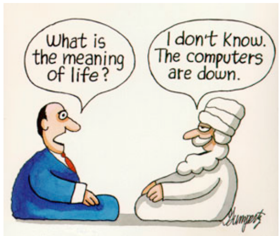
Universality and Computability



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.
- Hilbert, Gödel, Turing, Church, von Neumann.
- Automata, languages, computability, universality, complexity, logic.


David Hilbert


Kurt Gödel


Alan Turing


Alonzo Church


John von Neumann

7.4 Turing Machines

Challenge: Design simplest machine that is "as powerful" as conventional computers.


Alan Turing

Introduction to Computer Science • Sedgewick and Wayne • Copyright © 2007 • http://www.cs.Princeton.EDU/IntroCS

Turing Machine

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

Intuition. Simulate how humans calculate.

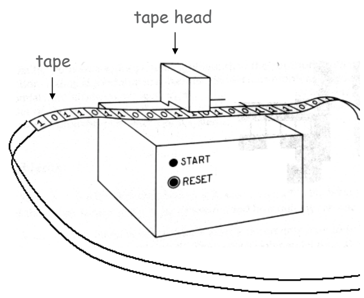
Ex. Addition.

			1	2	3	4	5	6		
		+	3	1	4	1	5	9		

Turing Machine: Tape

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.



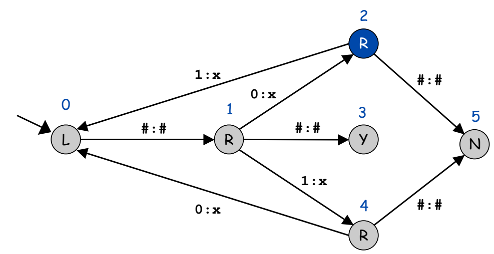
tape head

tape													
	...	#	1	1	0	0	+	1	0	1	1	#	...

Turing Machine: Fetch, Execute

States.
- Finite number of possible machine configurations.
- Determines what machine does and which way tape head moves.

State transition diagram.
- Ex. if in state 2 and input symbol is `1` then: overwrite the `1` with `x`, move to state 0, move tape head to left.

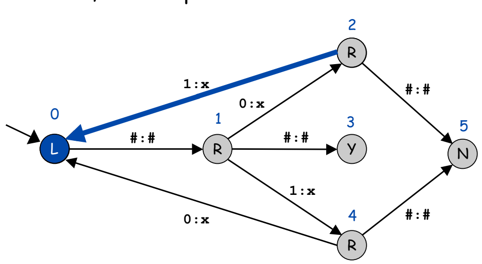


Before											
...	#	#	x	x	x	1	1	0	#	#	...

Turing Machine: Fetch, Execute

States.
- Finite number of possible machine configurations.
- Determines what machine does and which way tape head moves.

State transition diagram.
- Ex. if in state 2 and input symbol is `1` then: overwrite the `1` with `x`, move to state 0, move tape head to left.



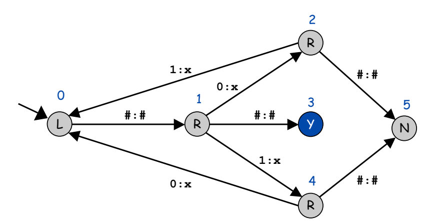
After											
...	#	#	x	x	x	x	1	0	#	#	...

Turing Machine: Initialization and Termination

Initialization.
- Set input on some portion of tape.
- Set tape head.
- Set initial state.

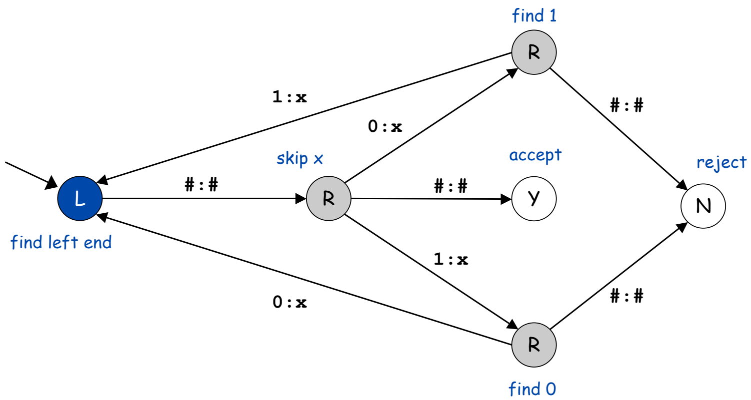
...	#	#	0	0	1	1	1	0	#	#	...

Termination.
- Stop if enter `yes`, `no`, or `halt` state.
- Infinite loop possible.
  - (definitely stay tuned !)



...	#	#	x	x	x	x	x	x	#	#	...

Example: Equal Number of 0's and 1's



...	#	#	0	0	1	1	1	0	#	#	...

7.5 Universality

Universality

Q. Which one of the following does not belong?


Cray


Dell PC

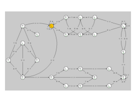

iMac


Espresso maker


Palm Pilot

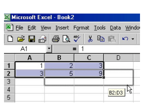

Xbox


Tivo

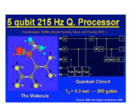

Turing machine


TOY

Java language

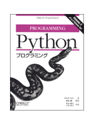

MS Excel


Java cell phone



Quantum computer


DNA computer



Python 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 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

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:

- DFA or regular expressions.
- Gaggia espresso maker.

Universal Turing Machine

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

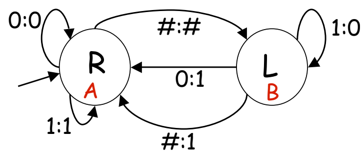
Java simulator in Java: Java program to simulate any Java program.
TOY simulator in TOY: TOY program to simulate any TOY program.
UTM: Turing machine that can simulate any Turing machine.

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, …

Representations of a Turing Machine

Graphical:



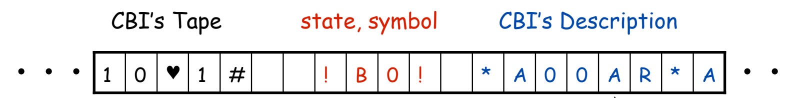
Continuous
Binary
Incrementer

Tabular:

Current state	Symbol read	Symbol to write	Next State	Direction
A	0	0	A	R
A	1	1	A	R
A	#	#	B	L
B	0	1	A	R
B	1	0	B	L
B	#	1	A	R

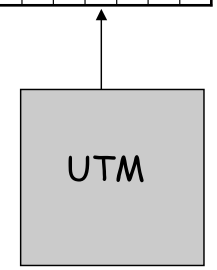
Linear: * A 0 0 A R * A 1 1 A R * A # # B L * B 0 1 A R * B 1 0 B L . . .

Universal Turing Machine



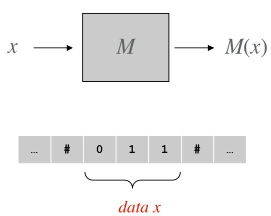
UTM Operation:

- Find state, symbol in Description
- Copy new symbol to CBI's tape
- Move ♥ L or R
- Update state, symbol
- Repeat



Universal Turing Machine (a more abstract view)

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

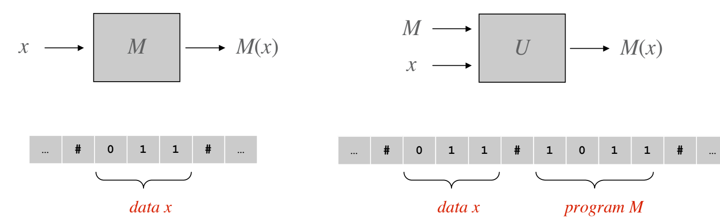


TM intuition. Software program that solves one particular problem.

Universal Turing Machine (a more abstract view)

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

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



TM intuition. Software program that solves one particular problem.
UTM intuition. Hardware platform that can implement any algorithm.

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

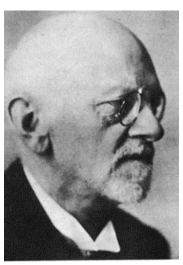
Evidence.

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

"universal"

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

7.6 Computability



Take any definite unsolved problem, such as the question as to the irrationality of the Euler-Mascheroni constant γ, 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

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.

relates to famous open math conjecture

Ex. Does `f(x)` terminate?

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

- `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 … 4 2 1
- `f(-17)`: -17 -50 -25 -74 -37 -110 -55 -164 -82 -41 -122 … -17 …

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: Preliminaries

Some programs take other programs as input

- Java compiler, e.g.

Can a program take *itself* as input ??

Why not ?

- EditDistance could take EditDistance.java as input, and compute edit distance between "DNA sequences" `public` and `class`
- GuitarHero could "play" the characters in GuitarHero.java

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 (something terribly clever) return true;
   else                           return false;
}
```

hypothetical halting function

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 (if perverse) to use for second input

```
public void strange(String f) {
   if (halt(f, f)) {
      // an infinite loop
      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.

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.

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.
- Closely related to practical problems.

No input halting problem. Give a function with no input, does it halt?

Program equivalence. Do two programs always produce the same output?

Uninitialized variables. Is variable x initialized?

Dead code elimination. Does control flow ever reach this point in a program?

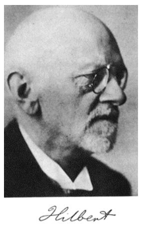
More Undecidable Problems

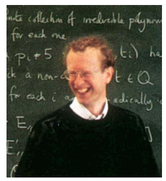
Hilbert's 10th problem.

- "*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)

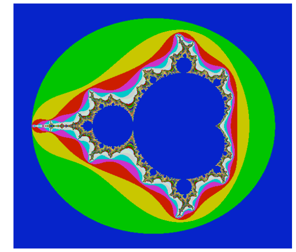




Andrew Wiles, 1995

More Undecidable Problems

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



Mandelbrot set (40 lines of code)

More Undecidable Problems

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

Context: Mathematics and Logic

Mathematics. Formal system powerful enough to express arithmetic.

Principia Mathematics
Peano arithmetic
Zermelo-Fraenkel set theory

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] 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!

Turing's Key Ideas

Turing machine.
formal model of computation

Program and data.
encode program and data as sequence of symbols

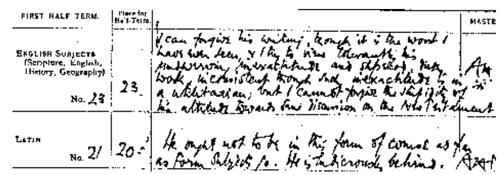
Universality.
concept of general-purpose, programmable computers

Church-Turing thesis.
computable at all == computable with a Turing machine

Computability.
inherent limits to computation

Alan Turing

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



FIRST HALF TERM.	Place for Half-Term.		MASTER
ENGLISH SUBJECTS (Scripture, English, History, Geography) No. 23	23	I can forgive his writing, though it is the worst I have ever seen, & I try to view tolerantly his unsatisfactoriness, inexactitude and slipshod, dirty work, inconsistent though such unsatisfactoriness is in a utilitarian; but I cannot forgive the stupidity of his attitude towards sane discussion on the New Testament	AH
LATIN No. 21	20	He ought not to be in this form of course as far as form subjects go. He is ludicrously behind.	AAT

Alan's report card at 14.

Alan Turing and his elder brother.