

# Awkward State Machines

Will Dengler

April 6, 2019

## 1 Introduction

Imagine you had a ball of yarn of infinite length and a pair of scissors. Now choose some arbitrary length of yarn and cut it - we'll refer to this length as 1. Next, cut a second strand of yarn that has length 2 relative to your original piece of yarn and set it aside. Now, extend the yarn to length 3 and repeat the following instructions:

1. For every strand of yarn you have cut so far (other than the strand of length 1), check if the current extension of yarn can be split into even segments of the same length as the current strand by 'walking' the shorter strand up the longer one until you reach the end of or pass the end of the extended piece of yarn; if you reach the end of the extended piece exactly, then the extension can be divided into even segments. If none of the previous strands can be used to divide the current extension evenly, then double the length of the extended piece of yarn, then cut it in half, finally, set the cut strand aside with the others. Do not cut the extension if one of the previous strands does divide the extension evenly.
2. Using your strand of length 1, increase the length of the extension by 1.
3. Repeat the above two steps.

If you follow the above instructions, then order the strands of yarn you cut by their length, and finally wrote out their lengths relative to the strand of length 1, then you will find yourself writing down the prime numbers in consecutive order. The fact that this simple experiment derives the prime numbers using relative distance has always fascinated me. I've always had an itching notion that an algorithm based solely on relative distance, rather

than integer arithmetic, would outperform the standard methods for discovering the primes in consecutive order. However, the problem of how to encode distance and divisibility without actually using numbers seemed to be impossible; after all, how do you tell a computer to use a ball of yarn and scissors? And even if we could, the act of walking the strands up the extension is going to be pretty slow.

Let's modify our experiment slightly to include tacks and a corkboard. Start again by cutting some length of yarn for defining length 1. Now, using your piece of yarn, separate two tacks on your board 1 unit away from one another. You can now determine the strand of length two by wrapping your ball of yarn around the tacks such that you start at one tack, wrap around the second, and then return to the first tack, then cut the yarn to produce the strand of length two. Now use the strand of length two to place two more tacks in your board at 2 units apart. Now using the tacks for length 1, create a strand of length 3 and repeat the following instructions:

1. For every pair of tacks on the board (other than the unit tacks), wrap the current extension of yarn around the tacks to check for divisibility. The extension is divisible if the yarn perfectly touches one of the tacks when it runs out of length. If none of the current pairs of tacks divide the extension evenly, then double the extension, cut it in half, and use the new strand to set apart a new pair of tacks on your board.
2. Use the unit pair of tacks to increase the length of the extension by 1.
3. Repeat the above two steps.

The above experiment is very similar to the first. The pairs of tacks you've placed will wind up enumerating the primes once again. We are also still stuck with the problem of how could we encode this algorithm without integers, and the algorithm still isn't performing very quickly. However, the algorithm does highlight an important concept, our tacks use circles (technically ovals) in order to check the length of the extended piece of yarn. This is amazing because it allows us to check for divisibility without counting anything out, instead, we just keep wrapping the yarn around until we get to the end; thus we don't need to know *how* many times we've wrapped the yarn around, we just have to look where it winds up at the end.