Context-Free Languages

Lecture 9

November 6, 2022

Objectives

By the end of this lecture, you should be able to

- Formally define a context-free grammar.
- Identify the language generated by a context-free grammar.
- Construct derivations of strings from context-free grammars.
- Design context-free grammars.

Context-Free Languages

- Many interesting non-regular languages belong to the class of **context-free languages** (CFLs).
- **Example**: The following are CFLs over $\Sigma = \{\mathtt{0}, \mathtt{1}\}$
  - $L_1 = \{\mathtt{0}^n\mathtt{1}^n |\ n \in \mathcal{N}\}$.
  - $L_2 = \{w|\ w$ has an equal number of 0s and 1s $\}$.
  - $L_3 = \{ww^{\mathcal{R}}|\ w \in \Sigma^*\}$
  - $L_4 = \{w|\ w$ is a palindrome over $\Sigma\}$.
- CFLs are necessarily recursive in structure.

Non Context-Free Languages

- Not all languages are context-free, though.
- **Example**: The following languages are not context-free.
  - $L_1 = \{\mathtt{0}^n\mathtt{1}^n\mathtt{2}^n \mid n \in \mathcal{N}\}$.
  - $L_2 = \{ww \mid w \in \Sigma^*\}$
  - $L_3 = \{\mathtt{0}^n\mathtt{1}^m \mid n, m \in \mathcal{N} \wedge m = n^2\}$.
  - $L_4 =$ the full-fledged English language.

Context-Free Grammars

- A **context-free grammar** (CFG) is a formal device used to describe a context-free language.
- CFGs were first introduced by Noam Chomsky (MIT Professor of linguistics and political activist).
- Chomsky's motivation was primarily (psycho)linguistic
  - He wanted to provide a formal account of the regularity in structure of natural language sentences.
- But CFGs have made their way into computer science where they are primarily used to describe the syntax of programming languages (and hence to design parsers).

CFGs: An Example

$$\begin{aligned} A &\longrightarrow \mathtt{0}A\mathtt{1} \\ A &\longrightarrow B \\ B &\longrightarrow \# \end{aligned}$$

CFGs: An Example

$$
\begin{aligned}
A &\longrightarrow \mathtt{0}A\mathtt{1} \\
A &\longrightarrow B \\
B &\longrightarrow \#
\end{aligned}
$$

A CFG consists of:

CFGs: An Example

$$
\begin{aligned}
A &\longrightarrow \mathtt{0}A\mathtt{1} \\
A &\longrightarrow B \\
B &\longrightarrow \#
\end{aligned}
$$

A CFG consists of:

1. Substitution rules (or productions).

CFGs: An Example

$$
\begin{array}{lcl}
A & \longrightarrow & \mathtt{0}A\mathtt{1} \\
A & \longrightarrow & B \\
B & \longrightarrow & \#
\end{array}
$$

A CFG consists of:

1. Substitution rules (or productions).
2. Variable symbols.

CFGs: An Example

$$A \longrightarrow \mathtt{0}A\mathtt{1}$$
$$A \longrightarrow B$$
$$B \longrightarrow \#$$

A CFG consists of:

1. Substitution rules (or productions).
2. Variable symbols.
   - One and only one on the left-hand side of each rule.
   - Zero or more on the right-hand side of each rule.

CFGs: An Example

$$
\begin{aligned}
A &\longrightarrow \mathtt{0}A\mathtt{1} \\
A &\longrightarrow B \\
B &\longrightarrow \#
\end{aligned}
$$

A CFG consists of:

1. Substitution rules (or productions).
2. Variable symbols.
   - One and only one on the left-hand side of each rule.
   - Zero or more on the right-hand side of each rule.
3. Terminal symbols. (These are the symbols that do *not* appear on the left-hand side of any rule.)

CFGs: An Example

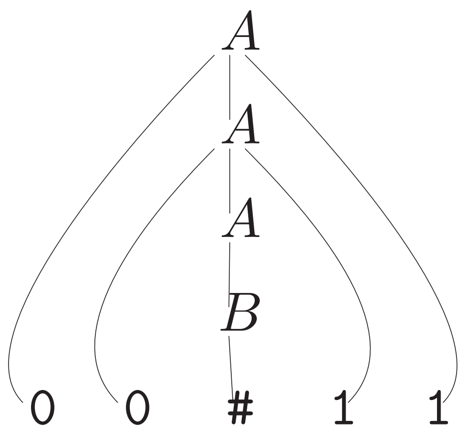
$$
\begin{array}{lcl}
A & \longrightarrow & \mathtt{0}A\mathtt{1} \\
A & \longrightarrow & B \\
B & \longrightarrow & \#
\end{array}
$$

A CFG consists of:

1. Substitution rules (or productions).
2. Variable symbols.
   - One and only one on the left-hand side of each rule.
   - Zero or more on the right-hand side of each rule.
3. Terminal symbols. (These are the symbols that do *not* appear on the left-hand side of any rule.)
4. A distinguished start variable.

Derivations

- A CFG **generates** a string by a sequence of substitutions, called a **derivation**.
- **Example**: The above CFG generates the string 00#11 by the following derivation
  - $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 00B11 \Rightarrow 00\#11$
- Derivations are often represented by **parse trees**.



Another Example: A Fragment of English

$$
\begin{array}{rcl}
\langle \text{SENTENCE} \rangle & \rightarrow & \langle \text{NOUN-PHRASE} \rangle \langle \text{VERB-PHRASE} \rangle \\
\langle \text{NOUN-PHRASE} \rangle & \rightarrow & \langle \text{CMPLX-NOUN} \rangle \mid \langle \text{CMPLX-NOUN} \rangle \langle \text{PREP-PHRASE} \rangle \\
\langle \text{VERB-PHRASE} \rangle & \rightarrow & \langle \text{CMPLX-VERB} \rangle \mid \langle \text{CMPLX-VERB} \rangle \langle \text{PREP-PHRASE} \rangle \\
\langle \text{PREP-PHRASE} \rangle & \rightarrow & \langle \text{PREP} \rangle \langle \text{CMPLX-NOUN} \rangle \\
\langle \text{CMPLX-NOUN} \rangle & \rightarrow & \langle \text{ARTICLE} \rangle \langle \text{NOUN} \rangle \\
\langle \text{CMPLX-VERB} \rangle & \rightarrow & \langle \text{VERB} \rangle \mid \langle \text{VERB} \rangle \langle \text{NOUN-PHRASE} \rangle \\
\langle \text{ARTICLE} \rangle & \rightarrow & \texttt{a} \mid \texttt{the} \\
\langle \text{NOUN} \rangle & \rightarrow & \texttt{boy} \mid \texttt{girl} \mid \texttt{flower} \\
\langle \text{VERB} \rangle & \rightarrow & \texttt{touches} \mid \texttt{likes} \mid \texttt{sees} \\
\langle \text{PREP} \rangle & \rightarrow & \texttt{with}
\end{array}
$$

Formal Definition of a CFG

- A context-free grammar is a 4-tuple (V, Σ, R, S), where
  1. V is a non-empty finite set of variables,
  2. Σ, is an alphabet, disjoint from V, whose symbols are called terminals,
  3. $R \subseteq V \times (V \cup \Sigma)^*$ is a non-empty finite set of rules, and
  4. $S \in V$ is the start variable.

The Language of a CFG

Let $G = (V, \Sigma, R, S)$ be a CFG. Let u, v, and $w \in (V \cup \Sigma)^*$

- If $(A \longrightarrow w) \in R$, then uAv **yields** uwv, written $uAv \Rightarrow uwv$.
- u **derives** v, written $u \overset{*}{\Rightarrow} v$ if
  1. $u = v$, or
  2. there is a sequence $u_1, u_2, \ldots, u_k$ for $k > 0$ such that

$$u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \ldots u_k \Rightarrow v$$

- The language of G is the set

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

Designing CFGs

- Designing CFGs is very similar to structured programming.
- You do not have the full power of a programming language.
- You only have sequencing and function calls.
  - No conditionals, but random guesses.
  - No iteration, but recursion.
- The start symbol corresponds to the main function.
- Every other variable corresponds to a sub-routine.
- Terminals correspond to primitive operations. Think of these as printing operations.
- Concatenation corresponds to sequencing.
- Your task is to write a program that would print all and only strings from the target language.

Example 1

Give a CFG that generates the language $L_1 = \{\mathtt{0}^n\mathtt{1}^n |\ n \geq 0\}$.
$(\Sigma = \{\mathtt{0}, \mathtt{1}\})$

Example 1

Give a CFG that generates the language $L_1 = \{\mathtt{0}^n\mathtt{1}^n |\ n \geq 0\}$.
$(\Sigma = \{\mathtt{0}, \mathtt{1}\})$

$$S \longrightarrow \mathtt{0}S\mathtt{1} \mid \varepsilon$$

Example 2

Give a CFG that generates the language $L_2 = \{ww^{\mathcal{R}} |\ w \in \Sigma^*\}$.
$(\Sigma = \{\mathtt{0}, \mathtt{1}\})$

Example 2

Give a CFG that generates the language $L_2 = \{ww^{\mathcal{R}} |\ w \in \Sigma^*\}$.
$(\Sigma = \{\mathtt{0}, \mathtt{1}\})$

$$S \longrightarrow \mathtt{0}S\mathtt{0} \mid \mathtt{1}S\mathtt{1} \mid \varepsilon$$

Example 3

Describe the language generated by the following CFG.

$$S \longrightarrow \mathtt{0}S\mathtt{0} \mid \mathtt{1}S\mathtt{1} \mid \mathtt{0} \mid \mathtt{1} \mid \varepsilon$$

Example 3

Describe the language generated by the following CFG.

$$S \longrightarrow \mathtt{0}S\mathtt{0} \mid \mathtt{1}S\mathtt{1} \mid \mathtt{0} \mid \mathtt{1} \mid \varepsilon$$

$L_3 = \{w \mid w \text{ is a palindrome over } \{\mathtt{0}, \mathtt{1}\}\}$.

Example 4

Give a CFG that generates the language $L_4 = \{w|\ w$ has an equal number of $\texttt{0}$s and $\texttt{1}$s $\}$. ($\Sigma = \{\texttt{0}, \texttt{1}\}$)

Example 4

Give a CFG that generates the language $L_4 = \{w|\ w$ has an equal number of $\mathtt{0}$s and $\mathtt{1}$s $\}$. $(\Sigma = \{\mathtt{0}, \mathtt{1}\})$

$$\begin{array}{lcl} S & \longrightarrow & \mathtt{1}A \mid \mathtt{0}B \mid \varepsilon \\ A & \longrightarrow & \mathtt{0} \mid \mathtt{0}S \mid \mathtt{1}AA \\ B & \longrightarrow & \mathtt{1} \mid \mathtt{1}S \mid \mathtt{0}BB \end{array}$$

Next time

- Ambiguity.
- Chomsky Normal Form.

Points to take home

- Formal definition of a CFG.
- Derivations.
- Parse trees.
- Language of a CFG.