Computer Science 351

Welcome To Computer Science 351!

Instructor: Wayne Eberly

Department of Computer Science
University of Calgary

Lecture #1

What is CPSC 351?

What is Computer Science 351?

- A course in ***mathematical foundations of computer science*** and ***computability theory*** required for the BSc in Computer Science program at the University of Calgary
- Intended primarily for computer science majors, and other students interested in careers in software development

Background: Mathematical Foundations

Prerequisite: Computer Science 251, or Statistics 213 and either Mathematics 271 or 273

- CPSC 351 continues the presentation of the mathematical foundations of computer science that was started in CPSC 251. It also introduces material in theoretical computer science that depend heavily on this.
- While CPSC 251 is now recommended for students in the CPSC major program, students transferring in from other instructions or programs, or combining studies in computer science with other disciplines, can satisfy these requirements by completing (equivalents for) Statistics 213 and either Mathematics 271 or 273, instead.

Background: Computer Programming

Prerequisite: One of Computer Science 219, 233 or 235

This is not a programming course. However...

- When Turing machines and decidability are discussed, a background in computer programming will help to understand various computational problems that are being introduced, and to understand why various "decision problems" are decidable, and why various functions are computable.
- The analysis of ***randomized algorithms*** — algorithms that make random choices when solving problems — is a significant application of the material from probability theory that is introduced in CPSC 351.

Additional Background

Prerequisite: One of Mathematics 249, 265 or 275.

- Ideas and results from calculus (introduced in Mathematics 249, 265 or 271) will be useful when various results about probability are being presented and applied.

Prerequisite: One of Philosophy 279 or 377

- A background in mathematical logic (introduced in Philosophy 279 or 377) will be useful when information about computability is being presented in this course.

Topic: A Simple — Limited — Model of Computation

The course will include an introduction to finite state machines — ***finite automata*** — and the class of languages that these can represent — ***regular languages***.

- These are — provably — not as powerful as real computers.
- This study includes an introduction to ***regular expressions*** — which are used to perform powerful searches in text editors, word processes, and data base. This material is also useful for software compilation — so it is definitely useful.

Learning Goals for This Part of the Course

On conclusion of this part of this course you should also be able to do the following.

- Learn to follow a ***design process*** — in order to obtain a deterministic finite automaton that has a given language. Then apply foundational material (from the course prerequisites) to provide a mathematically rigorous — and convincing — *proof* that your DFA is ***correct***.
- Demonstrate understanding of, and ability to use, various related models by converting from one representation of a regular language (as described in this course) to another — learning about ***simulations***, along the way.

Learning Goals for This Part of the Course

On conclusion of this part of this course you should be able to do the following.

- Apply proof techniques, that will be introduced in this course, to prove that a given language is ***not*** a regular language. The use of ***closure properties*** to prove that a language is — in some sense — not "easy to decide" will be introduced, as part of this.

Learning Goals for This Part of the Course

What Should have Happened, Here?

- Studying a simple model, like this, allows a variety of concepts and techniques to be introduced, without being distracted by a more complicated model. . . which leads to the topic that follows.

Topic: Turing Machines and the "Church-Turing Thesis"

The course will also provide an introduction to ***Turing machines*** and the ***Church-Turing thesis***.

- A (seemingly small) change to the "finite state" model results in a much more powerful model of computation, namely, a ***Turing machine***.
- Using ***simulations***, these can be shown to be computationally as powerful as "real computers" (if I/O conversion is overlooked).

Learning Goals for This Part of the Course

On conclusion of this part of this course you should be able to do the following.

- Use one or more design processes to develop Turing machines that decide ***simple*** languages or compute ***simple*** functions.
- Develop multi-tape Turing machines to decide (slightly) more complicated languages and to compute (slightly) more complicated languages.
- Develop ***simulations*** to relate the computational power of various machine models, including "variants" of Turing machines.

Learning Goals for This Part of the Course

NOT a Learning Goal:

- Designing Turing machines, to solve complicated problems, is ***not*** a learning goal for this course!
- Ideally, you will learn (approximately) enough about "Turing machine design" to appreciate (and believe) the ***Church-Turing thesis*** and to complete exercises included in the topic that follows this.

Topic: Reductions and Undecidability

The course will provide an introduction to ***reductions*** and ***undecidability***.

- This concerns the following question:

What are the fundamental capabilities and limitations of computers?

A computational problem that provably ***cannot*** be solved by a Turing machine — or with a more realistic computer — will be identified, along with a technique that can be used to prove that *other* problems cannot be solved in this way, as well.

Learning Goals for This Part of the Course

On conclusion of this part of this course you should be able to do the following.

- Use a ***reduction*** to prove that a given language is *undecidable.*
- Use a specific kind of reduction — a ***many-one reduction*** — to prove that a given language is *unrecognizable*.

Topic: Probability Theory

The course will include material from ***probability theory***.

- This continues a discussion of this topic from CPSC 251.
- The material being covered is reasonably standard — it is included in many introductory textbooks on probability and statistics.
- A significant application of this material, for computer science majors, is the analysis of ***randomized algorithms*** — algorithms that make random choices when solving problems.

Learning Goals for This Part of the Course

On conclusion of this part of this course you should be able to do the following.

- Model "real world" situations using the machinery that has been introduced, including probability distributions and random variables.
- Then, use results about probabilities and expected values, included in this material, to make useful or interesting conclusions about the thing(s) you have modelled.
- Occasionally, use *other* material about writing proofs, from CPSC 251 and this course, to prove results about probabilities or expected values that are of general use (or that help you to demonstrate your proof-writing skills).

Resources: Preparatory Reading

Lectures and tutorials will be ***synchronous***, but ***flipped***: Material that students are expected to work through before attending each lecture will be available on D2L.

- This may be provided as one or more videos in which the instructor works through lecture slides resembling those that could be used in an "unflipped" version of this course. The slides will also be available as a PDF file.
- Sometimes, one or more ***supplemental documents*** will be supplied. These help you to review background material, include additional details about claims being made, or provide additional examples to show how lecture material can be used.

Resources: Lecture Presentations

Lecture presentations (either remote, or face-to-face) will focus on how to solve problems that are related to the above material, which might resemble problems that students are asked to solve in tutorial exercises, assignments and tests. These might also present additional material that supplement the material students have worked through on their own,

- An outline for the presentation will generally be provided ahead of time. If there is time for it, students can try to complete this outline on their own, before the lecture presentation, to test their understanding of material in the lecture notes.
- A completed version of the outline will be made available shortly after the lecture presentation.

Resources Finishing Up

Additional material, to continue work after the lectures, will (sometimes) be provided.

- "Questions for Review" are intended to help you to focus on the lecture material that is especially important, and might be focussed on during test preparation.
- Additional exercises will be given, so that you can practice solving problems before assignments and tests.

Resources: Tutorials

Problems to be solved in tutorials will be described ***before*** the tutorials.

- Students will be asked to try to solve these problems before the tutorial too.
- ***Teaching assistants will not be presenting solutions for these problems in the tutorials.*** Instead, in meetings during scheduled tutorial times, teaching assistants will help students to see *how* these problems can be solved and *how* students' solutions can be evaluated
- Written ***solutions*** for the problems will be made available, on D2L, ***after*** the tutorials.

Figuring out how to solve tutorial exercise problems will help students to complete problems on later assignments and tests.

Resources: Office Hours

- Instructors of computer science courses have regularly scheduled ***office hours***. Students can attend these to get any extra help that is needed to understand lecture material, work on assignments, or prepare for tests.
- It is *not* necessary to arrange to meet with the instructor if you are using these regularly scheduled office hours (but you might have to wait for your turn to meet with the instructor if there is a line).
- If the regularly scheduled time is not convenient then you can meet with the CPSC 351 instructor at other times as well — but these other meetings must be requested and arranged by email, in advance.
- Teaching assistants are ***not paid*** to have office hours or other meetings with students outside the scheduled tutorial times.

Resources: Other Material

- The course material that is available on D2L should be sufficient for most students. However, some students find different explanations of material to be helpful. Students might also be interested in topics that would presented *after* the ones covered in this course.
- A list of ***recommended references*** will be also provided on the course web site. These are ***optional*** — none are required.

Resources: Other Material

- Please note that, if you use other material, ***this material cannot replace the material that is used in lectures and provided on the course web site***. In particular, if you find other material that somehow seems to disagree with or contradict course material, then the course material (provided the course web site) must be followed.

Why Do It This Way?

Why Do It This Way?

- ***You learn by doing:*** Many people remember, understand, and can use material better (to solve problems) *if they have done something with this material*, instead of hearing another people talk about it.

  Furthermore, when you are trying to do something that is new, you might need more than one attempt before you become comfortable with a process, and feel confident that you are doing it well.

Why Do It This Way?

Why Do It This Way?

- ***It isn't always obvious what is important:*** Sometimes you need to learn to do things in multiple stages — so that you are learning to do fundamental (sometimes, deceptively simple) things before you are ready for the more complex things that you are actually interested in.

  It can be tempting to skip over the easy-looking stuff at the beginning — especially if this does not seem to be very interesting or exciting. Unfortunately, this can result in your being unprepared for the more challenging stuff that will follow, without your realizing that this is happening. Putting time into this course, early on, really *can* help you to be prepared for later work.

Why Do It This Way?

Why Do It This Way?

- ***This discipline is changing rapidly, and will keep doing so!*** Some of the "cutting edge" stuff that you might expect to be learning about, right now, will be obsolete by the time you graduate. The stuff that is "cutting edge" shortly after you graduate might also be obsolete by the time you are in the middle of your career.

  This course introduces foundational material that will likely be the basis for all those new technologies that follow. Taking this course seriously will also you to improve your ***learning skills*** — which will you need, later in your career, as you need to learn all those new technologies and tools, on your own.

Why Do It This Way?

If You Really Don't Want to Deal with a "Flipped" Course:

- CPSC 351 is currently offered more than once, each academic year — and another section of this course, with more "traditional" lectures, will be available soon.

What's Next?

What's Next?

- A ***mathematics review*** — which covers material from CPSC 251 which we will need away — is available.
- Notes on ***alphabets, strings, and languages*** are available. These will be briefly discussed in the first lecture presentation.
- A supplemental document, with more information about ***course administration***, is available.

What's Next?

What's Next?

- Lecture #2 introduces ***deterministic finite automata*** — the simple (and restricted) model of computation that will be studied at the beginning of this course.