COMS W3261: Computer Science Theory.
Instructor: Tal Malkin
Date: October 7, 2022

Handout 4: 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. We also use this theorem to prove that there are languages for which the minimal DFA for a language has exponentially more states than the minimal NFA (thus showing that the exponential blowup in the subset construction is inherent – there is no general transformation of NFAs to DFAs which is more efficient).

This handout was written by Tal Malkin, based in part on a handout made by a 2017 TA for the class, Michael Tong. 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 Statement of the Myhill-Nerode Theorem

The key concept to 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 the following: suppose L is regular and is recognized by a DFA D. For any string s, Let $Q_D(s)$ be the state D is in after reading s. Then if $Q_D(x) = Q_D(y)$, then for any string z we will have $Q_D(xz) = Q_D(yz)$, so that D accepts either both xz and yz or none of them. Thus x and y are indistinguishable by L.

On the other hand, if $Q_D(x) \neq Q_D(y)$, then it's still possible that x and y are indistinguishable by L. In this case, we say that the states $Q_D(x)$ and $Q_D(y)$ are *equivalent*, and the two states can be combined into one state without changing the behavior of the DFA. Similarly, if x and y are in fact distinguishable by L, then $Q_D(x)$ and $Q_D(y)$ will not be equivalent to each other.

Proposition 2.2. *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 2.3. 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, the equivalence classes will contain strings which are either all in L or all not in L.

Remark 2.4. If L is regular and recognized by a DFA D, then x and y are in the same equivalence class of $\sim_L$ iff the states $Q_D(x)$ and $Q_D(y)$ are equivalent. Thus, if a DFA D has no states which are equivalent to each other, then x and y are in the same equivalence class iff $Q_D(x) = Q_D(y)$ and it can be shown that D has the least amount of states possible. So let L be regular and let D be its minimal DFA with states $\{q_0, q_1, \ldots, q_n\}$. We can then systematically define the equivalence classes of $\sim_L$ to be the sets $\langle i \rangle = \{w \in \Sigma^* \mid Q_D(w) = q_i\}$. Notice that this realization actually gives the above remark as a corollary.

We are now ready to state the 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.*

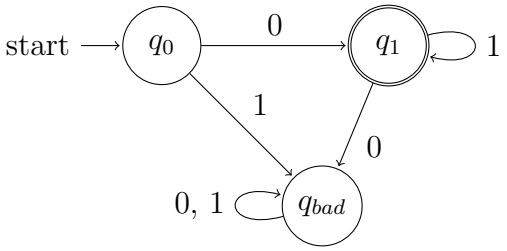
Thus, when talking about a regular language L, we can break it into its most essential pieces, namely the equivalence classes of $\sim_L$, which correspond to the minimal DFA for L.

On the other hand, to show that a language L is not regular, 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

Example 3.1. Let $L = L(01^*)$. 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. Let $L = \{w \in \{0,1\}^* \mid$ the number of 0s in w 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.

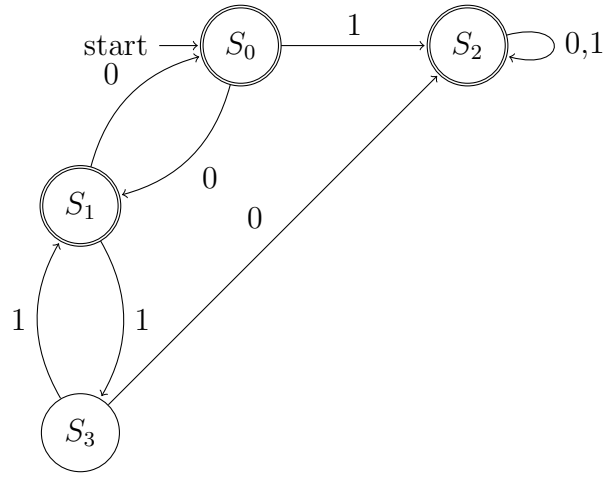
Example 3.3. Let $L = \{w \in \{a,b\}^n \mid w$ is a palindrome$\}$. We will prove that this language is not regular, by showing that it has infinitely many equivalence classes. Again, we can do this by coming up with an example of infinitely many strings so that no two are equivalent. We consider all 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.

Example 3.4. Returning to an example we saw in class and in handout 2, consider the following language over alphabet $\Sigma = \{0,1\}$:

$$L = \{xy \mid x \text{ has even number of } 0's, y \text{ has an even number of } 1's\}$$

In this example, finding the equivalence classes of L directly is harder than coming up with a DFA as we did in handout 2. However, we will use the Myhill-Nerode theorem to show that the DFA we obtained is minimal. Recall (see handout 2) that we constructed an NFA for the language, then used the subset construction to get a DFA with 5 states. We then noticed that S_2 and S_4 can be collapsed – this was because these states were equivalent under $\sim_L$. We thus obtained the following DFA:



Since this is a DFA for the language, we know that all strings whose computation ends in the same state in the DFA are in the same equivalence class. To show that this DFA is minimal, we need to show that no two states in the DFA are equivalent. We do this by showing a distinguishing extension for every pair, namely a string that leads from one of them to an accepting state, and from the other one to a non-accepting state.

- For S_0, S_2 take 01
- For S_0, S_1 take 1.
- For S_1, S_2 take 1.
- For S_3 and any other state, take ε.

This means that this 4-state DFA is the minimal DFA for the language.

As an aside, if we wanted to fully specify what the equivalence classes for the language are, we can prove that they are the following:

- Strings that have a prefix with odd 1s and even 0s (state S_2)
- Strings without the above prefix with even 0s and even 1s (state S_0).
- Strings without the above prefix with odd 0s and even 1s (state S_1).
- Strings without the above prefix with odd 0s and odd 1s (state S_3).

4 DFAs can be exponentially larger than NFA

We now prove a claim we mentioned without proof in class, namely that there are languages for which the number of states in the minimal DFA is exponentially larger than the number of states in an NFA for the same language.

Definition 4.1. For an arbitrary $n \in \mathbb{N}$, define the language

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

Proposition 4.2. *For all n, there is an NFA with $n+1$ states recognizing L_n.*

Theorem 4.3. *For any $n \in \mathbb{N}$, any DFA recognizing the language L_n must have at least 2^n states.*

Proof. 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. Since every n-bit string must be in a different equivalence class, and there are 2^n such strings, L_n has at least 2^n equivalence classes, so any DFA for L_n must have at least 2^n states. □

Remark 4.4. We note that L_n has exactly 2^n equivalence classes, namely 2^n states are also sufficient. This is because we can show 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 2^n states.

Remark 4.5. The above proves that exponential blowup is inherent in any general transformation of NFA to equivalent DFA. However, it does not mean that for *every* language the smallest DFA must be larger than the smallest NFA (we saw several examples where this was not the case; one simple example is the one-state DFA for the language Σ^*, which clearly cannot be improved even if we allow an NFA).