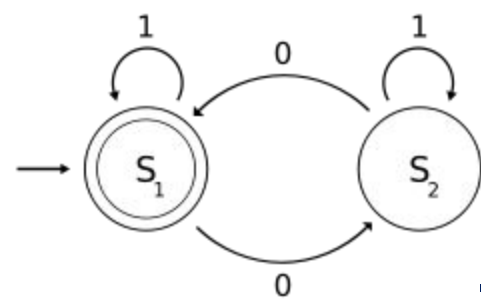
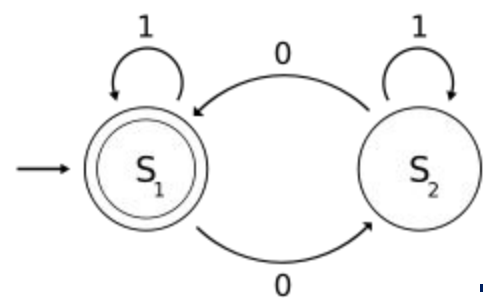




Building a Better Mousetrap

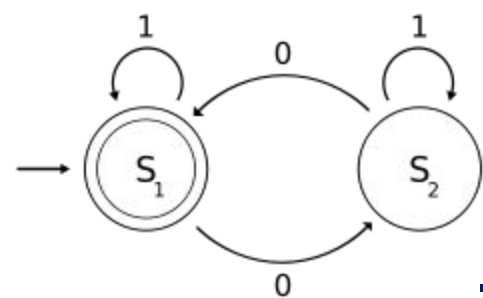


Today's adventures

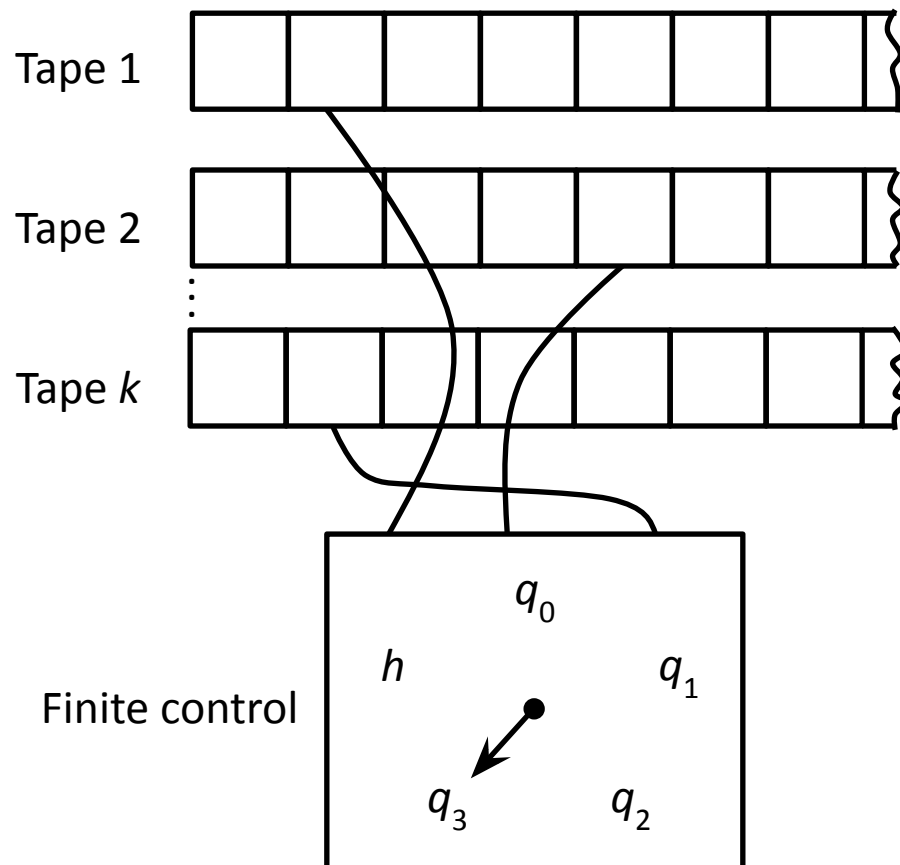


- Variants of TMs
 - Multitape TM
 - Nondeterministic TM
 - Enumerator
- Recursively enumerable languages
- Recognizable vs decidable
- Simplest impossible problem

Weekly tip: Curiosity is a muscle

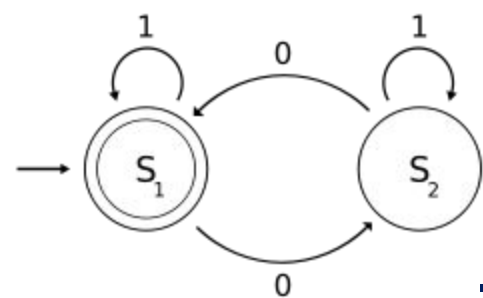


Multitape Turing Machines



Formally, we need only change the transition function to

$$\delta: Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R\}^k$$

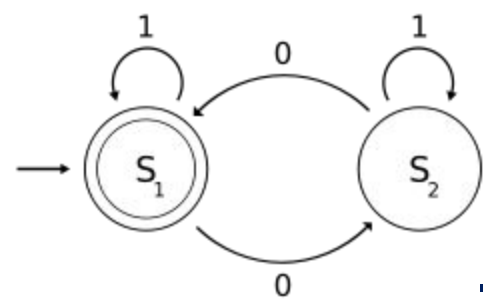


Exercise

Define a multitape Turing Machine M that can compute addition.

Namely, it will contain one integer input on one tape, one second integer input on another tape, and will finish computation with the sum of those two numbers on its third tape.

You may assume that the alphabet is $\{1\}^*$, so the integer 5 is represented as 11111.

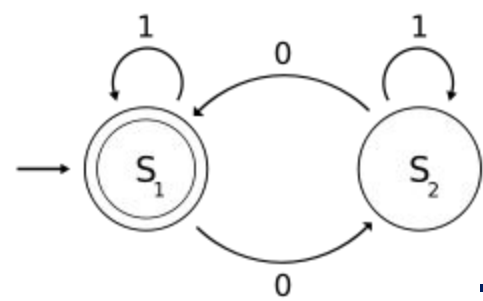


Evidence of Turing Robustness

Theorem. Every multitape Turing machine has an equivalent single tape Turing machine.

Proof Idea.

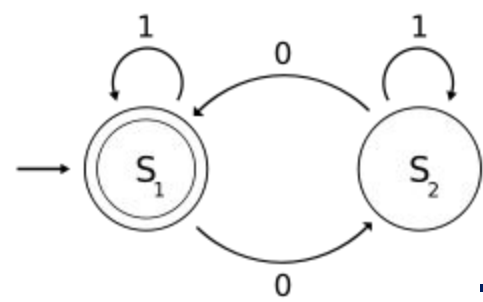
Corollary. A language is Turing-recognizable if and only if some multitape Turing machine recognizes it.



Exercise

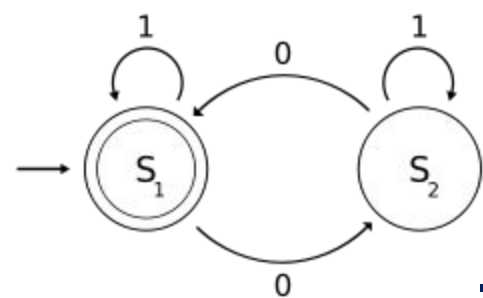
Design a Turing machine M that can solve the problem of whether a number is composite or prime.

Namely, suppose $L = \{ I^n : n \text{ is a composite number} \}$. Design a multitape Turing machine M that recognizes L .

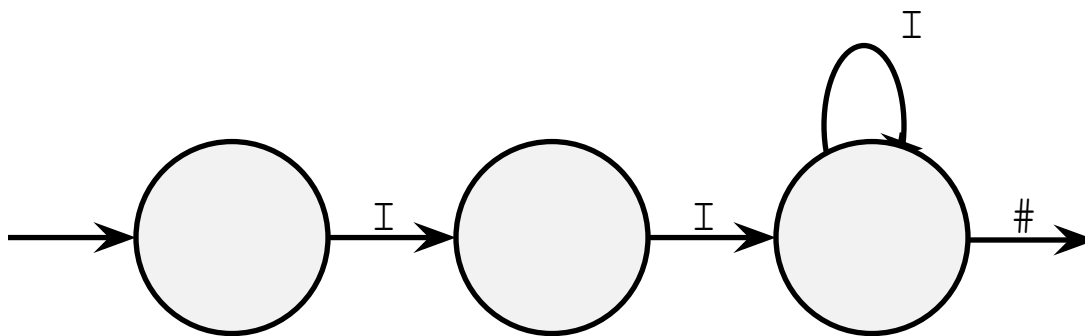


Nondeterministic Turing Machines

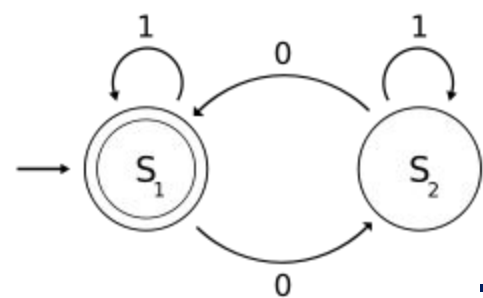
Design a machine M that on input I^n performs the following steps:



The Guessing Machine



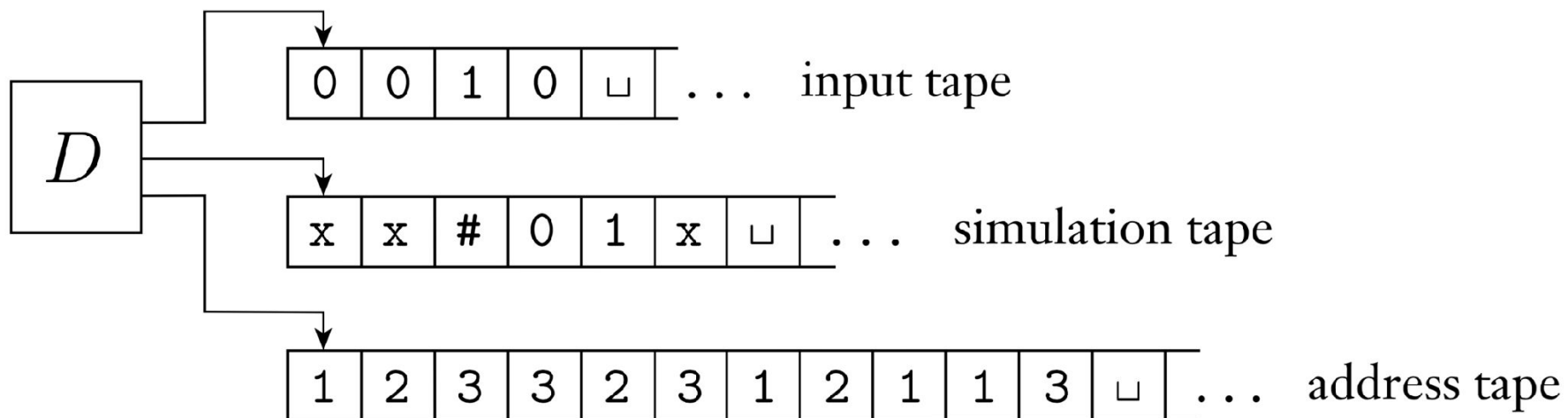
Again, the only difference between this variant and the standard TM is the transition function: $\delta: Q \times \Gamma \rightarrow P(Q \times \Gamma \times \{L, R\})$

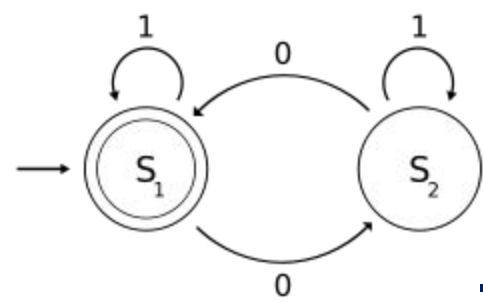


Guessing Doesn't Help

Theorem. Every nondeterministic Turing machine has an equivalent deterministic Turing machine.

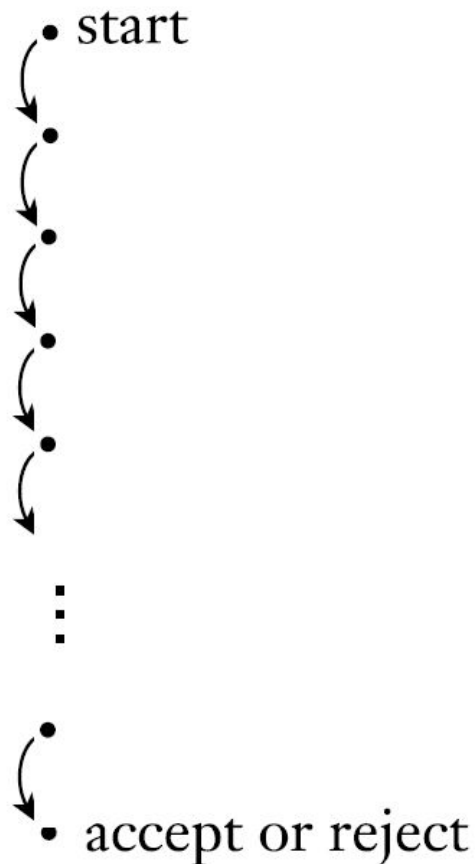
Proof Idea.



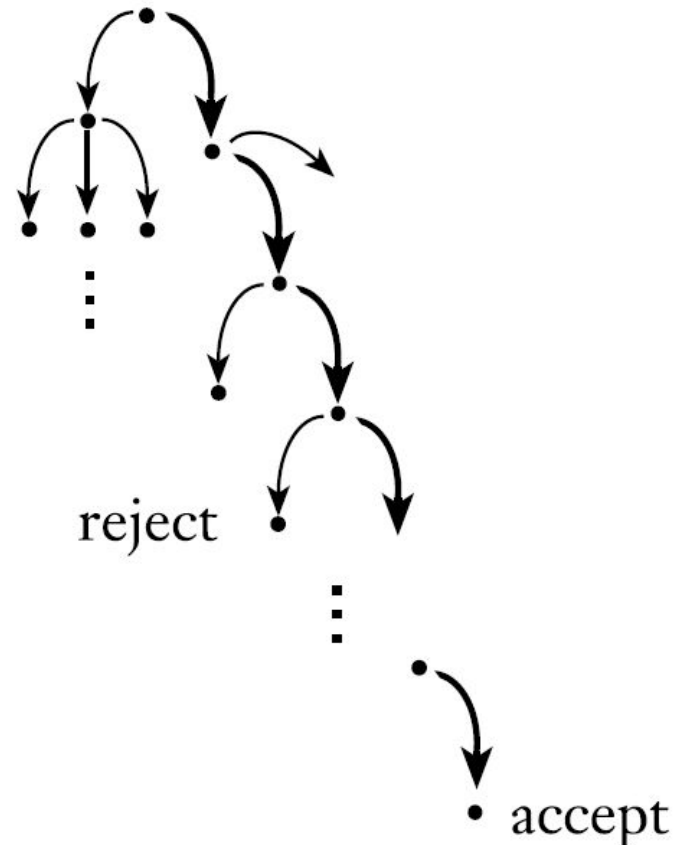


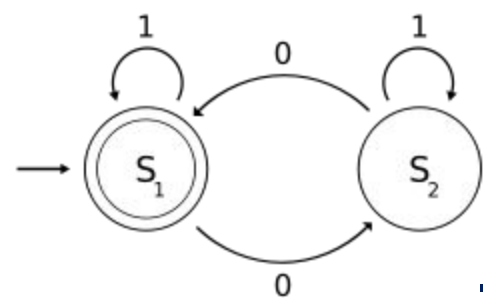
How Does That Compute?

Deterministic
computation

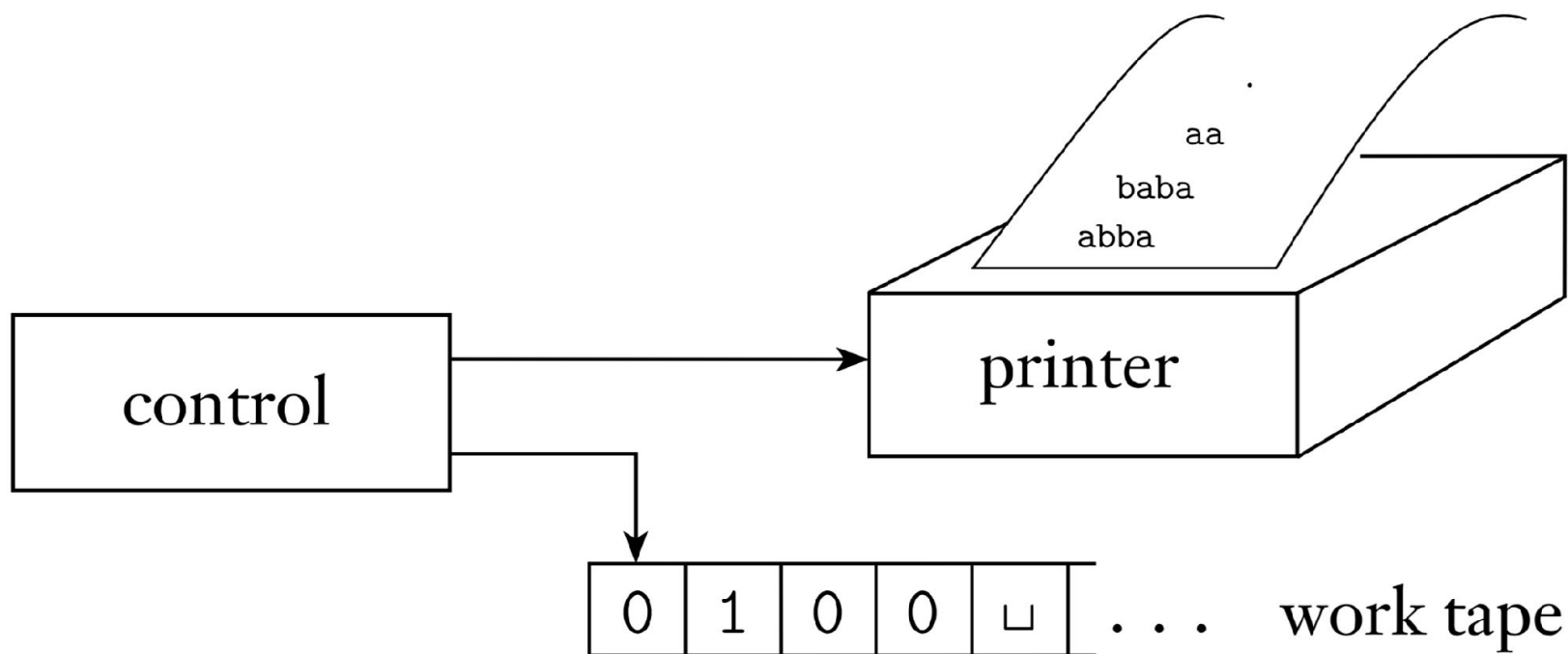


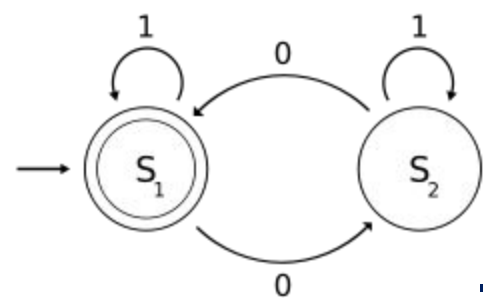
Nondeterministic
computation





Recursively Enumerable

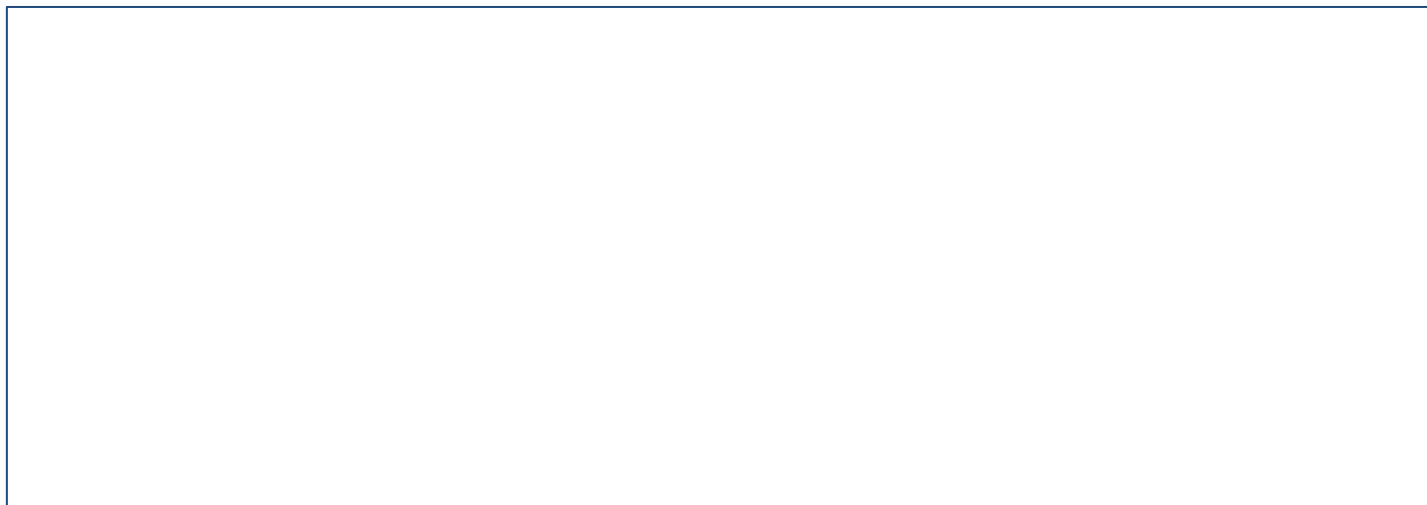


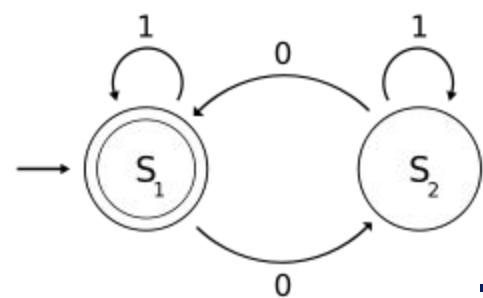


Enumerators

Theorem. A language is Turing-recognizable if and only if some enumerator enumerates it.

Proof. ($\Leftarrow$) Suppose enumerator E enumerates L . We will define a Turing machine M that recognizes L .

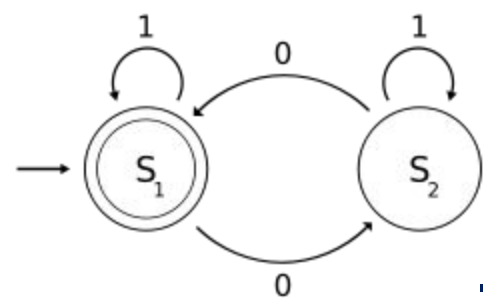




Recursively Enumerable

Theorem. A language is Turing-recognizable if and only if some enumerator enumerates it.

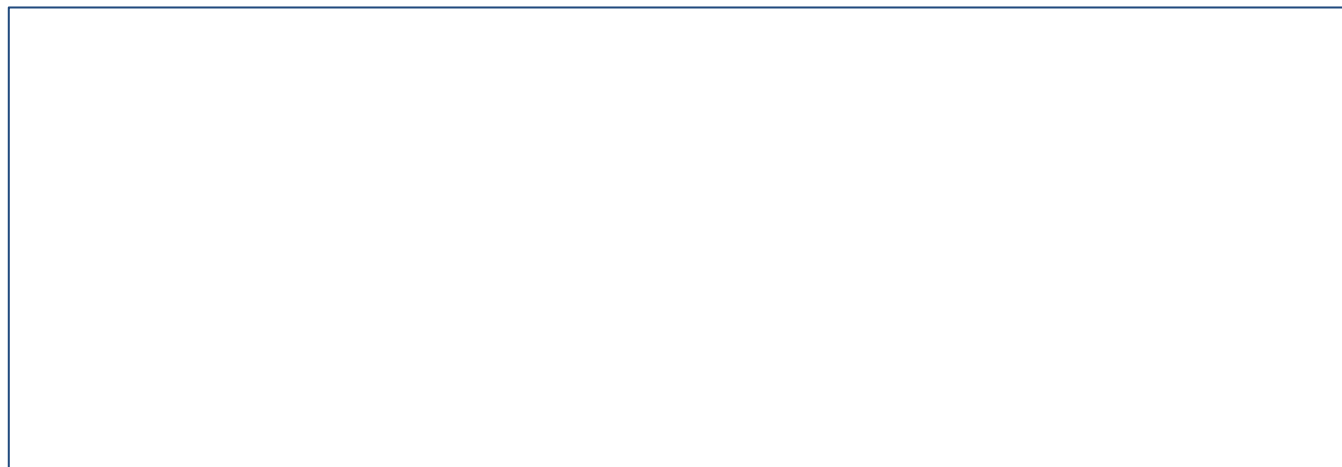
Proof. ($\Rightarrow$) Suppose TM M recognizes L . We will build an enumerator that enumerates L .
To start, build a lexicographic enumerator to generate the list of all possible strings $s_1, s_2, \dots$ over Σ .

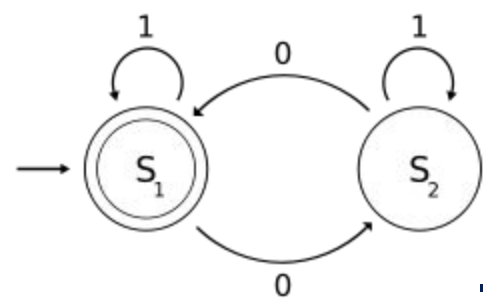


Recursively Enumerable

Theorem. A language is Turing-recognizable if and only if some enumerator enumerates it.

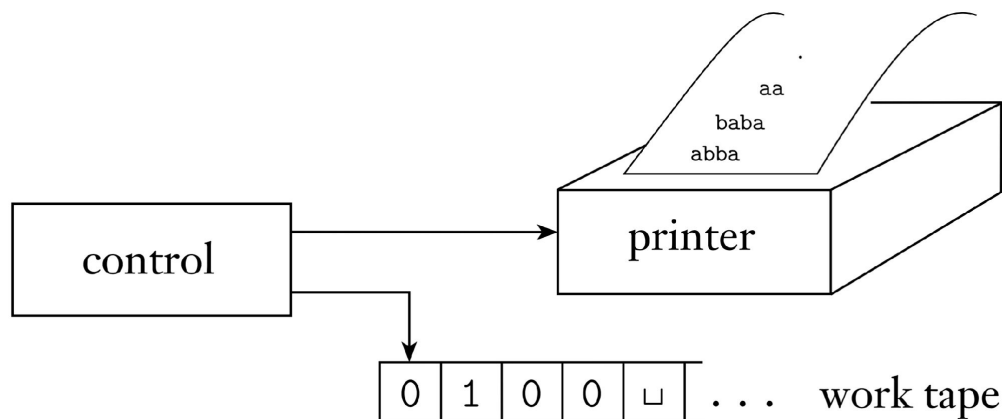
Proof. ($\Rightarrow$) Suppose TM M recognizes L . We will build an enumerator that enumerates L .
To start, build a lexicographic enumerator to generate the list of all possible strings $s_1, s_2, \dots$ over Σ .



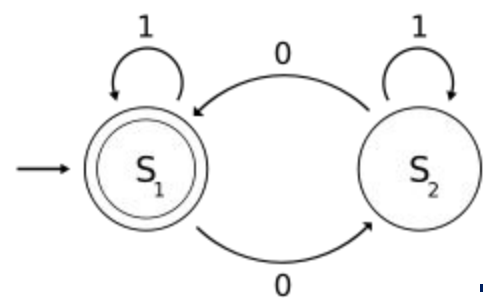


TM's Take Their Own Sweet Time

- Recognizers, like enumerators, may take a while to answer yes, ... and even longer to answer *no*.



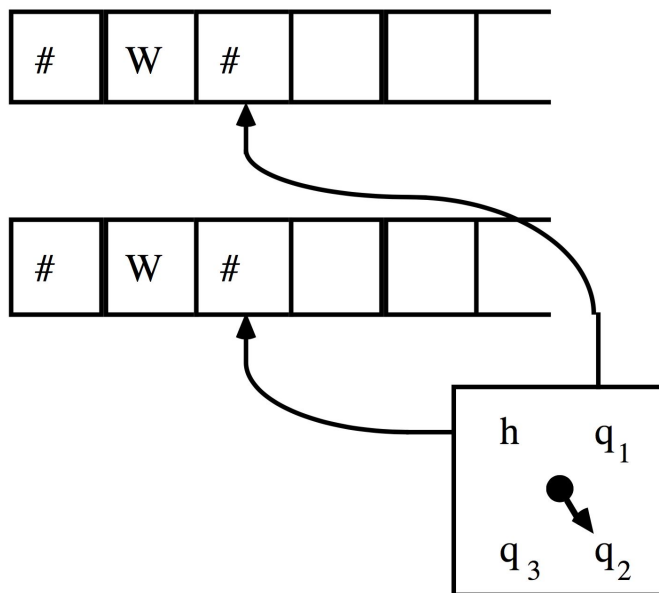
- A TM that halts on all inputs is called a decider. A ***decider*** that recognizes a language is said to ***decide*** that language.
- Call a language ***Turing-decidable*** if some Turing machine decides it.



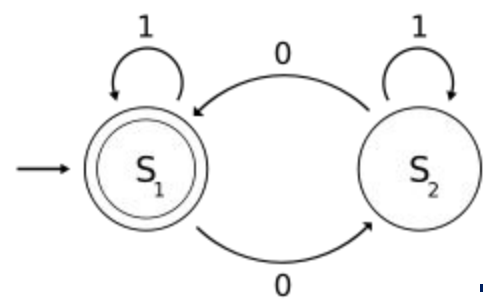
Recognizable versus Decidable

Theorem. A language is *Turing-decidable* if and only if both it and its complement are *Turing-recognizable*.

Proof. ($\Rightarrow$) By definition.



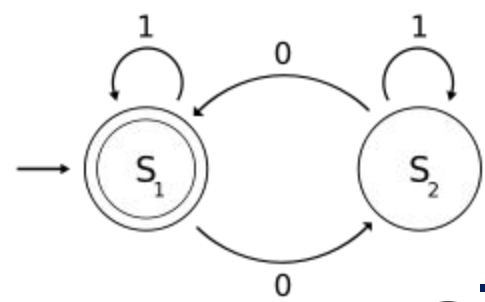
($\Leftarrow$) Simulate, in parallel, M_L on tape 1 and M_L on tape 2.



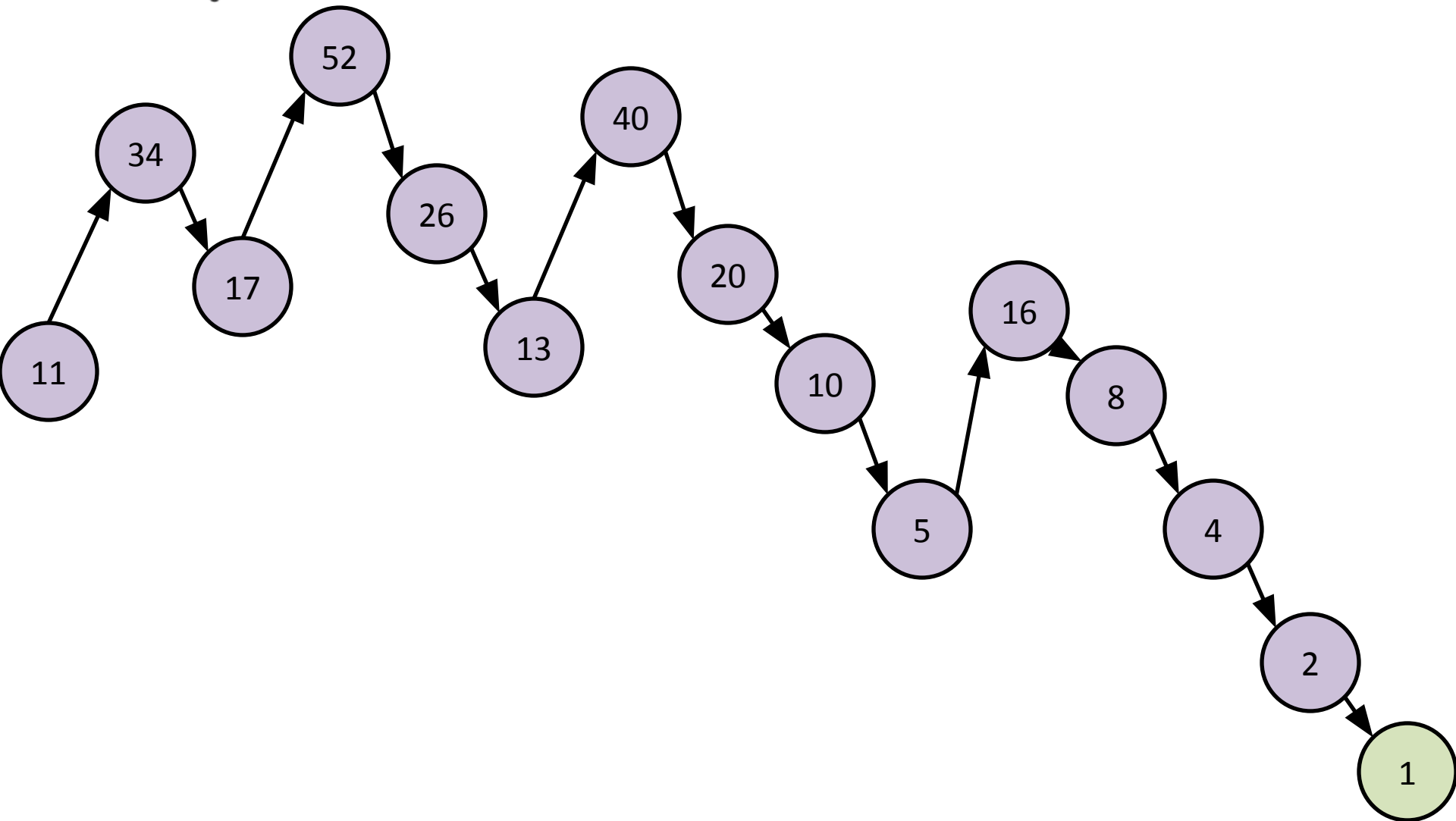
The Hailstone Sequence

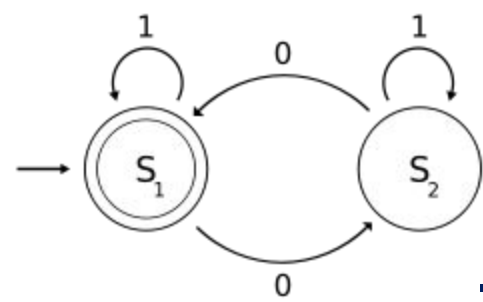
```
HailstoneSequence(n)
  if (n  $\neq$  1)
    if (n is even)
      HailstoneSequence(n/2)
    else
      HailstoneSequence(3*n+1)
```

Given an integer $n > 0$, does this process terminate?



Example Sequence, $n = 11$

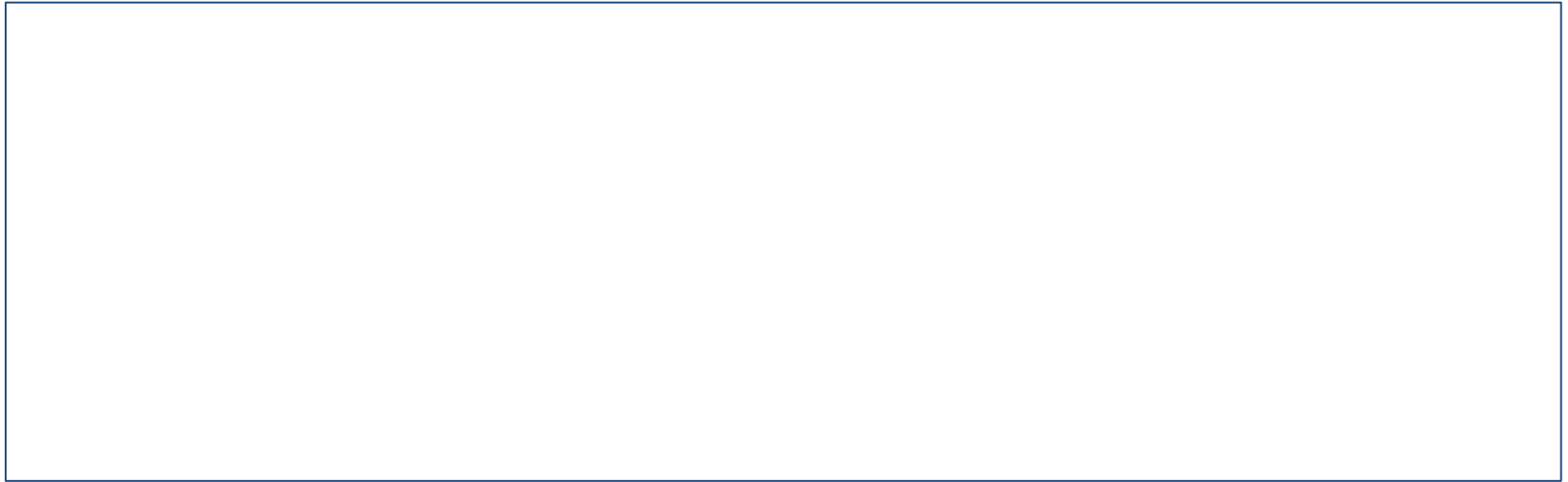


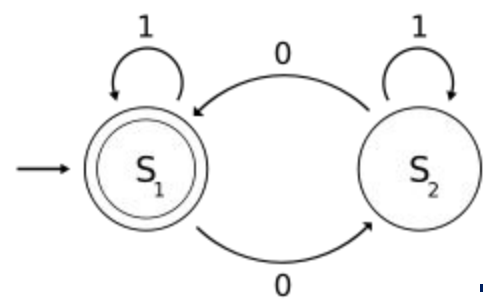


Hailstone Turing Machine

Let $H = \{ I^n \mid n > 0 \text{ and the hailstone sequence terminates for } n \}$.

We construct TM M to recognize language H .





The Simplest Impossible Problem

- Is it unknown whether this process will terminate for all natural numbers.
- It is unknown whether TM M might loop forever.