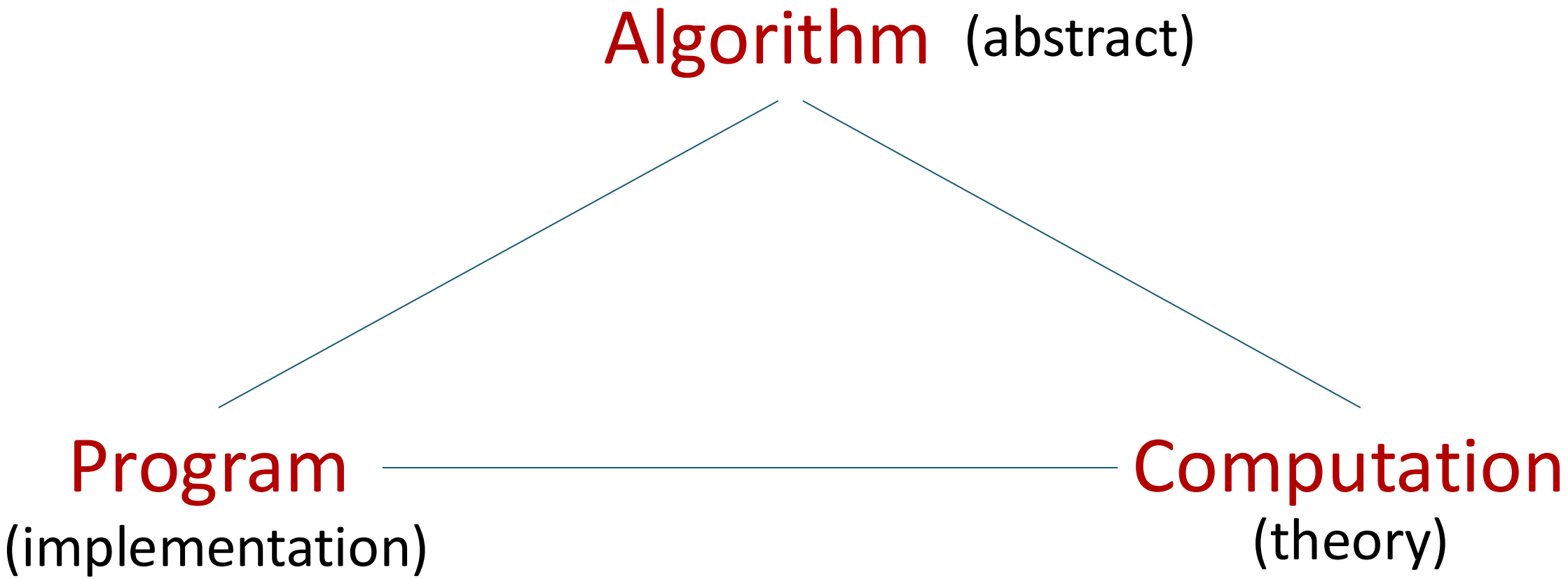
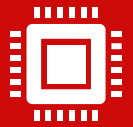
What is an algorithm?

What's the difference?



Game Plan



How to describe an algorithm?

Computational models

- Finite automata
- Turing machines
- Imperative programs (e.g., IMP and Dafny)

How to describe a problem?

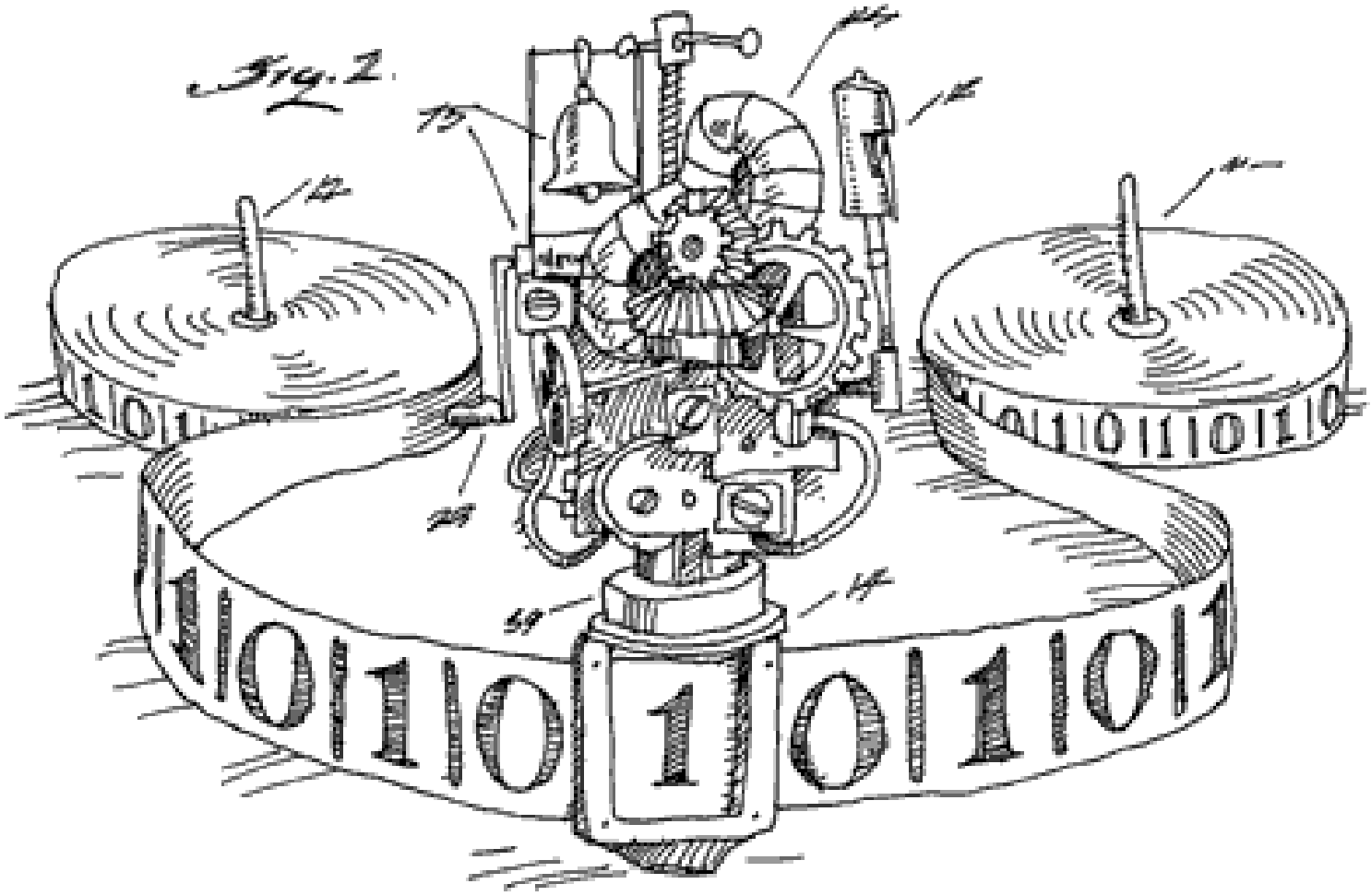
Mathematical logic

- First-order logic, monadic second-order logic, linear temporal logic

How to reason about an algorithm?

- Complexity
- Floyd-Hoare logic
- Synthesis

Turing Machine



https://www.felienne.com/archives/2974

Some history

Entscheidungsproblem (Hilbert, 1928)

- Is there an algorithm that determines whether $\Sigma \vDash \varphi$?
- "there would be no such thing as an unsolvable problem" E.g., *Hilbert's tenth problem*

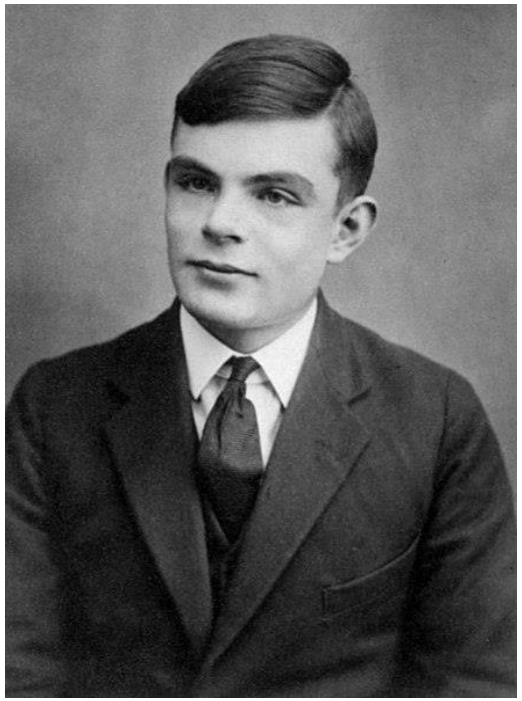
Gödel's Completeness Theorem (1929)

- Equivalent to $\Sigma \vdash \varphi$

What is an algorithm?

- Lambda calculus (Church, 1935)
- Turing machine (Turing, 1936)
- The halting problem

Church's Theorem (1935): The validity of FOL is undecidable.



What is computation?

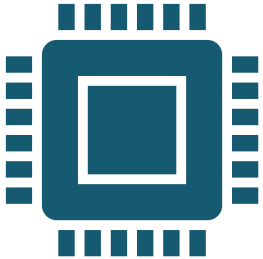
How do humans compute?

Use paper and pencil

May read and write on (infinitely long) scratch paper

May move pencil back and forth and take arbitrarily long time

May remember (a few) things in mind



Turing machines simulate how humans compute

An infinitely long tape (in both directions) divided into cells

A head positioned on a tape cell which can read, right, and move left or right

Finite control/states

How to run: based on the current state and the symbol read from the current cell, determines the next state, how to update the cell, and how to move the head

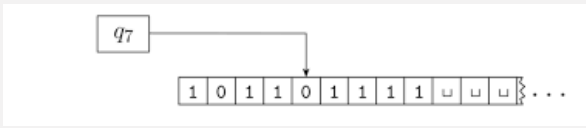
Turing machine

A *Turing machine* (TM) is $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ where

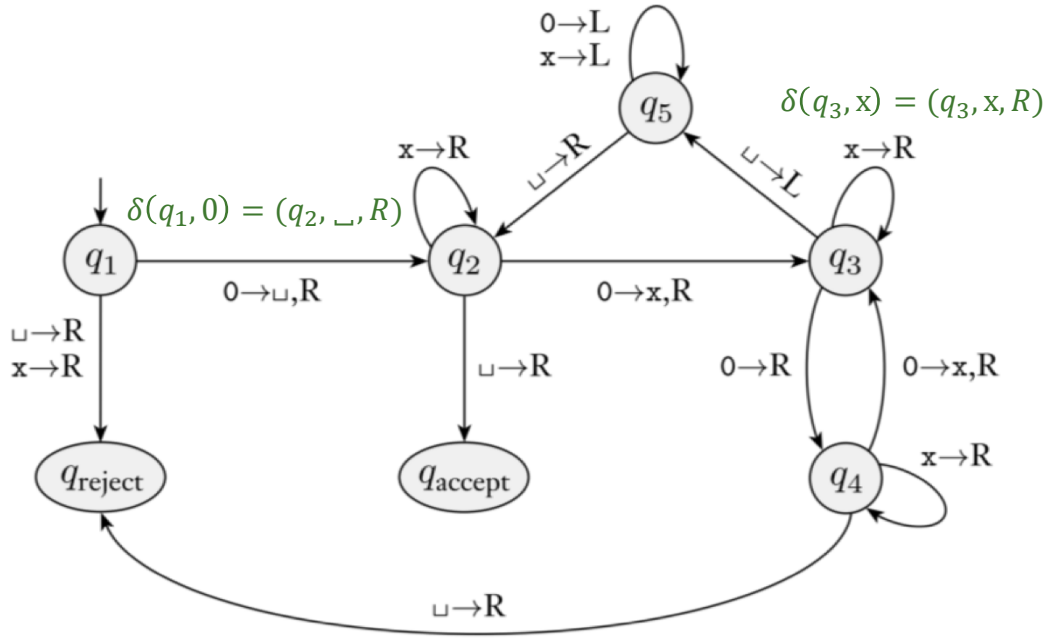
- Q is a finite set of states
- Σ is a finite set of input symbols
- $\Gamma \supseteq \Sigma \uplus \{\text{␣}\}$ is a finite set of tape symbols (␣ is the blank symbol)
- $\delta \in Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is a transition function
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$ is a set of accepting states

Configuration

- Represent the tape as a finite string $X_1 X_2 \ldots X_n$
- In state q and head on X_i:
  $X_1 X_2 \ldots X_{i-1} q X_i X_{i+1} \ldots X_n$
- E.g., $1011 q_7 0111$



Example



M. Sipser. Introduction to the Theory of Computation. Cengage Learning. 2012.

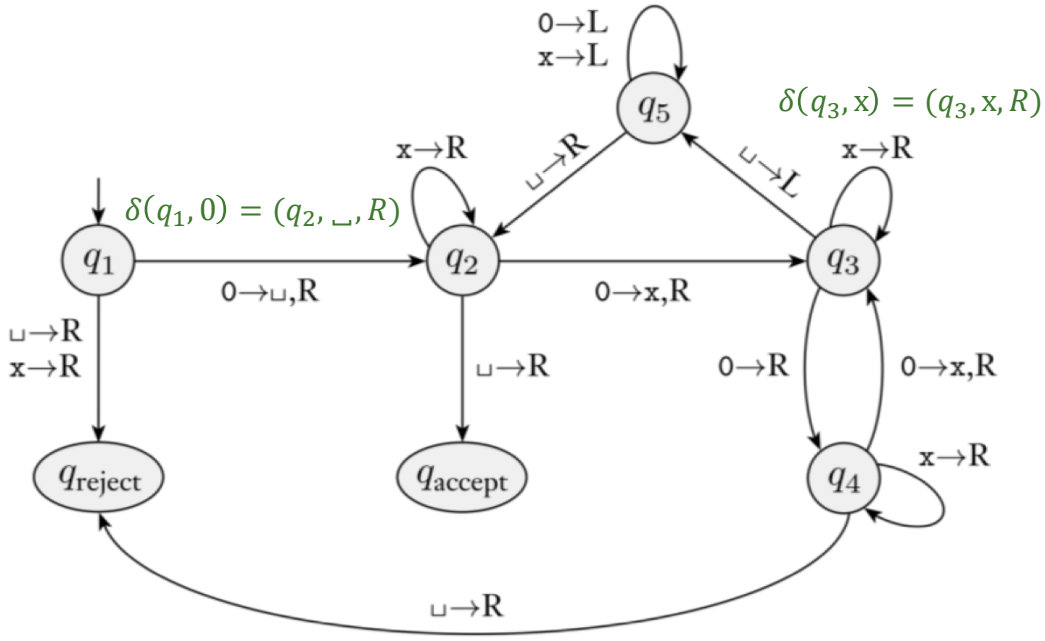
Example

A TM that decides if the input n is a power of 2.

- n is in unary representation (a string of n 0s)
- Head is at the left-hand end of the input

Idea:

1. Sweep left to right across the tape, crossing off every other 0
2. If in stage 1 the tape contained a single 0, accept
3. If in stage 1 the tape contained more than a single 0 and the number of 0s was odd, reject
4. Return the head to the left-hand end of the tape.
5. Go to stage 1.



M. Sipser. Introduction to the Theory of Computation. Cengage Learning. 2012.

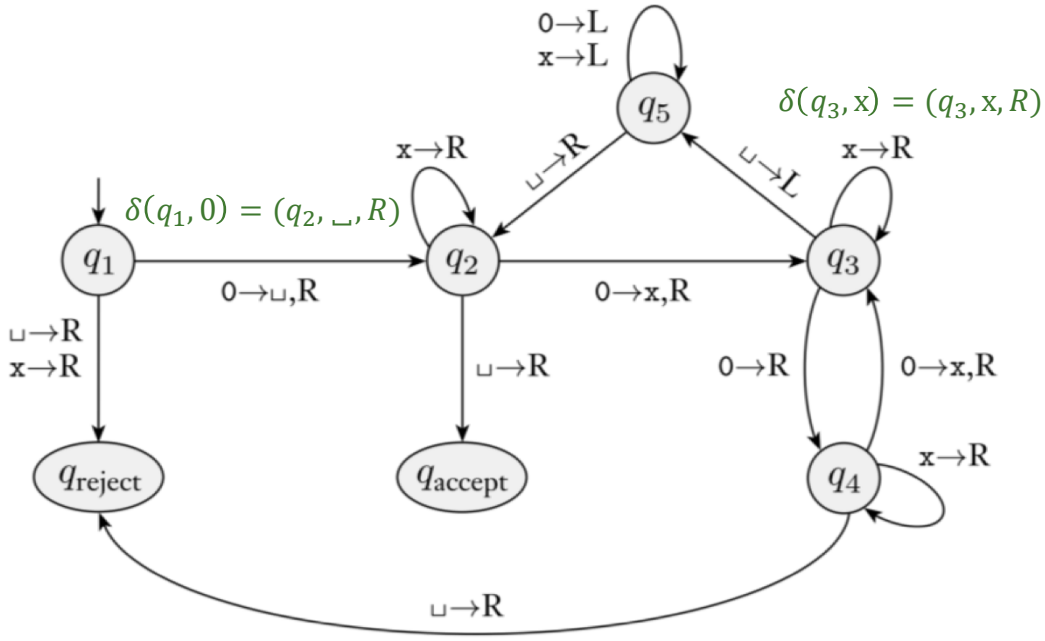
Example

A TM that decides if the input n is a power of 2.

- n is in unary representation (a string of n 0s)
- Head is at the left-hand end of the input

Idea:

1. Sweep left to right across the tape, crossing off every other 0
2. If in stage 1 the tape contained a single 0, accept
3. If in stage 1 the tape contained more than a single 0 and the number of 0s was odd, reject
4. Return the head to the left-hand end of the tape.
5. Go to stage 1.



M. Sipser. Introduction to the Theory of Computation. Cengage Learning. 2012.

Example

Design a TM that converts the unary input n to binary representation.

- E.g., 00000 -> 101
- Head starts/ends at the left-hand end of input/output

What we know/believe about TMs

Church-Turing Thesis: The intuitive notion of algorithms *equals* Turing machine algorithms.

- Equivalent to general recursive functions or λ-calculus
- Notion of computability and decidability
- Notion of complexity

Extensions to the basic TM do not add any computability power

- Multitape TMs (e.g., one for input, one for output, one for scratchpad)
- Nondeterministic TMs
- Computers that we use daily
- Basic TM can simulate all!

Theorem: the halting problem of TM is undecidable

- No TM can recognize all halting TMs
- The barber paradox

Two-counter machines

A *two-counter machine* (2CM) is an automaton with two counters (registers)

- Increment/decrement the counter
- Check if the counter is zero

$M = (Q, q_0, \delta, F)$

- Q is the set of states
- q_0 is the initial state
- $\delta \subseteq Q \times \{z, nz\}^2 \times Q \times \{-1,0,1\}^2$

Theorem (Minsky 1967): For every TM, there is a 2CM that simulates it.

Corollary: The halting problem of 2CM is undecidable.