COMS W3261: Computer Science Theory.
Instructors: Tal Malkin and William Pires
Date: September 2026

Lecture Notes: The Myhill-Nerode Theorem

1 Introduction

Bibliographic Notes. These are lecture notes accompanying Columbia's COMS W3261 Fall 2026 lecture 4 and the beginning of lecture 5. We are including them since this material is not directly covered in our textbook.[1] The lectures are based on previous handouts by Prof. Malkin and various TAs, most notably Michael Tong, in previous offerings of the class.

We note that the textbook uses *the pumping lemma* (Chapter 1.4) as its main tool for proving that a language is not regular. We will not cover this in class, and you are not responsible for that material. Interested students are welcome to read it, and the teaching staff will be happy to answer questions about the pumping lemma, but for class assignments and quizzes you cannot use theorems that were not covered in class (so you cannot use the pumping lemma). We chose to teach the Myhill-Nerode theorem instead of the pumping lemma for a couple of reasons. First, unlike the pumping lemma, Myhill-Nerode provides a *characterization* of regular languages: the theorem says that a language is regular *if and only if* some condition holds. Second, in many cases, proving a language is not regular is easier and cleaner with the MN theorem (for example, for non-regular languages that do satisfy the pumping lemma). It should also be easier to grade MN non-regularity proofs, since there are fewer moving parts, and (hopefully!) less room for mistakes.[2]

Overview of the Myhill-Nerode Theorem. The Myhill-Nerode (MN) theorem is a fundamental result in the theory of regular languages. We briefly summarize what it says and why it's useful, and provide details in the next section.

For any language L, we can define an equivalence relation $\sim_L$, which partitions the set of all strings in Σ^* into equivalence classes. The MN theorem tells us that the number of equivalence classes is a *lower bound* on the number of states in any DFA recognizing L (i.e., any such DFA must have at least one separate state per equivalence class). The MN theorem further tells us that, if L is regular, the number of equivalence classes is also an *upper bound* on the minimum number of states: we can construct a DFA recognizing L with exactly one state per equivalence class.

The latter part is at the heart of efficient DFA minimization algorithms (given any DFA, there's an efficient way to construct a minimal DFA for the same language), but we will not cover this in class. Our focus is on the first part, and in particular its use in proving that a language L is not regular: to show L is not regular, show that $\sim_L$ has infinitely many equivalence classes.

[1] The Sipser textbook has the MN 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.

[2] We note that understanding the pumping lemma for regular languages is helpful towards later understanding the tandem pumping lemma for context-free languages, giving a way to prove that a language is not context free. But since we will not be covering context free languages in the class this semester, this is not relevant for us.

2 The Myhill-Nerode Theorem

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 wrt L if exactly one of xz and yz is 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 and denote it by $x \sim_L y$.

Lemma 2.2. *Let $L \subseteq \Sigma^*$ be a regular language, and let $D = (Q, \Sigma, \delta, q_0, F)$ be a DFA recognizing L. For any $x, y \in \Sigma^*$, if x, y are distinguishable by L then the computations of D on x and on y must end in two different states of D.*

Proof. Intuitively, it is clear that if the computations of D on x and y end in the same state, then no matter which extension z we add, the computations of D on xz and on yz will also end in the same state. But if there's a distinguishing extension z, we need the computation of D on one of xz, yz to end in an accepting state, and the computation on the other one to end in a rejecting state. We provide a more formal proof below.

Let x, y be strings that are distinguishable by L, and let z be a distinguishing extension, namely exactly one of xz, yz is in L (and the other one isn't). Let the computation of D on x be $(r_0, \ldots, r_{|x|})$ and the computation of D on y be $(r'_0, \ldots, r'_{|y|})$. By definition of computation, this means that $r_0 = r'_0 = q_0$, and that $r_i = \delta(r_{i-1}, x_i)$ for all $1 \leq i \leq |x|$, and $r'_i = \delta(r'_{i-1}, y_i)$ for all $1 \leq i \leq |y|$. Suppose towards contradiction that they end in the same state, and let's call that state r^*_0, namely $r_{|x|} = r_{|y|} = r^*_0$ (so the computations on x, y both start with the same state q_0 and end with the same state r^*_0, even if they get to it using different paths). Now consider the computation of D on xz: it starts with processing x, reaching r^*_0, and then continues processing z character by character, namely it is of the form: $(q_0, \ldots, r^*_0, r^*_1, \ldots, r^*_{|z|})$ where we have that $r^*_i = \delta(r^*_{i-1}, z_i)$ for all $1 \leq i \leq |z|$. Similarly, when we consider the computation of D on yz it also has the same form $(q_0, \ldots, r^*_0, r^*_1, \ldots, r^*_{|z|})$, with the same suffix, since once it gets to r^*_0, it processes z using δ in exactly the same way as above.

We conclude that the computation of D on xz and on yz ends in the same state. However, since one of xz, yz is in L, this state must be accepting, and since one of xz, yz is not in L, it must be rejecting, leading to a contradiction. Thus, it is not possible that the computations on x and on y ended in the same state. □

The above lemma is the key idea in the proof of the following lower bound.

Lemma 2.3. *Let $L \subseteq \Sigma^*$ be any language, and let $k > 0$ be an integer. 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.*

Proof. Let $\{x_1, \ldots, x_k\}$ be a set of strings that are pairwise distinguishable by L, and assume towards contradiction that there is a DFA D with fewer than k states that recognizes L. Since we have k distinct strings and fewer than k states, by the pigeonhole principle, there are two strings x_i, x_j where $i \neq j$ such that the computations of D on x_i and on x_j end in the same state. This contradicts Lemma 2.2. □

It is not hard to check (we leave it as an exercise) that for any language L, the relation $\sim_L$ is an equivalence relation:

Proposition 2.4. *Let $L \subseteq \Sigma^*$ be a language, and let $\sim_L$ 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 any equivalence relation $\sim$ on a set induces *equivalence classes* which partition the set into disjoint 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 $[i] = \{n \in \mathbb{Z} \mid n \equiv i \pmod 3\}$.

Since for any language L over alphabet Σ the indistinguishability-by-L relation $\sim_L$ is an equivalence relation, it partitions the set of strings Σ^* into equivalence classes, where each string $x \in \Sigma^*$ is in the equivalence class $[x] = \{y \in \Sigma^* \mid x \sim_L y\}$. Note that $x \sim_L y$ if and only if $[x]$ and $[y]$ are equal and denote the same equivalence class. We are now ready to state our main theorem.

Theorem 2.5. (Myhill-Nerode Theorem) *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 need to prove the following three statements (with the first one being our main focus in this class):

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 L is regular, then the number of states in the minimal DFA recognizing L is exactly the number of equivalence classes.

Proof of 1: We will prove the contrapositive. Suppose $\sim_L$ has infinitely many equivalence classes. That means that there is a S set of infinitely many strings (one from each equivalence class), such that every two of them are distinguishable by L. Now by Lemma 2.3, 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). We define the DFA $D = (Q, \Sigma, \delta, q_0, F)$ as follows.

Q We take $Q = \{[x] \mid x \in \Sigma^*\}$, namely we 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 We take $F = \{[x] \mid x \in L\}$, namely an equivalence class is an accepting state if it corresponds to $[x]$ for $x \in L$.

δ For a state/equivalence class $[x]$ and character $a \in \Sigma$, we define $\delta([x], a) = [xa]$.

We must first check that the above construction is indeed well defined, namely that the definitions of F and of δ are consistent, regardless of which representative of the equivalence class we take.

To show that F is well-defined, we need to prove that in every equivalence class, either all strings are in L or all strings are not in L. Indeed, 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.

To show that δ is well-defined, we need to show that if $[x] = [y]$, then for any character a we have that $[xa] = [ya]$. Indeed, if $x \sim_L y$, then for all strings $z \in \Sigma^*$, $xz \in L$ if and only if $yz \in L$. This includes strings of the form $z = au$, where u is any string. Since for any string u, $xau \in L$ if and only if $yau \in L$, we know $xa \sim_L ya$, so $[xa] = [ya]$.

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 number of equivalence classes of $\sim_L$. Furthermore, by Lemma 2.3, there cannot be any DFA for L with fewer states than the number of equivalence classes (since if we take one string from each equivalence class, they are all pairwise distinguishable by L). 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 covering in class. The key idea is that, as we saw in Lemma 2.2, 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). Since this is the main way we use the MN theorem in our class, we highlight this as an explicit corollary (which follows from the MN theorem):

Corollary 2.6. *Let $L \subseteq \Sigma^*$ be a language. If there exists a set $S \subseteq \Sigma^*$ such that*

1. *S has infinitely many strings*
2. *$\forall x, y \in S$ where $x \neq y$, we have that x and y are distinguishable by L*

then L is not regular.

3 Examples

We will provide some examples mentioned in class (lectures 4 and 5). More examples will be provided in the handout and the homework.

3.1 Regular Languages

Example 3.1. Let $L = \{w \in \{0,1\}^* \mid |w| \text{ is divisible by } 3\}$

We already saw a 3-state DFA for this language in a previous class, where we had one state for strings of length 0 mod 3, one for strings of length 1 mod 3, and one for strings of length 2 mod 3. We can see that this exactly corresponds to what the MN theorem gives us as the minimal DFA. Indeed, the equivalence classes here are $[\varepsilon], [1], [11]$ (we can check that any two of the strings $\varepsilon, 1, 11$ are distinguishable, and that every other string is equivalent to one of these three strings).

Example 3.2. Let $L = \{w \in \{0,1\}^* \mid |w| \geq 2$, and second to last character in w is a $1\}$

What are the equivalence classes? First notice that $[00], [01], [10], [11]$ are all different equivalence classes. Indeed, every two of these strings have a distinguishing extension: for $x = 00, y = 01$ take $z = 0$; for $x = 00, y = 10$ take $z = \varepsilon$; for $x = 00, y = 11$ take $z = 0$; for $x = 01, y = 10$ take $z = \varepsilon$; for $x = 01, y = 11$ take $z = \varepsilon$; for $x = 10, y = 11$ take $z = 0$.

Next, we can prove that these are all the equivalence classes – every other string is equivalent to one of them. Indeed, for strings x of length $|x| \geq 2$ the string is equivalent to its 2-character suffix (i.e., $x_1 \dots x_n \sim_L x_{n-1}x_n$), because for all strings z, the last two characters in xz are the same as the last two characters in $x_{n-1}x_nz$. For strings with fewer than 2 characters, we can check them individually. $\varepsilon \sim_L 0 \sim_L 00$ because for any z, the second to last character in z is 1 if and only if the second to last character in $0z$ is 1 if and only if the second to last character in $00z$ is 1. Finally, $1 \sim_L 01$ because for any z, the second to last character in $1z$ is 1 if and only if the second to last character in $01z$ is 1.

Thus the minimal DFA for this language has 4 states. We use the construction from Theorem 2.5. Note that $[\epsilon] = [00]$ and $F = \{[11], [10]\}$ (the classes corresponding to strings in the language).

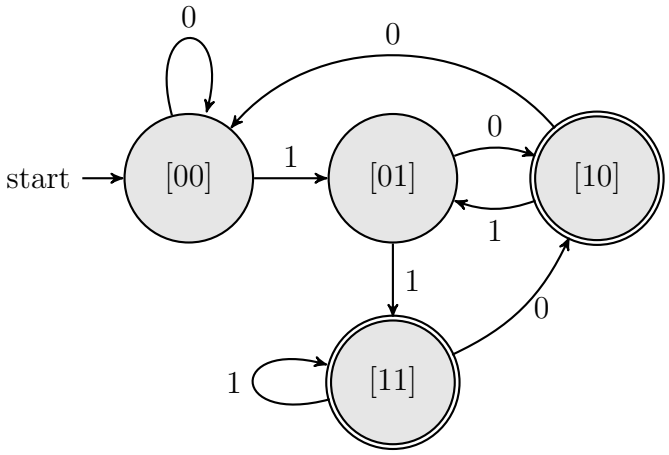

Figure 1: D the DFA recognizing L.

The above example can be generalized as follows.

Example 3.3. 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.

3.2 Non-Regular Languages

We will prove that all the following languages are non-regular.

Example 3.4. Let $L' = \{a^n b^n \mid n \geq 0\}$.

We will show that L' has infinitely many equivalence classes, and thus is not regular. Consider the set $S = \{a^i \mid i \geq 0\}$. These are infinitely many strings, and we claim that no two of them are equivalent. Indeed, for any $i \neq j$, consider the strings $x = a^i$ and $y = a^j$ in S. Choosing the extension $z = b^i$ gives $xz = a^i b^i$ which is in L', and $yz = a^j b^i$ which is not in L'. Thus, each equivalence class of L' can contain at most one string of the form a^i, so there must be infinitely many equivalence classes. So L' is not regular.

Example 3.5. Let $L = \{w \in \{0,1\}^* \mid$ the number of 0s in w is the same as the number of 1s in $w\}$.

We could use the MN theorem to prove this language has infinitely many equivalence classes (in fact, the same proof as above would work). But we show a different proof using closure properties.

We recall that above we proved that $L' = \{a^n b^n \mid n \geq 0\}$ is non-regular. We also proved in class that $L'' = \{a^i b^j \mid i, j \geq 0\}$ is regular (we showed a DFA for it). Now, notice that $L \cap L'' = L'$. Thus, if L was regular, by the above and the fact that the class of regular languages is closed under intersection (proved in class) we would have that L' is also regular, leading to a contradiction. Thus, L is not regular.

Example 3.6. Let $L_{\text{pali}} = \{w \in \{a,b\}^* \mid w$ 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 $xz = a^k b a^k \in L_{\text{pali}}$, and $yz = a^j b a^k \notin L_{\text{pali}}$. Therefore, all such pairs a^k, a^j are distinguishable, and we have infinitely many equivalence classes, so L_{pali} 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_{pali} is not regular.

Example 3.7. Let $L_2 = \{a^{2^i} \mid i \geq 0\}$ (here the alphabet is $\{a\}$ and the language consists of all strings whose length is a power of 2).

We will prove that this language is not regular, by showing that it has infinitely many equivalence classes. Let $S = \{a^{2^i} \mid i \geq 0\}$ (and note that S has infinitely many strings). For any $i \neq j$, consider the strings $x = a^{2^i}$ and $y = a^{2^j}$ which are both in S. Choose the extension $z = a^{2^i}$. We now have $xz = a^{2^i}a^{2^i} = a^{2^i+2^i} = a^{2^{i+1}}$ which is in L_2. However $yz = a^{2^j}a^{2^i} = a^{2^i+2^j}$. Now observe that $2^i + 2^j$ can't be a power of two since $i \neq j$ (to see this in more detail, suppose without loss of generality that $i < j$, and observe that $2^i + 2^j = 2^i(1 + 2^{j-i})$ is a product of 2^i with an odd number > 1). Thus $yz \notin L_2$. Hence, we proved that $\sim_{L_2}$ has infinitely many equivalence classes. So L_2 is not regular.

Note: in this proof we took the infinite set to be $S = L$, but there are many other possible infinite sets that would work. For example, we could take $S = \{a^i\}$ (which seems like a simpler set), and then work harder on finding appropriate distinguishing extensions for every two strings in the set. We thank the student who came to office hours to disucss this, it was fun to work together to find a solution for this choice as well!