

Solutions for Problem Set 2
cs172

Allison Coates

Prove the following languages are or are not regular.

- 1a. With the alphabet given as $\Sigma = \{a\}$,

$$L = \{a^n \mid n = 2^j \text{ with } j \in \mathbb{N}\}$$

I can prove this isn't regular using the pumping lemma. I will assume this language is regular, and then use the lemma to guarantee the existence of a pumping length p such that all strings of length p or greater in the language can be pumped. Then, find a string in L of length p or greater that cannot be pumped.

The pumping lemma is:

Lemma 1 *If A is a regular language, then there is a number p where if s is any string in A of length at least p , then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:*

1. $\forall i \geq 0, xy^iz \in A$,
2. $|y| > 0$,
3. $|xy| \leq p$.

Proof by contradiction. Assume the language L defined by the above strings is regular. Let p be the pumping length. According to the pumping lemma, for any string of at least p , the pumping lemma holds. Choose s to be the string a^{2^k} where 2^k is the smallest power of 2 that is greater than p . s is a member of L . Since s is a member of L , it can be divided into three pieces xyz such that $\forall i, xy^iz \in L$.

Pick $y = a^m, 0 < m \leq p, m \neq 2^i$, in accordance with condition 2 and 3. Hence, m is not a power of 2, while the length of s is a power of 2. The length $|xyyz| = 2^k + m$. This cannot be a power of 2 unless m is a power of 2. But m is not a power of 2, so $xyyz$ is not in L . Contradiction.

Pick $y = a^m, 0 < m \leq p, m = 2^i$. In this case, either p is a power of 2, and $y = s$ (and x and z are empty), or the length of y is a power of 2, $2^j, j < i$. Consider the first case where p is a power of 2, and $y = s$. In this case, x and z are the empty strings. Then $|xyyyz| = |yyy| = m + m + m = 3(2^i)$ which is not a power of 2.

Consider the second case, where the length of y is $2^j, j < i$. Pumping xyz , we find the length $|xyyyyz| = 2^j + 2^j + 2^j + 2^k = 3(2^j) + 2^k$ which is not a power of 2. Thus contradiction is unavoidable if we assume L is regular, so L is not regular.

- b. With the alphabet given as $\Sigma = \{0, 1\}$,

$$L = \{n + 2 \mid n \text{ in binary and } n = 2^j \text{ with } j \in \mathbb{N}\}$$

The language L is regular. To prove it, I will show the regular expression for the above language. First, the numbers that this language represents are: 3, 4, 6, 10, 18, ... These strings in binary are:

$$11, 100, 110, 1010, 10010, \dots$$

For numbers > 4 , strings in binary of the form $2^j + 2$ all fit the following form: a 1, followed by some number of zeros, then a 1 and a 0. This set of strings can be expressed by the regular expression 10^*10 . We must now include the strings for the numbers 2 and 4, which do not fit the above form. The final regular expression is

$$10^*10 \cup 11 \cup 100$$

- c. With the alphabet given as $\Sigma = \{a, b\}$,

$$a^n b^m \mid n \neq m \text{ and } n, m \in \mathbb{N}\}$$

I can prove this isn't regular as follows:

Proof by contradiction. assume L is regular. Then its complement is regular, since regular languages are closed under complement. Further, the regular languages are closed under intersection, so intersection of \bar{L} and the strings 0^*1^* is a regular expression. But $\bar{L} \cap 0^*1^* = \{0^n 1^m \mid n \neq m, n, m \geq 0\}$. We know that this language is not regular. Contradiction. Hence, L is not regular.

2.6 Given CFGs generating the following languages.

- a. The set of strings over the alphabet $\{a, b\}$ with twice as many a's as b's.

The base case consists of ϵ , the empty string, and the strings aab, aba, baa . so our production rules must allow that:

$$\begin{aligned} R &\rightarrow \epsilon \\ R &\rightarrow aab \mid aba \mid baa \end{aligned}$$

But this problem isn't as pretty as the palindrome problem: assuming w is an element of the language, we can't easily write down production rules that leave w alone, and simply add new elements onto w . As an example, look at this: $bbbaaaaa$. No amount of concatenating elements of the form aab, aba, baa would form that string. Continuing to look for a "recursive" step, notice that given a string w is in the language, crossing out any 2 a's and 1 b produces a string that is still in the language. In the other direction, this means we can add 2 a's and 1 b in any set of positions. The rule is:

$$R \rightarrow \epsilon \mid RaRaRbR \mid RaRbRaR \mid RbRaRaR$$

- b. The complement of the language $\{a^n b^n \mid n \geq 0\}$.

This seems difficult since there are so many possible strings. Let's look at it this way: we want $L = \Sigma - \{a^n b^n\}$. We know that this set includes $\{a^i b^j, i > j\}$ and $\{a^i b^j, i < j\}$. But it contains other strings as well—including any string not of the form $a^i b^j$. But this is enlightening. For a string not to be of the form $a^i b^j$, such a string contain the substring ba . We can write the set of all such strings as: $(a \cup b)^*(ba)(a \cup b)^*$.

Turning the first form into a set of production rules, we have:

$$\begin{aligned} S &\rightarrow A \\ A &\rightarrow a \mid aA \mid aAb \end{aligned}$$

Note that there are no ϵ s in this rule: so it must stop when there are more a's than b's. Similarly,

$$\begin{aligned} S &\rightarrow B \\ B &\rightarrow b \mid Bb \mid aBb \end{aligned}$$

corresponds to more bs than as.

The final form is encoded as:

$$\begin{aligned} S &\rightarrow C \\ C &\rightarrow ba \mid aC \mid bC \mid Ca \mid Cb \end{aligned}$$

Simplifying the above, we have

$$\begin{aligned} S &\rightarrow A|B|C \\ A &\rightarrow a|aA|aAb \\ B &\rightarrow b|Bb|aBb \\ C &\rightarrow ba|aC|bC|Ca|Cb \end{aligned}$$

- c. $\{w\#x|w^R \text{ is a substring of } x \text{ for } w, x \in \{0,1\}^*\}$.

Here, the point is to build up a palindromic string, but with other elements of $\{0,1\}^*$ in the middle and on the end.

$$\begin{aligned} S &\rightarrow TU \\ T &\rightarrow aTa|bTb|\#U \\ U &\rightarrow aU|bU|\epsilon \end{aligned}$$

- d. $\{x_1\#x_2\#x_3\#\dots\#x_k | k \geq 2, \text{ each } x_i \in a, b^*, \text{ and for some } i, j, x_i = x_j^R\}$.

It's a little difficult to figure out how to count through the $\#$ characters. the rule for T takes care of the palindromic portion of the strings. V takes care of the middle portions, which may consist of many strings. Notice that $\#$ symbols may occur next to each other, but if there is only 1 x_i there are no $\#$ symbols.

$$\begin{aligned} S &\rightarrow U_1TU_2 \\ T &\rightarrow aTa|bTb|\#V\#|\epsilon \\ V &\rightarrow aV|bV|\#V|\epsilon \\ U_1 &\rightarrow U_1\#|U_1a|U_1b|\epsilon \\ U_2 &\rightarrow \#U_2|aU_1|bU_1|\epsilon \end{aligned}$$

2.14 Convert the following CFG into an equivalent CFG in chomsky normal form using the procedure given in Theorem 2.6.

$$\begin{aligned} A &\rightarrow BAB|B|\epsilon \\ B &\rightarrow 00|\epsilon \end{aligned}$$

Chomsky normal form has rules of only the following two forms:

$$\begin{aligned} A &\rightarrow BC \\ A &\rightarrow a \end{aligned}$$

First, we add a new start symbol, S_0 , and a new rule $S_0 \rightarrow A$ so that the start symbol doesn't occur on the right side of any production rules.

we now have:

$$\begin{aligned} S_0 &\rightarrow A \\ A &\rightarrow BAB|B|\epsilon \\ B &\rightarrow 00\epsilon \end{aligned}$$

Next, we address the ϵ symbols. so, we rewrite the rule for B as $B \rightarrow 00$, and since A goes to ϵ already, we do not change A to account for our change to B . Then, we change A change the production rule of A accordingly.

$A \rightarrow BAB | AB | BA | A | \epsilon$. Then, we remove the ϵ from rule A by allowing our new start to have a production rule to ϵ . Now, our rules are:

$$\begin{aligned} S_0 &\rightarrow A | \epsilon \\ A &\rightarrow BAB | AB | BA \\ B &\rightarrow 00 \end{aligned}$$

Continuing, we remove the unit rule $A \rightarrow A$. We also remove the unit rule $S_0 \rightarrow A$.

$$\begin{aligned} S_0 &\rightarrow BAB | AB | BA \\ A &\rightarrow BAB | AB | BA \\ B &\rightarrow 00 \end{aligned}$$

Then, we add rules to adhere to the form of two variables on the right side.

$$\begin{aligned} S_0 &\rightarrow BA_1 | AB | BA \\ A &\rightarrow BA_1 | AB | BA \\ A_1 &\rightarrow AB \\ B &\rightarrow 00 \end{aligned}$$

Finally, we add the rule for one terminal symbol. The final rules are:

$$\begin{aligned} S_0 &\rightarrow BA_1 | AB | BA \\ A &\rightarrow BA_1 | AB | BA \\ A_1 &\rightarrow AB \\ B &\rightarrow O_1O_1 \\ O_1 &\rightarrow 0 \end{aligned}$$