Chapter 1

Introduction

1.1 Introduction and Overview

The goal of this lecture series is to provide training in formal thinking to students of applied informatics. Students should learn to analyze and solve problems using formal methods. The class will expose students to a wide set of problems and show ways of solving them. Formal methods, as they are used in theoretical computer science, constitute an essential part of a computer science education, not only for those who target an academic or research career, but also for practitioners. Although the main goal of theoretical computer science, and the use of formal methods, is to gain insights, it turns out that many of the methods have direct practical applications. We will make an attempt to always illustrate the theoretical ideas with concrete examples and case studies – which is sometimes easier and sometimes a bit harder.

Because the curriculum is toward applied informatics, we only provide an overview of the field – on each of the chapters, we could have an entire course itself (e.g. on complexity, on logic, on automata and languages, etc.). Thus, it is not possible to always dig very deeply into the subject matter, but we do hope that this is sufficient to appreciate the issues involved and their implications.

We will cover the classical topics – formal languages, automata theory, etc. – but we have added a number of subjects that reflect our background in artificial intelligence, artificial life, and complex systems. They all require clear, formal thinking, and provide an additional valuable set of tools. For example, we have included sections on cellular automata, L-systems, and fractals which we feel, every computer scientist should know about. We also bring in the relatively recent topic of network theory, which represents an extension of classical graph theory. Moreover, in the classical approach to artificial intelligence, the relation between computational models and theory of intelligence is often discussed; we will devote some final considerations to this question.

Theoretical computer science is built on the idea of computation as formalized by Alan Turing. While this idea of abstracting from the physical substrate and only considering the level of *computation* is extremely powerful – and has fundamentally changed society and our daily lives – it does have its limitations. This is why we also incorporated short chapters on *morphological computation* and on *quantum computing*, again concepts a computer science student should know about.

This script is only a short summary of contents treated in the lectures. Not all the material covered in class is covered here even though it may be relevant for the final examination. For certain topic areas, the students are referred to the pertinent literature for more detail.

Computer science – theoretical or applied – cannot be taught only theoretically, but active participation is extremely important. Consequently, we have included four substantial problem sheets that have the purpose to provide you with a deeper understanding of the ideas introduced. We do hope that the students will find them instructive and fun to do.

In terms of prerequisite, we expect the students to be familiar with the ideas taught during the course "Formal Methods, Part 1" which includes the following areas: information theory, Boolean algebra, predicate calculus, program verification, algorithmic complexity, trees, graphs, and relations. Some of these topics will be taken up again and deepened theoretically and through applications.

1.2 Acknowledgment

Rolf Pfeifer and his teaching assistants are grateful to Norbert Fuchs for generously providing materials and patiently answering our questions.

1.3 Outline of the Script

The topics which we will cover during the lecture – and thus the chapters of the present script – are structured and related to each other as follows (see Figure 1.1):

We will start with the **theory of formal languages**, which is about formalizing

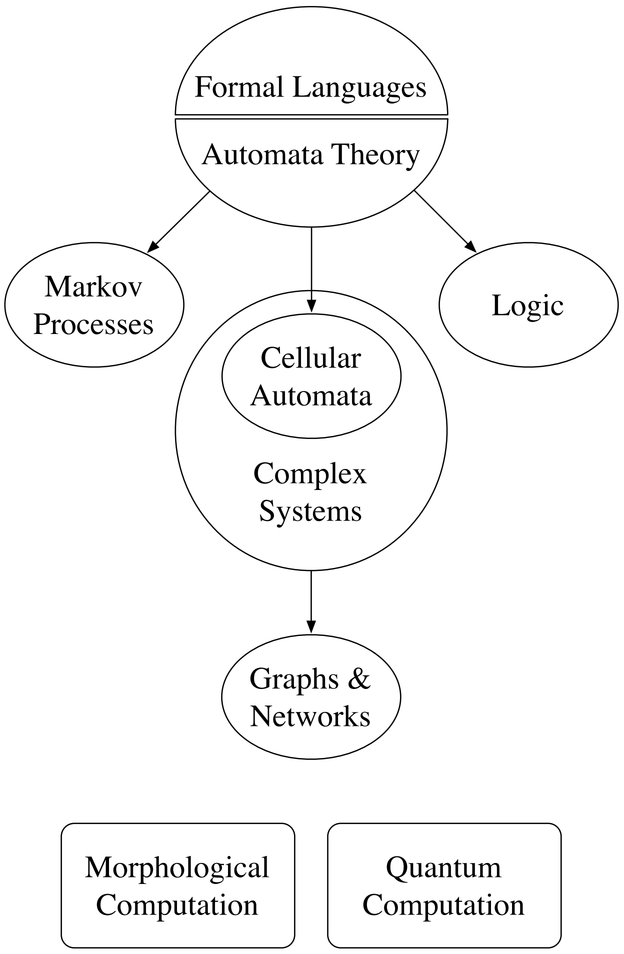

Figure 1.1: Overview of the topics that will be covered during the lecture.

the various structures found is virtually any text-based communication between any combination of humans and computers. We will not only learn the terminology, but also explore the various complexity levels found in formal languages.

The second topic, **automata theory**, is very closely related to formal languages: it deals with the various kinds of abstract machines capable of recognizing formal languages. In fact, the two topics of formal languages and automata theory constitute together the very basis of *computation*. Yet, this first part of the lecture is not only about theoretical computer science. We will also encounter various **applications**, many of which are found in the most modern software and devices.

We will explore different extensions of automata theory. The theory of **Markov processes** belongs to the probabilistic follows-up of automata theory, and has many applications, such as in text or speech recognition. **Logic** is also an extension of formal languages concerned with the question of how sentences in a formal language (such as mathematical theorems) can be logically derived from one another. We will also discuss how such a formalisms may allow computers to derive or proof theorems for us.

Another interesting extension of automata theory is the study of **cellular automata** – which are nothing else than many simple automata put on a grid next to each other. We will see that such systems have very surprising, even spectacular properties. In fact, we will see that cellular automata belong to a larger class of systems called **complex systems**.

An example of complex systems on which we will focus are **graphs and networks**. In this chapter, we will learn different techniques that enables us to deal with and characterize such kind of systems. We will also discuss some interesting effects found in different kind of static or growing networks.

Finally, we will conclude our journey through theoretical and applied computer science with two particular topics – **morphological** and **quantum computation** – which should provide insights on alternatives to the classical and prevalent theory of Turing computation.

1.4 Suggested Readings

Formal Languages and Automata Theory

- Hopcroft, J. E., Motwani, R., and Ullman, J. D. (2001). *Introduction to Automata Theory, Languages and Computation*. Addison Wesley, second edition.

- Sudkamp, T. A. (2006). *Languages and Machines: An introduction to the Theory of Computer Science*. Addison Wesley, third edition.

Logic and Models of Computation

- Papadimitriou, C. H. (1995). *Computational Complexity*. Addison Wesley.

- Rechenberg, P. and Pomberger, G. (2006). *Informatik Handbuch*. Hanser, fourth edition.

Fractals and Chaos

- Flake, G. W. (1998). *The Computational Beauty of Nature – Computer Explorations of Fractals, Chaos, Complex Systems, and Adaptation*. MIT Press.

Graphs and Networks

- Buchanan, M. (2002). *Nexus: Small Worlds and the Groundbreaking Science of Networks*. Norton Publishing. (Popular science)