Introduction

The mathematical theory of computing grew out of the quest of mathematicians to clarify the foundations of their field. Turing machines and other models of computation are tied to formal logical systems and algorithmic unsolvability to unprovability. Problems whose solutions lead to this understanding have included Hilbert's Entscheidungsproblem (deciding the logical validity of a first-order formula) or his program of establishing by finitary means the consistency of mathematics. This development would not have been possible without the prior advent of mathematical logic in the late nineteenth century, guided by contributions from a number of people. The most important was, in retrospect, Frege's *Begriffsschrift* [188] realizing facets of an old dream of Leibniz about the *calculus ratiocinator*.

Eventually the problems received answers opposite to those that Hilbert and others had hoped for (the Entscheidungsproblem was found to be algorithmically undecidable by Church [143] and Turing [494], and the fact that a proof of the consistency of a theory formalizing a non-trivial part of mathematics requires axioms from outside the theory, Gödel [200]). But the leap in the development of logic that this study of the foundations of mathematics stimulated was enormous compared with the progress of the previous two millennia. All this is discussed and analyzed in a number of books; I have found most stimulating the volume [224], edited by J. Van Heijenoort, of translations of selected papers from the era accompanied by knowledgeable commentaries.

The next significant step from our perspective was the formalization of an informal notion of a feasible algorithm by the formal notion of a polynomial time algorithm in the late 1960s. This was accompanied by the introduction of complexity classes, reductions among problems and discoveries of natural complete problems, all quite analogous to the earlier developments in mathematical logic around Turing computability. A plethora of open questions about the new concepts were posed. The most famous of them, the P vs. NP problem, is now recognized as one of the fundamental problems of mathematics.

Some of these new problems were, in fact, foreshadowed by specific problems in logic which did not explicitly mention machines or time complexity. These include

the spectrum problem in the model theory of Scholz [462] and Asser [25] (in the early 1950s), which, as we now know, asks for a characterization of NE sets (i.e. non-deterministic time $2^{O(n)}$ sets) and, in particular, whether the class NE is closed under complementation. They also include problems involving the rudimentary sets of Smullyan [476] and Bennett [65] (in the early 1960s). The first problem explicitly mentioning time complexity was perhaps the problem posed by Gödel to von Neumann in a 1956 letter [201]; there he asks about the time complexity of a problem we now know to be NP-complete (see the introduction to the volume of Clote and Krajíček [144] for the letter and its translation, and Buss [111] for the completeness result). Sipser [468] describes these developments clearly and succinctly.

In the light of this history it seems plausible that the P vs. NP problem has significant logical facets connected to the foundations of mathematics. It can even be seen as a miniaturization of the Entscheidungsproblem: replace first-order formulas by propositional formulas and ask for a feasible algorithm rather than just any algorithm. We do not know how hard the problem is. It may require just one trick (as some famous problems of the past did), one dramatically new idea (as was forcing for the independence of the continuum hypothesis) or a development of a whole part of mathematics (as was needed for Fermat's last theorem). In either case it seems sensible to try to develop an understanding of the logical side of the problem.

This thinking is, however, foreign to the canons of contemporary computational complexity theory. That field developed as a part of combinatorics and some early successes stimulated a lot of faith in that approach. However, although many interesting developments have taken place, progress on combinatorial insight into the original fundamental problems remains tentative. Despite this, complexity theorists sometimes go to great lengths to talk about logic without ever mentioning the word. Although it is remarkable that one can discuss provability or unprovability without using words like "a theory" or "a formula," it is hard to imagine that doing so is faithful to the topic at hand.

There are several research areas connecting mathematical logic and computational complexity theory; some of these connections are tighter than others. We shall concentrate on propositional proof complexity. Proof complexity (the adjective propositional is often left out, as it is tacitly understood) is now a fairly rich subject drawing its methods from many fields (logic, combinatorics, algebra, computer science, ...). The first results appeared in the 1960s, with a significant acceleration from the late 1980s to the early 2000s. For some years now, however, progress has lain more in sharpening and generalizing earlier advances and recasting them in a new formalism and in improving the technical tools at hand, but not in fact that much (at least not at the level of the progress in the 1990s) in developing completely new ideas. It thus seems a good time to write an up-to-date presentation of the field to enable newcomers to learn the subject and to allow active researchers to step back and contemplate a larger picture.

There are two approaches to proof complexity, which are complementary, in a sense, to each other. The first views problems, especially length-of-proof problems,

as purely combinatorial questions about the existence and properties of various combinatorial configurations. Prominent early examples of this approach are Tseitin's [492] 1968 lower bound for regular resolution or Haken's [216] 1985 lower bound for general resolution proofs of the pigeonhole principle. This approach, when successful, typically gives qualitatively very detailed results. It works only for weak proof systems however.

The second approach views problems through the eyes of mathematical logic. Representative examples of this view are Cook's [149] 1975 and Paris and Wilkie's [389] 1985 translations of bounded arithmetic proofs into (sequences of) propositional proofs, Ajtai's [5] 1988 lower bound, obtained by forcing, for pigeonhole-principle proofs in constant-depth Frege systems or my 1991 idea of the feasible interpolation method formulated in [276] (1994, preprint circulating in 1991). This approach typically yields a general concept, and its applications in specific cases require combinatorics as well.

The two approaches are not disjoint and can be fruitfully combined. Nevertheless, there has been much emphasis in recent years on a purely combinatorial analysis of proof systems related to specific SAT-solving or optimization algorithms. Only a small number of researchers have involved themselves in mathematical logic or investigations aimed at explaining some general phenomena. Of course, the original fundamental – logic-motivated – problems are dutifully mentioned when the goals in this area are discussed in talks or papers, but in actual research programs these problems are rarely studied. It seems that not only are the original problems abandoned but some of the proof complexity theory developed around them is becoming exotic to many researchers. I suppose this is due in part to the demise of logic from a standard computer science curriculum. Being able to formulate a theorem or a proof does not provide sufficient education in logic, just as being able to draw a graph is not a sufficient education in combinatorics. Without at least an elementary logic background the only part of proof complexity that remains accessible is the algorithm analysis aspect. Of course, it is a perfectly sound research topic but it is not the whole of proof complexity; it is not even close to half of it.

I think that logic will play eventually a greater role also in proof complexity research motivated by applications. I am not competent to judge how significant a contribution current proof complexity makes to the analysis and practice of real-life algorithms. Such contributions often contain ingenious technical innovations and ought to shed light on the performance of the respective algorithms. But SAT solving (or the design of other algorithms) is mostly an engineering activity, with little theory, as we mathematicians understand it, involved. Even the basic issue of how to compare the efficiency of two algorithms is in reality approached in an ad hoc, heuristic, mathematically non-principled way. Therefore I think that proof complexity could contribute most by introducing new original theoretical concepts and ideas that could influence how the whole field of SAT solving is approached in future. A significant contribution may come even from technically simple ideas, for example, how to use stronger proof systems in proof search or how to combine the

present-day optimization algorithms with a little logic. This could be, in my opinion, quite analogous to the fact that fairly general ideas of Turing and others influenced the thinking in computer science significantly. But from where else could genuinely new concepts and ideas, and not just technical innovations, come other than from considering the very foundational problems of the field?

In the book I stress my view of proof complexity as a whole entity rather than as a collection of various topics held together loosely by a few notions. The frame that supports it is logic. This is not a dogma but represents the current state of affairs. For this reason I choose as the motto at the start of the book a paraphrase of the famous sign on the gate to Plato's Academy, in order to underline the point, not to distress the reader.

Basic concepts, classical results, current work and possible future directions are presented. I have not attempted to compile a compendium of all known results, and have concentrated more on ideas than on the technically strongest statements. These ideas might be enlightening definitions and concepts, stimulating problems or original proof methods. I have given preference to statements that attempt some generality over statements that are strong in specific situations only.

The intended readers include researchers and doctoral students in mathematics (mathematical logic and discrete mathematics, in particular) and in theoretical computer science (in computational complexity theory, in particular). I do not necessarily present the material in the most elementary way. But I try to present it honestly, meaning that I attempt to expose the fundamental ideas underlying the topics and do not hide difficult or technically delicate things under the carpet. The book is meant to be self-contained, in a reasonable sense of the word, as a whole; it is not a collection of self-contained chapters and there is a lot of cross-referencing.

Organization of the Book

The book is divided into four parts. In Part I, "Basic Concepts," the first chapter recalls some preliminaries from mathematical logic and computational complexity and introduces the basic concepts and problems of proof complexity. The remaining six chapters in Part I give a number of examples of propositional proof systems and we prove there many of their fundamental properties.

The main topics of the second part, "Upper Bounds," are the relations between bounded arithmetic theories and proof systems and translations of arithmetic proofs into propositional logic, and how these can be utilized to prove length-of-proof upper bounds and simulations among proof systems.

The third part, "Lower Bounds," is devoted to known length-of-proof lower bounds. This is the royal subject of proof complexity, as the fundamental problems are formulated as questions about lengths of proofs. Some people reduce proof complexity to this topic alone.

The last part, "Beyond Bounds," discusses topics transcending the length-of-proof bounds for specific proof systems. I report on those developments attempting to

address all proof systems. It is here where there is, I think, a chance to develop some deeper understanding of the fundamental problems. I present there topics for which some general theory has emerged in the past and discuss directions that seem to me to be promising for future research. In the last chapter we step back a little, look at what has been achieved and offer a few thoughts on the nature of proof complexity.

Each part is divided into chapters, which further divide into sections with, I hope, informative titles. This gives a structure to the material that should be apparent at first glance. There are various novel results, topics, constructions and proofs.

In the main narrative, typically I attribute to sources only the main results, concepts, problems and ideas discussed. The qualification *main* is subjective, of course, and does not necessarily refer only to the technically hardest topics but sometimes also to simple observations that have proved to be very useful: I rather agree with the classical dictum that *the really important ideas are also simple*. Each chapter ends with a section on bibliographical and other remarks, where I attempt to give full bibliographical information for all the results covered, point to other relevant literature and discuss at least some topics that are related but were omitted.

Occasionally I use the adjectives *interesting* and *important* and similar qualifiers. This should be understood as meaning *interesting* (or *important*) *for me*. In fact, it is a dangerous illusion, to which some people succumb, that these words have an objective meaning in mathematical literature.

Conventions and Notation

Throughout the book we use the standard symbols of propositional and first-order logic. Formulas are denoted either by capital Latin letters or by lower-case Greek letters. We also use the O-notation and the related o-, Ω- and ω-notations, which are very handy for expressing upper and lower bounds in complexity theory. In particular, for $f, g\colon \mathbf{N} \to \mathbf{N}$, $g = O(f)$ means that $g(n) \leq cf(n) + c$ for some constant $c \geq 1$ and all n, $g = o(f)$ means that $g(n)/f(n)$ goes to 0 as $n \to \infty$, $g = \Omega(f)$ means that $g(n) \geq \epsilon f(n)$ for some constant $\epsilon > 0$ and all n and finally $g = \omega(f)$ means that $g(n)/f(n)$ cannot be bounded by a constant for all n.

The symbol $[n]$ denotes the set $\{1, \ldots, n\}$. The symbols $\subseteq$ and $\subsetneq$ denote set-theoretic inclusion and proper inclusion, respectively. We use $n \gg 1$ as a shorthand for *n is large enough*. Most first-order objects we consider are finite words (ordered tuples) over an alphabet (numbers, binary words, formulas, truth assignments, proofs, etc.). When it is important that a variable ranges over tuples and we need to address their elements, I use the notation $\overline{b}, \overline{x}, \ldots$ and denote the elements at the ith position $b_i, x_i, \ldots$

Only a few notions and concepts that are exceptionally important in proof complexity have their numbered definitions; all other notions (the vast majority) are defined in the text and the defined notion appears in boldface. To refer to the latter definitions, we use the section number where they appeared (the sections are fairly short and it is easy to navigate them). I think that this treatment of definitions

makes the structure of a long text more comprehensible. All lemmas, theorems and corollaries are syntactically distinguished and numbered in the conventional way.

All unfamiliar symbols are properly defined in the text and the special-symbol index near the end of the book lists for each such symbol the section where it appears first.

Remarks on the Literature

During the last ten years only two books about proof complexity have appeared: [155] by Cook and Nguyen and my text [304]. The Cook–Nguyen book develops various bounded arithmetic theories and their relations to weak computational complexity classes utilizing witnessing theorems, and defines associated propositional proof systems simulating the theories. The emphasis there is on the triple relation *theory/computational class/proof system*. However, no lower bounds are presented in that book, either old or more recent. My book [304] is a research monograph developing a particular construction of models of bounded arithmetic and how it can be applied to proof complexity lower bounds, but it reviews only snapshots of the proof complexity needed.

There is also a related book [421] by P. Pudlák about the foundations of mathematics. It contains one chapter (out of seven) about proof complexity (and one of its sections is on lower bounds), offering a few highlights. It is written in three levels of precision, with the top level giving details.

From older books the most used is perhaps my 1995 monograph [278], which presented the field in an up-to-date manner (relative to 1995). It may occur to the reader that it might have been more sensible to update that book rather than to write a new one. However, now I am putting a lot more emphasis on propositional proof systems while at least half the 1995 book was devoted to a development of bounded arithmetic theories, and only a smaller part of the latter is directly relevant to propositional proof complexity.

There is also an excellent book [145] from 2002 by P. Clote and E. Kranakis, which does contain one chapter (out of seven) about proof complexity lower bounds. It contains some results (and lower bounds, in particular) not covered in my 1995 book.

Handbook of Proof Theory (Elsevier, 1998) edited by S. R. Buss, contains a chapter on lengths of proofs by Pudlák [413]. The same volume includes introductory texts [115, 116] about first-order proof theory and theories of arithmetic, and we recommend those texts to all readers who need to learn that background. There are also a number of survey articles, usually addressing just a few selected topics; these include [497, 279, 281, 58, 117, 48, 297, 418, 120, 442].

Background

Although I review some elements of propositional and first-order logic in the first two sections of Chapter 1, this is mostly to set the scene and to introduce some formal-

ism and notation. It is helpful if the reader has a basic knowledge of mathematical logic at the level of an introductory course in first-order logic and model theory. This should include the completeness and compactness theorems and Herbrand's theorem (although I do present a proof of this theorem). I do not assume any specific knowledge of bounded arithmetic or the model theory of arithmetic.

From computational complexity theory I assume a knowledge of the basic concepts of Turing machines and their time complexity and of the definitions of standard complexity classes (although some of this is briefly recalled in Section 1.3). I do not assume a particular knowledge of circuit complexity (it is explained when needed) but it could make some ideas and arguments more transparent.