COMP2022|2922
Models of Computation

Administration

Sasha Rubin

August 4, 2022

THE UNIVERSITY OF SYDNEY

Acknowledgement of Country

Before we begin, I would like to acknowledge and pay respect to the traditional owners of the land on which the University of Sydney is built — the Gadigal people. As we share our own knowledge, teaching, learning, and research practices within this University may we also pay respect to the knowledge embedded forever within the Aboriginal Custodianship of Country.

Welcome to Models of Computation COMP2022/2922

Welcome post discussing:

- Where lectures are held
- Learning platforms (canvas, ed, gradescope)
- Ed forum (how to ask a good question, etiquette)
- Assignments (number, due dates, referencing policy, extensions policy, late-penalty policy, regrade requests)
- Tutorials
- Special considerations
- Recommended reading
- Should you take 2922 instead?
- Tips on succeeding from past tutors
- A little about me

https://edstem.org/au/courses/9602/discussion/943209

Here are some highlights...

Lecturer

sasha.rubin@sydney.edu.au

Tutors

1. Johnson Chau
2. Steven Condell
3. Linus Cooper (TA)
4. Cameron Eggins
5. Eve Fernando
6. Haowen Gao
7. Joe Godbehere
8. Luke Harris
9. Greg McLellan (2922 Tutor and Guest Lecturer)
10. Xinyi Sheng
11. Kim Zheng

Organisation

Email

- only send us email for private issues
- check your email regularly for course announcements

Ed

- main discussion forum and for all announcements
- questions, answers
- material (lecture slides, tutorial sheets and solutions, assignment sheets and solutions)
- for asking personal questions to the team via a private post.
- some assignment submissions

Gradescope

- quizzes (submission and grades)
- assignments (submission and grades)

Canvas

- lecture recordings, final exams

COMP2022 vs COMP2922

What's the difference?

- COMP2922 is an extension of COMP2022.
- Both units share this main lecture
- The advanced unit has an extra hour of lecture
- Assignments and final exam are slightly different
- The units share a Canvas site, an Ed site and a GS site, and COMP2922 has an additional GS site for additional assignment questions.

https://edstem.org/au/courses/9602/discussion/950926

Assessments

- Quizzes (10%)
  - best 10 of 12 quizzes, 1% each
- Assignments (40%)
  - 5 equal weighted assignments, 8% each
  - released in weeks 4,6,8,10,12 by midnight on Sundays.
  - exactly two weeks to complete each assignment.
- Final exam (50% with a 40% barrier)
  - open book, not-invigilated

It is a requirement of the School of Computer Science that in order to pass this unit, a student must achieve at least 40% in the written examination. For subjects without a final exam, the 40% minimum requirement applies to the corresponding major assessment component specified by the lecturer. A student must also achieve an overall final mark of 50 or more. Any student not meeting these requirements may be given a maximum final mark of no more than 45 regardless of their average.

Teaching/learning modes – Summary

- Lectures (recorded)
  - motivate and introduce new concepts, and give some examples.
- Tutorials (not recorded)
  - consolidate your understanding with easy questions, expand your understanding with exam-level questions, provide individual feedback that is not possible in large lectures, and provide some discussion-level questions.
  - if in person, please double check the room (because the schedule has changed)
- Quizzes
  - test you on the basics before the tutorial.
  - no partial credit
- Assignments
  - will challenge you, deepen your understanding, and provide feedback on your learning. These have questions of varying difficulty, from straightforward to difficult.

https://edstem.org/au/courses/9602/discussion/952393

Assumed knowledge

This course assumes that you know basic discrete mathematics, including basic mathematical notation, basic set theory, and basic mathematical arguments.

You will struggle without such a background.

It is your responsibility to make sure you are very comfortable with the assumed knowledge.

There is an "Assumed Knowledge" sheet on Ed that you can use to self-test.

https://edstem.org/au/courses/9602/resources?download=14245

Special Consideration

While you are studying there may be circumstances or essential commitments that impact your academic performance. Follow official procedures (do not email us) in case of:

- illness,
- misadventure,
- sport or government service,
- religious observance.

www.sydney.edu.au/students/special-consideration.html

- All requests must go through SC.
  - Upload documentation within the required time.
- We cannot not mark emailed assignments or exams unless authorised to do so by SC.

Simple extensions and late-penalty policy

1. There will be no simple extensions for this course.
2. Late submissions without approved SC are prohibited, and will receive a mark of zero.

These policies are not yet finalised due to shifting university policies.

I am actively trying to resolve this.[1]

[1] The formula is this: the absolute max number of calendar days an assignment can be extended (for any reason) is 11 days. This is because we plan to provide feedback to assignment X before assignment X+1 is due, i.e., within 14 days of assignment X being due.

Student life, well being, and support

The university has a wide range of support and services.

- Aboriginal and Torres Strait Islander student support
- Career services
- Health, wellbeing and support services
- Learning support
- Mentoring
- Disability Services

www.sydney.edu.au/students/support.html

Academic Integrity (i)

Read the University Policy.
www.sydney.edu.au/students/academic-dishonesty.html

Penalties can be severe.

1. You should submit typed pdfs that can be read by turnitin, or you will get zero for that answer. Do not submit handwritten answers (unless explicitly allowed, e.g., for diagrams).
2. You can discuss assignments at a high-level, but you must write your own code and reports in your own words, and acknowledge any other resources you've used.
3. The only exception to this is that you do not need to acknowledge any materials on the Ed or Canvas page for this course (including lecture notes, tutorial questions and solutions, quiz questions and solutions), however you do need to acknowledge any textbooks.

Academic Integrity (ii)

In addition, for the final exam you are allowed to use passive information sources (i.e., existing written materials such as books and websites); however,

- you must not ask other people for answers or post questions on forums;
- you must answer in your own words;
- you must not reveal the questions to anyone else.

Feedback

1. I often solicit anonymous mid-semester feedback from you.
2. Feel free to provide constructive feedback on Ed. We will consider every well thought out suggestion. You can make posts on Ed that are anonymous to your fellow students (staff can see your name).
3. This year I am also asking for a student to volunteer to be a class representative. This person will meet with me a few times during the semester to discuss any feedback that you give them privately/anonymously. To volunteer, please make a private post on Ed with one sentence saying why you want to volunteer.

About me

I joined USyd in August 2019, and taught this course in 2020 and 2021.

My field of research is Computational Logic and Automata Theory, with applications to AI.

What can you expect of me?

- "Very formal, very specific teaching that made learning much simpler to avoid confusion or any other issue."
- "The examples and history of where these models are applied in relation to computer science makes it more interesting to learn about them."
- "The lecturer takes time and patience to wait and answer student questions with different approaches in explaining complex concepts."
- "The lecturer explains new concepts clearly without rushing forward or taking too much time."
- "This course is very clear about what it expects which I appreciate. There were clear distinctions between what you are expected to get from the lectures and what you are meant to synthesize on your own."

Common questions about this course

1. What topics will we learn?
2. Why are we studying this?
3. What are applications?
4. Will this course help me become a better programmer?
5. What practical skills will we learn?
6. How is this course different from other courses in CS?

What topics are covered?

To understand computing, we will study formal (i.e., mathematical) models of computation and computing devices.

Formal/mathematical models...

- may be more understandable and precise than an informal description (e.g., the Python Language Reference is long and word, but its formal model is < 500 lines).
- may be used to reason about the objects they model, and so answer questions about computation. What sort of questions?
  1. What are the fundamental capabilities and limitations of computers?
  2. What makes some computational problems hard and some easy?

Intriguing questions

Which of the following computational problems can be solved by a computer?

1. Find all occurrences of "abracadabra" in the Harry Potter books.
2. Decide if a given Python program has no syntax errors.
3. Decide if two different Python programs give the same output when run on the same input.

Vote now!
https://www.menti.com/ib8rk2qsq7

What topics are covered?

Course Topics

1. Regular Expressions, Finite Automata
2. Turing Machines
3. Propositional Logic, Predicate Logic
4. (Context-free grammars)

These topics appear in other courses:

- `DATA2001:Data Science`
- `COMP2123:Data Structures and Algorithms`
- `COMP3027:Algorithm Design`
- `COMP3109:Programming Languages and Paradigms`
- `COMP3308:Introduction to Artificial Intelligence`
- `COMP5046:Natural Language Processing`
- `PHIL3610:Logic and Computation`
- `MATH3066:Algebra and Logic`

Why are we studying this?

COMP2x22 is Core in the CS major. Quotes from the Handbook (2021):

- A major in computer science covers the key concepts of computation. You will learn the principles and techniques needed to solve tasks efficiently with computation, and how to express those solutions in software. You will also discover how computation can be modelled and how to reason about the limits of what computation can achieve.

Students who graduate from Computer Science will be able to...

- Construct models of a computational process in appropriate formalisms at appropriate levels of abstraction and relate models in different formalisms to one another;
- Apply key ideas from the theory of computation and its limits, recognise tasks where efficient perfect solutions should not be expected and where approximate solutions are appropriate and communicate the implications for users who want to solve such tasks.

What are applications of these models?

Logic

- designing digitial circuits
- representing knowledge bases in AI
- a data model (notation for describing data, operations for manipulating data)
- foundations of Database Query Languages (SQL)
- foundations of Declarative Programming Languages (Datalog)

Finite Automata, Regular Expressions, Regular Grammars

- lexical analysers of compilers
- pattern matching large bodies of text (web search engines, grep, etc.)
- used as features in Natural Language Processing (NLP)
- design and implementation of interacting components (netword protocols, ecommerce, etc.)
- foundations of formal verification (i.e., checking software/hardware correct)
- foundations of query languages for graph-databases

Context-free grammars, Pushdown Automata

- parsers (aka Syntax analysis)

Turing Machines

- mainly of theoretical interest
- for understanding what is and is not computable

Killer app?

- FlashRelate: Extracting Relational Data from Semi-Structured Spreadsheets Using Examples
- Uses a type of regular expression!

  https://youtu.be/g2Dhf4Tmp8c

Will this course help me become a better programmer?

Computer science is not glorified programming.

Edsger Dijkstra — Turing-award winner

The ideas in this course are independent of any particular programming language, but will help you understand how programming languages and computing work.

In 20 years from now, computers and programming will likely be very different. However, this material will be very much the same, as it has been for over 50 years.

What practical skills will we learn?

(1) You will be able to better answer questions of the form:

What computational resources are required for solving a given problem?

- This is useful to understand if you can use existing tools to solve your problem, or if you need to create your own solution.
- Can my problem be solved...
  - with a regular expression?
  - with a single integer variable?
  - with a single stack?
  - with an SQL query?
  - with a program? i.e., can it be solved at all?

What practical skills will we learn?

(2) You will be able to write programs that process user commands.

- E.g.,
  - Scanner (converts character stream to token stream)
  - Parser (convertes token stream to parse tree)
  - Everytime you run a Python program, a parser converts your code into bytecode which is fed to an interpreter.
- This skill will be useful if you are asked to write a Domain Specific Language (aka mini-language) to solve problems in a specific domain.
- Examples of DSLs:
  - HTML for formatting web documents
  - SQL for relational database queries
  - MATLAB for numeric computing
  - Mathematica for symbolic computing
  - Verilog for describing hardware
  - PDDL for describing planning domains in AI.

What practical skills will we learn?

(3) You will develop skills of precise and formal reasoning, and so become more "mathematically mature".

- This is useful in the design and analysis of complex systems, e.g.,
  - `COMP2123:Data Structures and Algorithms`
  - `COMP3027:Algorithm Design`

How is this course different?

This course is more abstract and formal than most CS courses, but not as mathematical as most MATH courses.

Why is it so abstract?

- There is no fixed set of practical skills you can rely on to answer your computational problems. However, there is a set of abstract concepts that gives us a framework in which to think about and solve computational problems.
- There is beauty in mathematics! Just like in art and literature.

Why is it so formal?

- Leads to less confusion when learning concepts.
- Programs are very unforgiving of imprecise thinking.
- In your studies and work you will come across formal objects, so you have to learn how to be formal and precise.
  1. formal specifications (software engineering)
  2. logic (coding, database query languages, logic programming)
  3. formal languages (syntax of PLs).

Is this course difficult?

This course can be difficult because it covers abstract and mathematical topics which are not very easy to grasp without putting in a good deal of hard work.

So:

- I will try my best to contextualise the topics, and explain why they are useful.
- You should complete all tutorial questions, and start assignments early.
- If/when you get stuck, balance perseverance with asking for help!