

# Preface

## **Philosophical Orientation**

The general goal of this book is to develop an understanding of the ultimate and insurmountable limits of computing. In order to understand these limits better, we seek to view them from both sides — to understand both what can be computed and what is inherently impossible. These limits vary depending on the amount and character of the available computing resources. The boundaries of what's possible are explored for a variety of combinations of computing resources. To this end, a varied collection of computational models is considered. These models are formulated at a high level of abstraction. The purpose of this abstraction is to ensure that the analysis applies to *any* suitable “computer”. To achieve this level of abstraction, we develop precise (and machine independent) definitions, and follow them with rigorous deductions of the capabilities that are implied.

The material of this book introduces established formal models of computing. Developing the models involves understanding precise definitions, and rigorous reasoning based on these definitions. This precision and rigor provide a discipline that the author believes to be of great value to those working in computer science. On the other hand, formality cannot replace intuition. Insight and intuition are what provide the inspiration that guides technical development. However, intuition is by its very nature imprecise and therefore prone to error, and so an associated formalism is a necessity. A danger is that formality may obscure intuition. Formalism is developed in this book in the spirit of a ratification of the associated intuition, and the presentation weaves intuition and precision into a single fabric of insight and understanding. A significant strength of this book is that neither rigor nor intuition is sacrificed for the sake of the other. Rigor is developed as a *refinement of intuition*, and each is used to deepen the appreciation of the other.

This approach to understanding the inherent limitations of computation was initiated even before the invention of electronic computers! The profound work by Alan Turing provided an inspiration for the kind of deep insights that could be achieved through theoretical investigations. Turing was pursuing questions first raised in formal logic and his work pre-dated the development of general-purpose electronic computers. When John von Neumann was later formulating his ideas for these devices, he did interact with Turing. Nevertheless, the initial stages of development of electronic digital computers proceeded on an ad hoc basis. The subsequent recognition of computer science as a distinct intellectual discipline was initially largely motivated by theoretical studies along the lines presented in this book.

One of the features of this book is that in addition to the most familiar models, we introduce a number of variations and applications along the way. The variations are not essential to the main thread of the development, and sections that elaborate them can safely be omitted without loss of continuity. However, the selection of a few that attract the reader's interest can augment the main line of investigation and help achieve a deeper understanding. By exhibiting alternative models and examining their properties, we seek to shed light on what may appear to be arbitrary choices in the standard models. In addition, the brief digressions into a few practical applications are intended to keep the reader's motivation at a high level — a crucial ingredient in the learning process.

### **Comments for Readers**

Books that develop the basic theoretical ideas covered here usually fall into one of two categories. One approach is to begin with the most general models (e.g., Turing machines) and proceed through a series of restrictions on available resources and capabilities to simpler and simpler models, studying the reduction in the limits of computations that result. The second approach is the exact opposite, namely to begin with the simplest model (i.e., finite state machines) and consider a series of extensions of the computational resources and capabilities and the resulting expansions in computational power. The approach taken in this book is the latter. The finite state component persists throughout each of the enriched models of computing, and mastering the understanding of this aspect at the outset eases the way for the increasing complication. We take advantage of this in the presentation, and feel it aids in learning the material.

The material in this area is intellectually deep, and it is highly challenging to master. This book is unrelenting in its attention to the blending of intuition and rigor that is required to fully understand the area. This is one of the keys to achieving proficiency in this area. The intuitive perspective enters not only in the examples and discussions of the results, but also in the proofs. In fact, it is really in the formal proofs that intuition and rigor are most advantageously intertwined. The proofs are used not just to justify the results, but as a vehicle to create deeper insight.

This book emerged from notes and lectures resulting from many years of teaching the material. The level of the presentation assumes that the student is an upper-level undergraduate, or a beginning graduate student. The material presented covers all the topics suggested in the ACM curriculum guidelines for the Theory of Computation course<sup>1,2</sup>. The background assumed of the reader includes some general knowledge of computers and programming. We do not explicitly consider programs as such, although at a somewhat more abstract level the idea of algorithm is an emphasis of the book. However, for intuition and motivation, experience in programming computers is vital. The most crucial technical background is a good course in what is usually referred to in computer science curricula as “discrete structures”. In particular, some prior exposure to abstraction, rigor, and methods of formal proof is assumed. Especially the idea of inductive proof is heavily used. A brief synopsis of the most critical supporting material is given in Chapter 0.

In the author’s experience, working problems is an *essential* means of gaining a full understanding of the material. In this sense, there is a good analogy with learning computer programming — one can listen to lectures, read about programming, and study programs of others, but deeper understanding is not developed without actually writing programs yourself. So it is with the material in this book — to learn it well requires active participation, so work as many problems as you can. Therefore to this end, there are numerous problems at the end each chapter, and sample solutions

---

<sup>1</sup>A. B. Tucker (ed.), “Computing Curricula 1991, Report of the ACM/IEEE-CS Joint Curriculum Task Force”, ACM order number 201910 (full report); also, *Commun. ACM* 34, 6 (June 1991), 68-84 (summary).

<sup>2</sup>H. M. Walker & G. M. Schneider, “A revised model curriculum for a liberal arts degree in computer science”, *Commun. ACM* 39,12(Dec. 1996), 85-95.

for some are included in an appendix. The student is urged to attempt to solve a problem before consulting the sample solution. Also, note that these solutions are just samples and not the only correct ones, so if you devised a different solution, do not assume it is necessarily wrong or inferior.

### **Notational Conventions**

Headings of optional sections are marked with the symbol ‡. We also use the special symbol □ to denote the ends of proofs and examples, and by this means seek to avoid any confusion by the reader about when a technical portion has been completed and where follow-up discussion about it begins. The problems with sample solutions are marked with <sup>□</sup>, and the most challenging problems are marked with the symbol \*.

### **Electronic Links**

The author's Web page (<http://www.cs.uiowa.edu/~fleck/>) contains a link to material related to the book, including an up-to-date errata. The author invites readers to email a report ([fleck@cs.uiowa.edu](mailto:fleck@cs.uiowa.edu)) of any errors they detect.

### **Acknowledgements**

I am appreciative of the reactions and questions of many past students for the improved presentation that has resulted. Also, my thanks go to Sungwon Kang and Hantao Zhang for their helpful comments on earlier versions of the manuscript. Finally, the enthusiastic support of Teodor Rus for getting this book into print is gratefully acknowledged.

Arthur C. Fleck  
Iowa City, Iowa