Guide to the Myhill-Nerode Theorem

The Myhill-Nerode theorem is our go-to tool for proving that languages are not regular. This guide recaps the major definitions needed for the Myhill-Nerode theorem, then describes strategies for finding infinite distinguishing sets.

Core Definitions

Nonregular Languages

The regular languages are the languages for which you can either

- build a DFA that accepts every string in the language and rejects everything else, or
- build an NFA that accepts every string in the language and rejects everything else, or
- write a regular expression that matches all the strings in the language and nothing else.

On the other hand, a ***nonregular*** language is one where you can't do any of the above. You can't build a DFA for it, or design an NFA for it, or write a regex for it.

Distinguishability

At its core, the Myhill-Nerode theorem revolves around the notion of distinguishability. Suppose you have a language L over some alphabet Σ. Two strings x and y are called ***distinguishable*** relative to L when there is a string w such that exactly one of xw and yw is in L.

I like to visualize distinguishable strings in the following manner. To show that x and y are distinguishable relative to some language, draw the strings x and y over one another with a large horizontal line to their right. Your goal is to come up with a string w that you can put to the right of the line so that either (1) $xw \in L$ and $yw \notin L$ or (2) $xw \notin L$ and $yw \in L$.

Let's make this concrete. Consider the language $L = \{w \in \{\texttt{a}, \texttt{b}\}^\star \mid |w| = 4\}$. This is the language of all strings whose length is exactly four. We want to show that the strings `aba` and `bb` are distinguishable relative to L. We'll start off by writing `aba` on top of `bb` with a line to their right.

```
aba |
 bb |
```

We need to pick a single string we can put to the right of the line that puts one of the two resulting strings in L and the other outside of L. Since L is the language of all strings whose length is exactly four, we need a string on the right so that, of the two resulting strings, exactly one of them has length four. There are many ways to do this; here's two. I've added colors to the different strings here so that it's a bit easier to discuss them later:

```
aba | a        aba | bb
 bb | a         bb | bb
```

In the example on the left, we end up with `abaa` (length four) and `bba` (length three), so only one string is in L. In the second example, the strings are `ababb` (length five) and `bbbb` (length four).

You can think of this way of thinking about distinguishability as a "distinguishability game." If you want to know whether two strings are distinguishable relative to a language L, write those strings out as we did above. You win the game if you can find a suffix you can tack on to both strings that puts one in L and keeps the other out of L. In that case, the strings are distinguishable relative to L. You lose the game if, no matter what string you tack onto the end of both x and y, you end up with both strings staying in L or staying out of L. In that case, the strings are *indistinguishable* relative to L.

There are no restrictions on what kinds of strings you're allowed to append to x and y. The string w you append can be very short (even as short as ε) or very long.

Distinguishing Sets

If L is a language over Σ, a ***distinguishing set*** for L is a set $S \subseteq \Sigma^\star$ where the following is true:

$$\forall x \in S. \forall y \in S. (x \neq y \ \rightarrow \ x \not\equiv_L y).$$

Let's clarify what all the pieces here mean. We begin with some language L over an alphabet Σ. A distinguishing set S for L has to be a subset of $\Sigma^\star$, meaning that S itself is going to be a set of strings. (We would be correct in calling it a language over Σ because everything in S is a string over Σ, but for whatever reason we generally don't do that here.) Notice that S is just required to be a subset of $\Sigma^\star$ and that there's no requirement that S is a subset of L.

The key property about S is the one given in the FOL notation above. You can read that formula as saying "for any pair of two different strings x and y in S, the strings x and y are distinguishable relative to L." Stated differently, if you pick any two distinct strings out of S and play the distinguishability game with them, you're guaranteed that you can always win.

Let's see an example. In class, we looked at the language $E = \{\mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N}\}$ over the alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ and saw that $S = \{\mathtt{a}^n \mid n \in \mathbb{N}\}$ is a distinguishing set for E. Let's quickly check that is indeed the case.

First, we have to see whether $S \subseteq \Sigma^\star$. Every string in S is made purely of $\mathtt{a}$'s, and $\mathtt{a} \in \Sigma$, so, yep, it's the case that $S \subseteq \Sigma^\star$. Next, we have to check whether it's really the case that, given any two arbitrary distinct (different) strings chosen out of S, those strings are distinguishable relative to E. To do that, let's try out a few examples. Let's pick $\mathtt{a}^{137}$ and $\mathtt{a}^{103}$. To see if they're distinguishable relative to E, let's play the distinguishability game! We write them out like this:

$$\begin{array}{r|} \mathtt{a}^{137} \\ \mathtt{a}^{103} \end{array}$$

Now, we need to figure out what goes on the right. We need to tack something onto the end of both strings so that one ends up in the language E and one stays outside. Every string in E needs to consist of some number of $\mathtt{a}$'s followed by an equal number of $\mathtt{b}$'s. The two strings we have here already consist of some number of $\mathtt{a}$'s, and so a reasonable guess would be to tack on some number of $\mathtt{b}$'s to the end of each. We should pick the number of $\mathtt{b}$'s in a way that matches one of the number of $\mathtt{a}$'s, since otherwise neither string would consist of a number of $\mathtt{a}$'s followed by an equal number of $\mathtt{b}$'s. Here's two ways we could do this:

$$\begin{array}{r|l} \mathtt{a}^{137} & \mathtt{b}^{137} \\ \mathtt{a}^{103} & \mathtt{b}^{137} \end{array} \qquad\qquad \begin{array}{r|l} \mathtt{a}^{137} & \mathtt{b}^{103} \\ \mathtt{a}^{103} & \mathtt{b}^{103} \end{array}$$

In each case, one of the strings ends up in E, and the other doesn't.

Of course, we've just checked one pair of strings in S here, and the definition of a distinguishing set says that *every* pair of distinct strings must be distinguishable. But the above exercise was still worthwhile because it nicely generalizes. Now suppose we pick two strings $\mathtt{a}^m$ and $\mathtt{a}^n$ out of S, where $m \neq n$. (We don't need to worry about the case where $m = n$ because the strings would be the same in that case, and the definition of distinguishability promises that we only need to look at different strings.) We can set up our distinguishability game like this:

$$\begin{array}{r|} \mathtt{a}^{m} \\ \mathtt{a}^{n} \end{array}$$

And, by generalizing how we won this game in the concrete case, we see two ways to win:

$$\begin{array}{r|l} \mathtt{a}^{m} & \mathtt{b}^{m} \\ \mathtt{a}^{n} & \mathtt{b}^{m} \end{array} \qquad\qquad \begin{array}{r|l} \mathtt{a}^{m} & \mathtt{b}^{n} \\ \mathtt{a}^{n} & \mathtt{b}^{n} \end{array}$$

This argument explains why we S is indeed a distinguishing set for E.

Returning to an earlier point, notice that $S \not\subseteq E$. For example, we have $\mathtt{a} \in S$ even though $\mathtt{a} \notin E$. And that's perfectly fine: nothing in the definition of a distinguishing set says that a distinguishing set for a language must be a subset of the language.

One other point worth mentioning before we move on: notice that to show that S is a distinguishing set, we need to prove that for every pair of distinct strings x and y in S, there's some string w you can tack onto the end of x and y so that exactly one of xw and yw belongs to L. The quantifier ordering here is $\forall x. \forall y. \exists w.$, meaning that we pick x and y arbitrarily first, then get to choose what w is. And, importantly, because of this quantifier ordering, the choice of w can depend on what x and y are. You are not required to pick some w that works for every possible choice of x and y, and indeed in most cases it won't be possible to do that.

Infinite Distinguishing Sets

An ***infinite*** distinguishing set is simply a distinguishing set that contains infinitely many strings. Or, stated differently, S is an infinite distinguishing set for L, it means that $|S|$ is infinite and that S is a distinguishing set for L.

In the preceding section, we say that $S = \{\mathtt{a}^n \mid n \in \mathbb{N}\}$ is a distinguishing set for $E = \{\mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N}\}$. It's also an *infinite* distinguishing set: the set S contains infinitely many strings (specifically, $\mathtt{a}^0$, $\mathtt{a}^1$, $\mathtt{a}^2$, etc.).

Not all distinguishing sets are infinite. Later in this guide, we'll see that finite distinguishing sets can still be useful as a stepping stone toward finding infinite distinguishing sets. If we have enough time this quarter, you might see other applications of finite distinguishing sets either in lecture or on the problem sets. They're certainly useful in Theoryland!

The Myhill-Nerode Theorem

The Myhill-Nerode theorem says the following:

> ***Theorem:*** Let L be a language over Σ. If L has an infinite distinguishing set, then L is not regular.

The Myhill-Nerode theorem is an incredibly powerful tool. If you want to prove that a language is not regular, your first instinct should be to try to use this theorem. Stated differently, if you want to prove a language is nonregular, you should start off by searching for an infinite distinguishing set for that language.

For example, suppose you want to prove that the language $E = \{\mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N}\}$ is not regular. One way to do this would be to come up with the set $S = \{\mathtt{a}^n \mid n \in \mathbb{N}\}$ and prove that it is indeed an infinite distinguishing set for E. Once you've done that, you can immediately apply the Myhill-Nerode theorem to conclude that E is not regular.

In order to do this, you need to be able to find infinite distinguishing sets for a given language. That's a skill that you can improve over time, and it's the focus of the rest of this guide.

Finding Distinguishing Sets

To use the Myhill-Nerode theorem to prove that a language L is not regular, you need to find an infinite distinguishing set S for L. There are a lot of ways you can come up with these sets. Some degree of trial-and-error is likely required, but you can be more principled than just guessing random sets and hoping they work. This section introduces some useful techniques, each of which is accompanied by a worked example that shows the principle in action.

Start Small

Your goal is to come up with an infinite distinguishing set - a set of infinitely many strings where any two different strings in the set are distinguishable. Finding infinitely many strings is going to be tough if you can't even find a single pair of distinguishable strings. So one way to approach finding an infinite distinguishing set is to start off with two distinguishable strings, then find a third string distinguishable from each of those, then a fourth string distinguishable from the three others, etc. until you find a pattern that generalizes.

Let's see an example of this. Let $\Sigma = \{\mathtt{a}, \geq\}$ and consider this language L:

$$L = \{\mathtt{a}^m \geq \mathtt{a}^n \mid m, n \in \mathbb{N} \wedge m \geq n\}.$$

This language consists of strings made up of two groups of **a**'s, with a $\geq$ symbol separating them, such that there are at least as many **a**'s in the first group as there are in the second. So, for example, the string **aaaa** $\geq$ **aa** $\in L$ and **aaa** $\geq$ **aaa** $\in L$, as are **aa** $\geq$ and $\geq$ (in those cases, we have groups of zero **a**'s, which are fine). However, the string **aa** $\geq$ **aaa** is not in L (there's more **a**'s in the second group than the first), and the string **aa** $\geq$ **aa** $\geq$ **aa** is not in L (strings in L must specifically consist of some number of **a**'s, then a single $\geq$, and then a second group of **a**'s). It turns out that this language L is not regular, and our goal is going to be to prove this.

Let's begin by looking for two strings that are distinguishable relative to this language L. To do this, let's pick two strings or more less at random and see if they're distinguishable. Let's begin with **aaaa** $\geq$ **aa** and **aaa** $\geq$ **a**. Are these distinguishable relative to one another? To find out, let's set up our distinguishability game:

aaaa $\geq$ **aa**	
aaa $\geq$ **a**	

What could we tack onto the ends of these strings? If we append a $\geq$ to both strings, then neither ends up in the language; every string in L has exactly one $\geq$ symbol in it. If we append **a** to both strings, then they each stay inside the language. The first one becomes **aaaa** $\geq$ **aaa** and the second one is **aaa** $\geq$ **aa**, both of which are in L. If we append **aa** to both strings, they're both still in the language. If we append **aaa**, then they're both not in the language - the first one becomes **aaaa** $\geq$ **aaaaa** and the second one becomes **aaa** $\geq$ **aaaa**. And, with a little thought, we can see that appending any more **a**'s than this will make both strings not be in the language. So it looks like we just picked two strings that actually were *indistinguishable* relative to L. Oops.

But not to worry! We tried a thing and it didn't work, so let's reflect on why it didn't work and try something else. Looking at the two strings we picked, we can notice that, in retrospect, we accidentally picked two strings where the relative numbers of **a**'s on the sides of the $\geq$ were the same. That's why, regardless of how many **a**'s we tacked on the end, we ended up with both strings in L or both strings not in L.

With that in mind, let's try picking two other, similar strings where the relative difference in **a**'s in the two strings isn't the same. Let's try **aaaa** $\geq$ **aa** and **aaaa** $\geq$ **aaa**. Are these distinguishable? Once again, let's set up our game:

aaaa $\geq$ **aa**	
aaaa $\geq$ **aaa**	

As before, we can't tack $\geq$ onto the ends of these strings, since they already have their needed $\geq$'s. But maybe tacking on some number of **a**'s will do it? With a little thought, we can come up with this:

aaaa $\geq$ **aa**	**aa**
aaaa $\geq$ **aaa**	**aa**

Success! These strings are distinguishable. That's good progress.

We've just found two distinguishable strings. Can we find a third? The two strings we have above have four **a**'s on one side of the $\geq$ and a smaller number of **a**'s on the other. Maybe we could find another string of that form? With a little experimentation, we'll find that any other number of **a**'s in the right-hand group will work. So let's add to our two other strings a third string, say, **aaaa** $\geq$ **a**. We can see that it's distinguishable from the other two strings:

aaaa $\geq$ **aaa**	**aa**		**aaaa** $\geq$ **aa**	**aaa**	
aaaa $\geq$ **a**	**aa**		**aaaa** $\geq$ **a**	**aaa**	

Okay, we now have three distinguishable strings. Could we find a fourth? Sure - **aaaa** $\geq$ **aaaa** works. Here's how to distinguish it from the three other strings:

aaaa $\geq$ **aaa**	**a**		**aaaa** $\geq$ **aa**	**a**		**aaaa** $\geq$ **a**	**a**	
aaaa $\geq$ **aaaa**	**a**		**aaaa** $\geq$ **aaaa**	**a**		**aaaa** $\geq$ **aaaa**	**a**	

It's not too hard to see that we could also toss in **aaaa** $\geq$ into the mix. That would be distinguishable from the others as well. That gives us five distinguishable strings. And it turns out that **aaaa** $\geq$ **aaaaa** is also distinguishable from all the other strings in the group. Take a second to explore - do you see why?

So we now have this group of distinguishable strings:

aaaa $\geq$ **aaaa** $\geq$ **a** **aaaa** $\geq$ **aa** **aaaa** $\geq$ **aaa** **aaaa** $\geq$ **aaaa** **aaaa** $\geq$ **aaaaa**

All of these strings start with four **a**'s and a $\geq$ sign. Unfortunately, we can't add any more strings of that form into this set to get a bigger one. For example, **aaaa** $\geq$ **aaaaaa** is indistinguishable from **aaaa** $\geq$ **aaaaa**. No matter what we tack onto the ends of both of those strings, we end up with something not in the language L.

But that doesn't mean we've hit a dead-end. We ended up finding these strings by looking for strings with different imbalances between the number of **a**'s to the left and to the right of the $\geq$ sign. The fact that we picked strings that all start with **aaaa** is an artifact of our original exploration rather than something deep and intrinsic to the strings we picked. If we step away from the **aaaa** business and instead focus on finding strings with different imbalances, perhaps we'll have some more luck.

For example, let's look at our string **aaaa** $\geq$ **aaaa**. This is a string where there's an imbalance of 0 between the left and right sides. Lots of strings have an imbalance of zero; **aaa** $\geq$ **aaa** also works, as does **aa** $\geq$ **aa**. Is there a "simplest" string showing an imbalance like this? The answer is yes - it would be $\mathtt{a}^0 \geq \mathtt{a}^0$, which is just a fancy way of writing out the string $\geq$.

What about strings with an imbalance of 1 between the left and right groups? The simplest such string here would be $\mathtt{a}^1 \geq \mathtt{a}^0$, which is the string **a** $\geq$. And notice that $\mathtt{a}^0 \geq \mathtt{a}^0$ is distinguishable from $\mathtt{a}^1 \geq \mathtt{a}^0$, since we can tack **a** onto the end of both.

What about a string with an imbalance of 2? We could pick $\mathtt{a}^2 \geq \mathtt{a}^0$, or, more compactly, **aa** $\geq$. That string is distinguishable from the two previous strings - do you see why?

If we look at *these* strings, we can see that there is a nice pattern that will let us find an infinite distinguishing set. Simply look at all strings of the form $\mathtt{a}^n \geq$ for different values of n. We can use that to define our distinguishing set:

$$S = \{\mathtt{a}^n \geq \mid n \in \mathbb{N}\}.$$

The intuition for why this works is that given any two strings with different imbalances, we can tack on a number of **a**'s to both strings that will break the inequality in one of the two strings but not in the other.

I wanted to walk you through the process of getting here because I think it's helpful to see how things like this actually get worked out. We didn't come up with this set by starting at the definition of L, thinking really hard, and immediately coming up with it. Instead, we started with some small strings, found a modest-sized distinguishing set, realized that it wouldn't generalize to infinitely many strings, saw a different pattern that did generalize, and eventually used that pattern to make our infinite distinguishing set. When you're first starting off with Myhill-Nerode arguments, it's common to need to do this sort of trial and error! Over time you'll start picking up on patterns that will let you guess more effectively on unknown problems, but even in those cases usually some experimentation is required.

Now that we have our distinguishing set, how might we formally prove that L is nonregular? Essentially, we just need to argue why S is an infinite distinguishing set, then cite the Myhill-Nerode theorem. Here's one way to do that:

Theorem: L is nonregular.

Proof: Consider the set $S = \{\mathtt{a}^n \geq \mid n \in \mathbb{N}\}$. We claim that S is an infinite distinguishing set for L.

To see that S is infinite, note that S contains one string for each natural number.

To see that S is a distinguishing set, pick any two distinct strings $\mathtt{a}^m \geq \mathtt{a}^n \geq \in S$. We

need to show that $\mathtt{a}^m \geq\!\!\!\not\equiv_L \mathtt{a}^n \geq$. To see this, assume, without loss of generality, that $m > n$. Then we see that $\mathtt{a}^m \geq \mathtt{a}^m \in L$ but that $\mathtt{a}^n \geq \mathtt{a}^m \notin L$ (since $n < m$), so $\mathtt{a}^m \geq\!\!\!\not\equiv_L \mathtt{a}^n \geq$, as required.

Since S is an infinite distinguishing set for L, by the Myhill-Nerode theorem L is not regular. ■

Notice how this proof argues that S is infinite and explains why any two strings in S are distinguishable relative to L. Commonly, the justification as to why S is infinite will be relatively straightforward, as is the case here. The justification that S is a distinguishing set, though, may be more elaborate. Here, we pick two arbitrary strings in S and then find a concrete string we can tack onto the end of those strings to distinguish them.

You might notice that the strings we picked initially all happened to be in L, and our final distinguishing set S is a subset of L. That is purely a coincidence. There is nothing requiring that we do this. In fact, in general, you won't necessarily be able to find a distinguishing set for a language that is a subset of that language. The next section talks about an example of this and what to do in the (very common) case where that happens.

What "Must" You Remember?

Let's work through another example. Here, we'll let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ and will consider this language L:

$$\{w \in \Sigma^\star \mid w \text{ has the same number of } \mathtt{a}\text{'s and } \mathtt{b}\text{'s}\}.$$

We're going to show that this language is not regular, and we'll do so by finding an infinite distingushing set for it.

Let's begin by looking for any pair of distinguishable strings. In the previous section, we picked some sample strings from the language as our starting point. Let's try that here to see what happens. How about, say, `abba` and `aaabbb`? Are they distinguishable relative to L? Alas, no. The string `abba` has two `a`'s and two `b`'s, and `aaabbb` has three of each. Whatever string we tack onto the end of each will increase the relative counts of `a`'s and `b`'s the same way. And "relative" is the key word here - the language L doesn't care about precisely how many `a`'s and `b`'s there are in the string, but does care whether they're the same. So if each string starts with the same relative frequencies of `a`'s and `b`'s, and we append the same blend of characters to each, then the two strings will end up with the same relative frequencies as one another. Thus either both will be in L, or both won't be in L.

This immediately signals something to us - we're going to have to look for strings that are not in L if we're going to find an infinite distinguishing set.

This is surprisingly common. In fact, that's the case for the very first nonregular language we looked at, $E = \{\mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N}\}$. The distinguishing set we came up with, $S = \{\mathtt{a}^n \mid n \in \mathbb{N}\}$, is not a subset of E. It only contains one element that is also in E, and that's $\mathtt{a}^0 = \varepsilon = \mathtt{a}^0\mathtt{b}^0$, which is essentially only in E by pure coincidence.

If you have to pick strings not in L to build a distinguishing set, where should you start? You could, if you wanted to, try random strings to see what works and what doesn't, as we did in the previous example, and often times that will lead you in a promising direction. But it would be nicer to have something a bit more principled serving as a guide.

Here's a very useful technique for building distinguishing sets. Regular languages are those for which you can build a DFA. A DFA reads the characters of an input string from left to right, then, only at the end, decides whether to accept or reject. Now, a DFA only has finitely many states (the "F" in "DFA" stands for "finite"). However, there are infinitely many possible strings you can feed into a DFA. This means that as the DFA is processing the string it sees from left to right, it can't "remember" what string it's seen. Its only memory is which state it happens to be in, and there's more strings than states. Therefore, the DFA has to operate by "summarizing" the string it's seen thus far and just remembering that summary.

For example, take a look at the DFA we built in lecture that checks whether a string

contains `aa` as a substring. It has three states:

- q_0, which holds strings where we haven't seen `aa` yet and where the most recent character isn't `a`.
- q_1, which holds strings where we haven't seen `aa` but have just seen an `a`.
- q_2, which holds strings containing `aa` as a substring.

As the DFA operates, scanning the characters of the string from left to right, it essentially "summarizes" or "remembers" only one fact about the string it's seen thus far: which of the three cases above apply to it.

You can make a similar sort of argument about *any* DFA. Each state corresponds to some fact that must be "remembered" about the string that's been seen thus far. What that information is will depend on the choice of the language and the way the DFA is built.

Now, let's return to our language L from above, which is reprinted here:

$$\{w \in \Sigma^\star \mid w \text{ has the same number of } \texttt{a}\text{'s and } \texttt{b}\text{'s}\}.$$

Our goal is to show that this language isn't regular, which means that we need to show there's no DFA for it. But for the moment, let's pretend we didn't know that no DFA exists for L and instead think about how we might try to build such a DFA. How would it need to work? Well, at each point in time, it would need to somehow summarize the string it had seen thus far. What might that summary look like? The DFA wouldn't necessarily need to remember the exact string it had read - that's too specific. However, it *would* need to know something about the relative frequencies of `a`'s and `b`'s of what it's seen thus far. If the DFA couldn't remember that information, then upon seeing the rest of the string, it would have no way of knowing whether the counts of `a`'s and `b`'s are the same. In that sense, the DFA would "have to remember" the relative frequencies of `a`'s and `b`'s of the strings it processes.

What can we do with this knowledge? We are ultimately interested in building a distinguishing set for L. We also have some idea of what information a DFA for L would "have" to remember. We can therefore use the following, powerful intuition for building distinguishing sets:

> Choose each string in the distinguishing set to correspond to different pieces of information that "must" be remembered.

For example, the relative frequencies of the `a`'s and `b`'s in a string could be 0 (same number of `a`'s and `b`'s), +1 (one more `a` than `b`), -1 (one more `b` than `a`), +2 (two more `a`'s than `b`'s), -2 (two more `b`'s than `a`'s), etc. There's infinitely many different possibilities for what the relative delta could be. To build our distinguishing set, let's pick a set of infinitely many strings, each of which has a different relative balance of `a`'s and `b`'s. Each one of those strings will require that a DFA "remember" a different fact about it (specifically, the relative balance of `a`'s and `b`'s), and intuitively that should make them distinguishable.

One simple way to do this is to pick $S = \{\texttt{a}^n \mid n \in \mathbb{N}\}$. The string $\texttt{a}^n$ has a relative balance of $+n$, so each string has a different "fact" that must be remembered.

Does this make what we have a distinguishing set? Let's find out! Suppose we pick $\texttt{a}^m, \texttt{a}^n \in S$ where $m \neq n$. We can set up our distinguishability game below:

$\texttt{a}^m$	
$\texttt{a}^n$	

To win the game, we need to tack something onto the end of both strings to put one in L and to keep the other out. There's lots of ways to do this, but the simplest two options are these:

$\texttt{a}^m$	$\texttt{b}^m$
$\texttt{a}^n$	$\texttt{b}^m$

$\texttt{a}^m$	$\texttt{b}^n$
$\texttt{a}^n$	$\texttt{b}^n$

So it looks like, indeed, we have a distinguishing set - and an infinite one to boot!

Here's a formal proof that L is not regular that uses this distinguishing set.

> ***Proof:*** Let $S = \{\mathtt{a}^n \mid n \in \mathbb{N}\}$. We will prove that S is an infinite distinguishing set for L. By the Myhill-Nerode theorem, this proves L is not regular.
>
> To see that S is infinite, note that it contains one string for each natural number.
>
> To see that S is a distinguishing set for L, pick any distinct strings $\mathtt{a}^m, \mathtt{a}^n \in S$. We will show there is a string w such that $\mathtt{a}^m w \in L$ but $\mathtt{a}^n w \notin L$.
>
> Specifically, pick $w = \mathtt{b}^m$. Then we see that $\mathtt{a}^m w = \mathtt{a}^m\mathtt{b}^m \in L$ because it has m $\mathtt{a}$'s and m $\mathtt{b}$'s. On the other hand, $\mathtt{a}^n w = \mathtt{a}^n\mathtt{b}^m \notin L$ because it has n $\mathtt{a}$'s but $m \neq n$ $\mathtt{b}$'s. ■

"Hey wait a minute," you might be saying, "isn't that the same distinguishing set that we used for $E = \{\mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N}\}$?" Yes. Yes it is. That's not a problem - the same set can serve as a distinguishing set for multiple languages.

"Well, in that case, shouldn't we always try to use the set $S = \{\mathtt{a}^n \mid n \in \mathbb{N}\}$ as a distinguishing set? The answer is "it depends on what language you're working with." It's not bad to try guessing S will work just to see what happens, but that doesn't mean it's always going to work. There are many languages for which this choice won't work, and others for which it will, but where the proof is going to be gnarly.

Let's see an example of such a language.

Consider Changing Distinguishing Sets

In this section, let's pick the following language L over $\Sigma = \{\mathtt{a}\}$:

$$L = \{\mathtt{a}^{2^n} \mid n \in \mathbb{N}\}.$$

This is the language of all strings of $\mathtt{a}$'s whose lengths are powers of two. Thus we have $\mathtt{a} \in L$ $(2^0 = 1)$, and we have $\mathtt{aa} \in L$ $(2^1 = 2)$, and we have $\mathtt{aaaaaaaa} \in L$ $(2^3 = 8)$, etc. It turns out that this language isn't regular, and our goal will be to prove that.

Hot on the heels of the previous example, suppose we decide to try using $S = \{\mathtt{a}^n \mid n \in \mathbb{N}\}$ as a distinguishing set. I mean, it worked for two other languages, so I suppose it'll probably work here too, right?

The answer is "yes, technically." To see why, let's pick a few example strings and think about how to distinguish them. Let's take, say, $\mathtt{a}^6$ and $\mathtt{a}^{14}$. How might we distinguish them? There are lots of ways we can do this. Here's two:

$\mathtt{a}^6$	$\mathtt{a}^{10}$	$(16 = 2^4)$
$\mathtt{a}^{14}$	$\mathtt{a}^{10}$	(24 is not a power of two)
$\mathtt{a}^6$	$\mathtt{a}^{18}$	(24 is not a power of two)
$\mathtt{a}^{14}$	$\mathtt{a}^{18}$	$(32 = 2^5)$

In each case, we're adding an amount to bump one of the two numbers up to a power of two, but not the other.

However, the actual choice of how much to add is more subtle than it looks. The nearest powers of two to 6 and 14 are, respectively, $2^3 = 8$ and $2^4 = 16$. But we can't append $\mathtt{a}^2$ to the two strings, the amount needed to bump each up to its next power of two, because if we did that, we'd end up with both strings having lengths that were powers of two (8 and 16, respectively). The other choices shown above work, though.

Let's peek ahead to what would happen if we wanted to use this choice of S as our infinite distinguishing set. Given two strings $\mathtt{a}^m$ and $\mathtt{a}^n$, where $m \neq n$, we'd need to somehow prove there was a number k where exactly one of m and n is a power of two. It's certainly possible to do this (it's a true statement), but this is decidedly nontrivial. (We've left it as an exercise here if you'd like to try it!)

What we've encountered here is a common case when finding distinguishing sets. We've found a distinguishing set that works, but the justification that it is actually a distinguishing set is challenging. We're thus left at a junction: we could either press ahead and try to prove it's a distinguishign set, or we can change our distinguishing set to try to

make the proof easier. It's hard to know which of the two is the easier route, and some amount of guesswork is involved. A reasonable idea would be to spend a few minutes trying to get the proof to work, and if that doesn't work, a few minutes finding another distinguishing set, alternating until we get something that works.

For the purposes of this example, let's try doing some searching to find a better distinguishing set. Earlier on, we talked about how sometimes strings in the language L can serve as elements of a distinguishing set, while in other cases that's not possible. We haven't specifically looked at elements of L yet when trying to find distinguishable strings, so let's try that out.

Suppose we pick $\mathtt{a}^4$ and $\mathtt{a}^{16}$. To distinguish these, we need to append some string to both that makes one of them have a length that's a power of two and the other a length that isn't a power of two. How might we do that? Well, looking at $\mathtt{a}^4$, we can see that the next power of two after 4 is 8. So let's try appending $\mathtt{a}^4$. That would do the following:

$\mathtt{a}^4$	$\mathtt{a}^4$	$(8 = 2^3)$
$\mathtt{a}^{16}$	$\mathtt{a}^4$	$(20 = 4 \cdot 5$ is not a power of two $)$

Hey, that works!

Let's try another pair of strings - this time, $\mathtt{a}^{32}$ and $\mathtt{a}^{128}$. What do we append this time? Let's try the same technique we did above - take one of the strings and bump it up to the next power of two. The next power of two after 32 is 64, so let's append $\mathtt{a}^{32}$ to both strings:

$\mathtt{a}^{32}$	$\mathtt{a}^{32}$	$(64 = 2^6)$
$\mathtt{a}^{128}$	$\mathtt{a}^{32}$	$(160 = 5 \cdot 32$ is not a power of two $)$

Huh, this seems to be working! Let's try one last example: $\mathtt{a}^1$ and $\mathtt{a}^{256}$. If we try the same technique as before, we would try to bump 1 up to the next power of two (2) by adding 1:

$\mathtt{a}^1$	$\mathtt{a}^1$	$(2 = 2^1)$
$\mathtt{a}^{256}$	$\mathtt{a}^1$	$(257$ is not a power of two $)$

Okay, it seems like we're on to something! It looks like every pair of powers of two we look at can be distinguished by appending the shorter of the two strings to both. That might counsel us to pick $S = \{\mathtt{a}^{2^n} \mid n \in \mathbb{N}\}$, or, stated differently, to pick $S = L$. Can we do that? Sure! Nothing in the definition of a distinguishing set says that the distinguishing set has to be different than the language it's a distinguishing set for.

Now, to justify why this is a distinguishing set, we should come up with a general procedure to distinguish $\mathtt{a}^{2^m}$ from $\mathtt{a}^{2^n}$. Our idea is to take the shorter of the two (suppose $m < n$ here) and append it to both strings. We now have the strings $\mathtt{a}^{2^m}\mathtt{a}^{2^m}$ and $\mathtt{a}^{2^n}\mathtt{a}^{2^m}$. From the above examples, it seems like the first of these should have a length that's a power of two and the second should not. Now we need to see why that is.

First, let's look at $\mathtt{a}^{2^m}\mathtt{a}^{2^m}$. We can rewrite this as

$$\mathtt{a}^{2^m}\mathtt{a}^{2^m} = \mathtt{a}^{2^m + 2^m} = \mathtt{a}^{2 \cdot 2^m} = \mathtt{a}^{2^{m+1}}.$$

This string's length is a power of two, so it's in L.

Now, let's look at $\mathtt{a}^{2^n}\mathtt{a}^{2^n}$. We can rewrite this string as

$$\mathtt{a}^{2^n}\mathtt{a}^{2^m} = \mathtt{a}^{2^n + 2^m} = \mathtt{a}^{2^m \cdot (2^{n-m} + 1)}.$$

We know that $m < n$, which means that $n - m > 0$, and since m and n are integers, that means $n - m \geq 1$ and in turn that $n - m - 1 \geq 0$. We can therefore continue rewriting this string as follows:

$$\mathtt{a}^{2^m \cdot (2^{n-m} + 1)} = \mathtt{a}^{2^m \cdot (2 \cdot 2^{n-m-1} + 1)}.$$

Notice that that last term is an odd number: it's of the form $2k + 1$ for some integer k. (Specifically, since $n - m - 1 \geq 0$, we know that $n - m - 1$ is an integer, so 2^{n-m-1} is as well.) Moreover, it's an odd number greater than 1, since $2^{n-m-1} \geq 2^0 = 1$. That in turn means that $2^m \cdot (2 \cdot 2^{n-m-1} + 1)$ can't be a power of two - it has an odd divisor other than 1, and the only odd number that divides a power of two is 1. So after a lot of math and wrangling, we've shown that this string isn't in L!

The math here is definitely not trivial, but it's much easier than what we'd have to do if we were stuck with our original distinguishing set. We're able to heavily lean on the fact that the strings' lengths are powers of two to remove a bunch of cases and make the factorization of the products easier to do. As you're working with distinguishing sets, keep in mind that sometimes it's easier to change the shape of the strings in your distinguishing set to have a much more restrictive shape than it is to try to handle a ton of annoying or odd cases in the proof that what you have is indeed a distinguishing set.

Here's a proof showing how we'd argue L is nonregular:

> ***Proof:*** We will show that L is an infinite distinguishing set for itself, which, by the Myhill-Nerode theorem, proves L is nonregular. We thus must show that L is infinite and that L is a distinguishing set for itself.
>
> To see that L is infinite, note that it contains one string for each natural number; amely, for each $n \in \mathbb{N}$, it contains the string $\mathtt{a}^{2^n}$.
>
> To see that L is a distinguishing set for itself, pick any distinct strings $\mathtt{a}^{2^m}, \mathtt{a}^{2^n} \in L$ and assume without loss of generality that $m < n$. We will show that $\mathtt{a}^{2^m}\mathtt{a}^{2^m} \in L$ and that $\mathtt{a}^{2^n}\mathtt{a}^{2^m} \notin L$.
>
> First, focus on $\mathtt{a}^{2^m}\mathtt{a}^{2^m}$. Notice that
>
> $$\begin{aligned} \mathtt{a}^{2^m}\mathtt{a}^{2^m} &= \mathtt{a}^{2^m + 2^m} \\ &= \mathtt{a}^{2 \cdot 2^m} \\ &= \mathtt{a}^{2^{m+1}}, \end{aligned}$$
>
> which means that $\mathtt{a}^{2^m}\mathtt{a}^{2^m} \in L$.
>
> Now, focus on $\mathtt{a}^{2^n}\mathtt{a}^{2^m}$, which we can rewrite as $\mathtt{a}^{2^m + 2^n}$. Earlier we assumed that $m < n$, which means that there exists a natural number k such that $n = m + k + 1$. We thus see that
>
> $$\begin{aligned} 2^n + 2^m &= 2^{m+k+1} + 2^m \\ &= 2^m \cdot (2^{k+1} + 1) \\ &= 2^m \cdot (2 \cdot 2^k + 1). \end{aligned}$$
>
> We note that $2 \cdot 2^k + 1$ is odd. Moreover, since $2^k \geq 2^0 = 1$, we see that $2 \cdot 2^k + 1 \geq 3$. Therefore, $2^n + 2^m$ has an odd divisor greater than or equal to three, which means that $2^n + 2^m$ is not a power of two. Thus $\mathtt{a}^{2^n + 2^m} \notin L$, as required. ■

This is definitely a more involved proof than the other ones we've done thus far, but if you zoom out a bit it's still arguing that we found our infinite distinguishing set.

Exercises

Looking for some more practice with the Myhill-Nerode theorem? Feel free to work through the following (optional, not collected for a grade) exercises.

i. Let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ and let L be the following language over Σ:

$$L = \{w \in \Sigma^\star \mid |w| \text{ is odd and the middle character of } w \text{ is } \mathtt{a}\}.$$

Prove that L is not regular.

Solution

---

ii. Let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ and consider the finite language $L = \{\varepsilon, \mathtt{a}\}$. Draw the smallest possible DFA for L. By "smallest," we mean "using as few states as possible." Then, prove that every DFA for L has at least as many states as your DFA. To do so, use the following theorem, which you can use without proof: if S is a distinguishing set for L containing finitely many strings, then any DFA for L must have at least $|S|$ states.

Solution

iii. Let Σ be an arbitrary alphabet and let L be an arbitrary language where $L \neq \Sigma^\star$ and $L \neq \emptyset$. Prove that L has a distinguishing set of cardinality two.

Solution

iv. The language $E = \{\mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N}\}$ is the poster child of a nonregular language. This language has another interesting property not shared by all nonregular languages.

Let $L \subseteq E$ be a language where $|L|$ is infinite. Prove that L is not regular.

Solution

v. Let $\Sigma = \{\mathtt{a}\}$. Prove that $S = \{\mathtt{a}^n \mid n \in \mathbb{N}\}$ is a distinguishing set for $L = \{\mathtt{a}^{2^n} \mid n \in \mathbb{N}\}$. (This shows we weren't kidding earlier when we said that it's possible to use this as a distinguishing set but that it's nontrivial.)

Solution

Website programming by Julie Zelenski with minor edits by Keith Schwarz and Sean Szumlanski. This page last updated 2024-Dec-04.