University of Waterloo

CS 360: Introduction to the Theory of Computing

Fall 2017

Techniques for Proving Characterizations of Context-Free Languages

One of the things you should learn in this course is methods for proving that two different ways of characterizing context-free languages are really the same. Typically, one way is a mathematical or English description of L, and the other is a context-free grammar G. We wish to prove that $L = L(G)$.

Proving that $L = L(G)$ means that you must prove two things:
(i) $L(G) \subseteq L$; and
(ii) $L \subseteq L(G)$.
Typically, different techniques are used for each direction, so this handout has two different parts.

Let's start with techniques for (i). You must show that for each string of terminals $w \in \Sigma^*$ generated by G, the string w is in L. In other words, you must show that if $S \Longrightarrow^* w$, then $w \in L$.

(a) Let's start with the example of G given by $S \to aSb \mid ab$ and $L = \{a^i b^i \ : \ i \geq 1\}$. One way to prove $L(G) \subseteq L$ is to characterize the words generated by G, and for this we often use **induction on the number of steps in the derivation**. For this particular problem, the right induction hypothesis is the following: if $S \Longrightarrow^i w \in \Sigma^*$, then $w = a^i b^i$ (for $i \geq 1$).

The proof is by induction on i. If $i = 1$, then $w = ab$.

Otherwise, assume the result is true for $i = n$, and we prove it for $i = n+1$. If $S \Longrightarrow^{n+1} w$, then we can write this derivation as $S \Longrightarrow aSb \Longrightarrow^n w$. It follows that $w = axb$ for some string x, and that $S \Longrightarrow^n x$. Now, by induction, $x = a^n b^n$. Then $w = aa^n b^n b = a^{n+1} b^{n+1}$. We have now shown $L(G) \subseteq L$.

(b) If the grammar contains more than one variable, the following idea is often useful: **break up the set of variables into those that are recursive (i.e., eventually derive a string containing themselves) and those that are not recursive. Characterize the strings generated by recursive variables first (if there are any), using induction on characterizations of all the mutually recursive variables simultaneously. Then, working from the bottom up back to the start variable, prove characterizations of the other variables.**

Here's a simple example. Suppose we have the grammar

$$\begin{aligned} S &\to BC \\ B &\to aBb \mid ab \\ C &\to bCc \mid bc. \end{aligned}$$

Then we can use a proof similar to that just given in (a) to show that if $B \Longrightarrow^* w$ then $w \in L_1 = \{a^n b^n \ : \ n \geq 1\}$. Similarly, if $C \Longrightarrow^* w$ then $w \in L_2 = \{b^m c^m \ : \ m \geq 1\}$. Finally, it now follows that if $S \Longrightarrow^* w$ then $w \in L_1 L_2 = \{a^n b^{n+m} c^m \ : \ m, n \geq 1\}$.

(c) Here's another example with two mutually recursive variables, where you have to do a *simultaneous* induction on characterizations for both variables.

Consider G given by

$$\begin{aligned} S &\rightarrow aB \mid a \\ B &\rightarrow bS \mid b. \end{aligned}$$

Let $L_1 = (ab)^*(a \cup ab)$ and $L_2 = (ba)^*(b \cup ba)$. We claim that if $S \Longrightarrow^i w$ for $w \in \Sigma^*$, then $w \in L_1$, and if $B \Longrightarrow^i x$ for $x \in \Sigma^*$, then $x \in L_2$.

We prove the two claims simultaneously by induction on i. If $i = 1$, then $S \Longrightarrow^i a$ and $B \Longrightarrow^i b$. Otherwise, assume the result is true for $i = n$, and we prove it for $i = n + 1$. If $S \Longrightarrow^{n+1} w$, then we can write $S \Longrightarrow aB \Longrightarrow^n w$. We have $w = ax$, and $B \Longrightarrow^n x$. By induction we know $x \in L_2 = (ba)^*(b \cup ba)$. Then $w \in a(ba)^*(b \cup ba) = (ab)^* a(b \cup ba) = (ab)^* ab \cup (ab)^+ a \subseteq L_1$. Similarly, if $B \Longrightarrow^{n+1} x$, then we can write $B \Longrightarrow bS \Longrightarrow^n x$. Hence $x = bw$, and $S \Longrightarrow^n w$. By induction we know that $w \in L_1 = (ab)^*(a \cup ab)$. Then $x \in b(ab)^*(a \cup ab) = (ba)^* b(a \cup ab) = (ba)^* ba \cup (ba)^+ b \subseteq L_2$. We're done.

(d) Yet another method is to **determine an invariant property of all sentential forms that always holds at every step of a derivation, and which implies what you want when the sentential form contains no variables**. For example, suppose your grammar G is given by

$$\begin{aligned} S &\rightarrow AASB \mid AAB \\ A &\rightarrow a \\ B &\rightarrow bbb. \end{aligned}$$

We'd like to show $L(G) \subseteq L$, where $L = \{a^{2n} b^{3n} \ : \ n \geq 1\}$.

To do this, we try to find an invariant that is true of every sentential form derivable from S. A first try might be: if $S \Longrightarrow^* \alpha$, then $3|\alpha|_a = 2|\alpha|_b$. (Recall that $|x|_a$ means the number of occurrences of the letter a in x.) However, this is false: we have $S \Longrightarrow AAB \Longrightarrow aAB = \alpha$, and $3|\alpha|_a = 3 \neq 2|\alpha|_b$. So we might then try: if $S \Longrightarrow^* \alpha$, then $3(|\alpha|_a + |\alpha|_A) = 2(|\alpha|_b + 3|\alpha|_B)$. This is not strong enough, however, since a string of terminals that satisfies this equation might not necessarily be in L. (Example: *bbbaa*.) So here is our last invariant: if $S \Longrightarrow^* \alpha$, then $3(|\alpha|_a + |\alpha|_A) = 2(|\alpha|_b + 3|\alpha|_B)$, **and** in α all the a's and A's precede S (if it occurs in α), which in turn precedes all the b's and B's.

Let us prove that this invariant holds for all sentential forms α derivable from S. To do so, we use induction on the length of the derivation. Suppose $S \Longrightarrow^i \alpha$. If $i = 0$, then $\alpha = S$, and it is easy to see the invariant holds. Now assume it holds for $i \leq n$; we will try to prove it for $i = n + 1$. If $S \Longrightarrow^{n+1} \alpha$, then $S \Longrightarrow^n \beta \Longrightarrow \alpha$. The last step $\beta \Longrightarrow \alpha$ uses

the production $S \to AASB$ or $S \to AAB$ or $A \to a$ or $B \to bbb$. Now just check that each of these choices preserves the invariant.

We have now proved the invariant by induction. The invariant, though, implies that if $w \in \Sigma^*$ and $S \Longrightarrow^* w$, then $w = a^{2k}b^{3k}$ for some $k \geq 0$. Clearly $\epsilon \notin L(G)$. It follows that $L(G) \subseteq L$.

Now let's turn to techniques for proving (ii). In this case, we want to prove $L \subseteq L(G)$. In other words, for each string w from L, you must produce a derivation of w in G for it. Typically, this direction uses **induction on the length of the string** w.

(a) As an example, let's return to G given by $S \to aSb \mid ab$, and $L = \{a^ib^i \ : \ i \geq 1\}$. Let us prove, by induction on i, that $S \Longrightarrow^* a^ib^i$. For $i = 1$, this is clear, since we have $S \to ab$ is a production. Assume the result is true for $i = n$; we prove it for $i = n+1$. Then $S \Longrightarrow aSb$, and by induction, $S \Longrightarrow^* a^nb^n$. Putting these together, $S \Longrightarrow aSb \Longrightarrow^* a(a^nb^n)b = a^{n+1}b^{n+1}$, as desired.

(b) Sometimes the argument can get rather complicated. Here's an example: let G be given by

$$S \to a \mid aS \mid bSS \mid SSb \mid SbS,$$

and let $L = \{w \in \{a, b\}^* \ : \ |w|_a > |w|_b\}$. We want to show $L \subseteq L(G)$.

Let's prove by induction on $|w|$ that if $|w|_a > |w|_b$, then $S \Longrightarrow^* w$. If $|w| = 1$, then we must have $w = a$. Then we have the production rule $S \to a$.

Now assume that the result is true for $|w| \leq n$. We prove it for $|w| = n + 1$. There are a number of cases to consider:

Case 1. $w = by$ for some string y. Then we have $|w|_a - |w|_b \geq 1$, so $|y|_a - |y|_b \geq 2$. Looking at the productions of the grammar, this suggests trying to write $y = y_1y_2$, so that $|y_1|_a > |y_1|_b$ and $|y_2|_a > |y_2|_b$. To do this, define $D(x) = |x|_a - |x|_b$, and consider the prefixes y' of y. We have $D(\epsilon) = 0$ and $D(y) \geq 2$. As each new symbol is considered, D changes by at most 1. It follows that there must be a y_1 which is a proper prefix of y such that $D(y_1) = 1$. But then $D(y_2) \geq 1$. Clearly $|y_1|, |y_2| \leq n$. It follows by induction that $S \Longrightarrow^* y_1$ and $S \Longrightarrow^* y_2$. Hence $S \Longrightarrow bSS \Longrightarrow^* by_1y_2 = w$.

Case 2. $w = yb$ for some string y. This case is almost exactly the same as Case 1, except that we consider suffixes instead, and use the production $S \to SSb$.

Case 3. $w = aya$ for some string y. If y contains no b's, then we can easily derive y using the rules $S \to aS$, $S \to a$. Hence assume that y contains at least one b. Our goal is to write $w = xbz$, where $D(x) > 0$ and $D(z) > 0$. For $1 \leq i \leq r = |w|_b$ write $w = x_ibz_i$, where the b shown is the i'th one in w. If $D(x_r) > 0$, let $x = x_r$ and $z = z_r$. Since z_r contains no b's, and since w ends in an a, we must have $D(z_r) > 0$.

Otherwise, assume $D(x_r) \leq 0$ and let m be the smallest value of i for which $D(x_i) \leq 0$. The variable m is well-defined since m is at worst equal to r. Also, we know that $D(x_1) > 0$,

since w starts with a, and since x_1 contains no b's, it must contain at least one a. Hence $m \geq 2$, and $D(x_{m-1}) \geq 1$. Clearly $|x_m|_a \geq |x_{m-1}|_a$ and $|x_m|_b = |x_{m-1}|_b + 1$; subtracting these gives us $D(x_m) \geq D(x_{m-1}) - 1$. Hence $D(x_{m-1}) \leq D(x_m) + 1 \leq 1$. Since $D(x_{m-1}) \geq 1$, it follows that $D(x_{m-1}) = 1$. Hence $D(z_{m-1}) \geq 1$. Now take $x = x_{m-1}$ and $z = z_{m-1}$.

Now $|x|, |z| \leq n$, so by induction there is a derivation $S \Longrightarrow^* x$ and $S \Longrightarrow^* z$. It follows that $S \Longrightarrow SbS \Longrightarrow^* xbz = w$. Our proof is complete.