Lesson 3. Context-free languages

CSIE 3110 – Formal Languages and Automata Theory

Tony Tan
Department of Computer Science and Information Engineering
College of Electrical Engineering and Computer Science
National Taiwan University

Table of contents

Table of contents

Context-free grammar (CFG)

(Def.) A *context-free grammar* (CFG) is a system $\mathcal{G} = \langle \Sigma, V, R, S \rangle$, where each component is as follows.

- Σ is a finite set of symbols, called *terminals*.
- V is a finite set of *variables*, and $V \cap \Sigma = \emptyset$.
- R is a finite set of *rules*, where each rule is of the form $A \to w$, where $A \in V$ and $w \in (V \cup \Sigma)^*$.
- S is a special variable from V called the *start variable*.

Context-free grammar (CFG)

(Def.) A *context-free grammar* (CFG) is a system $\mathcal{G} = \langle \Sigma, V, R, S \rangle$, where each component is as follows.

- Σ is a finite set of symbols, called *terminals*.
- V is a finite set of *variables*, and $V \cap \Sigma = \emptyset$.
- R is a finite set of *rules*, where each rule is of the form $A \to w$, where $A \in V$ and $w \in (V \cup \Sigma)^*$.
- S is a special variable from V called the *start variable*.

Note that for every variable $A \in V$, there may be several rules, say

$$A \to w_1, \quad A \to w_2, \ldots\ldots, \quad A \to w_m \qquad\qquad \text{in } R$$

Context-free grammar (CFG)

(Def.) A *context-free grammar* (CFG) is a system $\mathcal{G} = \langle \Sigma, V, R, S \rangle$, where each component is as follows.

- Σ is a finite set of symbols, called *terminals*.
- V is a finite set of *variables*, and $V \cap \Sigma = \emptyset$.
- R is a finite set of *rules*, where each rule is of the form $A \to w$, where $A \in V$ and $w \in (V \cup \Sigma)^*$.
- S is a special variable from V called the *start variable*.

Note that for every variable $A \in V$, there may be several rules, say

$$A \to w_1, \quad A \to w_2, \ldots\ldots, \quad A \to w_m \qquad\qquad \text{in } R$$

usually abbreviated as:

$$A \quad \to \quad w_1 \mid w_2 \mid \cdots \mid w_m$$

Note also that we may have a rule of the form $A \to \varepsilon$.

Some examples of CFG

(Example 1) $\mathcal{G}_1 = \langle \Sigma, V, R, S \rangle$ where:

- $\Sigma = \{a, b\}$.
- $V = \{S\}$.
- R contains the rules: $S \to aSb \mid \varepsilon$.
- S is the start variable.

Some examples of CFG

(Example 2) $\mathcal{G}_2 = \langle \Sigma, V, R, S \rangle$ where:

- $\Sigma = \{a, b\}$.
- $V = \{S, T\}$.
- R contains the rules: $T \to SS$ and $S \to aSb \mid \varepsilon$.
- T is the start variable.

Some examples of CFG

(Example 3) $\mathcal{G}_3 = \langle \Sigma, V, R, S \rangle$ where:

- $\Sigma = \{a, b\}$.
- $V = \{S\}$.
- R contains the rules: $S \to S$.
- S is the start variable.

Some examples of CFG

(Example 4) $\mathcal{G}_4 = \langle \Sigma, V, R, S \rangle$ where:

- $\Sigma = \{a, b\}$.
- $V = \{S, X, A, B, C\}$.
- R contains the rules:

$$\begin{array}{rcl} S & \rightarrow & XAXBXC \mid AA \mid BBA \mid CCA \\ X & \rightarrow & \varepsilon \mid aX \mid bX \\ A & \rightarrow & aaX \mid bbA \\ B & \rightarrow & baX \mid bbB \\ C & \rightarrow & abX \mid bbC \end{array}$$

- S is the start variable.

Notations and terminology

Similar to DFA/NFA/regex, a CFG represents a language called context-free language (CFL).

Let $\mathcal{G} = \langle \Sigma, V, R, S \rangle$ be a CFG.

Notations and terminology

Similar to DFA/NFA/regex, a CFG represents a language called context-free language (CFL).

Let $\mathcal{G} = \langle \Sigma, V, R, S \rangle$ be a CFG.

(Def.) Let uAv be a word in which a variable $A \in V$ appears.

We say that *uAv yields uwv*, denoted by $uAv \Rightarrow uwv$, if there is a rule $A \to w$ in R.

Notations and terminology

Similar to DFA/NFA/regex, a CFG represents a language called context-free language (CFL).

Let $\mathcal{G} = \langle \Sigma, V, R, S \rangle$ be a CFG.

(Def.) Let uAv be a word in which a variable $A \in V$ appears.

We say that *uAv yields uwv*, denoted by $uAv \Rightarrow uwv$, if there is a rule $A \rightarrow w$ in R.

Intuitively, the rule $A \rightarrow w$ means variable A can be replaced with w.

Notations and terminology

Similar to DFA/NFA/regex, a CFG represents a language called context-free language (CFL).

Let $\mathcal{G} = \langle \Sigma, V, R, S \rangle$ be a CFG.

(Def.) Let uAv be a word in which a variable $A \in V$ appears.

We say that *uAv yields uwv*, denoted by $uAv \Rightarrow uwv$, if there is a rule $A \to w$ in R.

Intuitively, the rule $A \to w$ means variable A can be replaced with w.

(Def.) For $x, y \in (\Sigma \cup V)^*$, we say that *$x$ derives y*, denoted by $x \Rightarrow^* y$, if either $x = y$, or $x \Rightarrow z_1 \Rightarrow z_2 \Rightarrow \cdots \Rightarrow y$ (finitely many).

Notations and terminology

Similar to DFA/NFA/regex, a CFG represents a language called context-free language (CFL).

Let $\mathcal{G} = \langle \Sigma, V, R, S \rangle$ be a CFG.

(Def.) Let uAv be a word in which a variable $A \in V$ appears.

We say that uAv *yields* uwv, denoted by $uAv \Rightarrow uwv$, if there is a rule $A \to w$ in R.

Intuitively, the rule $A \to w$ means variable A can be replaced with w.

(Def.) For $x, y \in (\Sigma \cup V)^*$, we say that x *derives* y, denoted by $x \Rightarrow^* y$, if either $x = y$, or $x \Rightarrow z_1 \Rightarrow z_2 \Rightarrow \cdots \Rightarrow y$ (finitely many).

We will also say "y is derived from x" or "from x we can derive y."

Notations and terminology — continued

(Def.) For a variable A, $L(\mathcal{G}, A)$ denotes the language of all words over Σ that can be derived from variable A. Formally,

$$L(\mathcal{G}, A) \quad = \quad \{w \in \Sigma^* \;\mid\; A \Rightarrow^* w\}$$

Notations and terminology — continued

(Def.) For a variable A, $L(\mathcal{G}, A)$ denotes the language of all words over Σ that can be derived from variable A. Formally,

$$L(\mathcal{G}, A) \quad = \quad \{w \in \Sigma^* \ \mid \ A \Rightarrow^* w\}$$

(Def.) $L(\mathcal{G})$ denotes the language $L(\mathcal{G}, S)$, i.e., the language of all words over Σ that can be derived from the start variable S. Formally,

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

Notations and terminology — continued

(Def.) For a variable A, $L(\mathcal{G}, A)$ denotes the language of all words over Σ that can be derived from variable A. Formally,

$$L(\mathcal{G}, A) \quad = \quad \{w \in \Sigma^* \ \mid \ A \Rightarrow^* w\}$$

(Def.) $L(\mathcal{G})$ denotes the language $L(\mathcal{G}, S)$, i.e., the language of all words over Σ that can be derived from the start variable S. Formally,

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

$L(\mathcal{G})$ is called the language generated/defined/derived from/by $\mathcal{G}$.

Notations and terminology — continued

(Def.) For a variable A, $L(\mathcal{G}, A)$ denotes the language of all words over Σ that can be derived from variable A. Formally,

$$L(\mathcal{G}, A) \quad = \quad \{w \in \Sigma^* \mid A \Rightarrow^* w\}$$

(Def.) $L(\mathcal{G})$ denotes the language $L(\mathcal{G}, S)$, i.e., the language of all words over Σ that can be derived from the start variable S. Formally,

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

$L(\mathcal{G})$ is called the language generated/defined/derived from/by $\mathcal{G}$.

(Def.) A language L is called a *context-free language* (CFL), if there is a CFG $\mathcal{G}$ such that $L(\mathcal{G}) = L$.

Example 1: $\mathcal{G}_1 = \langle \Sigma, V, R, S \rangle$

$\Sigma = \{a, b\}$, $V = \{S\}$, S is the start variable and R contains the rule:

$$S \rightarrow aSb \mid \varepsilon$$

Example 1: $\mathcal{G}_1 = \langle \Sigma, V, R, S \rangle$

$\Sigma = \{a, b\}$, $V = \{S\}$, S is the start variable and R contains the rule:

$$S \quad \rightarrow \quad aSb \mid \varepsilon$$

- $S \Rightarrow \varepsilon$.

Example 1: $\mathcal{G}_1 = \langle \Sigma, V, R, S \rangle$

$\Sigma = \{a, b\}$, $V = \{S\}$, S is the start variable and R contains the rule:

$$S \quad \rightarrow \quad aSb \mid \varepsilon$$

- $S \Rightarrow \varepsilon$. So, $\varepsilon \in L(\mathcal{G}_1)$.

Example 1: $\mathcal{G}_1 = \langle \Sigma, V, R, S \rangle$

$\Sigma = \{a, b\}$, $V = \{S\}$, S is the start variable and R contains the rule:

$$S \quad \rightarrow \quad aSb \mid \varepsilon$$

- $S \Rightarrow \varepsilon$. So, $\varepsilon \in L(\mathcal{G}_1)$.
- $S \Rightarrow aSb \Rightarrow ab$.

Example 1: $\mathcal{G}_1 = \langle \Sigma, V, R, S \rangle$

$\Sigma = \{a, b\}$, $V = \{S\}$, S is the start variable and R contains the rule:

$$S \quad \rightarrow \quad aSb \mid \varepsilon$$

- $S \Rightarrow \varepsilon$. So, $\varepsilon \in L(\mathcal{G}_1)$.
- $S \Rightarrow aSb \Rightarrow ab$. So, $ab \in L(\mathcal{G}_1)$.

Example 1: $\mathcal{G}_1 = \langle \Sigma, V, R, S \rangle$

$\Sigma = \{a, b\}$, $V = \{S\}$, S is the start variable and R contains the rule:

$$S \quad \to \quad aSb \mid \varepsilon$$

- $S \Rightarrow \varepsilon$. So, $\varepsilon \in L(\mathcal{G}_1)$.
- $S \Rightarrow aSb \Rightarrow ab$. So, $ab \in L(\mathcal{G}_1)$.
- $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$. So, $aabb \in L(\mathcal{G}_1)$.

Example 1: $\mathcal{G}_1 = \langle \Sigma, V, R, S \rangle$

$\Sigma = \{a, b\}$, $V = \{S\}$, S is the start variable and R contains the rule:

$$S \quad \rightarrow \quad aSb \mid \varepsilon$$

- $S \Rightarrow \varepsilon$. So, $\varepsilon \in L(\mathcal{G}_1)$.
- $S \Rightarrow aSb \Rightarrow ab$. So, $ab \in L(\mathcal{G}_1)$.
- $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$. So, $aabb \in L(\mathcal{G}_1)$.
- In general, for every integer $n \geqslant 0$:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow \cdots \Rightarrow \underbrace{a \cdots a}_{n}\underbrace{b \cdots b}_{n}$$

That is, $S \Rightarrow^* a^n b^n$, i.e., $a^n b^n \in L(\mathcal{G}_1)$, for every integer $n \geqslant 0$.

Example 1: $\mathcal{G}_1 = \langle \Sigma, V, R, S \rangle$

$\Sigma = \{a, b\}$, $V = \{S\}$, S is the start variable and R contains the rule:

$$S \quad \rightarrow \quad aSb \mid \varepsilon$$

- $S \Rightarrow \varepsilon$. So, $\varepsilon \in L(\mathcal{G}_1)$.
- $S \Rightarrow aSb \Rightarrow ab$. So, $ab \in L(\mathcal{G}_1)$.
- $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$. So, $aabb \in L(\mathcal{G}_1)$.
- In general, for every integer $n \geqslant 0$:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow \cdots \Rightarrow \underbrace{a \cdots a}_{n}\underbrace{b \cdots b}_{n}$$

  That is, $S \Rightarrow^* a^n b^n$, i.e., $a^n b^n \in L(\mathcal{G}_1)$, for every integer $n \geqslant 0$.
- Is $ba \in L(\mathcal{G}_1)$? Is $aab \in L(\mathcal{G}_1)$?

Example 1: $\mathcal{G}_1 = \langle \Sigma, V, R, S \rangle$

$\Sigma = \{a, b\}$, $V = \{S\}$, S is the start variable and R contains the rule:

$$S \quad \to \quad aSb \mid \varepsilon$$

- $S \Rightarrow \varepsilon$. So, $\varepsilon \in L(\mathcal{G}_1)$.
- $S \Rightarrow aSb \Rightarrow ab$. So, $ab \in L(\mathcal{G}_1)$.
- $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$. So, $aabb \in L(\mathcal{G}_1)$.
- In general, for every integer $n \geqslant 0$:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow \cdots \Rightarrow \underbrace{a \cdots a}_{n}\underbrace{b \cdots b}_{n}$$

  That is, $S \Rightarrow^* a^n b^n$, i.e., $a^n b^n \in L(\mathcal{G}_1)$, for every integer $n \geqslant 0$.
- Is $ba \in L(\mathcal{G}_1)$? Is $aab \in L(\mathcal{G}_1)$? No!

Example 1: $\mathcal{G}_1 = \langle \Sigma, V, R, S \rangle$

$\Sigma = \{a, b\}$, $V = \{S\}$, S is the start variable and R contains the rule:

$$S \quad \to \quad aSb \mid \varepsilon$$

- $S \Rightarrow \varepsilon$. So, $\varepsilon \in L(\mathcal{G}_1)$.
- $S \Rightarrow aSb \Rightarrow ab$. So, $ab \in L(\mathcal{G}_1)$.
- $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$. So, $aabb \in L(\mathcal{G}_1)$.
- In general, for every integer $n \geqslant 0$:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow \cdots \Rightarrow \underbrace{a \cdots a}_{n} \underbrace{b \cdots b}_{n}$$

  That is, $S \Rightarrow^* a^n b^n$, i.e., $a^n b^n \in L(\mathcal{G}_1)$, for every integer $n \geqslant 0$.
- Is $ba \in L(\mathcal{G}_1)$? Is $aab \in L(\mathcal{G}_1)$? No!

In fact,

$$L(\mathcal{G}_1) \quad = \quad \{a^n b^n \mid n \geqslant 0\}$$

Example 2: $\mathcal{G}_2 = \langle \Sigma, V, R, S \rangle$

$\Sigma = \{a, b\}$, $V = \{S, T\}$, T is the start variable and R contains the rules:

$$T \to SS \qquad \text{and} \qquad S \to aSb \mid \varepsilon$$

Example 2: $\mathcal{G}_2 = \langle \Sigma, V, R, S \rangle$

$\Sigma = \{a, b\}$, $V = \{S, T\}$, T is the start variable and R contains the rules:

$$T \to SS \qquad \text{and} \qquad S \to aSb \mid \varepsilon$$

- $T \Rightarrow SS \Rightarrow \varepsilon S \Rightarrow \varepsilon$.

Example 2: $\mathcal{G}_2 = \langle \Sigma, V, R, S \rangle$

$\Sigma = \{a, b\}$, $V = \{S, T\}$, T is the start variable and R contains the rules:

$$T \to SS \qquad \text{and} \qquad S \to aSb \mid \varepsilon$$

- $T \Rightarrow SS \Rightarrow \varepsilon S \Rightarrow \varepsilon$. So, $\varepsilon \in L(\mathcal{G}_2)$.

Example 2: $\mathcal{G}_2 = \langle \Sigma, V, R, S \rangle$

$\Sigma = \{a, b\}$, $V = \{S, T\}$, T is the start variable and R contains the rules:

$$T \to SS \qquad \text{and} \qquad S \to aSb \mid \varepsilon$$

- $T \Rightarrow SS \Rightarrow \varepsilon S \Rightarrow \varepsilon$. So, $\varepsilon \in L(\mathcal{G}_2)$.
- $T \Rightarrow SS \Rightarrow aSbS \Rightarrow^* ab$.

Example 2: $\mathcal{G}_2 = \langle \Sigma, V, R, S \rangle$

$\Sigma = \{a, b\}$, $V = \{S, T\}$, T is the start variable and R contains the rules:

$$T \to SS \qquad \text{and} \qquad S \to aSb \mid \varepsilon$$

- $T \Rightarrow SS \Rightarrow \varepsilon S \Rightarrow \varepsilon$. So, $\varepsilon \in L(\mathcal{G}_2)$.
- $T \Rightarrow SS \Rightarrow aSbS \Rightarrow^* ab$. So, $\boldsymbol{ab} \in L(\mathcal{G}_2)$.

Example 2: $\mathcal{G}_2 = \langle \Sigma, V, R, S \rangle$

$\Sigma = \{a, b\}$, $V = \{S, T\}$, T is the start variable and R contains the rules:

$$T \to SS \qquad \text{and} \qquad S \to aSb \mid \varepsilon$$

- $T \Rightarrow SS \Rightarrow \varepsilon S \Rightarrow \varepsilon$. So, $\varepsilon \in L(\mathcal{G}_2)$.
- $T \Rightarrow SS \Rightarrow aSbS \Rightarrow^* ab$. So, $ab \in L(\mathcal{G}_2)$.
- $T \Rightarrow SS \Rightarrow aSbS \Rightarrow aSbaSb \Rightarrow^* abab$.

Example 2: $\mathcal{G}_2 = \langle \Sigma, V, R, S \rangle$

$\Sigma = \{a, b\}$, $V = \{S, T\}$, T is the start variable and R contains the rules:

$$T \to SS \qquad \text{and} \qquad S \to aSb \mid \varepsilon$$

- $T \Rightarrow SS \Rightarrow \varepsilon S \Rightarrow \varepsilon$. So, $\varepsilon \in L(\mathcal{G}_2)$.
- $T \Rightarrow SS \Rightarrow aSbS \Rightarrow^* ab$. So, $ab \in L(\mathcal{G}_2)$.
- $T \Rightarrow SS \Rightarrow aSbS \Rightarrow aSbaSb \Rightarrow^* abab$. So, $abab \in L(\mathcal{G}_2)$.

Example 2: $\mathcal{G}_2 = \langle \Sigma, V, R, S \rangle$

$\Sigma = \{a, b\}$, $V = \{S, T\}$, T is the start variable and R contains the rules:

$$T \to SS \qquad \text{and} \qquad S \to aSb \mid \varepsilon$$

- $T \Rightarrow SS \Rightarrow \varepsilon S \Rightarrow \varepsilon$. So, $\varepsilon \in L(\mathcal{G}_2)$.
- $T \Rightarrow SS \Rightarrow aSbS \Rightarrow^* ab$. So, $ab \in L(\mathcal{G}_2)$.
- $T \Rightarrow SS \Rightarrow aSbS \Rightarrow aSbaSb \Rightarrow^* abab$. So, $abab \in L(\mathcal{G}_2)$.
- In general, for every integer $n, k \geqslant 0$:

$$T \quad \Rightarrow^* \quad a^n b^n a^k b^k$$

Example 2: $\mathcal{G}_2 = \langle \Sigma, V, R, S \rangle$

$\Sigma = \{a, b\}$, $V = \{S, T\}$, T is the start variable and R contains the rules:

$$T \to SS \qquad \text{and} \qquad S \to aSb \mid \varepsilon$$

- $T \Rightarrow SS \Rightarrow \varepsilon S \Rightarrow \varepsilon$. So, $\varepsilon \in L(\mathcal{G}_2)$.
- $T \Rightarrow SS \Rightarrow aSbS \Rightarrow^* ab$. So, $ab \in L(\mathcal{G}_2)$.
- $T \Rightarrow SS \Rightarrow aSbS \Rightarrow aSbaSb \Rightarrow^* abab$. So, $abab \in L(\mathcal{G}_2)$.
- In general, for every integer $n, k \geqslant 0$:

$$T \quad \Rightarrow^* \quad a^n b^n a^k b^k$$

That is, $a^n b^n a^k b^k \in L(\mathcal{G}_2)$, for every integer $n, k \geqslant 0$.

Example 2: $\mathcal{G}_2 = \langle \Sigma, V, R, S \rangle$

$\Sigma = \{a, b\}$, $V = \{S, T\}$, T is the start variable and R contains the rules:

$$T \to SS \qquad \text{and} \qquad S \to aSb \mid \varepsilon$$

- $T \Rightarrow SS \Rightarrow \varepsilon S \Rightarrow \varepsilon$. So, $\varepsilon \in L(\mathcal{G}_2)$.
- $T \Rightarrow SS \Rightarrow aSbS \Rightarrow^* ab$. So, $ab \in L(\mathcal{G}_2)$.
- $T \Rightarrow SS \Rightarrow aSbS \Rightarrow aSbaSb \Rightarrow^* abab$. So, $abab \in L(\mathcal{G}_2)$.
- In general, for every integer $n, k \geqslant 0$:

$$T \quad \Rightarrow^* \quad a^n b^n a^k b^k$$

That is, $a^n b^n a^k b^k \in L(\mathcal{G}_2)$, for every integer $n, k \geqslant 0$.

In fact,

$$L(\mathcal{G}_2) \quad = \quad \{a^n b^n a^k b^k \mid n, k \geqslant 0\}$$

Example 3: $\mathcal{G}_3 = \langle \Sigma, V, R, S \rangle$

$\Sigma = \{a, b\}$, $V = \{S\}$, S is the start variable and R contains the rules:

$$S \rightarrow S$$

Example 3: $\mathcal{G}_3 = \langle \Sigma, V, R, S \rangle$

$\Sigma = \{a, b\}$, $V = \{S\}$, S is the start variable and R contains the rules:

$$S \quad \rightarrow \quad S$$

We can only have:

$$S \Rightarrow S \Rightarrow S \Rightarrow \cdots$$

Example 3: $\mathcal{G}_3 = \langle \Sigma, V, R, S \rangle$

$\Sigma = \{a, b\}$, $V = \{S\}$, S is the start variable and R contains the rules:

$$S \quad \rightarrow \quad S$$

We can only have:

$$S \Rightarrow S \Rightarrow S \Rightarrow \cdots$$

Thus,

$$L(\mathcal{G}_3) \quad = \quad \emptyset$$

Example 4: The C++ programming language

We can define a CFG that consists of all syntactically correct C++ programs.

Example 4: The C++ programming language

We can define a CFG that consists of all syntactically correct C++ programs.

The alphabet is the set of symbols in the keyboard.

Example 4: The C++ programming language

We can define a CFG that consists of all syntactically correct C++ programs.

The alphabet is the set of symbols in the keyboard.

The variables are A_{if}, A_{while}, $A_{\text{1-ins}}$, A_{seq}, ...

Example 4: The C++ programming language

We can define a CFG that consists of all syntactically correct C++ programs.

The alphabet is the set of symbols in the keyboard.

The variables are A_{if}, A_{while}, $A_{\text{1-ins}}$, A_{seq}, ...

Rules such as:

$$
\begin{aligned}
A_{\text{seq}} &\to A_{\text{1-ins}} \mid A_{\text{1-ins}} A_{\text{seq}} \\
A_{\text{if}} &\to \texttt{if } A_{\text{bool-cond}} \{ A_{\text{seq}} \} \mid \texttt{if } A_{\text{bool-cond}} \{ A_{\text{seq}} \} \texttt{ else } \{ A_{\text{seq}} \} \\
A_{\text{while}} &\to \texttt{while } A_{\text{bool-cond}} \{ A_{\text{seq}} \} \\
&\vdots
\end{aligned}
$$

Example 4: The C++ programming language

We can define a CFG that consists of all syntactically correct C++ programs.

The alphabet is the set of symbols in the keyboard.

The variables are A_{if}, A_{while}, $A_{\text{1-ins}}$, A_{seq}, $\ldots$.

Rules such as:

$$\begin{aligned}
A_{\text{seq}} &\rightarrow A_{\text{1-ins}} \mid A_{\text{1-ins}}A_{\text{seq}} \\
A_{\text{if}} &\rightarrow \texttt{if } A_{\text{bool-cond}} \; \{ \; A_{\text{seq}} \; \} \mid \texttt{if } A_{\text{bool-cond}} \; \{ \; A_{\text{seq}} \; \} \; \texttt{else} \; \{ \; A_{\text{seq}} \; \} \\
A_{\text{while}} &\rightarrow \texttt{while } A_{\text{bool-cond}} \; \{ \; A_{\text{seq}} \; \} \\
&\vdots
\end{aligned}$$

In fact, any programming language is defined by a CFG.

Closure under union, concatenation and Kleene star

Theorem 3.2
Context-free languages are closed under union, concatenation and Kleene star.

(Proof) Let $\mathcal{G}_1 = \langle \Sigma, V_1, R_1, S_1 \rangle$ and $\mathcal{G}_2 = \langle \Sigma, V_2, R_2, S_2 \rangle$. First, we rename the variables in V_1 and V_2 such that $V_1 \cap V_2 = \emptyset$.

Closure under union, concatenation and Kleene star

Theorem 3.2
Context-free languages are closed under union, concatenation and Kleene star.

(Proof) Let $\mathcal{G}_1 = \langle \Sigma, V_1, R_1, S_1 \rangle$ and $\mathcal{G}_2 = \langle \Sigma, V_2, R_2, S_2 \rangle$. First, we rename the variables in V_1 and V_2 such that $V_1 \cap V_2 = \emptyset$.

(Closure under union)

Consider the CFG $\mathcal{G} = \langle \Sigma, V, R, S \rangle$ defined as follows.

- $V = V_1 \cup V_2 \cup \{S\}$, where S is a "new" variable, i.e., $S \notin V_1 \cup V_2$.
- $R = R_1 \cup R_2 \cup \{S \to S_1 | S_2\}$.
- S is the start variable.

It can be verified that $L(\mathcal{G}) = L(\mathcal{G}_1) \cup L(\mathcal{G}_2)$.

Closure under union, concatenation and Kleene star

(Proof — continued)

(Closure under concatenation)

Consider the CFG $\mathcal{G} = \langle \Sigma, V, R, S \rangle$ defined as follows.

- $V = V_1 \cup V_2 \cup \{S\}$, where S is a "new" variable, i.e., $S \notin V_1 \cup V_2$.
- $R = R_1 \cup R_2 \cup \{S \to S_1 S_2\}$.
- S is the start variable.

It can be verified that $L(\mathcal{G}) = L(\mathcal{G}_1)L(\mathcal{G}_2)$.

Closure under union, concatenation and Kleene star

(Proof — continued)

(Closure under Kleene star)

Consider the CFG $\mathcal{G} = \langle \Sigma, V, R, S \rangle$ defined as follows.

- $V = V_1 \cup \{S\}$, where S is a "new" variable, i.e., $S \notin V_1$.
- $R = R_1 \cup \{S \to S_1 S | \varepsilon\}$.
- S is the start variable.

It can be verified that $L(\mathcal{G}) = L(\mathcal{G}_1)^*$. See Note 3 for more details.

Closure under union, concatenation and Kleene star

Theorem 3.2

Context-free languages are closed under union, concatenation and Kleene star.

Closure under union, concatenation and Kleene star

Theorem 3.2
Context-free languages are closed under union, concatenation and Kleene star.

Later we will see that context-free languages are <u>not</u> closed under intersection and complement.

Regular languages are CFL

Theorem 3.3

Every regular language is a context-free language.

Regular languages are CFL

Theorem 3.3

Every regular language is a context-free language.

For the proof, use Theorem 3.2. The details are left as homework.

Regular languages are CFL

Theorem 3.3

Every regular language is a context-free language.

For the proof, use Theorem 3.2. The details are left as homework.

(Note:) There is a context-free language that is not regular!

Table of contents

Derivation trees as an alternative condition for CFL membership

(Def.) A *derivation tree*, or a *parse tree*, of a CFG $\mathcal{G} = \langle \Sigma, V, R, S \rangle$ is a tree T in which:

- every vertex has a *label*, which is a symbol from $V \cup \Sigma \cup \{\varepsilon\}$;
- the label of an interior vertex is a variable from V;
- the label of a leaf vertex is either ε or a terminal from Σ;
- if an interior vertex has a label $A \in V$ and it has k children $n_1, \ldots, n_k$ (in the order from left to right) with labels $X_1, \ldots, X_k$, respectively, then $A \to X_1 \cdots X_k$ must be a rule in R.

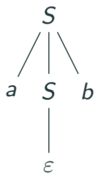
An example

- Every vertex has a *label*, which is a symbol from $V \cup \Sigma \cup \{\varepsilon\}$.
- The label of an interior vertex is a variable from V.
- The label of a leaf vertex is either ε or a terminal from Σ.
- If an interior vertex has a label $A \in V$ and it has k children $n_1, \ldots, n_k$ (in the order from left to right) with labels $X_1, \ldots, X_k$, respectively, then $A \to X_1 \cdots X_k$ must be a rule in R.

(Example) Consider a CFG with the following rules:

$$T \to SS \qquad \text{and} \qquad S \to aSb \mid \varepsilon$$

S, T are variables and the alphabet is $\{a, b\}$.



Definition of derivation trees — continued

(Def.) If the label of the root is a variable A, and the leaf vertices of T are $n_1, \ldots, n_m$ (in the order from left to right) with labels $u_1, \ldots, u_m$, we say that T is the *derivation tree of $\mathcal{G}$ from variable A on word $u_1 \cdots u_m$*.

Definition of derivation trees — continued

(Def.) If the label of the root is a variable A, and the leaf vertices of T are $n_1, \ldots, n_m$ (in the order from left to right) with labels $u_1, \ldots, u_m$, we say that T is the *derivation tree of $\mathcal{G}$ from variable A on word $u_1 \cdots u_m$*.

(Def.) When the label of the root is the start variable S, we simply say T is the *derivation tree of $\mathcal{G}$ on $u_1 \cdots u_m$*.

Definition of derivation trees — continued

(Def.) If the label of the root is a variable A, and the leaf vertices of T are $n_1, \ldots, n_m$ (in the order from left to right) with labels $u_1, \ldots, u_m$, we say that T is the *derivation tree of $\mathcal{G}$ from variable A on word $u_1 \cdots u_m$*.

(Def.) When the label of the root is the start variable S, we simply say T is the *derivation tree of $\mathcal{G}$ on $u_1 \cdots u_m$*.

Theorem 3.5
Let $\mathcal{G} = \langle \Sigma, V, R, S \rangle$ be a CFG. For every variable $A \in V$, for every word $w \in \Sigma^*$, the following holds.

$$A \Rightarrow^* w \quad \text{if and only if} \quad \text{there is a derivation tree of } \mathcal{G} \text{ from } A \text{ on } w.$$

In particular, $w \in L(\mathcal{G})$ if and only if there is a derivation tree of $\mathcal{G}$ on w.

Definition of derivation trees — continued

(Def.) If the label of the root is a variable A, and the leaf vertices of T are $n_1, \ldots, n_m$ (in the order from left to right) with labels $u_1, \ldots, u_m$, we say that T is the *derivation tree of $\mathcal{G}$ from variable A on word $u_1 \cdots u_m$*.

(Def.) When the label of the root is the start variable S, we simply say T is the *derivation tree of $\mathcal{G}$ on $u_1 \cdots u_m$*.

Theorem 3.5

Let $\mathcal{G} = \langle \Sigma, V, R, S \rangle$ be a CFG. For every variable $A \in V$, for every word $w \in \Sigma^*$, the following holds.

$$A \Rightarrow^* w \quad \text{if and only if} \quad \text{there is a derivation tree of } \mathcal{G} \text{ from } A \text{ on } w.$$

In particular, $w \in L(\mathcal{G})$ if and only if there is a derivation tree of $\mathcal{G}$ on w.

The proof is straightforward, but the idea is best illustrated by examples.

An example of derivation tree

Consider a CFG with the following rules:

$$T \to SS \qquad \text{and} \qquad S \to aSb \mid \varepsilon$$

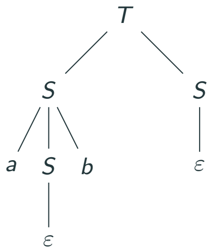
T is the start variable and the alphabet is $\{a, b\}$.

An example of derivation tree

Consider a CFG with the following rules:

$$T \to SS \qquad \text{and} \qquad S \to aSb \mid \varepsilon$$

T is the start variable and the alphabet is $\{a, b\}$.

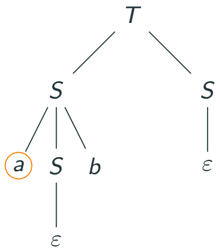


An example of derivation tree

Consider a CFG with the following rules:

$$T \to SS \qquad \text{and} \qquad S \to aSb \mid \varepsilon$$

T is the start variable and the alphabet is $\{a, b\}$.

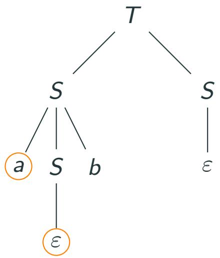


An example of derivation tree

Consider a CFG with the following rules:

$$T \to SS \qquad \text{and} \qquad S \to aSb \mid \varepsilon$$

T is the start variable and the alphabet is $\{a, b\}$.

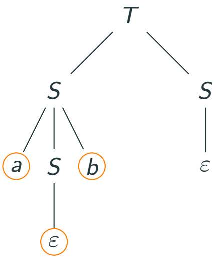


An example of derivation tree

Consider a CFG with the following rules:

$$T \to SS \qquad \text{and} \qquad S \to aSb \mid \varepsilon$$

T is the start variable and the alphabet is $\{a, b\}$.

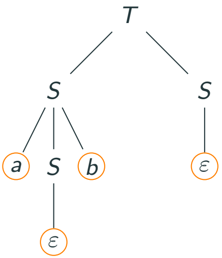


An example of derivation tree

Consider a CFG with the following rules:

$$T \to SS \qquad \text{and} \qquad S \to aSb \mid \varepsilon$$

T is the start variable and the alphabet is $\{a, b\}$.

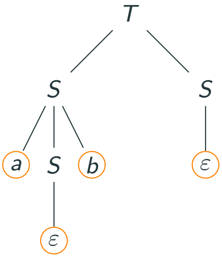


An example of derivation tree

Consider a CFG with the following rules:

$$T \to SS \qquad \text{and} \qquad S \to aSb \mid \varepsilon$$

T is the start variable and the alphabet is $\{a, b\}$.



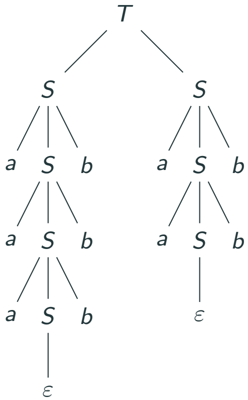
This is a derivation tree of the CFG on ab.

Another example

The rules: $T \to SS$ and $S \to aSb \mid \varepsilon$.

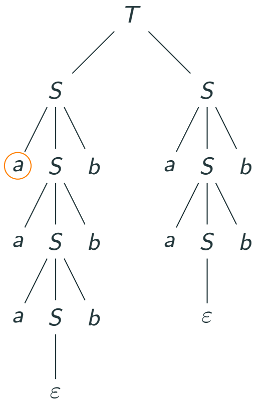
Another example

The rules: $T \to SS$ and $S \to aSb \mid \varepsilon$.



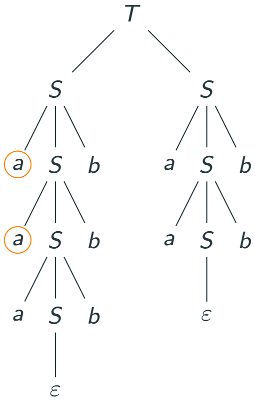
Another example

The rules: $T \to SS$ and $S \to aSb \mid \varepsilon$.



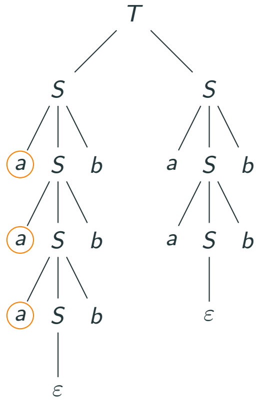
Another example

The rules: $T \to SS$ and $S \to aSb \mid \varepsilon$.



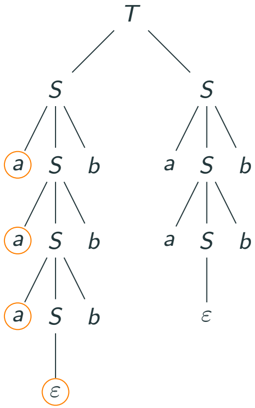
Another example

The rules: $T \to SS$ and $S \to aSb \mid \varepsilon$.



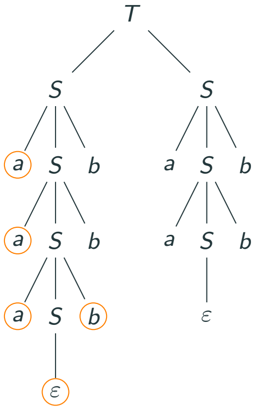
Another example

The rules: $T \to SS$ and $S \to aSb \mid \varepsilon$.



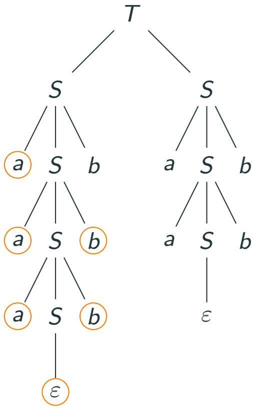
Another example

The rules: $T \to SS$ and $S \to aSb \mid \varepsilon$.



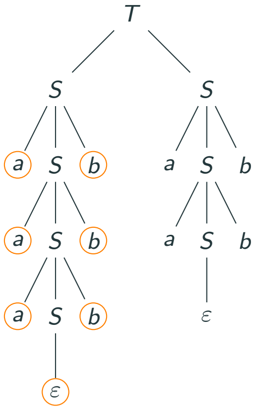
Another example

The rules: $T \to SS$ and $S \to aSb \mid \varepsilon$.



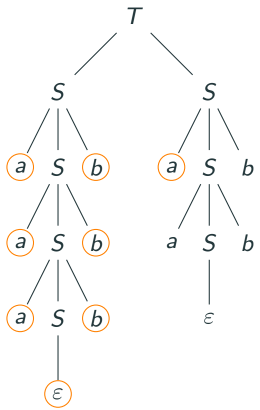
Another example

The rules: $T \to SS$ and $S \to aSb \mid \varepsilon$.



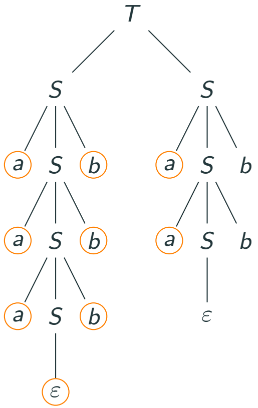
Another example

The rules: $T \to SS$ and $S \to aSb \mid \varepsilon$.



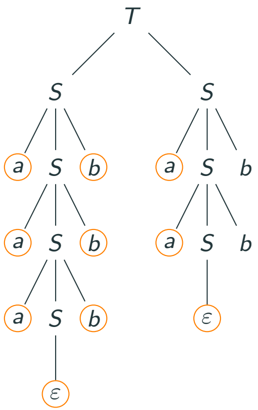
Another example

The rules: $T \to SS$ and $S \to aSb \mid \varepsilon$.



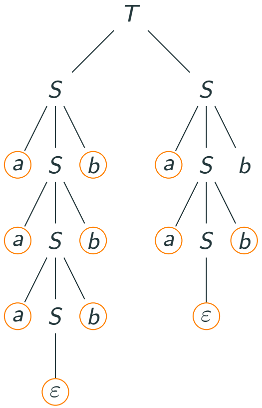
Another example

The rules: $T \to SS$ and $S \to aSb \mid \varepsilon$.



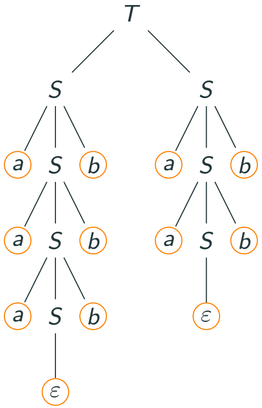
Another example

The rules: $T \to SS$ and $S \to aSb \mid \varepsilon$.



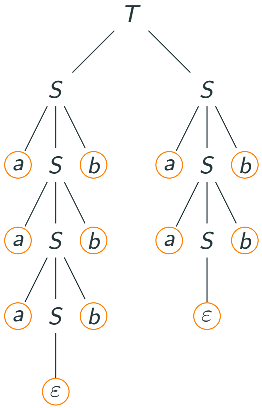
Another example

The rules: $T \to SS$ and $S \to aSb \mid \varepsilon$.



Another example

The rules: $T \to SS$ and $S \to aSb \mid \varepsilon$.



This is a derivation tree of the CFG on $a^3b^3a^2b^2$.

Derivation trees as an alternative condition for CFL membership

From the example, it is not difficult to see that derivation trees are just an alternative condition for CFL membership.

Theorem 3.5

Let $\mathcal{G} = \langle \Sigma, V, R, S \rangle$ be a CFG. For every variable $A \in V$, for every word $w \in \Sigma^*$, the following holds.

$$A \Rightarrow^* w \quad \text{if and only if} \quad \text{there is a derivation tree of } \mathcal{G} \text{ from } A \text{ on } w.$$

In particular, $w \in L(\mathcal{G})$ if and only if there is a derivation tree of $\mathcal{G}$ on w.

Table of contents

Pumping lemma

Similar to regular languages, CFL also has its own pumping lemma.

Pumping lemma

Similar to regular languages, CFL also has its own pumping lemma.

Lemma 3.6 (pumping lemma)
Let $\mathcal{G} = \langle \Sigma, V, R, S \rangle$ be a CFG. Then, there is an integer N such that every $w \in L(\mathcal{G})$ with length $\geqslant N$ can be partitioned into:

$$w \quad = \quad s\ x\ y\ z\ t$$

such that the following holds.

- $|x| + |z| \geqslant 1$.
- $|xyz| \leqslant N$.
- *For every $i \geqslant 0$, $sx^iyz^it \in L(\mathcal{G})$.*

Proof of pumping lemma

Let $\mathcal{G} = \langle \Sigma, V, R, S \rangle$ be a CFG and let $n = |V|$.

Proof of pumping lemma

Let $\mathcal{G} = \langle \Sigma, V, R, S \rangle$ be a CFG and let $n = |V|$.

Let $m = \max_{A \to w \in R} |w|$, i.e., the maximum length of the string u over all the rule $A \to u$ in R.

Proof of pumping lemma

Let $\mathcal{G} = \langle \Sigma, V, R, S \rangle$ be a CFG and let $n = |V|$.

Let $m = \max_{A \to w \in R} |w|$, i.e., the maximum length of the string u over all the rule $A \to u$ in R.

Intuitively, this means that in every derivation tree of $\mathcal{G}$, every node has at most m children.

Proof of pumping lemma

Let $\mathcal{G} = \langle \Sigma, V, R, S \rangle$ be a CFG and let $n = |V|$.

Let $m = \max_{A \to w \in R} |w|$, i.e., the maximum length of the string u over all the rule $A \to u$ in R.

Intuitively, this means that in every derivation tree of $\mathcal{G}$, every node has at most m children.

We define $N = m^n + 1$.

Proof of pumping lemma

Let $\mathcal{G} = \langle \Sigma, V, R, S \rangle$ be a CFG and let $n = |V|$.

Let $m = \max_{A \to w \in R} |w|$, i.e., the maximum length of the string u over all the rule $A \to u$ in R.

Intuitively, this means that in every derivation tree of $\mathcal{G}$, every node has at most m children.

We define $N = m^n + 1$.

Intuitively, this means that for a word of length $\geqslant N$, its derivation tree will have depth $\geqslant n + 1$.

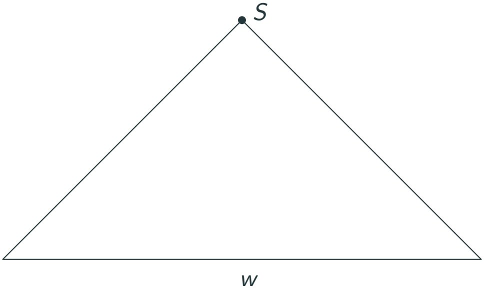
Proof of pumping lemma (Continued)

Let $w \in L(\mathcal{G})$ and $|w| \geqslant N$. Recall that $N = m^n + 1$.

Proof of pumping lemma (Continued)

Let $w \in L(\mathcal{G})$ and $|w| \geqslant N$. Recall that $N = m^n + 1$.

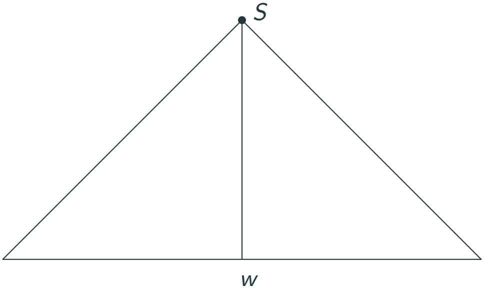
Consider its derivation tree T and its depth $\geqslant n + 1$.



Proof of pumping lemma (Continued)

Let $w \in L(\mathcal{G})$ and $|w| \geqslant N$. Recall that $N = m^n + 1$.

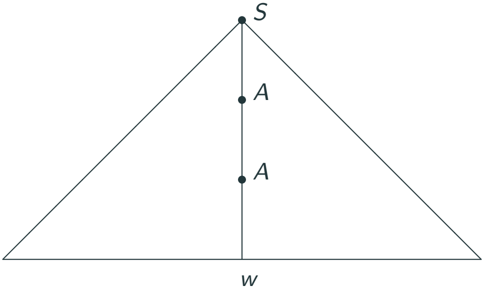
Consider its derivation tree T and its depth $\geqslant n + 1$.



Proof of pumping lemma (Continued)

Let $w \in L(\mathcal{G})$ and $|w| \geqslant N$. Recall that $N = m^n + 1$.

Consider its derivation tree T and its depth $\geqslant n + 1$.

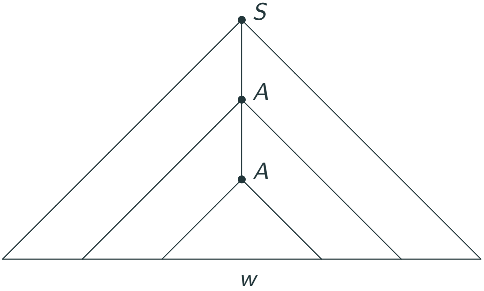


There exists a variable A that appears at least twice in the same path.

Proof of pumping lemma (Continued)

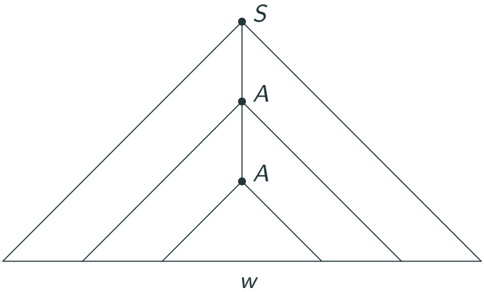
Let $w \in L(\mathcal{G})$ and $|w| \geqslant N$. Recall that $N = m^n + 1$.

Consider its derivation tree T and its depth $\geqslant n + 1$.

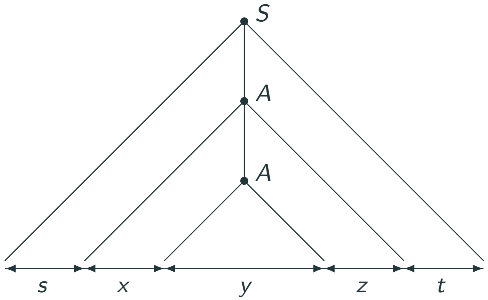


Pick the variable A such that in its subtree there is no variable that appears twice.

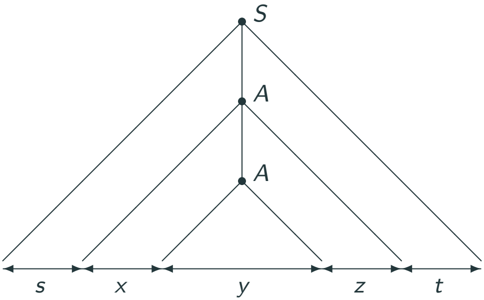
Proof of pumping lemma (Continued)



Proof of pumping lemma (Continued)



Proof of pumping lemma (Continued)



The three conditions hold.

- $|x| + |z| \geqslant 1$.
- $|xyz| \leqslant N$.
- For every $i \geqslant 0$, $sx^i yz^i t \in L(\mathcal{G})$. $\Rightarrow$ By "pumping" variable A.

Pumping lemma

Lemma 3.6 (pumping lemma)
Let $\mathcal{G} = \langle \Sigma, V, R, S \rangle$ be a CFG. Then, there is an integer N such that every $w \in L(\mathcal{G})$ with length $\geqslant N$ can be partitioned into:

$$w \quad = \quad s\ x\ y\ z\ t$$

such that the following holds.

- $|x| + |z| \geqslant 1$.
- $|xyz| \leqslant N$.
- *For every $i \geqslant 0$, $sx^i yz^i t \in L(\mathcal{G})$.*

The length N is usually called the pumping length of $\mathcal{G}$.

An example of a non CFL language

Similar to regular language, we can use pumping lemma to show that a language is not CFL.

An example of a non CFL language

Similar to regular language, we can use pumping lemma to show that a language is not CFL.

$$L \quad := \quad \{a^n b^n c^n \mid n \geqslant 0\}$$

An example of a non CFL language

Similar to regular language, we can use pumping lemma to show that a language is not CFL.

$$L \quad := \quad \{a^n b^n c^n \mid n \geqslant 0\}$$

Claim 1

The language L is not CFL.

Proof that $L = \{a^n b^n c^n \mid n \geqslant 0\}$ is not CFL

Proof that $L = \{a^n b^n c^n \mid n \geqslant 0\}$ is not CFL

If L is CFL, let N be its pumping length.

Proof that $L = \{a^n b^n c^n \mid n \geqslant 0\}$ is not CFL

If L is CFL, let N be its pumping length.

Consider the following string, where n is $\geqslant N$.

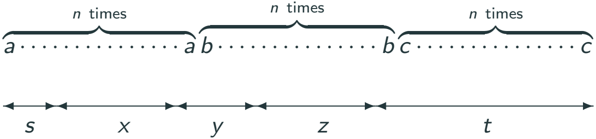
$$\overbrace{a \cdots\cdots\cdots\cdots a}^{n \text{ times}}\overbrace{b \cdots\cdots\cdots\cdots b}^{n \text{ times}}\overbrace{c \cdots\cdots\cdots\cdots c}^{n \text{ times}}$$

By pumping lemma, we can partition it into $sxyzt$ such that $sx^i yz^i t \in L$, for every $i \geqslant 0$.

Proof that $L = \{a^n b^n c^n \mid n \geqslant 0\}$ is not CFL

If L is CFL, let N be its pumping length.

Consider the following string, where n is $\geqslant N$.



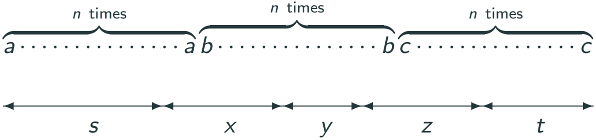
Pumping x and z will increase the number of a and b, but not c

By pumping lemma, we can partition it into $sxyzt$ such that $sx^i yz^i t \in L$, for every $i \geqslant 0$.

Proof that $L = \{a^n b^n c^n \mid n \geqslant 0\}$ is not CFL

If L is CFL, let N be its pumping length.

Consider the following string, where n is $\geqslant N$.



Pumping x and z will make some a appear after b

By pumping lemma, we can partition it into $sxyzt$ such that $sx^i yz^i t \in L$, for every $i \geqslant 0$.

Proof that $L = \{a^n b^n c^n \mid n \geqslant 0\}$ is not CFL

If L is CFL, let N be its pumping length.

Consider the following string, where n is $\geqslant N$.

$$\overbrace{a \cdots\cdots\cdots\cdots a}^{n \text{ times}}\overbrace{b \cdots\cdots\cdots\cdots b}^{n \text{ times}}\overbrace{c \cdots\cdots\cdots\cdots c}^{n \text{ times}}$$

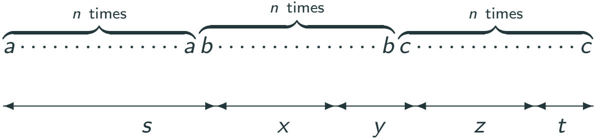
Pumping x and z will increase the number of b and c, but not a

By pumping lemma, we can partition it into $sxyzt$ such that $sx^i yz^i t \in L$, for every $i \geqslant 0$.

Proof that $L = \{a^n b^n c^n \mid n \geqslant 0\}$ is not CFL

If L is CFL, let N be its pumping length.

Consider the following string, where n is $\geqslant N$.



Pumping x and z will increase the number of b and c, but not a

By pumping lemma, we can partition it into $sxyzt$ such that $sx^i yz^i t \in L$, for every $i \geqslant 0$.

For any other partition, pumping x and z will result in a word not in L. Thus, contradicting pumping lemma.

CFL are not closed under intersection and complement

CFL are not closed under intersection and complement

Consider the following languages:

$$\begin{aligned} L_1 &:= \{a^n b^n c^k \mid n, k \geqslant 0\} \\ L_2 &:= \{a^k b^n c^n \mid n, k \geqslant 0\} \end{aligned}$$

Both L_1 and L_2 are CFL.

CFL are not closed under intersection and complement

Consider the following languages:

$$\begin{aligned} L_1 &:= \{a^n b^n c^k \mid n, k \geqslant 0\} \\ L_2 &:= \{a^k b^n c^n \mid n, k \geqslant 0\} \end{aligned}$$

Both L_1 and L_2 are CFL.

However, $L_1 \cap L_2 = \{a^n b^n c^n \mid n \geqslant 0\}$ which is not CFL.

CFL are not closed under intersection and complement

Consider the following languages:

$$\begin{aligned} L_1 &:= \{a^n b^n c^k \mid n, k \geqslant 0\} \\ L_2 &:= \{a^k b^n c^n \mid n, k \geqslant 0\} \end{aligned}$$

Both L_1 and L_2 are CFL.

However, $L_1 \cap L_2 = \{a^n b^n c^n \mid n \geqslant 0\}$ which is not CFL.

Note also that $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ (de Morgan's law). Thus, CFL are not closed under complement.

CFL are not closed under intersection and complement

Consider the following languages:

$$
\begin{aligned}
L_1 &:= \{a^n b^n c^k \mid n, k \geqslant 0\} \\
L_2 &:= \{a^k b^n c^n \mid n, k \geqslant 0\}
\end{aligned}
$$

Both L_1 and L_2 are CFL.

However, $L_1 \cap L_2 = \{a^n b^n c^n \mid n \geqslant 0\}$ which is not CFL.

Note also that $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ (de Morgan's law). Thus, CFL are not closed under complement.

Note: For a language L over alphabet Σ, $\overline{L} = \Sigma^* - L$, i.e., the complement of the language L.

End of Lesson 3