Building a Better Mousetrap

Multitape Turing Machines

Formally, we need only change the transition function to
\[ \delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R\}^k \]

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 $#I^n#I^n#I^n#$.
2. Multiplies $p$ by $q$ to obtain $#I^n#I^n#I^n#$.
3. Checks the number of $I$'s before and after the middle $#$ for equality. Accepts if equal, and rejects otherwise.

Guessing Doesn't Help

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

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

Proof. (⇐) Suppose enumerator $E$ enumerates $L$.
Define $M = "\text{On input } w:\"$

Run $E$. Every time $E$ outputs a string, compare it with $w$.
If $w$ ever appears in the output of $E$, accept."

Recursively Enumerable

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 = "\text{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 $s_j$."

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. (⇒) By definition.

($\iff$) Simulate, in parallel, $M_L$ on tape 1 and $M_{\overline{L}}$ on tape 2.
**The Hailstone Sequence**

\[
\text{HailstoneSequence}(n)
\]
\[
\text{if } (n \neq 1)
\]
\[
\text{if } (n \text{ is even}) \
\text{HailstoneSequence}(n/2)
\]
\[
\text{else} \
\text{HailstoneSequence}(3n+1)
\]

Given an integer \( n > 0 \), does this process terminate?

---

**Example Sequence, \( n = 11 \)**

\[11 \rightarrow 34 \rightarrow 17 \rightarrow 52 \rightarrow 26 \rightarrow 13 \rightarrow 40 \rightarrow 20 \rightarrow 10 \rightarrow 5 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1\]

---

**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 \( H \).

\( M = \) "On input \( \omega \):
1. If the input is \( \varepsilon \), reject.
2. If the input has length 1, accept.
3. If the input has even length, halve its length.
4. If the input has odd length, triple its length and append \( I \).
5. Go to stage 2."

---

**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.