1
Introduction

We shall study a particular class of propositional tautologies that seem to be good candidates for being hard for strong and possibly for all propositional proof systems. The formulas are called **τ-formulas** or alternatively **proof complexity generators**. They were defined by me [49] and independently by Alekhnovich et al. [5]. I shall describe my motivation for introducing these formulas later in the Introduction. The motivation of [5] was apparently different.

In the intervening 20+ years, a theory was developed around these formulas. Unfortunately, the authors of [5] abandoned the idea and – with the notable exception of [98] which was, however, written already in 2002/03 – did not contribute to it further. I regret this as the different perspective they seemed to have would undoubtedly enrich the theory. Be that as it may, the bulk of the theory was developed over the years in 14 papers of mine [49], [50], [51], [52], [54], [56], [57], [58], [61], [62], [66], [67], [68], [69] (some devoted to the topic entirely, some only in part) and in [60, Chpts. 29 to 31]. My student J. Pich contributed with his thesis [87], and more recently other people started to chip in.

These lecture notes present the theory around τ-formulas in a unified manner. I hope this will enable other researchers to learn its basic ideas and to contribute ideas of their own. Or that it will stimulate them to come up with an entirely different approach. Of course, it is a conjectural enterprise: we cannot be sure that the formulas are indeed hard and, even if they are, whether we will ever be able to prove their hardness. But if we don't even try, we will not get anywhere anyway. In any case, there is no other proposal (other than the reflection principles mentioned in the Preface) on the table supported by some nontrivial theory.

My motivation for introducing the formulas was a logic question about the dual weak pigeonhole principle (dWPHP) for p-time functions in a

weak bounded arithmetic theory S^1_2. Let me start with presenting briefly its background.

Bounded arithmetics are weak subtheories of Peano arithmetic, which relate to classes of functions with a restricted computational complexity analogously to the classical relation between subtheory $I\Sigma_1$ of PA (with induction restricted to recursively enumerable sets) and the class of primitive recursive functions. Feasible algorithms find it hard to count the number of elements of a finite set, and formalizing counting arguments in bounded arithmetic is similarly difficult. A. Woods [106] discovered that in such formalizations explicit counting may be often replaced by the PHP for bounded formulas, denoted Δ_0PHP. This statement says that no Δ_0-formula defines the graph of a function mapping $[0,a]$ injectively into $[0,a-1]$. It is still unknown whether Δ_0PHP is provable in bounded arithmetic (Macintyre's problem). Subsequently Paris, Wilkie and Woods [84] noted that a weaker version of PHP, the weak PHP denoted WPHP, can be often used instead and, crucially, that this principle is provable in bounded arithmetic (they used theory $I\Delta_0+\Omega_1$, extending the original theory of [82] by the Ω_1 axiom). The principle says that no bounded formula defines the graph of a function mapping $[0,2a-1]$ injectively into $[0,a-1]$. Around that time, Buss [10] defined his version of bounded arithmetic, theory S_2 (a conservative extension of $I\Delta_0+\Omega_1$) and its most important subtheory S^1_2, and proved that p-time functions are exactly those functions with $\mathcal{NP}$ graphs (represented by Σ^b_1-formulas) that are provably total in S^1_2.

Let us denote by dWPHP(f) the statement that the function f cannot map any interval $[0,a-1]$ onto $[0,2a-1]$,

$$\exists y < 2a \forall x < a\ f(x) \neq y, \tag{1.1}$$

and, following [49], denote the theory obtained by adding to S^1_2 all instances of dWPHP(f) for all p-time functions f by BT:

$$\mathrm{BT} \ := \ S^1_2 + \mathrm{dWPHP}(\Delta^b_1). \tag{1.2}$$

Functions f in the dWPHP scheme are allowed to have parameters but, in fact, it suffices to consider f without extra parameters, that is, depending only on x (see more about this in Section 2.1).

The development directly leading to my problem below was a theorem by A. Wilkie (the proof is in [45, 7.3.7]) that functions Σ^b_1-definable in BT are computable in randomized p-time. I realized that one ought to be able to use BT for formalizing randomized algorithms and to relate this theory to randomized p-time analogously to how S^1_2 relates to deterministic p-time. (I was rather excited by this idea and named the theory BT for basic theory.) This also led me to formulate the following problem.

Problem 1.0.1 (Conservativity problem [49, Problem 7.7]) Is BT Σ^b_1-conservative over S^1_2?

We shall discuss this in some detail in Chapter 2.

At that time E. Jeřábek was starting his PhD studies with me. Knowing his exceptional mathematical talent, I decided not to waste his time on some peripheral topic, and I proposed that he develop this conjectured relation between BT and randomized p-time. His PhD thesis and a subsequent series of papers [34], [35], [36], [37] are the most interesting things to have happened in bounded arithmetic over the last at least 20 years.

In order not to compete with his work, I decided to focus on the conservativity problem above and on the related propositional logic side of things, and this led me to proof complexity generators. They will be introduced in Chapter 3.

1.1 Prerequisites

The topic covered in these notes is a fairly advanced part of proof complexity, using concepts, methods and results from a large part of the field, as well as some more basic mathematical logic and computational complexity theory. This is not a textbook on either of these fields. We assume that the reader has a solid background in proof complexity, including basics of bounded arithmetic. It is unfeasible to review the necessary material here but the reader can find essentially all of it in [65] (and some bounded arithmetic facts in [45]; see also [18]). Chapter 2 can serve as an entrance test: it discusses a couple of key bounded arithmetic theories, some witnessing theorems, propositional translations, and some properties of strong proof systems.

Earlier abbreviated expositions of theory are in [60, Chpts. 29 and 30] and in [65, Sec. 19.4] but these are not prerequisites.

1.2 Content

The reader we had in mind while writing the text was a researcher (junior or senior) in proof complexity, or in closely related areas of computational complexity theory and mathematical logic, who wants to learn the theory around proof complexity generators in depth. We thus develop the theory systematically step-by-step (including the specific notation and terminology), and the text is meant to be read sequentially, starting at the beginning. In particular, later chapters use a lot of material covered in earlier chapters. Sampling and reading random parts may thus prove difficult.

Chapter 2 examines the dWPHP problem, asking whether $S^1_2(\mathrm{PV})$ equals BT, which is a (presumably) simpler version of the conservativity problem 1.0.1. This leads in Chapter 3 to the definition of central notions of the theory: proof complexity generators and τ-formulas, the hardness and the pseudosurjectivity, and two conjectures motivating much of the subsequent development.

Chapter 4 treats the issue of the output/input ratio and its relation to the Kolmogorov complexity and to the general compression/decompression issue. Three examples of proof complexity generators are presented in Section 4.3 and in Chapters 5 and 6, together with various results about them.

Chapter 7 studies the pivotal case of extended Frege systems. Chapter 8 establishes the consistency (with particular bounded arithmetic theories) of some statements related to the dWPHP problem and to the conjectures discussed in the earlier chapters, using proof-theoretic analysis (witnessing theorems) as well as some model theory. Chapter 9 overviews several topics outside proof complexity to which the theory of proof complexity generators (or ideas developed in the theory) relates in some nontrivial way. Chapter 10 discusses possible avenues for further research.

Not all of the concepts, results and problems discussed in the book appeared in the papers mentioned at the beginning of this introduction; most chapters contain new ideas and results.

The book ends with a list of special symbols and a general index. We do not have a name index, but instead each item in the References lists the page numbers where it is cited.

1.3 Notation, Terminology and Conventions

Some common notations have fixed meanings:

- $i < n$: i is an integer and runs over $0, 1, \ldots, n-1$
- $i \in n$: the same as $i < n$
- $[n]$: the set $\{1, \ldots, n\}$

Special Symbols lists all symbols, and recalls their definitions, in – roughly – their order of appearance.

We abbreviate *propositional proof systems* to just *proof systems*. Two expressions that are usually used informally (or defined each time ad hoc) will get specific technical definitions:

- *generator*: see Definition 3.1.2,
- *strong proof system*: see Definition 2.4.3.

We denote a tuple (of bits, variables, etc.) by a letter without the overline, its coordinates with indices, and elements of a tuple of tuples are distinguished by superscripts. For example, we may write $b \in \{0,1\}^m$ and b_i for the ith bit of b, and $(b^1, \ldots, b^t)$ for a t-tuple of strings from $\{0,1\}^m$. It eases on the notation and does not seem to lead to any confusion.