

# CS 6041

# Theory of Computation

## Overview

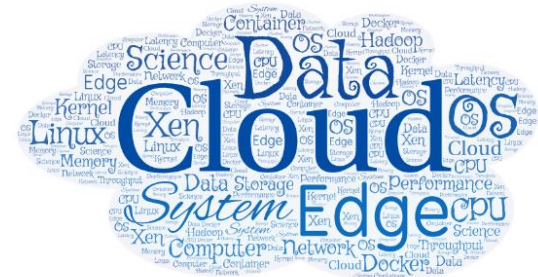
**Kun Suo**

Computer Science, Kennesaw State University

<https://kevinsuo.github.io/>

# Self Introduction

- Kun Suo, Ph.D.
  - Homepage, <https://kevinsuo.github.io/>
- Research interests:
  - Cloud computing and virtualization;
  - Operating systems, containers and kubernetes;
  - Software defined network (SDN) and network function virtualization (NFV)
  - Big data systems and machine learning systems
- Projects you may be interested in:
  - Several projects in Cloud & Data & Edge
  - <https://kevinsuo.github.io/code-lab.html>



# Now it's your turn

---

- Name, program/year, where from
- Your interests in Computer Science <https://www2.eecs.berkeley.edu/Research/Areas/CS/>
- What do you expect in the Theory of Computation?

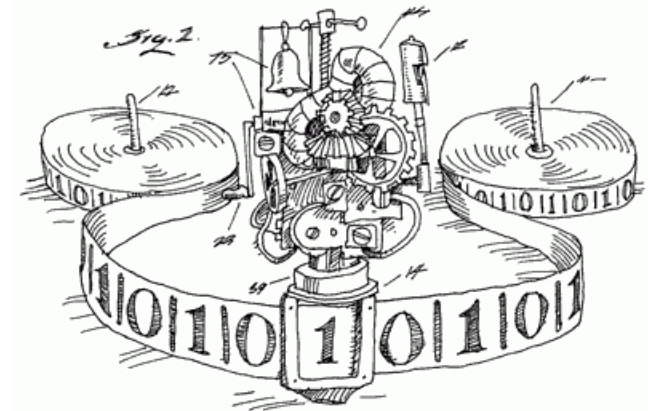
If you are in the online course, introduce yourself in D2L,  
Discussions → Self-Introduction



# What does Theory of Computation include?

---

- A study of topics from theoretical computer science that includes
  - Automata and languages
  - Computability theory
  - Complexity theory
- How efficiently (how fast, how much space, etc.) problems can be solved on a model of computation, using an algorithm



# What does Theory of Computation include?

---

- Example 1:  $L = \{s \mid \text{Binary strings end in 1s}\}$ .
  - Is it computable?
  - How can human beings solve it?
  - Can you build a machine to solve it?

0101010100011

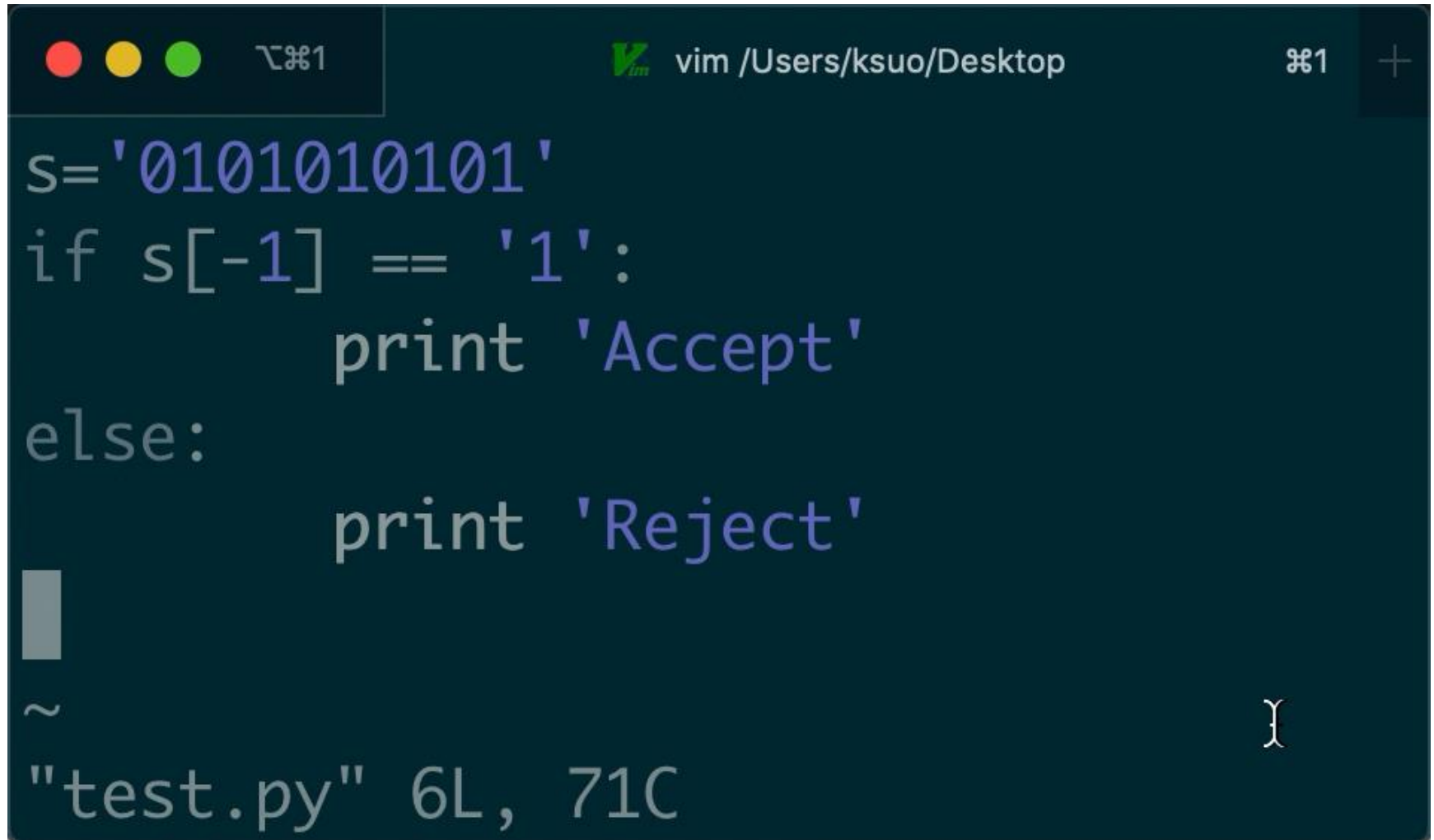
Accept

010100010

Reject



# What does Theory of Computation include?

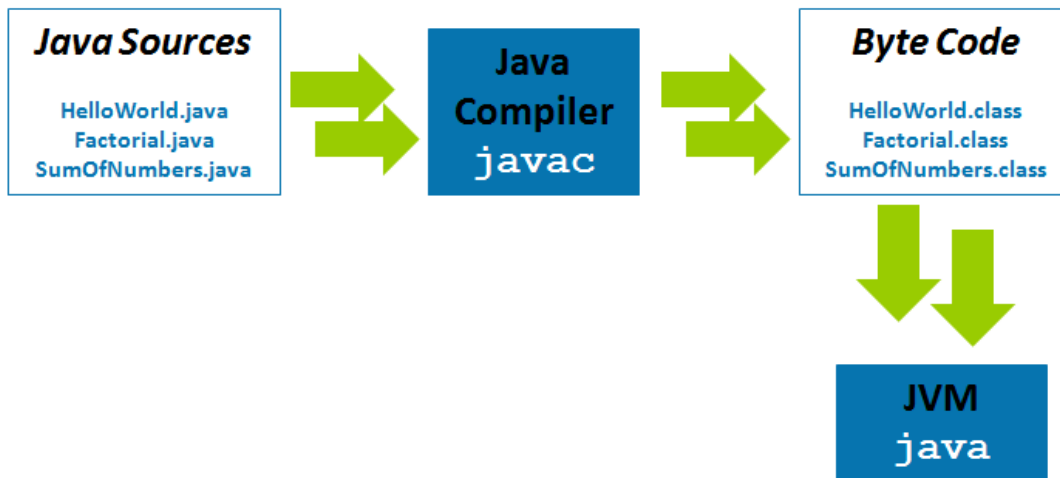
A screenshot of a Vim editor window. The title bar at the top shows three colored circles (red, yellow, green) on the left, the text "vim /Users/ksuo/Desktop" in the center, and "⌘1" and a plus sign on the right. The main editing area has a dark blue background with light blue text. The code is a Python script: 

```
s='01010101'
if s[-1] == '1':
    print 'Accept'
else:
    print 'Reject'
```

 Below the code, there is a vertical scrollbar on the left and a tilde symbol "~" on the right. At the bottom of the window, the status line displays `"test.py" 6L, 71C`.

# What does Theory of Computation include?

- Example 2: Accepts all valid Java code.
  - Is it computable?
  - Can you build a machine to solve it?
  - How can human beings solve it?



```
Output - compiler (run) x
run:
Success: true
java.lang.ClassNotFoundException: HelloWorld
    at java.net.URLClassLoader$1.run(URLClassLoader.java:372)
    at java.net.URLClassLoader$1.run(URLClassLoader.java:361)
    at java.security.AccessController.doPrivileged(Native Method)
    at java.net.URLClassLoader.findClass(URLClassLoader.java:360)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:424)
    at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:308)
    at java.lang.ClassLoader.loadClass(ClassLoader.java:357)
    at java.lang.Class.forName0(Native Method)
    at java.lang.Class.forName(Class.java:260)
    at CompileSourceInMemory.main(CompileSourceInMemory.java:50)
BUILD SUCCESSFUL (total time: 2 seconds)
```

A screenshot of a Java compiler output window titled "Output - compiler (run)". It shows the output of a compilation attempt. The first line is "run:". The second line is "Success: true". The third line is "java.lang.ClassNotFoundException: HelloWorld". This is followed by a stack trace with several lines of code references. The final line is "BUILD SUCCESSFUL (total time: 2 seconds)".

# What does Theory of Computation include?

---

- Example 3 (halting problem): for input  $w$ , determine whether  $w$  will end in finite time or infinite loop.
  - Is it computable?
  - Can you build a machine to solve it?
  - How can human beings solve it?

There is no program that solves the halting problem.

This proof was found by [Alan Turing](#) in 1936.

[https://scholar.google.com/scholar?hl=en&as\\_sdt=0%2C11&q=ON+COMPUTABLE+NUMBERS%2C+WITH+AN+APPLICATION+TO+THE+ENTSCHEIDUNGSPROBLEM&btnG=](https://scholar.google.com/scholar?hl=en&as_sdt=0%2C11&q=ON+COMPUTABLE+NUMBERS%2C+WITH+AN+APPLICATION+TO+THE+ENTSCHEIDUNGSPROBLEM&btnG=)





# What does Theory of Computation include?

---

- Example 1: Binary strings end in 0s.
- Example 2: Accepts all valid Java code.
- Example 3 (halting problem): for input  $w$ , determine whether  $w$  will end in finite time or infinite loop.

Easy vs Hard vs Not computable

Simple machine vs Complicated machine (with knowledge)

Fast vs Slow vs Not answer (proof)

1KB vs 500MB vs ...



# Course Information

---

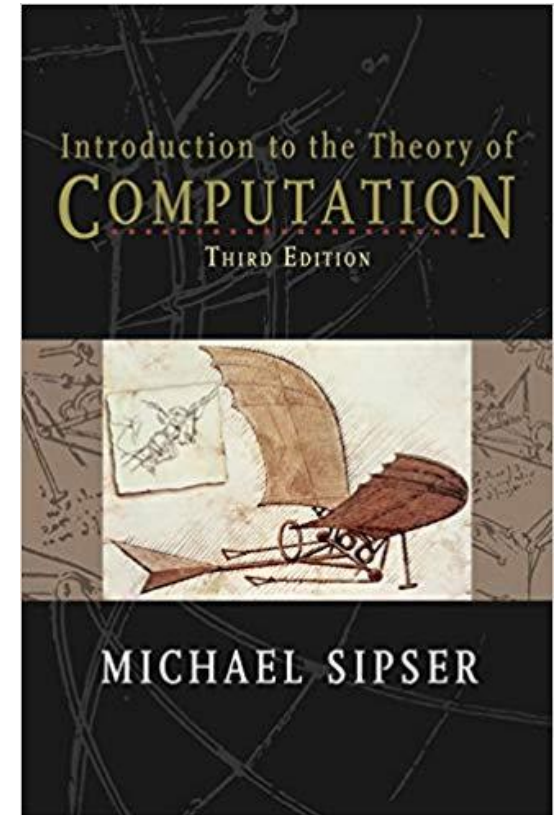
- Instructor: Dr. Kun Suo
- Office: J-3230
- Email: [ksuo@kennesaw.edu](mailto:ksuo@kennesaw.edu),
  - Only reply to e-mails that are sent from KSU student email accounts and title the course number [CS6041]
- Class Hours: online D2L
- Office Hours:
  - M/W, 4:00pm-5:00pm
  - By appointment
- Course Materials
  - Homework assignments, lecture slides, and other materials will be posted in the webpage (<https://kevinsuo.github.io/teaching.html>) and D2L.
  - All lectures will be recorded.



# Reference Book

---

- Professor Michael Sipser (MIT)
- Introduction to the Theory of Computation
- 3rd edition, Cengage Learning  
ISBN 13-978-1-133-18779-0
- Buy/Rent/eBook/Library



# Grading Policy

---

- Grading percentage

- ~~◉ In class discussion and attendance: 5%~~

- ◉ Homework (3x): 40% (please submit pdf file on D2L)

- ~~► One course presentation:~~

- ~~[https://docs.google.com/spreadsheets/d/1kGszd\\_RWYSKGaXkznMFz3mBWH17bPJvnzU8haclL\\_Co/edit?usp=sharing](https://docs.google.com/spreadsheets/d/1kGszd_RWYSKGaXkznMFz3mBWH17bPJvnzU8haclL_Co/edit?usp=sharing)~~

- ◉ 1<sup>st</sup> Midterm: 20%

- ◉ 2<sup>nd</sup> Midterm: 20%

- ◉ 3<sup>rd</sup> Final exam: 20%

*Late submission is not accepted.*



# Course Policy

---

- Grading scale

Percentage	Grade
90 - 100	A
80 - 89	B
70 - 79	C
60 - 69	D
Below 60	F



# Academic Integrity

---

- Academic dishonesty

[https://scai.kennesaw.edu/KSU\\_Codes\\_of\\_Conduct\\_2019-2020.pdf](https://scai.kennesaw.edu/KSU_Codes_of_Conduct_2019-2020.pdf)

- Cheating
- Plagiarism
- Collusion
- The submission for credit of any work or materials that are attributable in whole or in part to another person
- Taking an examination for another person
- Any act designed to give unfair advantage to a student or the attempt to commit

Receiving, attempting to receive, knowingly giving or attempting to give unauthorized assistance...



# How to succeed this class

---

- THINK hard, not WORK hard
- Scientific Thinking
- Passion to learn something NEW
- ASK ME (office hours) / Friends / Classmates questions
- Begin homework assignments EARLY



# Where to go for help ?

---

- Ask questions in class
- Ask questions outside class
  - Classmates and friends
- Attend office hours
  - Dr. Kun Suo: M/W 4:00PM – 5:00PM, J-3230 or make an appointment
- Search on the web
  - Stand on the shoulder of giants

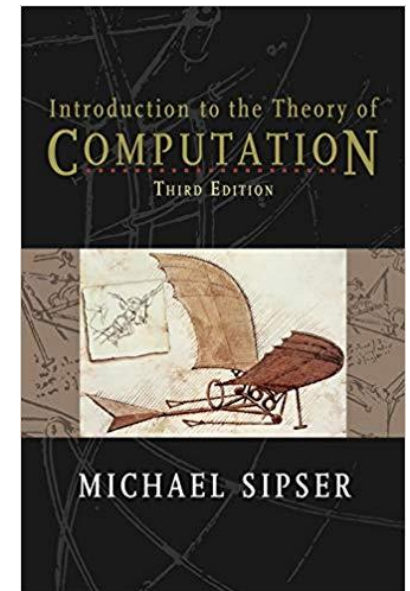




# Course Content Overview

---

- 0. Introduction
- 1. Regular language
- 2. Context-free language
- 3. Turing machine
- 4. Decidability
- 5. Reducibility
- 6. Advanced Topics in Computability Theory
- 7. Time complexity
- 8. Space complexity
- 9. Intractability
- 10. Advanced topics in complexity theory



# Course Content Overview

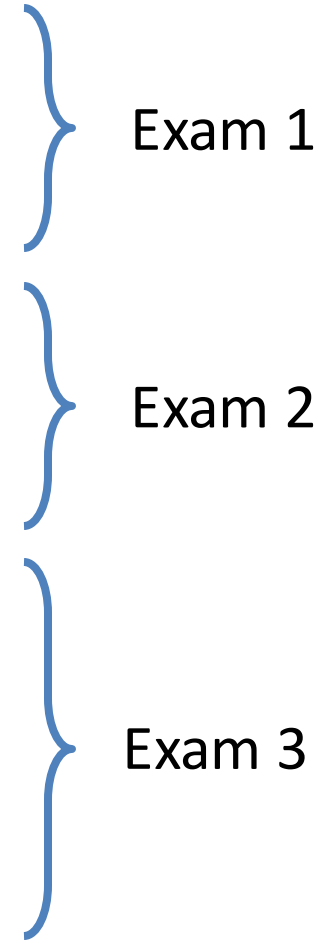
---

- Chapter 0 Introduction
- Chapter 1 Regular Languages
- Chapter 2 Context-Free Languages
- Chapter 3 The Church-Turing Thesis
- Chapter 4 Decidability
- Chapter 5 Reducibility
- Chapter 7 Time Complexity



# Course Content Overview

---

- Chapter 0 Introduction
  - Chapter 1 Regular Languages
  - Chapter 2 Context-Free Languages
  - Chapter 3 The Church-Turing Thesis
  - Chapter 4 Decidability
  - Chapter 5 Reducibility
  - Chapter 7 Time Complexity
- 
- Exam 1
- Exam 2
- Exam 3



# Course Content Overview

---

Week/Date	Topic	Chapters	Assignment
1	Introduction and overview	0	
2	Deterministic finite automata	1.1	
2	Nondeterministic finite automata	1.2	HW1
3	Regular expression and Regular language	1.3	
3	Non-regular language	1.4	
4	Exam 1		
4	Context free language	2.1	HW2
5	Pushdown Automata	2.2	
5	Non-Context free language	2.3	
6	Exam 2		
6	Turing machine	3	
7	Decidability	4	
7	Reducibility	5	HW 3
8	Complexity and NP-completeness	7	
8	Conclusion		
9	Exam 3		



# Language is the foundation of computation

---

```
package rentalStore;
import java.util.Enumeration;
import java.util.Vector;

class Customer {
    private String _name;
    private Vector<Rental> _rentals = new Vector<Rental>();

    public Customer(String name) {
        _name = name;
    }
    public String getMovie(Movie movie) {
        Rental rental = new Rental(new Movie("", Movie.NEW_RELEASE), 10);
        Movie m = rental._movie;
        return movie.getTitle();
    }
    public void addRental(Rental arg) {
        _rentals.addElement(arg);
    }
    public String getName() {
        return _name;
    }
}
```

Java source code

Read by people

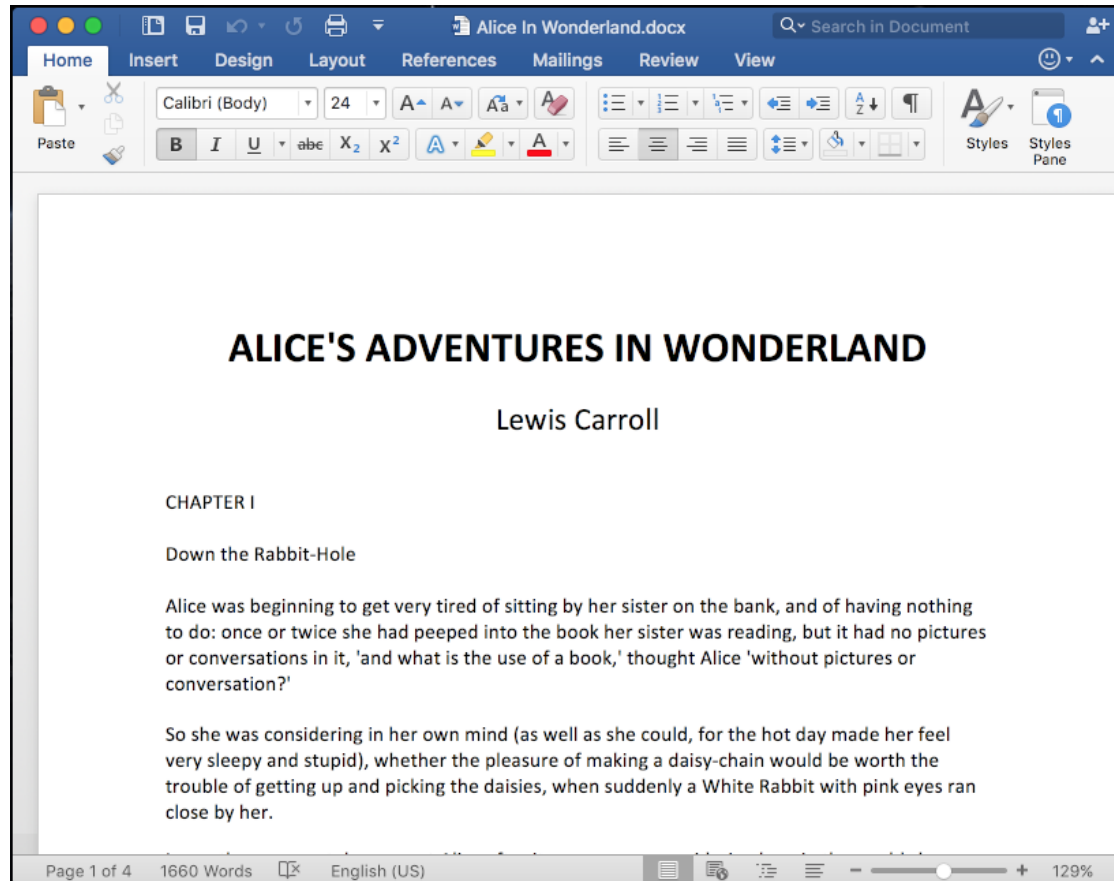


Binary code

Read by machines

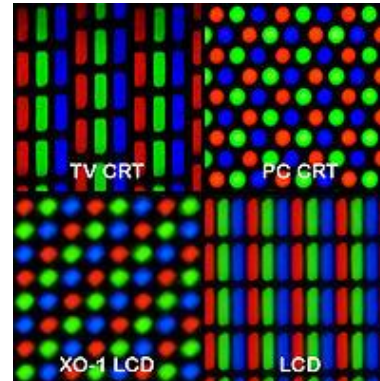
# Language is the foundation of computation

---

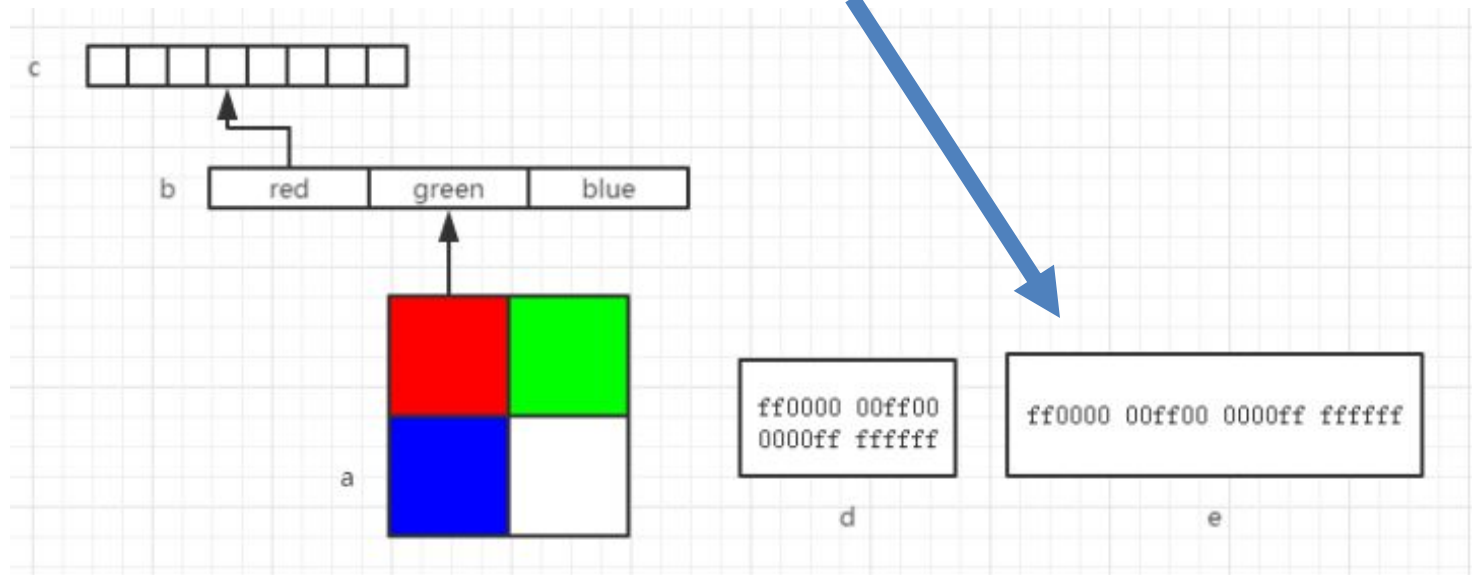


A word document is a language

# Language is the foundation of computation



A figure is a language



# Language universe in a big picture

---

- Are the languages all the same?





# Language universe in a big picture

---

- Are the languages all the same?



Satellite << Planet << Star << ...



Earth



Solar system

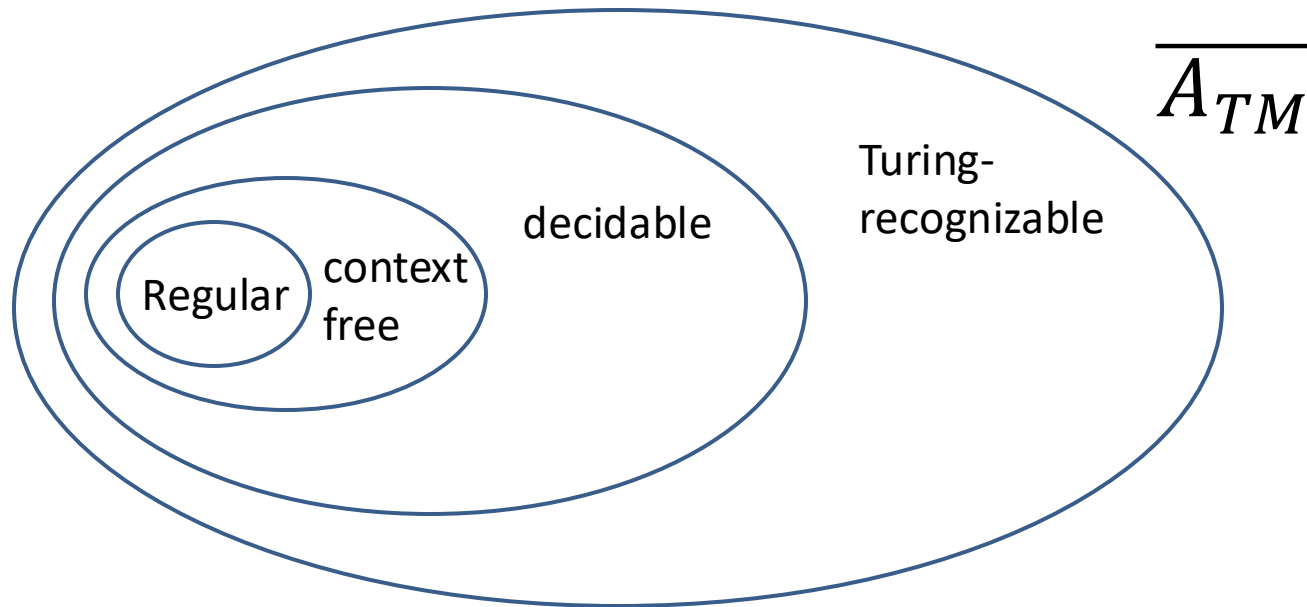


Galaxy



# Language universe in a big picture

---



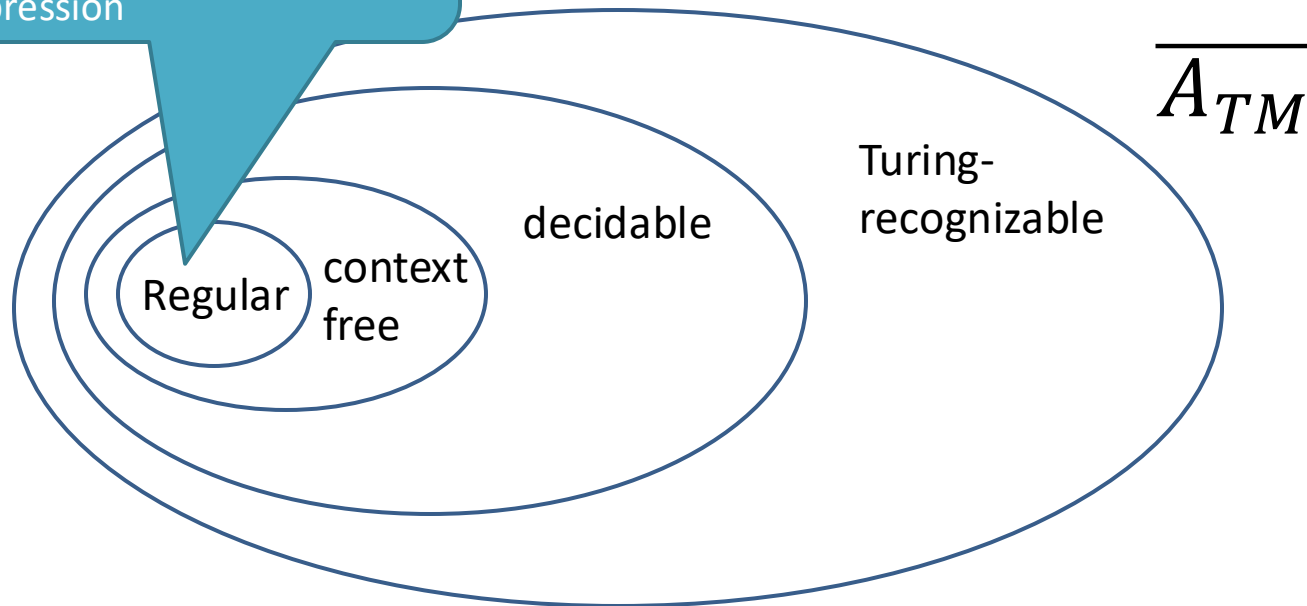
# Language universe in a big picture

Regular language  $\Leftrightarrow$

DFA: deterministic finite automata

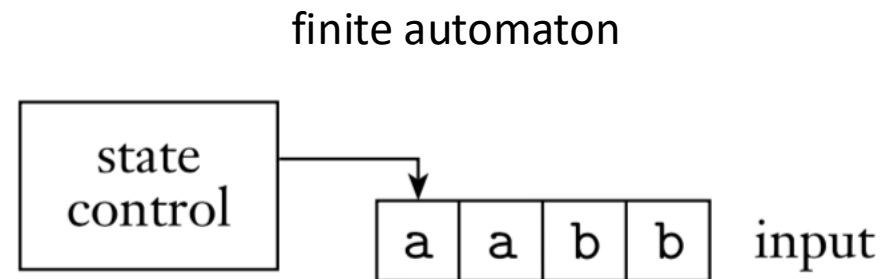
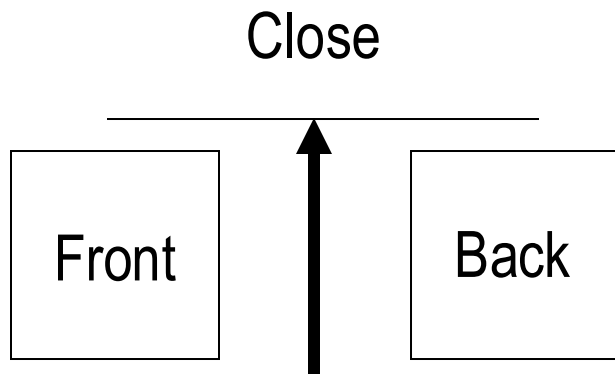
NFA: non-deterministic finite automata

RE: regular expression



# Language universe in a big picture

---



DFA example:  
Automatic Door

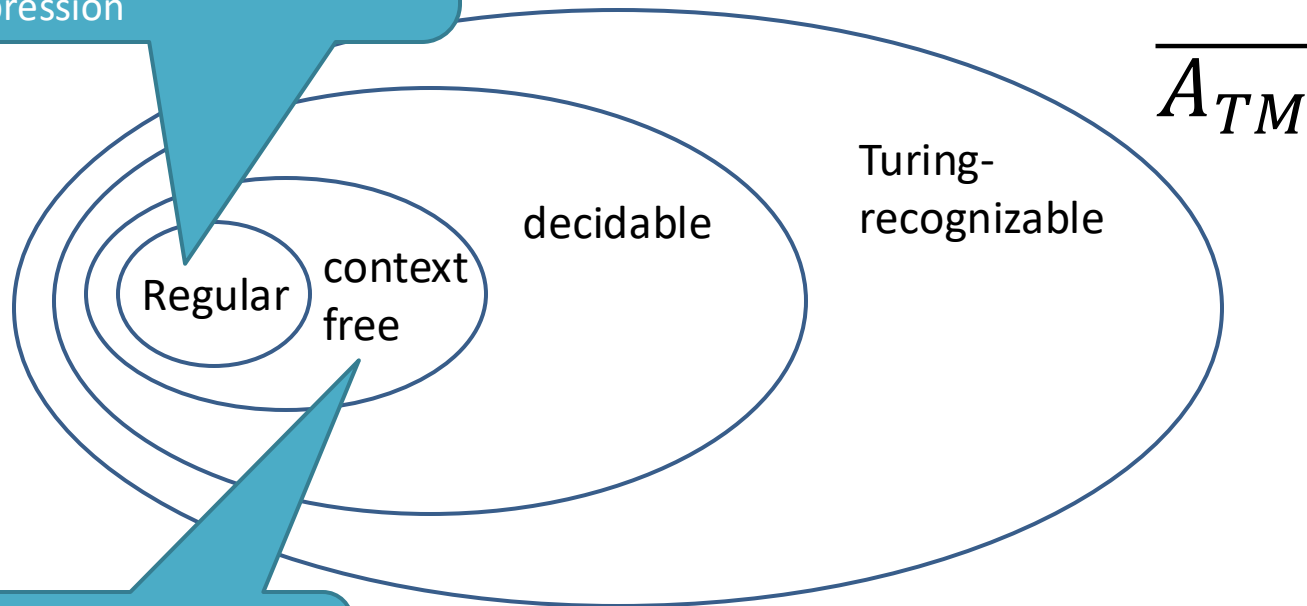
# Language universe in a big picture

Regular language  $\Leftrightarrow$

DFA: deterministic finite automata

NFA: non-deterministic finite automata

RE: regular expression



CFL: context free language

CFG: context free grammar

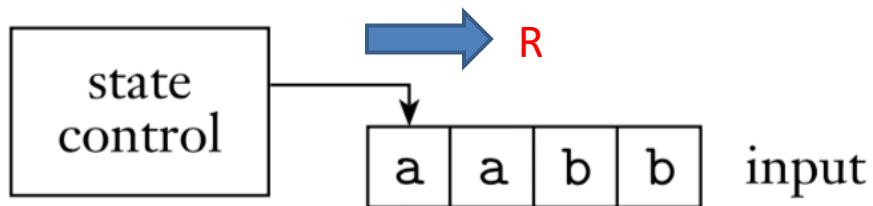
PDA: push down automata



# Language universe in a big picture

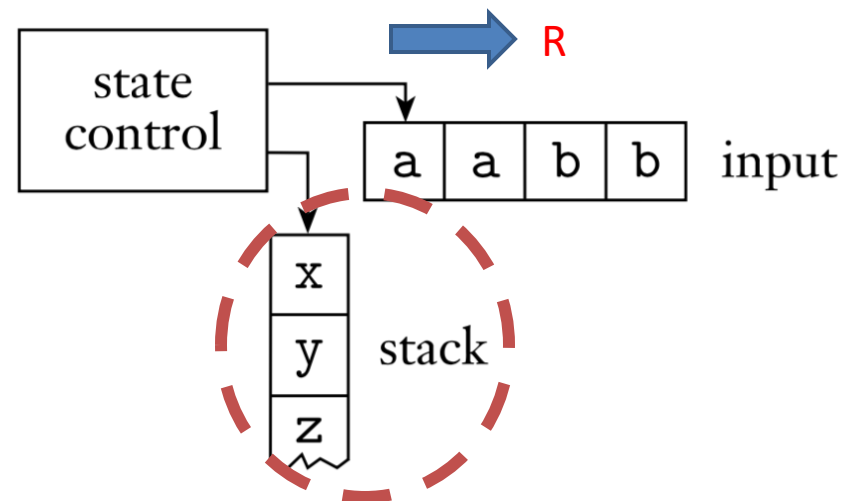
---

finite automaton



memory size = 1

pushdown automaton



Not only current input  
But also previous inputs

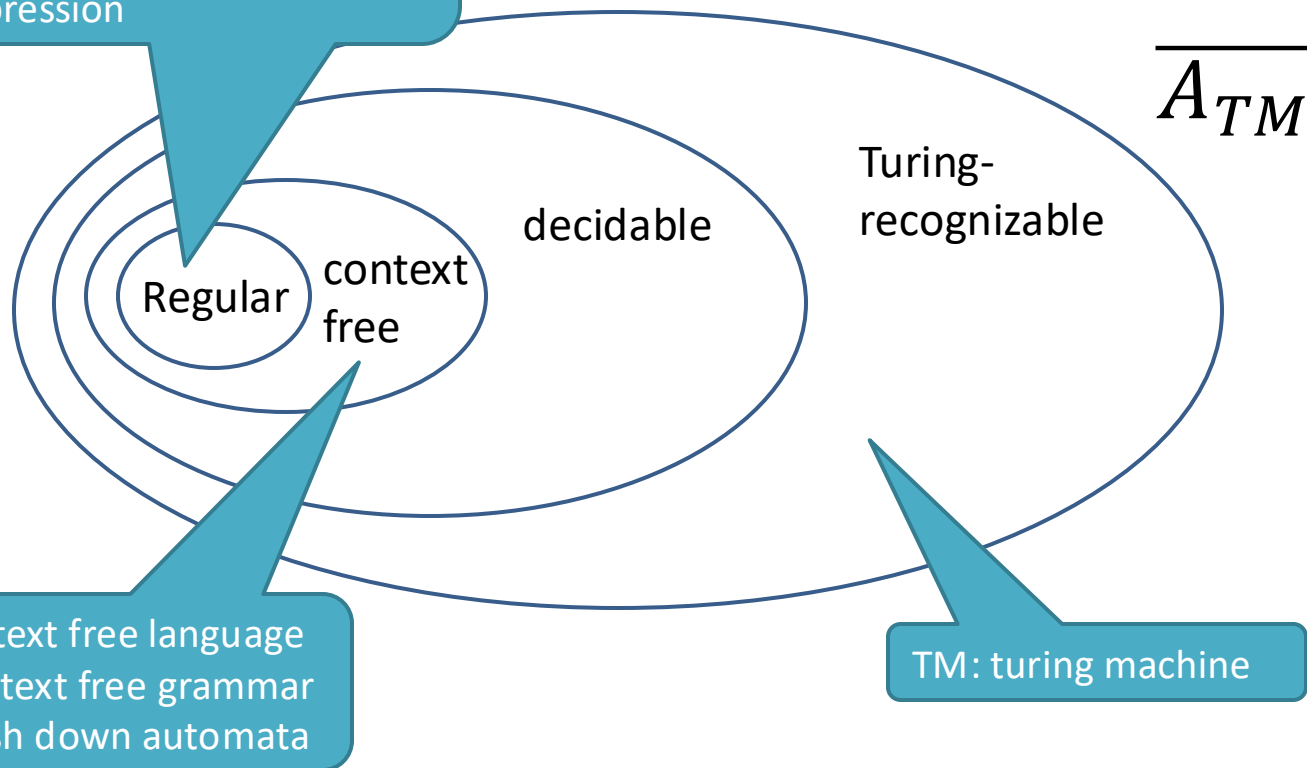
# Language universe in a big picture

Regular language  $\Leftrightarrow$

DFA: deterministic finite automata

NFA: non-deterministic finite automata

RE: regular expression



CFL: context free language

CFG: context free grammar

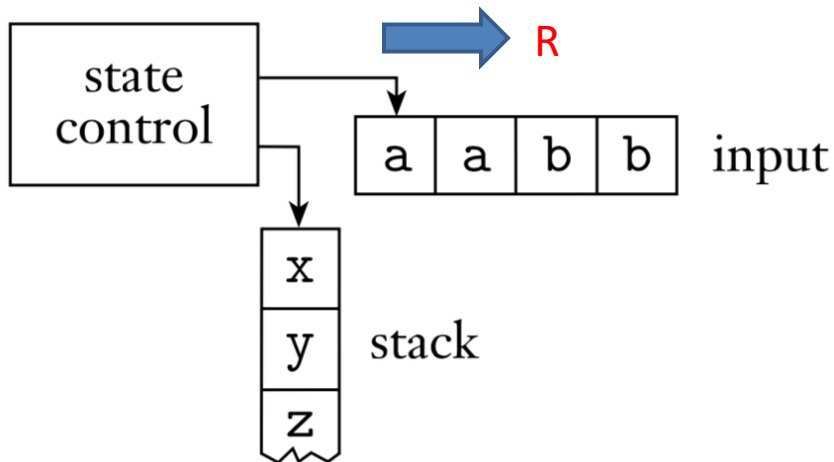
PDA: push down automata

TM: turing machine

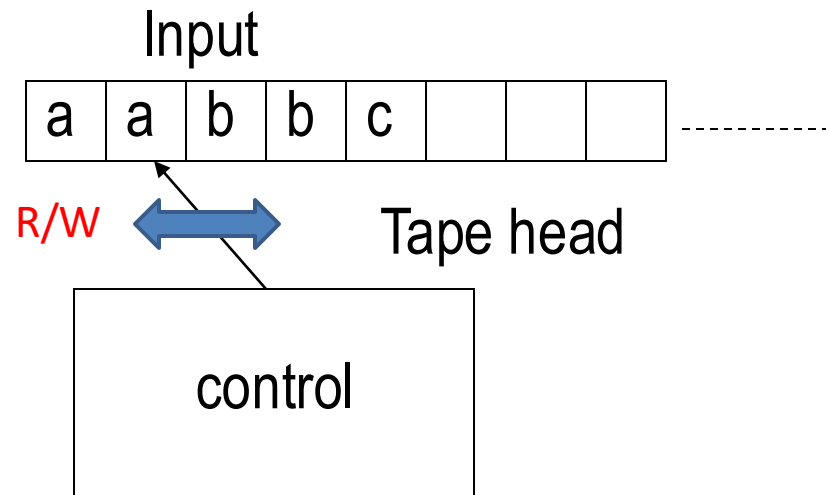


# Language universe in a big picture

pushdown automaton



Turing machine



Turing machines are equivalent to modern electronic computers at a certain theoretical level, but differ in details.



# Language universe in a big picture

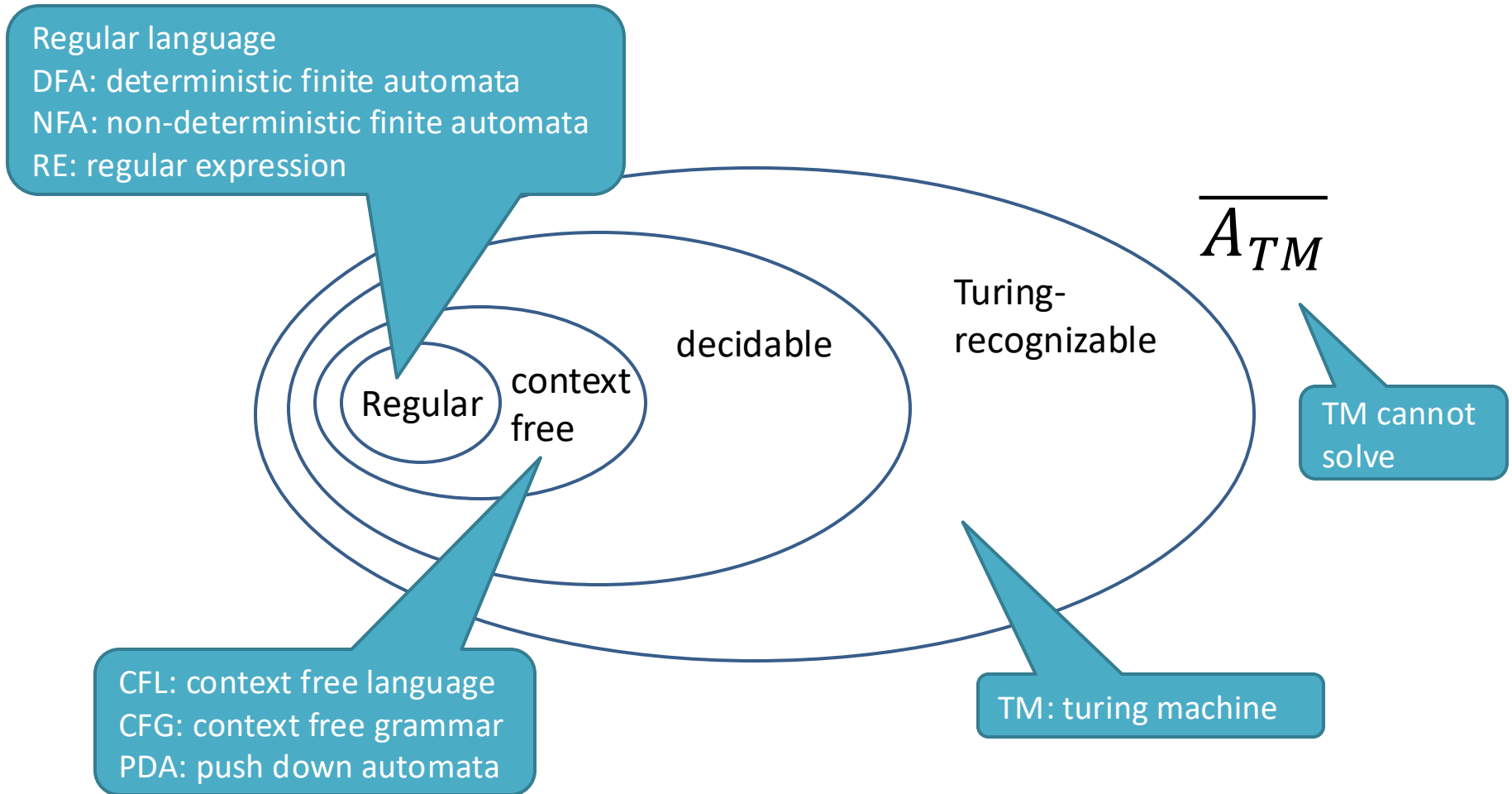
---



<https://www.youtube.com/watch?v=E3keLeMwfHY>



# Language universe in a big picture



# Language universe in a big picture

---

- P problem:
  - The general class of questions for which some algorithm can **provide** an answer in polynomial time
  - Give the answer!
- NP problem:
  - The class of questions for which an answer can be **verified** in polynomial time
  - Test the answer!



# Language universe in a big picture

---

- P problem:
  - Give the answer!
  - E.g., Sorting for  $n$  numbers, no longer than  $O(n^2)$
- NP problem:
  - Test the answer!
  - E.g., whether a given set has a subset that the sum of all its element is 0.
  - $s=\{-1,3,2,-5,6\}$ . subset= $\{3,2,-5\}$ , test in  $O(n)$  time



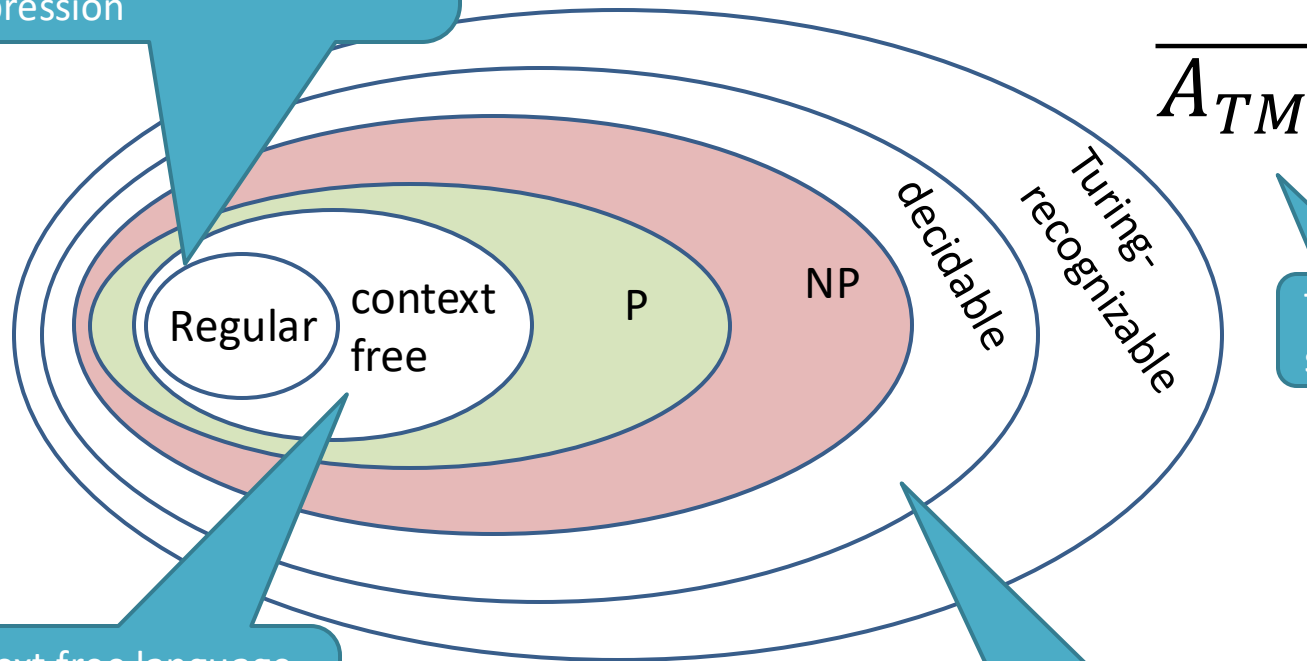
# Language universe in a big picture

Regular language

DFA: deterministic finite automata

NFA: non-deterministic finite automata

RE: regular expression



TM cannot solve

CFL: context free language

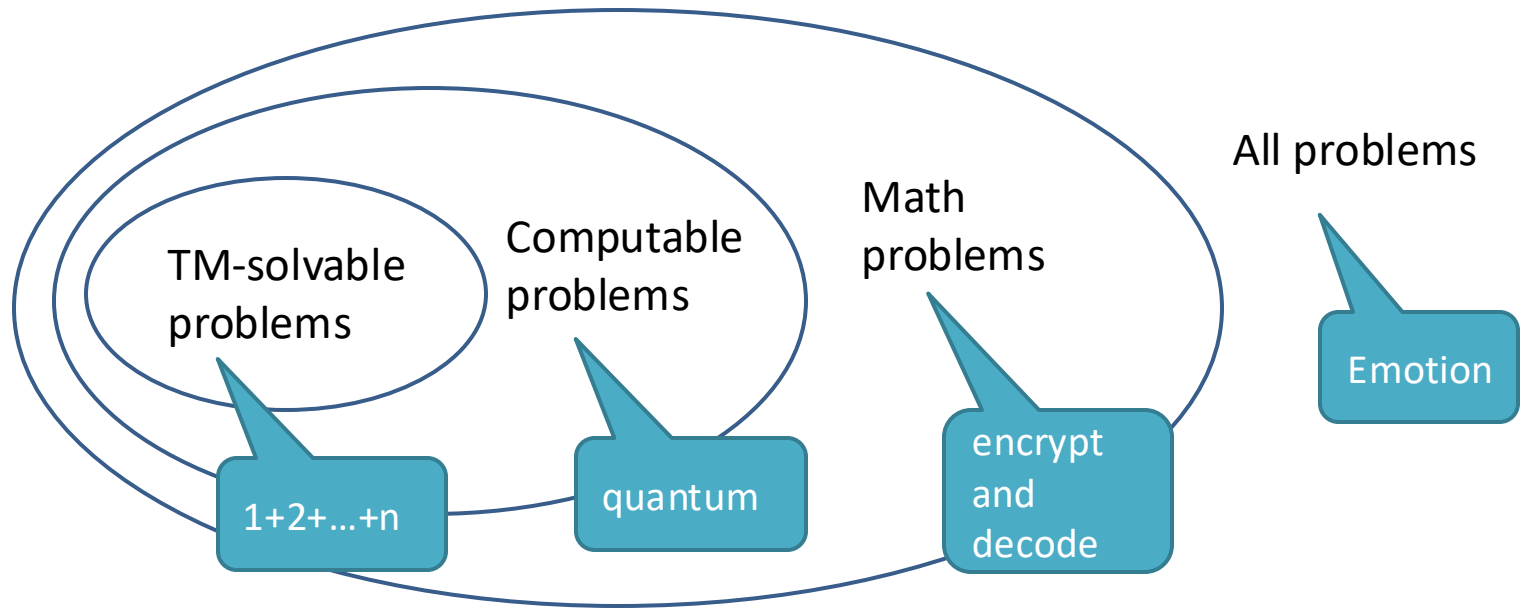
CFG: context free grammar

PDA: push down automata

TM: turing machine

# Problems

---



- Only a small proportion of problems can be solved by Turing machines in real life

# Conclusion

---

- Introduction of course
  - Information, book, grading, ...
- Course content overview
  - Calendar, content, schedule, ...
- Language universe in a big picture
  - Regular language
  - CFL
  - Turing machine
  - P vs. NP

