Lecture 19: Universality and Computability

Fundamental Questions

Universality. What is a general purpose computer?

Computability. Are there problems that no machine can solve?

Church-Turing thesis. Are there limits on the power of machines that we can build?

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

Turing Machine: Components

Alan Turing sought the most primitive model of a computing device.

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.

Java: As Powerful As Turing Machine

Turing machines are equivalent in power to TOY and Java.
- Can use Java to solve any problem that can be solved with a TM.
- Can use TM to solve any problem that can be solved with a TOY.
- Can use TOY to solve any problem that can be solved with Java.

Java simulator for Turing machines.

```java
State state = start;
while (true) {
    char c = tape.readSymbol();
    tape.write(state.symbolToWrite(c));
    state = state.next(c);
    if (state.isLeft()) tape.moveLeft();
    else if (state.isRight()) tape.moveRight();
    else if (state.isHalt()) break;
}
```
Turing Machine: As Powerful As TOY Machine

Turing machines are equivalent in power to TOY and Java.
- Can use Java to solve any problem that can be solved with a TM.
- Can use TM to solve any problem that can be solved with a TOY.
- Can use TOY to solve any problem that can be solved with Java.

Turing machine simulator for TOY programs.
- Encode state of memory, registers, pc, onto Turing tape.
- Design TM states for each instruction.
- Can do because all instructions:
  - examine current state
  - make well-defined changes depending on current state

Java, Turing Machines, and TOY

Turing machines are equivalent in power to TOY and Java.
- Can use Java to solve any problem that can be solved with a TM.
- Can use TM to solve any problem that can be solved with a TOY.
- Can use TOY to solve any problem that can be solved with Java.

Also works for:
- C, C++, Python, Perl, Excel, Outlook, . . .
- Mac, PC, Cray, Palm pilot, . . .
- TiVo, Xbox, Java cell phone, . . .

Does not work:
- DFA or regular expressions.
- Gaggia espresso maker.

TOY: As Powerful As Java

Turing machines are equivalent in power to TOY and Java.
- Can use Java to solve any problem that can be solved with a TM.
- Can use TM to solve any problem that can be solved with a TOY.
- Can use TOY to solve any problem that can be solved with Java.

TOY simulator for Java programs.
- Variables, loops, arrays, functions, linked lists, . . .
- In principle, can write a Java-to-TOY compiler!

Not Enough Storage?

Implicit assumption.
- TOY machine and Java program have unbounded amount of memory.
- Otherwise Turing machine is strictly more powerful.
- Is this assumption reasonable?
Universal Turing Machine

Java program: solves one specific problem.
TOY program: solves one specific problem.
TM: solves one specific problem.

Java simulator in Java: Java program to simulate any Java program.
TOY simulator in TOY: TOY program to simulate any TOY program.
UTM: Turing machine that can simulate any Turing machine.

General purpose machine.
- UTM can implement any algorithm.
- Your laptop can do any computational task: word-processing, pictures, music, movies, games, finance, science, email, Web, ...

Church-Turing Thesis

Implications:
- No need to seek more powerful machines.
- If a computational problem can't be solved with a Turing machine, then it can't be solved on any physical computing device.

Remarks:
- "Thesis" and not a mathematical theorem because it's a statement about the physical world and not subject to proof.

Turing machine: a simple and universal model of computation.
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.

```
int number = ...; // some integer that equals the sum of its proper divisors

if (number == 0) { ... } // handle the case where the number is 0
```

**Ex:** is there a perfect number of the form: \( 1, 1+x, 1+2x, 1+3x, ... \)
- \( x = 1 \): halts when \( n = 28 = 1 + 2 + 4 + 7 + 14 \).
- \( x = 2 \): finding odd perfect number is famous open math problem.

### Other Universal Models of Computation

<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>A method to define and manipulate functions. Basis of functional programming language like Lisp and ML.</td>
</tr>
<tr>
<td>Recursive Functions</td>
<td>Functions dealing with computation on natural numbers.</td>
</tr>
<tr>
<td>Unrestricted Grammars</td>
<td>Iterative string replacement rules used by linguists to describe natural languages.</td>
</tr>
<tr>
<td>Extended L-Systems</td>
<td>Parallel string replacement rules that model the growth of plants.</td>
</tr>
<tr>
<td>Cellular Automata</td>
<td>Boolean array of cells whose values change according only to the state of the adjacent cells, e.g., Game of Life.</td>
</tr>
<tr>
<td>Random Access Machines</td>
<td>Finitely many registers plus memory that can be accessed with an integer address. TOY, 65, Pentium IV.</td>
</tr>
<tr>
<td>Programming Languages</td>
<td>Java, C, C++, Perl, Python, PHP, Lisp, PostScript, Excel</td>
</tr>
</tbody>
</table>

Undecidable Problem

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

**Theorem (Turing, 1937).** The halting problem is undecidable.
- No Turing machine can solve the halting problem.
- By universality, not possible to write a Java function either.

**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 paradox:** self-reference.

---

Take any definite unsolved problem, such as the question as to the irrationality of the Euler-Mascheroni constant \( \gamma \), or the existence of an infinite number of prime numbers of the form \( 2^{n}+1 \). However unapproachable these problems may seem to us and however helpless we stand before them, we have, nevertheless, the firm conviction that their solution must follow by a finite number of purely logical processes.

- David Hilbert, in his 1900 address to the International Congress of Mathematics
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.

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.

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.
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.

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.

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.

Consequences

Halting problem is not "artificial."

- Undecidable problem reduced to simplest form to simplify proof.
- Self-reference not essential.
- Closely related to practical problems.

Examples.

- \( f(x, y, z) = 6x^2y - 2y^2 - 3z \): yes: \( f(5, 3, 0) = 0 \)
- \( f(x, y) = x^2 + y^2 - 3 \): no
- \( f(x, y, z) = x^n + y^n - z^n \)
- \( f(2, 3, 4, 5, 6) \): yes if \( n = 2 \), \( x = 3 \), \( y = 4 \), \( z = 5 \)
- \( f(2, 3, 4) \): no if \( n = 3 \) and \( x, y, z > 0 \). (Fermat’s Last Theorem)

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.”

Examples.

- \( f(x, y) = 6x^2y - 2y^2 - 3z \): yes: \( f(5, 3, 0) = 0 \)
- \( f(x) = x^2 + y^2 - 3 \): no
- \( f(x, y) = x^n + y^n - z^n \)
- \( f(2, 3, 4, 5, 6) \): yes if \( n = 2 \), \( x = 3 \), \( y = 4 \), \( z = 5 \)
- \( f(2, 3, 4) \): no if \( n = 3 \) and \( x, y, z > 0 \). (Fermat’s Last Theorem)
More Undecidable Problems

Polygonal tiling. Given a polygon, is it possible to tile the whole plane with copies of that shape?

Difficulty. Tilings may exist, but be aperiodic!

Reference: http://www.uwgb.edu/dutchs/symmetry/aperiod.htm

Virus identification. Is this program a virus?

```
Private Sub AutoOpen()
 On Error Resume Next
 If System.PrivateProfileString("",_CURRENT_USER\Software\Microsoft\Office\8.0\Word\Security", _"Level") <> ":" Then
 CommandBars("Macro").Controls("Security...").Enabled = False
 For co = 1 To Addinbox.AddressEntries.Count
 Peep = Addinbox.AddressEntries(co)
 BreakUnaffAll MAIL.Recipients.Add Deep
 k = k + 1
 If k > 10 Then co = Addinbox.AddressEntries.Count
 Next co
 BreakUnaffAllioSubject = "Important Message From " & Application.UserName
 BreakUnaffAllio.Body = "Here is that document you asked for... don't show anyone else :-("
 . . .
 Melissa Virus, March 28, 1999
```

Implications of Computability

Step-by-step reasoning.
- We assume that it will solve any technical or scientific problem.
- Not quite says the halting problem.

Practical implications.
- Work with limitations.
- Recognize and avoid undecidable problems.
- Anything that is (or could be) like a computer has the same flaw.

Speculative Models of Computation

Rule of thumb. Any pile of junk that has state and a deterministic set of rules is universal, and hence has intrinsic limitations!

<table>
<thead>
<tr>
<th>Model of Computation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantum Computer</td>
<td>Compute using the superposition of quantum states.</td>
</tr>
<tr>
<td>Billiard Ball Computer</td>
<td>Colliding billiard balls with barriers and elastic collisions.</td>
</tr>
<tr>
<td>DNA Computer</td>
<td>Compute using biological operations on DNA strands.</td>
</tr>
<tr>
<td>Soliton Collision System</td>
<td>Time-gated Manakov spatial solitons in a homogeneous medium.</td>
</tr>
<tr>
<td>Dynamical System</td>
<td>Dynamics based computing based on chaos.</td>
</tr>
<tr>
<td>Logic</td>
<td>Formal mathematics.</td>
</tr>
<tr>
<td>Human Brain</td>
<td>???</td>
</tr>
</tbody>
</table>
Turing’s Key Ideas

Turing’s 4 key ideas:

- Computing is the same as manipulating symbols.
  - Encode numbers as strings.
- Computable at all = computing with a Turing machine.
- Existence of Universal Turing machine.
  - Church-Turing thesis.
- Undecidability of the Halting problem.
  - Computers have inherent limitations.

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