Let's begin by considering a class of Turing machines that have the following properties:

- They have a single halting state as opposed to accepting / rejecting states.
- The HALT state does not count as one of the n states.
- They have an infinite, two-directional tape that starts out completely filled with zeroes. In other words, the 0 is our new blank symbol.

We can denote these Turing machines by $M = (Q \cup \{\text{HALT}\}, \Sigma, \delta, S)$ where Q is a finite set of n states, Σ is our tape alphabet, δ is our transition function, and $S \in Q$ is the start state. We will almost always label our states A, B, C, $\ldots$ and assume A is the start state. A 2-state, 2-symbol Turing machine of this form might be $M = (\{A, B\}, \{0, 1\}, \delta, A)$ where δ is defined as:

δ	A	B
0	1-Right-B	1-Left-A
1	1-Left-B	1-HALT

Here's a trace of M's execution:

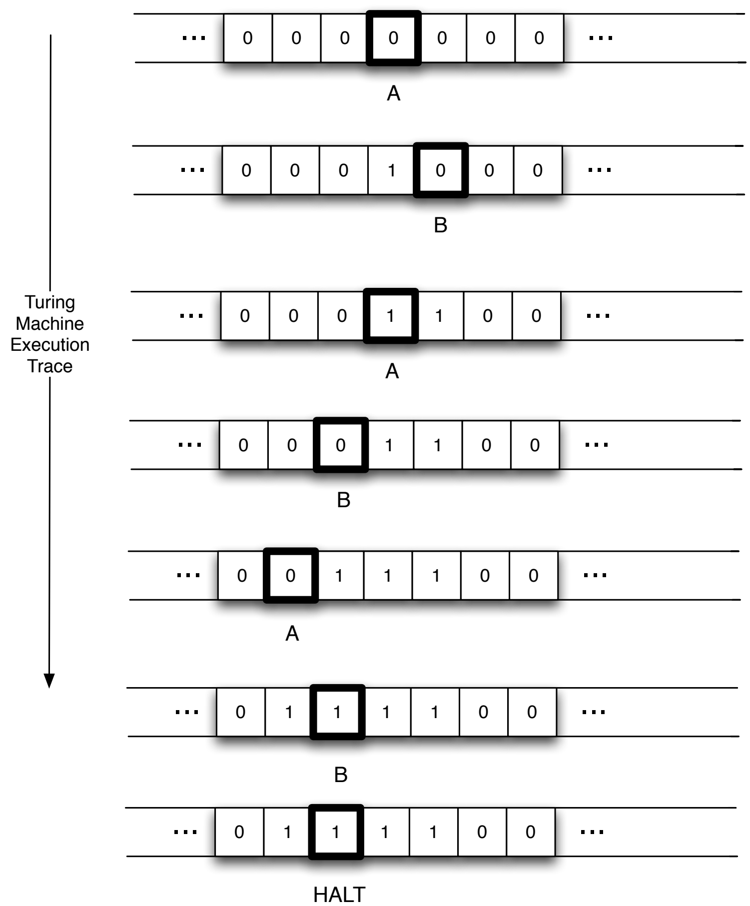

Figure 1: The execution trace of M

0.1 Busy Beavers

Imagine that we enumerated all 2-state, 2-symbol Turing machines. Some of these machines halt and some do not. However, at least one machine halts and writes at least as many 1s to the tape as any of the other 2-symbol, 2-state halting machines. We call this the 2-state, 2-symbol *busy-beaver* machine and take $\Sigma(2)$ to be the number of ones this machine halts with. Machine M from Figure 1 is the 2-state, 2-symbol busy-beaver machine. Thus $\Sigma(2) = 4$. In general, $\Sigma(n)$ is the maximum number of 1s a 2-symbol, n-state Turing machine can write to its tape and then halt.

Here is the transition function for the 2-symbol, 1-state busy beaver.

	A
0	1-Halt
1	Not reached

Thus $\Sigma(1) = 1$. The homework this week explores why $\Sigma(n)$ and its related function $S(n)$ – the maximum number of steps a 2-symbol, n-state Turing machine may take before halting – are undecidable functions. In particular, you might be able to see that if $S(n)$ were computable, then we could use it to decide any language.