Decidable or Undecidable?

- **Poll 1:** Let $\Sigma$ be a finite alphabet. Which of the following sets is countable?
  1. The set of decidable languages over $\Sigma$
  2. The set of all languages over $\Sigma$
  3. Both
  4. Neither
For all $n > 2$ there are no natural $a,b,c$ such that $a^n + b^n = c^n$. I have a truly marvelous demonstration of this proposition which this margin is too narrow to contain.
FERMIAT()
\[
 t ← 3 \\
 \text{while true} \\
 \quad \text{for all } n \in \{3, \ldots, t\} \text{ and } x, y, z \in \{1, \ldots, t\} \\
 \quad \quad \text{if } x^n + y^n = z^n \text{ then return } (x, y, z, n) \\
 \quad \text{end for} \\
 \quad t ← t + 1 \\
 \text{end while}
\]

**Question:** Does this program halt?

**Theorem:**
The Halting Problem is undecidable!

**Proof (by Pseudocode):**

- Suppose that there exists a procedure `HALT(program, input)`
- Consider the program:
  
  ```
  Turing(program) 
  \quad \text{if } HALT(program, program) \text{ then loop forever} 
  \quad \text{else return true}
  ```

- What is the output of `HALT(Turing, Turing)`?
  - If `HALT(Turing, Turing)` then `Turing(Turing)` doesn’t halt
  - If not `HALT(Turing, Turing)` then `Turing(Turing)` halts
**Proof (More Formal)**

- HALT = \{ (M, x) : M is a TM that halts on x \}
- Suppose the TM \( M_{\text{HALT}} \) decides HALT
- Consider the following TM \( M_{\text{TURING}} \)

Treat the input as \( \langle M \rangle \) for a TM \( M \)
Run \( M_{\text{HALT}} \) with input \( \langle M, M \rangle \)
If it accepts, go into an infinite loop
If it rejects, accept (i.e., halt)

**Proof (More Formal)**

- HALT = \{ (M, x) : M is a TM that halts on x \}
- Suppose the TM \( M_{\text{HALT}} \) decides HALT
- Consider the following TM \( M_{\text{TURING}} \)

**Proof (More Formal)**

What happens when \( \langle M_{\text{TURING}} \rangle \) is given as input to \( M_{\text{TURING}} \)?
**DIAGONALIZATION REDUX**

\[
\begin{array}{cccc}
(M_1) & (M_2) & (M_3) & (M_4) \\
\vdots & \vdots & \vdots & \vdots \\
M_1 & & \color{red} \times & \color{green} \times \\
M_2 & \color{red} \times & & \color{green} \times \\
M_3 & \color{red} \times & \color{green} \times & \\
M_4 & \color{red} \times & \color{green} \times & \color{red} \times \\
M_5 & \color{red} \times & \color{green} \times & \color{red} \times \\
\vdots & \vdots & \vdots & \vdots \\
\end{array}
\]

This is nothing but a diagonalization argument!

---

**Is there a way to show other languages are undecidable?**

---

**REDUCTIONS**

- We want to define \( A \leq B \) to mean that \( B \) is at least as hard as \( A \)

- That is:
  - \( B \) decidable \( \Rightarrow \) \( A \) decidable
  - \( A \) undecidable \( \Rightarrow \) \( B \) undecidable
REDUCTIONS

- **Terminology**: Let $A$ and $B$ be two languages, we say that $A$ reduces to $B$, and write $A \leq B$, if it is possible to decide $A$ using a TM that decides $B$ as a subroutine.

To show that problem $B$ is undecidable, we just need to show that $\text{HALT} \leq B$.

EXAMPLE: ACCEPTS

- $\text{ACCEPTS} = \{(M,x) : M\text{ is a TM that accepts } x\}$
- This means:
  - $(M,x) \in \text{ACCEPTS} \Rightarrow x\text{ leads to an accept state in } M$
  - $(M,x) \notin \text{ACCEPTS} \Rightarrow x\text{ leads to a reject state or } M\text{ does not halt}$
- **Theorem**: $\text{ACCEPTS}$ is undecidable
**Proof (by Illustration)**

- We will show that $\text{HALT} \leq \text{ACCEPTS}$
- Let $M_{\text{ACCEPTS}}$ be a TM that decides $\text{ACCEPTS}$
- Here is a TM that decides $\text{HALT}$:
  - On input $(M, x)$ run $M_{\text{ACCEPTS}}((M, x))$
  - If it accepts, accept
  - Reverse the accept and reverse states of $M$, call it $M'$
  - Run $M_{\text{ACCEPTS}}((M', x))$
  - If it accepts, accept, and reject otherwise
- Argue that:
  - If $(M, x) \in \text{HALT}$ then the machine accepts it
  - If $(M, x) \notin \text{HALT}$ then the machine rejects it $\square$

**Example: Empty**

- $\text{EMPTY} = \{M \mid M \text{ is a TM that accepts nothing}\}$
- Theorem: $\text{EMPTY}$ is undecidable
**PROOF**

- We will show that ACCEPTS ≤ EMPTY
- Given \((M, x)\), construct a TM \(M_x\) that, given \(y\), runs \(M(x)\) and returns its output
- The machine \(M_{ACCEPTS}\) constructs \(M_x\), runs \(M_{EMPTY}(M_x)\), and flips its output
- Two cases:
  - \(M\) accepts \(x\) \(\Rightarrow L(M_x) = \Sigma^* \Rightarrow M_{EMPTY}\) rejects \(M_x\)
  - \(M\) rejects \(x\) or doesn’t halt on \(x\) \(\Rightarrow L(M_x) = \emptyset\)
  \(\Rightarrow M_{EMPTY}\) accepts \((M_x)\) ■

---

**Post’s Correspondence Problem**

**Input**
A finite collection of “dominoes” with strings written on each half

**Output**
Accept if copies of the dominoes can be arranged so that the strings match

Undecidable! Proved in 1946 by Post
WANG TILES

**Input**
A finite collection of “Wang tiles” (squares) with colored edges

**Output**
Accept if the infinite plane can be tiled using tiles with matching sides

Undecidable! Proved in 1966 by Berger

BIG UNDECIDABLE PROBLEMS

- Entscheidungsproblem:
  - Can a first-order-logic formula be derived from given axioms?
  - Example: \( \neg\exists x, y, z, n \in \mathbb{N}: (n \geq 3) \land (x^n + y^n = z^n) \)
  - Formulated by Hilbert in 1928, proved undecidable by Turing in 1936 (and, independently, by Church)
- Hilbert’s 10th Problem (Diophantine equations):
  - Does a given multivariate polynomial with integer coefficients have an integer root?
  - Example: \( 3x^2 - 2xy - y^2z - 7 = 0 \) \( (x = 1, y = 2, z = -2) \)
  - One of 23 open problems on Hilbert’s famous 1900 list
  - Proved undecidable by Matiyasevich in 1970

DECIDABLE OR UNDECIDABLE?

- Poll 2: Which of the following problems is decidable?
  1. \( EQ = \{ (M, M') : M, M' \text{ TMs, } L(M) = L(M') \} \)
  2. \( GRAVITON = \emptyset \) if gravitons exist, \( \{1\} \) otherwise
  3. Both
  4. Neither
**INTERESTING VS. DECIDABLE**

All problems

So what next?

**SUMMARY**

- Terminology and concepts:
  - HALT, ACCEPTS, EMPTY
  - Reductions between computational problems
- Theorems:
  - Most problems are undecidable
  - HALT, ACCEPTS, EMPTY are undecidable
- Big ideas:
  - Exploring the limits of computation via reductions