Lecture 9: Context Free Languages and Grammars

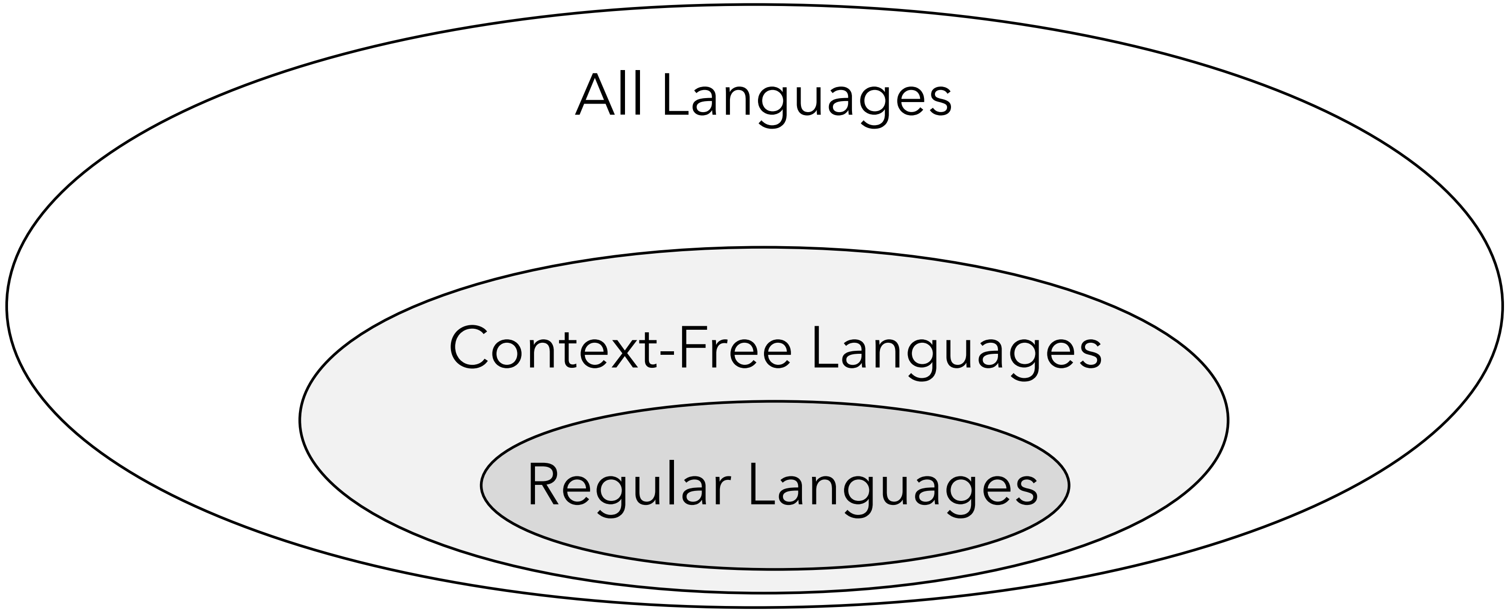
CSC 320: Foundations of Computer Science

Quinton Yong

quintonyong@uvic.ca

University of Victoria

Context-Free Languages



- The set of **regular languages** is the set of all languages recognized by DFAs, NFAs, and regular expressions

- The set of **context-free languages** is a **superset** of the regular languages

Grammars

Consider the following "more powerful" way to represent a language:

$$\boldsymbol{A} \rightarrow \mathbf{0}\boldsymbol{A}\mathbf{1}$$

$$\boldsymbol{A} \rightarrow \boldsymbol{B}$$

$$\boldsymbol{B} \rightarrow \#$$

This **grammar** $\boldsymbol{G}$ describes a language, which contains the strings which can be **derived** by the grammar.

We can derive a string from a grammar as follows:

$$\boldsymbol{A} \Rightarrow \mathbf{0}\boldsymbol{A}\mathbf{1} \Rightarrow \mathbf{00}\boldsymbol{A}\mathbf{11} \Rightarrow \mathbf{000}\boldsymbol{A}\mathbf{111} \Rightarrow \mathbf{000}\boldsymbol{B}\mathbf{111} \Rightarrow \mathbf{000\#111}$$

Context-Free Grammars (CFG)

This is an example of a **context-free grammar** (CFG) $\boldsymbol{G}$.

Start variable $\boldsymbol{A}$ $\boxed{\boldsymbol{A} \rightarrow \mathbf{0}\boldsymbol{A}\mathbf{1}}$ substitution / production rule

$\boxed{\boldsymbol{A}} \rightarrow \boxed{\boldsymbol{B}}$ variables $\{\boldsymbol{A}, \boldsymbol{B}\}$

$\boldsymbol{B} \rightarrow \boxed{\#}$ terminals $\{\mathbf{0}, \mathbf{1}, \#\}$

A CFG consists of the following:

- A collection of **substitution rules** (also called **production rules**)
  - <variable> → <variable and terminals>
- **Variable** symbols (use capital letters)
- **Terminal** symbols (alphabet symbols)
- One **start variable** (left-hand side of topmost substitution rule)

Context-Free Grammars (CFG)

$$A \to 0A1$$

$$A \to B$$

$$B \to \#$$

We **derive** strings using a CFG G as follows:

1. Write down the **start variable**

   $$A$$

2. **Replace a variable** with the right side of a rule starting with that variable

   $$0A1$$

3. **Repeat step 2** until no variables remain

   $$0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$$

Parse Tree

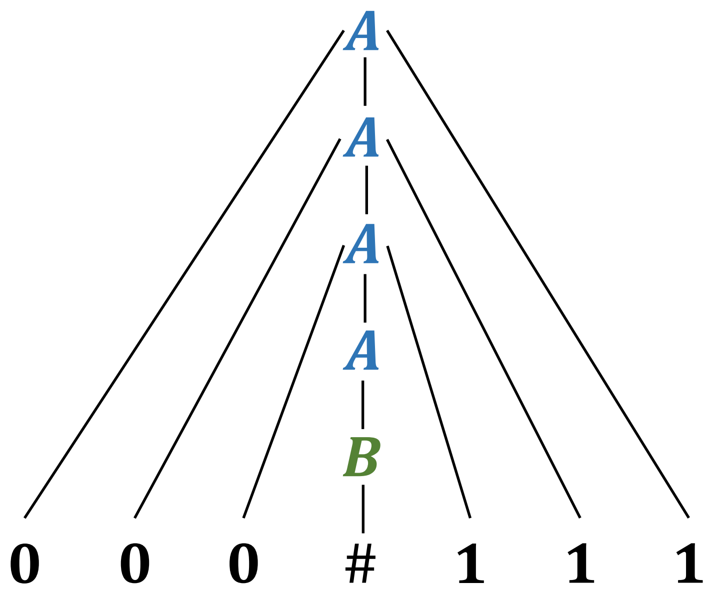
$$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$$

We can represent the **derivation** of a string pictorially using a **parse tree**

$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$



Context-Free Grammars (CFG)

$$A \rightarrow 0A1$$

$$A \rightarrow B$$

$$B \rightarrow \#$$

As a shorthand, we can write **multiple substitution rules** with the same left-hand variable as a **single rule separated by** |

$$A \rightarrow 0A1 \mid B$$

$$B \rightarrow \#$$

Context-Free Grammars (CFG)

- **All strings** that can be generate by a **CFG** $\boldsymbol{G}$ is the **language of** $\boldsymbol{G}$
- We write $\boldsymbol{L}(\boldsymbol{G})$ for the language of grammar $\boldsymbol{G}$

Context-Free Grammars (CFG)

- **All strings** that can be generate by a **CFG** $\boldsymbol{G}$ is the **language of** $\boldsymbol{G}$
- We write $\boldsymbol{L}(\boldsymbol{G})$ for the language of grammar $\boldsymbol{G}$

- What is the language of the following grammar $\boldsymbol{G}$?

$$\boldsymbol{A} \rightarrow \mathbf{0}\boldsymbol{A}\mathbf{1} \mid \boldsymbol{B}$$

$$\boldsymbol{B} \rightarrow \#$$

- $\boldsymbol{L}(\boldsymbol{G}) = \{\, \mathbf{0}^{\boldsymbol{n}}\#\mathbf{1}^{\boldsymbol{n}} \mid \boldsymbol{n} \geq \mathbf{0}\}$

Formal Definition: Context-Free Grammar

A **context-free grammar** is a 4-tuple $(\boldsymbol{V}, \boldsymbol{\Sigma}, \boldsymbol{R}, \boldsymbol{S})$ where

- $\boldsymbol{V}$ is a finite set called the **variables**
- $\boldsymbol{\Sigma}$ is a finite set called the **terminals** (disjoint from $\boldsymbol{V}$)
- $\boldsymbol{R}$ is a finite set of (substitution / production) **rules**
  - Each rule is a variable to be substituted by a string of variables and terminals
- $\boldsymbol{S} \in \boldsymbol{V}$ is the **start variable**

- **Note**: the right-hand side of a rule may be $\boldsymbol{\varepsilon}$, but we don't add $\boldsymbol{\varepsilon}$ to $\boldsymbol{\Sigma}$

Terminology

Given a grammar $G = (V, \Sigma, R, S)$

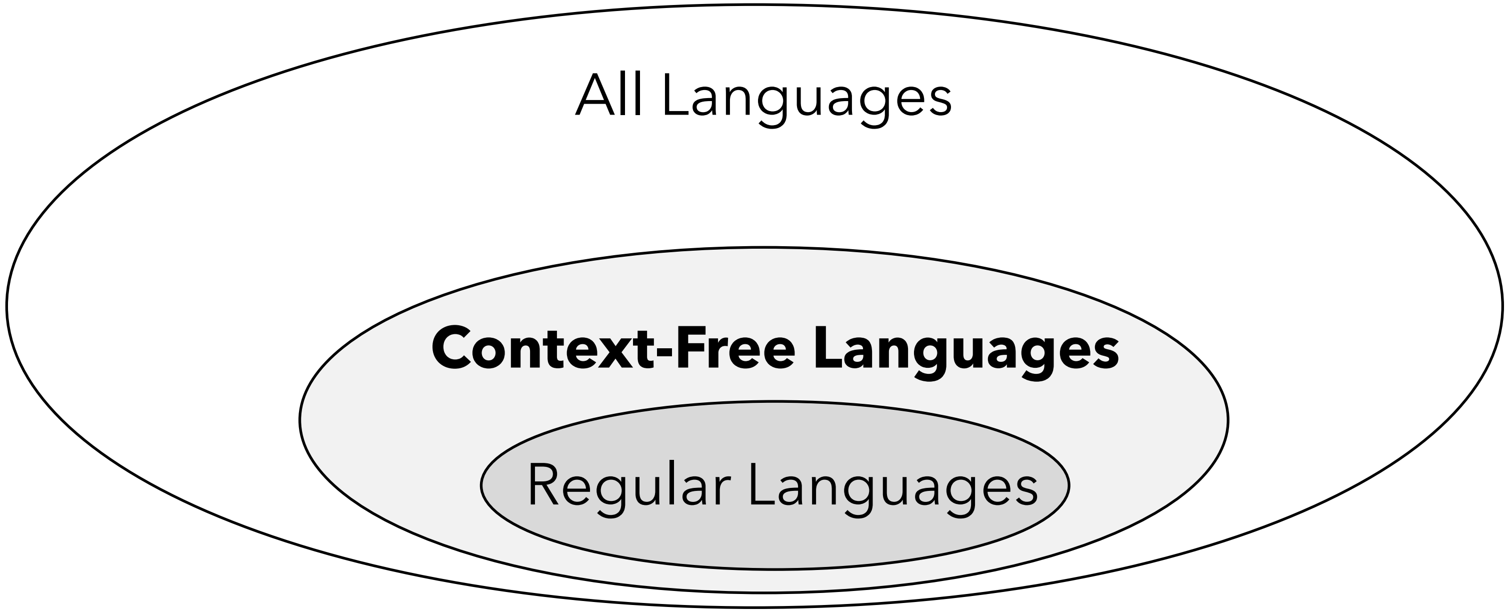
- Let u, v, and w be strings of variable and terminals
- Let $A \rightarrow w$ be a rule of G

- We say that uAv **yields** uwv, denoted $uAv \Rightarrow uwv$ (i.e. substitute one variable)

- We say that u **derives** v, denoted $u \overset{*}{\Rightarrow} v$, if a sequence $u_1, u_2, \dots, u_k$ exists for some $k \geq 0$ and $u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \cdots \Rightarrow u_k \Rightarrow v$

- The **language of grammar** G is $L(G) = \{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\}$

Context-Free Languages



The class of **context-free languages** is the class of languages recognized by **context-free grammars**

Example 1

Produce a grammar for the language $\{ \mathbf{0^n 1^n} \mid \boldsymbol{n} \geq \mathbf{0} \}$:

$$\boldsymbol{S} \rightarrow \mathbf{0}\boldsymbol{S}\mathbf{1} \mid \boldsymbol{\varepsilon}$$

Formally, $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{\Sigma}, \boldsymbol{R}, \boldsymbol{S})$ with

- $\boldsymbol{V} = \{\boldsymbol{S}\}$
- $\boldsymbol{\Sigma} = \{\mathbf{0}, \mathbf{1}\}$
- $\boldsymbol{R}$ is the set of rules

  $$\boldsymbol{S} \rightarrow \mathbf{0}\boldsymbol{S}\mathbf{1} \mid \boldsymbol{\varepsilon}$$

Example 2

Consider grammar $\boldsymbol{G} = (\{\boldsymbol{S}\}, \{\boldsymbol{a}, \boldsymbol{b}\}, \boldsymbol{R}, \boldsymbol{S})$ where R is

$$\boldsymbol{S} \rightarrow \boldsymbol{aSb} \mid \boldsymbol{SS} \mid \boldsymbol{\varepsilon}$$

What is $\boldsymbol{L}(\boldsymbol{G})$?

- Example strings in the language: $\boldsymbol{abab}$, $\boldsymbol{aaabbb}$, $\boldsymbol{aababb}$

- May be easier to see if we set $\boldsymbol{a} \coloneqq$ **(** and $\boldsymbol{b} \coloneqq$ **)**
- Example strings in the language: **()()**, **((()))**, **(()())**

$\boldsymbol{L}(\boldsymbol{G})$ is the language of all strings of properly nested parentheses

Example 3

Consider grammar $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{\Sigma}, \boldsymbol{R}, \boldsymbol{E})$ with $\boldsymbol{V} = \{\boldsymbol{E}, \boldsymbol{F}, \boldsymbol{T}\}$, $\boldsymbol{\Sigma} = \{\boldsymbol{a},\ +,\ \cdot\ ,\ (\ ,\ )\ \}$, and $\boldsymbol{R}$ is given by:

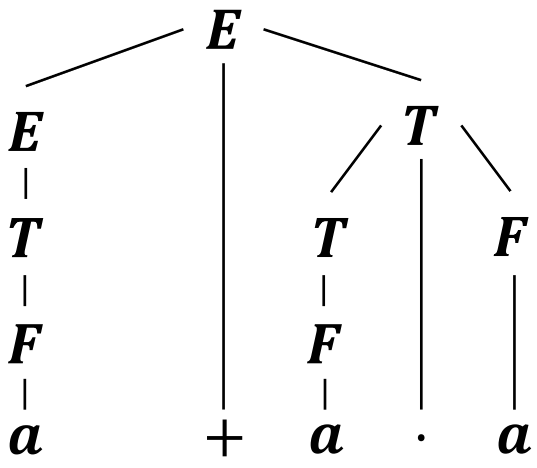
$$\boldsymbol{E} \rightarrow \boldsymbol{E} + \boldsymbol{T} \mid \boldsymbol{T}$$

$$\boldsymbol{T} \rightarrow \boldsymbol{T} \cdot \boldsymbol{F} \mid \boldsymbol{F}$$

$$\boldsymbol{F} \rightarrow (\boldsymbol{E}) \mid \boldsymbol{a}$$

Show how we can derive the string $\boldsymbol{a} + \boldsymbol{a} \cdot \boldsymbol{a}$ using a **parse tree**

Parse trees for this grammar reflect order of operations



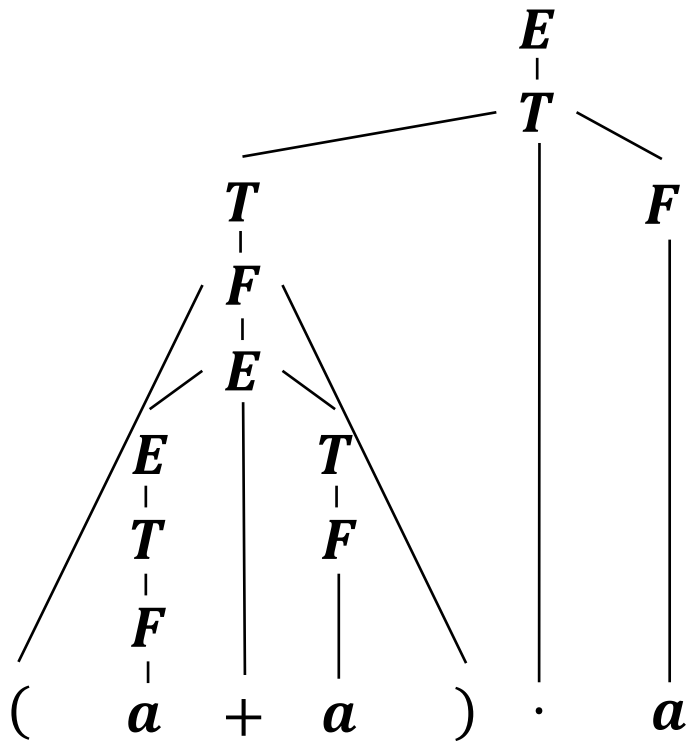
Example 3b

$$E \to E + T \mid T$$

$$T \to T \cdot F \mid F$$

$$F \to (E) \mid a$$

Show how we can derive the string $(a + a) \cdot a$ using a **parse tree**



Example CFG for Programming Languages

$$\mathit{Func} \rightarrow \mathit{Id} \; (\; \mathit{PrmtrList} \;) \; \mathit{Stmt}$$

$$\mathit{PrmtrList} \rightarrow \boldsymbol{\varepsilon} \mid \; \ldots$$

$$\mathit{Stmt} \rightarrow \{ \; \mathit{StmtList} \; \} \mid \mathit{Id} = \mathit{Expr}; \mid \textbf{if} \; (\; \mathit{Expr} \;) \; \mathit{Stmt} \mid \; \ldots$$

$$\mathit{StmntList} \rightarrow \mathit{Stmnt} \mid \mathit{StmntList} \;\; \mathit{Stmt}$$

$$\mathit{Expr} \rightarrow \mathit{Id} \mid \mathit{Num} \mid \mathit{Expr} \; \mathit{Optr} \; \mathit{Expr}$$

$$\mathit{Id} \rightarrow \mathbf{a} \mid \mathbf{b} \mid \mathbf{c} \mid \; \ldots$$

$$\mathit{Num} \rightarrow \mathbf{0} \mid \mathbf{1} \mid \mathbf{2} \mid \; \ldots$$

$$\mathit{Optr} \rightarrow + \mid - \mid > \mid \; \ldots$$

Exercise

Produce a context-free grammar for the language

$$\{\, \boldsymbol{w} \in \{\mathbf{0}, \mathbf{1}\}^* \mid \boldsymbol{w} \text{ has an equal number of } \mathbf{0}\text{s and } \mathbf{1}\text{s} \,\}$$

Leftmost Derivation

- When deriving a string by substituting variables, there's **normally no rule** on which variable to substitute first
- However, it is useful to **add some structure** to substituting variables

Leftmost derivation: We call a derivation in grammar $\boldsymbol{G}$ a **leftmost derivation** if at every step, the **leftmost remaining variable** is replaced

$$\boldsymbol{E} \rightarrow \boldsymbol{E} + \boldsymbol{T} \mid \boldsymbol{T}$$
$$\boldsymbol{T} \rightarrow \boldsymbol{T} \cdot \boldsymbol{F} \mid \boldsymbol{F}$$
$$\boldsymbol{F} \rightarrow (\boldsymbol{E}) \mid \boldsymbol{a}$$

leftmost derivation of $\boldsymbol{a} + \boldsymbol{a}$

$$\boldsymbol{E} \Rightarrow \boldsymbol{E} + \boldsymbol{T} \Rightarrow \boldsymbol{T} + \boldsymbol{T} \Rightarrow \boldsymbol{F} + \boldsymbol{T} \Rightarrow \boldsymbol{a} + \boldsymbol{T} \Rightarrow \mathbf{a} + \boldsymbol{F} \Rightarrow \mathbf{a} + \mathbf{a}$$

Ambiguous Grammars

- A string is derived **ambiguously** in a CFG if it has at **least two different leftmost derivations**
  - i.e. can derive the string in multiple ways even if always substituting the leftmost variable

- A context-free grammar is **ambiguous** if there **exists a string** that can be derived ambiguously

Ambiguous Grammars Example

Consider grammar $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{\Sigma}, \boldsymbol{R}, \boldsymbol{E})$ with $\boldsymbol{V} = \{\boldsymbol{E}, \boldsymbol{F}, \boldsymbol{T}\}$, $\boldsymbol{\Sigma} = \{\boldsymbol{a},\ +,\ \cdot\,,\ (\,,\ )\,\}$, and $\boldsymbol{R}$ is given by:

$$\boldsymbol{E} \to \boldsymbol{E} + \boldsymbol{E} \mid \boldsymbol{E} \cdot \boldsymbol{E} \mid (\boldsymbol{E}) \mid \boldsymbol{a}$$

Is this grammar ambiguous? **Yes**

Consider the string $\boldsymbol{a} \cdot \boldsymbol{a} + \boldsymbol{a}$

- Leftmost derivation 1:

$$\boldsymbol{E} \Rightarrow \boldsymbol{E} \cdot \boldsymbol{E} \Rightarrow \boldsymbol{a} \cdot \boldsymbol{E} \Rightarrow \boldsymbol{a} \cdot \boldsymbol{E} + \boldsymbol{E} \Rightarrow \boldsymbol{a} \cdot \boldsymbol{a} + \boldsymbol{E} \Rightarrow \boldsymbol{a} \cdot \boldsymbol{a} + \boldsymbol{a}$$

- Leftmost derivation 2:

$$\boldsymbol{E} \Rightarrow \boldsymbol{E} + \boldsymbol{E} \Rightarrow \boldsymbol{E} \cdot \boldsymbol{E} + \boldsymbol{E} \Rightarrow \boldsymbol{a} \cdot \boldsymbol{E} + \boldsymbol{E} \Rightarrow \boldsymbol{a} \cdot \boldsymbol{a} + \boldsymbol{E} \Rightarrow \boldsymbol{a} \cdot \boldsymbol{a} + \boldsymbol{a}$$