PDA and CFG Example

Shir Maimon

October 2018

We will show that the complement of the set \(\{xx \mid x \in \{a, b\}^*\}\) is context free. We will do this by giving a grammar for the language. The complement of the set can also be written as \(\{w \mid w\text{ is of odd length}\} \cup \{xy \mid x, y \in \{a, b\}^*, |x| = |y|, \text{ and } x \neq y\}\).

Note that for \(x, y\) where \(|x| = |y|\) and \(x, y \in \{a, b\}^*\), \(x \neq y\) is equivalent to there existing some index \(i\) such that the \(i\)th character of \(x\) is different from the \(i\)th character of \(y\), which is in turn equivalent to

\[
xy \in \Sigma_i^a \Sigma^j \Sigma^i \Sigma^j \cup \Sigma_i^b \Sigma^j \Sigma^i a \Sigma^j = \Sigma_i^a \Sigma^i \Sigma^j b \Sigma^j \cup \Sigma_i^b \Sigma^j \Sigma^i a \Sigma^j
\]

for some \(i, j\). Here \(\Sigma = \{a, b\}\). Now we can build our grammar.

\[
S \rightarrow S_1 | S_2
\]

\[
S_1 \rightarrow X | XX S_1
\]

\[
S_2 \rightarrow T_a T_b | T_b T_a
\]

\[
T_a \rightarrow XT_a X | a
\]

\[
T_b \rightarrow XT_b X | b
\]

\[
X \rightarrow a b
\]

Here \(S_1\) accepts strings of odd length. \(T_a\) produces \(\Sigma_i^a \Sigma^i\) for all \(i\), and \(T_b\) produces \(\Sigma_i^b \Sigma^j\) for all \(j\). Thus \(S_2\) produces \(\Sigma_i^a \Sigma^i \Sigma^j \cup \Sigma_i^b \Sigma^j \Sigma^i a \Sigma^j\) for all \(i, j\), so \(S\) produces \(\{w \mid w\text{ is of odd length}\} \cup \{xy \mid x, y \in \{a, b\}^*, |x| = |y|, \text{ and } x \neq y\}\), which is the complement of \(\{xx \mid x \in \{a, b\}^*\}\).

Note that for the complement of the set \(\{xcx \mid x \in \{a, b\}^*\}\), it is not sufficient to simply put a \(c\) between \(T_a\) and \(T_b\) in the \(S_2\) production. This will not accept things like \(abcbcb\).

The PDA is on the next page.
We will build the pushdown automata with the same logic. Note that I use $a|b$ in my PDA to read a character. This isn’t really allowed, but I just mean that the rule is the same if we are reading an $a$ or a $b$. You should not do this on your homeworks or tests.

First, we nondeterministically choose either odd length ($q_1$) or unequal first and second half ($q_3$). The former path should be clear.

For the bottom path, while looping on $q_3$, we first read in some number, say $i$ characters, and push $i$ X’s onto the stack. We then read either an $a$ or $b$ and transition to state $q_4$ or $q_7$ respectively. Suppose we read an $a$ and transition to $q_4$. Then we loop on $q_4$, reading characters and popping X’s off of the stack until there are none, meaning we have read another $i$ characters. Thus at this point, we have read $\Sigma^i a \Sigma^i$.

Once all of the $X$’s are popped, we can transition to $q_5$ (because we see the $S$ on the stack). Again, we read some number of characters, say $j$, and push $j$ $X$’s onto the stack. Now we read a $b$ to transition to $q_6$. While looping at $q_6$, we read characters and pop $X$’s until we reach the bottom of the stack, meaning we have read another $j$ characters. Thus in order to successfully reach $q_{10}$ and accept, we have to have read $\Sigma^i a \Sigma^j b \Sigma^j$.

Similarly, by using $q_7$, we accept strings of the form $\Sigma^i b \Sigma^j a \Sigma^j$. Based on our earlier description of the language, we can see why the PDA accepts the complement of the set $\{xx | x \in \{a, b\}^*\}$.