COMS W3261: Computer Science Theory.
Instructor: Tal Malkin
Date: September 2024

Handout 3: The Myhill-Nerode Theorem and Implications

1 Overview

The Myhill-Nerode theorem is a fundamental result in the theory of regular languages. It provides a characterization of regular languages, and hence can be used to prove whether or not a language L is regular. It can also be used to find the minimal number of states in a DFA which recognizes L if L is regular. In fact, this theorem is at the heart of efficient DFA minimization algorithms. Here we present the Myhill-Nerode theorem, and give some examples applying it to both regular and non-regular languages.

This handout was written by Tal Malkin, based in part on material written by 2017 TA Michael Tong, 2024 TAs Hans Shen and Hellen Zhao, and lecture notes by Luca Trevisan. The Sipser textbook has the Myhill-Nerode theorem as part of its solved exercise 1.52 (with the equivalence relation defined in problem 1.51). The textbook by Hopcroft, Motwani, Ullman has a more extensive treatment in section 4.4.

This handout is recommended reading for interested students, and you may use the Myhill-Nerode theorem to solve problems on homeworks and exams, if you want to. However, it is not part of the required material for class this semester, and you will not be assumed to know this theorem.

2 The Myhill-Nerode Theorem

A key concept underlying the Myhill-Nerode theorem is the distinguishing extension.

Definition 2.1. Let $L \subseteq \Sigma^*$ be any language over the alphabet Σ. For $x, y \in \Sigma^*$ we call $z \in \Sigma^*$ a *distinguishing extension* of x and y if exactly one of xz and yz are in L (where xz is the concatenation of x and z). If such a z exists, we say that x and y are *distinguishable by* L; otherwise we say x and y are *indistinguishable by* L.

The intuition behind this definition is that if two strings x, y are distinguishable by L, then in any possible DFA for L (if there is one), the computation of the DFA on x and on y must end up in two different states. This is because if x and y reach the same state in the DFA, then for any extension z, xz and yz will reach the same state. This is the key idea that allows to prove the following lemma.

Lemma 2.2. *Let $L \subseteq \Sigma^*$ be any language over the alphabet Σ. Suppose there is a set with k distinct strings $\{x_1, \ldots, x_k\}$ such that for any $i \neq j$, x_i and x_j are distinguishable by L. Then there does not exist a DFA with fewer than k states that recognizes L.*

It is not hard to check that for any language L, indistinguishability-by-L is an equivalence relation:

Proposition 2.3. *Let $L \subseteq \Sigma^*$ be a language. Define $\sim_L$ to be the relation on Σ^* where $x \sim_L y$ iff x and y are indistinguishable by L. Then $\sim_L$ is reflexive, symmetric, and transitive so that $\sim_L$ is an equivalence relation on Σ^*.*

Now recall that an equivalence relation $\sim$ on a set S induces *equivalence classes* which partition S into subsets of elements related to each other by $\sim$. For example, let $\sim$ be a relation on the set of integers (denoted $\mathbb{Z}$) defined by $a \sim b$ iff $a \equiv b \pmod 3$. Then $\sim$ is an equivalence relation and it partitions $\mathbb{Z}$ into three equivalence classes $[0], [1], [2]$ defined by $[i] = \{n \in \mathbb{Z} \mid n \equiv i \pmod 3\}$.

Remark 2.4. If $x \in L$ and $y \notin L$, then x and y are distinguishable by L: we can take the distinguishing extension to be the empty string ϵ. Thus, each equivalence class will contain strings which are either all in L or all not in L.

We are now ready to state the Myhill-Nerode theorem.

Theorem 2.5. *Let $L \subseteq \Sigma^*$ be a language. Then L is regular if and only if the number of equivalence classes of $\sim_L$ is finite. Furthermore, if L is regular then the number of equivalence classes of $\sim_L$ is also the number of states in the minimal DFA recognizing L.*

Proof. We provide a sketch of the proof here. We need to prove the following three statmenets:

1. If L is regular, then $\sim_L$ has finitely many equivalence classes.
2. If $\sim_L$ has finitely many equivalence classes, then L is regular.
3. If $\sim_L$ has finitely many equivalence classes, then the number of equivalence classes is the number of states in the minimal DFA recognizing L.

Proof of 1: We will prove the contrapositive. Suppose $\sim_L$ has infinitely many equivalence classes. That means that there is a set of infinitely many strings (one from each equivalence class), such that every two of them are distinguishable by L. Now by Lemma 2.2, for any number k we take, there cannot be a DFA with k states recognizing L. Thus, there's no DFA (with any finite number of states) recognizing L, meaning L is not regular.

Proof of 2: If $\sim_L$ has finitely many equivalence classes, we will prove that L is regular by constructing a DFA (and this DFA will also be used to prove part 3).

As a reminder, for a string x the notation $[x]$ refers to the equivalence class containing x, that is $[x]$ is the set of all strings y so that $x \sim_L y$. So if $x \sim_L y$, then $[x]$ and $[y]$ are equal and denote the same equivalence class.

An important observation is that when $x \sim_L y$, for any string w we have $[xw] = [yw]$. This is because if x and y are indistinguishable, then for all strings $z \in \Sigma^*$, $xz \in L$ if and only if $yz \in L$. This includes strings of the form $z = wu$, where u is any string. Since for any string u, $xwu \in L$ if and only if $ywu \in L$, we know $xw \sim_L yw$, so $[xw] = [yw]$.

With this in mind, we construct a DFA $(Q, \Sigma, \delta, q_0, F)$ for L as follows.

Q We will have one state corresponding to each equivalence class $[x]$ of $\sim_L$.

q_0 The start state is $[\epsilon]$, the equivalence class containing the empty string.

F As mentioned in Remark 2.4, every equivalence class contains strings that are either all in L or all not in L. We define F to to be the set of states that correspond to equivalence classes $[x]$ for $x \in L$.

δ For a state/equivalence class $[x]$ and character $a \in \Sigma$, let the DFA transition from $[x]$ to $[xa]$, that is $\delta([x], a) = [xa]$. We need to check that this function is well-defined (if $[x] = [x']$, we don't want $\delta([x], a) = [xa]$ and $\delta([x'], a) = [x'a]$ to be two different outputs). But indeed, as mentioned earlier, if $[x] = [x']$, then $[xz] = [x'z]$ for all strings z, including the character a.

Now that we've given a valid definition of a DFA, our final step is to show that this DFA does indeed recognize L. Let $x = x_1 x_2 \dots x_n$ be any string. Inputting this into the DFA, our DFA will start at $[\epsilon]$, then move to state $[x_1]$, then to $[x_1 x_2]$, and so on until it ends in state $[x_1 \dots x_n]$. By construction, $[x_1 \dots x_n] = [x]$ is an accepting state if and only if $x \in L$. Our DFA indeed recognizes L, so L is regular.

Proof of 3: Above we constructed a DFA for L that has the same number of states as the equivalence classes of $\sim_L$. Furthermore, by Lemma 2.2, there cannot be any DFA for L with fewer states than the number of equivalence classes. Thus, the DFA constructed above is a minimal DFA for L. □

Using the Myhill-Nerode Theorem. The Myhill-Nerode theorem is useful for regular languages, allowing us to break the language into its most essential pieces, namely the equivalence classes, which correspond to the minimal DFA for the language. The ideas discussed above are also at the heart of efficient DFA minimization algorithms (which we are not presenting here). The key idea is that, as mentioned above, in any DFA for a language, all strings that end up in the same state, must be in the same equivalence class. On the other hand, strings that end up in two different states may still be in the same equivalence class. In this case, those two states are said to be equivalent, and can be combined into one state without changing the behavior of the DFA.

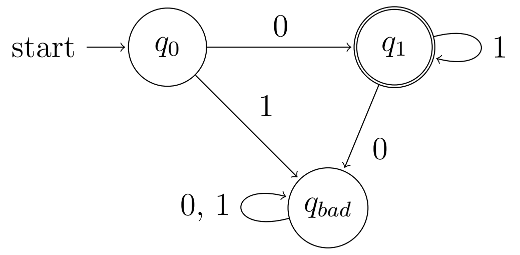
The Myhill-Nerode theorem is also very useful in order to prove that a language L is not regular. To do so, we need to show that the number of equivalence classes is infinite. This can be done by coming up with infinitely many strings such that no two of them are equivalent (i.e., every pair of them is distinguishable by L).

3 Examples

3.1 Regular Languages

Example 3.1. Let $L = \{01^n \mid n \geq 0\}$. What are the equivalence classes? First notice that ε is distinguishable from all other strings in $\{0,1\}^*$, so it's in its own equivalence class. Indeed, if w is a non-empty binary string then $w0$ is not in L but $\epsilon 0 = 0$ is. Another equivalence class consists of strings which are not in the right "format", since then no matter what extension z is added the string would still not be in the language. This would include strings which start with 1 and strings which contain more than one 0. A third equivalence class consists of the strings in L (do you see why for this language all strings in L are in the same equivalence class?). These are the only 3 equivalence classes, since from the definition of L, every string is either in L, or is empty, or starts with 1 or has more than one 0. Since there are 3 equivalence classes we know that L is regular, and has a minimal DFA with 3 states

On the other hand, consider the following minimal DFA:



It is now clearer exactly how the equivalence classes relate to states: each equivalence class simply consists of the strings which cause the DFA to be in the same state. Indeed, the initial state has no arrows going to it, and thus only corresponds to the string ϵ. The state q_{bad} refers to the strings with "improper format," which the DFA agrees are those strings which start with 1 or have more than one 0. And finally the state q_1 refers to the strings in the language. Indeed, the fact that L itself is contained entirely in one equivalence classes corresponds to the fact that the minimal DFA has only one accepting state.

Example 3.2. For any $n \in \mathbb{N}$, define the language

$$L_n = \{w \in \{0,1\}^* \mid \text{ the } n\text{-th to last symbol in } w \text{ is } 1\}$$

We show that for any n, the language L_n is regular, and the minimal DFA recognizing it has 2^n states.

There are 2^n strings of length n. We will prove that every pair of them has a distinguishing extension, thus each one is in a different equivalence class with respect to $\sim_{L_n}$. Consider any two strings $w = w_1 \dots w_n \in \{0,1\}^n$ and $w' = w'_1 \dots w'_n \in \{0,1\}^n$ where $w \neq w'$. Since $w \neq w'$, they must differ in at least one location. Take some $i \in \{1, \dots, n\}$ such that $w_i \neq w'_i$, which means one of w_i, w'_i is 0 and the other one is 1. Choose an extension z of length $i-1$, e.g., take $z = 0^{i-1}$. The n-th to last bit in wz is w_i, and the n-th to last bit in $w'z$ is w'_i, one of which is 0 and one of which is 1. Thus, one of $wz, w'z$ is in L_n and the other one is not, so w, w' are distinguishable.

We have proved that L_n has at least 2^n equivalence classes (one for each string of length n). We now claim that L_n has exactly 2^n equivalence classes. Indeed, it is not hard to see that every string w of length $|w| > n$ is equivalent to its n-bit suffix, and every string w of length $|w| < n$ is equivalent to the n-bit string $0^{n-|w|}w$. Thus, the minimal DFA for L_n has exactly 2^n states.

We note that in contrast, for any n there is an NFA with $n+1$ states for L_n.

3.2 Non-Regular Languages

Below are two examples we have proven in class are not regular, using the pumping lemma. Here we reprove them using the Myhill-Nerode theorem. In the next handout we will see examples of non-regular languages where using the pumping lemma to prove it is hard or impossible, but using the Myhill-Nerode theorem works.

Example 3.3. Let $L = \{w \in \{0,1\}^* \mid \text{the number of 0s in } w \text{ is the same as the number of 1s in } w\}$. We will show that L has infinitely many equivalence classes, and thus is not regular. Consider all strings of the form 0^k, for any $k \geq 0$. These are infinitely many strings, and we claim that no two of them are equivalent. Indeed, for any $k \neq j$, consider the strings $x = 0^k$ and $y = 0^j$. Choosing the extension $z = 1^k$ gives $xz = 0^k1^k$ which is in L, and $yz = 0^j1^k$ which is not in L. Thus, each equivalence class of L can contain at most one string of the form 0^k, so there must be infinitely many equivalence classes. So L is not regular.

Example 3.4. Let $L = \{w \in \{a,b\}^n \mid w \text{ is a palindrome}\}$. We will prove that this language is not regular, by showing that it has infinitely many equivalence classes. Again, we can do this by coming up with an example of infinitely many strings so that no two are equivalent. We consider all strings of the form a^k for $k \geq 0$, and show that no two of them are equivalent. Take $x = a^k, y = a^j$ for any $k \neq j$. Choosing the extension $z = ba^k$ gives $xz = a^kba^k \in L$, and $yz = a^jba^k \notin L$. Therefore, all such pairs a^k, a^j are distinguishable, and we have infinitely many equivalence classes, so L is not regular.

Note: there are many other possible infinite sets of strings we could use where every pair is distinguishable. In fact, in this example one could prove that every string is in its own equivalence class, namely no two strings are equivalent. But the proof we gave above is simpler, and suffices to show L is not regular.