Nonregular Languages

Alyssa Lytle

Fall 2025

1 A Motivating Proof

We've talked a lot about Regular Languages, and specifically how to *prove* they are regular. However, what if they're not? Can we prove that?

The answer is yes! And that is what we will be talking about!

Example 1

Let $B = \{a^n b^n \mid n \geq 0\}$

In other words, it would be the set of strings with a number of as followed by the same number of bs. One challenge of this is that is has to *remember* how many as it has seen by the center point and then confirm that it has the same number of bs. Additionally, it has to be able to do this for an *arbitrarily large* string!

We can prove this using a high-level (less-structured) *proof by contradiction.*

We are going to assume B *is* a regular language and prove that this leads to a contradiction!

So, going off this idea, that means there exists a DFA M such that $B = L(M)$.

We know that an automaton has a finite set of states Q. Let's say $|Q| = K$. Since we are looking at strings of the form $a^n b^n$ for arbitrarily large n, this means that there will inevitably be strings where $n > k$. In other words, there are more *inputs* than *states* which, by the pigeonhole principle, means one state must be visited more than once!

So, let's consider this case of an input of the form $a^n b^n$ with a very large n (specifically, greater than k).

It'll have to have a start state s and end at an accept state $r \in F$. You can see this depicted in Figure 1.

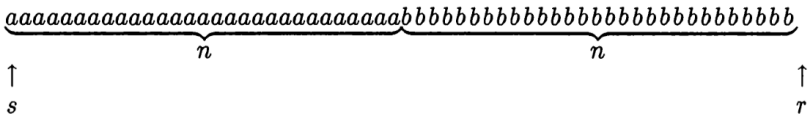

Figure 1: Example of input with a very large n.
[Koz07]

And now, let's consider this fact that at least one state must be visited more than once. Let's call that state p. We can also represent our string $a^n b^n = uvw$ where after the last character of u and before the first character of w, we are at state p. Additionally, since there are n, many a's in this string, and $n > k$, we are safe to define it so that both instances of state p appear over the part of the string that is strictly as, as shown in Figure 2.

Since v begins in and ends in the same state, then we should be able to delete it and still have an acceptable string. However, this isn't the case because if we delete v, then we have more bs than as which should *not* be an accepted string for this language! Therefore, we have a contradiction. $\rightarrow\leftarrow$

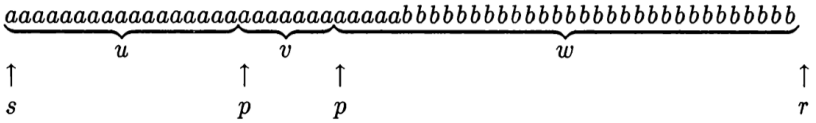

Figure 2: Example of input broken into substrings u, v, and w.
[Koz07]

1.1 The Pumping Lemma

This idea of being able to "delete" a middle part of the input and still get an accepted string actually has a name. It's called The Pumping Lemma. It comes from the idea that smaller strings like v that start and end in the same state can be "pumped" into an input an arbitrary number of times and it still will end at the same accept state.

Lemma 1: The Pumping Lemma

If A is a regular language, then there is a number p (the pumping length) where if s is any string in A of length at least p, then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. for each $i \geq 0$, $xy^iz \in A$
2. $|y| > 0$
3. $|xy| \leq p$

[Sip96]

References

[Koz07] Dexter C Kozen. *Automata and computability*. Springer Science & Business Media, 2007.

[Sip96] Michael Sipser. Introduction to the theory of computation. *ACM Sigact News*, 27(1):27–29, 1996.