COMP30026 Models of Computation

Lecture 1: Introduction

Mak Nazecic-Andrlon and William Umboh

Semester 2, 2024

Teaching Staff: Who Are We?

Lecturers:

- Dr. Mak Nazecic-Andrlon (formal methods and theoretical computer science)
- Dr. William Umboh (optimization algorithms)

Head tutor: Ari Boyd

Tutors: Dylan Cookson-Mleczko, Gerald Huang, Jiwon Bae, Lezhou Lang, Philip Cervenjak, Richard Li, Rohan Hitchcock, Thomas Choi, William Sutherland, Xing Li, Yixiang Wang, Zixuan Cheng

A Little History

Computer science is the study of computation.

The 20th century brought a revolution:

- Formal models of computation (Turing 1936, Church 1936).
- The electronic computer (ENIAC in 1945).

But humans have studied computation since the abacus!

Why Study Theory?

"There is nothing more practical than a good theory."
Kurt Lewin

As a software engineer, you will at some point have to:

- Clearly and precisely explain complicated code.
- Optimize some code without breaking it.
- Write a technical specification.
- Parse some text.
- Work with asynchronous systems.

Logic and automata theory are irreplaceable for getting these right.

More Applications

- Propositional and predicate logic: hardware and software verification, planning, testing and fault finding, automated theorem proving, and logic programming.
- Automata: Controllers/counters, pattern matching, computational linguistics.
- Formal languages: Parsing, semantics.
- Algebra: Databases, programming languages, program analysis.
- Integer programming: Planners and constraint solvers, operations research.
- Number theory: Verification, encryption and decryption.
- Graphs: Data structures and algorithms; information retrieval.

Topic: Informal Proof

The art and science of writing mathematical arguments.

Covered throughout subject.

Why? Need logical reasoning to use these tools.

Topic: Formal Logic

Objective: check proofs algorithmically.

Informal proofs are written in natural language:

"P implies Q, and P holds. Therefore Q holds."

Formal proofs are more like code:

$$P \to Q, P \vdash Q$$

Topic: Automata Theory

Study of various idealized computing machines.

Automata theory subtopics:

- How they work.
- What they can do (computability theory).
- Proving elementary properties.

Topic: Formal Language Theory

Study of sets of strings.

Very close connection to automata theory.

What kinds of grammars correspond to what types of automata?

Further Ahead in Your Study

Complexity theory: given a problem, how hard is it inherently?

Not enough time for complexity theory.

We only cover elementary computability theory.

Take COMP90057 Advanced Theoretical Computer Science for more.

Further Ahead in Your Working Life

Computing technology changes with dizzying speed.

But the theoretical underpinnings change very slowly.

These tools will stay relevant for the foreseeable future.

Over to You—Introductions

Please introduce yourself to your neighbours.

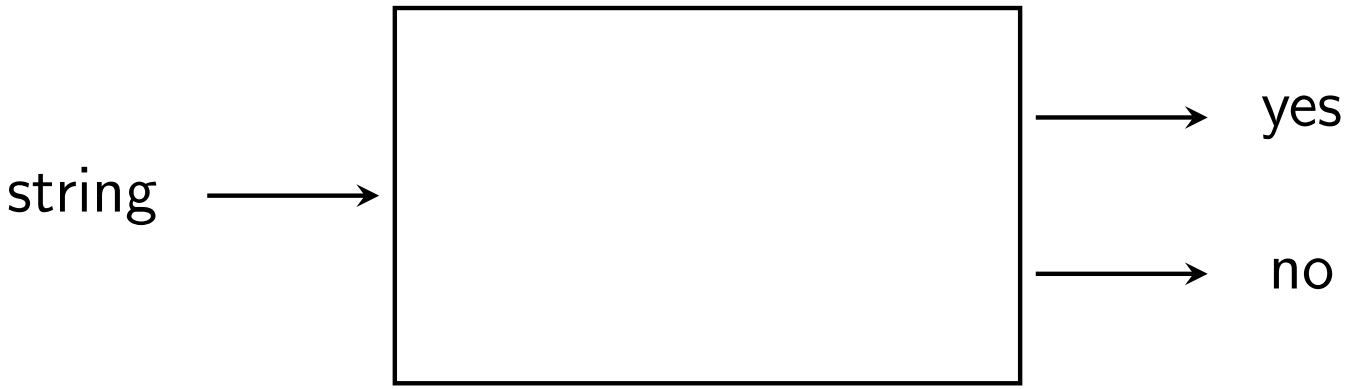
Tell them where you are from, what degree program you are enrolled in, something about languages or programming languages that you speak, or anything else that is interesting, like: Which is the best city you have visited? Which is the greatest film ever made?

Concept: Automaton

Idealized computing machines are generally called automata.

Most types of automata just read a string and output "yes" or "no".

Notable exception: Turing machines.



Concept: Formal Language

Definition

A (formal) language is a set of strings.

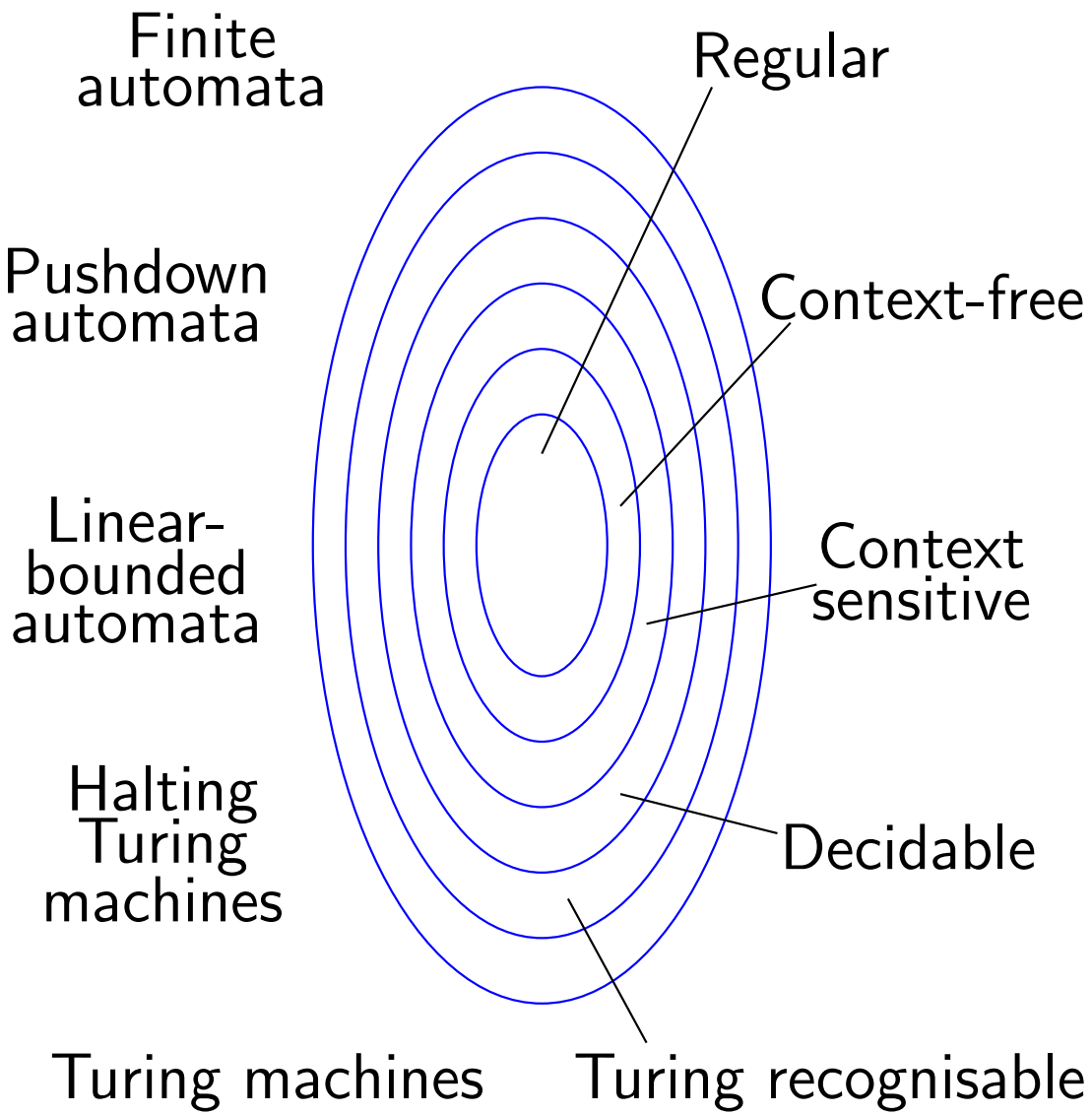
Example

$$\{\mathtt{ab}, \mathtt{aabb}, \mathtt{aaabbb}, \mathtt{aaaabbbb}, \ldots\}$$

We classify languages by:

- The sorts of rules (grammars) that generate them.
- The types of machines (automata) which can recognise or decide them.

Machines vs Languages



What is a Proof?

A deductive argument for a statement.

Conclusion must follow from the premises unconditionally.

That is, *evidence is irrelevant.*

Informal proofs

Example

Suppose that all philosophers are rabbits, and that Socrates is a philosopher.
Then, Socrates is a rabbit.

Figure: A rabbit named Socrates. (Image source: Reddit)

Formal proofs

Formal proofs are more like computer code:

Example

1. $\forall x(\mathit{Philosopher}(x) \to \mathit{Rabbit}(x))$ (premise)
2. $\mathit{Philosopher}(\mathit{Socrates})$ (premise)
3. $\mathit{Philosopher}(\mathit{Socrates}) \to \mathit{Rabbit}(\mathit{Socrates})$ (UI 1)
4. $\mathit{Rabbit}(\mathit{Socrates})$ (MP 3, 2)

Direct proof

Theorem

n^2 is even for all even integers n.

Proof.

Let n be an even integer.
Then $n = 2k$ for some integer k by definition.
Squaring both sides and rearranging, we get

$$n^2 = (2k)^2 = 4k^2 = 2 \cdot 2k^2.$$

Since k is an integer, so is $2k^2$.
Therefore n^2 is even by definition. □

Proof by contradiction

Theorem

$2n + 1$ is not even for any integer n.

Proof.

Let n be an integer, and suppose to the contrary that $2n + 1$ is even.
Then $2n + 1 = 2k$ for some integer k by definition.
Rearranging, we get $2k - 2n = 1$, and thus $k - n = \frac{1}{2}$.
But since k and n are integers, $k - n$ must also be an integer.
Contradiction!
Hence $2n + 1$ is not even. □

Proof by induction: motivation

Suppose you know the following two things:

1. 0 is purple.
2. If an integer n is purple, then $n + 1$ is also purple.

Challenge: prove that 3 is purple.

Induction: step by step

1. 0 is purple. (premise)
2. If an integer n is purple, then $n + 1$ is also purple. (premise)
3. If 0 is purple, then 1 is purple. (universal instantiation, from 2)
4. 1 is purple. (modus ponens, from 3 and 1)
5. If 1 is purple, then 2 is purple. (universal instantiation, from 2)
6. 2 is purple. (modus ponens, from 5 and 4)
7. If 2 is purple, then 3 is purple. (universal instantiation, from 2)
8. 3 is purple. (modus ponens, from 7 and 6)

Axiom schema of induction

Axiom

Let P be a property. If these two statements hold:

- *P holds of 0.*
- *For all $n \in \mathbb{N}$, if P holds of n, then it holds of $n+1$.*

Then P holds for all $n \in \mathbb{N}$.

Learning Support

Teaching materials will be posted on the LMS every week.

Lecturers and tutors will answer your questions during and after class.

Staff will also answer questions on the discussion forum (Ed).
(Strongly recommended!)

I will be running consultations every week. You don't have to bring specific questions!

Expectations

You cannot learn to do mathematics from watching lectures.

Deliberate practice is the only way.

If you want to succeed in this subject:

- Take out your calendar, and reserve 8 hours per week for practice!
- Attend all tutorials! They are very, very important! They start in week 1!

Assessment

- A 3-hour end-of-semester exam (70% of the final mark)
- Two assignments (due around Weeks 6 and 11) (12% each)
- Weekly worksheets (12 sheets, best 9 of 12)

There can be no extension to the worksheet deadlines, and we don't have any supplementary worksheets to give you. That's why we have the best 9 of 12 rule.

Academic misconduct is taken seriously at the University level. We do our best job to prevent this from happening and will report any potential misconduct that we find to the University for action.