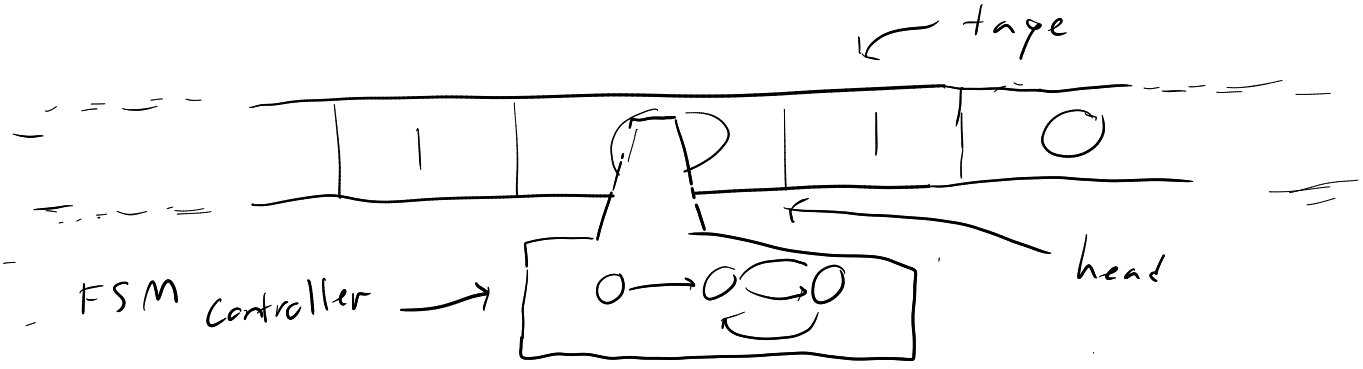
7/28/21 Justin Meiners

Almost Solving The Halting Problem

The halting problem is to devise a method which can determine whether a given Turing machine terminates in a finite amount of time. It is well known to be undecidable, no such method exists.

So if you ever come across a proof that solves it, you better be concerned. Well, in my master's thesis I did just that. It turns out I didn't make a major mistake, but discovered incomputability is quite a subtle thing. The details of the particular problem aren't important. They can be explained by studying Turing Machines themselves.

A Turing Machine has a tape, a read/write head, and a finite state machine controlling the head:



My idea for solving the halting problem is to run the machine for some large number of steps, then stop and conclude anything that runs longer must not terminate. You might say there is no way to determine such a bound, but there actually is.

A Turing machine has only finitely many non-blank squares. For a fixed state machine Q and alphabet S define $A(n, Q, S)$ to be the set of Turing machines in Q and S which halt and have exactly n non-blank squares. There are at most $|S|^n$ of these so $A(n, Q, S)$ is a finite set, so if we let $\ell(t)$ be the number of steps t runs, $\{\ell(t) : t \in A(n, Q, S)\}$ is finite and thus has a maximum M. So given any turing machine with this criteria, we can run it for M steps and if it goes longer, than we know it will never terminate!

We can even define a function $f(S, Q, N)$ which is the appropriate M value for our inputs!

So why can't we do this? Well we would actually have to be able to compute f with a procedure. Even easier, just compute a function that bounds f. Since it is impossible we must conclude f is larger than any of the functions we can compute. Polynomials, exponentials, none are big enough. Maybe our program could include a

lookup table. But this would require infinite space and is not a method at all.

this shows us that uncomputability is primarily a problem of growing too quickly, not any kind of tricky correspondence.

this also suggests why the Ackermann function requires more sophisticated models of computation.

References

Minsky. Computation: finite and infinite machines. Chapter 8.