COMS W3261: Computer Science Theory, Fall 2024
Yihan Shen, Hellen Zhao

Handout 4A: Proving Non-Regularity

Overview

On our homeworks and exams, we may be presented with some language L and be asked to prove whether L is regular or non-regular. What can make this challenging is that not only do we have to write a proof, we also have to figure out which claim (regular or non-regular) to even prove in the first place!

When approaching such a problem, it can be worthwhile to try to identify some intuitive clues that may point us in the right direction. Let's think about what we might notice when L is non-regular rather than regular. For one, L must be infinite because we've shown that finite languages are regular. Another heuristic that may be helpful is to think about what information the DFA (if it were to exist) must keep track of. Take for example the language

$$L = \{w \in \{0,1\}^* | L \text{ has an equal number of 0's and 1's}\}.$$

For each inputted character, we need to count how many 0's and 1's there are to determine if a string is in L. But it's impossible to keep track of this in finite states, so it's likely that L is non-regular (turns out it is!)

Note that saying this does NOT suffice to prove L is non-regular. It merely gives us some intuition, which can help us decide on the direction for our proof. To formally prove that L is non-regular, we can utilize closure properties, the Pumping Lemma, or the Myhill-Nerode Theorem (optional material). In the following sections, we'll provide some notes and tips on these proof strategies.

Clsoure Properties

So far, we've shown in class that regular languages are closed under three operations: complement, union, and intersection. In other words, given two REGULAR languages L_1 and L_2, the following languages are regular:

1. complement: $\overline{L_1} = \{w \in \Sigma^* | w \notin L_1\}$ and $\overline{L_2}$
2. union: $L_1 \cup L_2 = \{w \in \Sigma^* | w \in L_1 \text{ OR } w \in L_2\}$
3. intersection: $L_1 \cap L_2 = \{w \in \Sigma^* | w \in L_1 \text{ AND } w \in L_2\}$

To prove a language L is not regular using closure properties, we can do the following:

Proof Template: Using Closure Properties

1. Assume for the sake of contradiction that L is regular.
2. Consider the regular languages ______________. We know these languages are regular because we proved so in $\left[\text{class/homework/above/other}\right]$.
3. Let $L' = \underline{\text{ (non-regular language) }}$. L' is the result of the expression ______________, which applies complement, union, and intersection on L and the other regular languages from above. L' must be regular because regular languages are closed under complement, union, and intersection.
4. But we already proved in $\left[\text{class/homework/above/other}\right]$ that L' is non-regular, which is a contradiction. Thus L is not regular. $\square$

Note: as we prove closure of regular langauges under more operations, you could use those above, in addition to complement, union, and intersection.

Pumping Lemma

Another way to prove a language is non-regular is by using the Pumping Lemma, which we state below.

The Pumping Lemma

If L is a regular language, then there exists a positive integer p (known as the pumping length) where, for any string $w \in L$ of length $|w| \geq p$, w can be divided into three pieces, $w = xyz$, satisfying the following conditions:

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

So the pumping lemma is a special property that all regular languages must satisfy. This makes the pumping lemma a useful tool for proving that languages are not regular, as any language for which the pumping lemma is false can not be regular. Think of it as using the pumping lemma's contrapositive:

$$\begin{aligned} L \text{ is regular } &\implies L \text{ does satisfy the pumping properties} \\ L \text{ is NOT regular } &\impliedby L \text{ does NOT satisfy the pumping properties} \end{aligned}$$

Keep in mind that this bottom statement only works in one direction! All we know is that IF a language does not satisfy the pumping lemma, THEN it must be non-regular. But if a language does satisfy the pumping lemma, we can not assume it is regular, as some non-regular languages also satisfy the conditions in the pumping lemma. We'll see an example of this in the practice problems provided at the end!

Proof Template: Using Pumping Lemma

1. Assume for the sake of contradiction that L is regular. Let p be the pumping number, which exists by the pumping lemma.
2. Choose the string $w =$ ______________. Verify that $w \in L$ and $|w| \geq p$.
3. Consider any partition of w into three parts $w = xyz$, with $|y| > 0$ and $|xy| \leq p$. Choose some value $i =$ ______________, then show that the string $xy^iz \notin L$ because ______________.
4. The pumping lemma does not hold for this w. We arrive at a contradiction, and thus L is not regular.

The trick here is to make a good choice of $w \in L$ with $|w| \geq p$. Since the pumping lemma tells us that the pumping property is true for SOME valid partition $w = xyz$, to show the pumping lemma does not hold we must show that the pumping property is false for ALL valid partitions $w = xyz$.

So we cannot choose the way we partition w into x, y, and z. But we do know that the possible partitions of w are subject to some constraints, namely that $|y| > 0$ and $|xy| \leq p$. Sometimes it can be helpful use this information to choose a w that "forces" y to be a certain way.

An example of how to make this choice of w is shown in the following proof that L is not regular, where

$$L = \{0^n 1^n \mid n \in \mathbb{N}\}.$$

Example

Assume for the sake of contradiction that L is regular. Then there exists a pumping length p for L.

To derive a contradiction, we want to come up with a string w and number i that acts as a counterexample to the pumping lemma. That is, we want to choose a w and i so that, for any partition of $w = xyz$ satisfying the first two parts of the pumping lemma, $xy^iz \notin L$ for our chosen i.

Consider the string $w = 0^p 1^p \in L$ which has length $|w| = 2p \geq p$. By our initial assumption, the properties of the pumping lemma should be true for w.

Now consider any partition of w into three parts $w = xyz$ with $|y| > 0$ and $|xy| \leq p$. Since $|xy| \leq p$ and the first p characters of w are 0, it must be that $y = 0^k$ for some $0 < k \leq p$. But notice that when $i = 2$, the string xy^iz is not in L because $xy^2z = 0^{p+k}1^p$ and $p+k \neq p$.

So the pumping lemma does not hold in this case, which is a contradiction. Therefore, L is not regular.

Oftentimes the hardest part of the proof is choosing a suitable w! In this proof, by choosing a w where the first p characters are all the same, we can utilize the condition that $|xy| \leq p$ to "force" our y to consist of only the character 0.

Myhill Nerode Theorem

Let L be a language over Σ.

Definition: Pairwise Distinguishable

The two strings $x, y \in \Sigma^*$ are **pairwise distinguishable by L** if there exists some string $z \in \Sigma^*$ such that precisely one of xz and yz is in L. We call such a z a distinguishing extension of x and y.

Otherwise, x and y are **indistinguishable by L**, meaning for all strings z, xz and yz are both in L or both not in L. In other words, x and y have no distinguishing extension.

This "indistinguishable-ness" is an equivalence relation, which we denote as $x \sim_L y$. (Recall that an equivalence relation is reflexive, symmetric, and transitive. Can you prove this for $\sim_L$?)

All strings, including those in L and not in L, belong to one and only one equivalence class, so the set of all possible strings is partitioned under this equivalence relation $\sim_L$.

Depending on the language L, there could be an infinite number of strings that are distinguishable by L, and thus an infinite number of equivalence classes induced by $\sim_L$. The Myhill-Nerode Theorem tells us that in this case L is not regular, while When there is a finite number of equivalence classes it is regular.

Myhill-Nerode Theorem

L is regular if and only if $\sim_L$ has finitely many equivalence classes. Moreover, if L is regular, the number of equivalence classes of $\sim_L$ is the number of states in the minimal DFA recognizing L.

The proof for this theorem, along with additional examples, are in the Myhill-Nerode Theorem handout (handout 3, see class webpage).

The nice thing about the Myhill-Nerode theorem is that it provides a necessary AND sufficient condition for regular languages. So showing there are finite equivalence classes proves that L is regular, and showing there are infinitely many equivalence classes proves L is not regular. In contrast, the Pumping Lemma only provides a necessary (but not sufficient) condition for regular languages, which is why we only use it to prove the non-regular case.

With that said, to prove a language is non-regular using the Myhill-Nerode Theorem, we can follow the template below:

Proof Template: Using Myhill-Nerode Theorem

1. Consider the set of strings $S =$ ______________, which contains infinitely many strings.
2. Let x and y be any two strings from the set above. The string $z =$ ______________ is a distinguishing extension of x and y because ______________.
3. Since every pair of strings in S is distinguishable by L, each string in S belongs to a different equivalence class under the relation $\sim_L$. Since S has infinitely many strings, there must be infinitely many equivalence classes. By the Myhill-Nerode Theorem, this implies L is not regular. $\square$

Practice Problems

Prove the following languages are not regular using closure properties, the Pumping Lemma, or the Myhill-Nerode Theorem.

1. $L_1 = \{0^n 1^m | n \neq m\}$
2. $L_2 = \{ww | w \in \{a,b\}^*\}$
3. $L_3 = \{1^{2^n} | n \geq 0\}$
4. For the alphabet $\{a, b, c\}$, define

$$L_4 = \{c^n w | n \geq 0, w \in \{a,b\}^*, \text{and if } n \text{ is odd then } w = w^R\}$$