The Myhill-Nerode Theorem and DFA Minimization

Arthur Nunes-Harwitt

1 Introduction

The DFA model of computation has explicit state names for every possible state that a machine might be in. Nevertheless, when writing programs, although we do think about the different states that a program might be in, we do not explicitly label them. We now consider how to identify machine states associated with a language $L$ merely by identifying a particular relationship between strings in $L$ rather than discussing explicit labels.

The payoff for such an abstract notion of state is great. We will be able to articulate a new characterization of regular languages that will be useful for determining whether or not a language is regular. Further, we will also be able to define and compute a canonical minimal DFA — a DFA with the fewest possible states.

Section 2 introduces the relationship between strings and concludes with a theorem characterizing regular languages. Section 3 extends the ideas in section 2 and develops an algorithm for minimizing DFAs.

2 The Myhill-Nerode Theorem

Suppose we have a DFA $M$ and from the start state, $M$ reads a string $x$, and from the start state $M$ reads a string $y$. Do both strings lead to the same state? One way to check would be to inspect the state labels and see if they are identical. However, more abstractly, we might say that the states must be the same if no matter how we continue, we end up with the same result. We formalize that notion with the following definition.

Definition 1. Given a language $L \subseteq \Sigma^*$, and $x,y \in \Sigma^*$, $x \equiv_L y$ if for any $z \in \Sigma^*$, $xz \in L$ iff $yz \in L$.

Not surprisingly, this relation is an equivalence relation.

Theorem 1. Given a language $L \subseteq \Sigma^*$, $\equiv_L$ is an equivalence relation.

Proof

Exercise for the reader.

If some strings are equivalent, we might wonder how many strings we can find that are not equivalent. The next two definitions get at this idea.

Definition 2. Given a language $L \subseteq \Sigma^*$, and $X \subseteq \Sigma^*$, $X$ is pairwise distinguishable by $L$ if for any $x,y \in X$, $x \neq y$ implies $x \not\equiv_L y$.

Definition 3. Given a language $L \subseteq \Sigma^*$, the index of $L$ is $\max\{|X| \mid X$ is pairwise distinguishable by $L\}$, which may be infinite.

Before proceeding, we prove a technical lemma concerning the composition of the DFA transition function.
Lemma 1. Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$, for any $x, y \in \Sigma^*$, and any $q \in Q$, $\delta^*(\delta(q, x), y) = \delta^*(q, xy)$.

Proof

By structural induction on $y$.

- Observe that $\delta^*(\delta^*(q, x), \varepsilon) = \delta^*(q, x) = \delta^*(q, x\varepsilon)$.

- Assume that $\delta^*(\delta^*(q, x), y') = \delta^*(q, xy')$ if $y'$ is a substructure of $y$.

  Suppose $y = y'a$.

  $\delta^*(\delta^*(q, x), y) = \delta^*(\delta^*(q, x), y'a)$

  $= \delta(\delta^*(q, x), y'), a)$

  $= \delta^*(q, xy'a)$

  $= \delta^*(q, x'(y'a))$

  $= \delta^*(q, xy)$

$\Box$

The next theorem introduces a key point: if we have a set of strings that are not equivalent, the associated DFA can’t have fewer states than the size of that set — if so then at least two strings would be associated with the same state and hence be equivalent.

Theorem 2. Given a language $L \subseteq \Sigma^*$, and $X \subseteq \Sigma^*$, if $X$ is pairwise distinguishable by $L$ and $|X| = n$ then for any DFA $M = (Q, \Sigma, \delta, q_0, F)$, if $L(M) = L$ then $|Q| \geq n$.

Proof

By contradiction.

Suppose $X = \{x_1, \cdots, x_n\}$ is pairwise distinguishable by $L$, and there exists a DFA $M = (Q, \Sigma, \delta, q_0, F)$ such that $L(M) = L$ but $|Q| < n$.

Since there is a DFA $M$, we can apply the transition function to the strings in $X$.

Note that $|\{\delta^*(q_0, x_1), \cdots, \delta^*(q_0, x_n)\}| < n$. Thus for some $i \neq j$, $\delta^*(q_0, x_i) = \delta^*(q_0, x_j)$. And so, for any $z \in \Sigma^*$, $\delta^*(\delta(q_0, x_i), z) \in F$ if $\delta^*(\delta(q_0, x_j), z) \in F$. Hence, $\delta^*(q_0, x_i z) \in F$ if $\delta^*(q_0, x_j z) \in F$, which means that $x_i z \in L(M)$ if and only if $x_j z \in L(M)$. Therefore $x_i \equiv_L x_j$. However, $x_i \not\equiv_L x_j$ since $X$ is pairwise distinguishable.

Since $x_i \equiv_L x_j$ and $x_i \not\equiv_L x_j$ are contradictory, $X$ is pairwise distinguishable by $L$, and there exists a DFA $M = (Q, \Sigma, \delta, q_0, F)$ such that $L(M) = L$ but $|Q| < n$ cannot be the case.

$\Box$

We use the previous theorem to establish a property of a language’s index if the language is regular.

Theorem 3. Given a language $L \subseteq \Sigma^*$, if there exists a DFA $M = (Q, \Sigma, \delta, q_0, F)$ such that $L(M) = L$ and $|Q| = k$, then the index of $L$ is at most $k$.

Proof

By contradiction.

Suppose there exists a DFA $M$ such that $L(M) = L$ and $|Q| = k$, but the index of $L$ is greater than $k$.

From our assumption we can immediately derive that $|Q| = k$.

However, since the index of $L$ is greater than $k$, there exists an $X$ such that $X$ is pairwise distinguishable by $L$ and $|X| > k$. From theorem 2 it follows that $|Q| \geq |X|$. Therefore $|Q| > k$.

Since $|Q| = k$ and $|Q| > k$ are contradictory, there exists a DFA $M$ such that $L(M) = L$ and $|Q| = k$, but the index of $L$ is greater than $k$ cannot be the case.
If we have an equivalence relation on our language, we can partition it into non-overlapping classes. We intuitively understand that two strings are equivalent if they take us to the same state, so the equivalence class is all strings that take us to the same state. Thus we can use an equivalence class as a state. The next definition introduces (standard) notation for these equivalence classes.

**Definition 4.** Given a language \( L \subseteq \Sigma^* \) and \( x \in \Sigma^* \), \([x]_L\) is \( \{y \in \Sigma^* \mid y \equiv_L x\} \).

Now we have another technical lemma.

**Lemma 2.** Given a language \( L \subseteq \Sigma^* \) and \( x \in \Sigma^* \), \([x]_L \cap L \neq \emptyset \) iff \( x \in L \).

**Proof**

Show that \([x]_L \cap L \neq \emptyset \) implies \( x \in L \).

Suppose \([x]_L \cap L \neq \emptyset \).

Since the intersection is not empty, let \( y \in [x]_L \cap L \). Thus \( y \in L \). Further, since \( y \in [x]_L \), \( y \equiv_L x \), or \( yz \in L \) iff \( xz \in L \) for any \( z \in \Sigma^* \). By picking \( z = \epsilon \), \( y \in L \) implies \( x \in L \). Hence \( x \in L \).

Show that \( x \in L \) implies \([x]_L \cap L \neq \emptyset \).

Suppose \( x \in L \).

Since \( x \in [x]_L \), it follows that \([x]_L \cap L \neq \emptyset \).

Starting with the notion that an equivalence class is a state, the theorem below shows how we can build a DFA to recognize a language from such states. It turns out that this construction involves the fewest possible states. We will explore how to practically implement this construction in the next section.

**Theorem 4.** Given a language \( L \subseteq \Sigma^* \), if the index of \( L \) is \( k \), where \( k \) is finite, then there exists a DFA \( M = (Q, \Sigma, \delta, q_0, F) \) such that \( L(M) = L \) and \( |Q| = k \).

**Proof**

Suppose the index of \( L \) is \( k \), where \( k \) is finite.

First consider the following example DFA \( M \).

Let \( Q = \{[x]_L \mid x \in \Sigma^* \} \).

The alphabet \( \Sigma \) has been given.

Let \( \delta([x]_L, a) = [xa]_L \).

Let \( q_0 = [\epsilon]_L \).

Let \( F = \{q \in Q \mid q \cap L \neq \emptyset \} \).

Now observe that \( L(M) = L \).

\( \delta^*([x]_L, y) = [xy]_L \) follows by structural induction on \( y \).

- Observe that \( \delta^*([x]_L, \epsilon) = [x]_L = [x\epsilon]_L \).
- Assume that \( \delta^*([x]_L, y') = [xy']_L \) if \( y' \) is a substructure of \( y \).

Suppose \( y = y'a \).

\[ \delta^*([x]_L, y) = \delta^*([x]_L, y'a) = \delta(\delta^*([x]_L, y'), a) = \delta([xy']_L, a) = [xy'a]_L = [xy]_L \]
Therefore, \( \delta^*(q_0, y) = [y]_L \). Thus \( y \in L(M) \iff [y]_L \in F \iff [y]_L \cap L \neq \emptyset \iff y \in L \).

Finally, observe that \( |Q| = k \).

Since the index of \( L \) is \( k \), there exists an \( X \) such that \( X \) is pairwise distinguishable by \( L \) and \( |X| = k \). From theorem 2 it follows that \( |Q| \geq k \).

That it is not the case that \( |Q| > k \) is established by contradiction.

Suppose \( |Q| > k \).

Then \( Q = \{[x_1]_L , \ldots , [x_k]_L , \ldots \} \), and so \( X = \{x_1 , \ldots , x_{k+1} , \ldots \} \) is pairwise distinguishable by \( L \). But \( |X| \leq k \) since the index of \( L \) is \( k \).

Since \( |X| > k \) and \( |X| \leq k \) are contradictory, \( |Q| > k \) cannot be the case.

Because \( |Q| \geq k \) and \( |Q| \leq k \), it must be that \( |Q| = k \).

\[ \square \]

The Myhill-Nerode theorem follows from the previous two theorems. It establishes that a language is regular exactly when its index is finite. This abstract characterization can be used both to certify a language as regular and to prove that a language cannot be regular.

**Corollary 1** (Myhill-Nerode). *Given a language \( L \subseteq \Sigma^* \), \( L \) is regular iff the index of \( L \) is finite. Moreover, the index of \( L \) is the number of states of a DFA \( M \) with the fewest states that recognizes \( L \).*

**Proof**

Show that \( L \) is regular implies the index of \( L \) is finite.

Suppose \( L \) is regular.

By theorem 3 the index of \( L \) is finite.

Show that the index of \( L \) is finite implies \( L \) is regular.

Suppose the index of \( L \) is finite.

By theorem 4 there exists a DFA \( M \) such that \( L(M) = L \). Hence \( L \) is regular.

Show that the index of \( L \) is the number of states of a DFA \( M \) with the fewest states that recognizes \( L \).

Theorem 2 entails that a DFA \( M \) that recognizes \( L \) must be at least the index of \( L \). Theorem 4 shows that the smallest possible value is realizable.

\[ \square \]

**2.1 Distinguishability Examples**

**Theorem 5.** \( L = \{0^k 1^k \mid k \in \mathbb{N}\} \) is not regular.

**Proof**

Note that the index of \( L \) is at least as large as the size of any pairwise distinguishable set \( X \).

Consider \( X = \{0^k \mid k \in \mathbb{N}\} \).

Observe that \( 0^m \not\equiv_L 0^n \) if \( m \neq n \), since \( 0^m 1^m \in L \) but \( 0^n 1^n \not\in L \). Thus \( X \) is pairwise distinguishable by \( L \).

Note that \( X \) is infinite. Hence the index of \( L \) is infinite, and so \( L \) is not regular.

\[ \square \]

**Theorem 6.** \( L = \{x \in \{0, 1\}^* \mid x \text{ is a palindrome}\} \) is not regular.

**Proof**

Note that the index of \( L \) is at least as large as the size of any pairwise distinguishable set \( X \).

Consider \( X = \{0^k 1^k \mid k \in \mathbb{N}\} \).
Observe that $0^m1 \not\in L$, $0^n1$ if $m \neq n$, since $0^m10^m \in L$ but $0^n10^m \not\in L$. Thus $X$ is pairwise distinguishable by $L$.

Note that $X$ is infinite. Hence the index of $L$ is infinite, and so $L$ is not regular.

\[\Box\]

### 3 DFA Minimization

Let’s consider two questions. First, if we agree that strings $x$ and $y$ are equivalent, and we actually feed those strings into a DFA, will the states really be the same when we peek at their labels? Second, how can we construct a DFA using equivalence classes?

The answer to the first question is actually “no.” Nevertheless, since the states lead to the same results (acceptance or rejection), they are “equivalent” and could be collapsed. The notion of state equivalence is formalized below.

**Definition 5.** Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$, and $p, q \in Q$, $p \equiv_M q$ if for any $z \in \Sigma^*$, $\delta^*(p, z) \in F$ iff $\delta^*(q, z) \in F$.

**Theorem 7.** Given a DFA $M$, $\equiv_M$ is an equivalence relation.

**Proof**

Exercise for the reader.

\[\Box\]

The construction of the DFA in theorem 4 made use of equivalence classes as states. It’s hard to imagine how we could write a program to work with equivalence classes of strings. But we don’t have to. The language must be specified somehow. If it is regular, it is expressible as a DFA. Hence we can work with state equivalence rather than string equivalence. The following theorem formalizes our intuition that those notions are the same.

**Theorem 8.** Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$, and $x, y \in \Sigma^*$, $\delta^*(q_0, x) \equiv_M \delta^*(q_0, y)$ iff $x \equiv_{L(M)} y$.

**Proof**

Show that $\delta^*(q_0, x) \equiv_M \delta^*(q_0, y)$ implies $x \equiv_{L(M)} y$.

Suppose $\delta^*(q_0, x) \equiv_M \delta^*(q_0, y)$.

Then $\delta^*(\delta^*(q_0, x), z) \in F$ iff $\delta^*(\delta^*(q_0, y), z) \in F$ for any $z \in \Sigma^*$. Hence, by lemma 1, $\delta^*(q_0, xz) \in F$ iff $\delta^*(q_0, yz) \in F$. And so $xz \in L(M)$ iff $yz \in L(M)$. Thus $x \equiv_{L(M)} y$.

Show that $\delta^*(q_0, x) \not\equiv_M \delta^*(q_0, y)$ implies $x \not\equiv_{L(M)} y$.

Suppose $\delta^*(q_0, x) \not\equiv_M \delta^*(q_0, y)$.

Then $\delta^*(\delta^*(q_0, x), z) \in F$ but $\delta^*(\delta^*(q_0, y), z) \not\in F$ for some $z$. Hence $\delta^*(q_0, xz) \not\in F$ but $\delta^*(q_0, yz) \not\in F$. And so $xz \in L(M)$ but $yz \not\in L(M)$. Thus $x \not\equiv_{L(M)} y$.

\[\Box\]

It turns out that state non-equivalence is easier\(^1\) to work with. The following characterization follows immediately from definition 5.

**Lemma 3.** Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$, and $p, q \in Q$, $p \not\equiv_M q$ iff there exists $a z \in \Sigma^*$, $\delta^*(p, z) \in F$ $\Leftrightarrow$ $\delta^*(q, z) \in F$.

\(^1\)State equivalence is characterized by a greatest fixed point.
While the above characterization is fine from a logical perspective, it is not so easy to use for computing. Rather, we'd prefer a recursive specification. The next two definitions do just that; the definition is split into two parts for technical reasons to make lemma 5 simpler.

**Definition 6.** Given a DFA \( M = (Q, \Sigma, \delta, q_0, F) \), and \( p, q \in Q \). \( p \neq_M^\delta q \) is determined as follows.

- \( p \neq_M^0 q \) if \( p \in F \iff q \in F \), or
- \( p \neq_M^{n+1} q \) if there exists an \( a \in \Sigma \) such that \( \delta(p, a) \neq_M^n \delta(q, a) \).

**Definition 7.** Given a DFA \( M = (Q, \Sigma, \delta, q_0, F) \), and \( p, q \in Q \). \( p \neq_M^n q \) if there exists an \( n \in \mathbb{N} \) such that \( p \neq_M^n q \).

The two lemmas and the theorem that follow are about establishing that the recursive definition does in fact refer to the same relation that the non-recursive definition does.

**Lemma 4.** Given a DFA \( M = (Q, \Sigma, \delta, q_0, F) \), for any \( z \in \Sigma^* \), and \( p, q \in Q \), if \( \delta^*(p, z) \in F \iff \delta^*(q, z) \in F \), then \( p \neq_M q \).

**Proof**

By mathematical induction on the length of \( z \)

- Observe that when \( |z| = 0, z = \varepsilon \). In that case, \( \delta^*(p, \varepsilon) \in F \iff \delta^*(q, \varepsilon) \in F \) implies \( p \in F \iff q \in F \) which implies \( p \neq_M q \).
- Assume that if \( \delta^*(p, z') \in F \iff \delta^*(q, z') \in F \), then \( p \neq_M q \), when \( |z'| = k \).
  Suppose \( |z| = k + 1 \). Then \( z \) can be expressed as \( z = az' \), and \( |z'| = k \).
  \[
  \delta^*(p, z) \in F \iff \delta^*(q, z) \in F \implies 
  \delta^*(p, az') \in F \iff \delta^*(q, az') \in F 
  \implies 
  \delta^*(\delta(p, a), z') \in F \iff \delta^*(\delta(q, a), z') \in F 
  \implies 
  \delta(p, a) \neq_M \delta(q, a) 
  \implies p \neq_M q
  \]

**Lemma 5.** Given a DFA \( M = (Q, \Sigma, \delta, q_0, F) \), for any \( n \in \mathbb{N} \), and \( p, q \in Q \), if \( p \neq_M^n q \) then \( p \neq_M q \).

**Proof**

By mathematical induction.

- Observe that \( p \neq_M^0 q \) implies \( p \in F \iff q \in F \), which implies \( \delta^*(p, \varepsilon) \in F \iff \delta^*(q, \varepsilon) \in F \), so \( p \neq_M q \).
- Assume if \( p \neq_M^k q \) then \( p \neq_M q \).
  Suppose \( p \neq_M^k q \).
  Then there exists an \( a \in \Sigma \) such that \( \delta(p, a) \neq_M^k \delta(q, a) \). Hence, by assumption, \( \delta(p, a) \neq_M \delta(q, a) \), which means that \( \delta^*(\delta(p, a), z) \in F \iff \delta^*(\delta(q, a), z) \in F \) for some \( z \in \Sigma^* \). And so \( \delta^*(p, az) \in F \iff \delta^*(q, az) \in F \). Thus \( p \neq_M q \).

**Theorem 9.** Given a DFA \( M = (Q, \Sigma, \delta, q_0, F) \), and \( p, q \in Q \). \( p \neq_M q \) iff \( p \neq_M q \).

**Proof**
Show that \( p \not\equiv_M q \) implies \( p \not\equiv^*_M q \).

Suppose \( p \not\equiv_M q \).
Then there exists a \( z \in \Sigma^* \) such that \( \delta^*(q, z) \in F \iff \delta^*(q', z) \in F \). It immediately follows from lemma 4 that \( p \not\equiv^*_M q \).

Show that \( p \not\equiv^*_M q \) implies \( p \not\equiv_M q \).

Suppose \( p \not\equiv_M q \).
Then there exists an \( n \in \mathbb{N} \) such that \( p \not\equiv^n_M q \). It immediately follows from lemma 5 that \( p \not\equiv_M q \).

\( \square \)

We must be careful when implementing state non-equivalence. If there are \( n \) states, there are only \( n^2 \) pairs of states, so we expect a polynomial time computation, but a naive recursive implementation would involve multiple recursive calls, and therefore have exponential time complexity. The naive approach can be improved using either memoization or dynamic programming.