Building a Better Mousetrap
Formally, we need only change the transition function to

$$\delta: Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R\}^k$$
Evidence of Turing Robustness

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

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

- Let \( L = \{ I^n : n \text{ is a composite number} \} \).
- Designing a Turing machine to accept \( L \) would seem to involve factoring \( n \).
- However, if we could guess ...
Guessing Games

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

1. Nondeterministically choose two numbers $p, q > 1$ and transform the input into $#I^n#I^p#I^q#$.

2. Multiplies $p$ by $q$ to obtain $#I^n#I^{pq}$.

3. Checks the number of $I$'s before and after the middle $#$ for equality. Accepts if equal, and rejects otherwise.
Again, the only difference between this variant and the standard TM is the transition function: \( \delta: Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\}) \)
Guessing Doesn’t Help

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

![Diagram](image-url)
How Does That Compute?

Deterministic computation:
- start
- ...
- accept or reject

Nondeterministic computation:
- reject
- ...
- accept
Recursively Enumerable
Theorem. A language is Turing-recognizable if and only if some enumerator enumerates it.

Proof. ($\iff$) Suppose enumerator $E$ enumerates $L$.

Define $M = \text{"On input } w:\text{ run } E. \text{ Every time } E \text{ outputs a string, compare it with } w.$

If $w$ ever appears in the output of $E$, accept.
Theorem. A language is Turing-recognizable if and only if some enumerator enumerates it.

Proof. (⇒) Suppose TM \( M \) recognizes \( L \). Build a lexicographic enumerator to generate the list of all possible strings \( s_1, s_2, \ldots \) over \( \Sigma \).
Theorem. A language is Turing-recognizable if and only if some enumerator enumerates it.

Proof. (⇒) Suppose TM $M$ recognizes $L$. Build a lexicographic enumerator to generate the list of all possible strings $s_1, s_2, \ldots$ over $\Sigma$.

Define $E =$ “Ignore input.

Repeat the following for $i = 1, 2, 3, \ldots$

Run $M$ for $i$ steps on each of $s_1, s_2, \ldots, s_i$.

If any computation accepts, print corresponding $s_j$.”
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.
Theorem. A language is Turing-decidable if and only if both it and its complement are Turing-recognizable.

Proof. (⇒) By definition.

(⟸) Simulate, in parallel, $M_L$ on tape 1 and $M_{L'}$ on tape 2.
The Hailstone Sequence

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

HailstoneSequence(n)
  if ($n \neq 1$)
    if ($n$ is even)
      HailstoneSequence($n/2$)
    else
      HailstoneSequence($3*n+1$)
Example Sequence, $n = 11$
Let $H = \{ I^n \mid n > 0 \text{ and the hailstone sequence terminates for } n \}$.

We construct TM $M$ to recognize language $H$.

$M = \text{“On input } w:\n1. \text{ If the input is } \varepsilon, \text{ reject.} \n2. \text{ If the input has length 1, accept.} \n3. \text{ If the input has even length, halve its length.} \n4. \text{ If the input has odd length, triple its length and append } I. \n5. \text{ Go to stage 2.} \text{“} \n$
The Simplest Impossible Problem

- Is is unknown whether this process will terminate for all natural numbers.

- It is unknown whether \( TM\ M \) might loop forever.