Lecture 9

Properties of context-free languages

In this lecture we will examine various properties of the class of context-free languages, including the fact that it is closed under the regular operations, that every regular language is context free, and that the intersection of a context-free language and a regular language is always context free.

9.1 Closure under the regular operations

Let us begin by proving that the context-free languages are closed under the regular operations.

Theorem 9.1. *Let Σ be an alphabet and let $A, B \subseteq \Sigma^*$ be context-free languages. The languages $A \cup B$, AB, and A^* are context free.*

Proof. Because A and B are context-free languages, there must exist context-free grammars

$$G_A = \big(V_A, \Sigma, R_A, S_A\big) \quad \text{and} \quad G_B = \big(V_B, \Sigma, R_B, S_B\big) \tag{9.1}$$

such that $\mathrm{L}(G_A) = A$ and $\mathrm{L}(G_B) = B$. Because the specific names we choose for the variables in a context-free grammar have no effect on the language it generates, there is no loss of generality in assuming V_A and V_B are disjoint sets.

First let us construct a CFG G for the language $A \cup B$. This CFG will include all of the variables and rules of G_A and G_B together, along with a new variable S (which we assume is not already contained in V_A or V_B, and which we will take to be the start variable of G) and two new rules:

$$S \to S_A \mid S_B. \tag{9.2}$$

Formally speaking we may write

$$G = (V, \Sigma, R, S) \tag{9.3}$$

where $V = V_A \cup V_B \cup \{S\}$ and $R = R_A \cup R_B \cup \{S \to S_A,\ S \to S_B\}$. In the typical style in which we write CFGs, the grammar G looks like this:

$$S \to S_A \mid S_B$$

all rules of G_A

all rules of G_B

It is evident that $\mathrm{L}(G) = A \cup B$; each derivation may begin with $S \Rightarrow S_A$ or $S \Rightarrow S_B$, after which either S_A generates any string in A or S_B generates any string in B. As the language $A \cup B$ is generated by the CFG G, we have that it is context free.

Next we will construct a CFG H for the language AB. The construction of H is very similar to the construction of G above. The CFG H will include all of the variables and rules of G_A and G_B, along with a new start variable S and one new rule:

$$S \to S_A S_B. \tag{9.4}$$

Formally speaking we may write

$$H = (V, \Sigma, R, S) \tag{9.5}$$

where $V = V_A \cup V_B \cup \{S\}$ and $R = R_A \cup R_B \cup \{S \to S_A S_B\}$. In the typical style in which we write CFGs, the grammar G looks like this:

$$S \to S_A S_B$$

all rules of G_A

all rules of G_B

It is evident that $\mathrm{L}(G) = AB$; each derivation must begin with $S \Rightarrow S_A S_B$, and then S_A generates any string in A and S_B generates any string in B. As the language AB is generated by the CFG H, we have that it is context free.

Finally we will construct a CFG K for A^*. This time the CFG K will include just the rules and variables of G_A, along with a new start variable S and two new rules:

$$S \to S\,S_A \mid \varepsilon. \tag{9.6}$$

Formally speaking we may write

$$K = (V, \Sigma, R, S) \tag{9.7}$$

where $V = V_A \cup \{S\}$ and $R = R_A \cup \{S \to S\,S_A, S \to \varepsilon\}$. In the typical style in which we write CFGs, the grammar K looks like this:

$$S \to S\,S_A \mid \varepsilon$$

Every possible left-most derivation of a string by K must begin with zero or more applications of the rule $S \to S\,S_A$ followed by the rule $S \to \varepsilon$. This means that every left-most derivation begins with a sequence of rule applications that is consistent with one of the following relationships:

$$\begin{aligned} S &\stackrel{*}{\Rightarrow} \varepsilon \\ S &\stackrel{*}{\Rightarrow} S_A \\ S &\stackrel{*}{\Rightarrow} S_A\,S_A \\ S &\stackrel{*}{\Rightarrow} S_A\,S_A\,S_A \\ &\vdots \end{aligned} \tag{9.8}$$

and so on. After this, each occurrence of S_A generates any string in A. It is therefore the case that $\mathrm{L}(K) = A^*$, so that A^* is context free. □

9.2 Relationships to regular languages

This section discusses relationships between context-free languages and regular languages. In particular, we will prove that every regular language is context free, and (more generally) that the intersection between a context-free language and a regular language is always context free.

Every regular language is context free

Let us begin with the first fact suggested above, which is that every regular language is also context free. We will discuss two different ways to prove this fact.

Theorem 9.2. *Let Σ be an alphabet and let $A \subseteq \Sigma^*$ be a regular language. The language A is context free.*

First proof. With every regular expression R over the alphabet Σ, one may associate a CFG G by recursively applying these simple constructions:

1. If $R = \varnothing$, then G is the CFG
$$S \to S, \tag{9.9}$$
which generates the empty language $\varnothing$.
2. If $R = \varepsilon$, then G is the CFG
$$S \to \varepsilon, \tag{9.10}$$
which generates the language $\{\varepsilon\}$.
3. If $R = a$ for $a \in \Sigma$, then G is the CFG
$$S \to a, \tag{9.11}$$
which generates the language $\{a\}$.
4. If $R = (R_1 \cup R_2)$, then G is the CFG generating the language $\mathrm{L}(G_1) \cup \mathrm{L}(G_2)$, as described in the proof of Theorem 9.1, where G_1 and G_2 are CFGs associated with the regular expressions R_1 and R_2, respectively.
5. If $R = (R_1 R_2)$, then G is the CFG generating the language $\mathrm{L}(G_1)\,\mathrm{L}(G_2)$, as described in the proof of Theorem 9.1, where G_1 and G_2 are CFGs associated with the regular expressions R_1 and R_2, respectively.
6. If $R = (R_1^*)$, then G is the CFG generating the language $\mathrm{L}(G_1)^*$, as described in the proof of Theorem 9.1, where G_1 is the CFG associated with the regular expression R_1.

In each case, we observe that $\mathrm{L}(G) = \mathrm{L}(R)$.

Now, by the assumption that A is regular, there must exist a regular expression R such that $\mathrm{L}(R) = A$. For the CFG G obtained from R as described above, we find that $\mathrm{L}(G) = A$, and therefore A is context free. □

Second proof. Because A is regular, there must exist a DFA

$$M = (Q, \Sigma, \delta, q_0, F) \tag{9.12}$$

such that $\mathrm{L}(M) = A$.

We will define a CFG G that effectively simulates M, generating exactly those strings that are accepted by M. In particular, we will define

$$G = \big(V, \Sigma, R, X_{q_0}\big) \tag{9.13}$$

where the variables are $V = \{X_q \,:\, q \in Q\}$ (one variable for each state of M) and the following rules are to be included in R:

1. For each choice of $p, q \in Q$ and $a \in \Sigma$ satisfying $\delta(p, a) = q$, the rule

   $$X_p \to a X_q \tag{9.14}$$

   is included in R.

2. For each state $q \in F$, the rule

   $$X_q \to \varepsilon \tag{9.15}$$

   is included in R.

Now, by examining the rules suggested above, we see that every derivation of a string by G begins with the start variable (of course), involves zero or more applications of rules of the first type listed above, and then ends when a rule of the second type is applied. There will always be a single variable appearing after each step of the derivation, until the very last step in which this variable is eliminated. It is important that this final step is only possible when the variable X_q corresponds to an accept state $q \in F$. By considering the rules of the first type, it is evident that

$$\big[X_{q_0} \stackrel{*}{\Rightarrow} w X_q\big] \;\Leftrightarrow\; \big[\delta^*(q_0, w) = q\big]. \tag{9.16}$$

We therefore have $X_{q_0} \stackrel{*}{\Rightarrow} w$ if and only if there exists a choice of $q \in F$ for which $\delta^*(q_0, w) = q$. This is equivalent to the statement that $\mathrm{L}(G) = \mathrm{L}(M)$, which completes the proof. □

Intersections of regular and context-free languages

The context-free languages are not closed under some operations for which the regular languages are closed. For example, the complement of a context-free language may fail to be context free, and the intersection of two context-free languages may fail to be context free. We will observe both of these facts in the next lecture.

It is the case, however, that the intersection of a context-free language and a regular language is always context free, as we will now prove. The proof is more complicated than most of the other proofs we have seen thus far in the course—if it is not immediately clear, just do your best to try to understand the idea behind it.

Theorem 9.3. *Let Σ be an alphabet, let $A, B \subseteq \Sigma^*$ be languages, and assume A is context free and B is regular. The language $A \cap B$ is context free.*

Proof. The language A is context free, so there exists a CFG that generates it. As discussed in the previous lecture, we may in fact assume that there exists a CFG in Chomsky normal form that generates A. Having this CFG be in Chomsky normal form will greatly simplify the proof. Hereafter we will assume

$$G = (V, \Sigma, R, S) \tag{9.17}$$

is a CFG in Chomsky normal form such that $\mathrm{L}(G) = A$. Because the language B is regular, there must also exist a DFA

$$M = (Q, \Sigma, \delta, q_0, F) \tag{9.18}$$

such that $\mathrm{L}(M) = B$.

The main idea of the proof is to define a new CFG H such that $\mathrm{L}(H) = A \cap B$. The CFG H will have $|Q|^2$ variables for *each* variable of G, which may be a lot but that is not a problem—it is a finite number, and that is all we require of a set of variables of a context-free grammar. In particular, for each variable $X \in V$, we will include a variable $X_{p,q}$ in H for every choice of $p, q \in Q$. In addition, we will add a new start variable S_0 to H.

The intended meaning of each variable $X_{p,q}$ is that it should generate all strings that (i) are generated by X with respect to the grammar G, and (ii) cause M to move from state p to state q. We will accomplish this by adding a collection of rules to H for each rule of G. Because the grammar G is assumed to be in Chomsky normal form, there are just three possible forms for its rules, and they can be handled one at a time as follows:

1. For each rule of the form $X \to a$ in G, include the rule

   $$X_{p,q} \to a \tag{9.19}$$

   in H for every pair of states $p, q \in Q$ for which $\delta(p, a) = q$.
2. For each rule of the form $X \to YZ$ in G, include the rule

   $$X_{p,q} \to Y_{p,r} Z_{r,q} \tag{9.20}$$

   in H for every choice of states $p, q, r \in Q$.

3. If the rule $S \to \varepsilon$ is included in G and $q_0 \in F$ (i.e., $\varepsilon \in A \cap B$), then include the rule
$$S_0 \to \varepsilon \tag{9.21}$$
in H, where S_0 is the new start variable for H mentioned above.

Once we have added all of these rules in H, we also include the rule

$$S_0 \to S_{q_0,p} \tag{9.22}$$

in H for every accept state $p \in F$.

The intended meaning of each variable $X_{p,q}$ in H has been suggested above. More formally speaking, we wish to prove that the following equivalence holds for every nonempty string $w \in \Sigma^*$, every variable $X \in V$, and every choice of states $p, q \in Q$:

$$\big[X_{p,q} \stackrel{*}{\Rightarrow}_H w\big] \Leftrightarrow \big[\big(X \stackrel{*}{\Rightarrow}_G w\big) \wedge \big(\delta^*(p,w) = q\big)\big]. \tag{9.23}$$

The two implications can naturally be handled separately, and one of the two implications naturally splits into two parts.

First, it is almost immediate that the implication

$$\big[X_{p,q} \stackrel{*}{\Rightarrow}_H w\big] \Rightarrow \big[X \stackrel{*}{\Rightarrow}_G w\big] \tag{9.24}$$

holds, as a derivation of w starting from $X_{p,q}$ in H gives a derivation of w starting from X in G if we simply remove all of the subscripts on all of the variables.

Next, we can prove the implication

$$\big[X_{p,q} \stackrel{*}{\Rightarrow}_H w\big] \Rightarrow \big[\delta^*(p,w) = q\big] \tag{9.25}$$

by induction on the length of w. The base case is $|w| = 1$ (because we are assuming $w \neq \varepsilon$), and in this case we must have $X_{p,q} \Rightarrow_H a$ for some $a \in \Sigma$. The only rules that allow such a derivation are of the first type above, which require $\delta(p,a) = q$. In the general case in which $|w| \geq 2$, it must be that

$$X_{p,q} \Rightarrow Y_{p,r} Z_{r,q} \tag{9.26}$$

for variables $Y_{p,r}$ and $Z_{r,q}$ satisfying

$$Y_{p,r} \stackrel{*}{\Rightarrow}_H y \quad \text{and} \quad Z_{r,q} \stackrel{*}{\Rightarrow}_H z \tag{9.27}$$

for strings $y, z \in \Sigma^*$ for which $w = yz$. By the hypothesis of induction we conclude that $\delta^*(p,y) = r$ and $\delta^*(r,z) = q$, so that $\delta^*(p,w) = q$.

Finally, we can prove

$$\big[\big(X \stackrel{*}{\Rightarrow}_G w\big) \wedge \big(\delta^*(p,w) = q\big)\big] \Rightarrow \big[X_{p,q} \stackrel{*}{\Rightarrow}_H w\big], \tag{9.28}$$

again by induction on the length of w. The base case is $|w| = 1$, which is straightforward: if $X \Rightarrow_G a$ and $\delta(p,a) = q$, then $X_{p,q} \Rightarrow_H a$ because the rule that allows for this derivation has been included among the rules of H. In the general case in which $|w| \geq 2$, the relation $X \stackrel{*}{\Rightarrow}_G w$ implies that $X \Rightarrow_G YZ$ for variables $Y, Z \in V$ such that $Y \stackrel{*}{\Rightarrow}_G y$ and $Z \stackrel{*}{\Rightarrow}_G z$, for strings $y, z \in \Sigma^*$ satisfying $w = yz$. Choosing $r \in Q$ so that $\delta^*(p,y) = r$ (and therefore $\delta^*(r,z) = q$), we have that $Y_{p,r} \stackrel{*}{\Rightarrow} y$ and $Z_{r,q} \stackrel{*}{\Rightarrow} z$ by the hypothesis of induction, and therefore $X_{p,q} \Rightarrow_H Y_{p,r}Z_{r,q} \stackrel{*}{\Rightarrow}_H yz = w$.

Because every derivation of a nonempty string by H must begin with

$$S_0 \Rightarrow_H S_{q_0,p} \tag{9.29}$$

for some $p \in F$, we find that the nonempty strings w generated by H are precisely those strings that are generated by G and satisfy $\delta^*(q_0,w) = p$ for some $p \in F$. Equivalently, for $w \neq \varepsilon$ it is the case that $w \in \mathrm{L}(H) \Leftrightarrow w \in A \cap B$. The empty string has been handled as a special case, so it follows that $\mathrm{L}(H) = A \cap B$. The language $A \cap B$ is therefore context free. □

Remark 9.4. Notice that Theorem 9.3 implies Theorem 9.2; one is free to choose $A = \Sigma^*$ (which is context free) and B to be any regular language, and the implication is that $\Sigma^* \cap B = B$ is context free. Because the two proofs of Theorem 9.2 that we already discussed are much simpler than the one above, however, it makes sense that we considered them first.

9.3 Prefixes, suffixes, and substrings

Let us finish off the lecture with just a few quick examples. Recall from Lecture 6 that for any language $A \subseteq \Sigma^*$ we define

$$\mathrm{Prefix}(A) = \big\{x \in \Sigma^* \,:\, \text{there exists } v \in \Sigma^* \text{ such that } xv \in A\big\}, \tag{9.30}$$

$$\mathrm{Suffix}(A) = \big\{x \in \Sigma^* \,:\, \text{there exists } u \in \Sigma^* \text{ such that } ux \in A\big\}, \tag{9.31}$$

$$\mathrm{Substring}(A) = \big\{x \in \Sigma^* \,:\, \text{there exist } u, v \in \Sigma^* \text{ such that } uxv \in A\big\}. \tag{9.32}$$

Let us prove that if A is context free, then each of these languages is also context free. In the interest of time, we will just explain how to come up with context-free grammars for these languages and not go into details regarding the proofs that

these CFGs are correct. In all three cases, we will assume that $G = (V, \Sigma, R, S)$ is a CFG in Chomsky normal form such that $\mathrm{L}(G) = A$.

We will need to make one additional assumption on the grammar G, which is that none of the variables in G generates the empty language. A variable that generates the empty language is called a *useless variable*, and it should not be hard to convince yourself that useless variables are indeed useless (with one exception). That is, if you have any CFG G in Chomsky normal form that generates a nonempty language, you can easily come up with a new CFG in Chomsky normal form for the same language that does not contain any useless variables simply by removing the useless variables and every rule in which a useless variable appears.

The one exception is the empty language itself, which by definition requires that the start variable is useless (and you will need at least one additional useless variable to ensure that the grammar has a nonempty set of rules and obeys the conditions of a CFG in Chomsky normal form). However, we do not need to worry about this case because $\mathrm{Prefix}(\varnothing)$, $\mathrm{Suffix}(\varnothing)$, and $\mathrm{Substring}(\varnothing)$ are all equal to the empty language, and are therefore context free.

For the language $\mathrm{Prefix}(A)$, we will design a CFG H as follows. First, for every variable $X \in V$ used by G we will include this variable in H, and in addition we will also include a variable X_0. The idea is that X will generate exactly the same strings in H that it does in G, while X_0 will generate all the prefixes of the strings generated by X in G. We include rules in H as follows:

1. For every rule of the form $X \to YZ$ in G, include these rules in H:

$$\begin{aligned} X &\to YZ \\ X_0 &\to YZ_0 \mid Y_0 \end{aligned} \tag{9.33}$$

2. For every rule of the form $X \to a$ in G, include these rules in H:

$$\begin{aligned} X &\to a \\ X_0 &\to a \mid \varepsilon \end{aligned} \tag{9.34}$$

Finally, we take S_0 to be the start variable of H.

The idea is similar for the language $\mathrm{Suffix}(A)$, for which we will construct a CFG K. This time, for every variable $X \in V$ used by G we will include this variable in K, and in addition we will also include a variable X_1. The idea is that X will generate exactly the same strings in K that it does in G, while X_1 will generate all the suffixes of the strings generated by X in G. We include rules in K as follows:

1. For every rule of the form $X \to YZ$ in G, include these rules in K:

$$\begin{aligned} X &\to YZ \\ X_1 &\to Y_1Z \mid Z_1 \end{aligned} \tag{9.35}$$

2. For every rule of the form $X \to a$ in G, include these rules in K:

$$\begin{aligned} X &\to a \\ X_1 &\to a \mid \varepsilon \end{aligned} \tag{9.36}$$

Finally, we take S_1 to be the start variable of K.

To obtain a CFG J for Substring(A), we can simply combine the two constructions above (i.e., apply either one to G, then apply the other to the resulting CFG). Equivalently, we can include variables X, X_0, X_1, and X_2 in J for every $X \in V$ and include rules as follows:

1. For every rule of the form $X \to Y\,Z$ in G, include these rules in J:

$$\begin{aligned} X &\to Y\,Z \\ X_0 &\to Y\,Z_0 \mid Y_0 \\ X_1 &\to Y_1\,Z \mid Z_1 \\ X_2 &\to Y_1\,Z_0 \mid Y_2 \mid Z_2 \end{aligned} \tag{9.37}$$

2. For every rule of the form $X \to a$ in G, include these rules in J:

$$\begin{aligned} X &\to a \\ X_0 &\to a \mid \varepsilon \\ X_1 &\to a \mid \varepsilon \\ X_2 &\to a \mid \varepsilon. \end{aligned} \tag{9.38}$$

Finally, we take S_2 to be the start variable of J. The meaning of the variables X, X_0, X_1, and X_2 in J is that they generate precisely the strings generated by X in G, the prefixes, the suffixes, and the substrings of these strings, respectively.