

## 7.1 The Origin of Computer Science

Alan Mathison Turing (1912–1954)

“On Computable Numbers, with an Application to the Entscheidungsproblem” 1936



- 1936: Founded computer science as graduate student
- 1938–1945: WW II hero — breaking of German codes
- 1948: Almost qualified for UK Olympic team as marathon runner
- 1950: Seminal paper on AI — proposed Turing test
- 1952: Prosecuted for homosexuality, chose chemical castration over prison
- 1954: Suicide
- 1966: Turing Award introduced as highest prize in computer science
- 2009: Public apology by British government
- 2013: Pardon by Queen of England

[Dates from [http://en.wikipedia.org/wiki/Alan\\_Turing](http://en.wikipedia.org/wiki/Alan_Turing)]

**Q:** What Problem Was Turing Trying to Solve?

David Hilbert

“Mathematical Problems” 1900



Can mathematics be fully axiomatized and automated?

Kurt Gödel

“On Formally Undecidable Propositions . . .” 1931



Library of Congress

Mathematics cannot be fully axiomatized: some true statements will be unprovable.

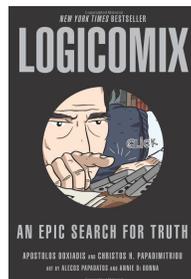
Alonzo Church

“An Unsolvable Problem of Elementary Number Theory” 1936

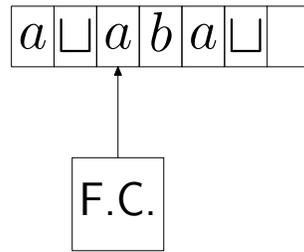


Independently of Turing: mathematics cannot be automated — cannot algorithmically distinguish provable from disprovable statements.

The Cliff’s Notes Version of History



## 7.2 The Basic Turing Machine



- Head can both read and write, and move in both directions
- Tape has unbounded length
- $\square$  is the blank symbol. All but a finite number of tape squares are blank.
- F.C. = finite-state control.

**Def:** A (deterministic) Turing Machine (TM) is a 6-tuple  $(Q, \Sigma, \Gamma, \delta, q_0, q_{halt})$ , where:

- $Q$  is a finite set of states, containing a *start state*  $q_0$  and a *halt state*  $q_{halt}$ .
- $\Sigma$  is the input alphabet
- $\Gamma$  is the tape alphabet, containing  $\Sigma$  and  $\square \in \Gamma - \Sigma$ .
- $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$  is the transition function

### Interpretation:

- $\delta(q, \sigma) = (q', \sigma', R)$ 
  - Rewrite  $\sigma$  as  $\sigma'$  in current cell
  - Switch from state  $q$  to state  $q'$
  - And move right
- $\delta(q, \sigma) = (q', \sigma', L)$ 
  - Same, but move left
  - *Unless* at left end of tape, in which case stay put



## 7.4 Formalizing Computation of TMs

- A configuration of a TM  $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{halt})$  is denoted  $uqv$ , where  $q \in Q$ ,  $u, v \in \Gamma^*$ .
  - Tape contents =  $uv$  followed by all blanks
  - State =  $q$
  - Head on first symbol of  $v$ .
  - *Equivalent* to  $uqv'$ , where  $v' = v\sqcup$ .
- Start configuration =  $q_0w$ , where  $w$  is input.
- One step of computation (denoted  $C \Rightarrow_M C'$ ): for  $C = uq\sigma v$  with  $q \in Q \setminus \{q_{halt}\}$ ,  $u, v \in \Gamma^*$ ,  $\sigma \in \Gamma$ ,
  - If  $\delta(q, \sigma) = (q', \sigma', R)$ , then  $C' = u\sigma'q'v$ .
  - If  $\delta(q, \sigma) = (q', \sigma', L)$ , and  $u = u'\tau$  for  $u' \in \Gamma^*$  and  $\tau \in \Gamma$ , then  $C' = u'q'\tau\sigma'v$ .
  - If  $\delta(q, \sigma) = (q', \sigma', L)$ , and  $u = \epsilon$ , then  $C' = q'\sigma'v$ .
- If  $q = q_{halt}$ , computation halts ( $C' = C$ ) and the *output* is the contents of the tape to the left of the first blank symbol.

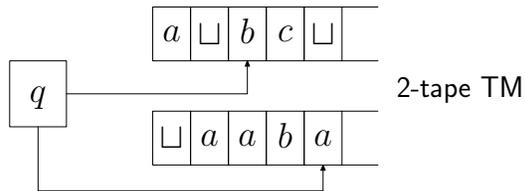
**Def:** TM  $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{halt})$  solves computational problem  $f : \Sigma^* \Rightarrow 2^{(\Gamma \setminus \{\sqcup\})^*}$  if for every  $w \in \Sigma^*$ , there is a sequence  $C_0, \dots, C_t$  of configurations of  $M$  such that:

1.  $C_0 = q_0w$
2.  $C_{i-1} \Rightarrow_M C_i$  for  $i = 1, \dots, t$
3. In  $C_t$ ,  $M$  is state  $q_{halt}$  and the contents of the tape to the left of the first blank symbol is an element of  $f(w)$ .

**Running time:** Defined analogously to Word-RAM, e.g.  $T(n) =$  maximum number of steps taken by  $M$  on all inputs of length  $n$  (cf.  $T(n, k)$  for Word-RAM.) algorithms.

## 7.5 Multitape TMs

There are a number of seemingly arbitrary choices in the definition of a TM (albeit less so than in the Word-RAM). We want to argue that these choices don't affect the power of the model. For example, what about multiple tapes?

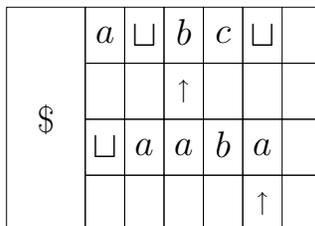


Convention: First tape used for I/O, like standard TM; Other tapes available for scratch work.

Formally, a  $k$ -tape TM has a transition function  $\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R\}^k$ .

### Simulation of multiple tapes by one

- Simulate a  $k$ -tape TM by a one-tape TM whose tape is split (conceptually) into  $2k$  tracks:
  - $k$  tracks for tape symbols
  - $k$  tracks for head position markers (one in each track)



- To simulate one move of the  $k$ -tape TM:
  - Start with the head on the left endmarker
  - Scan down the tape, remembering in the finite control the symbols “scanned” by the  $k$  heads
  - Scan back up the tape, revising each track in the vicinity of its head marker
  - Return the head to the left endmarker
- Also need transitions to reformat input and output at start and end of computation.

**Speed of the Simulation.** Note that the “equivalence” in ability to solve computational problems does not mean comparable speed.

e.g. A standard TM can recognize palindromes in time  $O(n^2)$  (as we saw), and it is known that this is the best possible. But there is an  $O(n)$ -time 2-tape TM for the same problem:

And this is tight — the simulation above gives at most a quadratic slowdown:

**Theorem 7.1** *If  $M$  is a multitape TM, then there is a constant  $c$  and 1-tape TM  $M'$  such that on every input  $w \in \Sigma^*$ ,  $M'$  produces the same output as  $M$  (or runs forever if  $M$  does), and if  $M$  halts in time  $T(w) \geq |w|$ , then  $M'$  halts in time at most  $c \cdot T(w)^2$ .*

## 7.6 RAMs vs. TMs

Now we show that, even though TMs may seem much simpler and weaker than Word-RAMs, they are actually equivalent in power.

**Theorem 7.2** *For every Word-RAM program  $P$ , there is a multitape TM  $M$  such that for every input  $x = (x_1, \dots, x_n) \in \mathbf{N}^*$ , it holds that  $M(\langle x \rangle)$  has the “same behavior” as  $P$  on input  $x$  and size parameter  $k = \max\{x_1, \dots, x_n, m\}$  where  $m$  is the largest constant occurring in  $P$ .*

- $\langle x \rangle$  denotes any reasonable encoding of  $k, x$  over the input alphabet  $\Sigma$  of  $M$ . For example, we can take  $\Sigma = \{0, 1, \|\}$  and  $\langle x \rangle = \langle x_1 \rangle \| \dots \| \langle x_n \rangle$ , where  $\langle y \rangle$  denotes the binary representation of  $y \in \mathbf{N}$ .
- If  $P$  halts with output  $y \in \mathbf{N}^*$ , then  $M$  will halt with output  $\langle y \rangle$ .
- If  $P$  does not halt, then  $M$  will also run forever.
- **Efficiency:** If  $P$  runs in time  $T = T(k, x) \geq |x|$ , then  $M$  will run in time at most  $T^2 \cdot \text{poly}(\log T, \log k)$ .

### Simulating a Word RAM Program by a Multitape TM

- Tape 1:  $M[0] \| M[1] \| \dots \| M[S-1]$  (sequence of  $w$ -bit binary strings).
- Tape 2:  $S$  (in binary)
- Tape 3:  $w$  (in unary — sequence of  $w$  ones)
- Tape 4:  $i \in \{0, \dots, S-1\}$  such that TM head on tape 1 is within  $M[i]$  (in binary)
- Tape 5:  $j \in \{0, \dots, w-1\}$  such that TM head on tape 1 is at the  $j$ 'th bit of  $M[i]$  (in unary)
- Tapes 6–(r+5): Registers  $R[0], \dots, R[r-1]$  (in binary)
- Tape 7: scratch tape
- Keep track of program counter  $\ell$  in state of  $M$

The TM can carry out each of the following instructions in time  $\text{poly}(w)$  where  $w$  is the current word size:

- $P_\ell = \text{“}R[i] \leftarrow R[j] \text{ op } R[k]\text{”}$  for each of the word operations  $\text{op}$  we allow.
- If  $P_\ell = \text{“IF } R[i] = 0, \text{ GOTO } \ell\text{”}$
- HALT

Each of the following can be implemented in time  $O(S \cdot w)$ , where  $S$  is the current space bound:

- $R[i] \leftarrow M[R[j]]$
- $M[R[i]] \leftarrow R[j]$
- MALLOC

Throughout the computation  $S \leq T$  (here we use  $T \geq n$ ) and  $w \leq \lceil \log \max\{k+1, S\} \rceil \leq \log \max\{k+1, T\}$ . ■

## 7.7 The Church–Turing Thesis

Many other models of computation are equivalent in power to Turing Machines:

- TMs with 2-dimensional tapes
- Word RAMs
- Integer RAMs
- General Grammars
- 2-counter machines
- Church’s  $\lambda$ -calculus ( $\mu$ -recursive functions)
- Markov algorithms
- Your favorite programming language (C, Python, OCaml, ...)

The equivalence of each to the others is a mathematical theorem. That these formal models of algorithms capture our intuitive notion of algorithms is the **Church–Turing Thesis**. (Church’s thesis = partial recursive functions, Turing’s thesis = TMs.) The Church–Turing Thesis is an extramathematical proposition, not subject to formal proof.

**The Extended Church-Turing Thesis:** Every “reasonable” model of computation can be simulated on a Turing machine with only a polynomial slowdown.

Counterexamples?

- Integer RAMs.
- Randomized computation.
- Parallel computation.
- Analog computers.
- DNA computers.
- Quantum computers.

→ Extended C-T Thesis needs to be qualified with “sequential and deterministic” and “bounded work per step”.

## 7.8 Languages and Complexity Classes

In classifying computational problems as solvable vs. efficiently solvable vs. unsolvable, it is convenient to restriction attention to *decision problems* — ones where the answer is YES or NO. These correspond to computing functions  $f : \Sigma^* \rightarrow \{0, 1\}$  (where 1=YES), or equivalently to deciding whether the input string is a particular *language*  $L \subseteq \Sigma^*$  (namely, the set of strings where the answer is YES).

Motivations for focus on decision problems:

- Don't need to worry about intractability because of  $|f(x)|$  being long.
- Many computational problems of interest can be reformulated as an essentially equivalent decision problem.

### Some Languages:

- PALINDROMES =  $\{x \in \Sigma^* : x = x^R\}$ .
- PRIMES =  $\{\langle N \rangle : N \text{ is a prime number}\}$ .
- MINIMUMSPANNINGTREE =  $\{\langle G, w \rangle : G \text{ a weighted graph with a minimum spanning tree of weight at most } w\}$ .
- RANK =  $\{\langle x_1, \dots, x_n, \ell \rangle : x_1 \text{ is among the smallest } \ell \text{ items in } x_1, \dots, x_n\}$ .
- $\langle \cdot \rangle$  = any “reasonable” encoding as a string. Which choices of encoding can make the difference between a computational problem being solvable or not? What about solvable in polynomial time?
  - Graph as adjacency matrix vs. list of edges?
  - Numbers in binary vs. base 10?
  - Numbers in binary vs. unary?

### Solvable Problems:

- A function  $f : \Sigma^* \rightarrow \Sigma^*$  is called *recursive* or *computable* if there is an algorithm that computes  $f$ .
- A language  $L \subseteq \Sigma^*$  is called *recursive* or *decidable* if the characteristic function of  $L$  is recursive. An algorithm that computes the characteristic function of  $L$  is also said to *decide* (membership in) the language  $L$ .
- $R = \{L : L \text{ is recursive}\}$ .

- Does not depend on the choice of computational model (TMs vs. Word RAMs vs. Lambda Calculus)!

### Efficiently Solvable Problems:

- $L \in \text{TIME}(T(n))$  if there is an algorithm deciding  $L$  that runs in time  $O(T(n))$ .
- Depends on the model of computation!
- So we should really write  $\text{TIME}_{\text{RAM}}(T(n))$ ,  $\text{TIME}_{\text{TM}}(T(n))$ , etc.  $\text{TIME}_{\text{TM}}(\cdot)$  conventionally refers to *multi-tape* TMs.

### Polynomial Time:

- $P = \bigcup_c \text{TIME}_{\text{TM}}(n^c) = \bigcup_c \text{TIME}_{\text{RAM}}(n^c)$
- Same for all “reasonable” models of computation and “reasonable” encodings of inputs.
- Coarse approximation to “efficiently solvable.”

We have  $\text{CF} \subseteq P \subseteq R \subseteq 2^{\Sigma^*}$ , where CF is the class of context-free languages (i.e. languages that can be generated by grammars as in the PROP programming problem from HW3).

### Questions:

- Are there non-recursive languages?
- Are there recursive languages not in P?
- Are there “natural” languages not in R? In  $R \setminus P$ ?
- Is FACTORING in P?
- Is  $\text{MST} \in \text{TIME}_{\text{RAM}}(O(n))$ ? Is  $\text{MST} \in \text{TIME}_{\text{TM}}(O(n))$ ?

- Can MULTIPLICATION be done in time  $O(n)$  on a Word-RAM? On a multitape TM?
- 
- 

## 7.9 Describing Algorithms

### Formal Description

- Word-RAM code or TM 6-tuple or TM state diagram
- Only when we ask for it!

### Implementation/Pseudocode Description

- Prose description of tape/memory contents, head movements (in case of TMs)
- High-level pseudocode and/or prose description of head movements,
- Omit details of states, transition functions, low-level RAM code (but do convince yourself that a TM/Word-RAM can do what you're describing!)
- Like our proofs that a TMs can simulate Multitape TMs can simulate Word-RAMs.

### High-Level Description

- Most of the algorithms descriptions we've seen so far.
- Freely use other algorithms we've seen as subroutines.
- Provide enough detail to be convinced of correctness and runtime on a Word RAM (possibly using some implementation-level specification or pseudocode to make things precise).
- Track number of executions of high-level steps, time for high-level steps (depends on size of data being manipulated).