Theory of computation

Lecture 1: Automata and Regular languages

Adnane Saoud

Monday, September 2, 2024

UM6P | College of Computing

About Me

- Engineering degree (École Mohammadia d’ingénieurs)
- Ph.D. in Control theory and Computer science at CentraleSupélec and ENS Paris-Saclay, France (2019)
- Postdoc researcher at UC Santa Cruz, USA (2020)
- Postdoc researcher at UC Berkeley, USA (2021)
- Professor at CentraleSupélec (Sep 2021 – Aug 2022)
- Professor at UM6P-CC (Since Sept 2023)

Research interests: Control theory, machine learning, computer science

The Team

Instructor: Adnane Saoud

Office: SHBM, College of Computing offices

Office hours: I will be available Wednesday and Thursday (between 16:00 and 18:00)

Or, send me an email and we will find a good time to meet

Email: adnane.saoud@um6p.ma

TA: Emmanuel Junior WAFO WEMBE (Ph.D. student, Control of monotone dynamical systems)

Email: EmmanuelJunior.WafoWembe@um6p.ma

TA: Sadek Belamfedel Alaoui (Postdoctoral researcher, Reinforcement learning-based control)

Email: sadek.belamfedel@um6p.ma

What is computation theory ?

Introduction

Computability Theory

- What is computable or not (problems that are solvable or not)
- Examples: program verification, mathematical truth
- Models of Computation:
  Finite automata: used in text processing, compilers, control of systems context-free grammar: used in programming languages and artificial intelligence

Complexity Theory

- What is computable in practice?
- What makes some problems computationally easy or hard
- Example: sorting and factoring problems
- P versus NP problem

About this course

Learning Objectives

The main learning objectives of the course are as follows:

- <u>Formal Languages:</u> Learn about formal languages, their syntax, and semantics. Describe and recognize different types of formal languages such as regular languages and context-free languages.

- <u>Automata Theory:</u> Learn the fundamentals of automata theory such as finite automata, pushdown automata, and Turing machines. Learn how to design these machines, analyze their behavior, and understand their computational power.

- <u>Computability Theory:</u> Learn the concept of computability, the Church-Turing thesis, and the Halting problem. Explore the limits of what is computationally solvable and understand undecidability.

- <u>Complexity Theory:</u> Learn about computational complexity tools such as time and space complexity classes like P, NP. Be able to distinguish between problems that can be solved efficiently and those that are NP-complete or harder.

- <u>Problem Solving and Proof Techniques:</u> Develop problem-solving skills by learning how to construct proofs. Be able to prove whether a problem is decidable or undecidable.

- <u>Practical Applications:</u> While the primary focus is on theory, the course provides insights into practical applications, demonstrating how theoretical concepts relate to real-world computational problems.

- <u>Preparation for Advanced Studies:</u> The course serves as a foundational course to pursue more advanced studies in computer science, in areas related to formal methods, automata theory.

Structure of the Course

Five components:

1. Lectures
   - Thursday's (and sometimes also Monday)

2. Tutorials
   - Monday (First session)

   Guided exercises sessions with the instructor

3. Labs
   - Thursdays (Second session)

   Hands-on experience on computation theory tools

   Labs will not be graded, but will help you to further understand the material presented in the lectures

   Final Labs will be graded

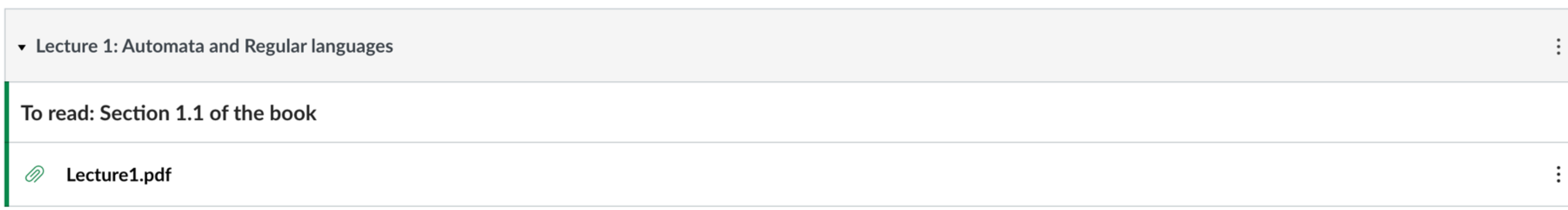
4. Recitations
   - Thursdays (Second session)

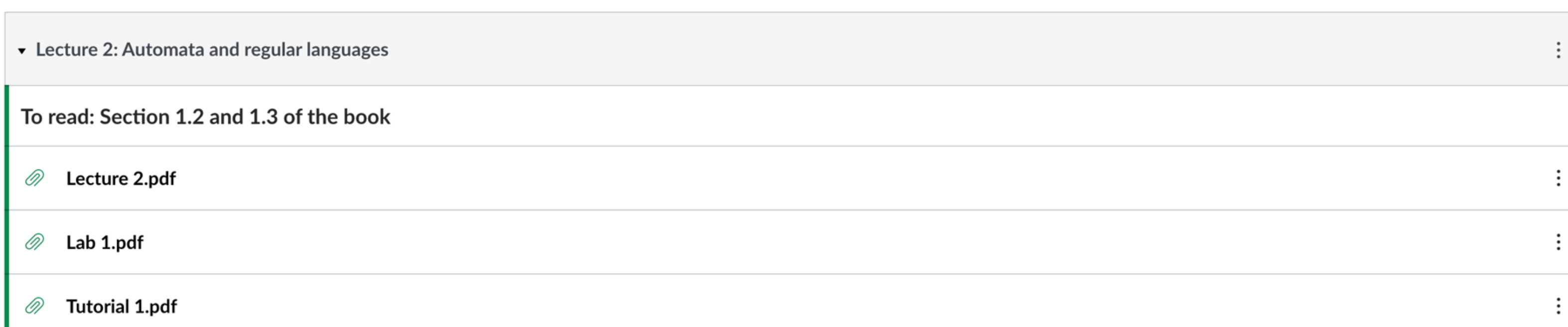
   Open exercises sessions for discussions and for answering questions

5. Project
   - Thursdays (Second session)

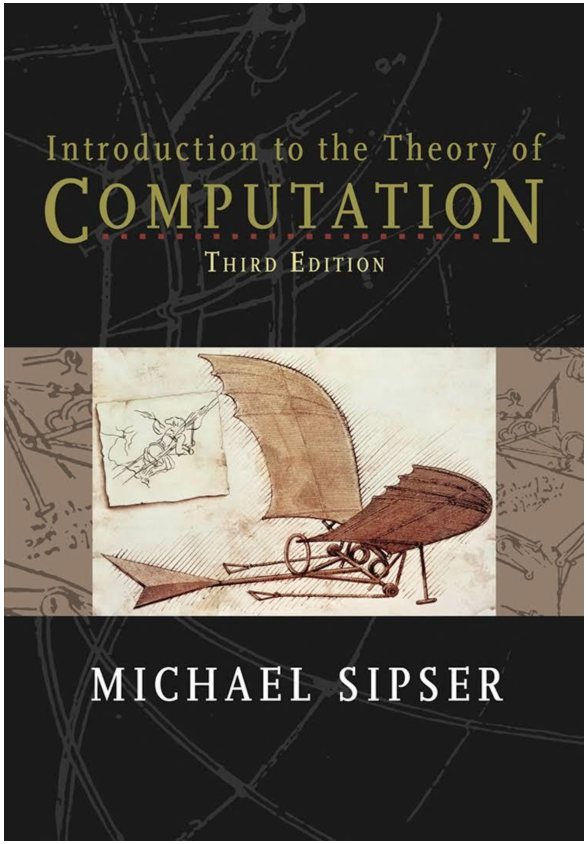
Course Logistics

- Canvas for Q&A and material
  - You will receive an email shortly providing a link the Canvas
  - All announcements will be posted there
  - Also, lecture slides, tutorials and labs
  - Please, participate and help each other!





Reading Material



- The book is publicly available in electronic form
  - Pointers to chapters for every lecture (see the canvas)

Some Personal Notes ☺

- Please ask questions, participate in discussions on Canvas
- Play with software tools. Apply what you've learnt in theory
  - This is the actual goal of the lab sessions and the recitations
- Give us your feedback!

Topics that will be covered

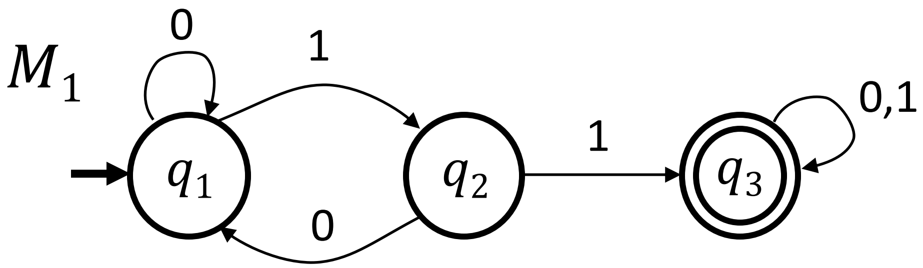
Idea about the Schedule (Subject to Change)

1. Introduction, Finite Automata, Regular Expressions
2. Nondeterminism, Closure Properties, from Regular Expressions to finite automata
3. The Regular Pumping Lemma, from Finite Automata to Regular Expressions, Context free grammar
4. Pushdown Automata, equivalence between pushdown automata and context free grammar
5. The Context free Pumping Lemma, Turing Machines
6. Turing machine Variants, the Church-Turing Thesis

…

Finite automata

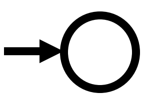
Finite Automata

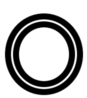


Input: finite string
Output: Accept or Reject

States: $q1\ q2\ q3$

Transitions: $\xrightarrow{1}$

Start state: 

Accept states: 

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

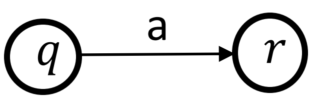
Examples: 01101 → Accept
00101 → Reject

$M1$ accepts exactly those strings in A where
$A = \{w |\ w \text{ contains substring } 11\}$.

A is the language of M_1 and that M_1 recognizes A and that $A = L(M_1)$.

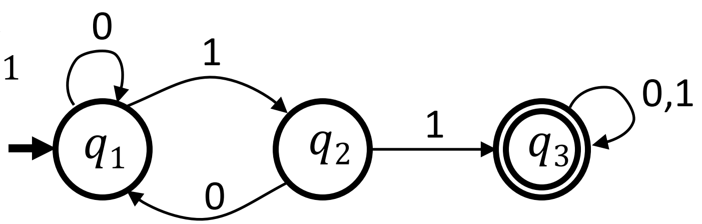
Finite Automata – Formal Definition

Definition: A finite automaton M is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$

- Q finite set of states
- Σ finite set of alphabet symbols
- δ transition function $\delta \colon Q \times \Sigma \rightarrow Q$ $\delta(q, a) = r$ means 

- q_0 start state
- F set of accept states

M_1



Example:

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

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

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

$F = \{q_3\}$

$\delta =$

	0	1
q_1	q_1	q_2
q_2	q_1	q_3
q_3	q_3	q_3

Regular languages

Finite Automata and Regular languages

Strings and languages

- A string is a finite sequence of symbols in Σ
- A language is a set of strings
- The empty string ε is the string of length 0
- The empty language $\emptyset$ is the set with no strings

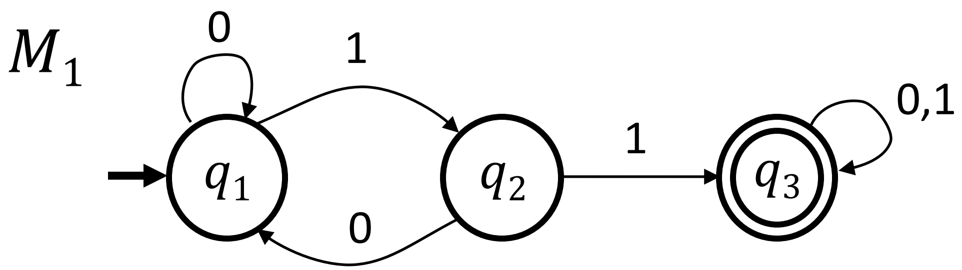
Definition: M accepts string $w = w_1 w_2 \ldots w_n$ each $w_i \epsilon \Sigma$ if there is a sequence of states $r_0, r_1, r_2, , \ldots, r_n \epsilon Q$ where:

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

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

Definition: A language is regular if some finite automaton recognizes it.

Regular Languages – Examples



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

Therefore A is regular

<u>Example of a non regular language:</u>

Let $C = \{w|\ w$ has equal numbers of 0s and 1s$\}$

C is <u>not</u> regular (will be proven in next lectures).

Regular Expressions

Regular operations. 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 $A = \{\text{ab,bc}\}$ and $B = \{\text{fg, hk}\}$.

- $A \cup B$
- $A \circ B = AB$
- A^*

Regular Expressions

Regular expressions

- Built from Σ, members $\Sigma, \emptyset, \varepsilon$ [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 $= L(M_1)$

Goal: Show that finite automata are equivalent to regular expressions

Closure properties

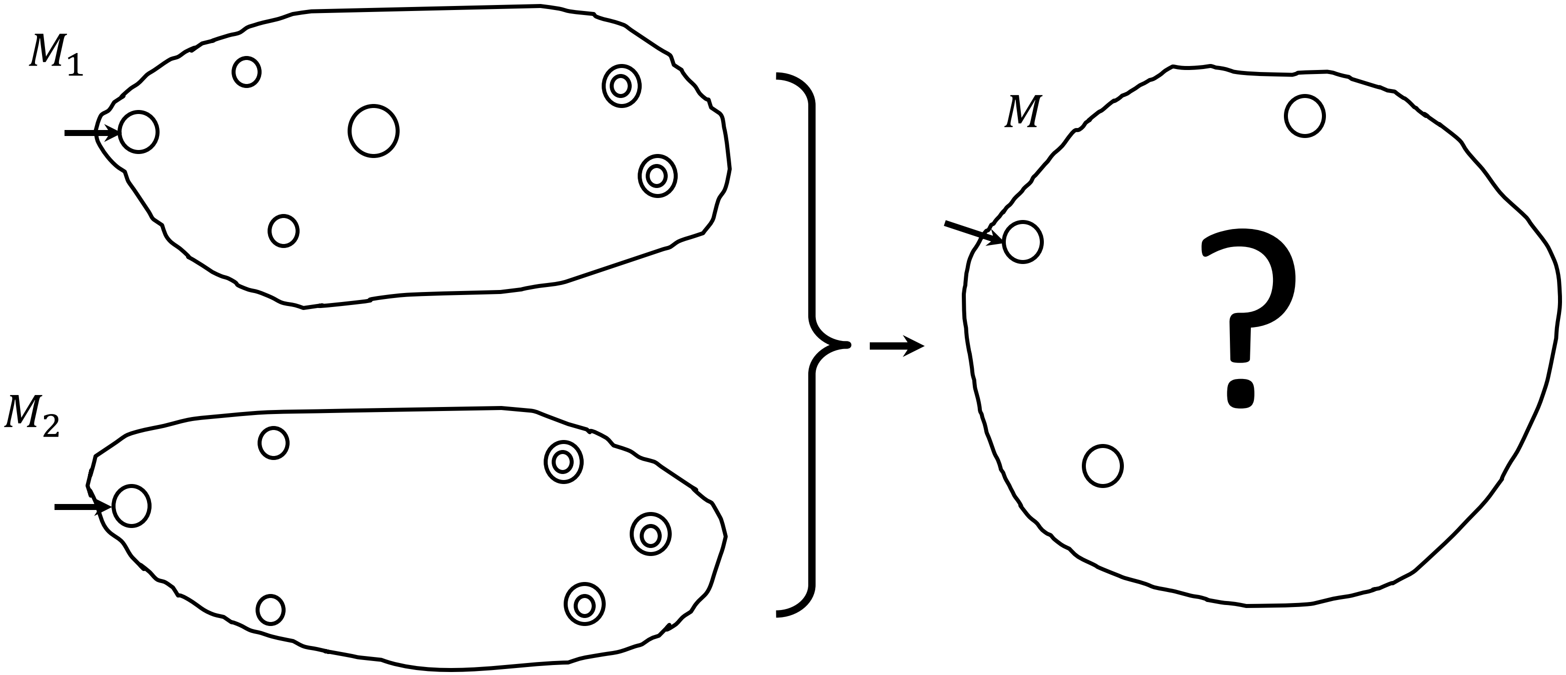
Closure Properties for Regular Languages

Theorem: If A_1, A_2 are regular languages, so is $A_1 \cup A_2$ (closure under $\cup$)

Proof: Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1

$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

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



Closure Properties for Regular Languages

Theorem: If A_1, A_2 are regular languages, so is $A_1 \cup A_2$ (closure under $\cup$)

Proof: Let $M_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ recognize A_1

$M_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$

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

Components of $\boldsymbol{M}$:

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

$$q_0 = (q_{01}, q_{02})$$

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

$$F = \cancel{F_1 \times F_2}$$ when we need this?

$$F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$$

Theorem: If A_1, A_2 are regular languages, so is $A_1 \cap A_2$

Summary

Summary

1. Introduction, outline of the course
2. Finite Automata and regular languages: formal definition
3. Regular Operations and Regular Expressions
4. Theorem: Class of regular languages is closed under union and intersection