Notes on the Myhill-Nerode Theorem

The purpose of this note is to give some details of the Myhill-Nerode Theorem and its proof, neither of which appear in the textbook. This theorem will be a useful tool in designing DFAs, as well as in characterizing the regular languages.

Definition 1. *Let $L \subseteq \Sigma^*$ be any language, and $x, y \in \Sigma^*$ be any strings. We say "x is* **equivalent** *to y with respect to L", written $x \approx_L y$ iff, for any $z \in \Sigma^*$,*

$$xz \in L \iff yz \in L$$

Observe: $\approx_L$ is an equivalence relation: it is reflexive, symmetric, and transitive. You should check this to convince yourself.

We will be interested in decomposing a language into its equivalence classes, which we write as:

$$[x] = \{y \mid x \approx_L y\} \text{ is the equivalence class of } x \text{ under } \approx_L$$

For example, consider $L = \{w \mid w \text{ is of even length}\} \subseteq \{a\}^*$. The equivalence classes of L are:

- $[\epsilon] = \{\epsilon, aa, aaaa, aaaaaa, \ldots\} = [aa] = [aaaa] = L$
- $[a] = \{a, aaa, aaaaa, \ldots\} = \{w \mid w \text{ is of odd length}\} = [aaa] = [aaaaaaa] = \cdots$

Notice that $[a] = \bar{L}$ is the complement of L, so this is all the equivalence classes. Also, note that $[\epsilon] = [aa] = [aaaa]$ (and so on), so we can call an equivalence class by many different names.

Theorem 2 (Myhill-Nerode Theorem)**.** *L is regular if and only if $\approx_L$ has finitely many equivalence classes.*

The idea is that each equivalence class will correspond to a state of the DFA. (This makes sense, since if x and y are in the same equivalence class, then for any string z we concatenate to the end, $xz \in L \iff yz \in L$ — that is, we want the DFA to either accept both xz and yz or reject both of them. This will correspond to starting from the same state, and then processing the characters of string z.)

Proof. There are two directions of the "if and only if".

$\Leftarrow$: If L is regular, then there is a DFA recognizing L which has finitely many states. Each state represents an equivalence class (of strings that reach that state). Consider two strings x and y which both finish in some state q_i. Then for any string z, the computation on xz will end up in the same state as the computation for yz, namely, whatever state the DFA reaches when it starts in state q_i and sees string z.

Since there are finitely many states and each state represents an equivalence class, there are finitely many equivalence classes.

$\Rightarrow$: If L has finitely many equivalence classes, then there is a DFA recognizing L with exactly that many states. We can construct it as follows. Define DFA $M = (Q, \Sigma, \delta, q_0, F)$:

$$\begin{aligned} K &= \{[x] \mid x \in \Sigma^*\} \\ q_0 &= [\epsilon] \\ F &= \{[x] \mid x \in L\} \\ \delta([x], \sigma) &= [x\sigma] \text{ for } [x] \in Q, \sigma \in \Sigma \end{aligned}$$

Note: δ is well-defined because $x \approx_L y$ iff $x\sigma \approx_L y\sigma$.

Some observations to make:

- for any string x, it is in some equivalence class $[x]$ and it will end up in the state corresponding to $[x]$
- for any string x, if $x \in L$ then the state corresponding to $[x]$ is a final state (by the construction rule given above), so x will be accepted
- for any string $x \notin L$, the state corresponding to $[x]$ is not a final state. Why?
  A tiny proof-by-contradiction:
  Suppose $x \notin L$ but the state q corresponding to $[x]$ was in F.
  Because $q \in F$, it must be that $[x] = [y]$ for some $y \in L$ (by the construction rule given above for set F).
  If these two equivalence classes are equal, that means $x \approx_L y$ (by definition of equivalence classes).
  Thus for all z, $xz \in L \iff yz \in L$.
  Take $z = \epsilon$. Then we have $x\epsilon = x \in L$ is FALSE but $y\epsilon = y \in L$ is TRUE. Contradiction! $\Rightarrow\Leftarrow$

Thus the DFA given by this construction recognizes the language L. $\square$

Corollary 3. *Let L be a language with $k \in N$ equivalence classes under $\approx_L$. Then every DFA recognizing L has at least k states.*

And note, for L with k equivalence classes, the above construction gives a DFA with exactly k states — a minimal DFA, the smallest one possible.[1]

Practice problem 1: Consider again the example language: $L = \{w \mid w \text{ is of even length}\} \subseteq \{a\}^*$. The equivalence classes of L are:

- $[\epsilon] = \{\epsilon, aa, aaaa, aaaaaa, \ldots\} = [aa] = [aaaa] = L$
- $[a] = \{a, aaa, aaaaa, \ldots\} = \{w \mid w \text{ is of odd length}\} = [aaa] = [aaaaaaa] = \cdots$

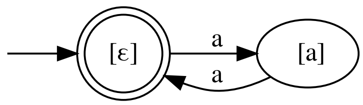
The Myhill-Nerode Theorem says that because L has finitely many equivalence classes[2], it should be a regular language. Can you use the insight of the proof to come up with a (very, very simple) DFA that accepts this language L? (Ideally, you would only have as many states as there are equivalence classes.) Answer on the next page.

Practice problem 2: Consider the language:

$$L = \{w \in \{0,1\} \mid w \text{ represents a number divisible by 3 in binary notation}\}$$

How many equivalence classes does this L have? What are they? Can you come up with a DFA to recognize this language?

[1] Fun thought experiment and proof-writing practice: why would any smaller DFA *not* be able to recognize L?
[2] Check for yourself: how many are there? 2. Sanity check: is 2 finite? Yeah.


[ε]
a
[a]
a