THE MYHILL-NERODE THEOREM

MICHAEL TONG

Abstract. The Myhill-Nerode theorem is a fundamental result in the theory of regular languages. It can be used to prove whether or not a language L is regular and it can be used to find the minimal number of states in a DFA which recognizes L if L is regular. This handout is recommended reading for interested students, but it is not part of the required material for the class this semester.

1. Statement of the Theorem

The key concept to the Myhill-Nerode theorem is the distinguishing extension.

Definition 1.1. 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. (For a more rigorous definition of state equivalence, see section 4.4 of HMU)

Proposition 1.2. *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^*.*

Date: February 5, 2017.

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 1.3. 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 1.4. 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 first remark as a corollary.

We are now ready to state the theorem.

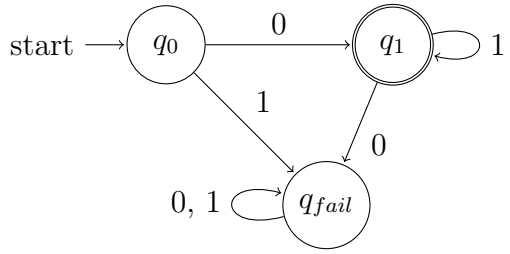
Theorem 1.5. *Let $L \subseteq \Sigma^*$ be a language. Then L is regular iff the number of equivalence classes of $\sim_L$ is finite; furthermore, if L is regular then the number of equivalence classes of $\sim_L$ is also the number of states in the minimal DFA.*

Thus, when talking about a regular language L, we can break it into its most essential pieces, namely the equivalence classes of $\sim_L$. Now let's look at some examples.

2. Examples

Example 2.1. Let $L = 01^*$. What are the equivalence classes? First notice that ϵ is distinguishable from all other strings in $\{0, 1\}^*$. Indeed, if w is a non-empty binary string then $w0$ is not in L but $\epsilon 0 = 0$ is. Another equivalence class would be 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. There is only one other equivalence class, and these are the strings in L, which gives 3 equivalence classes and thus L is regular and has a minimal DFA with 3 states.

On the other hand, consider the following minimal DFA:



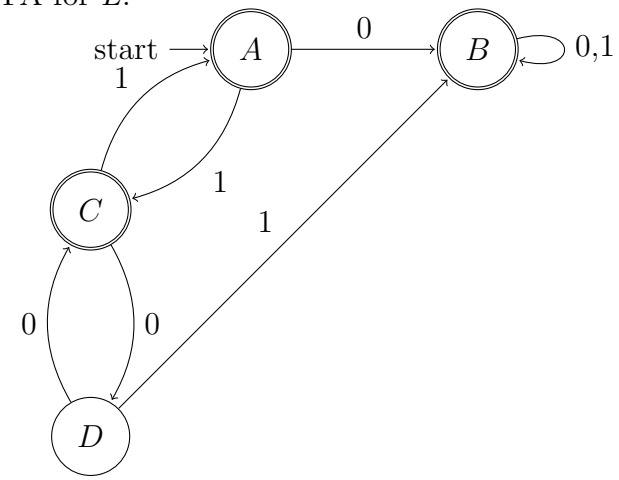
It is now clearer exactly how the equivalence classes relate to states: each equivalence class simply consists of the strings which cause the DFA to be in the same state. Indeed, the initial state has no arrows going to it, and thus only corresponds to the string ϵ. The state q_{fail} refers to the strings 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 2.2. Let $L = \{a^n b^n \mid n \geq 0\}$, the classic example of a language that is not regular. We will show that L has infinitely many equivalence classes by showing that a^k and a^j are distinguishable by L whenever $k \neq j$. Indeed, for $x = a^k$ and $y = a^j$, we may take $z = b^k$. Then $xz = a^k b^k$ is in L but $yz = a^j b^k$ is not. Thus, each equivalence class of L can contain at most one string of the form a^j, so there must be infinitely many equivalence classes. So L is not regular.

Example 2.3. Let

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

be the language discussed in class. As we have seen, the following is a DFA for L:



However, even with this state diagram, it is difficult to understand intuitively why this DFA works (we obtained it by transforming an NFA for the language). A student who chooses to stay anonymous has obtained this DFA directly and offered the following explanation: first, notice that if a string has either an even number of 0s or an even number of 1s, then it will be in the language since we can take x or y to be the empty string, respectively. So the only strings which may possibly be rejected are those with an odd number of zeros and an odd number of ones. Furthermore, if a string w has a prefix which has an even number of 1s and an odd number of 0s, then $w \in L$. Indeed, if such a w has an even number of 0s or 1s then it will be accepted and otherwise it has an odd number of 0s and 1s. In that case if we take x to be the prefix with an even number of 1s and odd number of 0s, then y will have an even number of 0s, so w is in the language. Conversely, if a string w with an odd number of 1s and 0s has a parsing as xy that puts it in the language, it must be the case that x has an even number of 1s (by definition) and an odd number of 0s (since y has an even number of 0s), so that this condition characterizes the language (it is an if and only if).

With this in mind, the state diagram becomes much more clear. Each state corresponds to the parity of each symbol in the string: state A refers to even 0s and even 1s, state C is even 0s and odd 1s, state D is odd 0s and odd 1s, and state B is odd 0s and even 1s, at which point it loops back on itself indefinitely for the reasons stated above. In terms of Myhill-Nerode, the equivalence classes of $\sim_L$ are the following:

- (B) Strings with a prefix with odd 0s and even 1s
- (A) Strings without the above prefix with even 0s and even 1s
- (C) Strings without the above prefix with even 0s and odd 1s
- (D) Strings without the above prefix with odd 0s and odd 1s

Indeed, none of these classes could be collapsed, as for any two of them there is a distinguishing extension (for D and any other class take ϵ, for A, B take 10, for A, C take 0, for B, C take 0). This means that this 4-state DFA is the minimal DFA for the language.

Example 2.4. We saw in class that the language

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

cannot have a DFA with fewer than 2^n states. While we didn't use this terminology, our proof relied on proving that all n-bit strings are distinguishable by L_n, which implies that there are at least 2^n equivalence classes (one for each n-bit string). We can prove that there are exactly 2^n equivalence classes by showing that every string of length $> n$ 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.