Myhill-Nerode Theorem

**Definition** Let \( A \) be any language over \( \Sigma^* \). We say that strings \( x \) and \( y \) in \( \Sigma^* \) are *indistinguishable by \( A \)* iff for every string \( z \in \Sigma^* \) either both \( xz \) and \( yz \) are in \( A \) or both \( xz \) and \( yz \) are not in \( A \). We write \( x \equiv_A y \) in this case.

Note that \( \equiv_A \) is an equivalence relation. (Check this yourself.)

**Definition** Given a DFA \( M = (Q, \Sigma, \delta, s, F) \) we say that two strings \( x \) and \( y \) in \( \Sigma^* \) are *indistinguishable by \( M \)* iff \( \delta^*(s, x) = \delta^*(s, y) \), i.e. the state reached by \( M \) on input \( x \) is the same as the state reached by \( M \) on input \( y \). We write \( x \equiv_M y \) in this case.

Note that \( \equiv_M \) is an equivalence relation and that it only has a finite number of equivalence classes, one per state. In fact, the equivalence classes of \( \equiv_M \) are precisely the sets of inputs that you would have used to document the states of \( M \).

**Lemma 1** If \( A = L(M) \) for a DFA \( M \) then for any \( x, y \in \Sigma^* \) if \( x \equiv_M y \) then \( x \equiv_A y \).

**Proof** Suppose that \( A = L(M) \). Therefore \( w \in A \iff \delta^*(s, w) \in F \). Suppose also that \( x \equiv_M y \). Then \( \delta^*(s, x) = \delta^*(s, y) \).

Let \( z \in \Sigma^* \). Clearly \( \delta^*(s, xz) = \delta^*(s, yz) \). Therefore

\[
\begin{align*}
  xz \in A & \iff \delta^*(s, xz) \in F \\
        & \iff \delta^*(s, yz) \in F \\
        & \iff yz \in A
\end{align*}
\]

It follows that \( x \equiv_A y \). \( \square \)

This lemma says that whenever two elements arrive at the same state of \( M \) they are in the same equivalence class of \( \equiv_A \). This means that each equivalence class of \( \equiv_A \) is a union of equivalence classes of \( \equiv_M \).

**Corollary 2** If \( A \) is regular then \( \equiv_A \) has a finite number of equivalence classes.

**Proof** Let \( M \) be a DFA such that \( A = L(M) \). The Lemma shows that \( \equiv_A \) has at most as many equivalence classes as \( \equiv_M \), which has a finite number of equivalence classes (equal to the number of states of \( M \)). \( \square \)

We now get another way of proving that languages are not regular, namely given \( A \) find an infinite sequence of strings \( x_1, x_2, \ldots \) and prove that they are not equivalent to each other with respect to \( \equiv_A \).
Claim 3 $A = \{0^n1^n : n \geq 0 \}$ is not regular.

Proof Consider the infinite sequence of strings $x_1, x_2, \ldots$ where $x_i = 0^i$ for $i \geq 1$. We now see that no two of these are equivalent to each other with respect to $\equiv_A$: Consider $x_i = 0^i$ and $x_j = 0^j$ for $i \neq j$. Let $z = 1^i$ and notice that $x_iz = 0^i1^i \in A$ but $x_jz = 0^j1^i \notin A$. Therefore no two of these strings are equivalent to each other under $\equiv_A$, so $\equiv_A$ has an infinite number of equivalence classes. Therefore by the above Corollary, $A$ cannot be regular.

One nice thing about this method for proving things nonregular is that, unlike the pumping lemma, it is always guaranteed to work because the corollary above is a precise characterization of the regular languages. The statement of this fact is known as the Myhill-Nerode Theorem after the two people who first proved it.

Theorem 4 (Myhill-Nerode Theorem) $A$ is regular if and only if $\equiv_A$ has a finite number of equivalence classes. Furthermore there is a DFA $M$ with $L(M) = A$ having precisely one state for each equivalence class of $\equiv_A$.

Proof The corollary above already gives one direction of this statement. All we now need to show is that if $\equiv_A$ has a finite number of equivalence classes then we can build a DFA $M = (Q, \Sigma, \delta, s, F)$ accepting $A$ where there is one state in $Q$ for each equivalence class of $\equiv_A$. Here is how it goes:

Let $A_1, \ldots, A_r$ be the equivalence classes of $\equiv_A$. Remember that the $A_i$ are disjoint and their union is all of $\Sigma^*$. Define $Q = \{q_1, \ldots, q_r\}$. Our goal will be to define the machine $M$ so that $\delta^*(s, x) = q_j \iff x \in A_j$.

Let $s \in Q$ be the one $q_i$ such that $\epsilon \in A_i$.

Note that for any $A_j$ and any $a \in \Sigma$, for every $x, y \in A_j$, $xa$ and $ya$ will both be contained in the same equivalence class of $\equiv_A$. (For any $z \in \Sigma^*$, $xaz \in A \iff yaz \in A$ since $x$ and $y$ are in the same equivalence class of $\equiv_A$.)

To figure out what $\delta(q_j, a)$ should be, all we do is pick some $x \in A_j$, find the one $k$ such that $xa \in A_k$ and set $\delta(q_j, a) = q_k$. The answer will be the same no matter which $x$ we choose.

To pick the final states, note that for each $j$, either $A_j \subset A$ or $A_j \cap A = \emptyset$. Therefore we let $F = \{q_j \mid A_j \subseteq A\}$.

It is easy to argue by induction that $\delta^*(s, x) = q_j \iff x \in A_j$. This, together with the choice of $F$ ensures that $L(M) = A$. □

By the proof of the corollary above we know that the number of states of $M$ constructed above is the smallest possible. (In fact, if one looks at things carefully one can see that all DFA’s of that size for $A$ have to look the same except for the names of the states.)

However, in general, even though $A$ is a regular language we may not have a nice description of $\equiv_A$ at our disposal in order to build $M$. What happens if all we have is some DFA accepting $A$? That’s the subject of the next handout, Minimizing DFAs.