Module 5 Overview

Decidability & Reductions (Chapters 4 & 5)

Module Intro

In Chapter 3 we introduced the Turing machine as a model of a general purpose computer and defined the notion of algorithm in terms of Turing machines by means of the Church–Turing thesis.

In this chapter we begin to investigate the power of algorithms to solve problems. We demonstrate certain problems that can be solved algorithmically and others that cannot. Our objective is to explore the limits of algorithmic solvability. You are probably familiar with solvability by algorithms because much of computer science is devoted to solving problems. The unsolvability of certain problems may come as a surprise.

We presented several examples of problems that are solvable on a Turing machine and gave one example of a problem, ATM, that is computationally unsolvable. In Chapter 5 we examine several additional unsolvable problems. In doing so, we introduce the primary method for proving that problems are computationally unsolvable. It is called reducibility.

A reduction is a way of converting one problem to another problem in such a way that a solution to the second problem can be used to solve the first problem. Such reducibilities come up often in everyday life, even if we don't usually refer to them in this way.

Reducibility plays an important role in classifying problems by decidability, and later in complexity theory as well. When A is reducible to B, solving A cannot be harder than solving B because a solution to B gives a solution to A. In terms of computability theory, if A is reducible to B and B is decidable, A also is decidable. Equivalently, if A is undecidable and reducible to B, B is undecidable. This last version is key to proving that various problems are undecidable.

In short, our method for proving that a problem is undecidable will be to show that some other problem already known to be undecidable reduces to it.

Objectives

Upon completion of this module, students will be able to:

1. Explain what it means for a set to be countable
2. Demonstrate how a given set is countable
3. Use Cantor's Diagonalization Argument to show a set is not countable
4. Provide examples of uncountable sets
5. Explain what it means for a Turing Machine to be robust
6. Define the Church-Turing thesis and strong Church-Turing thesis and explain why they are important
7. Define a Turing decidable language
8. Define a Turing recognizable language
9. Define what a language is, and how that relates to the number of possible strings from a given alphabet
10. Explain how reductions work and the purpose of reductions
11. Explain how the mapping reduction works and their benefits
12. Explain how each language in a mapping reduction is related to the other language
13. Use Rice's Theorem to prove a language is undecidable
14. Understand A_{TM} and how to use it in a reduction

Readings and Resources

You can explore this module's information in multiple ways:

- View slides
- Access this information from the textbook (**Chapters 4 & 5)** [Course Textbook (https://virginiacommonwealth.instructure.com/courses/119232/pages/course-textbook)](https://virginiacommonwealth.instructure.com/courses/119232/pages/course-textbook)
- Solve practice activities

Module at a Glance

Below is an overview of this module. Pay particular attention to items with **points values** and **due dates,** as these are graded assignments. (Note: module overview will not display if you are accessing Canvas on a mobile device.)