CS340: Theory of Computation Sem I 2019-20

Lecture Notes 8: Context-Free Grammar

Raghunath Tewari IIT Kanpur

So far we have studied the class of languages known as regular languages. They can be expressed using a finite automata or regular expressions (both are equivalent). Also we have shown the existence of non-regular languages.

1 Context-Free Grammar

- Aim is to recognize more languages.
- Context-free grammars are a model to recognize languages.
- Recognizes all regular languages (needs proof).
- Recognizes some non-regular languages.
- Has applications in programming languages and compiler design.

A context-free grammar consists of the following components:

- *Terminals*: Symbols of the alphabet.
- *Variables*: A second type of symbols that can be replaced. Usually represented with capital letters. Generates a set of strings.
- *Substitution rules* (or *productions* or simply *rules*): Rules that dictate how a variable gets replaced. The left hand side has the variable that is to be replaced and the right hand side has replacement "content".
- *Start variable*: A special variable from which the process starts.

1.1 Process to generate strings from a given grammar

1. Write the start variable.
2. Pick a variable in the current string and replace it with one of its substitution rules.
3. Continue step 2 until no more variable remains in the current string (has only terminals).

Let us look at an example. Consider the following context-free grammar (in short CFG).

$$\begin{aligned} S &\longrightarrow 0S1 \\ S &\longrightarrow \epsilon \end{aligned}$$

The grammar has two terminals, 0 and 1. It has only one variable S, hence it is also the start variable. There are two productions rules for S as seen above.
Now let us see how a string gets generated.

$$S \longrightarrow 0S1 \longrightarrow 00S11 \longrightarrow 000S111 \longrightarrow 000111$$

In the first three steps we replace S by its first substitution rule, whereas in the last step we replace S by its second substitution rule. Since the last "string" has only terminals, the process stops. This generates the string $0^3 1^3$.

What other strings can this grammar generate?

All strings of the form $0^n 1^n$ for $n \geq 0$.

Can it generate any strings not of the form $0^n 1^n$?

No.

Therefore the language of the grammar is

$$L = \{0^n 1^n \mid n \geq 0\}.$$

We known that L is a non-regular language. Hence CFGs accept languages that are not regular.

Note 1. A *derivation* of a string is a sequence of substitutions, starting from the start variable that produces the string. The *language of a grammar* is the set of all strings that can be derived from the start symbol.

Another example:

$$\begin{aligned} S &\longrightarrow ASB \mid \epsilon \\ A &\longrightarrow a \\ B &\longrightarrow bb \end{aligned}$$

The language generated by the above grammar is

$$L = \{a^n b^{2n} \mid n \geq 0\}$$

which is also not regular.

Exercise 1. What happens if we add the production rule

$$B \longrightarrow \epsilon$$

to the above grammar?

1.2 Formal definition

Definition 1.1. A *context-free grammar* is the 4-tuple (V, Σ, P, S), where

- V is a finite set called variables,
- Σ is a finite set disjoint from V called terminals,
- $P \subseteq V \times \{V \cup \Sigma\}^*$ is a finite set of production rules, and
- $S \in V$ is the start variable.

Note that in the above definition, P is a relation since the same variable can have several production rules.
We next define some more notations and terminology related to CFGs. Let $G = (V, \Sigma, P, S)$ be a CFG.

- If $A \in V$, $u, v, w \in \{V \cup \Sigma\}^*$ and $A \longrightarrow w$ is a rule, then we say that uAv *yields* uwv in one step. This is denoted as $uAv \Rightarrow uwv$.
- For $u, v \in (V \cup \Sigma)^*$, we say that u yields v, denoted as $u \stackrel{*}{\Rightarrow} v$, if
  - $u = v$, or
  - there exists strings $u_0, u_1, \ldots, u_k$ (for $k \geq 1$) in $\{V \cup \Sigma\}^*$, such that $u_0 = u$, $u_k = v$ and $u_{i-1} \Rightarrow u_i$ for all $1 \leq i \leq k$.
- The language of G,
$$L(G) = \{w \in \Sigma^* \mid S \stackrel{*}{\Rightarrow} w\}.$$
- A language $L \subseteq \Sigma^*$ is called a *context-free language* (CFL in short) if there is a CFG G, such that $L = L(G)$.

1.3 Examples of CFGs

1. $L_1 = \{a^n b^m \mid n \geq 0, n \leq m \leq 2n\}$

   CFG for L_1:

$$\begin{aligned} S &\longrightarrow ASB \mid \epsilon \\ A &\longrightarrow a \\ B &\longrightarrow bb \mid b \end{aligned}$$

   Intuition: Since the number of b's can be any number between n and $2n$, we have two variables B_1 and B_2 for every occurrence of A. B_1 produces only b hence there are n b's corresponding to B_1. B_2 produces b and ϵ, hence there are k b's corresponding to B_2, where $0 \leq k \leq n$, is the number of times the rule $B_2 \longrightarrow b$ is used.

   Exercise 2. Construct CFGs for the following languages:

   (a) $L = \{a^n b^m \mid n \geq 0,\ 3n \leq m \leq 5m\}$

   (b) $L = \{a^n b^m \mid n \geq 0,\ n < m \leq 2n\}$

   (c) $L = \{a^n b^m c^l \mid m = n + l,\ l, m, n \geq 0\}$

2. $L_2 = \{w \in \{0,1\}^* \mid \#_0(w) = \#_1(w)\}$

   CFG for L_2:

$$S \longrightarrow 1S0S \mid 0S1S \mid \epsilon$$

   Intuition: Divide all non-empty strings in L_2 into two cases: strings that begin with a 0 and strings that begin with a 1. Let w be a string in first case. Then w is of the form, $w = 0w_1 1w_2$, where w_1 and w_2 are strings with equal number of 0's and 1's (can be either of the two cases). Here the 1 between w_1 and w_2 is the 'matching' 1 for the starting 0. The second case is symmetric.

3. A string of parentheses is balanced if each left parenthesis has a matching right parenthesis and the matching pairs are well nested. Consider the following language consisting of balanced parentheses:

   $L_3 = \{w \in \{[,]\}^* \mid w \text{ is balanced}\}$

   CFG for L_3:

$$S \longrightarrow [S] \mid SS \mid \epsilon$$

   Intuition: For every string w in L_3, either the first open bracket is matched with the last close bracket or it is matched with some intermediate close bracket. If the first open bracket is matched with the last close bracket then the intermediate string is also balanced and is strictly smaller in length. This is handled by the first rule. Otherwise if the first open bracket is matched by an intermediate close bracket then w can be written as $w = w_1w_2$ such that both w_1 and w_2 are balanced and w_1 ends with the matching close bracket for the first open bracket of w. This case is handled by the second rule.

   Exercise 3. Construct a CFG for the following language:

$$\text{PALINDROMES} = \{w \in \{a, b\}^* \mid w = \text{rev}(w)\}$$

1.4 Regular languages are context-free

Let L be a regular language accepted via a DFA $D = (Q, \Sigma, \delta, q_0, F)$. We will construct a CFG $G = (V, \Sigma, P, R_0)$ for L.

- $V = \{R_i \mid q_i \in Q,\ \forall i\}$. In other words, there is a variable in the CFG corresponding to every state in the DFA.
- The set of terminals is the same as the alphabet of D.
- R_0 is the start variable, that is, the variable corresponding to the start state.
- If $\delta(q_i, a) = q_j$, then the substitution rule $R_i \longrightarrow aR_j$ is in P.
  Also if $q_i \in F$, then $R_i \longrightarrow \epsilon$ is a substitution rule in P.

Lemma 1. $L(D) = L(G)$

Proof. First we will show that $L(D) \subseteq L(G)$.

Claim 2. *For $w \in \Sigma^*$, if $\delta(q_0, w) = q_i$ in D then $R_0 \stackrel{*}{\Rightarrow} wR_i$ in G.*

Proof. We will prove by in induction on the length of w. If $w = \epsilon$, then the claim follows from the definition of $\stackrel{*}{\Rightarrow}$. Let $w = xa$ where $a \in \Sigma$. Then, $\delta(q_0, w) = \delta(q_0, xa) = \delta(\delta(q_0, x), a) = q_i(say)$. Let $\delta(q_0, x) = q_j$. By induction hypothesis we have $R_0 \stackrel{*}{\Rightarrow} xR_j$. Also since $\delta(q_j, a) = q_i$, we have the substitution rule $R_j \longrightarrow aR_i$ in G. Therefore, $R_0 \stackrel{*}{\Rightarrow} xaR_i$. □

Now if $w \in L(D)$, then $q_i \in F$, which implies that $R_i \longrightarrow \epsilon$ is a substitution rule, and therefore $R_0 \stackrel{*}{\Rightarrow} w$.
We will now show that $L(G) \subseteq L(D)$.

Claim 3. *For $w \in \Sigma^*$, if $R_0 \stackrel{*}{\Rightarrow} wR_i$ in G, then $\delta(q_0, w) = q_i$ in D.*

Proof. We will prove by in induction on the derivation length of w. If length is 0 then the claim follows trivially. Let $w = xa$ where $a \in \Sigma$. If $R_0 \stackrel{*}{\Rightarrow} xaR_i$ then $R_0 \stackrel{*}{\Rightarrow} xR_j$ and $R_j \longrightarrow aR_i$ is a substitution rule. This is because of the type of substitution rules that G has (a terminal followed by a variable on the right hand side). BY induction hypothesis we have $\delta(q_0, x) = q_j$ and since $R_j \longrightarrow aR_i$ is a substitution rule, we have $\delta(q_j, a) = q_i$. Therefore, $\delta(q_0, xa) = q_i$. □

Now if $w \in L(G)$, then $R_i \longrightarrow \epsilon$ is a substitution rule, which implies that $q_i \in F$, and therefore $\delta(q_0, w) \in F$.
Hence the lemma follows. □

2 Ambiguity

2.1 Parse Trees

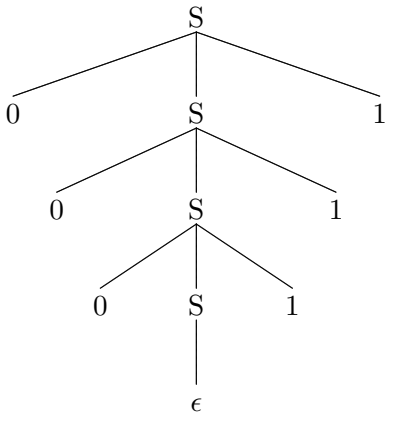
A *parse tree* for a string w with respect to a CFG G is a rooted, ordered tree that represents the derivation of w with respect to the grammar. It represents the "syntactic" structure of the string.

Examples of Parse Trees

1. Consider the CFG, say G_1,

$$S \longrightarrow 0S1 \mid \epsilon$$

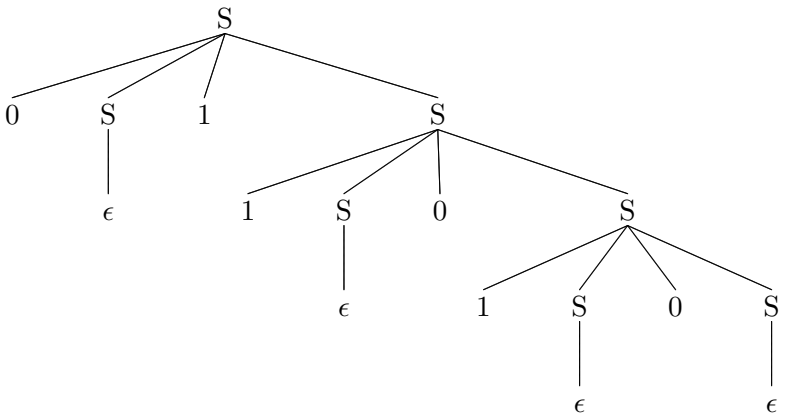
The parse tree of the string 000111 with respect to the above grammar is



2. Now consider the CFG, say G_2,

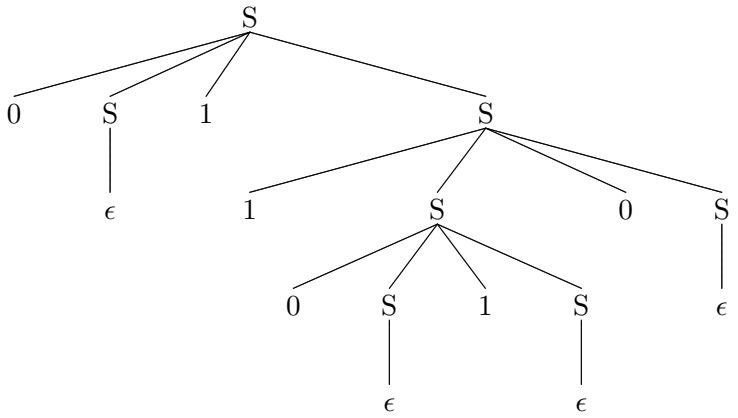
$$S \longrightarrow 1S0S \mid 0S1S \mid \epsilon$$

The parse tree of the string 011010 with respect to the above grammar is



Note that in a parse tree concatenating the symbols from left to right gives the original string.

Another parse tree for the *same* string with respect to the *same* grammar.



Observe that in the first example there is a unique parse tree for every string in the language. But in the second example we have a string in the language that has two parse trees.

Remark. Given a CFG G, there can exist multiple parse trees for a string $w \in L(G)$.

Here are some properties of a parse tree that will be useful for us later on.

1. Every internal node of the tree is labelled by a variable.
2. Every leaf node is labelled either by a terminal or by ϵ. Moreover, if a leaf is labelled ϵ then it is the only child of its parent.
3. If an internal node is labelled A and its children are labelled $X_1, X_2, \ldots, X_k$ in order from left to right, then $A \longrightarrow X_1 X_2 \ldots X_k$ is a rule in the CFG.

2.2 Derivation and Ambiguity

The *derivation* of a string w with respect to a CFG is a sequence of substitutions that yields w starting from the start variable.

Examples

1. Derivation of 000111 with respect to G_1.

$$S \implies 0S1 \implies 00S11 \implies 000S111 \implies 000111$$

   Observe that the string 000111 has only one derivation with respect to G_1, because the string generated in each intermediate step has exactly one variable. In fact every string in $L(G_1)$ has exactly one derivation. However this need not be the case with every grammar as we shall see.

2. Consider the string 0101 in the language of the CFG G_2. Below we show 3 derivations of the string with respect to G_2.

   Derivation 1: $\underline{S} \implies 0\underline{S}1S \implies 01\underline{S} \implies 010\underline{S}1S \implies 0101\underline{S} \implies 0101$

   Derivation 2: $\underline{S} \implies 0S1\underline{S} \implies 0S10\underline{S}1S \implies 0S101\underline{S} \implies 0\underline{S}101 \implies 0101$

   Derivation 3: $\underline{S} \implies 0\underline{S}1S \implies 01\underline{S}0S1S \implies 010\underline{S}1S \implies 0101\underline{S} \implies 0101$

   The variable that gets replaced at each step of the derivation is underlined.

Let G be a grammar and $w \in L(G)$.

- A derivation of w with respect to G is a *leftmost derivation* if at every step the leftmost variable gets substituted.

  In example 2 above, derivations 1 and 3 are leftmost derivations, whereas 2 is not.

- We say that w is *ambiguous* with respect to G if w has at least 2 leftmost derivations with respect to G. Otherwise we say that w is *unambiguous*.

- A CFG G is said to be *ambiguous* if it generates some string ambiguously. The grammar G_1 above is unambiguous and G_2 is ambiguous.

- A CFL is said to be *inherently ambiguous* if all CFGs that accept the language are ambiguous.

Theorem 4. *Let G be a grammar and $w \in L(G)$. w is unambiguous with respect to G if and only if w has a unique parse tree with respect to G.*

Exercise 4. Prove Theorem 4.

Exercise 5. Give an example of an unambiguous grammar that has at least 2 derivations for some string.