Lecture 7

Context-free grammars and languages

The next class of languages we will study in the course is the class of *context-free languages*. They are defined by the notion of a *context-free grammar*, or a CFG for short, which you will have encountered previously in your studies (such as in CS 241).

7.1 Definitions of CFGs and CFLs

We will start with the following definition for context-free grammars.

Definition 7.1. A *context-free grammar* (or *CFG* for short) is a 4-tuple

$$G = (V, \Sigma, R, S), \tag{7.1}$$

where V is a finite and non-empty set (whose elements we will call *variables*), Σ is an *alphabet* (disjoint from V), R is a finite and nonempty set of *rules*, each of which takes the form

$$A \to w \tag{7.2}$$

for some choice of $A \in V$ and $w \in (V \cup \Sigma)^*$, and $S \in V$ is a variable called the *start variable*.

Example 7.2. For our first example of a CFG, we may consider $G = (V, \Sigma, R, S)$, where $V = \{S\}$ (so that there is just one variable in this grammar), $\Sigma = \{0, 1\}$, S is the start variable, and R contains these two rules:

$$\begin{aligned} S &\to 0\,S\,1 \\ S &\to \varepsilon. \end{aligned} \tag{7.3}$$

It is often convenient to describe a CFG just by listing the rules, as we have in (7.3). When we do this, it is to be understood that the set of variables V and

the alphabet Σ are determined implicitly: the variables are the capital letters and the alphabet contains the symbols on the right-hand side of the rules that are left over. Moreover, the start variable is understood to be the variable appearing on the left-hand side of the first rule that is listed.

Note that these are just conventions that allow us to save time, and you could simply list each of the elements V, Σ, R, and S if it was likely that the conventions would cause confusion (such as a case in which you might prefer a capital letter to be treated as an alphabet symbol, or to have a variable that doesn't appear in any rules).

Every context-free grammar $G = (V, \Sigma, R, S)$ *generates* a language $\mathrm{L}(G) \subseteq \Sigma^*$. Informally speaking, this is the language consisting of *all* strings that can be obtained by the following process:

1. Write down the start variable S.
2. Repeat the following steps any number of times:
   a. Choose any rule $A \to w$ from R.
   b. Within the string of variables and alphabet symbols you currently have written down, replace any instance of the variable A with the string w.
3. If you are eventually left with a string of the form $x \in \Sigma^*$, so that no variables remain, then stop. The string x has been obtained by the process, and is therefore among the strings generated by G.

Example 7.3. The CFG G described in Example 7.2 generates the language

$$\mathrm{L}(G) = \{0^n 1^n \,:\, n \in \mathbb{N}\}. \tag{7.4}$$

This is because we begin by writing down the start variable S, then we choose one of the two rules and perform the replacement in the only way possble: there will always be a single variable S in the middle of the string, and we replace it either by $0S1$ or by ε. The process ends precisely when we choose the rule $S \to \varepsilon$, and depending on how many times we chose the rule $S \to 0S1$ we obtain one of the strings

$$\varepsilon,\ 01,\ 0011,\ 000111,\ \ldots \tag{7.5}$$

and so on. The set of all strings that can possibly be obtained is therefore given by (7.4).

The description of the language generated by a CFG suggested above provides an intuitive, human-readable way to explain this concept, but it is not very satisfying from a mathematical viewpoint—we would prefer a definition based on sets,

functions, and so on (rather than one that refers to "writing down" variables, for instance). One way to define this notion mathematically begins with the specification of the *yields relation* of a grammar that captures the notion of performing a substitution.

Definition 7.4. Let $G = (V, \Sigma, R, S)$ be a context-free grammar. The *yields relation* defined by G is a relation defined for pairs of strings over the alphabet $V \cup \Sigma$ as follows:

$$uAv \Rightarrow_G uwv \tag{7.6}$$

for every choice of strings $u, v, w \in (V \cup \Sigma)^*$ and a variable $A \in V$, provided that the rule $A \to w$ is included in R.[1]

The interpretation of this relation is that $x \Rightarrow_G y$, for $x, y \in (V \cup \Sigma)^*$, when it is possible to replace one of the variables appearing in x according to one of the rules of G in order to obtain y.

It will also be convenient to consider the reflexive transitive closure of this relation, which is defined as follows.

Definition 7.5. Let $G = (V, \Sigma, R, S)$ be a context-free grammar. For any two strings $x, y \in (V \cup \Sigma)^*$ it holds that

$$x \stackrel{*}{\Rightarrow}_G y \tag{7.7}$$

if there exists a positive integer m and strings $z_1, \ldots, z_m \in (V \cup \Sigma)^*$ such that $x = z_1$, $y = z_m$, and $z_k \Rightarrow_G z_{k+1}$ for all $k \in \{1, \ldots, m-1\}$.

In this case the interpretation of this relation is that $x \stackrel{*}{\Rightarrow}_G y$ holds when it is possible to transform x into y by performing zero or more substitutions according to the rules of G.

When a CFG G is fixed or can be safely taken as implicit, we will sometimes write $\Rightarrow$ rather than $\Rightarrow_G$, and likewise for the starred version.

We can now use the relation just defined to formally define the language generated by a given context-free grammar.

Definition 7.6. Let $G = (V, \Sigma, R, S)$ be a context-free grammar. The *language generated* by G is

$$\mathrm{L}(G) = \left\{ x \in \Sigma^* \,:\, S \stackrel{*}{\Rightarrow}_G x \right\}. \tag{7.8}$$

[1] Recall that a relation is a subset of a Cartesian product of two sets. In this case, the relation is the subset $\{(uAv, uwv) \,:\, u, v, w \in (V \cup \Sigma)^*, A \in V, \text{ and } A \to w \text{ is a rule in } R\}$. The notation $uAv \Rightarrow_G uwv$ is a more readable way of indicating that the pair (uAv, uwv) is an element of the relation.

If it is the case that $x \in \mathrm{L}(G)$ for a context-free grammar $G = (V, \Sigma, R, S)$, and $z_1, \ldots, z_m \in (V \cup \Sigma)^*$ is a sequence of strings for which it holds that $z_1 = S$, $z_m = x$, and $z_k \Rightarrow_G z_{k+1}$ for all $k \in \{1, \ldots, m-1\}$, then the sequence $z_1, \ldots, z_m$ is said to be a *derivation* of x. If you unravel the definitions above, it becomes clear that there must of course exist at least one derivation for every string $x \in \mathrm{L}(G)$, but in general there may be more than one.

Finally, we define the class of *context-free languages* to be those languages that are generated by context-free grammars.

Definition 7.7. Let Σ be an alphabet and let $A \subseteq \Sigma^*$ be a language. The language A is *context-free* if there exists a context-free grammar G such that $\mathrm{L}(G) = A$.

Example 7.8. The language $\{0^n 1^n : n \in \mathbb{N}\}$ is a context-free language, as has been established in Example 7.3.

7.2 Basic examples

We've seen one example of a context-free language so far: $\{0^n 1^n : n \in \mathbb{N}\}$. Let us now consider a few more examples.

Example 7.9. The language

$$\mathrm{PAL} = \left\{ w \in \Sigma^* : w = w^{\mathrm{R}} \right\} \tag{7.9}$$

over the alphabet $\Sigma = \{0, 1\}$ is context-free. (In fact this is true for any choice of an alphabet Σ, but we'll stick to the binary alphabet for now for simplicity). To verify that this language is context-free, it suffices to exhibit a context-free grammar that generates it. Here is one that works:

$$\begin{aligned} S &\to 0\,S\,0 \\ S &\to 1\,S\,1 \\ S &\to 0 \\ S &\to 1 \\ S &\to \varepsilon \end{aligned} \tag{7.10}$$

We've named the language in the above example PAL because it is short for *palindrome*, which is something that reads the same forwards and backwards (like "Never odd or even" or "Yo, banana boy," so long as you ignore punctuation, spaces, and letter case—although in this example we're restricting our attention to binary string palindromes).

We often use a short-hand notation for describing grammars in which the same variable appears on the left-hand side of multiple rules, as is the case for the grammar described in the previous example. The short-hand notation is to write the variable on the left-hand side and the arrow just once, and to draw a vertical bar (which can be read as "or") among the possible alternatives for the right-hand side like this:

$$S \to 0\,S\,0 \,\big|\, 1\,S\,1 \,\big|\, 0 \,\big|\, 1 \,\big|\, \varepsilon \tag{7.11}$$

(If you use this short-hand notation when you are writing by hand, such as on an exam, be sure to make your bars tall enough so that they are easily distinguished from 1s.)

Sometimes it is easy to see that a particular CFG generates a given language—for instance, I would consider this to be obvious in the case of the previous example. In other cases it can be more challenging, or even impossibly difficult, to verify that a particular grammar generates a particular language. The next example illustrates a case in which such a verification is nontrivial.

Example 7.10. Let $\Sigma = \{0, 1\}$ be the binary alphabet, and define a language $A \subseteq \Sigma^*$ as follows:

$$A = \big\{w \in \Sigma^* \,:\, |w|_0 = |w|_1\big\}. \tag{7.12}$$

Here we are using a convenient notation: $|w|_0$ denotes the number of times the symbol 0 appears in w, and similarly $|w|_1$ denotes the number of times the symbol 1 appears in w. The language A therefore contains all binary strings having the same number of 0s and 1s. This is a context-free language, as it is generated by this context-free grammar:

$$S \to 0\,S\,1\,S \,\big|\, 1\,S\,0\,S \,\big|\, \varepsilon. \tag{7.13}$$

Now, it is pretty clear that every string generated by the grammar (which we will call G) described in the above example is contained in A; we begin a derivation with just the variable S, so there are an equal number of 0s and 1s written down at the start (zero of each, to be precise), and every rule maintains this property as an invariant.

On the other hand, it is not immediately obvious that every element of A can be generated by G. Let us prove that this is indeed the case.

Claim 7.11. $A \subseteq \mathrm{L}(G)$.

Proof. Let $w \in A$ be a string contained in A and let $n = |w|$. We will prove that $w \in \mathrm{L}(G)$ by (strong) induction on n.

The base case is $n = 0$, which means that $w = \varepsilon$. We have that $S \Rightarrow_G \varepsilon$ represents a derivation of ε, and therefore $w \in \mathrm{L}(G)$.

For the induction step, we assume that $n \geq 1$, and we assume that for all strings $x \in A$ with $|x| < n$ it holds that $x \in \mathrm{L}(G)$. Our goal is to prove that G generates w. Let us write

$$w = \sigma_1 \cdots \sigma_n \tag{7.14}$$

for $\sigma_1, \ldots, \sigma_n \in \Sigma$. We have assumed that $w \in A$, and therefore

$$|\sigma_1 \cdots \sigma_n|_0 = |\sigma_1 \cdots \sigma_n|_1. \tag{7.15}$$

Next, let $m \in \{1, \ldots, n\}$ be the *minimum* value for which it holds that

$$|\sigma_1 \cdots \sigma_m|_0 = |\sigma_1 \cdots \sigma_m|_1; \tag{7.16}$$

we know that this equation is satisfied when $m = n$, and there might be a smaller value of m that works—but in any case we know that m is a well-defined number. We will prove that $\sigma_1 \neq \sigma_m$.

We can reason that $\sigma_1 \neq \sigma_m$ using proof by contradiction. Toward this goal, assume $\sigma_1 = \sigma_m$, and define

$$d_k = |\sigma_1 \cdots \sigma_k|_1 - |\sigma_1 \cdots \sigma_k|_0 \tag{7.17}$$

for every $k \in \{1, \ldots, m\}$. We know that $d_m = 0$ because (7.16) holds. Moreover, because the equations

$$\begin{aligned} |\sigma_1 \cdots \sigma_m|_1 &= |\sigma_1 \cdots \sigma_{m-1}|_1 + |\sigma_m|_1 = |\sigma_1 \cdots \sigma_{m-1}|_1 + |\sigma_1|_1 \\ |\sigma_1 \cdots \sigma_m|_0 &= |\sigma_1 \cdots \sigma_{m-1}|_0 + |\sigma_m|_0 = |\sigma_1 \cdots \sigma_{m-1}|_0 + |\sigma_1|_0 \end{aligned} \tag{7.18}$$

hold, we conclude that

$$d_m = d_{m-1} + d_1 \tag{7.19}$$

by subtracting the second equation from the first. Therefore, because $d_m = 0$ and d_1 is nonzero, it must be that d_{m-1} is also nonzero, and more importantly d_1 and d_{m-1} must have opposite sign. However, because consecutive values of d_k must always differ by 1 and can only take integer values, we conclude that there must exist a choice of k in the range $\{2, \ldots, m-2\}$ for which $d_k = 0$, for otherwise it would not be possible for d_1 and d_{m-1} to have opposite sign. This, however, is in contradiction with m being the minimum value for which (7.16) holds. We have therefore concluded that $\sigma_1 \neq \sigma_m$.

At this point it is possible to describe a derivation for w. We have $w = \sigma_1 \cdots \sigma_n$, and we have that

$$|\sigma_1 \cdots \sigma_m|_0 = |\sigma_1 \cdots \sigma_m|_1 \quad \text{and} \quad \sigma_1 \neq \sigma_m \tag{7.20}$$

for some choice of $m \in \{1, \ldots, n\}$. We conclude that

$$|\sigma_2 \cdots \sigma_{m-1}|_0 = |\sigma_2 \cdots \sigma_{m-1}|_1 \quad \text{and} \quad |\sigma_{m+1} \cdots \sigma_n|_0 = |\sigma_{m+1} \cdots \sigma_n|_1. \tag{7.21}$$

By the hypothesis of induction it follows that

$$S \stackrel{*}{\Rightarrow}_G \sigma_2 \cdots \sigma_{m-1} \quad \text{and} \quad S \stackrel{*}{\Rightarrow}_G \sigma_{m+1} \cdots \sigma_n. \tag{7.22}$$

Therefore the string w satisfies

$$S \Rightarrow_G 0\,S\,1\,S \stackrel{*}{\Rightarrow}_G 0\,\sigma_2 \cdots \sigma_{m-1}\,1\,\sigma_{m+1} \cdots \sigma_n = w \tag{7.23}$$

(in case $\sigma_1 = 0$ and $\sigma_m = 1$) or

$$S \Rightarrow_G 1\,S\,0\,S \stackrel{*}{\Rightarrow}_G 1\,\sigma_2 \cdots \sigma_{m-1}\,0\,\sigma_{m+1} \cdots \sigma_n = w \tag{7.24}$$

(in case $\sigma_1 = 1$ and $\sigma_m = 0$). We have proved that $w \in \mathrm{L}(G)$ as required. □

Here is another example that is related to the previous one. It is an important example and we'll refer to it from time to time throughout the course.

Example 7.12. Consider the alphabet $\Sigma = \{\,(\,,\,)\,\}$. That is, we have two symbols in this alphabet: left-parenthesis and right-parenthesis.

To say that a string w over the alphabet Σ is *properly balanced* means that by repeatedly removing the substring $()$, you can eventually reach ε. More intuitively speaking, a string over Σ is properly balanced if it would make sense to use this pattern of parentheses in an ordinary arithmetic expression (ignoring everything besides the parentheses). These are examples of properly balanced strings:

$$(\,(\,)\,(\,)\,)\,(\,), \quad (\,(\,(\,)\,(\,)\,)\,), \quad (\,(\,)\,), \quad \text{and} \quad \varepsilon. \tag{7.25}$$

These are examples of strings that are not properly balanced:

$$(\,(\,(\,)\,)\,(\,), \quad \text{and} \quad (\,)\,)\,(. \tag{7.26}$$

Now define a language

$$\mathrm{BAL} = \bigl\{w \in \Sigma^* \,:\, w \text{ is properly balanced}\bigr\}. \tag{7.27}$$

It is the case that the language BAL is context-free—here is a very simple context-free grammar that generates it:

$$S \to (\,S\,)\,S \;\big|\; \varepsilon. \tag{7.28}$$