Harvard CS 121 and CSCI E-207

Lecture 13: Turing Machines

Salil Vadhan

October 16, 2012

- **Reading:** Sipser, §3.1.

Turing Machines

Objective: Define a computational model that is

- General-purpose:

  (as powerful as programming languages)

- Formally Simple:

  (we can prove what cannot be computed)

The Origin of Computer Science

Alan Mathison Turing

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

What Problem Was Turing Trying to Solve?

- David Hilbert

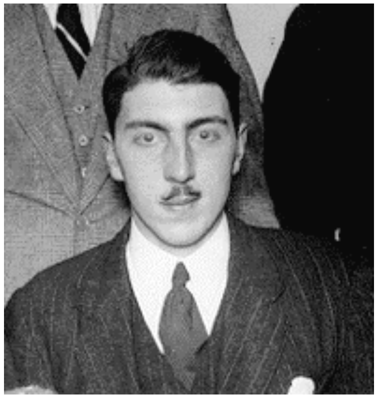
  "Mathematical Problems" 1900

The Logicians

- Kurt Gödel 

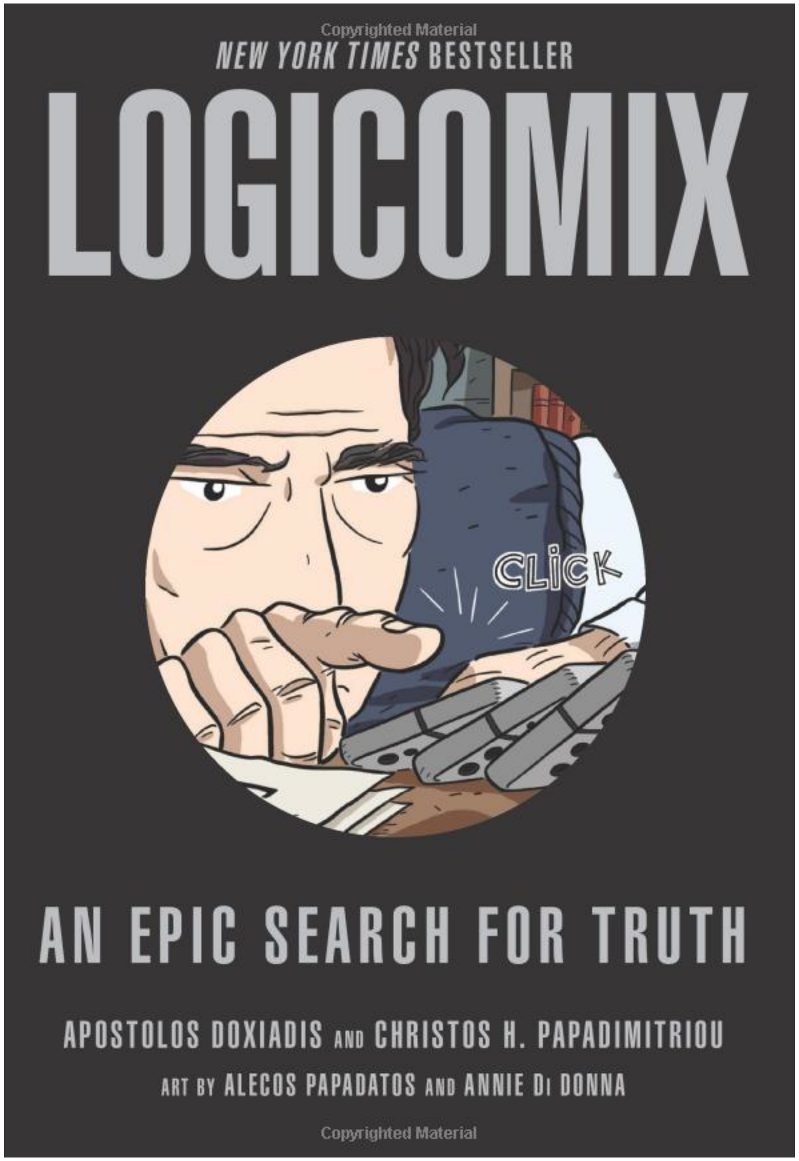
Library of Congress

“On Formally Undecidable Propositions . . . ” 1931

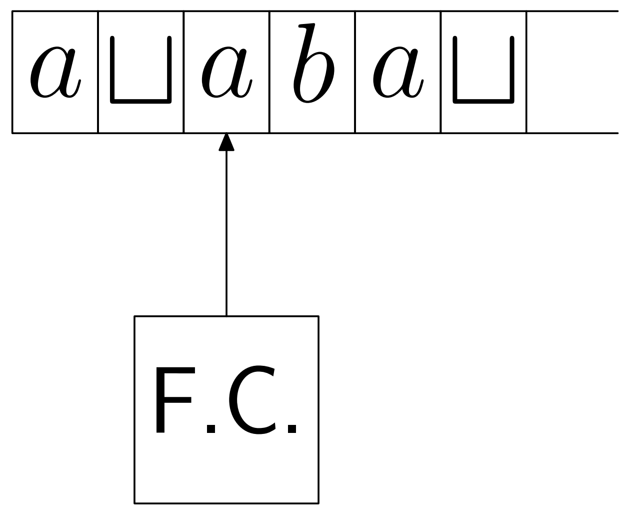
- Alonzo Church 

“An Unsolvable Problem of Elementary Number Theory” 1936

The Cliff's Notes Version of History



The Basic Turing Machine



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

Formal Definition of a TM

A (deterministic) Turing Machine (TM) is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, where:

- Q is a finite set of states, containing
  - the start state q_0
  - the accept state q_{accept}
  - the reject state q_{reject} $(\neq q_{\text{accept}})$
- Σ is the input alphabet
- Γ is the tape alphabet
  - Contains Σ
  - Contains "blank" symbol $\sqcup \in \Gamma - \Sigma$

The transition function

$$Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$$

- L and R are “move left” and “move right”

- $\delta(q, \sigma) = (q', \sigma', R)$
  - Rewrite σ as σ' 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

Computation of TMs

- A configuration is 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 = $q_0 w$, where w is input.
- One step of computation:
  - $uq\sigma v$ yields $u\sigma' q' v$ if $\delta(q, \sigma) = (q', \sigma', R)$.
  - $u\tau q\sigma v$ yields $uq'\tau\sigma' v$ if $\delta(q, \sigma) = (q', \sigma', L)$.
  - $q\sigma v$ yields $q'\sigma' v$ if $\delta(q, \sigma) = (q', \sigma', L)$.
- If $q \in \{q_{\text{accept}}, q_{\text{reject}}\}$, computation halts.

TMs and Language Membership

- M accepts w if there is a sequence of configurations $C_1, \ldots, C_k$ such that
  1. $C_1 = q_0 w$.
  2. C_i yields C_{i+1} for each i.
  3. C_k is an accepting configuration (i.e. state of M is q_{accept}).
- $L(M) = \{w : M \text{ accepts } w\}$.
- L is Turing-recognizable if $L = L(M)$ for some TM M, i.e.
  - $w \in L \Rightarrow M$ halts on w in state q_{accept}.
  - $w \notin L \Rightarrow$
    M halts on w in state q_{reject} OR M never halts (it "loops").

Decidability, a.k.a. Recursiveness

- L is <u>(Turing-)decidable</u> if there is a TM M s.t.
  - $w \in L \Rightarrow M$ halts on w in state q_{accept}.
  - $w \notin L \Rightarrow M$ halts on w in state q_{reject}.
- Other common terminology
  - Recursive = decidable
  - Recursively enumerable (r.e.) = Turing-recognizable
  - Because of alternate characterizations as sets that can be defined via certain systems of recursive (self-referential) equations.

Example

- **Claim:** $L = \{a^n b^n c^n : n \geq 0\}$ is decidable.

Questions

- Does every TM recognize some language?
- Does every TM decide some language?
- How many Turing-recognizable languages are there?
- How many decidable languages are there?