Computability and Randomness (I)

Yijia Chen
Fudan University
About me

Professor in Computer Science.

Ph.D. in Computer Science (Shanghai Jiaotong), Ph.D. in Mathematics (Freiburg).

Research interests: logic in computer science, computational complexity, and graph theory.
Then why am I teaching here?

Why philosophers should care about computer science?

Immediate answers:

1. I will write a paper in philosophy using Microsoft Word.
2. I have to send an email to my colleague.
3. I need to conduct a philosophical experiment using computer.
Computer Science is no more about computers than astronomy is about telescopes.
What will be covered?

1. I will discuss some philosophical questions from a computer science perspective, e.g., computability and complexity.

2. I will sneak in some of my favourite results and questions from computer science with certain philosophical contents.

3. I will present you some of the proofs, at least one proof for each lecture. I won’t give all the details but try to convey the ideas.
Hilbert’s Dream
Hilbert’s Dream

*Entscheidungsproblem*: Design a *mechanical procedure* that determines the truth or falsehood of any mathematical statement.

Can Artificial Intelligence replace Human Intelligence?
The negative answer of Gödel, Church, and Turing

Now matter how powerful a computer $C$ is,

- either $C$ makes some mistake,

- or there is a true mathematical statement $\varphi_C$ such that $C$ doesn’t know that $\varphi_C$ is correct.

Remark

A more powerful $C'$ might know the correctness of $\varphi_C$. But again, there will be a further $\varphi_{C'}$ whose truth is unknown to $C'$. 
Liar's Paradox: I'm lying.

A rough mathematical version: $C$ can’t prove that I’m correct.

The technical challenge: how to formulate “I?”
Turing machines are theoretical model of almost all modern computers (of the Von Neumann architecture).

Prior to Turing machines, every computer was designed to tackle one specific problem, and there was no concept of programming language.

Turing exhibited a universal Turing machine $U$ which encompasses all possible Turing machines, based on which he gave a machine version of the Liar’s Paradox:

I (as a machine) won’t halt.
Universal Turing Machine

A universal Turing machine $U$ has two inputs

1. a Turing machine $M$, or more precisely its encoding
2. and an input $x$ of $M$.

$U$ simulates $M$ on $x$, and

1. either never halts, provides $M$ never halts on $x$,
2. gives the same output of $M$, i.e., $U(M, x) = M(x)$.

In more modern term, $U$ is an interpreter of Turing machines.
The halting problem

<table>
<thead>
<tr>
<th><strong>Halt</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> $M$ without any input.</td>
</tr>
<tr>
<td><strong>Problem:</strong> Does $M$ halt?</td>
</tr>
</tbody>
</table>

1. Does this App never freeze?
2. Does a communication protocol always end?
3. Does a discussion lead to a conclusion?
No Turing machine for $\text{HALT}$

Assume that there is a Turing machine $Halt$ for $\text{HALT}$. We consider the following $\text{Diag}(x)$:

1. view $x$ as a Turing machine with one input
2. $M \leftarrow U(x, x)$
3. if $Halt(M) = \text{yes}$ then never halt
4. else halt

Does $\text{Diag}$ halt on input $\text{Diag}$?
- If $\text{Diag}(\text{Diag})$ halts, then it never halts.
- If $\text{Diag}(\text{Diag})$ never halts, then it halts.

This implies that $Halt$ doesn’t exist.
This is a variant of Cantor’s diagonal argument!

Anyone remember?
Crushing Hilbert’s Dream

For every Turing machine $M$ without any input,

"$M$ halts"

is a mathematical statement, either true or false.

If there is a Turing machine that decides mathematical truth, then there is a Turing machine that decides $\text{HALT}$. 
A leap of argument?

Is any mechanical procedure equal to a Turing machine?

1. Computable functions.

2. Lambda Calculus.

3. Post systems.

4. . . .

They are all provably equivalent to Turing machines.

K. Gödel. “[T]he correct definition of mechanical computability was established beyond any doubt by Turing.”
An interesting consequence

Consider the functions

\[ step(M) = \begin{cases} 
  n & \text{if } M \text{ halts in exactly } n \text{ steps}, \\
  * & \text{otherwise.} 
\end{cases} \]

\[ length(\varphi) = \begin{cases} 
  n & \text{A shortest proof of } \varphi \text{ has length } n, \\
  * & \text{otherwise.} 
\end{cases} \]

There is no Turing machine computing \( step \) or \( length \).
An interesting consequence (cont’d)

Let $f$ be a function such that

$$f(\varphi) \geq \text{length}(\varphi)$$

for every $\varphi$. Then there is no Turing machine computing $\varphi$.

In particular

$$\text{length}(\varphi) \not\leq 2^{2^{2^{2^{2|x|}}}}$$

Therefore there are true mathematical statements with enormously long proofs.
Do we care about proofs that can’t even be fit in the whole universe?

Probably not.
A modified version of Hilbert's Dream

\textbf{Bounded-Proof}

\begin{itemize}
\item \textit{Input:} A mathematical statement $\varphi$ and a natural number $n$.
\item \textit{Problem:} Does $\varphi$ have a proof of length at most $n$?
\end{itemize}

Of course, there is a Turing machine $M$ solving \textbf{Bounded-Proof}. Why?

A \textit{brute-force} $M$ needs roughly $2^n$ steps, which won't halt on a reasonable $n = 1000$ until our universe collapses to a black hole.
In a letter from Gödel to Von Neumann dated in 1956:

*If there actually were a machine with [running time] \( \sim Kn \) (or even only with \( \sim Kn^2 \)) [for some constant \( K \) independent of \( n \)], this would have consequences of the greatest magnitude. That is to say, it would clearly indicate that, despite the unsolvability of the Entscheidungsproblem, the mental effort of the mathematician in the case of yes-or-no questions could be completely [added in a footnote: apart from the postulation of axioms] replaced by machines. One would indeed have to simply select an \( n \) so large that, if the machine yields no result, there would then also be no reason to think further about the problem.*
Recognizing vs. searching for a proof

Given $\varphi$ and $n$, the difference between:

1. checking whether a given $\pi$ with $|\varphi| \leq n$ is a proof of $\varphi$ efficiently

2. finding such a $\pi$ or reporting no one exists efficiently

is self-evident.

It’s the difference between Diligence and Creativity.
Gödel’s question, or the modified version of Hilbert’s Dream, is still wide open despite tremendous efforts of more than 40 years.

This is one of the seven million-dollar Clay Millennium Prize Problems whether \( P = NP \).
S. Smale (Fields Medal, 1966)

\[ \text{P versus NP is a gift to mathematics from computer science.} \]

I will try to convince you

\[ \text{P versus NP is also gift to philosophy from computer science.} \]
Another Example

(S. Aaronson)
According to the Great Internet Mersenne Prime Search, the largest prime number known today is

\[ 2^{57,885,161} - 1. \]

Note its decimal representation can’t be read by a human. Then why can we be convinced that it’s a prime number?
What is knowing?

Possible explanations:
(a) the expression $2^{57,885,161} - 1$ picks out a unique positive integer, and
(b) that integer has been proven to be prime.

But what about

the first prime number larger than $2^{57,885,161} - 1$?
Knowing via efficient computation

We know how to compute the decimal representation of $2^{57,885,161} - 1$ efficiently, but not the first prime number larger than $2^{57,885,161} - 1$ (why?).
Concluding message

We will see through this course that there are many philosophical questions which can be studied from a computational perspective.

Warning: one should also refrain from over-interpretation.