The Myhill-Nerode Theorem: Automata as quotient structures

Remember that a quotient structure provides a "compressed" view of a structure under an equivalence relation. Instead of individual elements of the original structure, the quotient structure only operates with the equivalence classes induced by the equivalence relation. One of the most intriguing results of formal language theory is that FSAs are quotient structures of regular string languages.

In order to see this, we first need a suitable equivalence relation: the **right congruence relation** $\equiv_L$ with respect to some string language L. Let L be some arbitrary set of strings over Σ. Then $u \equiv_L v$ iff u and v have the same **good continuations** with respect to L. That is to say, if we can add some string s to u and get a member of L, then we can also add s to v and get a member of v (and the other way round).

Example 1.
Let $L := (ab)^+$. Consider the string aba. It is not a member of L, but we can add a b at the end, yielding $aba \cdot b = abab \in L$. But there's infinitely many other strings we could've added: bab, $babab$, and so on. These are the good continuations of aba with respect to L.

Now consider the string $ababa$. Like aba, it does not belong to L, but we can add various strings to it to obtain a member of L. But as you can verify for yourself, aba and $ababa$ have exactly the same good continuations. For every string $u \in \Sigma^*$, $abau$ is a well-formed string of L iff $ababau$ is. Hence aba and $ababa$ are right congruent: $aba \equiv_L ababa$.

Exercise 1.
Describe the good continuations of ab with respect to $(ab)^+$ and show that $ab \equiv_L abab$.

Caution: The empty string is a possible continuation, too.

Exercise 2.
Just because two strings are right congruent with respect to some string language L does not mean that they are right congruent with respect to some other string language L'. Suppose that L' is $ababc(ab)^+$. Show that ab and $abab$ are not right congruent with respect to L'.

It is fairly easy to see that $\equiv_L$ is an equivalence relation.

Exercise 3.
Explain why! You'll have to remember which properties a relation has to meet in

order to be an equivalence relation. For each property, explain in intuitive terms why it holds.

Now consider the partition induced by $\equiv_L$. That is to say, we group all the strings into equivalence classes such that the equivalence class $[u]$ contains u and all strings v that are right congruent with u.

Exercise 4.
If $u \equiv_L v$, then $[u] = [v]$. Explain why!

All these equivalence classes are connected by a relation Δ. Let σ be some symbol of our alphabet Σ. We say that $[x]$ is Δ-related to $[y]$ via σ iff $[y] = [x\sigma]$.

Example 2.
The equivalence class $[ab]$ with respect to $\equiv_L$ contains ε, ab, $abab$, and so on. That's because L is $(ab)^+$ and all these strings have the same good continuations with respect to L. The equivalence class $[ab]$ is a-related to $[aba] := \{a, aba, ababa, ...\}$. In addition, $[ab]$ is b-related to $[abb]$, which contains all strings that are not members of L (i.e. $\Sigma^* - L$).

The equivalence classes, combined with Δ, form a quotient structure: instead of adding a symbol a to a string s and then computing the good continuations of $s \cdot a$, one can immediately move from $[s]$ to $[sa]$.

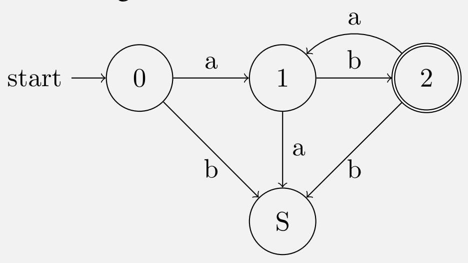
Now here's the cool part: The deterministic FSA for some regular language L is a quotient structure in this sense. The states of the automaton correspond to the equivalence classes in the quotient structure. Two strings are in the same equivalence class if they lead to the same state. For any given state/equivalence class the set of good continuations corresponds to the set of paths that lead from the state to a final state. And Δ above is exactly the same as the transition relation Δ of the FSA. So deterministic FSAs are quotient structures, with states as the counterpart of equivalence classes of strings.

Example 3.
Consider once more the string language $L := (ab)^+$. The right congruence relation $\equiv_L$ partitions the set Σ^* of all strings into four equivalence classes:

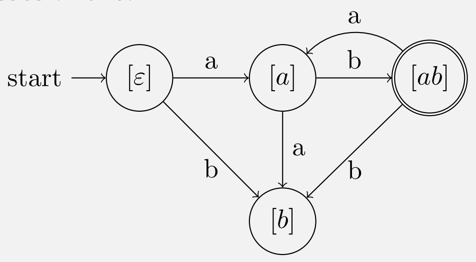
- $[\varepsilon]$ consists only of the empty string, which can be continued with any member of $(ab)^+$,
- $[a] := \{a, aba, ababa\}$ contains all strings that can be continued with b, bab, $babab$, and so on,
- $[ab] := \{ab, abab, ababab\}$ contains all strings that can be continued with the empty string or any member of $(ab)^+$,

- $[b]$ contains strings such as b, abb, $ababbabaa$, i.e. any string that has no good continuations at all.

The automaton for $(ab)^+$ is shown below, with an explicit sink state for ill-formed strings.



We can rename the states to highlight their connection to the equivalence classes above.



Exercise 5.
Describe the partition induced by $\equiv_L$ if $L := (ab)^*$ — that is to say, L contains the empty string, every member of $(ab)^+$, and nothing else. Then draw the automaton for L. Which state corresponds to which equivalence class?

The connection between congruence classes and states of an FSA isn't just a nice mathematical curiosity. It is the foundation of the famous Myhill-Nerode theorem.

Theorem 1. A string language L is regular iff $\equiv_L$ has finite index.

For an equivalence relation to have **finite index** means that it induces only finitely many equivalence classes. So the Myhill-Nerode theorem states that a string language is regular iff its quotient structure under $\equiv_L$ has only finitely many equivalence classes. But remember, these equivalence classes correspond to what we represent as states in an FSA. We already know that a language is regular iff it is recognized by an FSA, and FSAs only have finitely many states, i.e. equivalence classes. This is the connection exploited by the Myhill Nerode theorem.

Let's look at it step by step. If a language is regular, then it must be recognized by some FSA. If so, then $\equiv_L$ induces only finitely many equivalence classes. Here's why this must be the case: consider two arbitrary strings s and t of Σ^*. We will look at how some arbitrary deterministic FSA A behaves with respect to those strings. To simplify things, we will assume that A has a sink. As adding a sink is trivial, this simplifying assumption does not endanger the validity of the argument.

Suppose that the runs $r(s)$ and $r(t)$ of A over s and t end in the same state q. Now if u is a good continuation of s with respect to L, then there must be some path from q to a final state. But then this path is also available for t, because t also ends in q. Hence u must also be a good continuation of t. Therefore any two strings whose run ends in the same state have the same good continuations, and whenever $r(s)$ and $r(t)$ end in the same state it must be the case that $r \equiv_L t$.

Since we added a sink state to A, every run of A over a Σ-string must end in some state. Hence we can partition Σ^* such that $s \equiv_A t$ iff $r(s)$ and $r(t)$ end in the same state. But since A has only finitely many states, there can be only finitely many equivalence classes. And we know from our previous observations that $s \equiv_A t$ implies $s \equiv_L t$. So the fact that $\equiv_A$ has finite index implies that $\equiv_L$ has finite index. This proves one direction of the Myhill-Nerode theorem: if a language L is regular, then $\equiv_L$ has finite index.

The same reasoning can be applied in the other direction to show that whenever $\equiv_L$ has finite index, once can construct an FSA from $\equiv_L$ that recognizes a string s iff $s \in L$.

And here's why this result is important: it allows us to show that some string languages are not regular.

Example 4.
Consider the string language $a^n b^n$. The good continuations of a are b, abb, $aabbb$, and so on, i.e. $a^n b^{n+1}$. The good continuations of aa, on the other hand, are bb, $abbb$, $aabbbb$, i.e. $a^n b^{n+2}$. In general, the good continuations of a^n are $a^m b^{m+n}$. But this entails that $[a]$, $[aa]$, $[aaa]$, ..., are all distinct equivalence classes. Every string of the form a^+ has its own equivalence class. But clearly there are inifnitely many such strings, which means that the quotient structure must contain infinitely many equivalence classes. In other words, $\equiv_L$ does not have finite index, wherefore L is not regular.

Exercise 6.
Use the Myhill-Nerode theorem to show that the following language is not regular if the alphabet Σ contains at least two distinct symbols (e.g. a and b): $\{ww^R \mid w \in \Sigma^*\}$, where w^R is the reverse of w. This language is known as the **palindrome language**. The palindrome language contains strings such as $abccba$ or $aaaa$, but not $abcabc$ or $aaab$.

Exercise 7.
Use the Myhill-Nerode theorem to show that the following language is not regular if

the alphabet Σ contains at least two distinct symbols (e.g. a and b): $\{ww \mid w \in \Sigma^*\}$. This language is known as the **copy language**. The copy language contains strings like *abab*, *aaaa*, or *abbababbab*, but not *abba*, *aabaa*, or *baaaba*.

Exercise 8.
Show that both the palindrome language and the copy language are regular if the alphabet contains only a single symbol, e.g. a.

You might wonder why we need the Myhill Nerode theorem to show that a language is or isn't regular. Isn't it enough to just write down an FSA? If we can't do that, then the language apparently isn't regular. Well, you're half-right. If we can write down an FSA, then the language is regular (assuming our FSA works correctly). But if we can't write down an FSA, nothing much follows. Perhaps there is an FSA and we just haven't found it yet. After all, there's infinitely many FSAs, so there's always the chance that the next FSA we try might be the right one. The Myhill-Nerode theorem acts as a safeguard here. We can use it to show quickly that a language is not regular, potentially saving us days or weeks in search of a working FSA that does not exist.