Theory of Computation Lecture 1

Dr. Nahla Belal

Book

- The primary textbook is:
  Introduction to the Theory of Computation by Michael Sipser.

Grading

- 10%: Weekly Homework.
- 30%: Two quizzes and one exam.
- 20%: One quiz and one exam.
- 40%: Final Exam.

Syllabus

- Automata and Languages
  - Finite Automata
  - DFAs and NFAs
  - Regular Expressions
  - Non-regular Languages
  - Context-Free Languages (Grammars, Pushdown Automata, non-context-free languages)
- Computatbility Theory
  - Turing Machines
- Complexity Theory

Theory of Computation

- Theory of computation is the branch that deals with how efficiently problems can be solved on a model of computation, using an algorithm. In order to perform a rigorous study of computation, computer scientists work with a mathematical abstraction of computers called a model of computation. There are several models in use, but the most commonly examined is the Turing machine.
- Alan Turing in 1937 proposed that all computation could be performed by a special kind of a machine called a Turing machine. He based the model on the actions that people perform when involved in computation. He abstracted these actions into a model for a computational machine that has really changed the world.

About the Course

- Fundamental mathematical properties of hw, sw, and applications.
- Determine what can and can not be computed, how much memory, which computational models.
- Theory expands your mind, gives you the ability to think, solve problems.
- Know the capabilities and limitations of computers.

Automata, Computability and Complexity

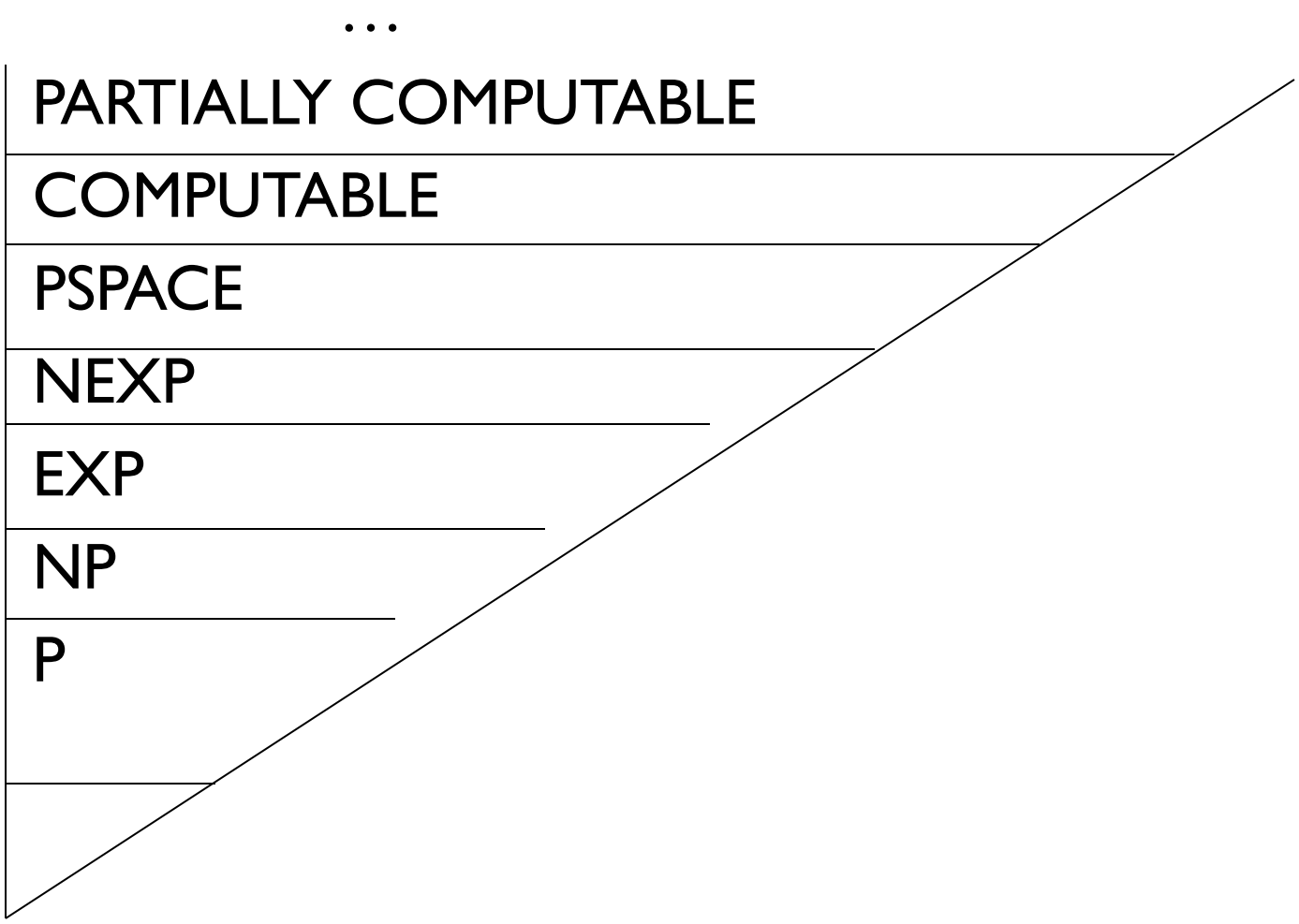
- What are the fundamental capabilities and limitations of computers? Three interpretations.
- Complexity Theory
  - What makes some problems computationally hard and others easy?
  - Classifying problems according to their computational difficulty.
- Computability Theory
  - Which problems are solvable and which are not.
- Automata Theory
  - Definitions and properties of mathematical models of computation.

Complexity of Algorithms

- If an algorithm runs in time $O(\log(n))$, $O(n)$, or $O(n^4)$, then faster machines may make a big difference

- If an algorithm runs in time $O(2^n)$, or $O(n^{\log(n)})$, then faster machines may not make a big difference

Problem Classes



Mathematical Notation

- Sets, power sets, cartesian product
- Sequences and Tuples
- Functions and Relations
- Graphs
- Strings and Languages
- Types of Proofs(construction, contradiction, induction, …)

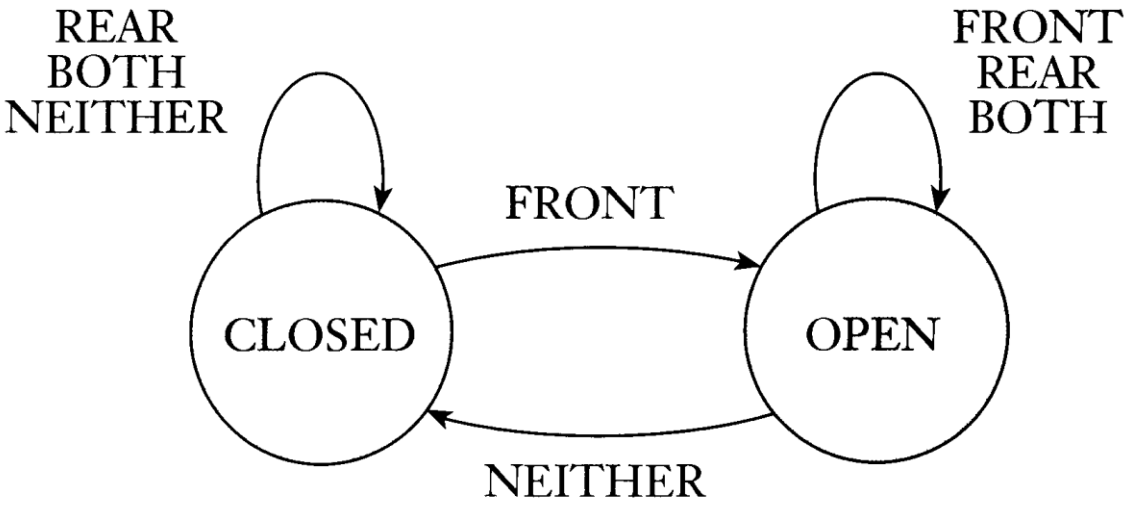
Computational Model

- Theory of computation begins with a question:
  - What is a computer?
- Idealized computer.
  - A computational model is a mathematical model in computer science to study the behavior of a system by computer simulation.
- A Turing machine is a mathematical model of a hypothetical computing machine with infinite memory.
- Simplest model is the finite state machine or finite automaton.

Finite Automata

- One pass through the input
- No capability to write
- Turing Machines with constant amount of memory
- What can we do with such a small memory?

Example



state	input signal: NEITHER	FRONT	REAR	BOTH
CLOSED	CLOSED	OPEN	CLOSED	CLOSED
OPEN	CLOSED	OPEN	OPEN	OPEN

Deterministic Finite Automata
DFA definition

- A finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$
  - Q is a set of *states*
  - Σ is an alphabet over which input strings are defined
  - δ: $Q \times \Sigma \rightarrow Q$ is a *transition function* between states
  - q_0 is the start state
  - F is a set of *accept states*

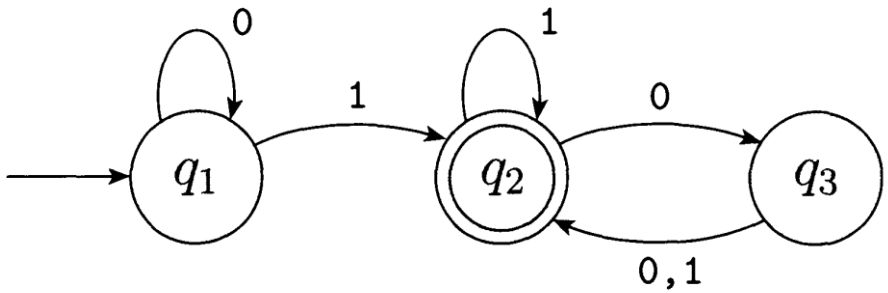
Alphabets, Σ

- Any set of distinct symbols, for example:
  - {0,1}
  - {a,b,c,d … z}
  - {a,b,c,d, …z, A, B, … Z, 0,1,2 …9}
  - ASCII

“A String over an Alphabet”

- A sequence of letters (with replacement) from that alphabet:
  - 0001111000
  - SOS

Examples of Finite Automata: 1



1. $Q = \{q_1, q_2, q_3\}$,

2. $\Sigma = \{0,1\}$,

3. δ is described as

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

4. q_1 is the start state, and

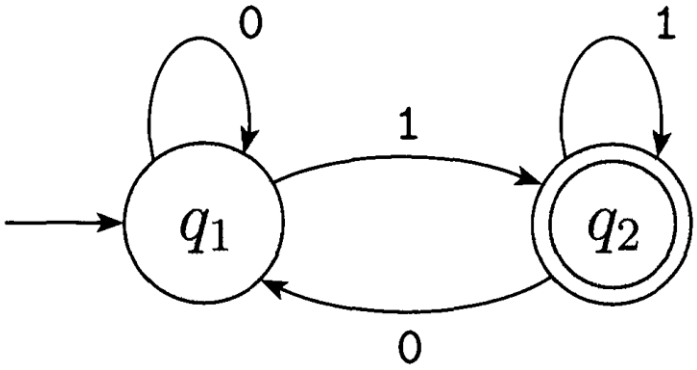
5. $F = \{q_2\}$.

$A = \{w |$ w contains at least one 1 and an even number of 0s follow the last 1$\}$.

$$L(M_1) = A$$

M_1 recognizes A

Examples of Finite Automata: 1

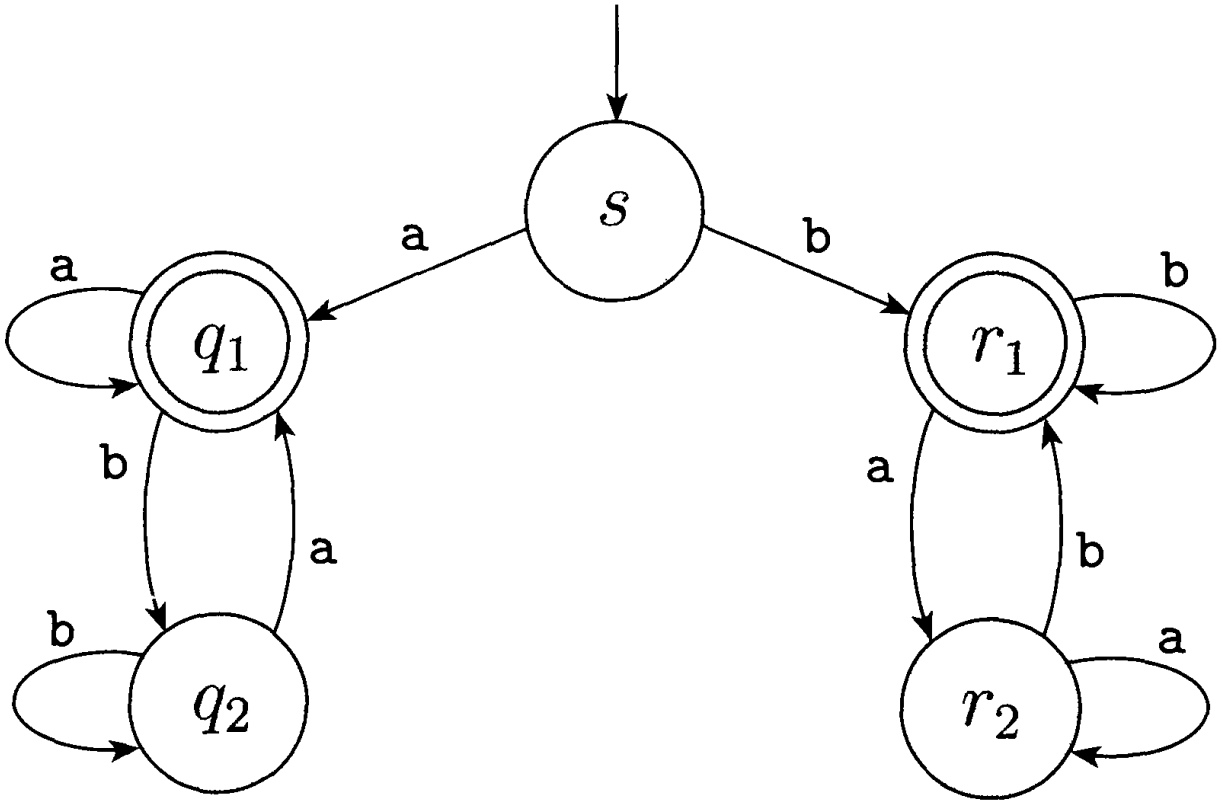


$$M_2 = (\{q_1, q_2\}, \{0,1\}, \delta, q_1, \{q_2\})$$

	0	1
q_1	q_1	q_2
q_2	q_1	q_2

$$L(M_2) = \{w | \, w \text{ ends in a } 1\}$$

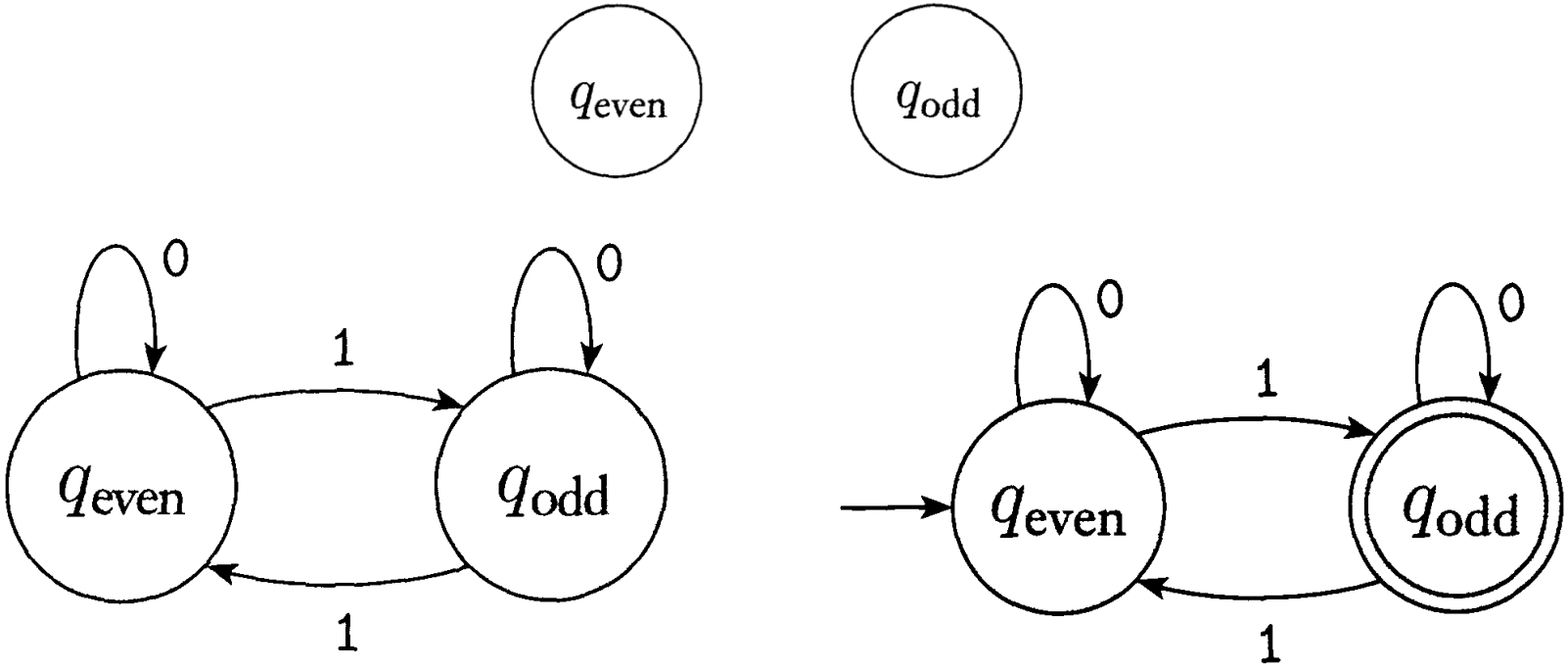
Examples of Finite Automata: 2



Languages

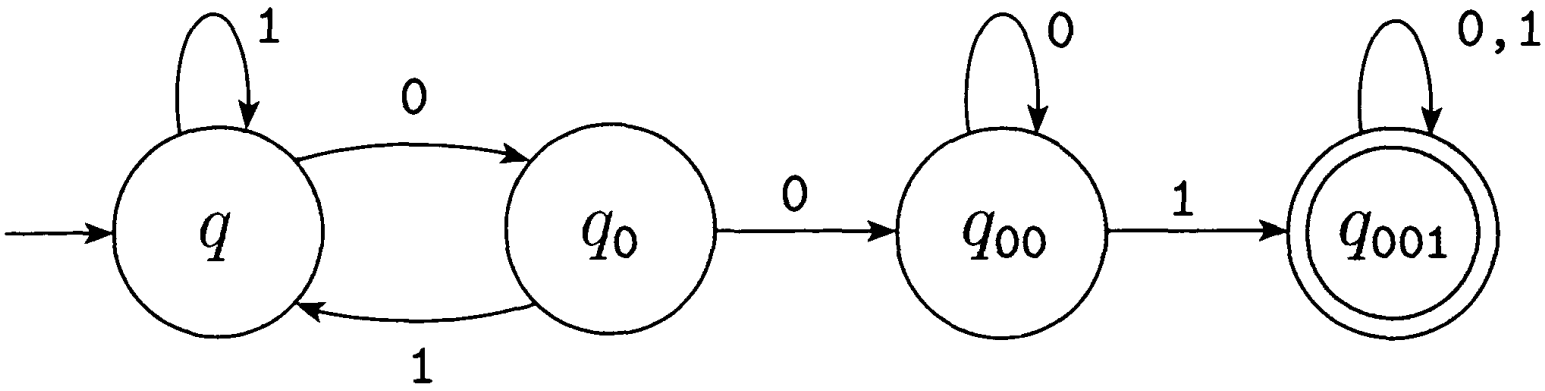
- A language L is a subset of S^*. A binary language is a subset of $\{0,1\}^*$.
- The language accepted (computed) by a DFA *M*= L(*M*) is the set of all strings *w* such that *M* ends in an accept state on input *w*.
- L is a regular language if there is a DFA that accepts it.

Designing Finite Automata: Odd 1s



Designing Finite Automata: 2

- Accept strings that contain the substring 001



The Regular Operations

- Union
- Concatenation
- Star
- The class of regular languages is closed under the union and concatenation operations.

Reading Materials

- Chapter 0
- Chapter 1: Section 1.1 Finite Automata