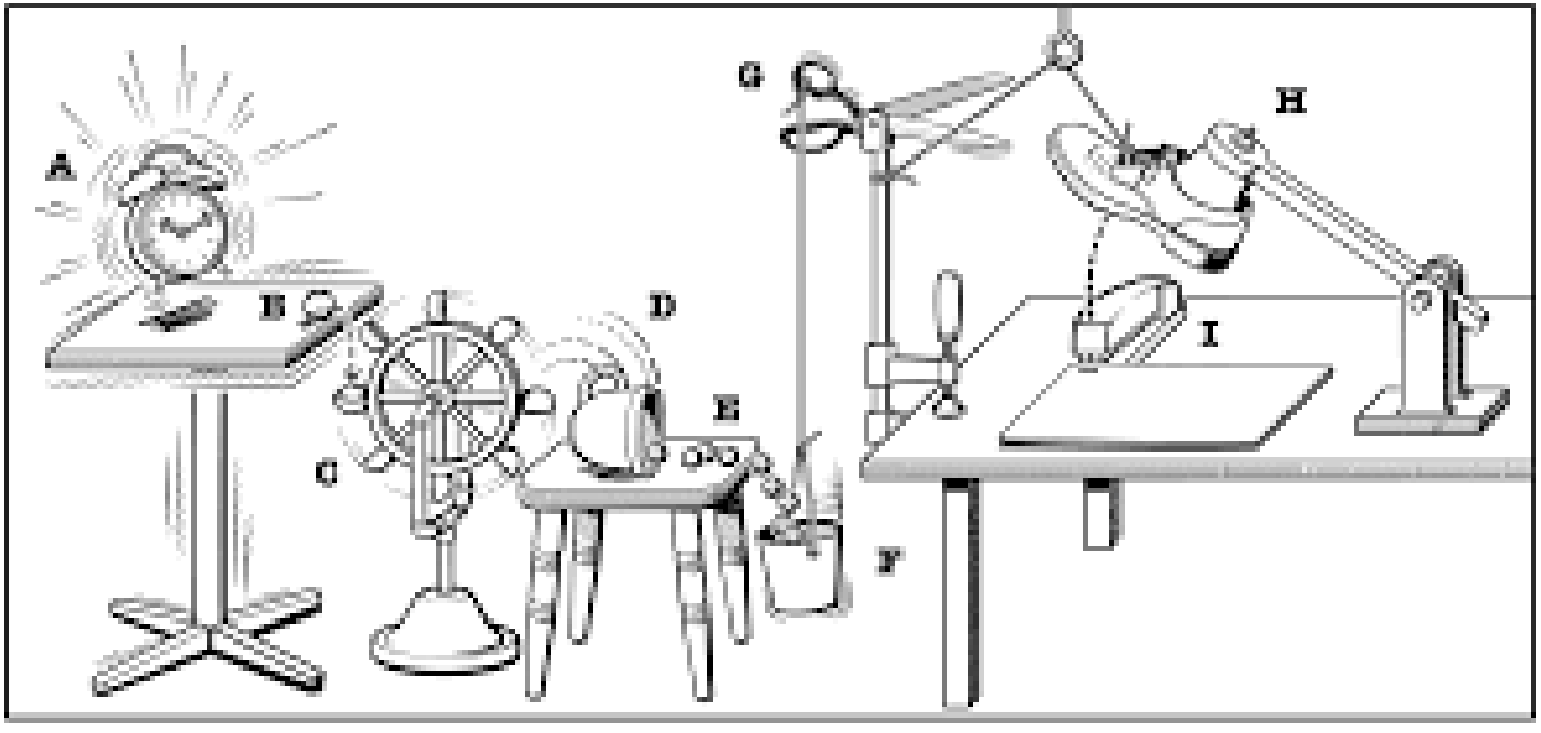
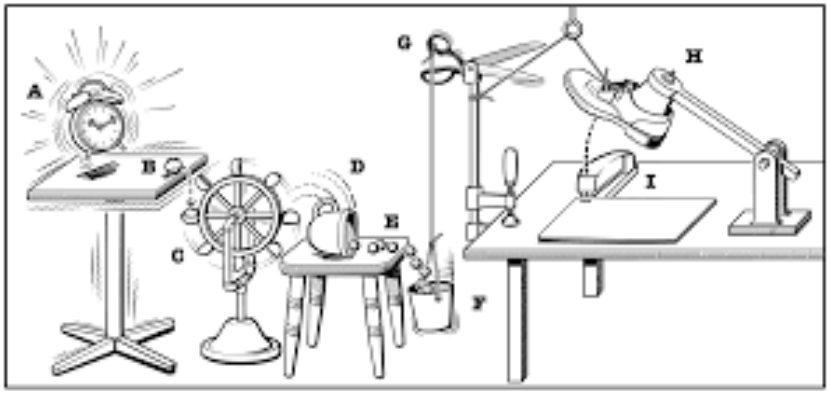
CS/Math 573:Theory of Computation

Stephen Kobourov
Prof. of Computer Science



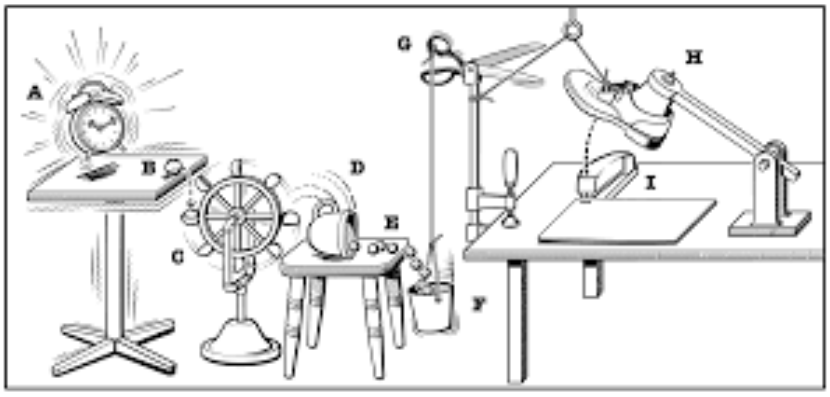
Theory of Computation



Computability:

- what problems cannot be solved: Hilbert was wrong?
- for solvable problems, how powerful a machine do we need?
- what computation models are there?

Theory of Computation



Computability:

- what problems cannot be solved: Hilbert was wrong?
- for solvable problems, how powerful a machine do we need?
- what computation models are there?

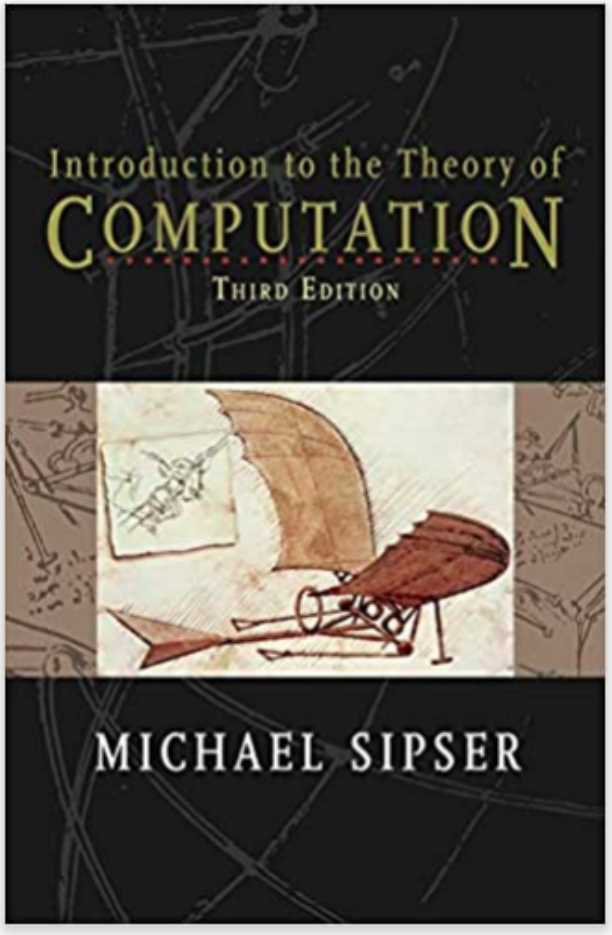
Complexity:

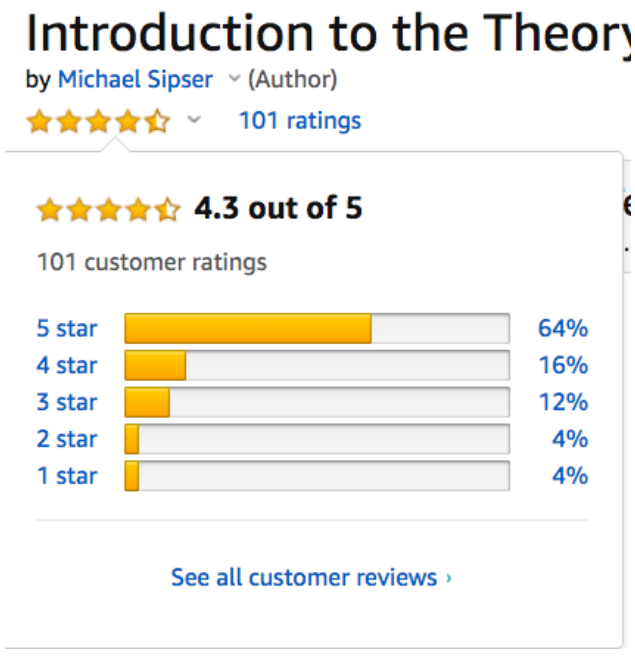
- the big divide: the classes P and NP ($1M)
- NP-completeness and reductions
- approximation/randomized algorithms

Famous and Important Results Everyone Should Know

- Chomsky's Language Hierarchy
- The Church-Turing thesis
- Der Entscheidungsproblem
- Gödel's Incompleteness Theorems
- The Cook-Levin Theorem
- The Recursion Theorem

Textbook and Other Reading





Edition: 2nd or 3rd

$20 for a used one on abebooks.com

Class Format

- Lectures, Tu/Th, 12:30-1:45pm
- 6 written homework assignments (every 2 weeks)
- Midterm (in class) exam
- Comprehensive final exam
- Starting in-person...
- Syllabus
- Questionnaire
- Reading: Chapter 0

"Okay, who invited that clown to this meeting?"

Automata, Grammars, and Languages

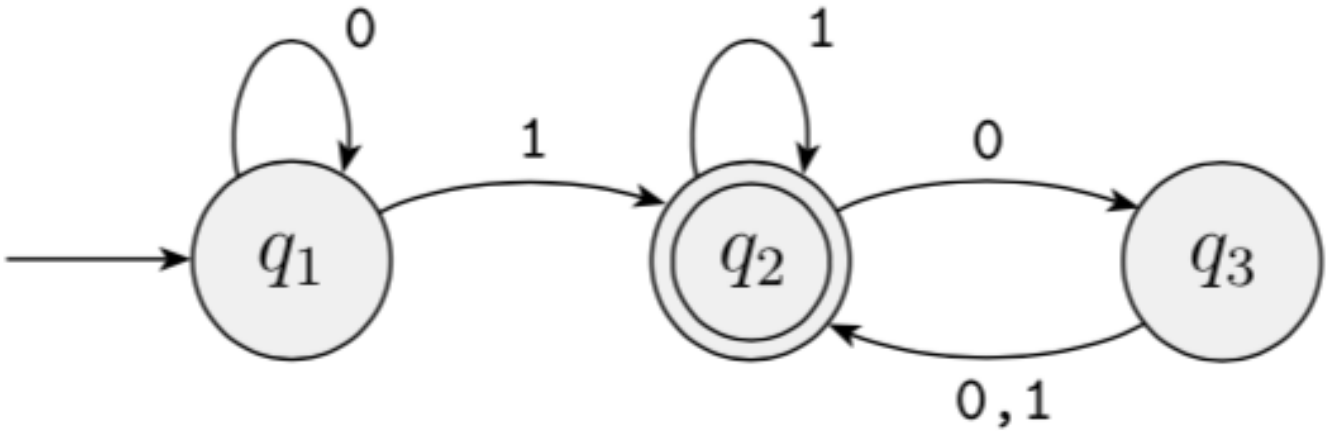
Recall that Theory of Computation deals with two majors aspects: computability and complexity

- Computability: What are the fundamental capabilities and limitations of computers?
- Complexity: What makes some problems computationally hard and others easy?
- Reading: Chapter 1

Automata, Grammars, and Languages

- Automata (machines) come in different strengths, depending on their ability to remember stuff (memory) and on the way to look up stuff (access)
- Automata can be seen as processing strings over some alphabet (e.g., binary) and either accept or reject each valid string
- The language of an automaton is the set of strings it accepts
- A grammar is made of rules that generate all strings in a given language
- For now computation simply means determining whether a given string is in the language or not (decision problem)

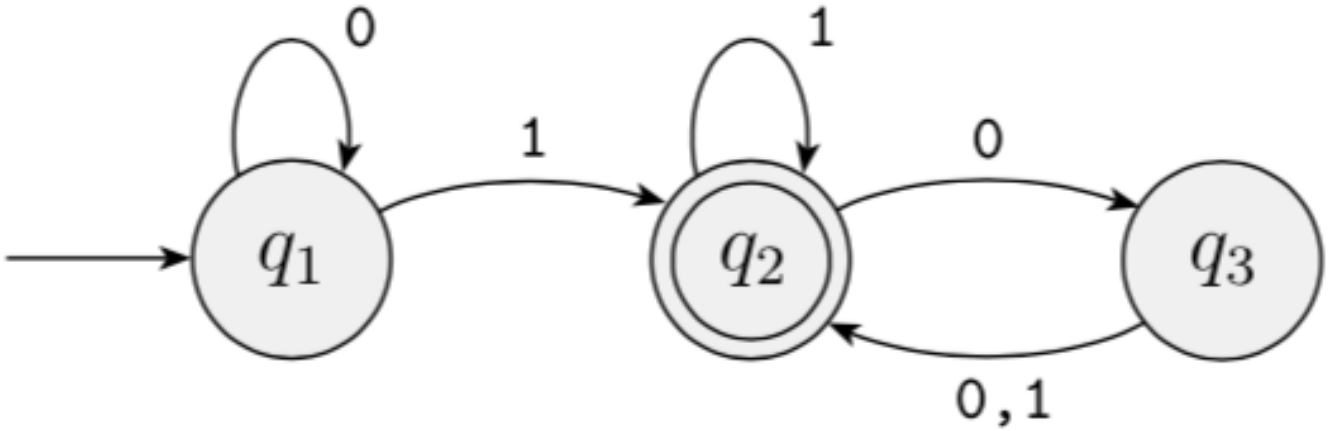
The Simplest Computational Model: Finite Automata



A finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_s, F)$, where

- Q is a finite set called the states,
- Σ is a finite set called the alphabet,
- $\delta : Q \times \Sigma \to Q$ is the transition function,
- $q_s \in Q$ is the start state,
- $F \subseteq Q$ is the set of accept states.

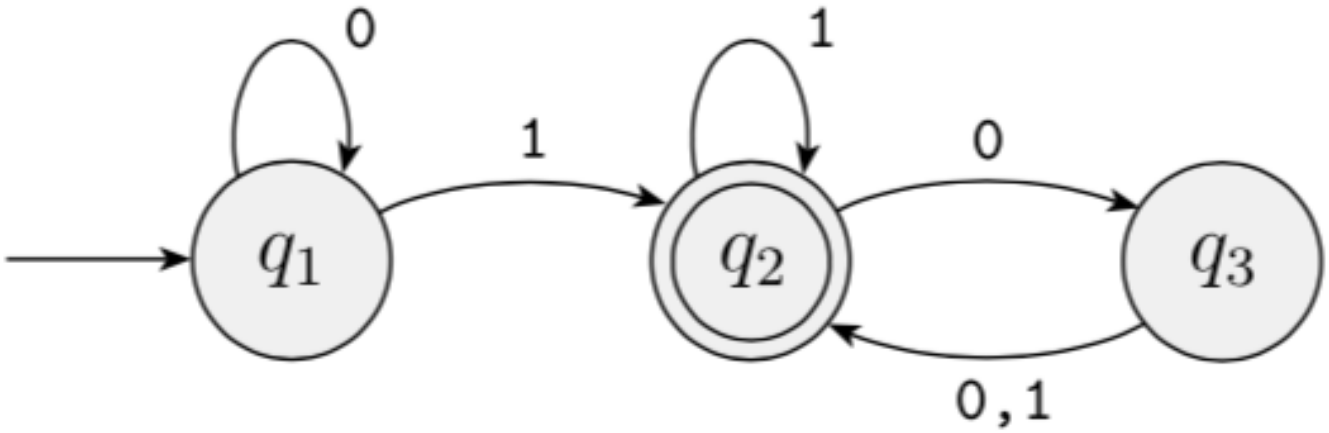
Finite Automata



This particular finite automaton is given by the 5-tuple $(Q, \Sigma, \delta, q_s, F)$, where

- $Q =$
- $\Sigma =$
- δ is:
- start state is:
- accepts states are:

Finite Automata



- What does this machine M do?
- The set of all strings that M accepts is the language of the machine
- $A = \{w | w$ contains at least one 1 and an even number of 0s follow the last $1\}$.

Computation of a Finite Automaton

Let $M = (Q, \Sigma, \delta, q_S, F)$ be a finite automaton and let $w = w_1 w_2 \dots w_n$ be a string where each w_i is a member of the alphabet Σ. Then M accepts w if a sequence of states $r_0, r_1, \dots, r_n$ in Q exists with three conditions:

- $r_0 = q_S$,
- $\delta(r_i, w_{i+1}) = r_{i+1}$, for $i = 0, ..., n-1$,
- $r_n \in F$.

A language is called a **regular language** if some finite automaton recognizes it.

Regular Operations

Let A and B be languages. We define the regular operations union, concatenation, and star as follows:

- Union: $A \cup B = \{x | x \in A \text{ or } x \in B\}$.
- Concatenation: $A \circ B = \{xy | x \in A \text{ and } y \in B\}$.
- Star: $A^* = \{x_1 x_2 \dots x_k | k \geq 0 \text{ and each } x_i \in A\}$.

A good to introduce:

- ϵ: the empty string
- ϵ: the empty language

Properties of Regular Languages

Theorem

The class of regular languages is closed under the union operation.

Proof.

Given two regular languages A_1 and A_2 we want to show that $A_1 \cup A_2$ also is regular. Let M_1 be a finite automaton for A_1 and M_2 be a finite automaton for A_2. To prove that $A_1 \cup A_2$ is regular, we construct a finite automaton M that recognizes $A_1 \cup A_2$, using M_1 and M_2 as building blocks. M works by simultaneously simulating M_1 and M_2 and accepting if either of the simulations accept. M can keep track of the simulations by using as many states as the product of the states in M_1 and M_2. □

Closure under Union

Example