Fundamental Questions

Q. What is a general-purpose computer?
Q. Are there limits on the power of digital computers?
Q. Are there limits on the power of machines we can build?

Pioneering work in the 1930s.
- Princeton == center of universe.
- Hilbert, Gödel, Turing, Church, von Neumann.
- Automata, languages, computability, universality, complexity, logic.

David Hilbert
Kurt Gödel
Alan Turing
Alonzo Church
John von Neumann

Turing Machine

Desiderata. Simple model of computation that is "as powerful" as conventional computers.

Intuition. Simulate how humans calculate.

Ex. Addition.

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>+</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Tape.
- Stores input, output, and intermediate results.
- One arbitrarily long strip, divided into cells.
- Finite alphabet of symbols.

Tape head.
- Points to one cell of tape.
- Reads a symbol from active cell.
- Writes a symbol to active cell.
- Moves left or right one cell at a time.

State. What machine remembers.
State transition diagram. Complete description of what machine will do.

Tape (before) ... # 1 1 0 0 + 1 0 1 1 # ...
Tape (after) ... # 1 1 1 0 + 1 0 1 1 # ...

Binary Adder
subtract one from \( y \) find plus sign
find right end of \( y \) add one to \( x \) clean up halt

0 : 1 # : 1 + : # 1 : 0 0 : 1 # : 1 + : # 1 : 0 # : 1 # : 1 + : # 1 : 0
Turing Machine: Initialization and Termination

**Initialization.** Set input on some portion of tape; set tape head.

**Termination.** Stop if enter yes, no, or halt state.

---

Program and Data

**Data.** Sequence of symbols (interpreted one way).

**Program.** Sequence of symbols (interpreted another way).

---

Ex 1. A **compiler** is a program that takes a program in one language as input and outputs a program in another language.

---

Ex 2. **Self-replication.** [von Neumann 1940s]

Print the following statement twice, the second time in quotes.

"Print the following statement twice, the second time in quotes."
Data. Sequence of symbols (interpreted one way).
Program. Sequence of symbols (interpreted another way).

Ex 3. Self-replication. [Watson-Crick 1953]

7.5 Universality
Universal Turing Machine

**Universal Turing machine** \(U\). Given input tape with \(x\) and \(M\), universal Turing machine \(U\) outputs \(M(x)\).

\[
\begin{array}{c}
\text{data } x \\
\downarrow
\end{array}
\begin{array}{c}
\text{program } M \\
\downarrow
\end{array}
\begin{array}{c}
\text{data } x \\
\downarrow
\end{array}
\begin{array}{c}
\text{output } M(x) \\
\downarrow
\end{array}
\begin{array}{c}
\text{output } M(x) \\
\downarrow
\end{array}
\]

**Church-Turing Thesis**

**Church Turing thesis (1936).** Turing machines can do anything that can be described by any physically harnessable process of this universe.

**Remark.** "Thesis" and not a mathematical theorem because it’s a statement about the physical world and not subject to proof. But can be falsified

**Implications.**
- No need to seek more powerful machines or languages.
- Enables rigorous study of computation (in this universe).

**Bottom line.** Turing machine is a simple and universal model of computation.

**Church-Turing Thesis: Evidence**

**Evidence.**
- 7 decades without a counterexample.
- Many, many models of computation that turned out to be equivalent.

<table>
<thead>
<tr>
<th>model of computation</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enhanced Turing machines</td>
<td>multiple heads, multiple tapes, 2D tape, non-determinism</td>
</tr>
<tr>
<td>untyped lambda calculus</td>
<td>method to define and manipulate functions</td>
</tr>
<tr>
<td>recursive functions</td>
<td>functions dealing with computation on integers</td>
</tr>
<tr>
<td>unrestricted grammars</td>
<td>iterative string replacement rules used by linguists</td>
</tr>
<tr>
<td>extended L-systems</td>
<td>parallel string replacement rules that model plant growth</td>
</tr>
<tr>
<td>programming languages</td>
<td>Java, C, C++, Perl, Python, PHP, Lisp, PostScript, Excel</td>
</tr>
<tr>
<td>random access machines</td>
<td>registers plus main memory, e.g., TOY, Pentium</td>
</tr>
<tr>
<td>cellular automata</td>
<td>cells which change state based on local interactions</td>
</tr>
<tr>
<td>quantum computer</td>
<td>compute using superposition of quantum states</td>
</tr>
<tr>
<td>DNA computer</td>
<td>compute using biological operations on DNA</td>
</tr>
</tbody>
</table>

**Consequences.** Your laptop (a UTM) can do any computational task.
- Java programming.
- Pictures, music, movies, games.
- Email, browsing, downloading files, telephony.
- Word-processing, finance, scientific computing.
- ...

Again, it [the Analytical Engine] might act upon other things besides numbers... the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent. — Ada Lovelace
7.6 Computability

Halting Problem

Halting problem. Write a Java function that reads in a Java function f and its input x, and decides whether f(x) results in an infinite loop.

Ex. Does f(x) terminate?

```java
public void f(int x) {
    while (x != 1) {
        if (x % 2 == 0) x = x / 2;
        else x = 3*x + 1;
    }
}
```

relates to famous open math conjecture

- f(6): 6 3 10 5 16 8 4 2 1
- f(27): 27 82 41 124 62 31 94 47 142 71 214 107 322 97 292 146 73 220 110 55 166 83 250 125 375 187 562 281 844 422 211 634 317 952 476 238 119 358 179 536 268 134 67 202 101 304 152 76 38 19 58 29 170 85 254 127 382 191 574 287 862 431 1294 647 1942 971 2914 1457 4372 2186 1093 3280 1640 820 410 205 616 308 154 77 232 116 58 29 88 44 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
- f(-17): -17 -50 -25 -74 -37 -110 -55 -164 -82 -41 -122 ... -17 ...
A yes-no problem is undecidable if no Turing machine exists to solve it.

Proof intuition: lying paradox.
- Divide all statements into two categories: truths and lies.
- How do we classify the statement: I am lying.

Key element of lying paradox and halting proof: self-reference.

Theorem. [Turing 1937] The halting problem is undecidable.

Halting Problem Proof

Assume the existence of \( \text{halt}(f, x) \):
- Input: a function \( f \) and its input \( x \).
- Output: true if \( f(x) \) halts, and false otherwise.

Note. \( \text{halt}(f, x) \) does not go into infinite loop.

We prove by contradiction that \( \text{halt}(f, x) \) does not exist.
- Reductio ad absurdum: if any logical argument based on an assumption leads to an absurd statement, then assumption is false.

\[
\text{encode } f \text{ and } x \text{ as strings}
\]

### public boolean \( \text{halt}(String f, String x) \) {
  \[
  \text{if (something terribly clever)} \quad \text{return true;}
  \text{else}
  \text{return false;}
  \]
}

hypothetical halting function

Construct function \( \text{strange}(f) \) as follows:
- If \( \text{halt}(f, f) \) returns true, then \( \text{strange}(f) \) goes into an infinite loop.
- If \( \text{halt}(f, f) \) returns false, then \( \text{strange}(f) \) halts.

\[
f \text{ is a string so legal (if perverse)}
\text{ to use for second input}
\]

### public void \( \text{strange}(String f) \) {
  \[
  \text{if (\( \text{halt}(f, f) \))}
  \text{// an infinite loop}
  \text{while (true) {}}
  \]
}

In other words:
- If \( f(f) \) halts, then \( \text{strange}(f) \) goes into an infinite loop.
- If \( f(f) \) does not halt, then \( \text{strange}(f) \) halts.

Call \( \text{strange()} \) with \( \text{ITSELF} \) as input.
- If \( f(f) \) halts, then \( \text{strange} \) goes into an infinite loop.
- If \( f(f) \) does not halt, then \( \text{strange}(f) \) halts.

Either way, a contradiction. Hence \( \text{halt}(f, x) \) cannot exist.
Q. Why is debugging hard?
A. All problems below are undecidable.

Halting problem. Give a function f, does it halt on a given input x?
Totality problem. Give a function f, does it halt on every input x?
No input halting problem. Give a function f with no input, does it halt?
Program equivalence. Do two functions f and always return same value?
Uninitialized variables. Is the variable x initialized before it’s used?
Dead code elimination. Does this statement ever get executed?

More Undecidable Problems

Hilbert's 10th problem.

Deive a process according to which it can be determined by a finite number of operations whether a given multivariate polynomial has an integral root.

- $f(x, y, z) = 6x^3y z^2 + 3xy^2 - x^3 - 10$. yes: $f(5, 3, 0) = 0$.
- $f(x, y) = x^2 + y^2 - 3$. no.

Definite integration. Given a rational function $f(x)$ composed of polynomial and trig functions, does $\int f(x) \, dx$ exist?

- $g(x) = \cos x (1 + x^2)^{-1}$ yes, $\int g(x) \, dx = \pi/e$.
- $h(x) = \cos x (1 - x^2)^{-1}$ no, $\int h(x) \, dx$ undefined.

Optimal data compression. Find the shortest program to produce a given string or picture.

Mandelbrot set (40 lines of code)

Virus identification. Is this program a virus?

Private Sub AutoOpen()
On Error Resume Next
If System.PrivateProfileString("", CURRENT_USER\Software\Microsoft\Office\9.0\Word\Security", "Level") <> "" Then
CommandBars("Macro").Controls("Security...").Enabled = False
For oo = 1 To AddyBook.AddressEntries.Count
Peep = AddyBook.AddressEntries(x)
BreakUncOffASlice.Recipients.Add Peep
x = x + 1
If x > 50 Then oo = AddyBook.AddressEntries.Count
Next
. . .
BreakUncOffASlice.Subject = "Important Message From " & Application.UserName
BreakUncOffASlice.Body = "Here is that document you asked for ... don't show anyone else ;-)"
. . .

Can write programs in MS Word. This statement disables security.

Melissa virus
March 28, 1999
Mathematics. Any formal system powerful enough to express arithmetic.

Complete. Can prove truth or falsity of any arithmetic statement.
Consistent. Can’t prove contradictions like 2 + 2 = 5.
Decidable. Algorithm exists to determine truth of every statement.

Q. [Hilbert, 1900] Is mathematics complete and consistent?
A. [Gödel’s Incompleteness Theorem, 1931] No!!

Q. [Hilbert’s Entscheidungsproblem] Is mathematics decidable?
A. [Church 1936, Turing 1936] No!

Alan Turing

Alan Turing (1912-1954).
• Father of computer science.
• Computer science’s "Nobel Prize" is called the Turing Award.

It was not only a matter of abstract mathematics, not only a play of symbols, for it involved thinking about what people did in the physical world. … It was a play of imagination like that of Einstein or von Neumann, doubting the axioms rather than measuring effects… What he had done was to combine such a naïve mechanistic picture of the mind with the precise logic of pure mathematics. His machines — soon to be called Turing machines — offered a bridge, a connection between abstract symbols, and the physical world. — John Hodges