Universality and Computability

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.
- Automata, languages, computability, universality, complexity, logic.

7.4 Turing Machines

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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>9</td>
</tr>
</tbody>
</table>

Alan Turing (1912-1954)
Last lecture: DFA

Tape.
- Stores input.
- 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.
- Moves right one cell at a time.

This lecture: Turing machine

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.

Last lecture: Deterministic Finite State Automaton (DFA)

Simple machine with N states.
- Begin in start state.
- Read first input symbol.
- Move to new state, depending on current state and input symbol.
- Repeat until last input symbol read.
- Accept input string if last state is labeled Y.

This lecture: Turing Machine

Simple machine with N states.
- Begin in start state.
- Read first input symbol.
- Move to new state and write new symbol on tape, depending on current state and input symbol.
- Move tape head left if state is labeled L, right if state is labeled R.
- Repeat until entering a state labeled H.
- Accept input string state is labeled Y, reject if N
Simple machine with $N$ states.
• Begin in start state.
• Read first input symbol.
• Move to new state and write new symbol on tape, depending on current state and input symbol.
• Move tape head left if state is labeled $L$, right if state is labeled $R$.
• Repeat until entering a state labeled $H$.
• Accept input string state is labeled $Y$, reject if $N$.
Q. What happens if we try to decrement 0?
TM Example 4: Binary Adder

Ex. Use simulator to understand how this TM works.

7.5 Universality

Universal Machines and Technologies

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.

Your program is DATA to a compiler.
Program and Data

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

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

**Ex 2.** A simulator is a program that takes a program for one machine as input and simulates the operation of that program.

![Simulator Diagram]

Universal Turing Machine

**Turing machine** $M$. Given input tape $x$, Turing machine $M$ outputs $M(x)$.

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

![Universal Turing Machine Diagram]

**TM intuition.** Software program that solves one particular problem.

**UTM intuition.** Hardware platform that can implement any algorithm.

**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.
- ...
Universal Turing Machine

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

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.

Use simulation to prove models equivalent.
• TOY simulator in Java
• Java compiler in TOY.

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.

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, nondeterminism</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>

Lindenmayer Systems: Synthetic Plants

A Puzzle: Post's Correspondence Problem

Given a set of cards:
- N card types (can use as many copies of each type as needed).
- Each card has a top string and bottom string.

Example 1:

<table>
<thead>
<tr>
<th>BAB</th>
<th>A</th>
<th>AB</th>
<th>BA</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>ABA</td>
<td>B</td>
<td>B</td>
</tr>
</tbody>
</table>

Puzzle:
- Is it possible to arrange cards so that top and bottom strings match?

Solution 1.
- Yes.

Given a set of cards:
- N card types (can use as many copies of each type as needed).
- Each card has a top string and bottom string.

Example 1:

<table>
<thead>
<tr>
<th>BAB</th>
<th>A</th>
<th>AB</th>
<th>BA</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>ABA</td>
<td>B</td>
<td>B</td>
</tr>
</tbody>
</table>

Puzzle:
- Is it possible to arrange cards so that top and bottom strings match?

Solution 1.
- Yes.
A Puzzle: Post's Correspondence Problem

Given a set of cards:
• N card types (can use as many copies of each type as needed).
• Each card has a top string and bottom string.

Example 2:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>ABA</th>
<th>B</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>BAB</td>
<td>B</td>
<td>A</td>
<td>B</td>
</tr>
</tbody>
</table>

N = 4

Puzzle:
• Is it possible to arrange cards so that top and bottom strings match?

Solution 2.

No. First card in solution must contain same letter in leftmost position.

Challenge:
• Write a program to take cards as input and solve the puzzle.

Surprising fact:
• It is NOT POSSIBLE to write such a program!
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.

Easy for some functions, not so easy for others.

Ex. Does \( f(x) \) terminate?

```java
public void \texttt{f} (\texttt{int} \texttt{x})
{
  \texttt{while} (\texttt{x} \neq 1)
  {
    \texttt{if} \ (\texttt{x} \mod 2 == 0) \texttt{x} = \texttt{x} / 2;
    \texttt{else} (\texttt{x} \mod 2 == 0) \texttt{x} = 3*\texttt{x} + 1;
  }
}
```

\( f(6): \quad 6 \ 3 \ 10 \ 5 \ 16 \ 8 \ 4 \ 2 \ 1 \)
\( f(27): \quad 27 \ 82 \ 41 \ 124 \ 62 \ 31 \ 94 \ 47 \ 142 \ 71 \ 214 \ 107 \ 322 \ 64 \ 32 \ 16 \ 8 \ 4 \ 2 \ 1 \)
\( f(-17): \quad -17 \ -50 \ -25 \ -74 \ -37 \ -110 \ -55 \ -164 \ -82 \ -41 \ -122 \ -61 \ -184 \ -92 \ -46 \ -23 \ -70 \ -35 \ -106 \ -53 \ -160 \ -80 \ -40 \ -20 \ -10 \ -5 \ -16 \ -8 \ -4 \ -2 \ 1 \)

Undecidable Problem

A yes-no problem is undecidable if no Turing machine exists to solve it.

and (by universality) no Java program either.

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

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.

Halting Problem Proof

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

```java
public boolean \texttt{halt} (\texttt{String} \texttt{f}, \texttt{String} \texttt{x})
{
  \texttt{if} (something terribly clever) \texttt{return} \texttt{true};
  \texttt{else} \texttt{return} \texttt{false};
}
```

Hypothetical halting function

Halting Problem Proof

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

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

```java
\texttt{f} is a string as legal (if perverse)
to use for second input

public void \texttt{strange} (\texttt{String} \texttt{f})
{
  \texttt{if} (\texttt{halt} (\texttt{f}, \texttt{f}))
  \{
    \texttt{// an infinite loop}
    \texttt{while} (\texttt{true}) \texttt{;} \texttt{
  }
}
```

Hypothetical halting function
Assume the existence of \( \text{halt}(f, x) \):

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

Construct function \( \text{strange}(f) \) as follows:

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

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 ITSELF as input.

- If \( \text{strange}(\text{strange}) \) halts then \( \text{strange}(\text{strange}) \) does not halt.
- If \( \text{strange}(\text{strange}) \) does not halt then \( \text{strange}(\text{strange}) \) 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 \( g \) always return same value?
- **Uninitialized variables.** Is the variable \( x \) initialized before it’s used?
- **Dead-code elimination.** Does this statement ever get executed?
Post’s Correspondence Problem

Given a set of cards:
• N card types (can use as many copies of each type as needed).
• Each card has a top string and bottom string.

Puzzle:
• Is it possible to arrange cards so that top and bottom strings match?

Challenge:
• Write a program to take cards as input and solve the puzzle.

is UNDECIDABLE

More Undecidable Problems

Hilbert’s 10th problem.

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

• \( f(x, y, z) = 6x^3 y z^2 + 3xy^2 - x^3 - 10 \).
• \( f(x, y) = x^2 + y^2 - 3 \).

Yes: \( f(5, 3, 0) = 0 \).
No.

Definite integration.

Given a rational function \( f(x) \) composed of polynomial and trig functions.

Does \( \int_{-\infty}^{\infty} f(x) \, dx \) exist?

• \( g(x) = \cos x (1 + x^2)^{-1} \)
• \( h(x) = \cos x (1 - x^2)^{-1} \)

Yes, \( \int_{-\infty}^{\infty} g(x) \, dx = \pi/e \).
No, \( \int_{-\infty}^{\infty} h(x) \, dx \) undefined.

More Undecidable Problems

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

Virus identification.

Is this program a virus?

Private Sub AutoOpen()
On Error Resume Next
If System.PrivateProfileString(S“, 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(oo)
BreakUnoffAddIn.Recipients.Add Peep
x = x + 1
If x > 50 Then oo = AddyBook.AddressEntries.Count
Next oo
BreakUnoffAddIn.Subject = “Important Message From “ & Application.UserName
BreakUnoffAddIn.Body = “Here is that document you asked for ... don’t show anyone else ;-)“
Next oo

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

Mandelbrot set (40 lines of code)

Mellisa virus
March 28, 1999
Turing's Key Ideas

- Turing machine. formal model of computation
- Program and data. encode program and data as sequence of symbols
- Universality. concept of general-purpose, programmable computers
- Church-Turing thesis. computable at all == computable with a Turing machine
- Computability. inherent limits to computation

Hailed as one of top 10 science papers of 20th century.


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