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.

Universality

Which one of the following does not belong?


Cray


Dell PC


iMac


Espresso maker


Palm Pilot

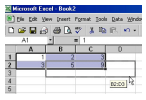

Xbox


Tivo

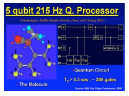

Turing machine


TOY

Java language


Excel


Java cell phone



Quantum computer


DNA computer

Perl language

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.

```
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 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

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!

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:

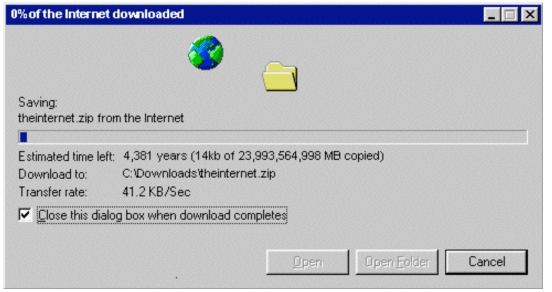
- C, C++, Python, Perl, Excel, Outlook, ...
- Mac, PC, Cray, Palm pilot, ...
- TiVo, Xbox, Java cell phone, ...

Does not work for Gaggia espresso maker or DFA.

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.

⇧ including itself!

UTM is a "general purpose machine."

- Can be used to implement any algorithm.
- Anticipated development of first general purpose computer.

Church-Turing Thesis

Church-Turing thesis (1936).

- Q. Which decision problems can a Turing machine solve?
- A. Any decision problem that any real computer can solve.

"Thesis" and not a mathematical theorem.

- Can't be proved because it's a statement about the physical world.

Implications:

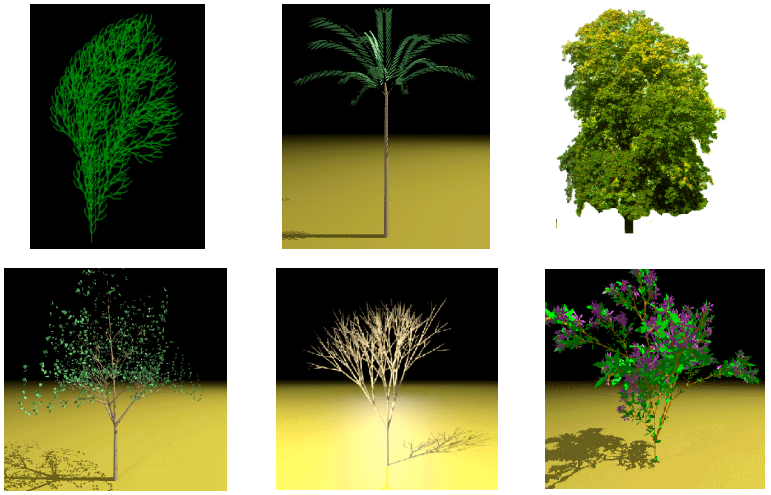
- No need to seek more powerful machines.
- If a decision problem can't be solved by any Turing machine, then it can't be solved on ANY physical computing device.

Turing machine is a simple and universal model of computation.

Other Universal Models of Computation

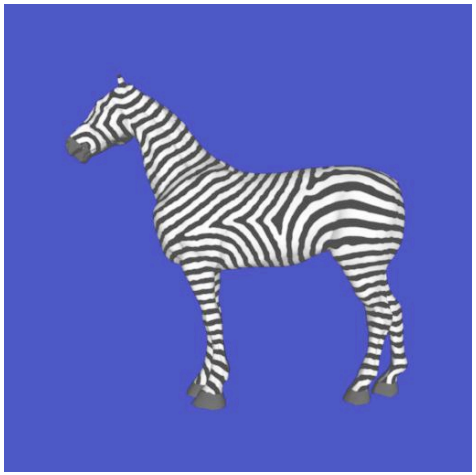
Model of Computation	Description
Enhanced Turing Machines	Multiple heads, multiple tapes, two dimensional tape, nondeterminism.
Lambda Calculus	A method to define and manipulate functions. Basis of functional programming language like Lisp and ML.
Recursive Functions	Functions dealing with computation on the natural numbers.
Unrestricted Grammars	Iterative string replacement rules used by linguists to describe natural languages.
Extended L-Systems	Parallel string replacement rules that model the growth of plants.
Cellular Automata	Boolean array of cells whose values change according only to the state of the adjacent cells, e.g., Game of Life.
Random Access Machines	Finitely many registers plus memory that can be accessed with an integer address. TOY, PowerPC, Pentium IV.
Programming Languages	Java, C, C++, Perl, Python, PHP, Lisp, PostScript, Excel

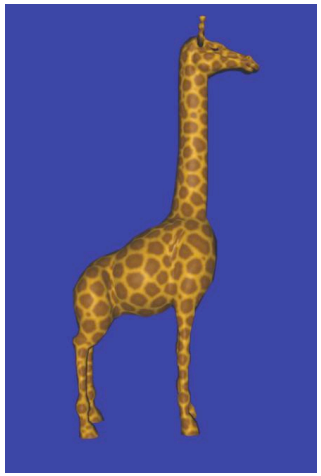
Lindenmayer Systems: Synthetic Plants



Reference: http://astronomy.swin.edu.au/~pbourke/modelling/plants/

Cellular Automata: Synthetic Zoo





Reference: *Generating Textures on Arbitrary Surfaces Using Reaction-Diffusion*" by Greg Turk, SIGGRAPH, 1991.

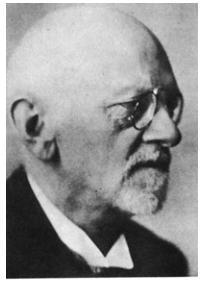
Implicit Physical Principles

Turing machine embodies physical constraints to which all concrete computational processes are subjected.

3 of the Fredkin-Toffoli axioms.
E. F. Fredkin and T. Toffoli. Conservative logic. *International Journal of Theoretical Physics*, 21(3/4):219--253, 1982.

- *The speed of propagation of information is bounded.*
  - no "action at a distance"
  - TM head only move to adjacent cells
- *The amount of information which can be encoded in the state of a finite system is bounded.*
  - TM stores finitely many symbols per tape cell
- *It is possible to construct macroscopic, dissipative physical devices which perform in a recognizable and reliable way the logical functions AND, OR, and FAN-OUT.*
  - can fabricate TM out of physical parts, and run it reliably

Computability



Hilbert

" Every mathematical problem can be solved. We are convinced of that. After all, one of the things that attracts us most when we apply ourselves to a mathematical problem is precisely that within us we always hear the call: here is the problem, search for the solution, you can find it by pure thought, for in mathematics there is no *ignorabimus*. "

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:

	BAB	A	AB	BA
	A	ABA	B	B
	0	1	2	3

N = 4

Puzzle:

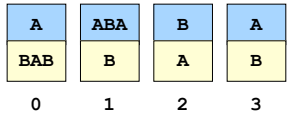
- Is it possible to arrange cards so that top and bottom strings are the same?

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:

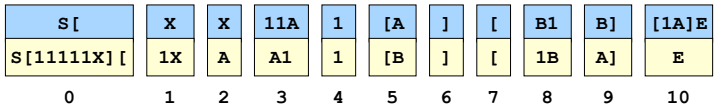


N = 4

Puzzle:

- Is it possible to arrange cards so that top and bottom strings are the same?

PCP Puzzle Contest



Contest:

- Additional restriction: string must start with 'S'.
- Be the first to solve this puzzle and earn extra credit!
- Check solution from Lectures web page.

Hopeless challenge for the bored:

- Write a Java program that reads in any given set of Post cards, and determines whether or not there is a solution.

Reference: this puzzle created by Andrew Appel.

Halting Problem

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

- Infinite loop often signifies a bug.
- Would be a useful compiler feature.

integer that equals the sum of its proper divisors

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.

```
public void f(int x) {
   for (long n = 1; true; n = n + x) {
      long sum = 0;
      for (long i = 1; i < n; i++)
         if (n % i == 0) sum = sum + i;
      if (sum == n) return;
   }
}
```

halt if n is perfect

Undecidable Problem

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

Theorem (Alan Turing, 1937). 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.

Halting Problem Proof

Assume the existence of `halt(f,x)`:

- Input: a function `f` and its input `x`.
- Output: `true` if `f(x)` halts, and `false` otherwise.
- Note: `halt(f,x)` does not go into infinite loop.

We prove by contradiction that `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.

encode f and x as strings

```
public boolean halt(String f, String x) {
   if (???) return true;
   else     return false;
}
```


Halting Problem Proof

Assume the existence of `halt(f,x)`:

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

Construct function `strange(f)` as follows:

- If `halt(f,f)` returns `true`, then `strange(f)` goes into an infinite loop
- If `halt(f,f)` returns `false`, then `strange(f)` halts.

f is a string so legal to use for either input

```
public void strange(String f) {
   if (halt(f, f)) {
      while (true)
         ;
   }
}
```


Halting Problem Proof

Assume the existence of `halt(f,x)`:

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

Construct function `strange(f)` as follows:

- If `halt(f,f)` returns `true`, then `strange(f)` goes into an infinite loop
- If `halt(f,f)` returns `false`, then `strange(f)` halts.

In other words:

- If `f(f)` halts, then `strange(f)` goes into an infinite loop.
- If `f(f)` does not halt, then `strange(f)` halts.

Call `strange()` with ITSELF as input.

- If `strange(strange)` halts then `strange(strange)` does not halt.
- If `strange(strange)` does not halt then `strange(strange)` halts.

Either way, a contradiction. Hence `halt(f,x)` cannot exist.

Consequences

Halting problem is not "artificial."

- Undecidable problem reduced to simplest form to simplify proof.
- Self-reference not essential.
  - do not need to call a function with itself as input
  - halting problem still undecidable even if function takes no input!
- Closely related to practical problems.

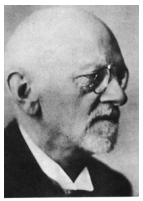
More Undecidable Problems

Hilbert's 10th problem.

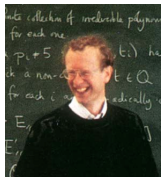
- *"Devise a process according to which it can be determined by a finite number of operations whether a given multivariate polynomial has an integral root."*

Examples.

- $f(x, y, z) = 6x^3 y z^2 + 3xy^2 - x^3 - 10$. ⬅ yes: f(5, 3, 0) = 0
- $f(x, y) = x^2 + y^2 - 3$. ⬅ no
- $f(x, y, z) = x^n + y^n - z^n$ ⬅ yes if n = 2, x = 3, y = 4, z = 5
  ⬅ no if $n \geq 3$ and $x, y, z > 0$. (Fermat's Last Theorem)



Hilbert



Andrew Wiles, 1995

More Undecidable Problems

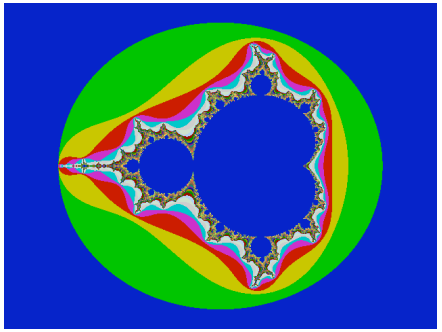
Hilbert's 10th problem.
Program equivalence

- Do two programs always produce the same output?
  - where's my bug?
  - useful for debugging

More Undecidable Problems

Hilbert's 10th problem.
Program equivalence.
Optimal data compression.

- Find the shortest program to produce a given string or picture.



Mandelbrot Set (40 lines of code)

More Undecidable Problems

Hilbert's 10th problem.
Program equivalence.
Optimal data compression.
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)
   BreakUmOffASlice.Recipients.Add Peep
   x = x + 1
   If x > 50 Then oo = AddyBook.AddressEntries.Count
Next oo
. . .
BreakUmOffASlice.Subject = "Important Message From " & Application.UserName
BreakUmOffASlice.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

More Undecidable Problems

Hilbert's 10th problem.
Program equivalence.
Optimal data compression.
Virus identification.
Halting problem.
Post correspondence problem.
. . . .

Impossible to write a Java program to solve any of these problem!

Implications of Computability

We "assume" that any step-by-step reasoning 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 may have intrinsic flaws!

Model of Computation	Description
Quantum Computer	Compute using the superposition of quantum states.
Billiard Ball Computer	Colliding billiard balls with barriers and elastic collisions.
DNA Computer	Compute using biological operations on DNA strands.
Soliton Collision System	Time-gated Manakov spatial solitions in a homogeneous medium.
Dynamical System	Dynamics based computing based on chaos.
Logic	Formal mathematics.
Human Brain	? ? ?
Universe	? ? ?

Summary

Turing's key ideas:

- Computing is same thing as manipulation symbols.
  - encode numbers as strings
- "Computable at all" == Computable with a TM.
  - Church-Turing thesis
- Existence of general-purpose computer (UTM).
  - programmable machine
- Halting problem is undecidable.
  - not possible to write a program to solve certain problems

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

Reference: *On Computable Numbers, With an Application to the Entscheidungsproblem* by A. M. Turing. In Proceedings of the London Mathematical Society, ser. 2. vol. 42 (1936-7), pp.230-265.