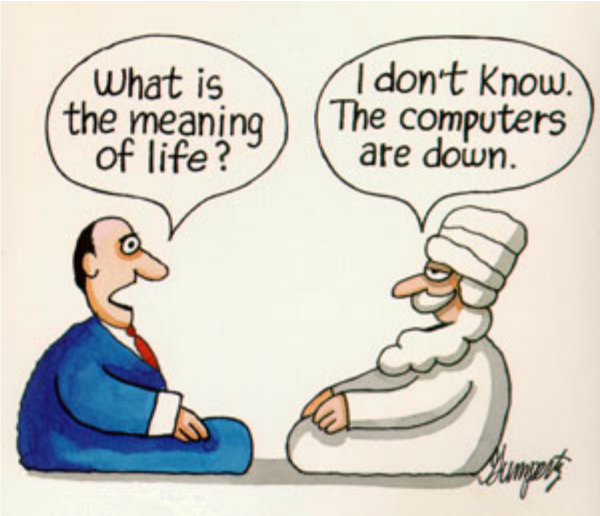
Universality and Computability



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.

David Hilbert

Kurt Gödel

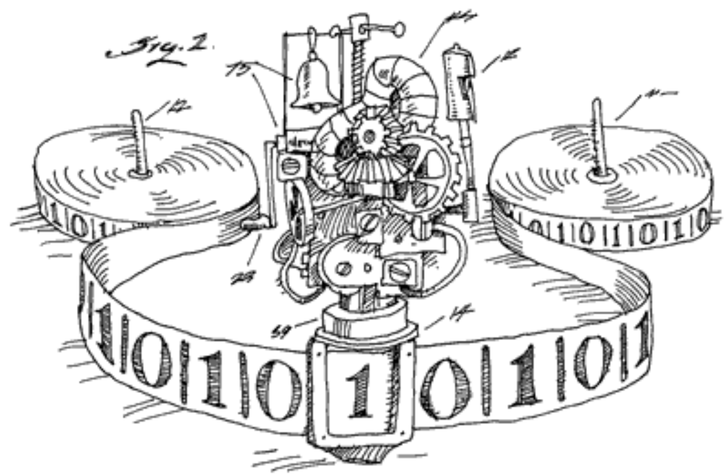
Alan Turing

Alonzo Church

John von Neumann

7.4 Turing Machines

Alan Turing (1912-1954)



Turing Machine by Tom Dunne
American Scientist, March-April 2002

Turing Machine

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

Intuition. Simulate how humans calculate.

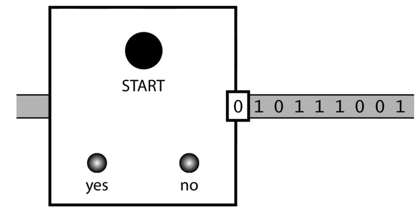
Ex. Addition.

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

Last Lecture: DFA

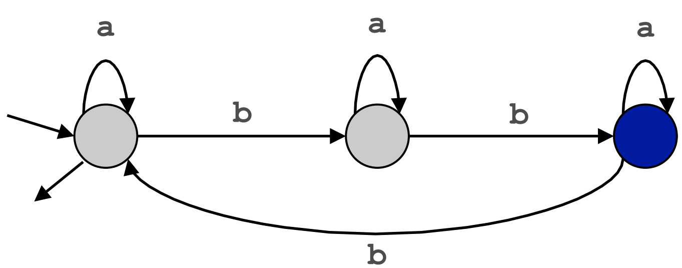
Tape. Stores input on one arbitrarily long strip, divided into cells.

- Tape head points to one cell.
- Read a symbol from tape head.
- Move tape head right one cell at a time.



State. What machine remembers.

State transition diagram. Description of what machine will do.



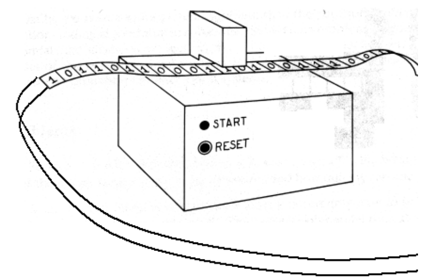
if in this state and input symbol is **b***:*

- *move to leftmost state*
- *move tape head right*

This Lecture: Turing Machine

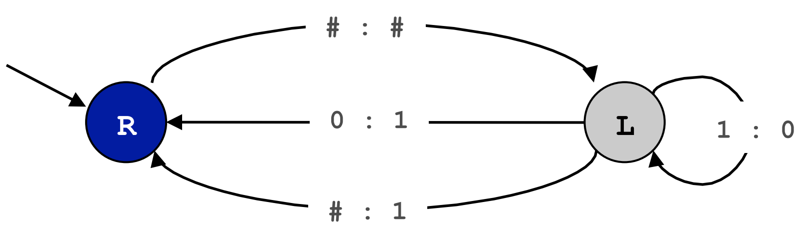
Tape. Stores input, output, and intermediate results.

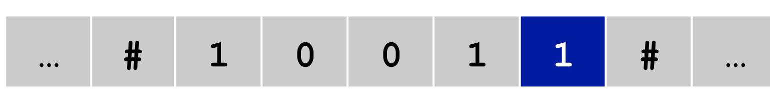
- Tape head points to one cell of tape.
- Read a symbol from cell and write a symbol to cell.
- Move tape head left or right one cell at a time.



State. What machine remembers.

State transition diagram. Description of what machine will do.





Turing Machine: Execution

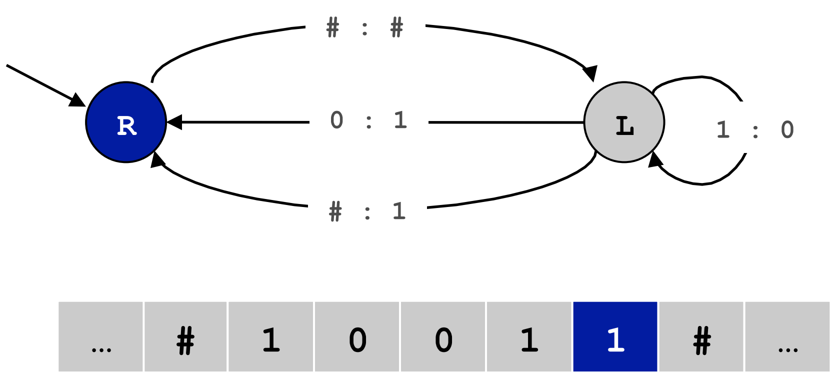
Simple machine with N states.

- Begin in start state.
- Stop upon reaching a **yes**, **no**, or **halt** state.

infinite loop possible!

Repeat the following:

- Read symbol from tape.
- Depending on current state and tape symbol,
  - move to new state
  - write a symbol on tape
- Move tape head left or right, depending on label of new state.



if in this state and input symbol is **0** *or* **1***:*

- *don't write anything*
- *stay in same state*
- *move tape head right*

Turing Machine: Execution

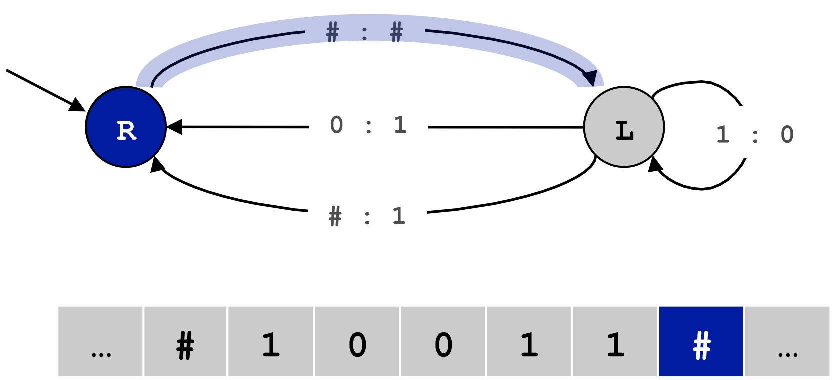
Simple machine with N states.

- Begin in start state.
- Stop upon reaching a **`yes`**, **`no`**, or **`halt`** state.

infinite loop possible!

Repeat the following:

- Read symbol from tape.
- Depending on current state and tape symbol,
  - move to new state
  - write a symbol on tape
- Move tape head left or right, depending on label of new state.



if in this state and input symbol is **#**:

- *write a* **#**
- *move to other state*
- *move tape head left*

Turing Machine: Execution

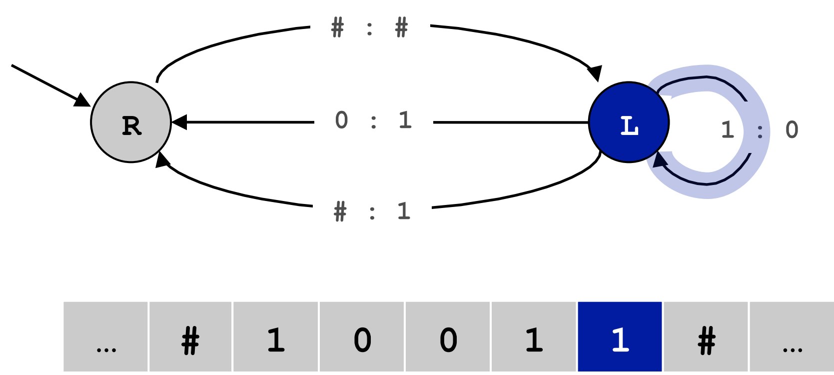
Simple machine with N states.

- Begin in start state.
- Stop upon reaching a **yes**, **no**, or **halt** state.

infinite loop possible!

Repeat the following:

- Read symbol from tape.
- Depending on current state and tape symbol,
  - move to new state
  - write a symbol on tape
- Move tape head left or right, depending on label of new state.



if in this state and input symbol is **1**:
- *write a* **0**
- *stay in same state*
- *move tape head left*

Turing Machine: Execution

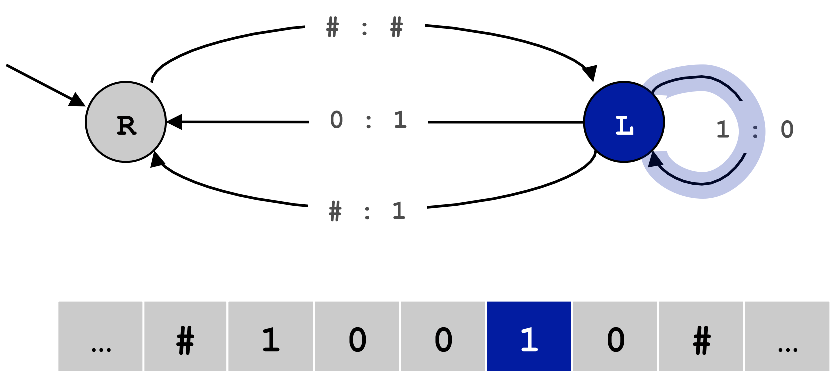
Simple machine with N states.

- Begin in start state.
- Stop upon reaching a **yes**, **no**, or **halt** state.

infinite loop possible!

Repeat the following:

- Read symbol from tape.
- Depending on current state and tape symbol,
  - move to new state
  - write a symbol on tape
- Move tape head left or right, depending on label of new state.



if in this state and input symbol is **1***:*

- *write a* **0**
- *stay in same state*
- *move tape head left*

Turing Machine: Execution

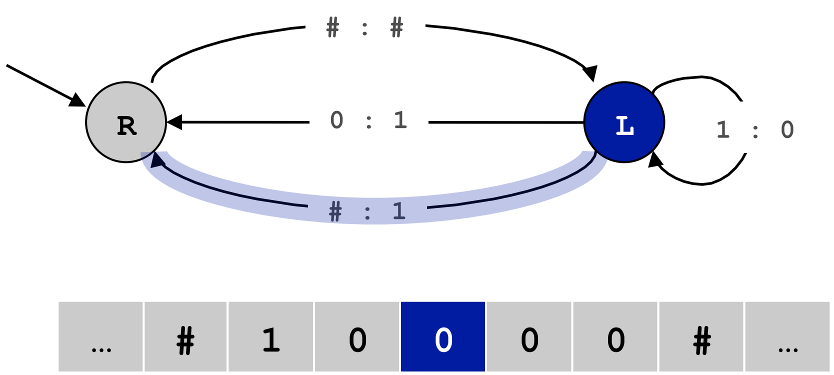
Simple machine with N states.

- Begin in start state.
- Stop upon reaching a **yes**, **no**, or **halt** state.

infinite loop possible!

Repeat the following:

- Read symbol from tape.
- Depending on current state and tape symbol,
  - move to new state
  - write a symbol on tape
- Move tape head left or right, depending on label of new state.



if in this state and input symbol is **0***:*

- *write a* **1**
- *move to other state*
- *move tape head right*

Turing Machine: Execution

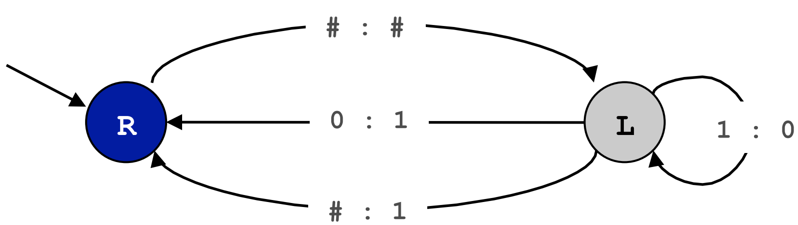
Simple machine with N states.

- Begin in start state.
- Stop upon reaching a **`yes`**, **`no`**, or **`halt`** state.

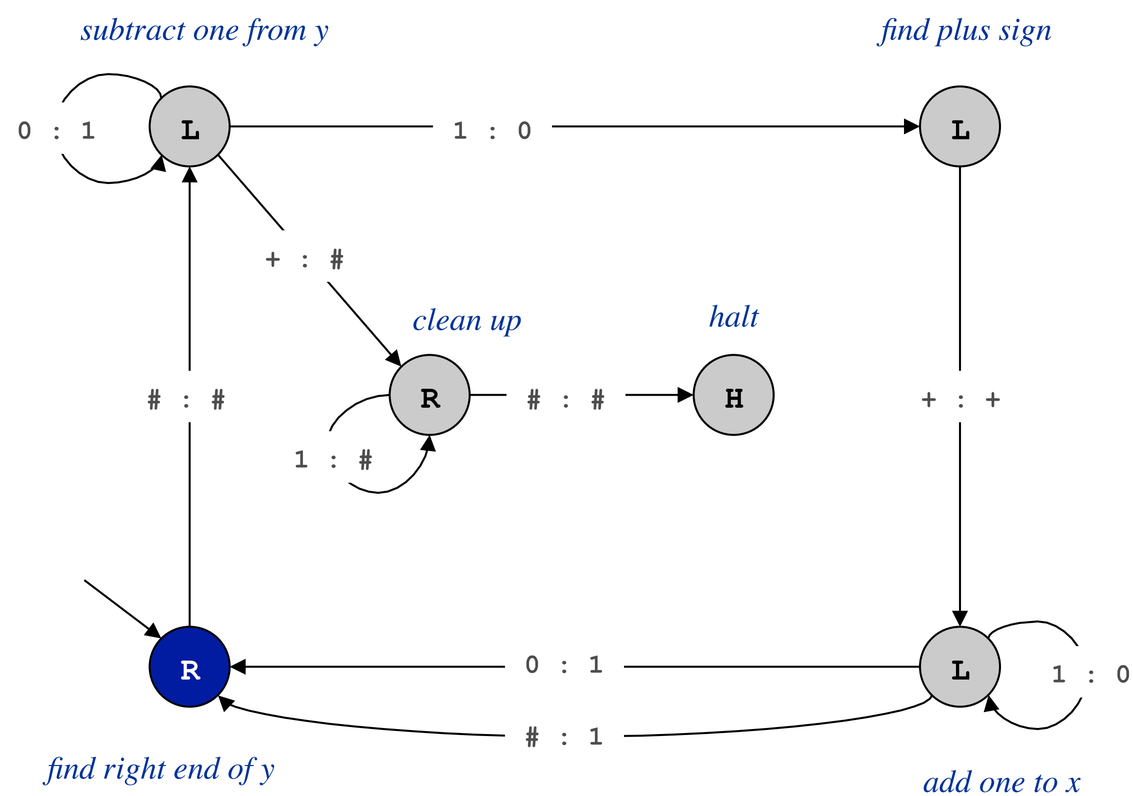
infinite loop possible!

Repeat the following:

- Read symbol from tape.
- Depending on current state and tape symbol,
  - move to new state
  - write a symbol on tape
- Move tape head left or right, depending on label of new state.



Binary Adder

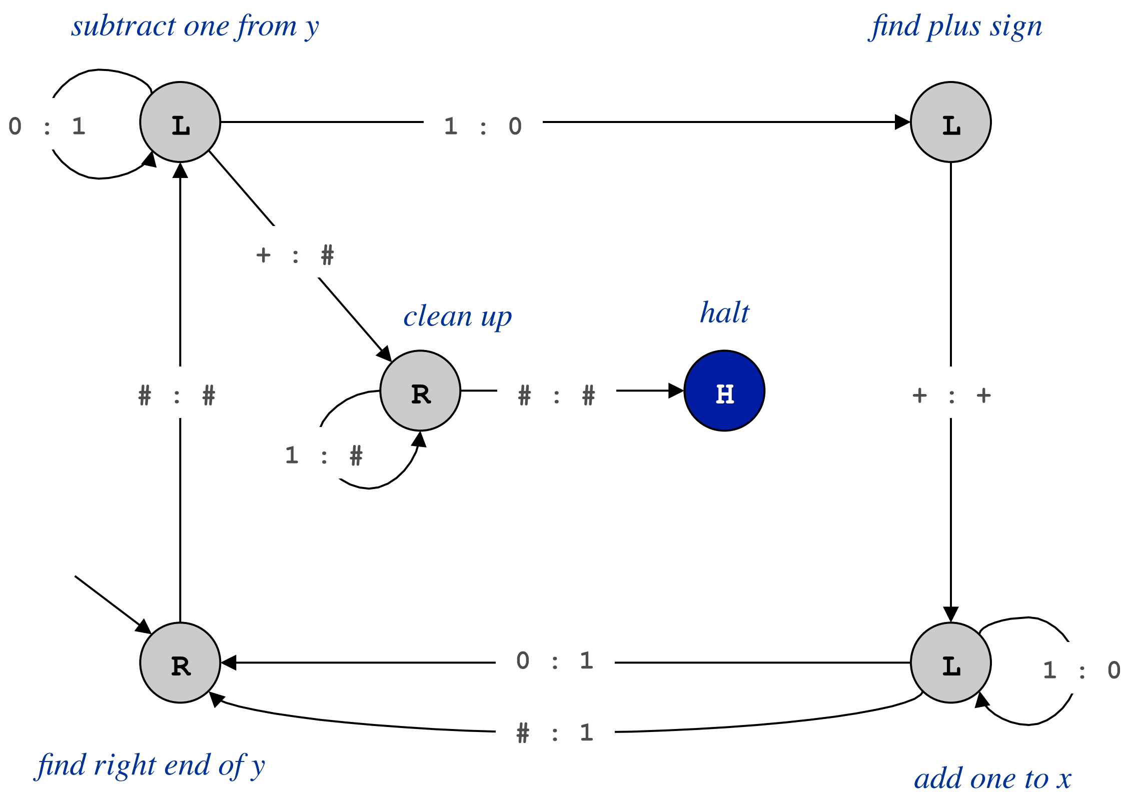


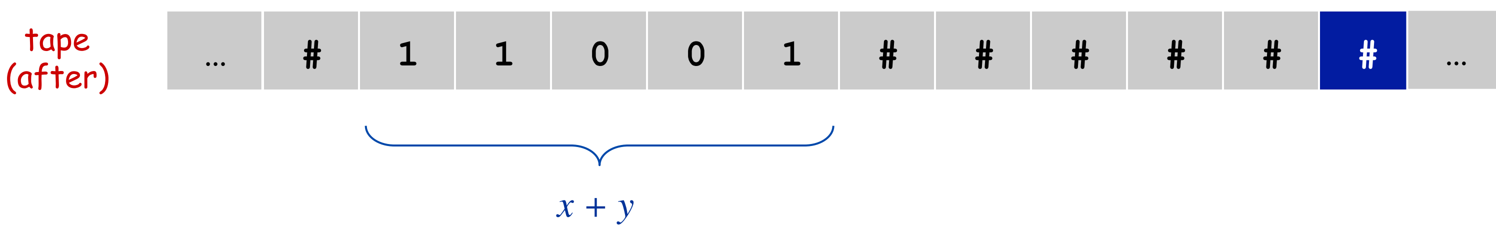
tape (before)

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

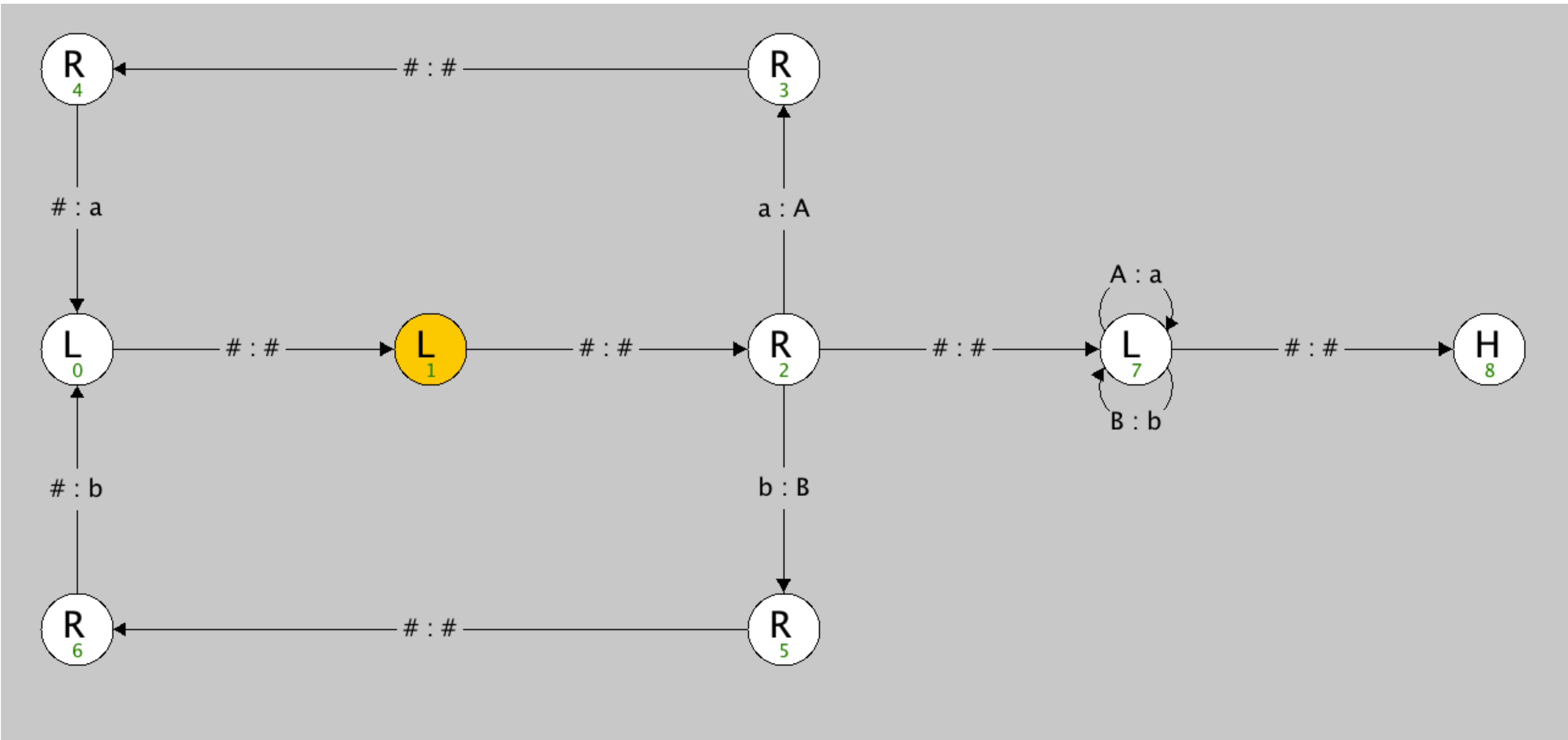
x   y

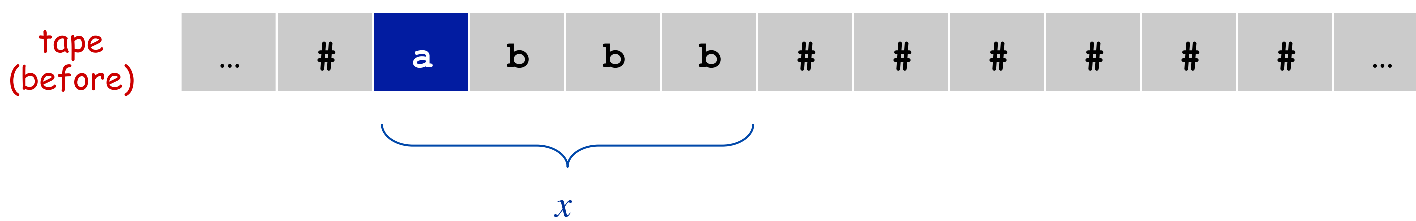
Binary Adder



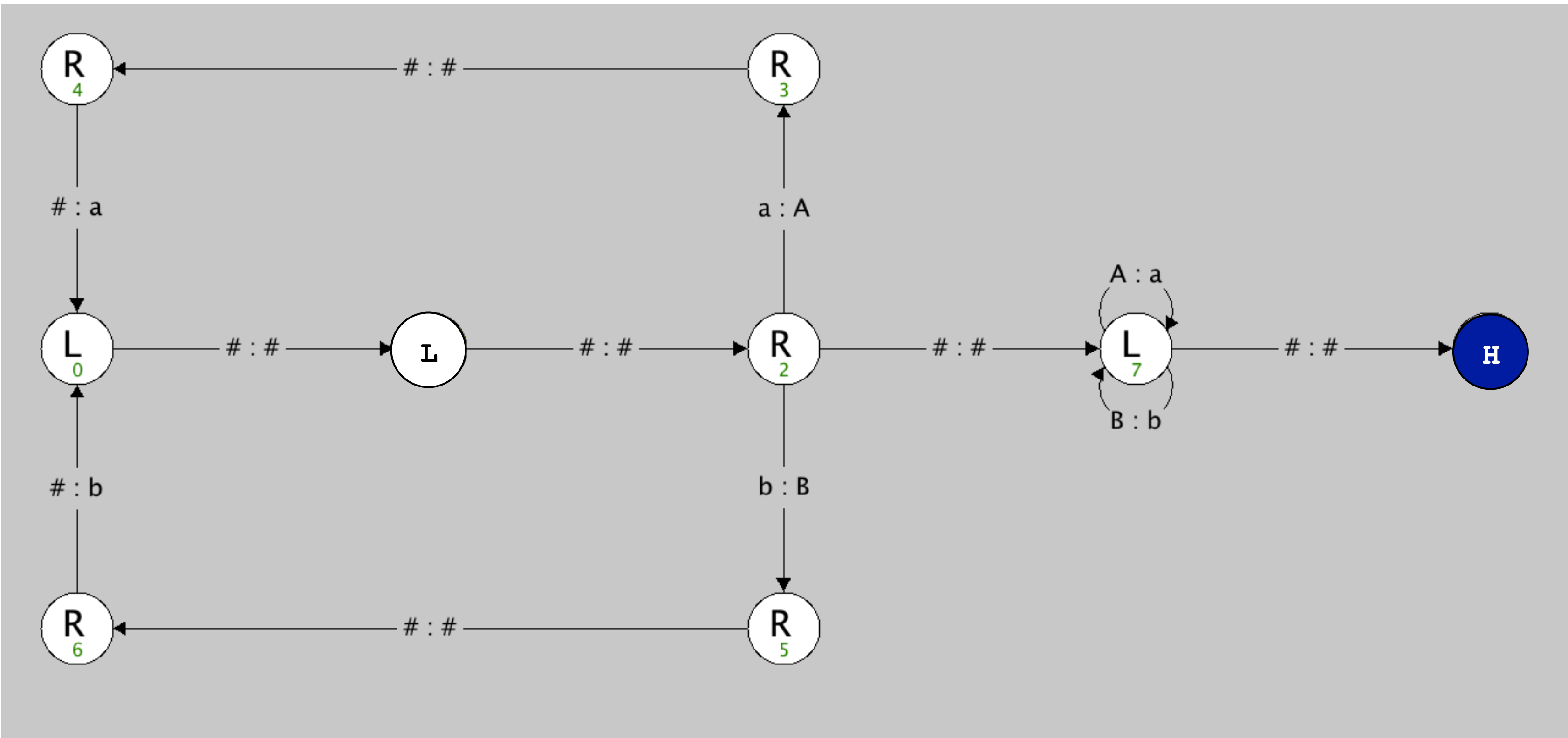


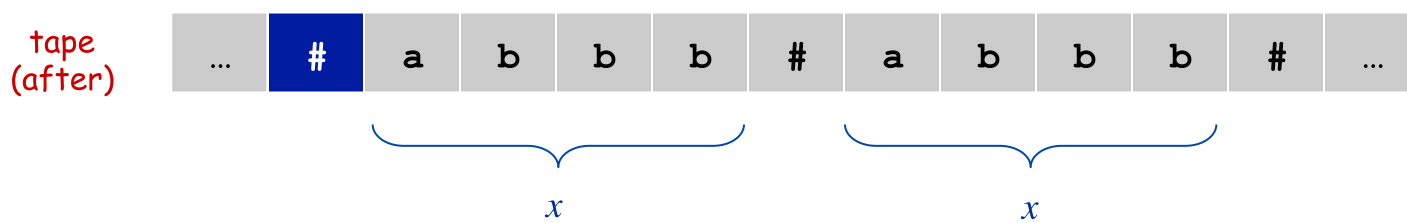
Copy





Copy





Program and Data

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.

Java

machine language

your program

```
public class HelloWorld {
   public static void main(String[] args) {
      System.out.println("Hello, World");
   }
}
```

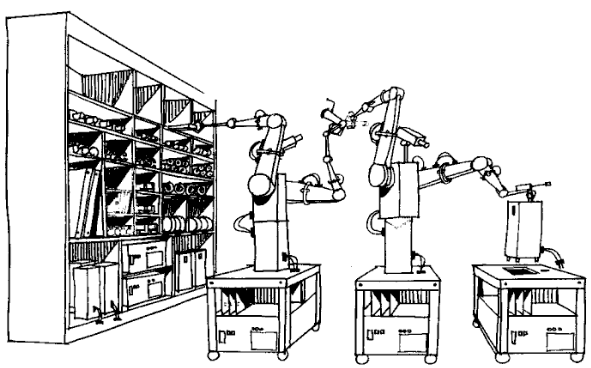
is data to a compiler

Program and Data

Data. Sequence of symbols (interpreted one way).
Program. Sequence of symbols (interpreted another way).

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

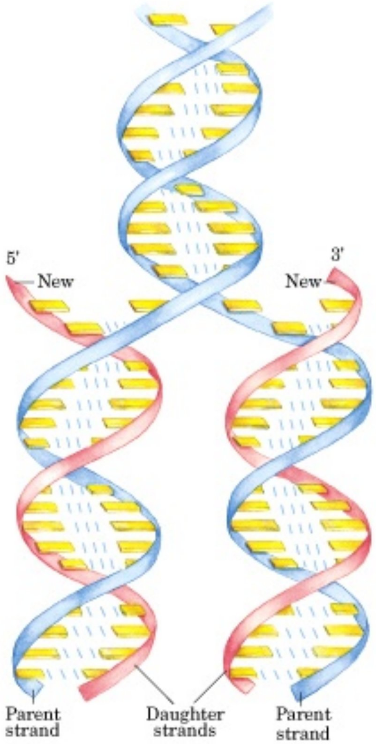
> Print the following statement twice, the second time in quotes.
> "Print the following statement twice, the second time in quotes."



Program and Data

Data. Sequence of symbols (interpreted one way).
Program. Sequence of symbols (interpreted another way).

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

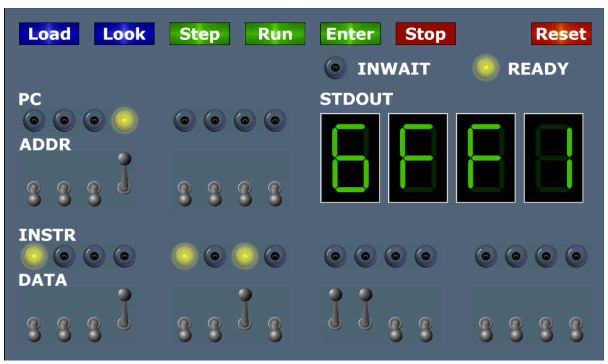


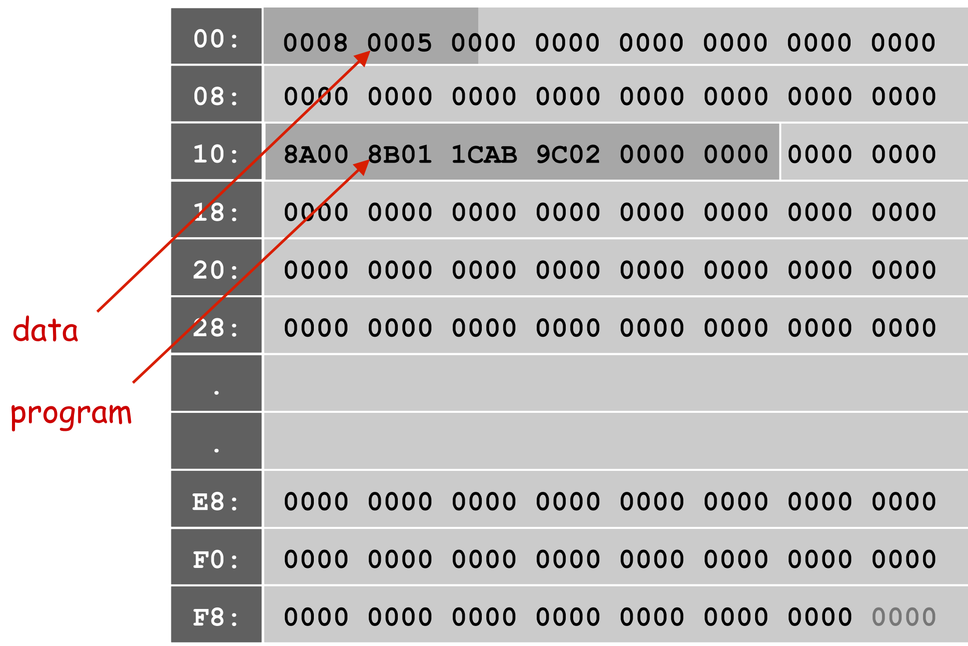
self-replicating DNA

Program and Data

Data. Sequence of symbols (interpreted one way).
Program. Sequence of symbols (interpreted another way).

Ex 4. TOY / von Neumann architecture.

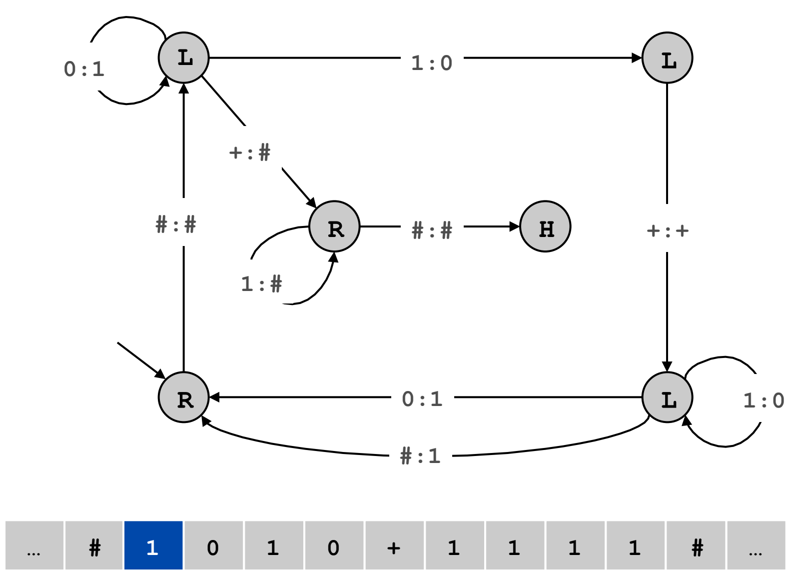




Program and Data

Data. Sequence of symbols (interpreted one way).
Program. Sequence of symbols (interpreted another way).

Ex 5. Turing machine.



graphical representation

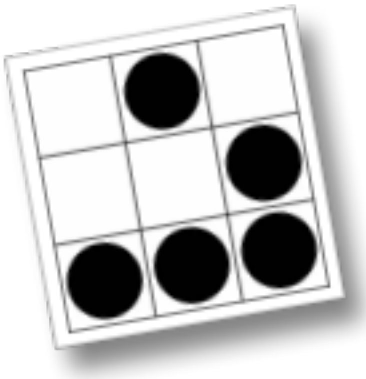
```
% more adder.tur
vertices
2 R
0 L
1 L
3 L
4 R
5 H

edges
0 0 0 1
0 1 1 0
0 4 + #
1 3 + +
2 0 # #
3 2 # 1
3 2 0 1
3 3 1 0
4 4 1 #
4 5 # #

tape
[1] 0 1 0 + 1 1 1 1
```

program

data

text representation

7.5 Universality



Universal Machines and Technologies

Dell PC

iMac

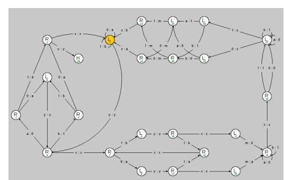
Diebold voting machine

iPod

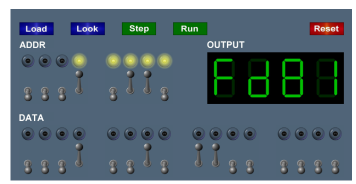
Printer

Xbox

Tivo

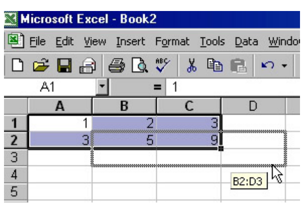


Turing machine



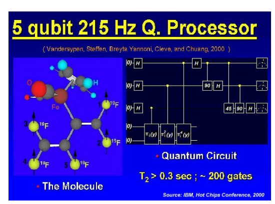
TOY

Java language

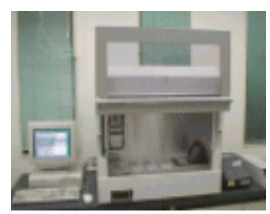


MS Excel

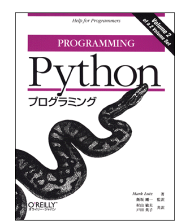
Blackberry



Quantum computer



DNA computer

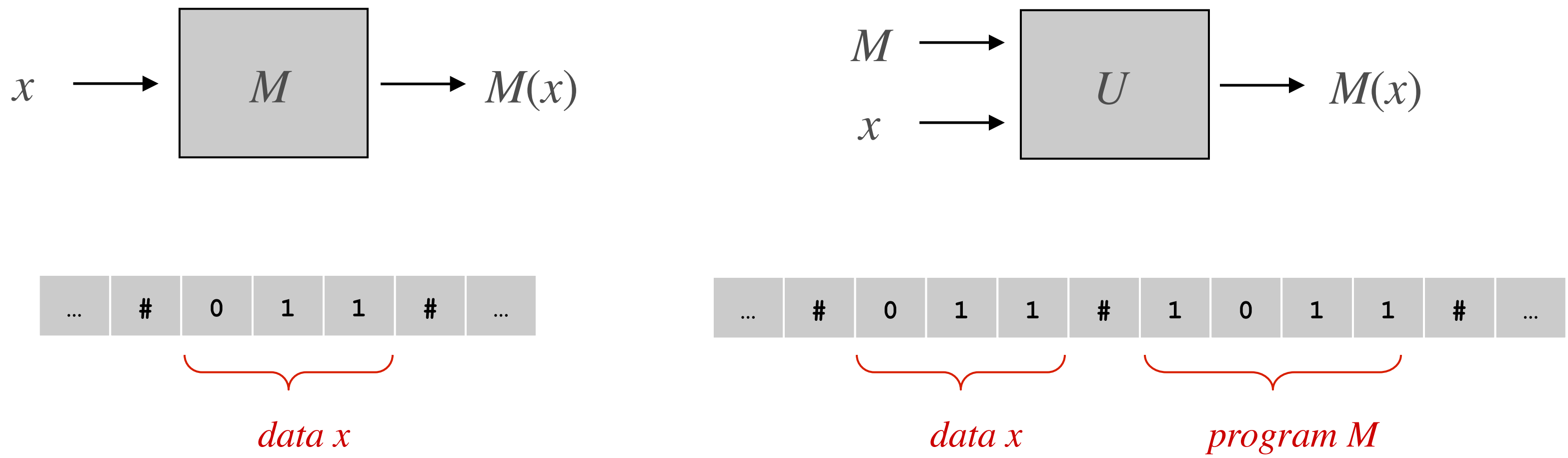


Python language

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 $M(x)$.



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

Universal Turing Machine

Your laptop (a UTM) can perform any computational task.

- Java programming.
- Pictures, music, movies, games.
- Email, browsing, downloading files, telephony.
- Word-processing, finance, scientific computing.
- …

even tasks not yet contemplated when laptop was purchased

" 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.
- Turing machine simulator in Java.
- TOY simulator on a Turing machine.
- ...

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

Church-Turing Thesis: Evidence

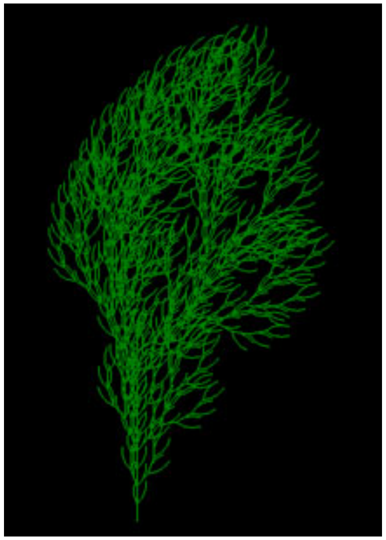
Evidence.

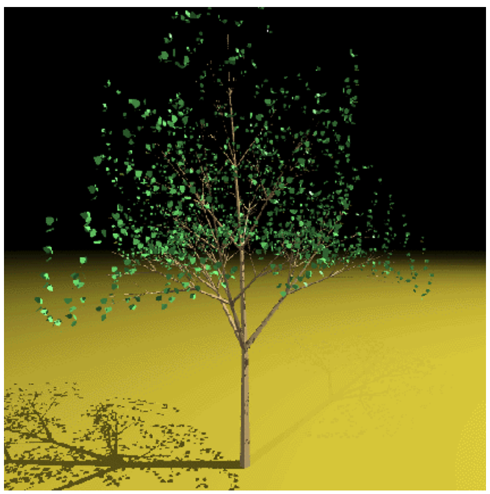
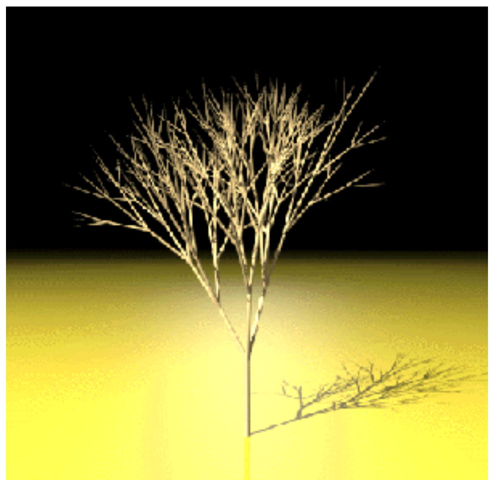
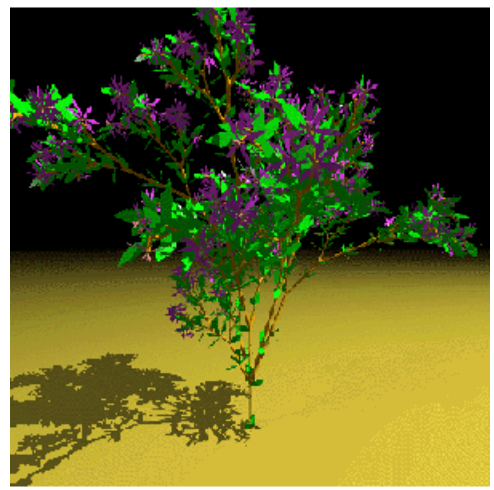
- 7 decades without a counterexample.
- Many, many models of computation that turned out to be equivalent.

"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
quantum computer	compute using superposition of quantum states
DNA computer	compute using biological operations on DNA
human brain †	???

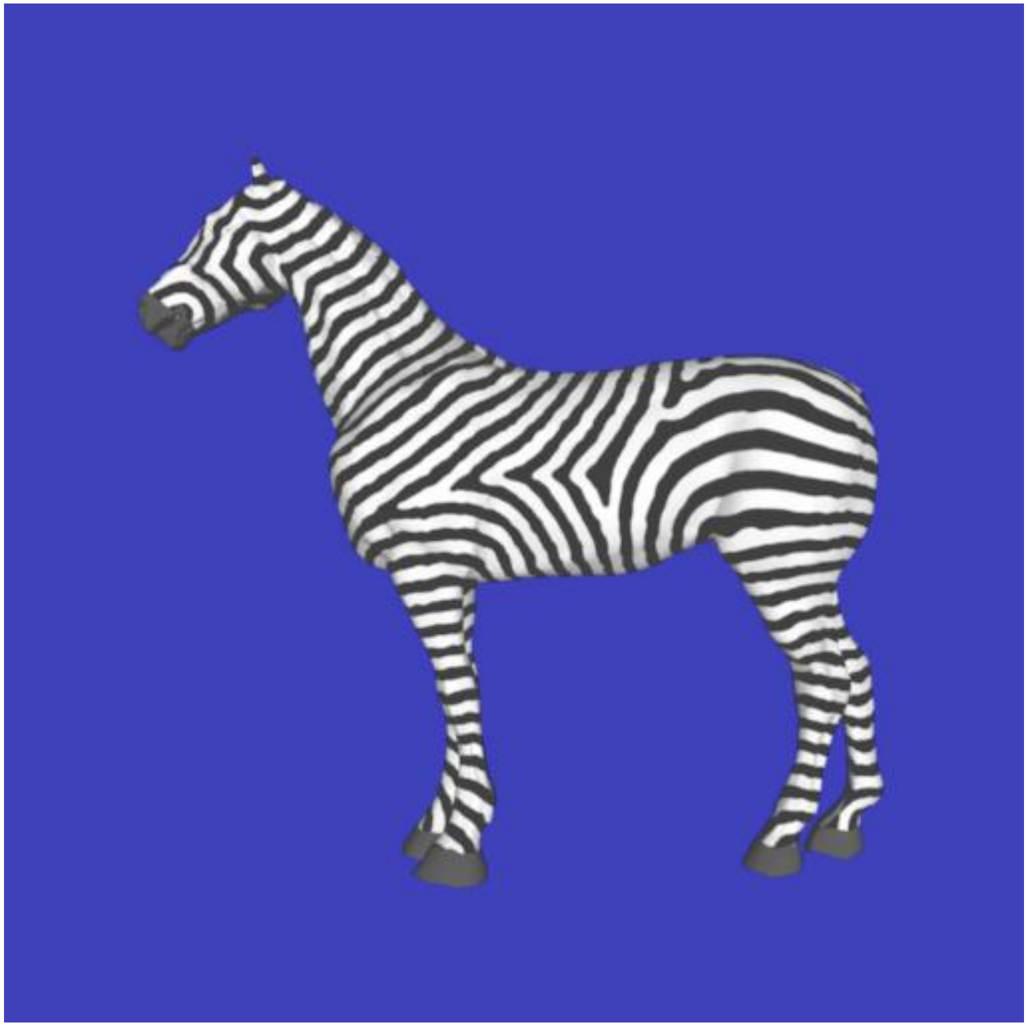
Lindenmayer Systems: Synthetic Plants

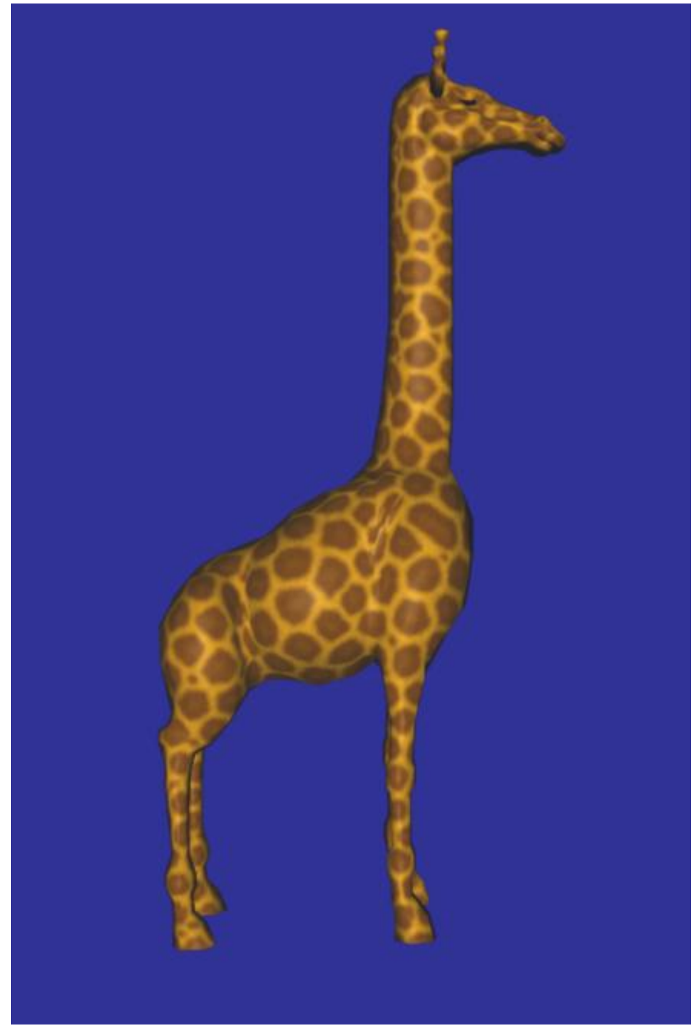


http://astronomy.swin.edu.au/~pbourke/modelling/plants

Cellular Automata: Synthetic Zoo

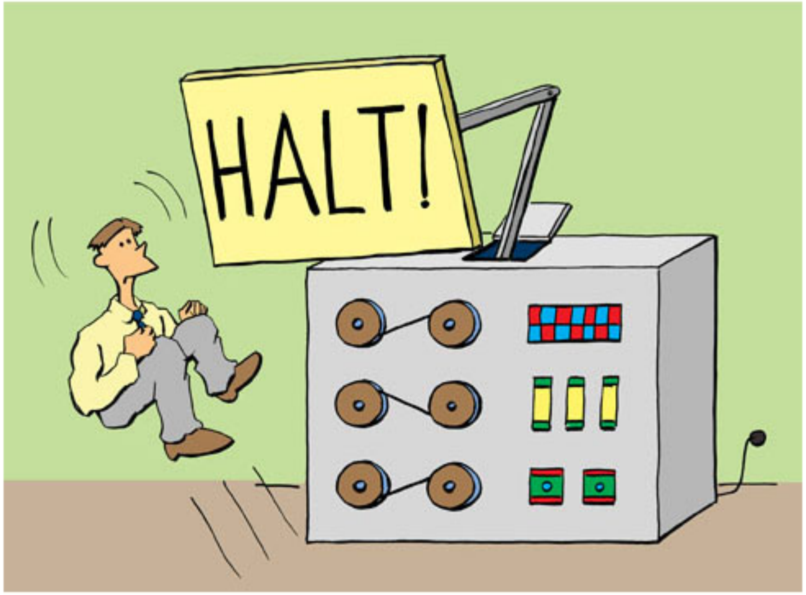




Reference: *Generating textures on arbitrary surfaces using reaction-diffusion* by Greg Turk, SIGGRAPH, 1991.

History: *The chemical basis of morphogenesis* by Alan Turing, 1952.

7.6 Computability

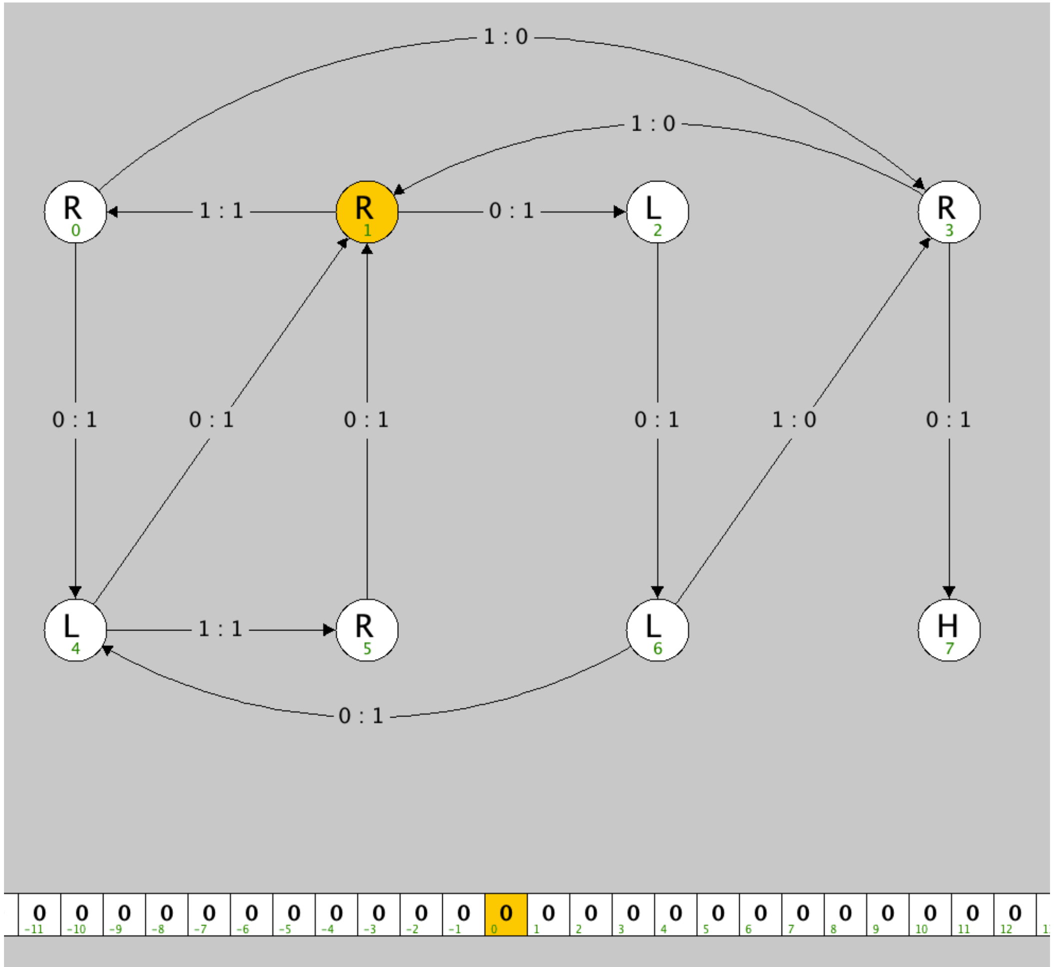


Alan designed the perfect computer

http://www.coopertoons.com/education/haltingproblem/haltingproblem.html

Halting Problem

Halting problem. Write a Turing machine that reads a Turing machine and its input, and decides whether it results in an infinite loop.



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.

Collatz sequence 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 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.

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

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 ?

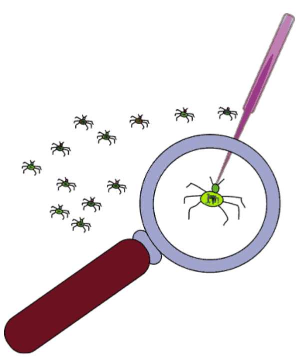
Totality problem. Give a function f, does it halt on every input x ?

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

Program equivalence. Do functions f and g and always return same value?

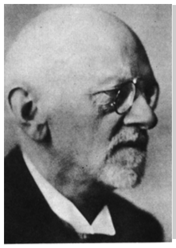
Uninitialized variables. Is the variable x initialized before it's used?

Dead-code elimination. Does this statement ever get executed?



More Undecidable Problems

Hilbert's 10th problem.



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

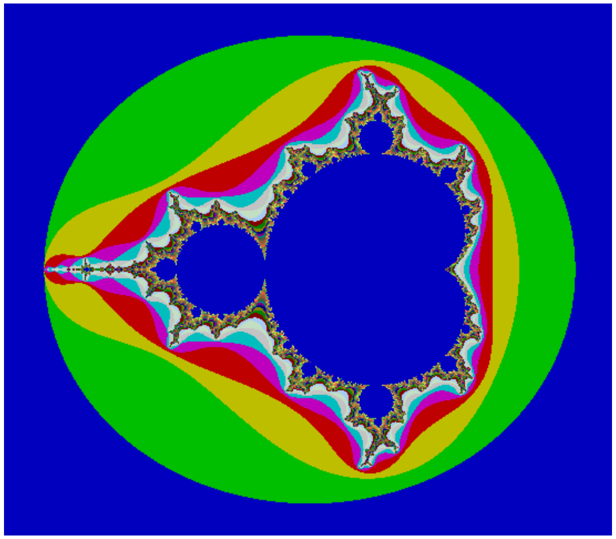
- $f(x, y, z) = 6x^3 y z^2 + 3xy^2 - x^3 - 10.$ yes : $f(5, 3, 0) = 0.$
- $f(x, y) = x^2 + y^2 - 3.$ no.

Definite integration. Given a rational function $f(x)$ composed of polynomial and trig functions, does $\int_{-\infty}^{+\infty} f(x)\,dx$ exist?

- $g(x) = \cos x \, (1 + x^2)^{-1}$ yes, $\int_{-\infty}^{+\infty} g(x)\,dx = \pi / e.$
- $h(x) = \cos x \, (1 - x^2)^{-1}$ 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.



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

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.

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.

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

Alan Turing (left)
Elder brother (right)