Great Theoretical Ideas in CS

Lecture 8:
Turing’s Legacy: Undecidability

Anil Ada
Ariel Procaccia (this time)
Our Protagonists

Georg Cantor
1845-1918
Father of set theory

Alan Turing
1912-1954
Father of CS
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
Maybe undecidable problems are not interesting?
The Halting Problem

- Input: Program pseudocode, input to the program
- Output: True if the given program halts on the given input, false otherwise

Why is it interesting?
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.
FERMAT()

\[
t ← 3
\]

\[
\text{while true}
\]

\[
\text{for all } n ∈ \{3, ..., t\} \text{ and } x, y, z ∈ \{1, ..., t\}
\]

\[
\text{if } x^n + y^n = z^n \text{ then return } (x, y, z, n)
\]

\[
\text{end for}
\]

\[
t ← t + 1
\]

\[
\text{end while}
\]

Question: Does this program halt?
Theorem:
The Halting Problem is undecidable!
Proof (by Pseudocode)

- Suppose that there exists a procedure \( \text{HALT}(\text{program, input}) \)
- Consider the program:

\[
\text{Turing}(\text{program}) \\
\text{if } \text{HALT}(\text{program, program}) \text{ then loop forever} \\
\text{else} \\
\text{return true}
\]

- What is the output of \( \text{Halt(Turing, Turing)} \)?
  - If \( \text{Halt(Turing, Turing)} \) then \( \text{Turing(Turing)} \) doesn’t halt
  - If not \( \text{Halt(Turing, Turing)} \) then \( \text{Turing(Turing)} \) halts
Proof (More Formal)

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

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

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

What happens when \( \langle M_{TURING} \rangle \) is given as input to \( M_{TURING} \)?
### Diagonalization Redux

<table>
<thead>
<tr>
<th></th>
<th>$M_1$</th>
<th>$M_2$</th>
<th>$M_3$</th>
<th>$M_4$</th>
<th>$M_5$</th>
<th>\ldots</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>\cellcolor{red!40}</td>
<td>\cellcolor{red!40}</td>
<td>\cellcolor{green!40}</td>
<td>\cellcolor{red!40}</td>
<td>\cellcolor{green!40}</td>
<td>\ldots</td>
</tr>
<tr>
<td>$M_2$</td>
<td>\cellcolor{green!40}</td>
<td>\cellcolor{red!40}</td>
<td>\cellcolor{red!40}</td>
<td>\cellcolor{green!40}</td>
<td>\cellcolor{red!40}</td>
<td>\ldots</td>
</tr>
<tr>
<td>$M_3$</td>
<td>\cellcolor{red!40}</td>
<td>\cellcolor{green!40}</td>
<td>\cellcolor{green!40}</td>
<td>\cellcolor{green!40}</td>
<td>\cellcolor{red!40}</td>
<td>\ldots</td>
</tr>
<tr>
<td>$M_4$</td>
<td>\cellcolor{red!40}</td>
<td>\cellcolor{green!40}</td>
<td>\cellcolor{green!40}</td>
<td>\cellcolor{green!40}</td>
<td>\cellcolor{green!40}</td>
<td>\ldots</td>
</tr>
<tr>
<td>$M_5$</td>
<td>\cellcolor{red!40}</td>
<td>\cellcolor{red!40}</td>
<td>\cellcolor{red!40}</td>
<td>\cellcolor{red!40}</td>
<td>\cellcolor{green!40}</td>
<td>\ldots</td>
</tr>
<tr>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
</tr>
<tr>
<td>$M_{TURING}$</td>
<td>\cellcolor{red!40}</td>
<td>\cellcolor{green!40}</td>
<td>\cellcolor{red!40}</td>
<td>\cellcolor{red!40}</td>
<td>\cellcolor{green!40}</td>
<td>\ldots</td>
</tr>
</tbody>
</table>

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:
  o $B$ decidable $\Rightarrow A$ decidable
  o $A$ 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**

- \( \text{ACCEPTS} = \{ \langle M, x \rangle : M \text{ is a TM that accepts } x \} \)

- This means:
  - \( \langle M, x \rangle \in \text{ACCEPTS} \implies x \text{ leads to an accept state in } M \)
  - \( \langle M, x \rangle \notin \text{ACCEPTS} \implies x \text{ leads to a reject state or } M \text{ does not halt} \)

- **Theorem:** \( \text{ACCEPTS} \) is undecidable
Proof (by Illustration)

\[
\langle M, x \rangle \rightarrow M_{\text{HALT}} \rightarrow M_{\text{ACCEPTS}} \rightarrow \langle M' \rangle \rightarrow \langle M', x \rangle \rightarrow M_{\text{ACCEPTS}}
\]

Reverse accept and reject states
**Proof (More Formal)**

- 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 $\langle M, x \rangle$ run $M_{\text{ACCEPTS}}(\langle M, x \rangle)$
  - If it accepts, accept
  - Reverse the accept and reverse states of $M$, call it $M'$
  - Run $M_{\text{ACCEPTS}}(\langle M', x \rangle)$
  - If it accepts, accept, and reject otherwise

- Argue that:
  - If $\langle M, x \rangle \in \text{HALT}$ then the machine accepts it
  - If $\langle M, x \rangle \notin \text{HALT}$ then the machine rejects it
**Example: Empty**

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

![Fuel Gauge Image](image-url)
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 ⇒ L(M_x) = Σ* ⇒ M_{EMPTY} rejects ⟨M_x⟩
  - M rejects x or doesn’t halt on x ⇒ L(M_x) = ∅ ⇒ M_{EMPTY} accepts ⟨M_x⟩
These problems involve TMs, are there undecidable problems that don’t?
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:
  o Pronunciation: [link to pronunciation video]
  o Can a first-order-logic formula be derived from given axioms?
  o Example: \( \neg \exists x, y, z, n \in \mathbb{N}: (n \geq 3) \land (x^n + y^n = z^n) \)
  o Formulated by Hilbert in 1928, proved undecidable by Turing in 1936 (and, independently, by Church)

• Hilbert’s 10th Problem (Diophantine equations):
  o Does a given multivariate polynomial with integer coefficients have an integer root?
  o Example: \( 3x^2 - 2xy - y^2z - 7 = 0 \) (\( x = 1, y = 2, z = -2 \))
  o One of 23 open problems on Hilbert’s famous 1900 list
  o Proved undecidable by Matiyasevich in 1970
Decidable or Undecidable?

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

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