We know that every regular language has a DFA accepting the language, but how about other languages? We can define an abstract acceptor for any language $L \subseteq \Sigma^*$, but an infinite set of states is required for non-regular languages.

Use the standard definition of a DFA $(Q, \Sigma, q_0, \delta, F)$, but drop the requirement that $Q$ be finite, leaving everything else unchanged. We’ll call this a deterministic acceptor (DA). It is now easy to see that there is at least one DA for any language.

To define a DA $D(L) = (Q, \Sigma, q_0, \delta, F)$ for a language $L$, let

- $Q = \Sigma^*$
- $q_0 = \varepsilon$
- $\forall \sigma \in \Sigma \forall x \in \Sigma^* \delta(x, \sigma) = x\sigma$
- $F = L$

$D(L)$ can be viewed as an infinite tree:

- Nodes are elements of $\Sigma^*$.
- The root is $\varepsilon$.
- Each node $x$ is connected to $x\sigma$ for each $\sigma \in \Sigma$.
- Nodes that are elements of $L$ are accepting.

**State equivalence for $D(L)$**

The definition of state equivalence for a DA is the same as for a DFA $(Q, \Sigma, q_0, \delta, F)$:

$$q \equiv q' \iff \forall z \in \Sigma^* [\delta(q, z) \in F \iff \delta(q', z) \in F]$$

As with DFAs, this equivalence relation is a congruence.

The relation $\equiv$ for $D(L)$ depends only on $L$, and can be restated thus:

$$x \equiv y \iff \forall z \in \Sigma^* [xz \in L \iff yz \in L]$$

Expressed this way, $\equiv_L$ is a relation on $\Sigma^*$, called the **Myhill-Nerode relation** for $L$. (When it is clear that we are talking about a specific $L$, we can drop the subscript $L$ from $\equiv_L$. Similarly, the equivalence classes $[x]_L$ can drop the subscript.)

**Observations** about any language $L \subseteq \Sigma^*$:

1. $\forall x \in L [x] \subseteq L$.
2. $L$ is the union of zero or more equivalence classes of $\equiv$, namely $L = \cup \{[x] \mid x \in L\}$. 
Proofs of observations:

1. Suppose that \( x \in L \), and let \( y \) be an arbitrary element of \([x]\), to show \( y \in L \). In the definition of \( \equiv \), take \( z = \varepsilon \), giving \( x\varepsilon \in L \iff y\varepsilon \in L \). But \( x\varepsilon = x \) and \( y\varepsilon = y \), so \( x \in L \iff y \in L \). Thus, from \( x \in L \), also \( y \in L \).

2. Because \( x \in [x] \), we have \( L \subseteq \bigcup \{[x] \mid x \in L \} \). To show \( \bigcup \{[x] \mid x \in L \} \subseteq L \), consider an arbitrary \( x \in L \), and an arbitrary \( y \in [x] \). From 1, \( y \in L \).

Myhill-Nerode Theorem

\( L \subseteq \Sigma^* \) is regular iff \( \equiv_L \) has a finite set of equivalence classes.

Proof:

\((\Rightarrow)\) Suppose \( L \) is regular. Let \( M = (Q, \Sigma, q_0, \delta, F) \) be a DFA accepting \( L \). For each \( q \in Q \), define \( <q> = \{ x \in \Sigma^* \mid \delta(q_0, x) = q \} \), the set of strings leading from the initial state to \( q \).

Claim: For any \( q \in Q \), if \( x \in <q> \), then \( <q> \subseteq [x] \). That is, the set of strings leading to \( q \) is contained within a single equivalence class.

To see this, suppose that \( y \in <q> \). Then \( \delta(q_0, x) = q \) and \( \delta(q_0, y) = q \).

So for any \( z \in \Sigma^* \), \( \delta(q_0, xz) = \delta(\delta(q_0, x), z) = \delta(\delta(q_0, y), z) = \delta(q_0, yz) \), thus \( \delta(q_0, xz) \in F \iff \delta(q_0, yz) \in F \), which means \( x \equiv y \), thus \( y \in [x] \).

As we have shown above, every set of strings leading to a specific state of \( M \) maps to a single Myhill-Nerode equivalence class. Because the set of such sets is finite, the set of equivalence classes must also be finite.

\((\Leftarrow)\) Suppose the set of equivalence classes of \( \equiv_L \) is finite. Construct a DFA \( M \) accepting \( L \) by using the equivalence classes as states. The initial state is \([\varepsilon]\). The transition function \( \delta \) is defined by

\[ \forall x \in \Sigma^* \ \forall \sigma \in \Sigma \ \delta([x], \sigma) = [xo] \]

which is well-defined, because if \([x] = [y]\), then \([xo] = [yo]\) by the congruence property of \( \equiv \). Finally the accepting states of are those classes \([x]\) where \( x \in L \).

It is easy to show by induction that \( \forall x, y \in \Sigma^* \ \delta([x], y) = [xy] \). Therefore \( \forall x \in \Sigma^* \ \delta([\varepsilon], x) = [x] \). Thus \( M \) accepts a string \( x \) iff \( x \in L \).

Corollary: \( L \subseteq \Sigma^* \) is regular iff \( \equiv_L \) has a finite set of equivalence classes and \( L \) is the union of some of those classes.

Proof: This follows from the theorem and Observation 1.
Example

Consider the following DFA $M$:

![DFA Diagram]

There are just two Myhill-Nerode Classes:

$[\varepsilon] = \{\varepsilon\} \cup \{x1 \mid x \in \{0, 1\}^*\} = \text{strings that are empty or end in 1}$

$[0] = \{x0 \mid x \in \{0, 1\}^*\} = \text{strings ending in 0}$

The diagram below shows the classes as the states of the minimal-state DFA that would be constructed in our proof. The accepting state is $[0]$ because 0 is in the language $\{x0 \mid x \in \{0, 1\}^*\}$.

![Minimal DFA Diagram]

Example

Consider the language $\{0^n1^n \mid n \geq 0\}$. Here there is an infinite set of equivalence classes:

$[\varepsilon] = \{\varepsilon\}$ \hspace{1cm} open to receive anything

$[01] = \{0^n1^n \mid n > 0\}$ \hspace{1cm} not open to receive anything

For each $n > 0$: $[0^n] = \{0^n\}$ \hspace{1cm} open to receive more 0's or 0's then 1's

For each $m > 0$: $[0^n1^{n-m}] = \{0^n1^{n-m} \mid n > 0\}$ \hspace{1cm} open to receive up to $m$ 1's

One class containing everything not included above.
Shortcut for showing L is not regular

To use the Myhill-Nerode theorem to show that a language is not regular, it is not necessary to display all of the equivalence classes. All we need to do is to show that the set of classes is infinite. A sufficient condition for the set to be infinite is that we can find an infinite set S of strings where no two strings in S are in the same equivalence class, i.e. no two strings in S are equivalent.

For the example $L = \{0^n1^n \mid n \geq 0\}$, one such set is $S = \{0^n \mid n \geq 0\}$. We can easily see that no two strings in this S are equivalent: Consider $0^m$ vs. $0^n$ where $m \neq n$. These strings are not equivalent because there is a string, namely $1^m$, where $0^m1^m \in L$ but $0^n1^m \notin L$.

Constructing the Myhill-Nerode Classes given a DFA

If L is known to be regular, then the Myhill-Nerode classes can be constructed from any DFA for L. First determine which states of the DFA are equivalent, then construct a minimal-state DFA from that information. Here’s how to do this informally:

Divide the states into two sets: accepting and non-accepting. These sets form the blocks of an equivalence relation, but not necessarily the one corresponding to equivalence. (We know that states in different blocks cannot be equivalent.)

Within each block, examine pairs of states $q, q'$ within that set. If there is a $\sigma \in \Sigma$ such that $\delta(q, \sigma)$ and $\delta(q', \sigma)$ are in different blocks, then $q$ and $q'$ should be in different blocks too. Doing this for all pairs within a block forms another equivalence relation.

Repeat the preceding paragraph, until there is no change in the blocks. The result represents state equivalence.

Example

Consider the following DFA:

![DFA Diagram]

The algorithm starts with the partition into accepting and non-accepting:

$\{\{p_0, p_1, p_2\}, \{p_3, p_4, p_5\}\}$. 
We then compare pairs:
- \( p_0, p_1: \delta(p_0, 0) \) and \( \delta(p_1, 0) \) are the same block, as are \( \delta(p_0, 1) \) and \( \delta(p_1, 1) \), so this pair stays in the same block for the time being.
- \( p_0, p_2: \delta(p_0, 0) \) and \( \delta(p_2, 0) \) are in different blocks, so this pair splits a block, giving
  \[
  \{\{p_0, p_1\}, \{p_2\}, \{p_3, p_4, p_5\}\}.
  \]
- \( p_3, p_4: \delta(p_3, 0) \) and \( \delta(p_4, 0) \) are the same block, as are \( \delta(p_3, 1) \) and \( \delta(p_4, 1) \), so this pair stays in the same block for the time being.
- \( p_4, p_5: \delta(p_4, 0) \) and \( \delta(p_5, 0) \) are the same block, as are \( \delta(p_4, 1) \) and \( \delta(p_5, 1) \), so this pair stays in the same block for the time being.

Now return to the new block \( \{p_0, p_1\} \):
- \( p_0, p_1: \delta(p_0, 0) \) and \( \delta(p_1, 0) \) are still in the same block, but are \( \delta(p_0, 1) \) and \( \delta(p_1, 1) \), are not, so this pair splits, giving
  \[
  \{\{p_0\}, \{p_1\}, \{p_2\}, \{p_3, p_4, p_5\}\}
  \]

Now there are no further splits, so we have that states in each block are equivalent.

**Forming the Minimal DFA**

Given the partition for equivalent states, we may form the minimal DFA equivalent to the original as follows:
- The states are the blocks.
- The initial state is the block containing the original initial state.
- The accepting states are blocks containing accepting states.
- \( \delta \) is defined by
  \[
  \forall \sigma \in \Sigma \ \delta([q], \sigma) = [\delta(q, \sigma)]
  \]
That is, the transition from the block containing \( q \) via \( \sigma \) is to the block containing \( \delta(q, \sigma) \). (The similarity to the construction in the Myhill-Nerode proof should be noted.)

The minimal DFA for the previous example is shown below.