Theoretical Computer Science

Overture

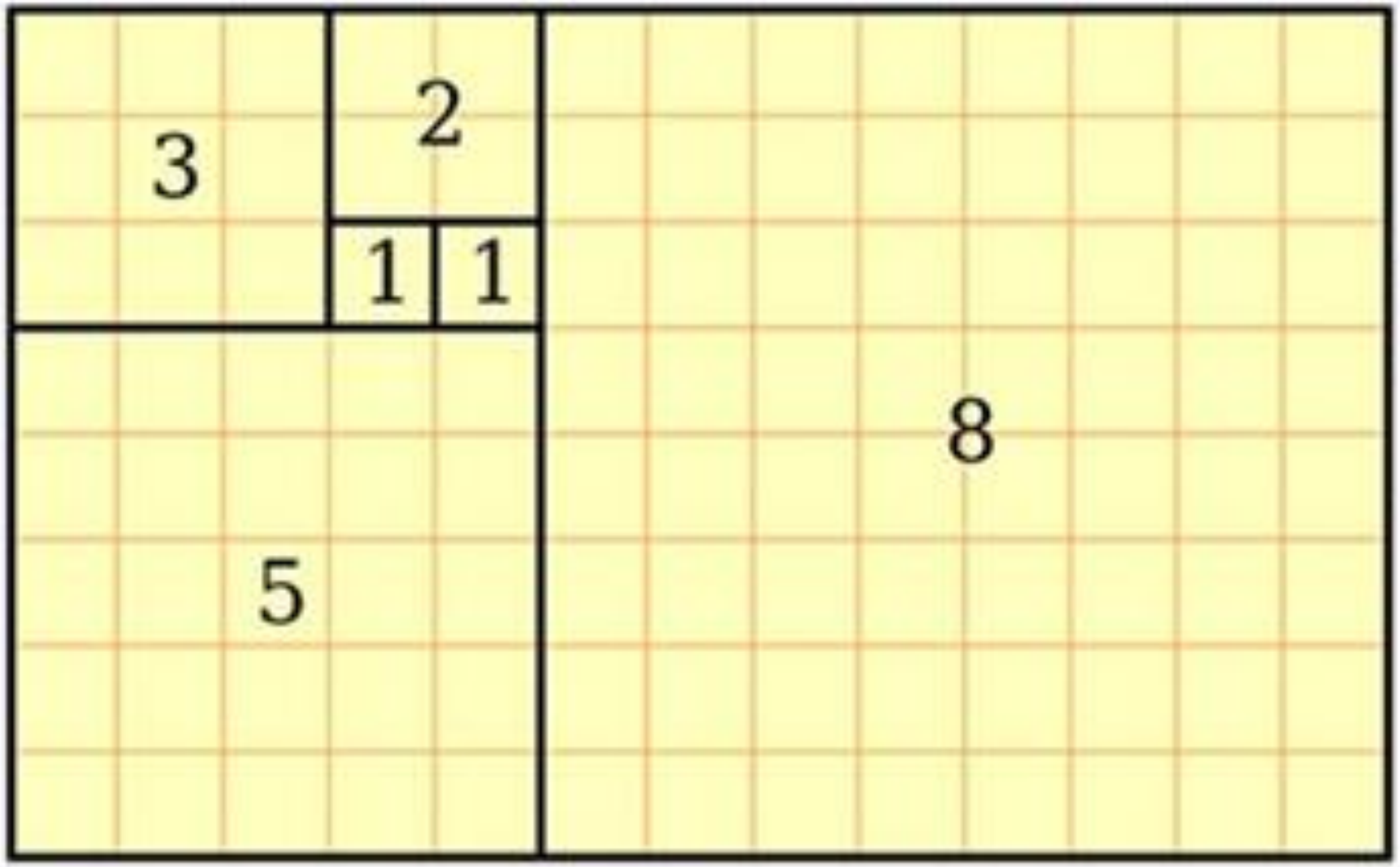
Lecture 1 - Manuel Mazzara

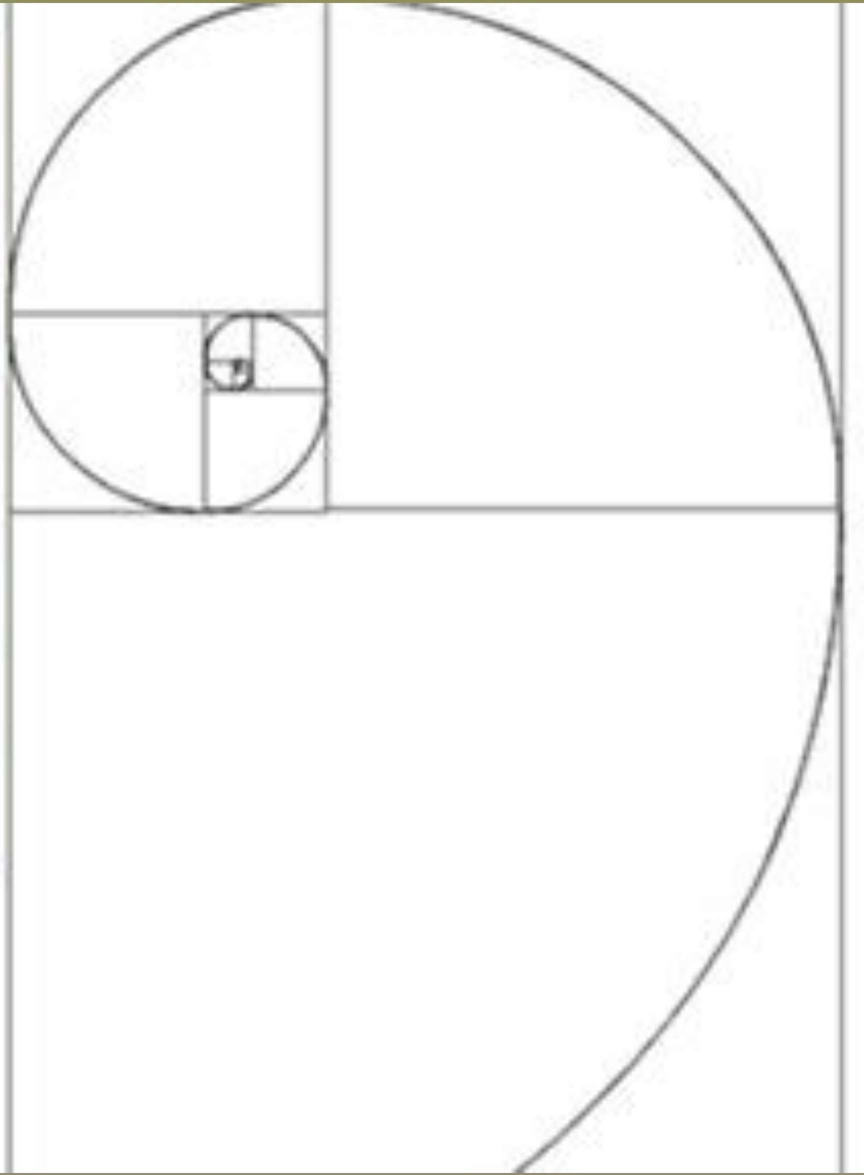
Golden ratio and Fibonacci

Φφ

- Golden ratio (“phi”)
  - Approximately 1.618
  - **Irrational number** like pi
  - Its decimal digits continue forever without repeating a pattern

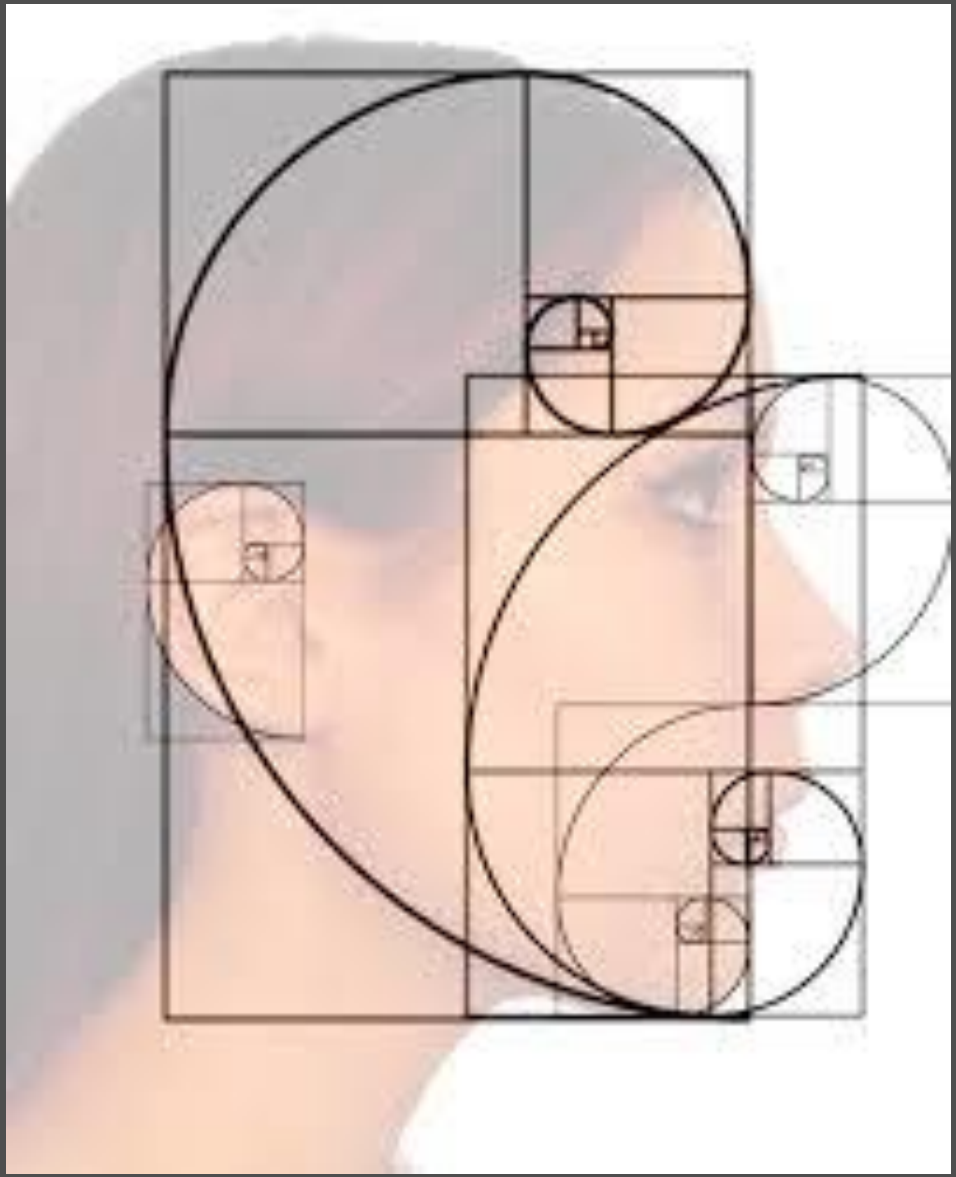
- Fibonacci sequence: 1, 1, 2, 3, 5, 8, 13, 21, ...
  - Dividing each number in the Fibonacci sequence by the previous one, **the resulting sequence converges to phi**



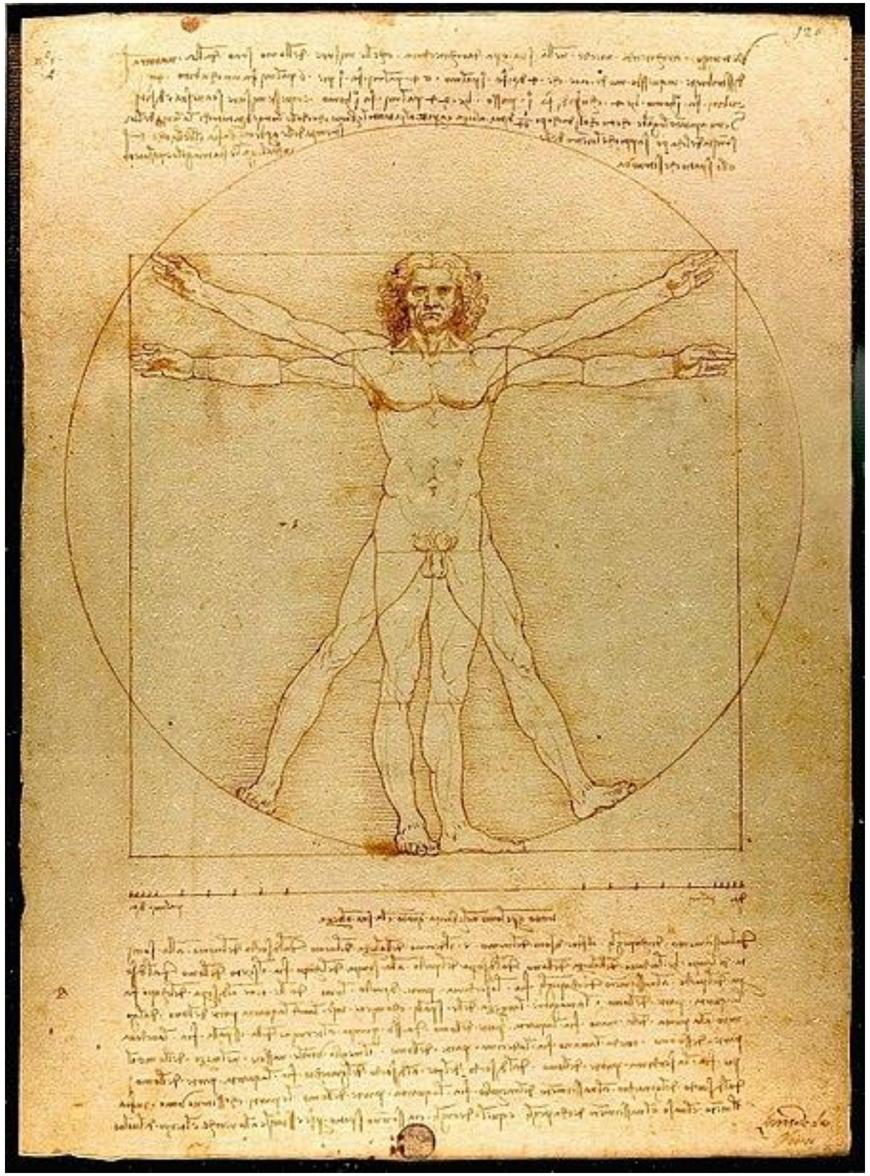


The Golden Rectangle forms
the Golden Spiral


Fibonacci sequence in our hand allows for it to form a perfect curl when we clench our fist.
2
3
5
8
FUNsubstance
Kill your time at FUNsubstance.com



The Bramante Staircase of the Vatican Museums, designed by Giuseppe Momo in 1932



Sacred Geometry

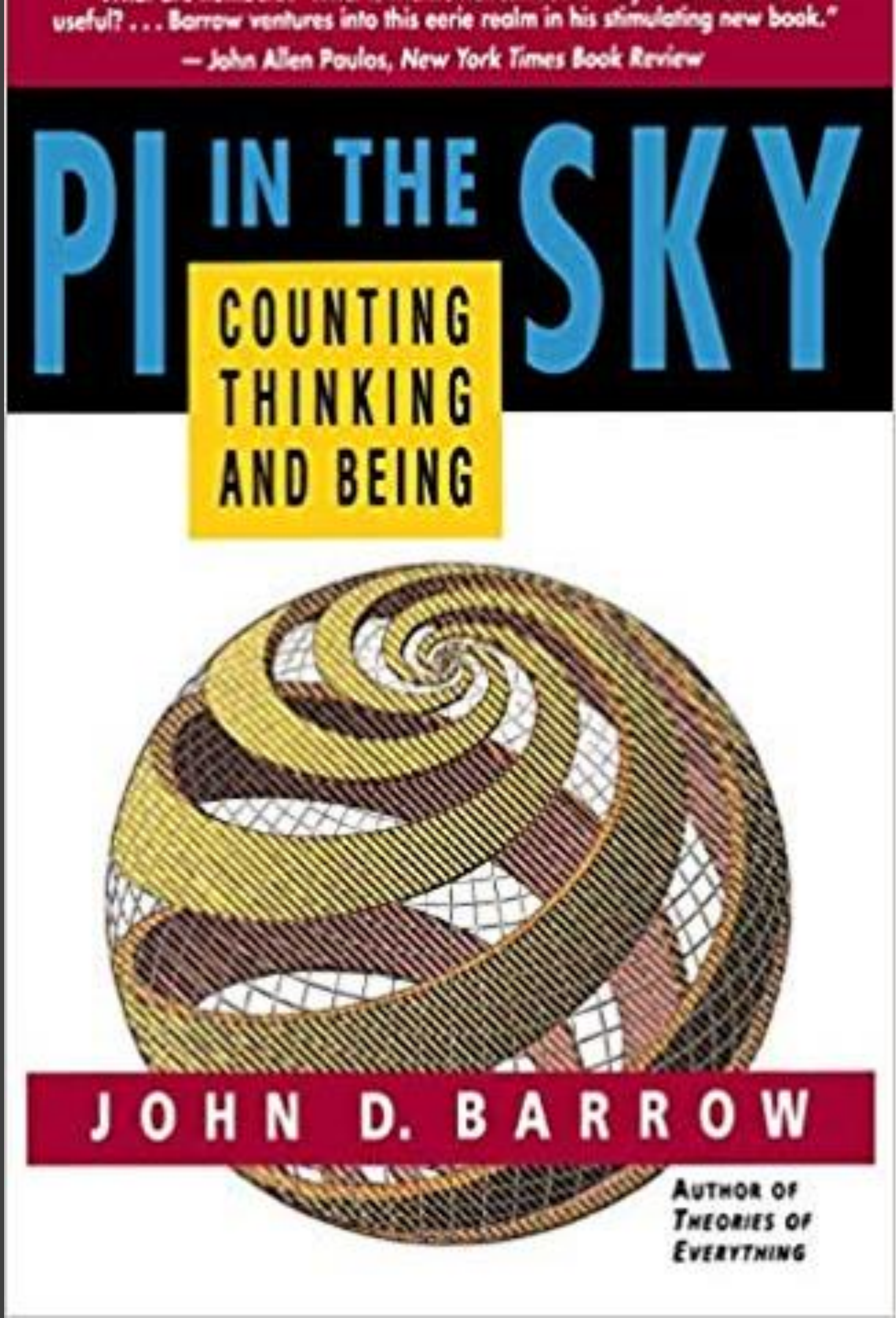
- Medieval European cathedrals
  - based on geometries to show the world through mathematics
  - gain a better understanding of the divine
- Leon Battista Alberti
  - *De re aedificatoria* (**On the Art of Building**) written in 1443-1452
  - It describes the ideal church in terms of spiritual geometry
  - **Alberti cipher**
- Vitruvian Man
  - By **Leonardo da Vinci** (about 1490)
  - It represents sacred geometry principles of the human body

Alberti cipher (1467)

Pi in the Ski

Counting, Thinking, and Being

– John Barrow



“A large part of mathematics which becomes useful developed with absolutely no desire to be useful, and in a situation where nobody could possibly know in what area it would become useful; and there were no general indications that it ever would be so.”

"The Role of Mathematics in the Sciences and in Society" (1954) an address to Princeton alumni, published in *John von Neumann : Collected Works* (1963) edited by A. H. Taub

"By and large it is uniformly true in mathematics that there is a time lapse between a mathematical discovery and the moment when it is useful; and that this lapse of time can be anything from 30 to 100 years, in some cases even more; and that the whole system seems to function without any direction, without any reference to usefulness, and without any desire to do things which are useful.

"The Role of Mathematics in the Sciences and in Society" (1954) an address to Princeton alumni, published in *John von Neumann : Collected Works* (1963) edited by A. H. Taub

RSA

bitcoin

Has BTC anything to do with us?

- Cryptographic schemes in practical use today, like the RSA, are based on the hardness of **number theoretic problems**
- Public key cryptography suffered a foundational crisis in 1994
  - Every number theoretic problem used in cryptography could be efficiently solved by quantum computers
- Will every cryptographic scheme based on number theoretic problems will be rendered insecure once quantum computers become reality?


P=NP?

Computability vs Complexity

- **Computability** is about what *can* be computed
- **Complexity** is about *how efficiently* can it be computed
- You *should* have attended courses on computability, models of computation, algorithms and their complexity during your BS studies

$$\mathcal{O}(n \log n)$$

$$\aleph_0 < \aleph_1$$

$$\aleph_1 = 2^{\aleph_0}$$

$$\aleph_0$$

Do you recognize any of these symbols?

$$\mathcal{O}(n \log n)$$

Sorting Algorithms?

A little secret to start with ☺

- Quora is my secret place to look at common students' questions ☺

- **<u>What is the benefit of studying theory of computation?</u>**

  - What do I gain by studying TOC? Does it create a foundation for other CS courses? (Other than compiler design)
  - https://www.quora.com/What-is-the-benefit-of-studying-theory-of-computation

More questions

- <u>**What kind of math is involved in Theory of Computation?**</u>
  - I would like to self-study the Theory of Computation over the summer because it sounds like a very interesting subject. I am told that it is heavily based on mathematics, so I would first like to review my fundamentals before diving in.
  - https://www.quora.com/What-kind-of-math-is-involved-in-Theory-of-Computation
- At the end of this course you have to able to answer such questions, and maybe help your colleagues ☺

Play and find questions on Complexity Theory and Big O notation

Theoretical Computer Science

Course Concepts and Structure

Lecture 1 - Manuel Mazzara

Moodle

- Information will be ***all*** on Moodle
  - **[S21] Theoretical Computer Science**

- There you will find:
  - the lecture material, just after the class (sometime before)
  - Tutorials and the lab sessions with exercises

- Plus any other information and all your grades

Prerequisite courses

- Logic
- Discrete Math
- Calculus I
- Algorithms and Data Structures is a plus

- If you have not attended or do not remember need to catch up a bit
- We will try to keep things as simple as possible
- **<u>We will go through the math foundation again</u>**

Team

- **Manuel Mazzara (m.mazzara@innopolis.ru) – classes**
- **Mansur Khazeev (m.khazeev@innopolis.ru) – tutorials**
- **Timur Fayzrakhmanov (t.fayzrakhmanov@innopolis.ru) – labs**
- **Robiul Islam (r.islam@innopolis.university) - labs**
- **Swati Megha (s.megha@innopolis.university) - labs**
- **Mariya Naumcheva (m.naumcheva@innopolis.university) – labs**

Lectures and labs

- Lectures are every **Thursday** at 9:30am-11:00 am (like today!)
  - **Room 108 ☺**
- Tutorials and lab sessions are on **Thursdays** at:
  - **Tutorial 11:10-12:40**
    - **Mansur Khazeev: all, room 108**
  - **Labs 13:10-14:40**
    - **Mariya Naumcheva: B20-02, room 313**
    - **Timur Fayzrakhmanov: B20-03, room 303**
    - **Swati Megha: B20-06, room 314**
  - **Labs 14:50 – 16:20**
    - **Mariya Naumcheva: B20-01, room 303**
    - **Timur Fayzrakhmanov: B20-04, room 421 (online)**
    - **Swati Megha: B20-05, room 314**
    - **Robiul Islam: B20-07, room 301 (online)**

Please note!

- **Tutorials and Lab sessions start on 28.1.2020** (next week)
- Tutorials and labs generally cover the topics presented in the previous week
- **Be careful, rooms occasionally may change – time and instructors should not not ☺**

Course Delivery

- Two weekly academic hours of lectures
- Two weekly academic hours of tutorials
- Two weekly academic hours of exercise sessions
- For fourteen weeks starting from January 21, 2021
- There is a mid-term exam, a final examination and three assignments

Self-study

- Overall the course should take on average **10 hours per week** of your life ☺
  - 150 hours (6 credits) over 14 weeks, i.e. about 10+ hours per week
- 6 hours are of classes, tutorials and labs
- The amount of study for the course is estimated in about 4 hours/week
- These are indeed just numbers…

What will you learn (among other things)?

- **Mathematical foundations** necessary to study **compilers** functioning
- Some **History** of computing and its theory and **major personalities**
- **Limits** of computation, i.e. **what computers cannot do**
- As a consequence we will understand **what computers can do**

Course Syllabus

- Language properties, operations on languages, Kleene operator, language cardinality
- Finite state automata and pushdown automata
- Relationships between automata and languages, pumping lemma, closure of regular languages
- Regular expressions and regular languages, regular languages and decidability
- Formal Grammars: Chomsky grammars and productions
- Turing Machines and universal Turing Machines
- Halting problem , Turing computability, Rice's theorem

Required background knowledge

- The course is intended to be self-contained, requiring basic knowledge of **logic**, **set theory** and **discrete mathematics**

- **Prerequisite courses: i**t is recommended to pass **Discrete Math/logic** and **Calculus I** course before Theoretical Computer Science to be familiar with the basic mathematical machinery. Previous or concurrent attendance of **Algorithms and Data Structures** is a plus

Textbooks and suggested readings

- J.E.Hopcroft and J.D.Ullman. Introduction to Automata Theory, Languages, and Computation. Addsion Wesley (1979).
- M. Davis, R. Sigal and E.J. Weyuker. Computability, complexity, and languages: fundamentals of theoretical computer science. 2nd ed., Academic Press (1994).
- J. Hromkovic. Algorithmic Adventures: From Knowledge to Magic. Springer (2009)

Assessment

- Mid-term Exam (25%), Final Exam (25%) assignments (45%) and participation (5%, +5 extra points)

- There will be three assignments

- Exact deadlines will appear soon
  - The first one should be at week 3-4
  - The second one at about week 7-8
  - The third one at about week 11-12

Why studying TOC?

- A good software developer ignorant of how the **mechanics** of a compiler works is not better than a good pilot when it comes to fix the engine
- He will definitively not be able to provide more than **average solutions** to the problems he is employed to solve
- Like automotive engineering teaches us, races can only be won by the right synergy of a good **driving style and mechanics**
- **Limits of computation cannot be ignored** in the same way we precisely know how accelerations, forces and frictions prevent us from racing at an unlimited speed

What the course will be about?

- Prerequisites to understand **compilers functioning**
- The act of compilation appears **deceptively simple**
- Great minds and results are behind this major achievement
- All starts with the **Epimenides** paradox (about 600 BC), which emphasizes a problem of self-reference in logic
- In the short time window between WWI and WW2 **Alan Turing** proved that a general procedure to identify algorithm termination does not exist (and cannot exist)
- **Noam Chomsky** in 1956 described a hierarchy of grammars

Theoretical Computer Science

Models of Computation

Lecture 1 - Manuel Mazzara

Machines and Grammars

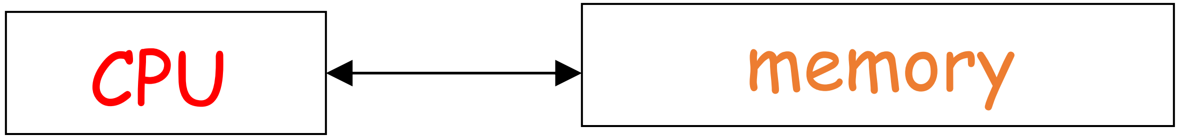
1. Computation is elegantly modeled trough simple mathematical objects
   - Finite automata, pushdown automata, Turing machines, ...

2. Methods of generating languages: regular expressions, grammars…

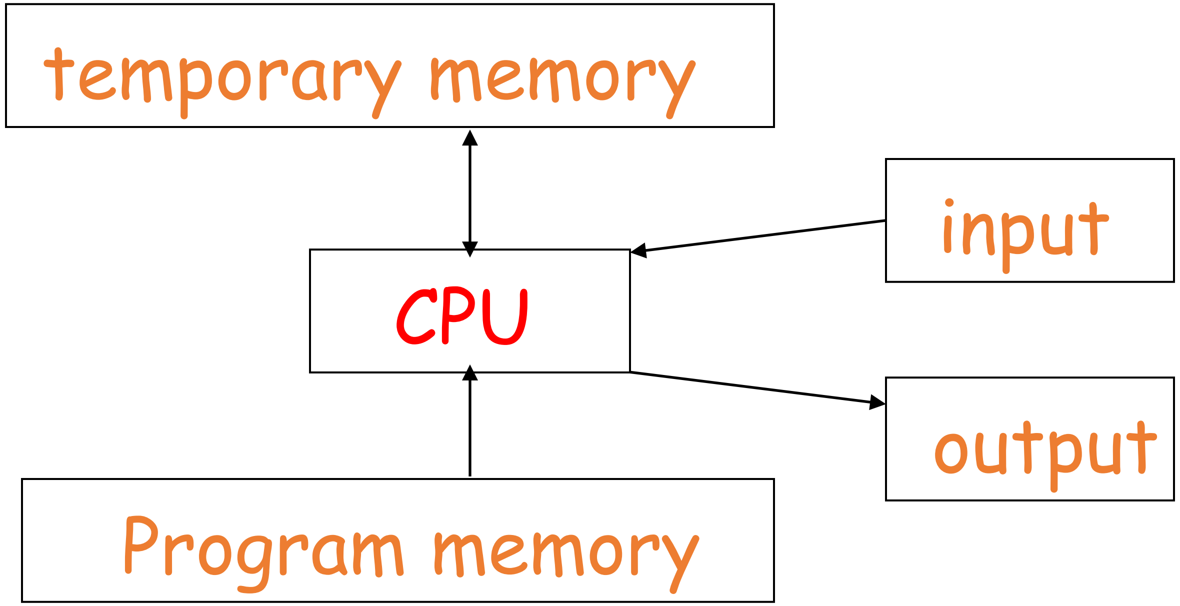
3. Computability theory

Models of Computation

Computation

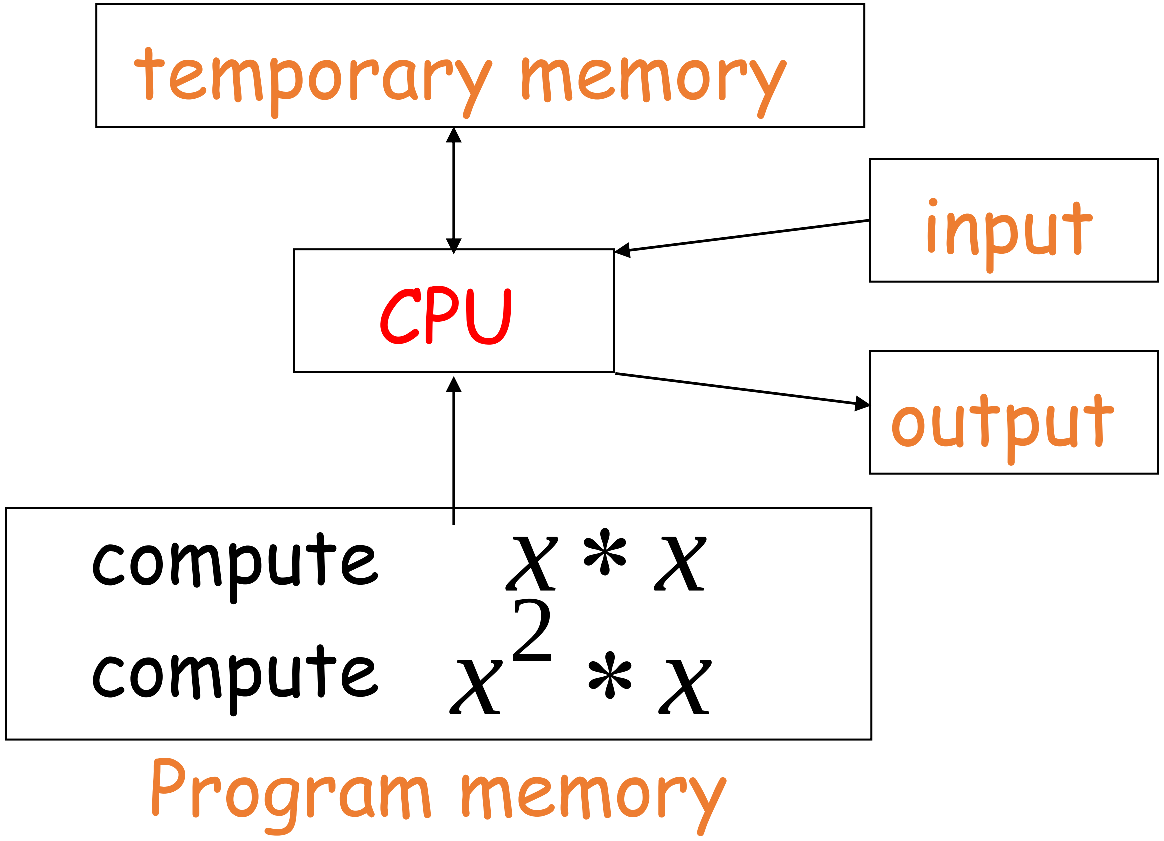


CPU, Memory, I/O



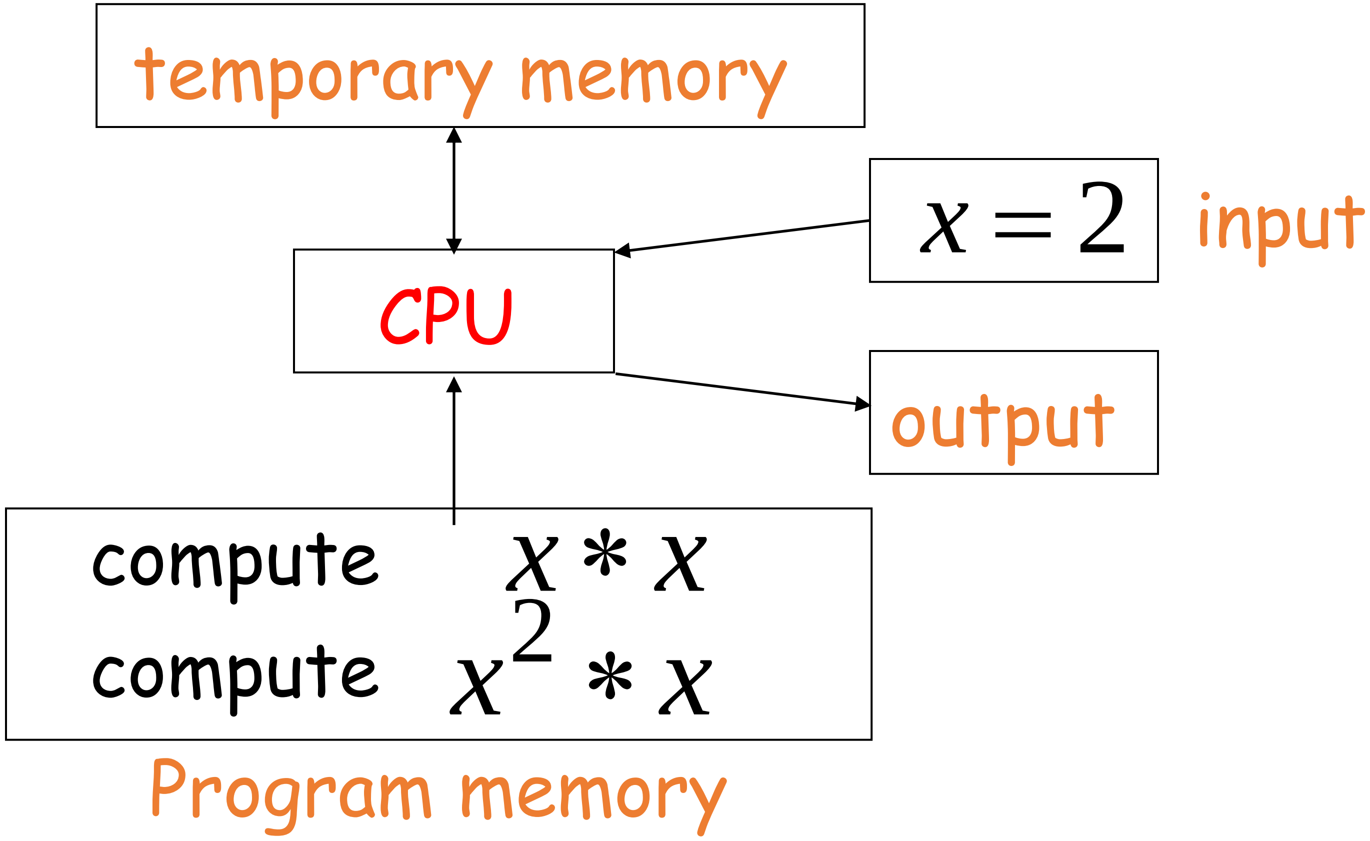
Example (1)

$$f(x) = x^3$$



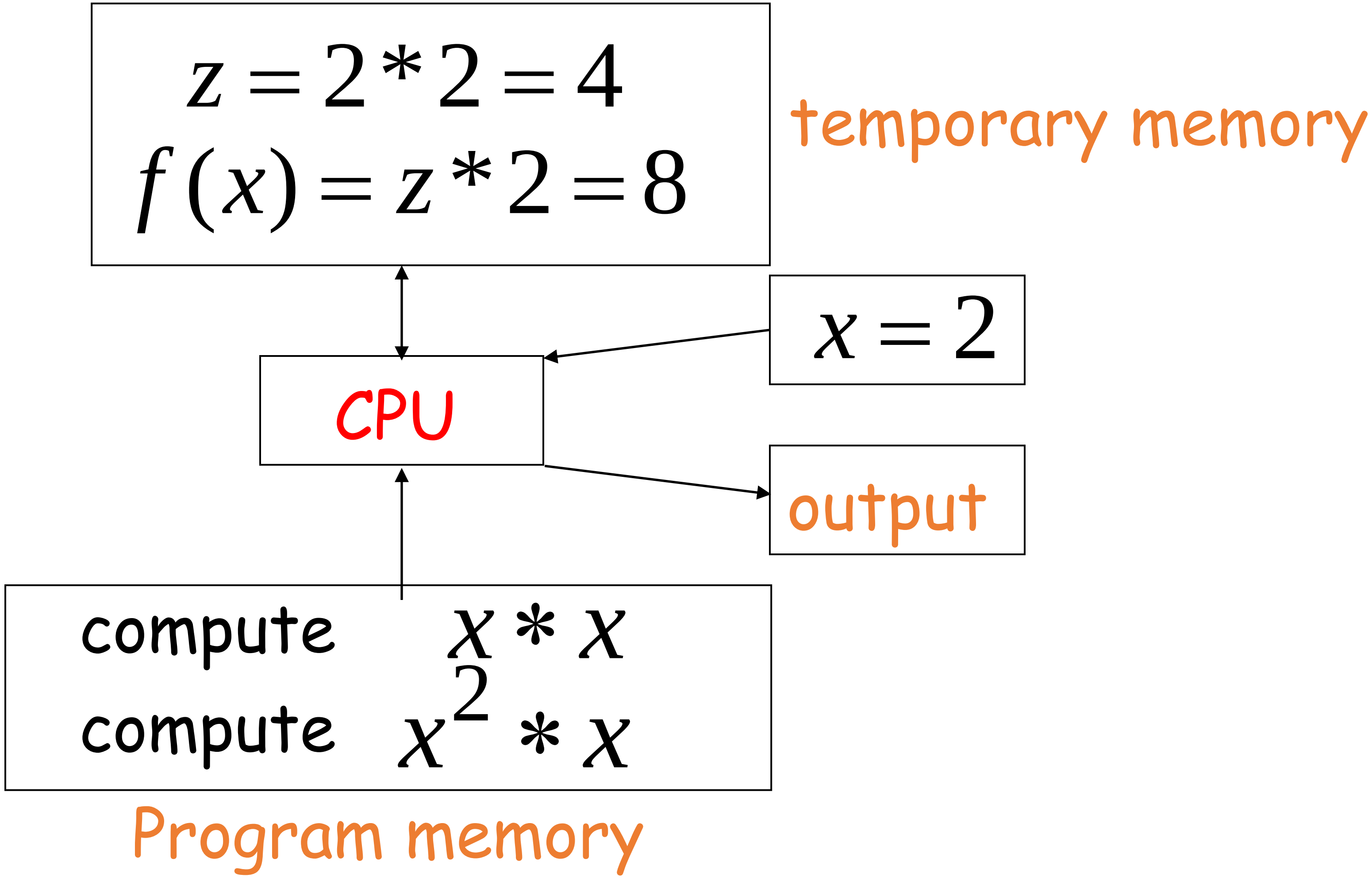
Example (2)

$$f(x) = x^3$$



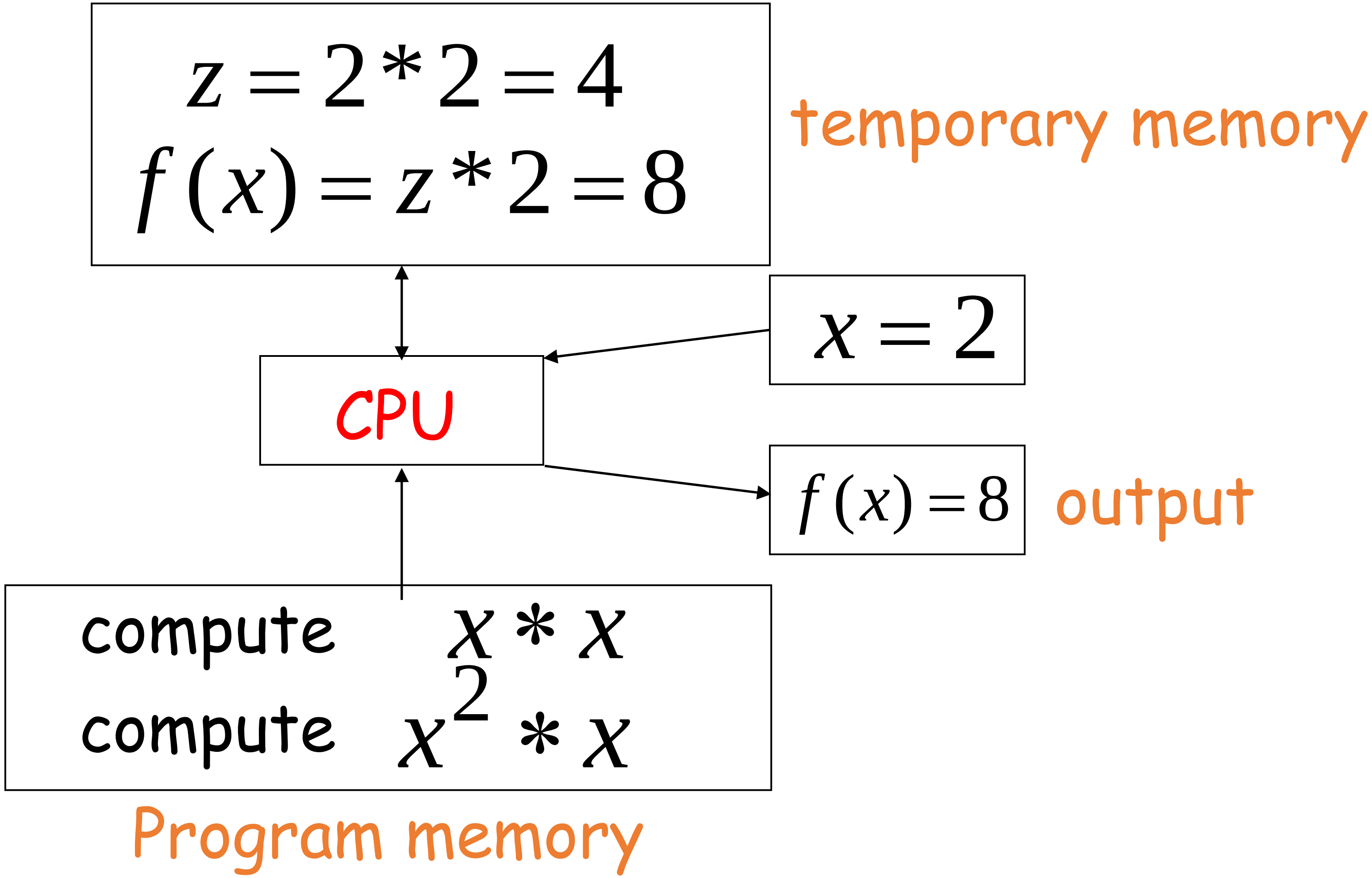
Example (3)

$$f(x) = x^3$$

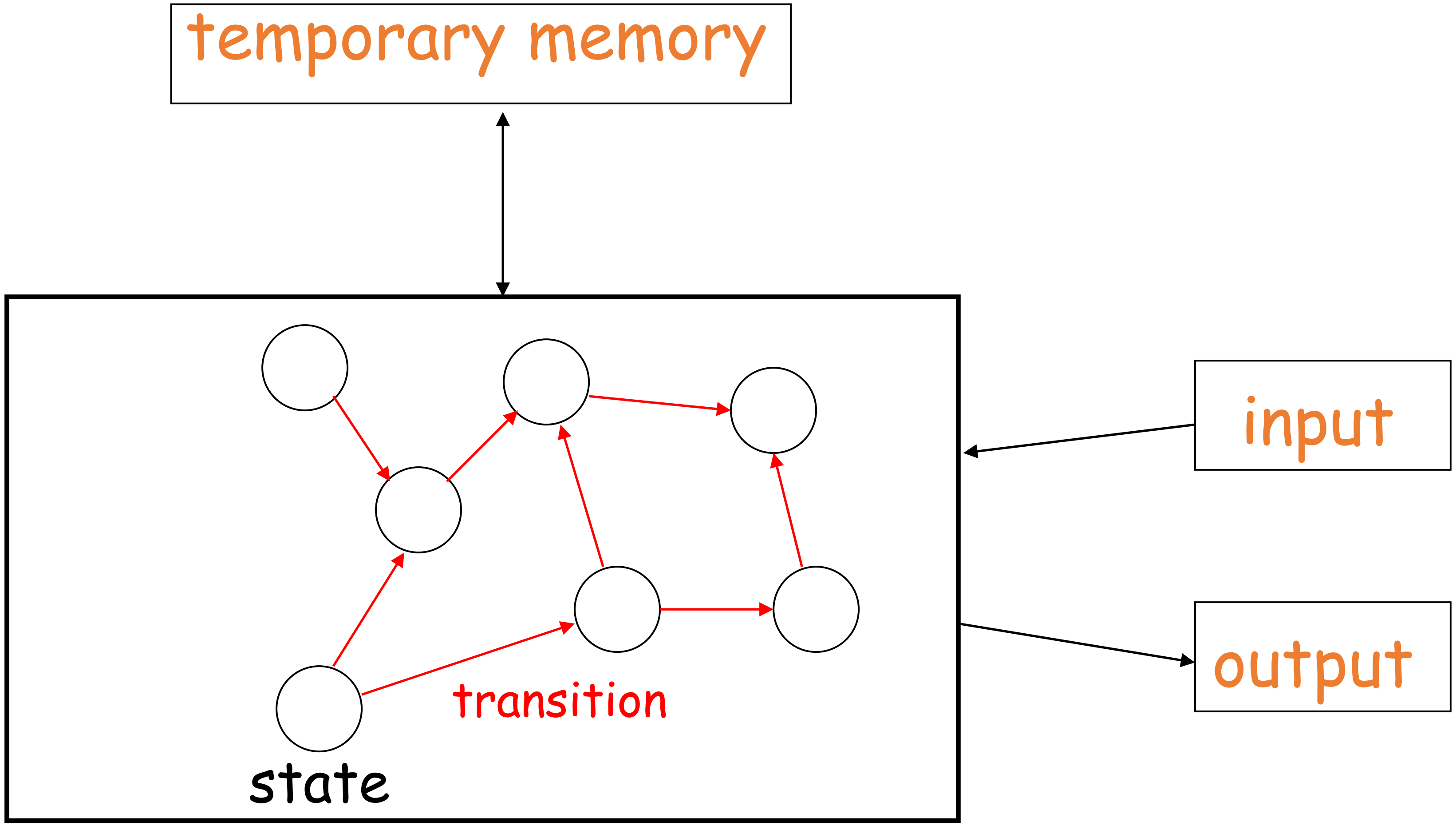


Example (4)

$$f(x) = x^3$$



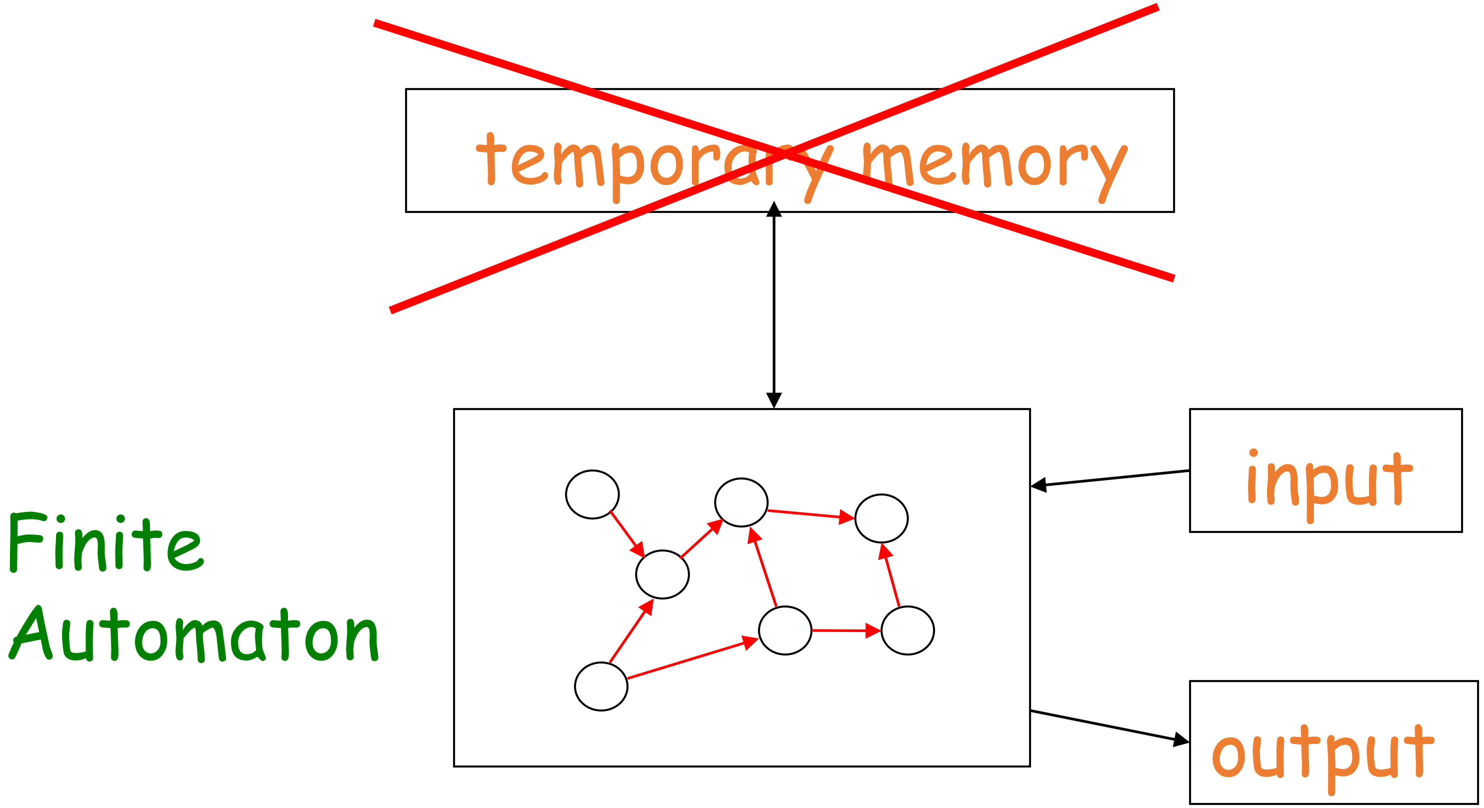
Automaton



Different kind of Automata

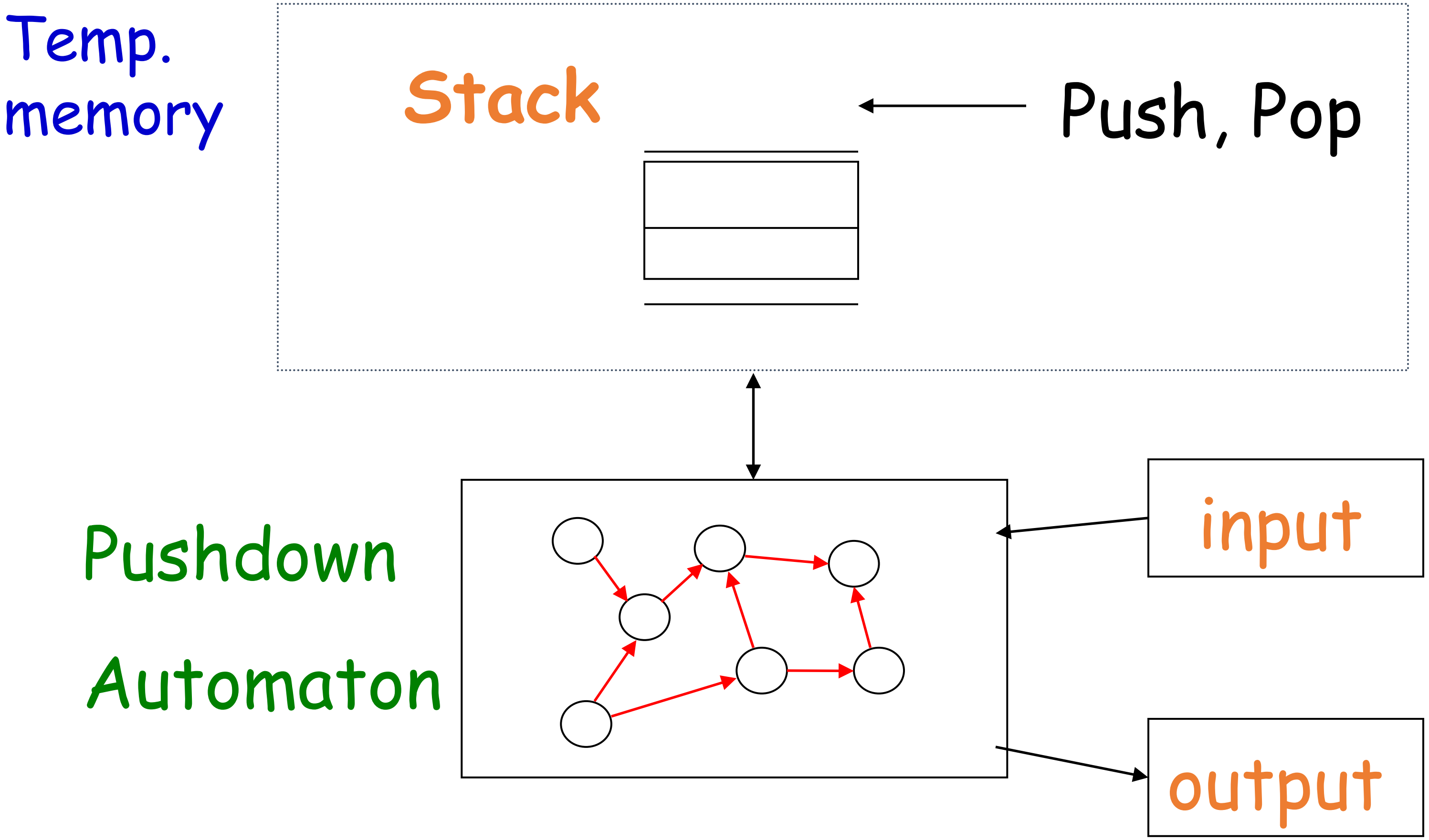
- Finite State Automata (FSA): no temporary memory
- Pushdown Automata (PDA): stack (destructive memory)
- Turing Machines (TMs) : random (**non-sequential**) access memory

FSA



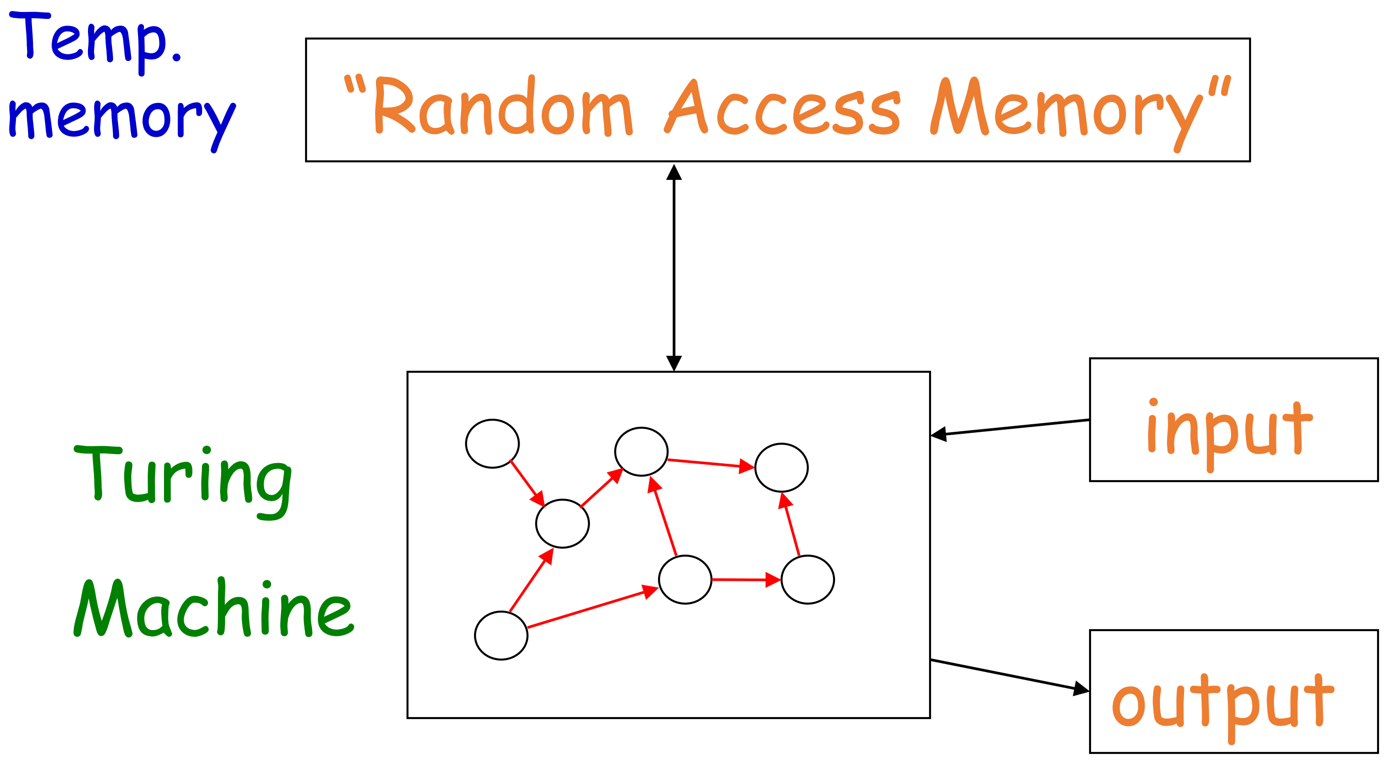
Example: Elevators, Vending Machines   (“small” computing power)

PDA



Example: Compilers for Programming Languages (“medium” computing power)

TM



Any Algorithm ("highest" known computing power)

Power of Automata

Simple problems	More complex problems	Hardest problems
Finite Automata $<$	Pushdown Automata $<$	Turing Machine

Less power → More power

Solve more computational problems

A course-long question

- **Turing Machine** is the most powerful computational model known
- Are there computational problems that a Turing Machine cannot solve?
- The Answer is “yes” (unsolvable problems)
- **There are indeed unsolvable problems**, and we will see in detail what it means