Theory of Computation

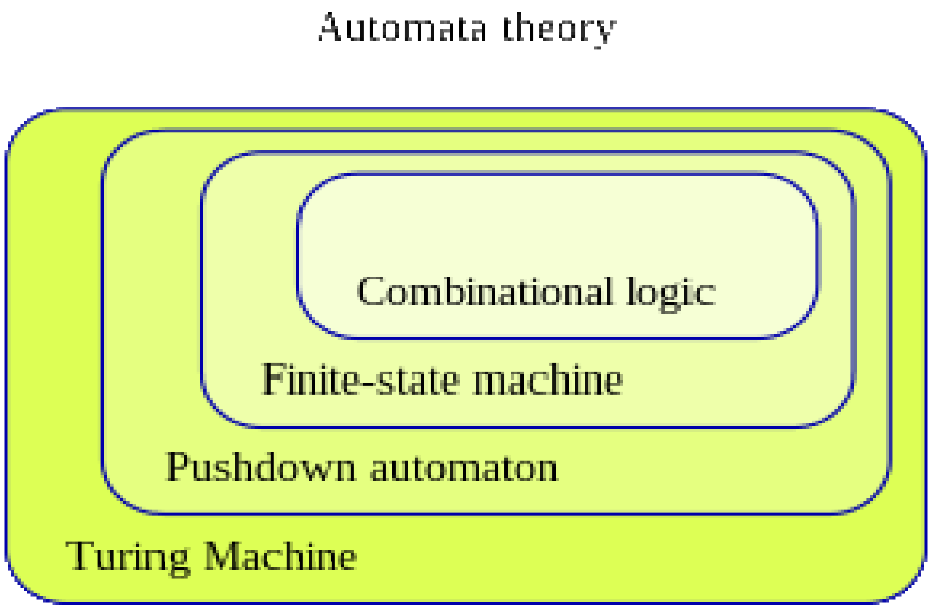
2023--2024 Fall
Zeynep Altan

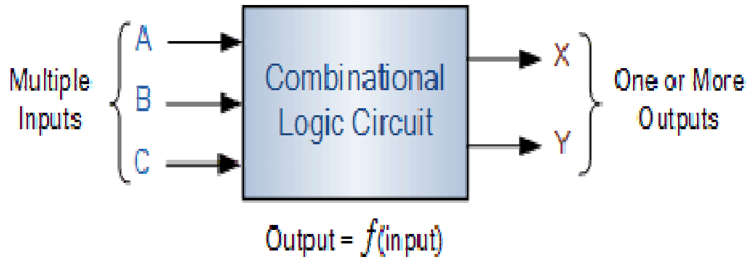
Types of Theory of Computation

- ❑ Automata Theory -
  - ❖ Game of Life- Cellular Automaton - British mathematician John Horton Conway 1970
    - ✓ Turing Complete - no physical system can have infinite memory, but if the limitation of finite memory is ignored, most programming languages are Turing-complete.
- ❑ Formal Language
- ❑ Computability theory
- ❑ Complexity theory

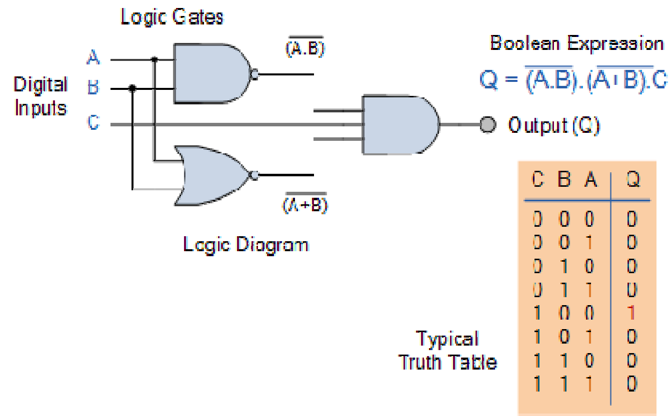
Automata Theory

An automaton with a finite number of states is called a Finite Automaton (FA) or Finite State Machine (FSM)





Boolean Algebra
Truth Table
Logic Diagram



Basic terms in Automata Theory

Symbols: These are either individual objects or separate entities. These can be any letter, alphabet or any picture.

Strings: These are a finite collection of symbols from the alphabet, and are denoted by w.

Language: A collection of appropriate strings is called a language. A language can be Finite or Infinite.

Automata Theory

❑ How a vending machine works?

Vending machine room seen in Hokkaido, Japan 2004

How to design a vending machine?

$\rightarrow$ *Use a finite automaton!*

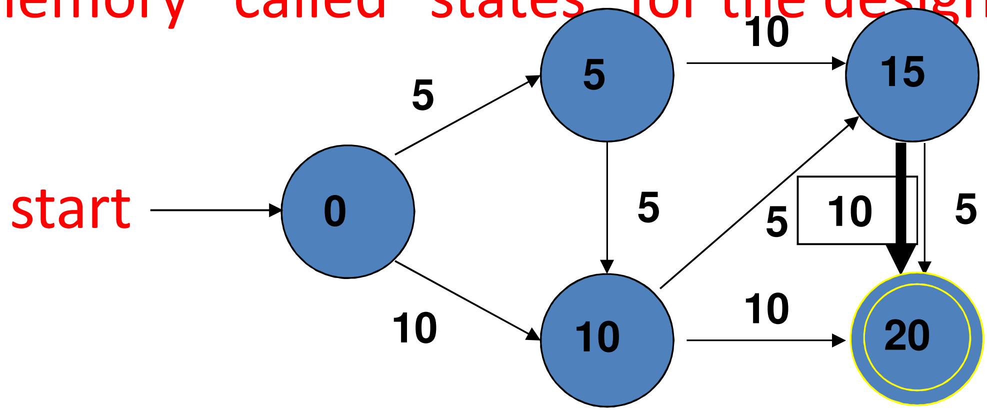
Automata Theory

An example: Assumptions (for simplicity)

Only 5-L and 10-L are used.

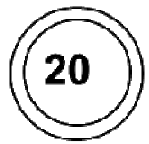
Only drinks all of 20 L are sold.

Requiring "memory" called "states" for the design.



transition diagram

10 ↓ ➡ 5 is returned as "output" (not shown)



⇨ "Final state"

Vending Machine Autamaton

The states of Deterministic Finite Automata (DFA) for Vending Machines include:

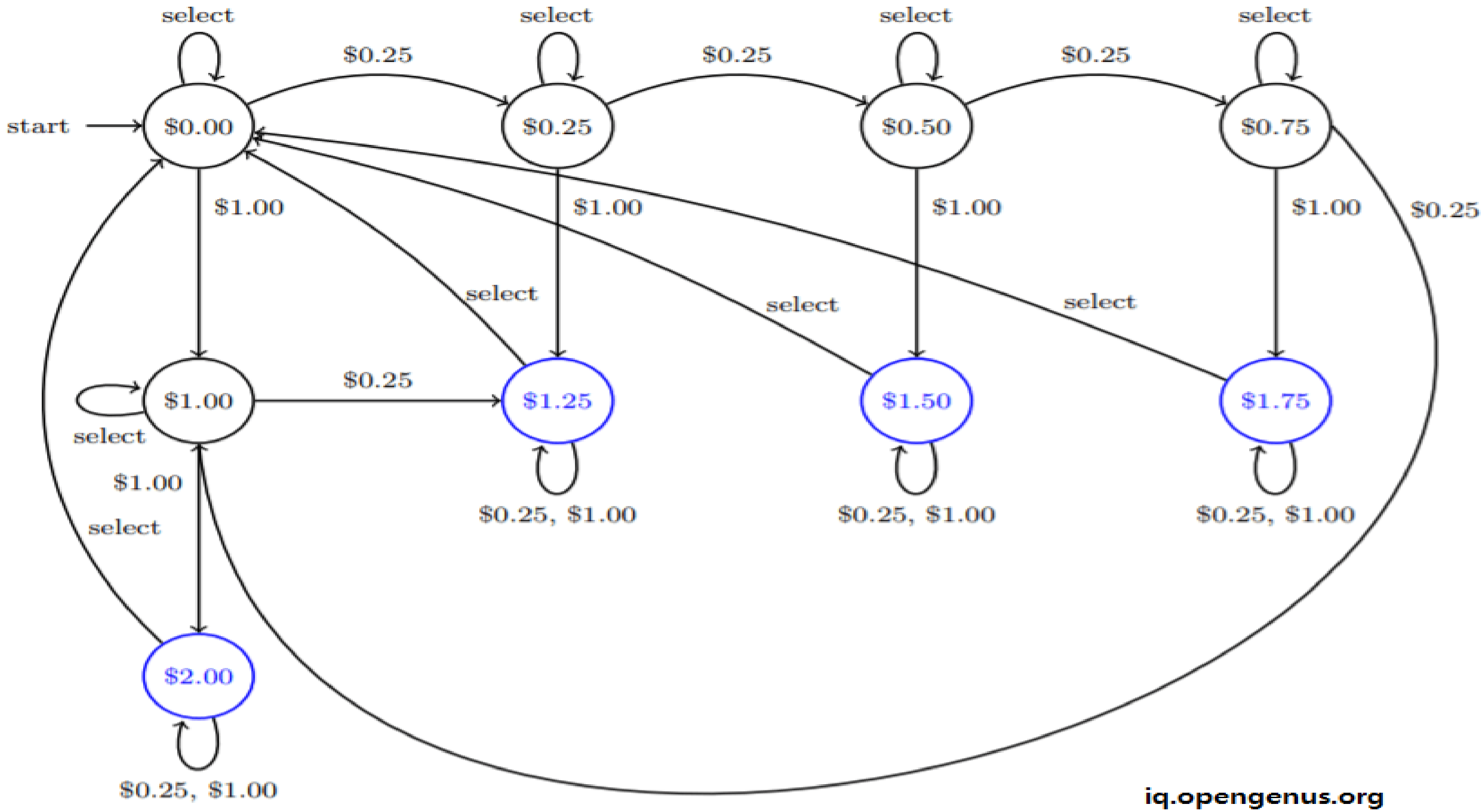
Q = {\$0.00, \$0.25, \$0.50, \$0.75, \$1.00, \$1.25, \$1.50, \$1.75, \$2.00} (states)

Σ = {\$0.25, \$1.00, select} is the alphabet

q_0 = \$0.00 is the start state

A = $\emptyset$ is the set of accept states

Vending Machine



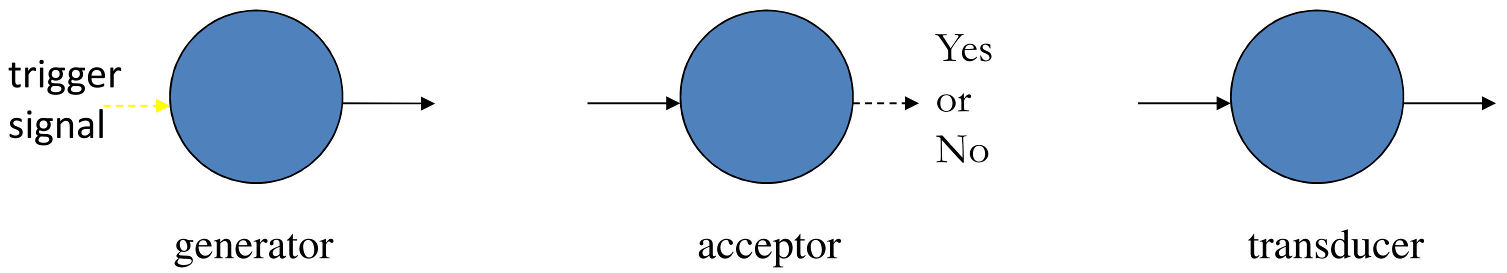
Other Two Applications of Theory of Computation

- ❑Entire Universe
  - ❖model with a Automata Machine
    - ✓Theory of Computation
  - ❖similar to existing theories in Physics

    https://www.bristol.ac.uk/maths/research/highlights/riemann-hypothesis/
- ❑Complexities in Natural Selection in Biology -Automata Theory

Automata Theory

- ❑Three major models of automata
  - generator --- with output and without input
  - acceptor --- with input and without output
  - transducer --- both with input and with output



Automata Theory Model Type I: Generator

- "natural language" grammar

(generating "sentences" spoken by people)

- reception robot

(speaking organized words and sentences)

- context-free grammar

(generating strings of symbols)

Reception robot
--- Expo 2005

Automata Theory Model Type II: Acceptor

- **digital lock**

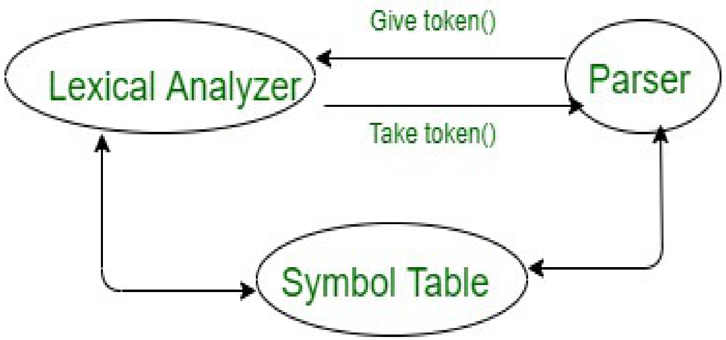
  (accepting digits)

- **lexical analyzer**

  (recognizing computer language keywords)

- **finite automaton**

  (accepting valid strings of symbols)



Automata Theory Model Type III: Transducer

- ❑ <u>Interpreter</u>

(translating natural languages)

- ❑ <u>Compiler</u>

(translating high-level languages into machine codes)

- ❑ <u>Turing machine</u>

(transforming strings of symbols)

Real Life Applications of Deterministic Finite Automata

Traffic Lights

Video Games

CPU Controllers

Protocol Analysis /Design

Regular Expression Matching

Vending Machines

Speech Recognition

Natural Language Processing

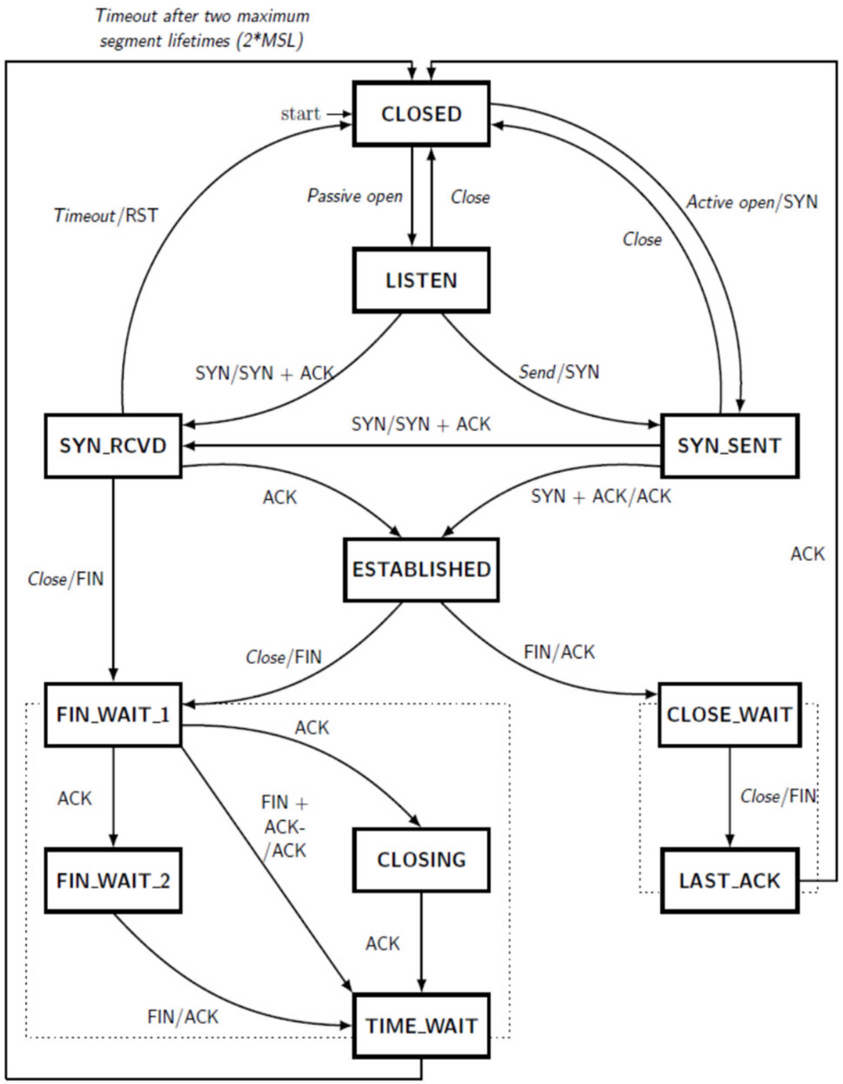
Asynchronous circuits /Digital Circuit Design

Coding theory,

Concurrent Systems

Software and Hardware verification

Hardware Testing



Internet Protocols: TCP as a DFA

Examples of Automata I: Sequential Machines

- ❑ Asynchronous circuits /digital circuit design
- ❑ Coding theory,
- ❑ Concurrent systems
- ❑ Software and hardware verification
- ❑ Hardware testing
- ❑ Protocol design

Example of Automata II: Vending Machines

- A vending machine is an automated machine that dispenses numerous items such as cold drinks, snacks, beverages, alcohol etc. to sales automatically, after a buyer inserts currency or credit into the machine.

- Vending machine is works on finite state automate to *control the* functions *process.*

Example of Automata III: Traffic Lights

- The optimization of traffic light controllers in a city is a systematic representation of handling the instructions of traffic rules.

- Its process depends on a set of instruction works in a loop with switching among instruction to control traffic.

Examples of Automata IV

Video Games: Video games levels represent the states of automata. In which a sequence of instructions are followed by the players to accomplish the task.

Text Parsing: Text parsing is a technique which is used to derive a text string using the production rules of a grammar to check the acceptability of a string.

Regular Expression Matching: It is a technique to checking the two or more regular expression are similar to each other or not. The finite state machine is useful to checking out that the expressions are acceptable or not by a machine or not.

Example of Automata V: Speech Recognition

- Speech recognition via machine is the technology enhancement that is capable to identify words and phrases in spoken language and convert them to a machine-readable format
- *Receiving words* and phrases from real world and then *converting* it into *machine readable language* automatically is effectively solved by using finite state machine.

Summary: Applications of Theory of Computation

- Text analysis
  - text search
  - text editing
- Compiler design
  - lexical analysis
  - parser generation
- Digital system design
  - computer design
  - special digital system design
- Protocol modeling and verification
- Expert system design
- Cryptography …
- Language design
  - programming language design
  - document description language design
    - e.g., HTML, XML, …
  - picture language design
    - e.g., SVG, VHML, …
  - special language design

Fields Related to Scope of Theory of Computation

Fields	Related theory
Compiling theory	Formal languages
Switching circuit theory	Automata theory
Algorithm analysis	Computational complexity
Natural language processing	Formal languages
Syntactic pattern recognition	Formal languages
Programming languages	Formal languages
Artificial intelligence	Formal languages and automata theory
Neural networks	Automata theory

Introduction to Computability

- Is there a flight between Detroit and New York for less that $100?
  - Such a question is a **decision problem**
    - we want a decision on whether the answer is **yes or no.**
- We can generalize the question above into a **predicate** that takes an input value:
  - Is there a flight between Detroit and x for less than $100?
    - Given any particular destination x, the answer to this question is still either **yes or no.**
- We can further generalize this predicate to work with multiple input values:
  - Is there a flight between x and y for less than z?

Computability Theory / Recursion Theory

- An effective procedure that can be carried out by following specific rules.
- We might ask whether there is some effective procedure, some algorithm
  - given a sentence about the integers will decide whether that sentence is true or false.
- The set of even integers is **decidable**

The difference between Decidable and Recognizable

- A language is said to be **Decidable** if there is a machine that will
  - accept strings in the language

  and
  - reject strings not in the language

- A language is called **Turing Recognizable** if some Turing Machine recognizes it.
- A Language is called Turing Decidable if some Turing Machine decides it.

Problems Studied in Theory of Computation

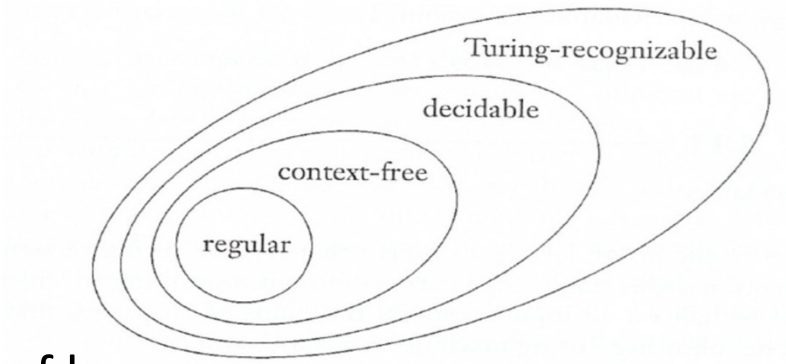
- "What are the fundamental capabilities and limitations of computers?"
  - What can a computer do at all? ---

    studied in the domain of Computability!
  - What can a computer do efficiently? --- studied in the domain of

    Computational complexity!

How do you Determine Decidable?

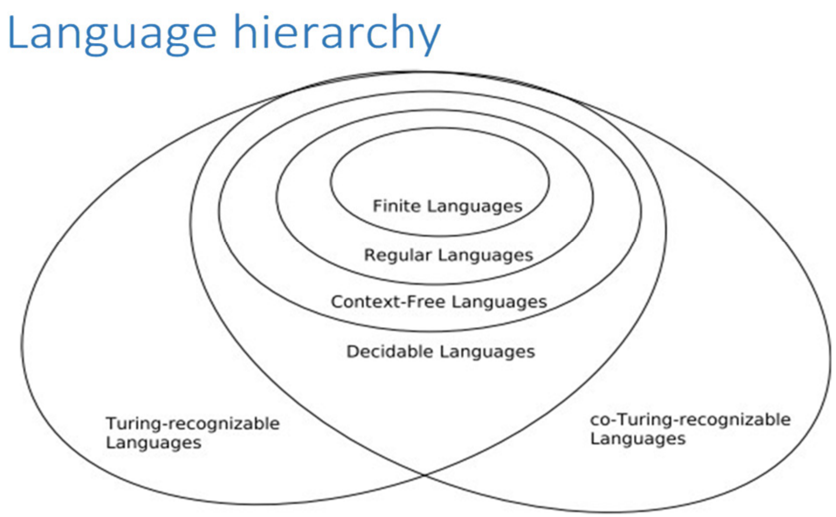
- A language is called Decidable or Recursive if there is a Turing machine which accepts and halts on every input string w.
- Every decidable language is Turing-Acceptable.
- A decision problem P is decidable if the language L of all yes instances to P is decidable.



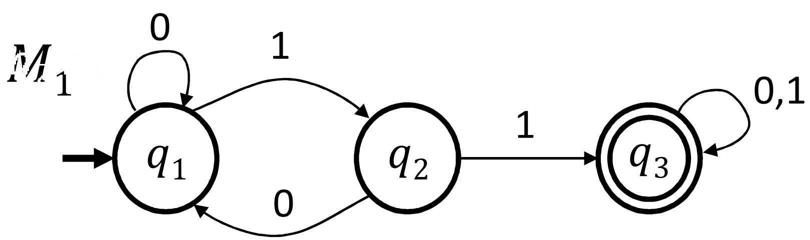
The relationships among classes of languages

Language Hierarchy

- **Turing-recognizability** means that there is a program that can confirm that a string w is in a language
- **co-Turing-recognizability** means that there is a program that can confirm that a string w is not in the language.



Finite Automata



States: q_1 q_2 q_3

Transitions: 

Start state: : q_1

Accept states: q_3

Input: finite string

Output: Accept or Reject

Computation process: Begin at start state, read input symbols, follow corresponding transitions, Accept if end with accept state, Reject if not.

Examples: 01101 → Accept

00101 → Reject

M_1 accepts exactly those strings in A where

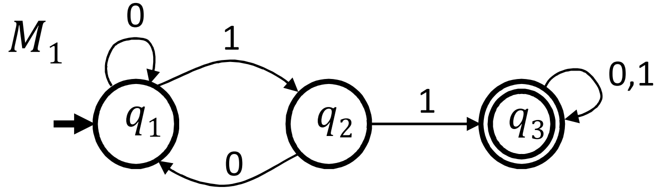
$A = \{w \mid w$ contains substring $11\}$.

Definition FA

A <u>finite automaton</u> M is a 5-tuple $(Q, \Sigma, \delta, q_0, A)$

Q finite set of states

- Σ finite set of alphabet symbols
- δ transition function $\delta\colon\ Q \times \Sigma \ \rightarrow\ Q$
- q_0 start state
- A set of accept states



$M_1 = (Q, \Sigma, \delta, q_1, F)$

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

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

$= \{q_3\}$

$\delta =$	0	1
q_1	q_1	q_2
q_2	q_1	q_3
q_3	q_3	q_3

Finite Automata – Computation

Strings and languages

- A <u>string</u> is a finite sequence of symbols in Σ
- A <u>language</u> is a set of strings (finite or infinite)
- The <u>empty string</u> ε is the string of length 0
- The <u>empty language</u> ø is the set with no strings

Definition: M <u>accepts string</u>

$w = w_1 w_2 \ldots w_n$ each $w_i \epsilon \Sigma$

if there is a sequence of states $r_0, r_1, r_2, , \ldots, r_n \epsilon Q$
where:

- $r_0 = q_0$
- $r_i = \delta(r_{i-1}, w_i)$ for $1 \le i \le n$
- $r_n \epsilon F$

Recognizing languages

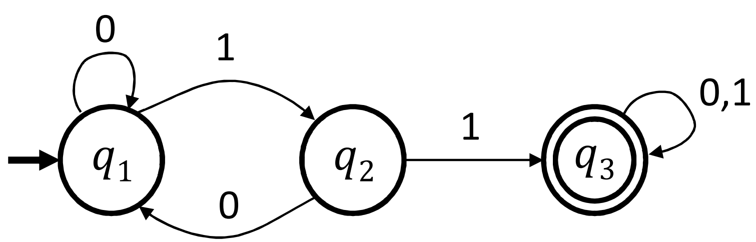
$L(M)$ = $\{w|\ M$ accepts $w\}$
$L(M)$ is the language of M
M recognizes $L(M)$

Definition

A language is regular if some finite automaton recognizes it.

Regular Languages – Examples

M_1



$L(M_1) = \{w \mid w \text{ contains substring } 11\} = A$

<u>Therefore A is regular</u>

More examples:

Let $B = \{w \mid w$ has an even number of 1s$\}$
B is regular

Let $C = \{w \mid w$ has equal numbers of 0s and 1s$\}$
C is <u>not</u> regular

Regular Expressions

Let A,B be languages:

Union: $A \cup B = \{w \mid w \in A \text{ or } w \in B\}$

Concatenation: $A \circ B = \{xy \mid x \in A \text{ and } y \in B\} = AB$

Star: $A^* = \{x_1 \ldots x_k \mid \text{each } x_i \in A \text{ for } k \geq 0\}$

Note: $\varepsilon \in A^*$ always

Example Let A={good, bad} and B={boy, girl}.

$A \cup B$= {good, bad, boy, girl}

$A \circ B = AB$= {goodboy, goodgirl, badboy, badgirl}

A^*= {ε, good, bad, goodgood, goodbad, badgood,
badbad, goodgoodgood, goodgoodbad, … }

Finite Automata equivalent to Regular Expressions

Regular expressions

Built from Σ, members Σ, $\emptyset$, ε [Atomic]

By using $\cup$, $\circ$, $*$ [Composite]

Examples:

$(0 \cup 1)^* = \Sigma^*$ gives all strings over Σ

$\Sigma^* 1$ gives all strings that end with 1

$\Sigma^* 11 \Sigma^*$ = all strings that contain 11 = $L(M_1)$

Closure Properties for Regular Languages

Theorem: If r_1, r_2 are regular languages, so is $r_1 \cup r_2$ (closure under ∪)

Proof: Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, A_1)$ recognize r_1
$M_2 = (Q_2, \Sigma, \delta_2, q_2, A_2)$ recognize r_2

Construct $M = (Q, \Sigma, \delta, q_0, A)$ recognizing $r_1 \cup r_2$
M should accept input w if either M_1 or M_2 accept w.

Components of $\boldsymbol{M}$:

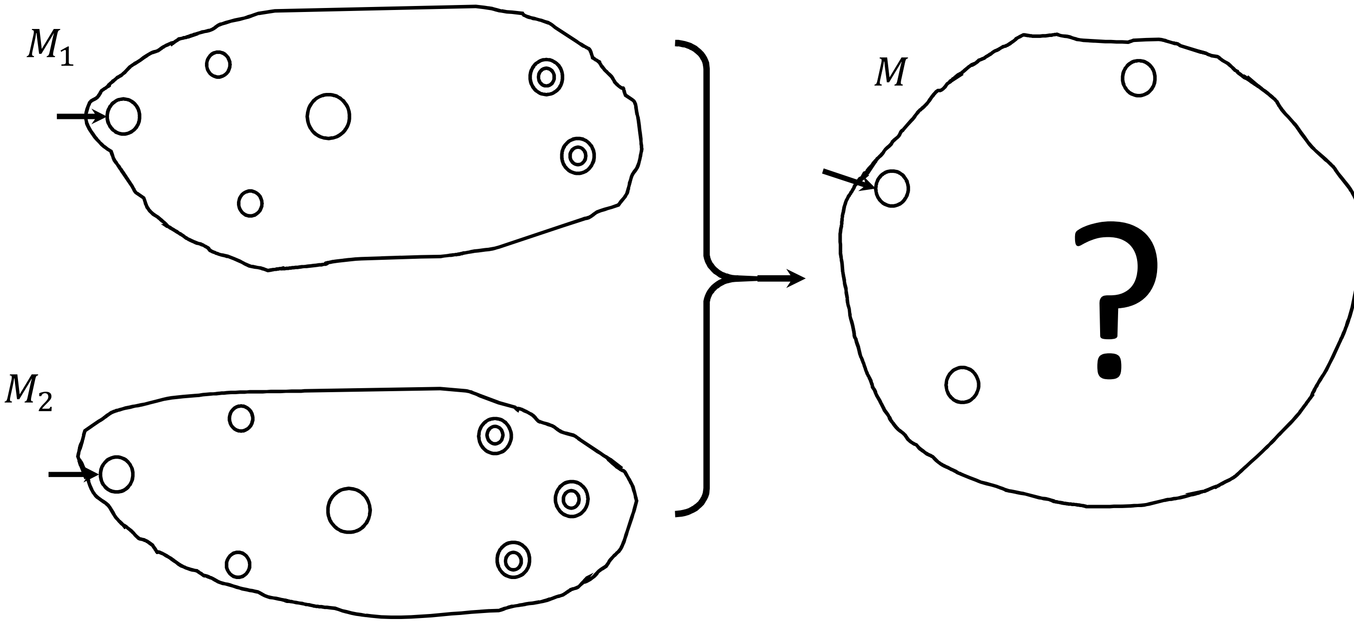
$Q = Q_1 \times Q_2$
$= \{(q_1, q_2) \mid q_1 \in Q_1 \text{ and } q_2 \in Q_2\}$

$q_0 = (q_1, q_2)$

$\delta((q,r),a) = (\delta_1(q,a), \delta_2(r,a))$

~~$A = A_1 \times A_2$~~ gives no intersection

$A = (A_1 \times Q_2) \cup (Q_1 \times A_2)$

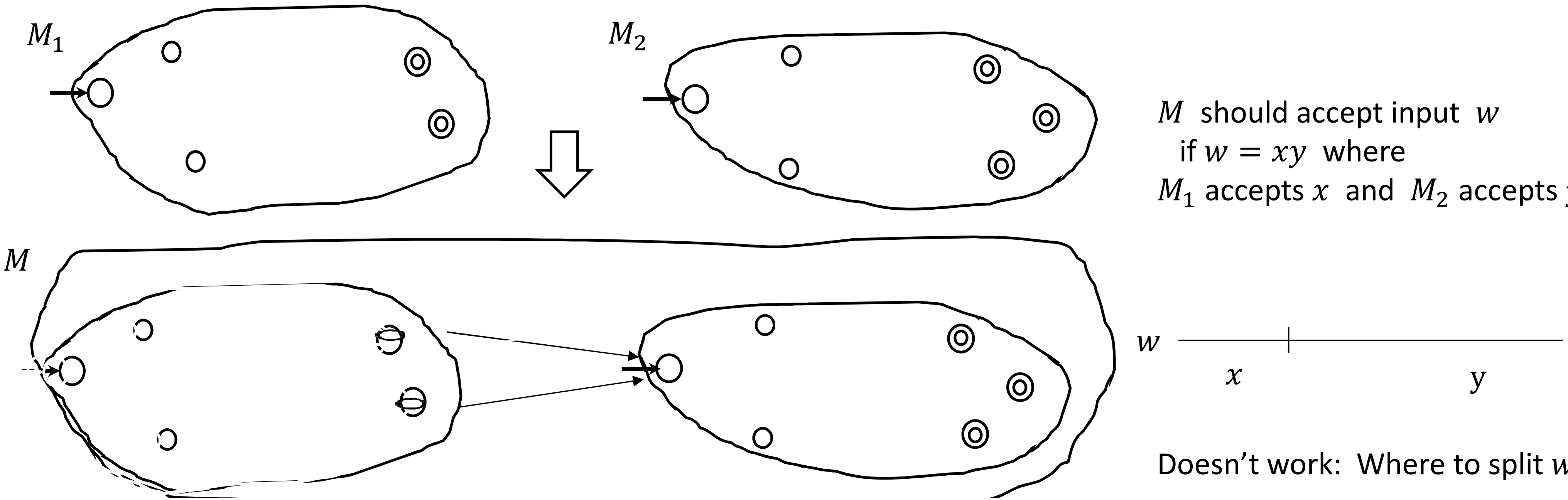


Closure Properties for Regular Languages

Theorem: If r_1, r_2 are regular languages, so is $r_1 r_2$ (closure under $\circ$)

Proof: Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, A_1)$ recognize r_1
$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize r_2

Construct $M = (Q, \Sigma, \delta, q_0, F)$ recognizing $r_1 r_2$



M should accept input w
if $w = xy$ where
M_1 accepts x and M_2 accepts y.

Doesn't work: Where to split w?