

# Context Free Language and Context Free Grammar

Noman Amin  
Lecturer, Dept. of CSE  
Southeast University

# CFG

There are four important components in a grammatical description of a language:

1. There is a finite set of symbols that form the strings of the language being defined. This set was  $\{0, 1\}$  in the palindrome example we just saw. We call this alphabet the *terminals*, or *terminal symbols*.
2. There is a finite set of *variables*, also called sometimes *nonterminals* or *syntactic categories*. Each variable represents a language; i.e., a set of strings. In our example above, there was only one variable,  $P$ , which we used to represent the class of palindromes over alphabet  $\{0, 1\}$ .
3. One of the variables represents the language being defined; it is called the *start symbol*. Other variables represent auxiliary classes of strings that are used to help define the language of the start symbol. In our example,  $P$ , the only variable, is the start symbol.
4. There is a finite set of *productions* or *rules* that represent the recursive definition of a language. Each production consists of:
  - (a) A variable that is being (partially) defined by the production. This variable is often called the *head* of the production.
  - (b) The production symbol  $\rightarrow$ .
  - (c) A string of zero or more terminals and variables. This string, called the *body* of the production, represents one way to form strings in the language of the variable of the head. In so doing, we leave terminals unchanged and substitute for each variable of the body any string that is known to be in the language of that variable.

# Example

1.  $P \rightarrow \epsilon$
2.  $P \rightarrow 0$
3.  $P \rightarrow 1$
4.  $P \rightarrow 0P0$
5.  $P \rightarrow 1P1$

Figure 5.1: A context-free grammar for palindromes

The four components just described form a *context-free grammar*, or just *grammar*, or *CFG*. We shall represent a CFG  $G$  by its four components, that is,  $G = (V, T, P, S)$ , where  $V$  is the set of variables,  $T$  the terminals,  $P$  the set of productions, and  $S$  the start symbol.

# Example

1.  $E \rightarrow I$
2.  $E \rightarrow E + E$
3.  $E \rightarrow E * E$
4.  $E \rightarrow (E)$
5.  $I \rightarrow a$
6.  $I \rightarrow b$
7.  $I \rightarrow Ia$
8.  $I \rightarrow Ib$
9.  $I \rightarrow I0$
10.  $I \rightarrow I1$

Figure 5.2: A context-free grammar for simple expressions

The grammar for expressions is stated formally as  $G = (\{E, I\}, T, P, E)$ , where  $T$  is the set of symbols  $\{+, *, (, ), a, b, 0, 1\}$  and  $P$  is the set of productions shown in Fig. 5.2. We interpret the productions as follows.

# Derivation

**Example 5.5:** The inference that  $a * (a + b00)$  is in the language of variable  $E$  can be reflected in a derivation of that string, starting with the string  $E$ . Here is one such derivation:

$$E \Rightarrow E * E \Rightarrow I * E \Rightarrow a * E \Rightarrow$$

$$a * (E) \Rightarrow a * (E + E) \Rightarrow a * (I + E) \Rightarrow a * (a + E) \Rightarrow$$

$$a * (a + I) \Rightarrow a * (a + I0) \Rightarrow a * (a + I00) \Rightarrow a * (a + b00)$$

# Derivation



**Leftmost derivation** – A leftmost derivation is obtained by applying production to the leftmost variable in each step.



**Rightmost derivation** – A rightmost derivation is obtained by applying production to the rightmost variable in each step.

# Leftmost Derivation

**Example 5.6:** The derivation of Example 5.5 was actually a leftmost derivation. Thus, we can describe the same derivation by:

$$E \Rightarrow_{lm} E * E \Rightarrow_{lm} I * E \Rightarrow_{lm} a * E \Rightarrow_{lm}$$

$$a * (E) \Rightarrow_{lm} a * (E + E) \Rightarrow_{lm} a * (I + E) \Rightarrow_{lm} a * (a + E) \Rightarrow_{lm}$$

$$a * (a + I) \Rightarrow_{lm} a * (a + I0) \Rightarrow_{lm} a * (a + I00) \Rightarrow_{lm} a * (a + b00)$$

# Right most Derivation

$$E \Rightarrow_{rm} E * E \Rightarrow_{rm} E * (E) \Rightarrow_{rm} E * (E + E) \Rightarrow_{rm}$$

$$E * (E + I) \Rightarrow_{rm} E * (E + I0) \Rightarrow_{rm} E * (E + I00) \Rightarrow_{rm} E * (E + b00) \Rightarrow_{rm}$$

$$E * (I + b00) \Rightarrow_{rm} E * (a + b00) \Rightarrow_{rm} I * (a + b00) \Rightarrow_{rm} a * (a + b00)$$

# Parse Tree

What are Derivation Tree?

A Derivation Tree, also known as a **Parse Tree**, is a visual representation of the process by which a context-free grammar generates a particular string. It provides a hierarchical breakdown of the string. This illustrates the sequence of production rules applied to derive the string from the grammar's start symbol.

# Parse Tree

## Key Elements of a Derivation Tree

Before learning how to construct a derivation tree, let's understand their essential components:

- **Root Vertex** – The root vertex represents the starting point of the derivation process and is always labeled with the grammar's start symbol.
- **Vertices (Internal Nodes)** – These nodes represent the non-terminal symbols of the grammar, serving as intermediate stages in the derivation.
- **Leaves** – These nodes represent the terminal symbols of the grammar, forming the final string derived from the grammar. They can also be labeled with the empty symbol,  $\epsilon$ , if the grammar allows for empty productions.

# Parse Tree Derivation

## Left Derivation Tree and Right Derivation Tree

There are two primary methods for constructing derivation trees: The left derivation and right derivation. These methods dictate the order in which production rules are applied to non-terminal symbols within the sentential form.

- **Left Derivation Tree** – A Left Derivation Tree is generated by consistently applying production rules to the leftmost variable in each step of the derivation. This method uses expanding the non-terminal symbols on the left side of the sentential form.
- **Right Derivation Tree** – A Right Derivation Tree is obtained by applying production rules to the rightmost variable in each step. This method uses expanding non-terminal symbols on the right side of the sentential form.

# Example

1.  $E \rightarrow I$
2.  $E \rightarrow E + E$
3.  $E \rightarrow E * E$
4.  $E \rightarrow (E)$
5.  $I \rightarrow a$
6.  $I \rightarrow b$
7.  $I \rightarrow Ia$
8.  $I \rightarrow Ib$
9.  $I \rightarrow I0$
10.  $I \rightarrow I1$

Figure 5.2: A context-free grammar for simple expressions

1.  $P \rightarrow \epsilon$
2.  $P \rightarrow 0$
3.  $P \rightarrow 1$
4.  $P \rightarrow 0P0$
5.  $P \rightarrow 1P1$

Figure 5.1: A context-free grammar for palindromes

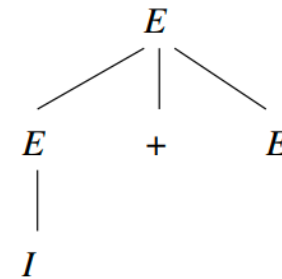


Figure 5.4: A parse tree showing the derivation of  $I + E$  from  $E$

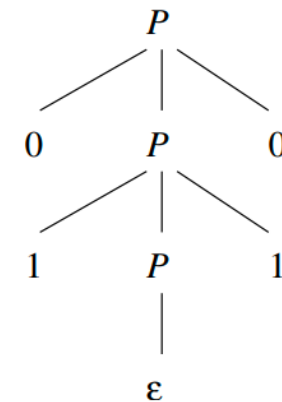


Figure 5.5: A parse tree showing the derivation  $P \xRightarrow{*} 0110$

# Example

1.  $E \rightarrow I$
2.  $E \rightarrow E + E$
3.  $E \rightarrow E * E$
4.  $E \rightarrow (E)$
5.  $I \rightarrow a$
6.  $I \rightarrow b$
7.  $I \rightarrow Ia$
8.  $I \rightarrow Ib$
9.  $I \rightarrow I0$
10.  $I \rightarrow I1$

Figure 5.2: A context-free grammar for simple expressions

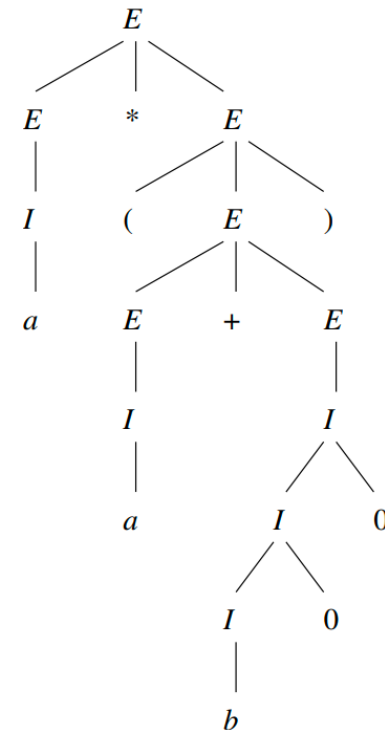


Figure 5.6: Parse tree showing  $a * (a + b00)$  is in the language of our expression grammar