous** because there are **two** parse trees for the word  $2 * x + 1$ :



So, there are **two** left-most (or right-most) derivations for  $2 * x + 1$ .

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

There are **two** left-most derivations for  $2 * x + 1$ :

- 1 Applying the production rule  $S \rightarrow S+S$  first:

$$\begin{aligned} S &\Rightarrow_L S+S &\Rightarrow_L S*S+S &\Rightarrow_L N*S+S &\Rightarrow_L D*S+S &\Rightarrow_L 2*S+S \\ &\Rightarrow_L 2*X+S &\Rightarrow_L 2*x+S &\Rightarrow_L 2*x+N &\Rightarrow_L 2*x+D &\Rightarrow_L 2*x+1 \end{aligned}$$

- 2 Applying the production rule  $S \rightarrow S*S$  first:

$$\begin{aligned} S &\Rightarrow_L S*S &\Rightarrow_L N*S &\Rightarrow_L D*S &\Rightarrow_L 2*S &\Rightarrow_L 2*S+S \\ &\Rightarrow_L 2*X+S &\Rightarrow_L 2*x+S &\Rightarrow_L 2*x+N &\Rightarrow_L 2*x+D &\Rightarrow_L 2*x+1 \end{aligned}$$



Unfortunately,

- There is **NO** general algorithm to remove ambiguity from a CFG.
- There is even **NO** algorithm to determine a CFG is ambiguous.

Fortunately, there are well-known techniques to manually **eliminate** the ambiguity in a given grammar commonly used in programming languages.

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

$$N \rightarrow D \mid DN$$

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

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

For example, an equivalent but unambiguous grammar is:

$$S \rightarrow T \mid S+T$$

$$T \rightarrow F \mid T*F$$

$$F \rightarrow N \mid X \mid (S)$$

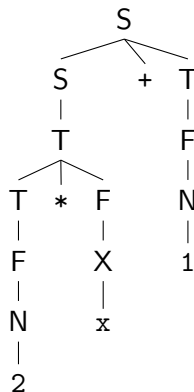
$$N \rightarrow D \mid DN$$

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

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

# Eliminating Ambiguity

Now, the unique parse tree for  $2 * x + 1$  is:

$$\begin{aligned} S &\rightarrow T \mid S + T \\ T &\rightarrow F \mid T * F \\ F &\rightarrow N \mid X \mid (S) \\ N &\rightarrow D \mid DN \\ D &\rightarrow 0 \mid \dots \mid 9 \\ X &\rightarrow a \mid \dots \mid z \end{aligned}$$


Let's try to understand how to eliminate the ambiguity in the original grammar.

First, analyze why the original grammar is ambiguous.

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

$$N \rightarrow D \mid DN$$

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

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

- **Precedence** is not specified between different operators (+ and \*).

- For example, two parse trees for  $1 * 2 + 3$  interpreted as:

$$1 * (2 + 3) \quad \text{and} \quad (1 * 2) + 3$$

- Let's give \* higher precedence than + to interpret it as  $(1 * 2) + 3$ .

- **Associativity** for the same operator (+ or \*).

- For example, two parse trees for  $1 + 2 + 3$  interpreted as:

$$1 + (2 + 3) \quad \text{and} \quad (1 + 2) + 3$$

- Let's give the left-associativity to + to interpret it as  $(1 + 2) + 3$ .

To enforce the **precedence**, define new variables  $F$  for factors and  $T$  for terms:

- A **factor** is a number, a variable, or a parenthesized expression:

$$42, \quad x, \quad (1 + 2), \quad \dots$$

In the grammar,  $F$  is defined as:

$$F \rightarrow N \mid X \mid (S)$$

- A **term** is the multiplication of one or more factors:

$$42, \quad 2 * x, \quad 2 * (1 + 2), \quad 1 * (x * y) * z, \quad \dots$$

In the grammar,  $T$  is defined as:

$$T \rightarrow F \mid T * F$$

- An **expression** is the addition of one or more terms:

$$42, \quad 1 + 2, \quad 1 + 2 * 3, \quad (1 + 2) * 3 + 4), \quad \dots$$

In the grammar,  $S$  is defined as:

$$S \rightarrow T \mid S + T$$

The unambiguous grammar is:

$$S \rightarrow T \mid S+T$$

$$T \rightarrow F \mid T*F$$

$$F \rightarrow N \mid X \mid (S)$$

$$N \rightarrow D \mid DN$$

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

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

- This grammar supports the **left-associativity** of + and \*. Why?

The unambiguous grammar is:

$$\begin{aligned} S &\rightarrow T \mid S+T \\ T &\rightarrow F \mid T*F \\ F &\rightarrow N \mid X \mid (S) \\ N &\rightarrow 0 \mid \dots \mid 9 \mid 0N \mid \dots \mid 9N \\ X &\rightarrow a \mid \dots \mid z \end{aligned}$$

- This grammar supports the **left-associativity** of + and \*. Why?
  - $S \rightarrow S+T$  and  $T \rightarrow T*F$  are **left-recursive**.
- Then, how to support the **right-associativity** of + and \*?
  - Replace the **left-recursive** rules with **right-recursive** rules!

$$\begin{aligned} S &\rightarrow T \mid T+S \\ T &\rightarrow F \mid F*T \\ &\dots \end{aligned}$$

So far, we have discussed the **ambiguity** for **grammars**.  
We will now discuss the **inherent ambiguity** for **languages**.

## Definition (Inherent Ambiguity)

A language  $L$  is **inherently ambiguous** if all CFGs whose languages are  $L$  are ambiguous. (i.e. there is no unambiguous grammar for  $L$ )

For example, the following language is **inherently ambiguous**:

$$L = \{a^i b^j c^k \mid i, j, k \geq 0 \wedge (i = j \vee j = k)\}$$

An example of ambiguous grammar for  $L$  is:

$$\begin{array}{lll} S \rightarrow L \mid R & L \rightarrow X \mid Lc & R \rightarrow Y \mid aR \\ & X \rightarrow \epsilon \mid aXb & Y \rightarrow \epsilon \mid bYc \end{array}$$

While we can prove that  $L$  is inherently ambiguous using the Ogden's lemma<sup>1</sup>, we will not discuss it in this course.

---

<sup>1</sup>[https://en.wikipedia.org/wiki/Ogden's\\_lemma](https://en.wikipedia.org/wiki/Ogden's_lemma)

## 1. Parse Trees

Definition

Yields

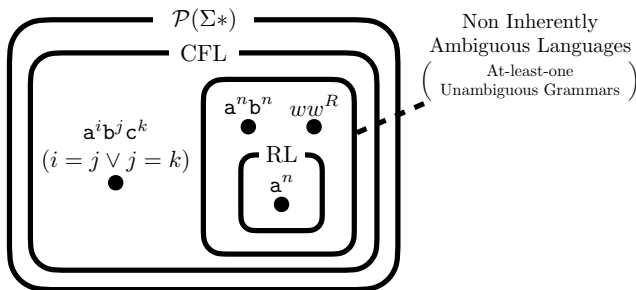
Relationship between Parse Trees and Derivations

## 2. Ambiguity

Ambiguous Grammars

Eliminating Ambiguity

Inherent Ambiguity





- 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/>

- Pushdown Automata (PDA)

Jihyeok Park

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

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