Proof complexity generators

(version v5a - February 2024)

Jan Krajíček

Faculty of Mathematics and Physics
Charles University

1

To my family

Contents

Preface

Proof complexity (tacitly propositional) has a number of facets linking it with mathematical logic, computational complexity theory, automated proof search and SAT algorithms and other areas, and there are many open problems. The royal subject is the task - still open - to establish lengths-of-proofs lower bounds for strong and possibly for all proof systems. This is *the fundamental* open problem as establishing super-polynomial lower bounds for all proof systems is equivalent to showing that the computational class $\mathcal{NP}$ is not closed under complementation, and establishing lower bounds at least for a particular proof system implies the consistency of $\mathcal{NP} \neq \mathrm{co}\mathcal{NP}$ with a first-order theory of arithmetic associated with the system.

For some specific proof systems strong lower bounds are known. The experience with these lower bounds shows that it is instrumental to have plausible candidates for hard tautologies with a clear combinatorial or logical meaning. To define such hard formulas is difficult and one reason for this is the close relationship between proof systems and first-order theories alluded to above.

There are at present only two classes of such formulas known that are supported by some non-trivial theory: reflection principles and proof complexity generators, also known as τ-formulas. The former is a classic topic of proof complexity that is treated in literature in details. The theory supporting the latter class is on the other hand spread over a number of papers and even proof complexity experts do not seem to be aware of its main points. It is the purpose of these notes to present the underlying theory as a coherent whole.

Acknowledgments

The first version of the book was written while I enjoyed a six-month sabbatical from the Faculty of Mathematics and Physics of the Charles University during August 2023 - January 2024.

I am indebted to Jan Pich (Oxford) for offering numerous comments on the draft, to Hanlin Ren (Oxford) for pointing out a problem with the original version of Section 6.5, . . .

Chapter 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**. The formulas were defined by K.[49] and independently by Alekhnovich et al. [5]. I shall describe my motivation for introducing these formulas below. 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 notable exception of [97] which was, however, written already in 2002/03 - did not contribute to it further. I regret this as a different perspective they seemed to have would undoubtedly enrich the theory. Be 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.29-31]. My student J.Pich contributed in 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, if we will ever be able to prove their hardness. But without even trying we will not get anywhere anyway. In any case, there is no other proposal on the table supported by some non-trivial knowledge.

My motivation for introducing the formulas was a logic question about the dual weak PHP 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 r.e. sets and the class of primitively 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 [105] discovered that in such formalizations explicit counting may be often replaced by the pigeonhole principle PHP for bounded formulas, denoted Δ_0PHP. This statement says that no Δ_0-formula defines the graph of a function mapping $[0, a + 1]$ injectively into $[0, a]$. 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 Δ_0WPHP, 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 $[0, 2a]$ injectively into $[0, a]$, 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 function f cannot map any interval $[0, a]$ onto $[0, 2a]$:

$$\exists y < 2a \forall x < a,\ f(x) \neq y \tag{1.0.1}$$

(f may have other arguments than just x) 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:

$$\text{BT} \ := \ S^1_2 + \text{dWPHP}(\Delta^b_1) \tag{1.0.2}$$

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

A development directly leading to my problem below was a theorem by A.Wilkie (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 lead 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 it 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 to him to develop this conjectured relation between BT and randomized p-time. His PhD Thesis and a subsequent series of papers [34, 35, 36, 37] is the most interesting thing that happened in bounded arithmetic during the last at least twenty years.

In order not to interfere with his work I decided to focus on the provability/conservativity problem above and on the related propositional logic side of things, and this lead 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 text-book of 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 the theory are in [60, Chpts.29-30] and in [65, Sec.19.4]; their knowledge is not required here.

1.2 Content

Chapter 2 examines the dWPHP problem. This leads in Chapter 3 to the definition of central notions of the theory: proof complexity generators and τ-formulas, the hardness and the pseudo-surjectivity, and to two conjectures motivating a lot of the subsequent development.

Chapter 4 treats the issue of the output/input ratio and its relation to the Kolmogorov complexity and to 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 basic 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) and 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) relate in some non-trivial way. The last Chapter 10 discusses possible avenues for further research.

The book ends with a general index and with an index of special symbols. We do not have a name index but instead each item in the Bibliography is attached a list of 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 $0, 1, \ldots, n-1$
- $i \in n$: same as $i < n$
- $[n]$: the set $\{1, \ldots, n\}$

The Special symbol index 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 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 i-the 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.

Chapter 2

Background: the dWPHP problem

A silent prerequisite for the Conservativity problem 1.0.1 was the negative answer to the following question.

Problem 2.0.1 (The dWPHP problem)
Does S^1_2 prove the dWPHP for all p-time functions, i.e. $S^1_2 = BT$?

The quantifier complexity of the instances of dWPHP(Δ^b_1) is $\forall\Sigma^b_2$ and hence showing its unprovability may be, in principle, easier than proving the non-conservativity.

It is convenient to expand the language of S^1_2 by adding symbols for all clocked p-time algorithms and adding also as additional axioms all axioms of theory PV_1: the universal theory whose axioms are universal formulas codifying how algorithms are defined one from another using Cobham's [16] limited recursion on notation and composition, and adding also axioms of induction for open formulas. The resulting theory is denoted $S^1_2(\mathrm{PV})$. It is fully conservative over S^1_2 and $\forall\Sigma^b_1(\mathrm{PV})$-conservative over PV_1. Theories PV_1 and $S^1_2(\mathrm{PV})$ are different unless $\mathcal{NP} \subseteq \mathcal{P}/poly$. These are classic notions and results of bounded arithmetic stemming from [17, 10, 11, 75, 43] and can be all found in [45].

Using the expanded language we can formulate the dWPHP problem using the formula dWPHP(f) as defined in (1.0.1) as

- *Does $S^1_2(PV)$ prove dWPHP(PV), i.e. all formulas dWPHP(f) for all function symbols f in the language?*

Let us note that the dWPHP problem is open for theory PV_1 too. We shall discuss the problem more in Chapter 8.

The dWPHP problem has several facets which we shall discuss in the next three sections. Links to computational and proof complexity are fostered by witnessing methods and by propositional translations of proofs of Π^b_1-formulas in a theory. Both these connections are very general and provide a triangle correspondence among theories, proofs systems and computational complexity classes. We shall restrict only to the cases of $S^1_2(\mathrm{PV})$ and PV_1. The reader can find a general treatment in [45], see also [65] for the translations.

2.1 Logic: provability and axiomatization

One of main motivations for the dWPHP problem 2.0.1 was also the fundamental problem of bounded arithmetic, namely the **Finite axiomatizability problem**:

- *Is full bounded arithmetic S_2 finitely axiomatizable?*

In particular, is $S^1_2 = S_2$? The scheme dWPHP(PV) seemed to be a good candidate to separate these two theories (but the dWPHP problem turned out to be very hard). The reader can find (essentially) all known results about the finite axiomatizability problem in [45].

Let f be a PV function symbol and think in the dWPHP(f) formula about the parameter as $a := 2^n$. The formula (1.0.1) then says that f does not map $\{0,1\}^n$ onto $\{0,1\}^{n+1}$. For any specific domain $\{0,1\}^n$ the function f is computed by a circuit, say C. If f depended just on x the size of C would be $n^{O(1)}$. But the symbol f may have other arguments than just x; for example, $f(x,y)$. Picking some specific $y := e$ may thus force the size of C to be bigger.

On the other hand, if we have a function g computed by a family of circuits $\{C_n\}_n$ (not necessarily of polynomial size) the dWPHP for g follows from the instance of the principle for the p-time **circuit value function** $CV(x,y)$ that evaluates circuit y on input x. Namely, picking $y := C_n$ implies that CV satisfies dWPHP on $\{0,1\}^n$ iff g does. This gives the following statement pointed out in [35].

Theorem 2.1.1

BT is axiomatized over S^1_2 by the instance of dWPHP for the circuit value function CV.

Now we turn to the $\forall\Sigma^b_1(\mathrm{PV})$-consequences of BT. It can be analyzed using Herbradization as in [35] or via model-theory as in [14, 15]. Consider the following principle $\mathrm{dWPHP}_1(f, g)$:

$$\exists y < 2a,\ g(y) \geq a \vee f(g(y)) \neq y \tag{2.1.1}$$

formalizing that f, g is not a pair of a function f violating dWPHP and a function g that is its inverse (both functions may have additional parameters). Clearly all instances of $\mathrm{dWPHP}_1(\mathrm{PV}, \mathrm{PV})$ follow over $S^1_2(\mathrm{PV})$ from dWPHP(PV). The next statement is a form of a converse.

Theorem 2.1.2 ([35, Cor.4])
Any $\forall\Sigma^b_1(PV)$-consequence of BT is implied over $S^1_2(PV)$ by the axiom $dWPHP_1(CV, CV)$, the instance of $dWPHP_1(f, g)$ for both f, g being the circuit value function (with different parameters).

Corollary 2.1.3

1. *The dWPHP problem 2.0.1 has the negative answer, i.e. $S^1_2 \neq BT$ iff S^1_2 does not prove formula $dWPHP(CV)$.*
2. *The conservativity problem 1.0.1 has the negative answer, i.e. $S^1_2 \not\preceq_{\Sigma^b_1} BT$ iff S^1_2 does not prove formula $dWPHP_1(CV, CV)$.*

Let us note that all statements in this section hold also if S^1_2 is replaced by PV_1, i.e. also BT gets replaced by $\mathrm{PV}_1 + \mathrm{dWPHP}(\mathrm{PV})$. This last theory is called APC_1 and [36] used it instead of (possibly) stronger BT to formalize approximate counting methods.

The functions entering the dWPHP scheme (or dWPHP^1) are allowed to have parameters. But, in fact, parameters are not needed if we work over $S^1_2(\mathrm{PV})$.

Theorem 2.1.4

1. *For every p-time function f with parameters there is a p-time function g without parameters such that $S^1_2(PV)$ proves the implication:*

$$dWPHP(g) \to dWPHP(f)\ . \tag{2.1.2}$$

2. *There is one p-time function g without parameters such that $S^1_2(PV)$ proves (2.1.2) for all pt-ime f (with parameters).*

We stated the first part separately although it is implied by the second one as its proof in [102, L.3.8] is simpler than the proof in [34, 35] of the second part. The function featuring in the second part is the truth-table function we shall introduce and discuss in Section 4.3. Let us note that it is unknown (and unlikely by results in Chapter 8) whether this theorem holds also for PV_1 or T_{PV}.

It is occasionally suggested that because BT (or APC_1) are related to randomized computations while $S^1_2(\mathrm{PV})$ (or PV_1) to p-time computations one ought to expect - in an analogy with the hypothesis of universal derandomization - that $S^1_2(\mathrm{PV}) = \mathrm{BT}$ (or $\mathrm{PV}_1 = \mathrm{APC}_1$). This analogy is fallacious: the theories correspond to the classes of functions via their $\forall\Sigma^b_1$-consequences (see Section 2.2) while both BT and APC_1 are $\forall\Sigma^b_2$-axiomatized. The fallacy of the analogy is clearly seen at the following example: both PV_1 and $S^1_2(\mathrm{PV})$ correspond to p-time functions but are different unless $\mathcal{NP} \subseteq \mathcal{P}/poly$, cf. [75].

Let us also remark that it follows from these statements that $\mathcal{NP}$- and Σ^p_2-search problems definable in BT or APC_1 can be reduced to the search problems determined by $\mathrm{dWPHP}_1(CV, CV)$ and $\mathrm{dWPHP}(CV)$, respectively. What *reduced* means exactly depends on whether S^1_2 or just PV_1 (or even a weaker theory) was used as the base theory. We shall return to search problems briefly in Section 9.5.

2.2 Computational complexity: witnessing

The link from theories, in our case bounded arithmetics, to computational complexity is provided by witnessing theorems. In general they assert that if a theory T proves a statement of the form $\forall x \exists y A(x, y)$ with A from a syntactic class Γ then there is a function f in a computational complexity class $\mathcal{C}$ that witnesses the statement:

$$\forall x,\ A(x, f(x))\ .$$

For example, for T being PV_1 or S^1_2 and $\Gamma = \Sigma^b_1$ the class $\mathcal{C}$ can be just the class of p-time functions; for PV_1 this is a simple consequence of Herbrand's theorem, for S^1_2 this is Buss's theorem. In fact, Buss's theorem can be used to prove that S^1_2 is $\forall\Sigma^b_1(\mathrm{PV})$-conservative over PV_1, cf.[10, 45].

An immediate consequences of these witnessing theorems is the following statement.

Corollary 2.2.1

Assume BT is $\forall\Sigma^b_1$*-conservative over* S^1_2*. Then any formula from the class* $dWPHP_1(PV, PV)$ *can be witnessed by a p-time function.*

It is easy that we can witness formula $\mathrm{dWPHP}_1(f, g)$ by a randomized p-time algorithm: pick independently and at random polynomially many potential witnesses y and check whether one of them witnesses the formula. This will fail to happen with an exponentially small probability. Hence assuming that universal derandomization is possible we would also get a p-time witnessing function. This would seem to suggest, assuming universal derandomization, that the Conservativity problem 1.0.1 ought to have the affirmative solution: BT ought to be $\forall\Sigma^b_1$-conservative over S^1_2. However, such an argument would work only if the derandomization were provable in S^1_2. I find that unlikely. For example, we shall see in Theorem 4.3.2 that the existence of Boolean functions that has no circuits of size $\leq 2^{\epsilon n}$ is actually equivalent over S^1_2 to the dWPHP for p-time functions. In particular, the popular hypothesis used in universal derandomization that the computational class $\mathcal{E}$ (small exponential time $2^{O(n)}$) contains languages whose characteristic functions require so big circuits is unlikely to be provable in S^1_2 unless it equals to BT.

Formulas in dWPHP(PV) are Σ^b_2 so we need witnessing for proofs of such formulas in PV_1 or S^1_2. This time there is a difference between the two theories: axioms of S^1_2 are themselves Σ^b_2 but $\mathrm{PV}_1 \neq S^1_2(\mathrm{PV})$ unless $\mathcal{NP} \subseteq \mathcal{P}/poly$ by [75]. The class of functions where we shall find witnessing functions are those computable in a particular interactive manner.

Assume we are given a formula of the form:

$$\forall x \exists y(|y| \leq |x|^c)\forall z(|z| \leq |x|^d),\ A(x, y, z) \tag{2.2.1}$$

for some constants $c, d \geq 1$ (it is actually not necessary to assume these bounds but it simplifies the discussion). A witnessing function f should thus from input $x \in \{0, 1\}^n$ compute some y, $|y| \leq n^c$ such that

$$\forall z(|z| \leq n^d),\ A(x, y, z)\ . \tag{2.2.2}$$

The function will be computed interactively by two players: student S and teacher T. Student is a p-time algorithm while teacher has unlimited powers (i.e. it is an oraculum). Upon receiving input x S computes its first candidate solution y_1. If it satisfies (2.2.2) then T acknowledges that and the computation stops with the output y_1. If y_1 is incorrect T will provide to S a counter-example: a z_1, $|z_1| \leq n^d \wedge \neg A(x, y, z)$. Knowing z_1 S computes its new candidate solution y_2. In general we are interested in the number of rounds S needs in the worst case to solve the task for all $x \in \{0,1\}^n$. We will call this type of computation briefly **S-T computations**.

Note that formula (2.2.1) is $\forall\Sigma_3^b$ if $A \in \Sigma_1^b$ and that it is $\forall\Sigma_2^b(\mathrm{PV})$ if $A(x,y,z)$ is the dWPHP formula

$$y < 2x \wedge (z < x \to f(z) \neq y) \ .$$

Theorem 2.2.2

Let f be a PV function symbol and assume that dWPHP(f) is provable in (a) PV_1 or in (b) $S^1_2(PV)$.

Then there is a p-time student S that interacting with any T computes a function witnessing the formula in (a) $O(1)$ rounds or in (b) $n^{O(1)}$ rounds, respectively.

The theorem has a more delicate form that we shall need later; namely theory PV_1 *proves* that S solves the task. A student working in a constant number of rounds, say $k \geq 1$, can be represented by k p-time functions $S_1(x), S_2(x, z_1), \ldots, S_k(x, z_1, \ldots, z_{k-1})$ computing his moves in each round. The fact that he succeeds is equivalent to the validity of disjunction

$$\bigvee_{1 \leq i \leq k} (S_i(x, z_1, \ldots, z_{i-1}) < 2x \wedge f(z_i) \neq S_i(x, z_1, \ldots, z_{i-1})) \ . \tag{2.2.3}$$

A student working in n^k rounds will be represented by a p-time machine $S(x, z)$ that has a limited oracle access to string $z = (z_1, \ldots, z_t)$ of t strings $z_j < x$; we shall write this briefly as

$$z \in [x]^t$$

and we shall denote by $z|i$ the initial part of z consisting of first i strings z_j with $z|0$ being the empty string. The fact that S always succeeds in n^k steps is now equivalent to the validity of

$$z \in [x]^{|x|^k} \to \exists i < |x|^k,\ S(x, z|i) < 2x \wedge f(z_{i+1}) \neq S(x, z|i) \tag{2.2.4}$$

Theorem 2.2.2 can now be strengthen to

Theorem 2.2.3

Let f be a PV function symbol and assume that dWPHP(f) is provable in (a) PV_1 or in (b) $S^1_2(PV)$.

Then there is a p-time student S that interacting with any T computes a function witnessing the formula in (a) $O(1)$ rounds or in (b) $n^{O(1)}$ rounds such that (2.2.3) and (2.2.4) are provable in PV_1, respectively.

The opposite implications also hold.

We shall encounter S-T computations a number of times later. In particular, and Section 8.1 we give a variant of Theorem 2.2.3 and we shall discuss in Section 8.4 a relation between the assumption that dWPHP cannot be witness by S-T computation with polynomially many (or constantly many) rounds with another computational hypotheses.

2.3 Proof complexity: τ-formulas

Witnessing we discussed in Section 2.2 presupposes that the formula in question has an existential quantifier to witness. If a formula is open (no quantifiers at all), universal or, more generally, Π^b_1(PV) we deduce some information from the existence of its proof in a theory using the concept of *propositional translation.*

The translation assigns to a Π^b_1(PV)-formula $B(x)$ (with one free variable x for the simplicity of the notation) a sequence $\{\|B\|^n\}_n$ of propositional formulas. The n-th formula has atoms $p = (p_1, \dots, p_n)$ and some auxiliary atoms q, polynomially many in n of them, and it is constructed so that for all $b \in \{0,1\}^n$:

$$\mathbf{N} \models B(b) \ \Leftrightarrow\ \|B\|^n(b, q) \in \mathrm{TAUT}\ .$$

The translation is quite natural: it commutes with the logical connectives and replaces sharply bounded quantifiers by big disjunctions or conjunctions. Atomic formulas are translated using natural circuits computing the PV-functions involved in the formula. It is analogous to the standard proof of the $\mathcal{NP}$-completeness of SAT. We shall just summarize in the next two statements the key properties the translation has. The reader can find details in [65, 12.3], [45] or in original [17, 70].

Lemma 2.3.1

For a $\Pi^b_1(PV)$-formula $B(x)$ there is a p-time function (represented by a PV-function symbol) f such that

$$f \ : \ 1^{(n)} \ \rightarrow \ \|B\|^n$$

and PV_1 proves

$$\forall x(|x| = n), \quad B(x) \ \equiv \ (\|B\|^n(x, q) \in \mathit{TAUT}) \ .$$

The key fact is that a theory T is attached to a proof system P such that whenever $\forall x B(x)$ is T-provable then formulas $\|B\|^n$ have p-size P-proofs. We state this just for the theories used earlier in this chapter.

The following notation is handy:

- $P \vdash_* \alpha_n$: there are p-size P-proofs of formulas α_n,
- $\pi : P \vdash \beta$: π is a P-proof of β.

Theorem 2.3.2

Assume that $B(x) \in \Pi^b_1(PV)$ and that $S^1_2(PV)$ proves $\forall x B(x)$. Then $EF \vdash_* \|B\|^n$.

In fact, there is a p-time function (represented by a PV-function symbol) f such that PV_1 proves

$$f(1^{(n)}) \ : \ EF \ \vdash \ \|B\|^n \ .$$

Applying the translation and this theorem to formulas (2.2.3) and (2.2.4) yields the following statement key for next chapter.

Corollary 2.3.3

Let f be a PV function symbol and assume that (a) PV_1 or (b) $S^1_2(PV)$ proves $WPHP(f)$.

Then the $\| \dots \|^n$ translations of the formulas (a)

$$\forall z_1, \dots, z_k < x' \bigvee_{1 \leq i \leq k} (S_i(x, z_1, \dots, z_{i-1}) < 2x \wedge f(z_i) \neq S_i(x, z_1, \dots, z_{i-1})) \tag{2.3.1}$$

or (b)

$$\forall z(|z| < x^{|x|^k} \exists i < |x|^k, \ S(x, z|i) < 2x \wedge f(z_{i+1}) \neq S(x, z|i) \tag{2.3.2}$$

have p-size EF -proofs, respectively. Moreover, these proofs can be constructed provably in PV_1 by a p-time function.

2.4 Strong proof systems

Recall that a **Cook-Reckhow proof system** [19] is a p-time decidable binary (provability) relation $P(x, y)$ such that $\exists y P(x, y)$ defines TAUT, and that we write the relation $P(\alpha, \pi)$ as $\pi : P \vdash \alpha$; we call the string π a P-**proof** of α.

The efficiency of any proof system P is measured primarily by its **lengths-of-proofs function** $\mathbf{s}_P$. For a proof system P and a formula α put:

$$\mathbf{s}_P(\alpha) \ := \ \min\{|\pi| \mid \pi : P \vdash \alpha\}$$

if $\alpha \in \text{TAUT}$, and $\mathbf{s}_P(\alpha) \ := \ \infty$ otherwise. P is **p-bounded** iff

$$\forall \alpha \in \text{TAUT}\ \mathbf{s}_P(\alpha) \leq |\alpha|^{O(1)} \ .$$

Theorem 2.4.1 ([19])

A p-bounded proof system exists if and only if $\mathcal{NP} = co\mathcal{NP}$.

Our fundamental task is therefore to decide the existence of a p-bounded proof system. The following definition is handy when discussing tautologies hard to prove.

Definition 2.4.2 (hard sets of tautologies)

A subset $H \subseteq TAUT$ is **hard** *for a proof system P iff for any $c \geq 1$ the inequality $\mathbf{s}_P(\alpha) \leq (|\alpha| + c)^c$ holds for at most finitely many formulas in H.*

A hard set exists for P iff P is not p-bounded, meaning that $\mathbf{s}_P$ is not bounded by a polynomial. Thus if we believe that $\mathcal{NP} \neq co\mathcal{NP}$ the task becomes to show that all P admit a hard set (of tautologies). It may be that actually $\mathcal{NP} = co\mathcal{NP}$ but a good strategy to show that still may be to try to define candidate hard sets and see where the obstacle lies.

We aim primarily at strong proof systems which are, informally, those in the top two levels of the partitioning of proof systems into four levels in [65, Chpt.22]. To simplify writing technical hypotheses in many statements we adopt the following formal definition of strong proof systems.

Definition 2.4.3 (strong proof systems)

A proof system P is **strong**, *written $P \supseteq EF$, iff P is EF augmented by a p-time subset $A \subseteq TAUT$ as additional axioms: any substitution instance of any formula in A can be used in a proof. Such system will be denoted $EF + A$.*

The usefulness of this definition stems from the following properties systems $\mathrm{EF} + A$ have (this uses just classic proof complexity, cf. [70, 45, 65]).

Theorem 2.4.4

Strong proof systems P have the following properties:

1. *Any proof system Q can be p-simulated (provably in PV_1) by a strong proof system.*
2. *$P \vdash_* \|Con_P\|^n$ as well as $P \vdash_* \|Ref_P\|^n$, where Con_P and Ref_P are the consistency and the reflection principles for P.*
3. *There is $c \geq 1$ such that:*
    - *whenever $\sigma \in TAUT$ and σ' is obtained from σ by substituting for some atoms constants 0 or 1 then $s_P(\sigma') \leq s_P(\sigma)^c$, and*
    - *for all α, β: $s_P(\beta) \leq (s_P(\alpha) + s_P(\alpha \to \beta))^c$.*

Chapter 3

τ-formulas and generators

This chapter introduces the key definitions of τ-formulas, generators, and their hardness and pseudo-surjectivity, and states and proves several basic facts about them. We also present two conjectures and we discuss their implications for the original dWPHP problem 2.0.1. Further we outline a model-theoretic view of the conjectures. Finally we give some examples how are (and how are not) pseudo-random generators related to proof complexity generators.

3.1 τ-formulas and generators

A Boolean circuit C of size s with n inputs $x = x_1, \dots, x_n$ and m outputs $z = z_1, \dots, z_m$ is a series of s intermediate values $y = y_1, \dots, y_s$ defined by instructions how to compute each y_i using De Morgan basis functions from inputs x, constants $0, 1$ or from earlier y_js (we shall sometimes refer to y_i themselves as instructions). The m-tuple z is just the m-tuple of the last m intermediate values y_is. Hence computation can written also as $y_1, \dots, y_{s-m}, z_1, \dots, z_m$. Each instruction can be written as 3-CNF, so all s instructions of C can be collected in one 3-CNF we shall denote $\mathrm{Def}_C(x, y, z)$ or $\mathrm{Def}_C^{n,m,s}(x, y, z)$ when we want to stress the parameters. Note that the formula has at most $3s$ 3-clauses.

Assume $1 \leq n < m$ and let $g_n : \{0,1\}^n \to \{0,1\}^m$ be a function computed by a size s circuit C_n with n inputs x_u, m outputs z_v, and instructions y_i, as above. The complement of the range of g_n, $\{0,1\}^m \setminus rng(g_n)$, contains at least half of elements of $\{0,1\}^m$ and, in particular, it is non-empty.

Definition 3.1.1 (τ-formulas)

Given any string $b \in \{0,1\}^m$ define the propositional **τ-formula** $\tau(C_n)_b$ *to be the 3DNF:*

$$\neg Def_{C_n}(x, y, z) \ \vee \ \bigvee_{i \in [m]} b_i \neq z_i \ .$$

The size of the formula is $O(s)$ and for all $b \in \{0,1\}^m$:

$$\tau(C_n)_b \in \text{TAUT} \ \text{ iff } \ b \notin rng(g_n) \ .$$

When we want to stress the propositional atoms in the formula we may sometimes use p for (bits of) x and q for (bits of) y.

We want to study the complexity of τ-formulas determined by one function $g : \{0,1\}^* \to \{0,1\}^*$ for unbounded input size n. We shall consider functions g defined by a sequence of circuits $\{C_n\}_n$ that compute finite functions $g_n := g \restriction \{0,1\}^n$, the restrictions of g to $\{0,1\}^n$. The following definition is handy to avoid long technical hypotheses of various statements.

Definition 3.1.2 (generators)

A function $g = \{C_n\}_n$ is **generator** *iff it satisfies the following two conditions:*

1. *g is* **stretching***:* There is a function $n \to m := m(n) > n$ such that for any $n \geq 1$, C_n has $m(n)$ outputs.

   The function $m(n)$ is called **the stretch***.*

2. The size of C_n is $m^{O(1)}$.

Sometimes it is useful to assume that the stretch is an injective function; that implies that a string b can be in $rng(g_n)$ for at most one n. We shall call such functions g **uniquely stretching**. The second condition implies that the size of $\tau(C_n)_b$ is $m^{O(1)}$ which is also $|b|^{O(1)}$.

Calling functions from the definition *generators* is in order to keep up with the somewhat unfortunate but established terminology calling the functions *proof complexity generators*. The term *generator* was used at the start: [5] specifically targeted pseudo-random generators and their role in proof complexity, and to me it looked like that the dWPHP problem 2.0.1 will have a lot to do with cryptographic primitives (one-way functions and pseudo-random generators) and their formalization in bounded arithmetic. The connection to pseudo-randomness turned out to be eventually less direct and more subtle and we shall discuss it in Section 3.6.

To find our peace with the term *generator* we may interpret it as meaning that any such g generates a class of τ-formulas $\tau(C_n)_b$, $n \geq 1$ and $b \in \{0,1\}^m \setminus rng(g_n)$. We shall often use simpler notation $\tau(g)_b$ for the τ-formulas when circuits C_n are clear from the context.

If a generator g is computed by a specific deterministic algorithm (i.e. a Turing machine) running in time polynomial in $m(n)$ we assume that the algorithm determines canonically circuits C_n. One may use, for example, the construction underlying the usual proof of the $\mathcal{NP}$-completeness of SAT. We may stress this by saying that g is a **uniform generator** (and we refer sometimes to general generators as **non-uniform**). Talking about a generator as of a function in this case is a mild abuse of language as the τ-formulas are determined by the underlying algorithm and not by the function. However, when defining various uniform g there is always a canonical algorithm computing g and there is no danger of a confusion. Moreover, the candidate uniform generators ought to be hard for all algorithms computing them.

For a generator g we shall denote by $\tau\mathrm{Fla}(g)$ the set of all τ-formulas determined by g:

$$\tau\mathrm{Fla}(g) \ := \ \{\tau(g)_b \mid b \in \bigcup_{n\geq 1} \{0,1\}^{m(n)} \setminus rng(g_n)\} \ . \tag{3.1.1}$$

It will be clear after defining the hardness in the next section why we leave out b whose length is not $m(n)$ for some n. In fact, strictly speaking it is not necessary as we have not defined the τ-formulas for b which do not have the length $m(n)$ for some $n \geq 1$. However, if we look at τ-formulas as being the translations of the formula (3.1.2) we could substitute into it also strings b that do not have the appropriate length and that could lead to a confusion: for example, if $|g(x)| = 2|x|$ it is easy to prove that strings of odd size are not in the range of g.

Note that we have a symbol in the language of PV for any uniform and p-time generator g and that if g is uniquely stretching then the τ-formula $\tau(g)_b$ is simply the propositional translation of Section 2.3 of the arithmetic formula expressing that $y \notin rng(g)$:

$$\forall x(|x| \leq |y|)\ g(x) \neq y \tag{3.1.2}$$

with b substituted for (bits of) y.

Let us remark that formula (3.1.2) remains Π^b_1 even if g is only $\mathcal{NP} \cap co\mathcal{NP}$ and hence the τ-formulas could be defined for these functions (even for non-uniform variants) as well. We shall discuss this in Chapter 5.3.

3.2 Hardness and the working conjecture

The following elegant definition was given (somewhat informally) in [5]. I originally used in [49, 50] instead a model-theoretic condition described here in Section 3.5.

Definition 3.2.1 (hard generators, [5])
A generator g is **hard** *for a proof system P if and only if the set $\tau Fla(g)$ is hard for P. That is, for all $c \geq 1$, for all but finitely many $\tau(g)_b \in \tau Fla(g)$*

$$\mathbf{s}_P(\tau(g)_b) \ > (|\tau(g)_b| + c)^c \ . \tag{3.2.1}$$

If the inequality (3.2.1) holds even with an exponential term $2^{|\tau(g)_{\overline{b}}|^{\Omega(1)}}$ we shall call g **exponentially hard** for P.

Now we can state our first working conjecture. The qualification *working* is meant to stress that while we think it is true we do not consider it carved in stone and we take it primarily as a sign-post for further research.

Conjecture 3.2.2 (Working conjecture, [51])
There exists a uniform p-time generator g with the stretch $n + 1$ that is hard for all proof systems P.

The requirement on the stretch is not essential (we can always truncate a hard p-time generator to stretch $n + 1$ and keep the hardness) but it allows us to reformulate the conjecture in the following simple but elegant way.

Lemma 3.2.3 ([51, 68])
A p-time g with the stretch $n+1$ satisfies the working conjecture 3.2.2 iff $rng(g)$ intersects all infinite $\mathcal{NP}$ sets (i.e. $rng(g)$ is $\mathcal{NP}$-immune).

Proof:

Assume w.l.o.g. that P is a strong proof system (Def 2.4.3) and that condition (3.2.1) fails for some fixed $c \geq 1$ and infinitely many $b \notin rng(g)$. Define set

$$\{b \in \{0,1\}^* \mid \mathbf{s}_P(|\tau(g)_b|) \leq (|\tau(g)_b| + c)^c\} \ .$$

It is in $\mathcal{NP}$, infinite and is disjoint with $rng(g)$.

For the opposite direction assume that an infinite $\mathcal{NP}$ set A is defined by the condition

$$x \in A \ \Leftrightarrow \ \exists y(|y| \leq |x|^d) R_A(x,y)$$

where R_A a p-time relation, and it is disjoint with $rng(g)$. Then g is not hard for the strong proof system extending EF by accepting also as a proof of the τ-formula $\tau(g)_b$ any string π such that

$$|\pi| \leq |b|^d \ \wedge \ R_A(b,\pi) \ .$$

q.e.d.

The lemma can be modified to characterize uniform generators hard for a given proof system using the following notion (quite close to resultants in model theory, hence the name).

Definition 3.2.4 (resultant, [51])

For a proof system P and uniform generator g define the **resultant** *to be the set Res_g^P of all $\mathcal{NP}$ sets which can be defined by a Σ_1^b-formula $A(x)$ such that P proves by p-size proofs that $\{y \mid A(y)\}$ is disjoint from $rng(g)$:*

$$P \ \vdash_* \ \|g(x) = y \rightarrow \neg A(y)\|^n \ . \tag{3.2.2}$$

Lemma 3.2.5 ([51])

Assume P is a strong proof system and g is a p-time generator. Then g is hard for P iff Res_g^P contains no infinite set.

Proof:

Assume a p-time g is not hard for P, i.e. for some $c \geq 1$ the inequality $\mathbf{s}_P(\tau(g)_b) \leq |b|^c$ holds for infinitely many b (using that $|\tau(g)_b| \leq |b|^{O(1)}$). Define $\mathcal{NP}$ set by the formula

$$A(y) \ := \ [\exists x \leq y |g(x)| = |y|] \wedge [\exists \pi(|\pi| \leq |y|^c) \ \pi : P \vdash \tau(g)_y] \ .$$

As we assume that P is strong, it proves (by Theorem 2.4.4) by p-size proofs its own soundness, and hence the condition (3.2.2) holds. The resultant thus contains an infinite set.

The opposite direction is proved analogously as in Lemma 3.2.3.

q.e.d.

The uniform version of resultant in Def.3.2.4 is from [65, Sec.19.4]. Originally [51] considered a version for non-uniform generators $g = \{C_n\}_n$ and the resultant in that case refers to $\mathcal{NP}/poly$ sets. If that resultant contains no infinite set then g is hard for P but to get an equivalence one needs to restrict advices the sets in the resultant may use to circuits C_n.

Let us conclude this section by recording an obvious observation.

Lemma 3.2.6

For any strong proof system P: there is a generator (exponentially) hard for P iff the circuit value function CV is (exponentially) hard for P.

3.3 The pseudo-surjectivity conjecture

The idea underlying hard generators g is that these ought to be functions that violate - relative to a proof system - the dWPHP. That is, one can think consistently - in the theory associated to the proof system - that some g_n is onto. Consider, however, the situation when g is hard but you can shortly prove infinitely many disjunctions

$$\tau(C_n)_{b^1} \vee \tau(C_n)_{b^2}$$

for $n \geq 1$ and $|b^i| = m(n)$.

To give another example, and a general definition of similar disjunctions later, we need to make in τ-formulas explicit some atoms. Recall that for a generator $g = \{C_n\}_n$, when writing $\tau(C_n)_b(p)$ we mean that p is an n-tuple of atoms corresponding to x in (the translation of) the statement $g(x) \neq b$; there are other atoms q corresponding to the intermediate values of C_n in Def_{C_n}. Hence the disjunction above can be written as

$$\tau(C_n)_{b^1}(p^1) \vee \tau(C_n)_{b^2}(p^2) \ .$$

In the second example assume you can shortly prove a bit more involved disjunctions of the form

$$\tau(C_n)_{b^1}(p^1) \vee \tau(C_n)_{B^2}(p^2)$$

where $B^2(p^1)$ is a circuit computing m-string from an n-string p^1. This latter disjunction may appear as a translation of a natural first-order statement

$$g(x^1) \neq b^1 \vee g(x^2) \neq f(x^1)$$

where f is a p-time function, and the formula $\tau(C_n)_{B^2}$ involves defining B^2 using Def_{B^2}.

Note that in both these examples we cannot consistently think that C_n is surjective: in the first case one of b^1, b^2 cannot be in the range and in the second case either b^1 is not in the range or, if $C_n(a^1) = b^1$, then string $b^2 := B^2(a^1)$ is not in the range.

The general form of disjunctions for generator $g = \{C_n\}_n$ we need to consider is this:

$$\tau(g)_{B^1}(p^1) \vee \tau(g)_{B^2}(p^1, p^2) \vee \cdots \vee \tau(g)_{B^t}(p^1, \ldots, p^t) \qquad (3.3.1)$$

where B^i are circuits with inputs $p^1, \ldots, p^{i-1}$. The following definition is crucial.

Definition 3.3.1 (pseudo-surjectivity, [51])

A generator $g = \{C_n\}_n$ is **pseudo-surjective** *for a proof system P iff for any $c \geq 1$, for at most finitely many $n \geq 1$ and disjunctions (3.3.1) with B^i having $m(n)$ outputs have P-proof of size less than $m(n)^c$.*

Similarly as with the hardness, if there are no P-proofs of size less than $\exp(m^{\Omega(1)})$ we say that g is **exponentially pseudo-surjective** for P.

Note that the pseudo-surjectivity obviously implies the hardness. Analogously to Lemma 3.2.6 we have

Lemma 3.3.2

For any strong proof system P: there is a generator (exponentially) pseudo-surjective for P iff the circuit value function CV is (exponentially) pseudo-surjective for P.

We shall see in Section 4.3 another example of a function that has this universal property.

Now we can state our second conjecture.

Conjecture 3.3.3 (Pseudo-surjectivity conjecture, [51])

There exists a p-time generator with the stretch $n+1$ that is pseudo-surjective for EF.

Results in Sections 4.3 and 4.4 will imply that it is not reasonable to expect that a pseudo-surjective generator exists for all proof systems, unless you are prepared to believe that $\mathcal{NE} \cap co\mathcal{NE} \subseteq \mathcal{P}/poly$, cf. [51].

The next theorem will show that there exists a function pseudo-surjective for EF unless EF simulates a proof system that appears to be stronger. The proof system in question is WF (for **weak PHP Frege**), an extension of the proof system CF (standing for **circuit Frege**, a reformulation of EF). Both were defined in [34, 35] in a way equivalent to the following one, cf.[65, Sec.7.2].

Starting with a Frege system F in the DeMorgan language we define a **CF-proof** of a target circuit B from initial circuits A_j to be a sequence of circuits $\pi = C_1, \dots, C_k$ such that:

- Each C_i:
  - is either one of initial circuits A_j,
  - or it is derived from some some earlier circuits $C_{j_1}, \dots C_{j_\ell}$, $j_1, \dots, j_\ell < i$ by an inference rule of F:

$$\frac{D_1, \dots, D_\ell}{D_0} \tag{3.3.2}$$

    That is, there is a substitution σ of circuits for atoms in the formulas D_u such that $\sigma(D_u) = C_u$ for $u = 0, \dots \ell$,
  - or there is $j < i$ such that C_i is similar to C_j,
- $C_k = A$.

The **similarity of circuits** E, E' means that when we unwind the them (in some unique way) to formulas then these two formula are identical. Note that similarity of circuits is $\mathcal{P}$ (cf. [65, L.7.2.1]).

Having CF we define a **WF-proof** of B from $A_1, \dots, A_t$ to be a CF-proof that can also use the following rule:

- *For any $1 \leq n < m$ and any collection $\mathcal{C}$ of m circuits $C_i(x)$, all with n inputs x, introduce a new m-tuple of atoms $r = (r_1, \dots, r_m)$ that is attached to the collection $\mathcal{C}$ such that no r_i occurs in any of $B, A_1, \dots, A_t, C_1, \dots, C_m$, and for any circuits $D_1, \dots, D_n$ (which may contain r) we may use the axiom:*

$$\bigvee_{i \leq m} C_i(D_1, \dots, D_n) \neq r_i \ .$$

Theorem 3.3.4 ([51, Thm.5.2])

Assume that EF does not simulate the proof system WF. Then EF admits a p-time pseudo-surjective generator.

The proof can be found in [51] or after [65, L.19.5.4]. The generator is the truth-table function $\mathbf{tt}_{s,k}$ with $s = 2^{\delta k}$ which we shall introduce in Definition 4.3.1.

3.4 Consequences for the dWPHP problem

Using suitable witnessing theorems and propositional translations (Sections 2.2,2.3) we derive an implication for the dWPHP problem 2.0.1.

Theorem 3.4.1

Assume that there is a p-time generator g that is pseudo-surjective for EF. Then $S^1_2(PV)$ does not prove dWPHP(g), i.e. $BT \neq S^1_2(PV)$.

Proof:

We shall use Corollary 2.3.3. Assume that g is a p-time generator pseudo-surjective for EF. By truncating its output we may assume w.l.o.g. that its stretch is $n + 1$.

Assume for the sake of contradiction that dWPHP(g) is provable in S^1_2(PV). By Corollary 2.3.3 (part (b)) the propositional translation of formula (2.3.2) has p-size EF-proofs. This translation has the form of the disjunction (3.3.1):

$$\tau(g)_{B^1}(p^1) \vee \tau(g)_{B^2}(p^1, p^2) \vee \dots \vee \tau(g)_{B^t}(p^1, \dots, p^t)$$

where $1 \leq t \leq n^{O(1)}$ and circuits B^i compute student's i-th move. As the student is p-time the sizes of B^is are polynomial in $m(= n+1)$. This contradicts the assumed pseudo-surjectivity of g for EF.

q.e.d.

The argument can be modified for theory PV_1 in place of $S^1_2(\mathrm{PV})$ (i.e. the dWPHP problem becomes $\mathrm{PV}_1 =_? \mathrm{APC}_1$) using the following notion from [50, Def.6.1] (it actually preceded the pseudo-surjectivity).

Definition 3.4.2 (k-freeness, [50])

Let $k \geq 1$ be fixed. A generator $g = \{C_n\}_n$ is k-**free** *for proof system P iff for any $c \geq 1$, for at most finitely many $n \geq 1$ and disjunctions (3.3.1) with $t = k$ and with B^i having $m(n)$ outputs have P-proof of size less than $m(n)^c$.*

A generator is **free** *iff it is k-free for all $k \geq 1$.*

The next statement is derived analogously to Theorem 3.4.1 using part (a) of Corollary 2.3.3 instead of part (b).

Theorem 3.4.3

Assume that there is a p-time generator g that is free for EF. Then PV_1 does not prove $dWPHP(g)$, i.e. $PV_1 \neq APC_1$.

3.5 A model-theoretic characterization

There is a well-known tight relation between the existence of short proofs and extensions of models of bounded arithmetic. We shall formulate it only in the version suitable for our purposes; the phenomenon is much more general (cf. [45, 65]). Section 7.2 will be concerned with the closely related issue of expansions of pseudo-finite structures. General background can be found in [65, Chpt.20].

Let $T \supseteq \mathrm{PV}_1$ be a theory in the language of PV_1 and let P be a strong proof system. We say that T **and** P **correspond to each other** iff the following two conditions are met:

1. $T \vdash Con_P$,
2. P **simulates** T: if $B(x)$ is a Π^b_1-formula and $T \vdash B$ then $P \vdash_* \|B\|^n$.

Note that by [17] and Theorem 2.3.2 both theories PV_1 and $S^1_2(\mathrm{PV})$ correspond to EF. This is because only the Π^b_1-consequences of T play a role in the definition.

Theorem 3.5.1 ([72])

Let $T \supseteq PV_1$ be a theory in the language of PV_1 and P be a strong proof system that correspond to each other. Let $\mathbf{M}$ be a model of T and assume $\tau \in \mathbf{M}$ is a tautology in the model.

Then the following two statements are equivalent:

- $\mathbf{M}$ *has an extension to* $\mathbf{M}'$ *such that* $\mathbf{M}' \models T + \neg\tau \in SAT$.
- $\mathbf{M} \models P \not\vdash \tau$.

A simple (though rarely useful) way how to construct non-standard models of PV_1 and $S^1_2(\mathrm{PV})$ is to take a nonstandard model $\mathbf{M}$ of true arithmetic in the language of $S^1_2(\mathrm{PV})$, its non-standard element $n \in \mathbf{M} \setminus \mathbf{N}$ and define the **small canonical model** to be the substructure $\mathbf{M}_n$ of $\mathbf{M}$ with the universe

$$\{u \in \mathbf{M} \mid |u| \leq n^k \text{ , some } k \in \mathbf{N}\} \ .$$

It is a cut in $\mathbf{M}$. **Large canonical models** $\mathbf{M}^*_n$ are defined analogously, just the universes are larger:

$$\{u \in \mathbf{M} \mid |u| \leq 2^{n^{1/k}} \text{ , all } k \in \mathbf{N}\} \ .$$

Theorem 3.5.2

Let P be a strong proof system, $T \supseteq PV_1$, and assume they correspond to each other. Let g be a p-time generator.

Assume further that any small canonical model $\mathbf{M}_n$ has for any $b \in \{0,1\}^m$ an extension $\mathbf{M}' \supseteq \mathbf{M}_n$ such that

$$\mathbf{M}' \models T + b \in rng(g_n)$$

where $m = m(n)$.

Then the generator g is hard for P.

Proof:

If g is not hard for P it means that for some $c \geq 1$ and infinitely many $n' \in \mathbf{N}$ there are formulas $\tau(g)_b \in \tau\mathrm{Fla}(g)$, $|b| = m(n')$ that have P-proofs π_b of size $\leq (n')^c$ (here we use that g is p-time so $m(n')$ is polynomial in n').

Hence in $\mathbf{M}$ there is a non-standard n for which there is a formula $\tau(g)_b \in \tau\mathrm{Fla}(g)$, $|b| = m(n)$ that has a P-proof π_b of size $\leq n^c$. Therefore also $\pi_b \in \mathbf{M}_n$ and hence, as $\mathbf{M}'$ is a model of T and $T \vdash Con_P$, $b \notin rng(g_n)$ in any extension $\mathbf{M}'$ of $\mathbf{M}_n$.

q.e.d.

We remark that if we assume in addition that T is a universal theory in the language of PV_1 then also the opposite statement holds.

The existence of an expansion where the dWPHP fails can be equivalently characterized using the notions of pseudo-surjectivity and freeness from Section 3.3. We shall outline the proof; the reader can find details in [50].

Theorem 3.5.3

Let P be a strong proof system, $T \supseteq PV_1$ be a true universal theory in the language of PV_1, and assume P and T correspond to each other. Let g be a p-time generator.

Then the following two statements are equivalent:

- *Generator g is free for P.*
- *Every small canonical model $\mathbf{M}_n$ has an extension $\mathbf{M}' \supseteq \mathbf{M}_n$ such that*

$$\mathbf{M}' \models T + rng(g_n) = \{0,1\}^m$$

where $m = m(n)$.

The same is true for pseudo-surjectivity when $\mathbf{M}'$ is required to be a model of $T + S^1_2(PV)$.

Proof:

We shall treat the case of pseudo-surjectivity as it is going to be used later. Assume first that g is not pseudo-surjective for P. As in the previous proof there is a non-standard $n \in \mathbf{M}$ such that $\mathbf{M}_n$ contains a P-proof of a disjunction having the form as in the definition of pseudo-surjectivity. This proof will be also in any $\mathbf{M}'$ and hence the disjunction will be a tautology in $\mathbf{M}'$ too. Hence g_n cannot violate the dWPHP.

Now assume that g is pseudo-surjective for P and hence for no non-standard n does $\mathbf{M}_n$ contain a P-proof of a pseudo-surjectivity disjunction. Assume for the sake of a contradiction that no extension with the required properties exists. This mean that theory

$$T \ + \ S^1_2(\mathrm{PV}) \ + \ Diag(\mathbf{M}_n)$$

where $Diag(\mathbf{M}_n)$ is the atomic diagram of the small canonical model proves that g_n is not onto.

By a variant of the witnessing Theorem 2.2.3 for T this means that $T + Diag(\mathbf{M}_n)$ proves a disjunction as in (2.2.4) expressing that a p-time student solves the witnessing task in n^k rounds. By the correspondence between T and P, the propositional translation $\| \dots \|^n$ of the disjuction has a p-size P-proof in $\mathbf{M}$ from (translations of) sentences in $Diag(\mathbf{M}_n)$. But all (translations of) sentences on the diagram are just true Boolean sentences that are proved in P by their evaluations. This gives a p-size P-proof $\pi_n \in \mathbf{M}$ of some disjunction as in the pseudo-surjectivity. That is a contradiction.

q.e.d.

Statements analogous to these two theorems about exponential hardness (or exponential freeness or exponential pseudo-surjectivity) hold when one uses large canonical models instead small ones.

The key message from this section is that Theorem 3.5.2 suggests a way to prove the hardness of g (for particular P): find a construction of extensions of small canonical models satisfying suitable theory T corresponding to P. We shall discuss a related approach in Sections 7.2 and 7.3.

3.6 A relation to pseudo-randomness

The authors of [5] insisted on the role of pseudo-random number generators (PRNGs, in short), stressing it already in the title of their paper. I just thought originally (as articulated in [49, 50]) that a random behavior of generators will be important (and sufficient). Things developed in a bit more subtle way.

The Nisan-Wigderson generator treated at length in [5, 97] is still a good candidate generator - and we shall discuss it in Chapter 5 and Section 5.3 - but no other commonly studied PRNG was ever proposed as a candidate proof complexity generator. In fact, we shall see below that the construction of PRNGs from one-way permutations via hard bits does not lead to generators hard for all proof systems (often not even for EF).

I also moved away from my initial view that random behavior may be crucial and I think now that the impossibility to witness errors by restricted computational means is more crucial. This is meant in the formalism of [60] and we shall discuss it in Chapter 7 (Sections 7.4 and 7.5).

Nevertheless, in this section we present a few examples and statements illustrating the role of PRNGs. Let us recall first the notion of pseudo-random number generators; we shall deviate slightly from the standard terminology in order to avoid a clash with our notions of hardness.

The **PRNG-hardness** $H(g)$ of stretching function $g = \{g_n\}_n$, $g_n : \{0,1\}^n \rightarrow \{0,1\}^{m(n)}$, is the function assigning to $n \geq 1$ the minimum S such that there is a circuit $C(y)$ with $m(n)$ inputs and of size $\leq S$ such that

$$|\mathrm{P}rob_{x\in\{0,1\}^n}[C(g(x))=1] \ - \ \mathrm{P}rob_{y\in\{0,1\}^m}[C(y)=1]| \ \geq \ \frac{1}{S} \ .$$

A **pseudo-random number generator** is a p-time stretching function g that has super-polynomial hardness: $H(g) \geq n^{\omega(1)}$.

The reader ought to recall the concept of feasible interpolation, cf. [65, Chpt.17-18].

Theorem 3.6.1 ([5])

Assume that g is a PRNG with stretch $m(n) \geq 2n+1$ and define $g^* : \{0,1\}^{2n} \rightarrow \{0,1\}^m$ for $u, v \in \{0,1\}^n$ by:

$$g^*(u,v) \ := \ g(u) \oplus g(v)$$

where $\oplus$ denotes the bit-wise sum modulo 2.

Then g^* is a (proof complexity) generator hard for all proof systems simulating resolution R and admitting feasible interpolation.

Proof:

Assume P is a proof system that admits feasible interpolation and that formula $\tau(g^*)_b$ has a size s P-proof. Then (the $\| \dots \|$ translation of)

$$g(u) \neq y \ \vee \ g(v) \neq y \oplus b$$

has a size $s + m^{O(1)} = s^{O(1)}$ P-proof.

The feasible interpolation property then yields a size $s^{O(1)}$ circuit I with m inputs y defining a set (also denoted I) separating $rng(g)$ from $b \oplus rng(g)$:

$$rng(g) \subseteq I \ \text{ and } \ I \cap b \oplus rng(g) = \emptyset \ .$$

If I contains at most a half of $\{0,1\}^m$ then $\neg I$ defines a subset of measure $\geq \frac{1}{2}$ in the complement of $rng(g)$ and hence $H(g) \leq |I| \leq s^{O(1)}$. Otherwise $I \oplus \overline{b}$ defines such a subset. Hence $s^{O(1)} \geq H(g)$.

Therefore, if s were $n^{O(1)}$, g is not PRNG.

q.e.d.

For the next statement let $h : \{0,1\}^* \to \{0,1\}^*$ be a permutation (a bijection preserving the length) and assume it is a **one-way permutation** (OWP, shortly), and further assume that B is a **hard bit** predicate for h. Then by [106] the generator

$$\overline{x} \to (h(\overline{x}), B(\overline{x}))$$

is a PRNG.

Theorem 3.6.2 ([50])

Assume h is a OWP and B is its hard bit predicate, and let g be the PRNG as defined above. Assume further that P is a strong proof system such that

$$P \vdash_* \|h(u) = h(v) \to u = v\|^n \ . \tag{3.6.1}$$

Then g is not a hard proof complexity generator for P.

In particular, if g is constructed in this way from the RSA and B is the parity of the pre-image then g is not hard for EF.

Proof:

Take any $b \in \{0,1\}^{n+1} \setminus rng(g_n)$. As h is a permutaion we have $rng(h_n) = \{0,1\}^n$ and so for some $a \in \{0,1\}^n$

$$h(a) = b \ \text{ and } \ B(a) \neq b_{n+1} \ . \tag{3.6.2}$$

A P-proof of $\tau(g_b)$ can be thus given as follows: take a and verify (3.6.2), and subsequently derive $b \notin rng(g_n)$ by using the injectivity (3.6.1) of h, the translation of

$$h(x) = b \to x = a \ .$$

The statement about EF follows as EF has p-size proofs of the injectivity of the RSA is by [73] provable in $S^1_2(\text{PV})$ (and use propositional translation).

q.e.d.

The theorem implies that PRNGs are not a priori hard proof complexity generators but that a PRNG may be a hard proof complexity generator because of its specific construction (the prominent example is the Nisan-Wigderson generator - Chapter 5).

We shall mention one more example from the worlds of pseudo-randomness. Rudich [99] attempted to generalize the concept of natural proofs of [98] to non-deterministic circuit complexity. One notion he considered goes under the name *demi-bit*. Given a generator g consider *non-deterministic circuits* C_n with $m = m(n)$ inputs satisfying

$$C_n^{(-1)}(1) \cap rng(g_n) = \emptyset.$$

The demi-bit hardness of g is the minimal $s = s(n)$ such that there are such C_n of size $\leq s$ satisfying also the following largeness condition:

$$|C_n^{(-1)}(1)| \geq 2^m/s \ .$$

A generator based on the subset sum following [32] is proposed in [99] as a candidate for having large hardness in the above sense but no (even informal) evidence for that is offered. Cf. also [60, Sec.30.4].

Chapter 4

The stretch

A view of generators we explore in this chapter is that they can be thought of as decompression algorithms. Hence their range contains only strings w that allow in a sense for shorter than size $|w|$ description. Two prominent ways how to formalize compressibility are Kolmogorov's complexity and circuit complexity. I think that both of them are too universal concepts to allow to prove the hardness of some specific generators but nevertheless we ought to be aware of these connections. In fact, it may turn out that results about proof complexity generators will imply statements about Kolmogorov or circuit complexity.

4.1 Stretch and Kolmogorov complexity

Every string $e \in \{0,1\}^*$ is also interpreted as a code of a unique Turing machine. We take a time-restricted universal Turing machine U with three inputs: machine code e, input to that machine u and string $1^{(t)}$ of t ones bounding the time. Machine U will simulate machine e on input u for at most t steps. It will stop, and output the same string, if e stops in $\leq t$ steps. Otherwise U just outputs 0. The simulation runs in p-time (in the length of all three inputs).

Fixing U, **the time-bounded Kolmogorov complexity** of a string $w \in \{0,1\}^*$ is (cf.[79]):

$$Kt(w) \ := \ \min\{|e| + \lceil \log t \rceil \mid U(e, 0, 1^{(t)}) = w\} \ .$$

For a fixed function $t(x)$ bounding the time there is also this measure:

$$K^t(w) \ := \ \min\{|e| \mid U(e, 0, 1^{(t(|w|))}) = w\} \ .$$

Measure Kt looks more elegant as you do not have to fix the time bound in advance. By the same token, measure K^t considers only codes e and does not mix it with time.

Assume now that a uniform p-time generator g has the stretch $m := m(n)$. This means that any $w \in rng(g)$ satisfies

$$K^t(w) \leq n + O(1) \ \text{ and } \ Kt(w) \leq n + O(1) + O(\log n)$$

where the $O(1)$ term accounts for the code of the algorithm defining g and the $O(\log n)$ term accouns for the (logarithm of) time.

Hence if the stretch is at least

$$m \geq n + \omega(\log n) \tag{4.1.1}$$

we have:

$$w \in rng(g) \to K^t(w) \leq Kt(w) < |w| \ .$$

This means that if the working conjecture 3.2.2 is true for a p-time generator of stretch at least (4.1.1) the following open problem must have an affirmative answer.

Problem 4.1.1 (Kt problem [68, Problem 5.2])

Does every infinite $\mathcal{NP}$ set A contain a string $w \in A$ with $Kt(w) < |w|$?

Putting it differently: Is it true that the set $\{w \mid Kt(w) \geq |w|\}$ is $\mathcal{NP}$-immune?

Ruling out generators for the working conjecture 3.2.2 by answering the problem in the negative seems to be difficult because of the next theorem. Given a binary relation $R(x, y)$ satisfying

$$R(x, y) \to |y| \leq 2^{c|x|}$$

for some $c \geq 1$ such that R is decidable in time $2^{O(n)}$ for $n = |x|$, consider the following search task: given x, find y such that $R(x, y)$, if it exists. This is termed $\mathcal{NE}$ **search problem** in [6]. We shall use the following notation from that paper: for any $A \subseteq \{0, 1\}^*$, $Kt_A : \mathbf{N}^+ \to \mathbf{N}^+$ is the function defined by

$$Kt_A(m) \ := \ \min\{Kt(w) \mid w \in \{0, 1\}^m \cap A\}$$

(we leave $Kt_A(m)$ undefined otherwise).

Theorem 4.1.2 ([6, Cor.7,Thm.8])

There exists an infinite $\mathcal{NP}$ set A s.t. $Kt_A(w) = \omega(\log|w|)$ for infinitely many $w \in A$ iff there exists an $\mathcal{NE}$ search problem s.t.:

- *$\exists y R(x, y)$ is satisfied for infinitely many x,*
- *every algorithm running in time $2^{O(n)}$ solves the search problem for a finite number of inputs x only.*

Not only is the affirmative answer to the problem implied by the existence of suitable generators but it itself implies the existence of an interesting function too.

Theorem 4.1.3 ([68, Thm.5.3])

If Problem 4.1.1 has the affirmative answer then $\mathcal{NP}$ is a proper subclass of $\mathcal{EXP}$.

Proof:

There is a function g computable in time $2^{O(n)}$ such that

$$rng(g_n) \ = \ \{w \in \{0,1\}^{n+1} \mid Kt(w) \leq n\} \ .$$

The complement $\{0,1\}^* \setminus rng(g)$ is infinite and is in $\mathcal{E}$ but it cannot be - assuming the affirmative answer to the problem - in $\mathcal{NP}$. Hence that neither $\mathcal{E}$ nor $\mathcal{EXP}$ are subclasses of $\mathcal{NP}$. As $\mathcal{NP} \subseteq \mathcal{EXP}$ we have $\mathcal{NP} \subset \mathcal{EXP}$.

q.e.d.

Let us conclude this section by noticing that while we cannot presumably express a lower bound to $Kt(w)$, say $Kt(w) \geq |w|/2$, by a p-size tautology, for a fixed p-time $t(n)$ we can take complexity K^t and consider the universal Turing machine U restricted to time $t(|u|)$; call it U^t. Machine U^t runs in p-time if t is a polynomial (though not in time t itself), takes just inputs e, u, and simulates machine with code e on u for time $t(|u|)$. We consider U^t as mapping $n' = n'(n)$-bit strings where $n'(n) := n + \omega(1)$ (e.g. $n + \log n$, for example) to size $m = m(n)$ strings. The term $\omega(1)$ accounts for the description of a machine) and we assume w.l.o.g. that all outputs have size $m = m(n)$ exactly. Hence U^t is a p-time generator and it satisfies

$$rng(g) \ \subseteq \ rng(U^t) \tag{4.1.2}$$

whenever g is a uniform generator computed in time $t(n)$ with the stretch $m(n)$.

Theorem 4.1.4

Let $t(n)$ be a polynomial time bound and let P be a strong proof system. If there is any uniform generator g computable in time $t(n)$ and with the stretch $m(n) > n'(n)$ which is hard for P, so is U^t.

Proof:

The construction of U^t can be readily formalized in theory PV_1 and thus the propositional translations of (4.1.2) have p-size EF proofs.

Hence if some τ-formulas resulting from U^t have short P-proofs so do some $\tau(g)$-formulas.

q.e.d.

Tautologies similar to $\tau(U^t)$-formulas using measure KT, a variant of Kt, were considered in [91].

4.2 Strong feasible disjunction property and the $\bigvee$-hardness

Assume we have a generator g with the stretch $n+1$. The simplest way how to increase the stretch is to compute g at parallel on many independent inputs. For $t \geq 1$ take map

$$t \times g \ : \ (x^1, \dots, x^t) \in \{0,1\}^{tn} \rightarrow (g(x^1), \dots, g(x^t)) \in \{0,1\}^{t(n+1)} \ . \quad (4.2.1)$$

The time to compute $t \times g$ is at most t-times longer than the time needed to compute g on size n inputs and the input size is tn. Hence irrespective of t this map will be p-time too.

For $b = (b^1, \dots, b^t) \in \{0,1\}^{t(n+1)}$ the $\tau(t \times g)_b$ formula looks as the disjunction

$$\bigvee_{i \leq t} \tau(g)_{b^i} \quad (4.2.2)$$

with all t $\tau(g)$ formulas in disjoint sets of atoms.

We have seen such a disjunction (of two formulas) at the beginning of Section 3.3 when introducing the pseudo-surjectivity. What we want is a notion of hardness of g, closer to the hardness rather than to the pseudo-surjectivity, that would imply that for (some range of t) the disjunction (4.2.2) is hard to prove.

Definition 4.2.1 ($\bigvee$-hardness, [68])

Let P be a proof system. Generator $g = \{C_n\}_n$ with stretch $m := m(n)$ is **$\bigvee$-hard** *for P iff for any $c \geq 1$ only finitely many disjunctions*

$$\tau(g_n)_{b^1} \vee \cdots \vee \tau(g_n)_{b^t} \ , \tag{4.2.3}$$

with $n, t \geq 1$ and all $b_i \in \{0,1\}^m$, have a P-proof of size at most m^c.

Note that we bound the size of proofs by a polynomial in m and not in the size of the disjunction (which is $O(tm^{O(1)})$).

I do not see a reason why the hardness of g ought to imply the $\bigvee$-hardness. However, for proof systems with a certain property - to be defined next - this will be true. The following notion was introduced in [58] for the purpose of an analysis of a particular generator (see also [65, Subsec.17.9.2]). The special case of two disjuncts was studied since early 1980s in propositional logic with several authors giving incorrect proofs of fdp for various strong systems. Later it was considered in [94] in a connection with the feasible interpolation method under the name *existential interpolation*.

Definition 4.2.2 (strong feasible disjunction property, [58])

Proof system P has the **strong feasible disjunction property** *(abbreviated* strong fdp*) iff there exists a constant $c \geq 1$ such that whenever a disjunction*

$$\bigvee_{1 \leq i \leq r} \alpha_i \tag{4.2.4}$$

of r formulas, no two having atoms in common, has a P-proof of size s then one of α_i has a P-proof of size $\leq s^c$.

The fdp without the qualification *strong* refers to the case of $r = 2$.

The strong fdp plays a role in analysis of a proof complexity generator in [58] (a remark at the end of Section 8.5, see also [65, Subsec.17.9.2]). Our intended use of the property is outlined by the next two lemmas.

Lemma 4.2.3

Assume a pps P has the strong fdp. Then any generator hard for P is also $\bigvee$-hard for P.

Lemma 4.2.4

Let g be a generator with stretch $n+1$ and assume that it is $\bigvee$-hard for a pps P.

Then for all $\delta > 0$ there is generator g' with the stretch $\geq n + n^{1-\delta}$ that is $\bigvee$-hard for P too.

Proof:

Take for $g' := t \times g$, where $t := n^c$ and $1/(c+1) \leq \delta$. It stretches $n' := n^{c+1}$ bits into $\geq n' + (n')^{1-\delta}$ bits.

q.e.d.

The lemmas suggest that for proof systems with the strong fdp we can always extend a stretch of a hard generator almost to $2n$. But the issue is that no strong proof systems having the strong fdp are known. In particular, it is an open problem ([65, Prob.17.9.1]) whether, for example, EF has the (strong) fdp. As a corollary to some proofs of the Feasible interpolation theorem for resolution (cf. [48], [65, Chpt.17]) it can be seen that resolution R has the strong fdp. On the other hand, a proof systems $R(k)$ of [49], a mild extension of R, has no fdp, cf. [26].

There is, however, a way out if we remember what our main goal is: to show that no proof system is p-bounded. It was pointed out in [58] that for the purpose of proving lengths-of-proofs lower bounds for some pps P we may simply *assume* w.l.o.g. that P satisfies the strong fdp.

Lemma 4.2.5

Assume a proof system P has no strong fdp. Then it is not p-bounded.

Proof:

As the disjunction (4.2.4) has a proof it is a tautology. This implies, using that sets of atoms of different α_i are disjoint, that one of α_i is a tautology. It would have a p-size P-proof if P were p-bounded.

q.e.d.

This means that for the purpose of developing the theory and extending the stretch we may assume the strong fdp: if the assumption is incorrect then we do not need to bother with any theory.

Next we give a limitation on the strong fdp, assuming the working conjecture 3.2.2 and hypothesis from [33] underlying universal derandomization. We first employ the latter to show that the dWPHP can be witnessed by S-T computations with polynomially many rounds by a rather lazy student: he does not care what the teacher says.

Denote by $\mathrm{Size}^A(s(k))$ the class of languages L such that L_k, all $k \geq 1$, can be computed by a circuit of size $\leq s(k)$ that is allowed to query oracle A.

Lemma 4.2.6 ([49, Sec.7])

Assume that there is $L \in \mathcal{E}$ such that for every $\mathcal{NP}$ set A there is $\epsilon > 0$ such that $L \notin Size^A(2^{\epsilon k})$.

Let g be a p-time generator with the stretch $n + 1$. Then the formula dWPHP(g) can be witnessed by an S-T computation with a p-time student within $n^{O(1)}$ rounds and the student does not uses the counter-examples provided by the teacher.

Proof:

Assume g is computed in time n^k. The construction in [33] yields, under the hypothesis of the lemma, a pseudo-random generator

$$G \ : \ \{0,1\}^{O(\log n)} \to \{0,1\}^{n+1}$$

such that no non-deterministic algorithm running in time $O(n^k)$ can distinguish random elements of $\{0,1\}^{n+1}$ from pseudo-random ones from $rng(G)$. In particular, it must holds that

$$rng(G) \ \not\subseteq rng(g)$$

as otherwise the property to belong to $rng(g)$ would yield a discrepancy at least $1/2$ in the probability of accepting random and pseudo-random elements, respectively.

Hence even a Student unwilling to learn anything from the Teacher may simply produce in succession all elements of $rng(G)$ as candidate solutions, waiting until the Teacher gives up an accepts one as correct.

q.e.d.

Theorem 4.2.7

Assume that there is $L \in \mathcal{E}$ such that for every $\mathcal{NP}$ set A there is $\epsilon > 0$ such that $L \notin Size^A(2^{\epsilon k})$. Assume also that the working conjecture 3.2.2 holds true for a p-time generator g.

Then there exists a proof system Q such that no strong proof system P that simulates Q has the fdp.

Proof:

Take the function G from Lemma 4.2.6 and let its domain be $\{0,1\}^{c \log n}$ for definiteness. The fact that $rng(G) \not\subseteq rng(g)$ means that formulas

$$\bigvee_{i < c \log n} \tau(g)_{b^i}$$

where $\{b^i\}_{i < c \log n}$ enumerates $rng(G) \cap \{0,1\}^{n+1}$ are tautologies. Their set is p-time a hence we may consider a strong proof system Q that extends EF by all these formulas as extra axioms.

If P simulates Q it has, in particular, p-size proof of these disjunctions. If P had also the strong fdp it would mean that one of the disjuncts (for each $n \geq 1$) has a p-time P-proof. Hence g is not hard for P, contradicting the hypothesis.

q.e.d.

4.3 The truth-table function

The first systematic study of circuit complexity (and lower bounds, in particular) in weak formal systems is in [95] using first-order formalization in a particular formal system related to bounded arithmetic. The propositional side of things was emphasized in [51] where the truth-table function was considered as a proof complexity generator.

Note that a circuit with k inputs and of size $s \geq k$ can be encoded by $10s \log s$ bits which is less than 2^k if $s \leq 2^k/10k$.

Definition 4.3.1 (the truth-table function)

Given parameters $1 \leq k \leq s \leq 2^k/10k$ the **truth-table function $\mathbf{tt}_{s,k}$** *maps $\{0,1\}^n$ into $\{0,1\}^m$ where*

$$n := 10s \log s \ < \ m := 2^k$$

by interpreting $a \in \{0,1\}^n$ as a description of a size $\leq s$ circuit C with k inputs outputting $b := \mathbf{tt}_{s,k}(a) \in \{0,1\}^m$, where b is the truth-table computed by circuit C on inputs from $\{0,1\}^k$.

Note that $\mathbf{tt}_{s,k}$ is indeed a uniform generator in the sense of Definition 3.1.2 as it is computed in time polynomial in m. Note that the dWPHP formula for the truth-table function is Σ^b_2 and not Σ^b_1 as it is sometimes claimed even for as small s as $s = O(k)$.

The τ-formula $\tau(\mathbf{tt}_{s,k})_b$ expresses that the Boolean function on $\{0,1\}^k$ whose truth-table is b has circuit complexity bigger than s. Proving such statements is the holy grail of circuit complexity and this makes these τ-formulas attractive.

The function has a key property related to the dWPHP problem 2.0.1.

Theorem 4.3.2 ([35, Cor.3.6])

Let $1 > \epsilon > 0$ be arbitrary rational and let $s := 2^{\epsilon k}$. Then dWPHP($\mathbf{tt}_{s,k}$) implies over $S^1_2(PV_1)$ instances of the dWPHP for all p-time functions.

The requirement that ϵ is rational allows to define the value of s in the theory.

The propositional side of things is represented by Theorem 4.3.5 stating that the truth-table function is the hardest generator w.r.t. to the pseudo-surjectivity.

To motivate its proof think about a way how to iterate a generator g having the minimal required stretch $n+1$. We may apply it first repeatedly to first n bits of the output to generate in n rounds $2n$ bits from its original n bits; let g' be this enhanced generator with the stretch $2n$. Then we may iterate g' itself applying it always at parallel to the first n bits and to the last n bits of the output, getting in t parallel rounds $2^t n$ output bits. Observe that to compute this function we compute g' locally at nodes of a binary tree of depth t $(2^t - 1)$-times, hence we compute the original g $n(2^t - 1)$-times. Taking for $t := (c-1)\log n$ we can get a generator g'' with the stretch n^c. Moreover, to compute any particular bit of a string in $rng(g'')$ we need to compute g' at most $((c-1)\log n)$-times along a particular branch in the binary tree underlying the iteration of g'. Similarly, if we want to get a stretch m then to compute any bit of any string in the range of such generator will need $\log m$ calls to g' and hence $n \log m$ calls to original g.

A general form of such an iteration is captured by the following notion.

Definition 4.3.3 (iteration protocol, [51])

An **iteration protocol** Θ *for circuit* C *with* n *inputs and* $m > n$ *outputs is a sequence of instructions*

$$C(u^1) = v^1, C(u^2) = v^2, \ldots, C(u^t) = v^t$$

where

- *each* u^i *is an* n*-tuple of distinct atoms,*
- *each* v^i *is an* m*-tuple of distinct atoms,*
- *every atom occurs in at most one* u^i *and in at most one* v^i*,*
- *if an atom occurs in some* u^i*,* $i > 1$*, then it also occurs in some* v^j *with* $j < i$*.*

Here atoms u^1 *are inputs of the protocol and atoms* v^i_j *that do not occur in any* u^r *are outputs of* Θ*. The size of the protocol is defined to be* t*.*

Protocol Θ *defines a circuit* $Iter(C/\Theta)$ *computed by iterating* C *along protocol* θ*; its input and output variables are those (atoms) of* Θ*.*

The following statement is a simplified version of [51, Thm.3.4].

Theorem 4.3.4 ([51, Thm.3.4])

Let P *be a strong proof system. Assume* $g = \{C_n\}_n$ *is a generator with the stretch* $m = m(n)$ *that is pseudo-surjective for* P*. Let* $\Theta_n := C_n(u^1) = v^1, C_n(u^2) = v^2, \ldots, C_n(u^t) = v^t$ *be iteration protocols with* $t \leq m^c$*, for some constant* $c \geq 1$ *and* $n \geq 1$*.*

Then the generator h *defined by circuits* $\{Iter(C_n/\Theta)\}_n$ *is pseudo-surjective for* P *too.*

If g *is exponentially pseudo-surjective for* P *and* t *is sub-exponential,* $t \leq 2^{m^{o(1)}}$*, then* h *is exponentially pseudo-surjective for* P*.*

Proof:

Denote the circuit $Iter(C_n/\Theta)$ simply D_n, so $h = \{D_n\}_n$. Assume it is not pseudo-surjective and, in particular, that P proves in size $\leq m^b$ infinitely many disjunctions

$$\tau(D_n)_{B^1} \vee \cdots \vee \tau(D_n)_{B^r}$$

with circuits B^i having the properties as required in Definition 3.3.1. The idea is simple: replace everywhere D_n by its definition from C_n via Θ. The following claim is utilized to show that the proof after the substitution does not increase too much.

Claim: *The formula $\neg\tau(D_n)_y(x)$ follows from the negations of the formulas in Θ by a P-proof of size $O(t)$ where x are the variables u^1 and y are the output variables of Θ.*

The claim can be established by induction on t (cf. [51, Sec.3]) for details.

q.e.d.

We formulate the following statement for strong proof systems as it allows for a simpler model-theoretic proof (and strong proof systems are our target). This argument illustrates better, I think, what is going on.

However, the theorem holds for proof systems containing resolution R and the reader can find the original proof-theoretic argument for that more general case in [51, Sec.4]. It is also that argument that generalizes to iterability in Theorem 4.3.7.

Theorem 4.3.5 ([51, Thm.4.2])

Assume P is a strong proof system P. Then the following two statements hold:

1. *There exists a generator g with the stretch $n+1$ which is (exponentially) pseudo-surjective for P iff for any $0 < \delta < 1$, the truth table function $\mathbf{tt}_{s,k}$ with $s = 2^{\delta k}$ is (exponentially) pseudo-surjective for P.*

2. *There exists a generator g with stretch $n+1$ which is exponentially pseudo-surjective for P iff there is $c \geq 1$ such that for $s = k^c$ the truth table function $\mathbf{tt}_{s,k}$ is exponentially pseudo-surjective for P.*

Proof:

The if-parts of both statements are obvious. We shall prove the only-if-part of statement 1 for pseudo-surjectivity; the exponential version and statement 2 are proved analogously choosing suitable parameters.

Let g be a p-time generator with the stretch $n+1$ which is pseudo-surjective for P. We shall use Theorem 3.5.3 so let $\mathbf{M}_m$ be an arbitrary small canonical model; the theorem gives us its extension $\mathbf{M}'$ such that

$$\mathbf{M}' \models T + S^1_2(\mathrm{PV}) + rng(g_n) = \{0,1\}^{n+1}$$

where $T \supseteq \mathrm{PV}_1$ is a true $\forall\Pi_1^b$-theory corresponding to P.

Now perform in $\mathbf{M}'$ the iteration of g described before Definition 4.3.3: get g' with the stretch $2n$ and then g'' with the stretch n^c. As observed there, any particular bit of the string in $b := g''(a)$ for $a \in \{0,1\}^n$ can be computed with at most $cn \log n$ calls to the original g. That is, for any fixed $c \geq 1$ the bits of b can be computed using as advice a in time $< n^{d+2}$ (i.e. by a circuit of size $\leq n^{d+2}$) for $n >> 1$, where n^d is the time needed to compute g.

For $\epsilon > 0$ and $d \geq 1$ fixed put $k := (d+2)(\log n)/\epsilon$ and choose $c \geq 1$ such that for $s := 2^{\epsilon k}$

$$n^c \geq 2^k \ .$$

We want to argue that $\mathbf{tt}_{s,k}$ is in $\mathbf{M}'$ onto $\{0,1\}^m$ where $m = 2^k$ and hence it is (by Theorem 3.5.3) pseudo-surjective.

To show that $\mathbf{tt}_{s,k}$ is onto it suffices to show that any string b as above is equal to $g''(a)$, for some $a \in nn$. This is established using induction on t in the definition of g'', quite similarly as it is in the proof of the WPHP in [84]. The induction is on the length (we have $t \leq O(lonn)$) and for a Σ_1^b-formula, hence it can be performed in $\mathbf{M}'$ as that is a model of $S_2^1(\mathrm{PV})$.

q.e.d.

In general we shy away in these lecture notes from proving results about very weak proof systems but we make an exception now and modify the preceding theorem so that it can be used (in next section) more readily for resolution or alike weak system. The problem with the pseudo-surjectivity for weak proof systems is that weak system handle poorly general circuits B^i that appear in Definition 3.3.1. This lead to the following definition.

Definition 4.3.6 (iterability, [51])

Assume a proof system P simulates resolution R. A generator $g = \{C_n\}_n$ with stretch $n+1$ is **iterable** *for P iff it satisfies conditions of Definition 3.3.1 with the restriction that circuits B^i, $1 \leq i \leq t$, are just substitutions of constants and atoms for atoms.*

Similarly to pseudo-surjectivity we say that g is **exponentially iterable** for P if the lower bound in Definition 3.3.1 is exponential $2^{m^{\Omega(1)}}$

The following theorem can be proved analogously as the original proof of Theorem 4.3.5 in [51, Sec.4] (there are some technicalities about how are circuit encoded).

Theorem 4.3.7

Theorem 4.3.5 is true for the iterability in place of the pseudo-surjectivity too.

4.4 Hardness of the truth-table function

The $\tau(\mathbf{tt}_{s,k})$-formulas express circuit lower bounds $> s(k)$ and thus the hardness of $\mathbf{tt}_{s,k}$ means that no such lower bound has a feasible proof for any specific (function given by) truth-table $b \in \{0,1\}^{2^k}$. This should not be confused with the provability of the existence of hard function: this is just the dWPHP($\mathbf{tt}_{s,k}$) formula. For example, a simple counting argument proves that most functions are hard but even in full ZFC we do not know how to prove in p-size (any fixed polynomial) any statement

$$b \notin rng(\mathbf{tt}_{s.k})$$

for any specific b with $k >> 0$.

In this section we present several statements showing that the truth-table function is unlikely to be hard for all proof systems but that finding any proof system for which it is not hard is likely a very difficult task itself. We shall also give unconditional result about resolution R to be used in later chapters.

Recall that for function $s(k)$ the class $Size(s)$ is the class of languages L whose characteristic functions χ_L on $\{0,1\}^k$ can be computed by circuits of size $\leq s(k)$. The infinitely-often symbol $\mathcal{C} \subseteq_{i.o.} \mathcal{C}'$ used in the next lemma means that for all $L \in \mathcal{C}$ it holds that $L \in_{i.o.} \mathcal{C}$, and this means that there is a language L' in class $\mathcal{C}'$ such that $L_k = L'_k$, the restrictions of the languages to input length k, holds for infinitely many lengths $k \geq 1$.

Lemma 4.4.1

Let $1 \leq k \leq s = s(k) \leq 2^{k/2}$ *and assume that*

$$\mathcal{NE} \cap co\mathcal{NE} \not\subseteq_{i.o.} Size(s) \ .$$

Then there exists a strong proof system for which $\mathbf{tt}_{s,k}$ *is not hard.*

Proof:

For any specific language $L \in \mathcal{NE} \cap co\mathcal{NE}$ and its characteristic function $g := \chi_L$, the set of the truth tables of g_k, $k \geq 1$, is in $\mathcal{NP}$.

Define a proof system extending EF whose proofs of a formula φ are either EF-proofs or, if $\varphi = \tau(\mathbf{tt}_{s,k})_b$ for b, the $\mathcal{NP}$-witnesses that φ is the truth-table of g_k.

q.e.d.

Lemma 4.4.2

Let $1 \leq k \leq s = s(k) \leq 2^{k/2}$.

1. *If for some proof system and for some $s(k) \geq 2^{\Omega(k)}$ the function $\mathbf{tt}_{s,k}$ is not hard for P then $\mathcal{BPP} \subseteq_{i.o.} \mathcal{NP}$.*

2. *If for some proof system and for some $s(k) \geq k^{\omega(1)}$ the function $\mathbf{tt}_{s,k}$ is not hard for P then $\mathcal{NEXP} \not\subseteq \mathcal{P}/poly$.*

Proof:

For the first statement, we can use the hypothesis and modify the construction of [81, 33] derandomizing BPP a bit:

1. guess a pair (b, π), where $b \in \{0,1\}^{2^k}$ is the truth-table of a function with circuit complexity $\geq s(k)$ and π is a size $m^{O(1)} = 2^{O(k)}$ P-proof of $\tau(\mathbf{tt}_{s,k})_b$,

2. use b as in [81, 33].

To prove the second statement we use that by [31] $\mathcal{NEXP} \not\subseteq \mathcal{P}/poly$ holds if one could certify by p-size strings a super-polynomial circuit complexity of a function. This is exactly what the hypothesis guarantees.

q.e.d.

Several possibilities how the hypotheses of the two lemmas may arise were discussed in [60, Sec.30.1] (Possibilities A, B, and C there).

We shall now present two results about resolution R as they are going to be used in some applications in Chapter 9.

The following statement is proved analogously as Theorem 3.6.1, using the concept of *natural proofs* of [98] and PRNGs (see Section 3.6).

Lemma 4.4.3 ([60, Thm.29.2.3])

Assume that for some $\epsilon > 0$ *there exists a PRNG* g *with exponential hardness* $H(g) \geq 2^{n^\epsilon}$. *Let* $s(k) \geq k^{\omega(1)}$.

Then the truth-table function $\mathbf{tt}_{s,k}$ *is hard for any proof system* P *that simulates resolution* R *and admits feasible interpolation. In particular, the function is hard for* R.

The statement was generalized in [90]. In a subsequent development [91] link the hardness of the truth-table function for EF to one of the conjectures from [99] mentioned at the end of Section 3.6.

The next theorem follows immediately from Theorem 4.3.7 and Theorem 5.2.2 to be discussed in Section 5.2.

Theorem 4.4.4

There is $c \geq 1$ *such that the truth-table function with* $s(k) = k^c$ *is exponentially iterable for* R.

Chapter 5

Nisan-Wigderson generator

The Nisan-Wigderson generator (NW generator, for short) is a fundamental object of computational complexity. It was taken up in [5] as a model for a class of generators that could be hard proof complexity generators. A variant of the construction was proposed as a non-uniform candidate for a generator hard for all proof systems in [51].

5.1 The definition and its variants

The **Nisan-Wigderson generator** is determined by

- an $m \times n$ 0-1 matrix A with ones in row i exactly in positions $j \in J_i := J_i(A)$ where:

$$J_i(A) = \{j \in [n] \mid A_{ij} = 1\} \ , \quad \text{for } i \in [m] \ ,$$

- an ℓ-ary Boolean function f.

There is an additional parameter $d \geq 1$ and matrix A is required to be a **(d, ℓ)-design**:

- $|J_i| = \ell$, all $i \in [m]$,
- $|J_u \cap J_v| \leq d$ for all different $u \neq v \in [m]$.

Combinatorial designs with various ranges of parameters were shown to exists via various arguments in [81, Sec.2].

The Nisan-Wigderson generator $NW_{A,f}(x)$ maps $\{0,1\}^n$ into $\{0,1\}^m$ and the i-th bit of the output of the generator on $x \in \{0,1\}^n$ is:

$$f(x(J_i)) \ , \quad \text{where} \ \ x(J_i) = x_{j_1}, \dots, x_{j_\ell}$$

for $J_i = \{j_1 < \cdots < j_\ell\}$. The role NW generator plays in computational complexity theory can be hardly overestimated.

It was suggested in [5] that the NW generator, when based on a *function* f *that is hard to handle in a particular proof system* P, could be hard for P as a proof complexity generator. The expression that f is *hard to handle* means that f may not be definable by formulas P operates with or, if it is, P does not prove its basic properties.

Some proofs in [5] (and subsequently in [97, 51] too) used extra combinatorial requirements on matrix A.

A **boundary** $\partial_A(I)$ of a set of rows $I \subseteq [m]$ is the set

$$\{j \in [n] \mid \exists! i \in I \ A_{ij} = 1\}$$

($\exists!$ means exists exactly one). For $1 \leq r \leq m$ and $\epsilon > 0$ any parameters, matrix A is an (r, ϵ)**-expander** iff for all $I \subseteq [m]$, $|I| \leq r$, $|\partial_A(I)| \geq \epsilon\ell|I|$.

Expanders simulate, in a sense, matrices with disjoint sets $J_i(A)$'s of the maximum size ℓ. In such a case it would hold that $|\partial_A(I)| = \ell|I|$. An (r, ϵ)-expander achieves (as long as $|I| \leq r$) at least an ϵ-percentage of this maximum value.

The existence of expanders can be proved by a probabilistic argument. A matrix A is called ℓ**-sparse** if each rows contains at most ℓ ones.

Theorem 5.1.1 ([5, Thm.5.1])

For every $\delta > 0$ *there is an* $\ell \geq 1$ *such that for all sufficiently large* n *there exists* ℓ*-sparse* $n^2 \times n$*-matrix that is an* $(n^{1-\delta}, 3/4)$*-expander.*

Another combinatorial notion is a (r, d)**-lossless expander** (cf. [97]) requiring that A satisfies for all sets of rows $I \subseteq [m]$:

$$|I| \leq r \ \rightarrow \ \sum_{i \in I} |J_i(A)| \ - \ |\partial_A(I)| \ \leq \ d|I| \ .$$

Their existence is proved via a probabilistic argument.

Theorem 5.1.2 ([97, Thm.2.5])

For sufficiently small $\epsilon > 0$ and large enough $n \geq 1$ there exists an $m \times n$ matrix A that is an $(n^{\Omega(1)}, O(\log m / \log n))$-lossless expander and $m \geq 2^{m^{\epsilon}}$.

A different variant of the NW construction was proposed in [51, Sec.2] as a candidate (non-uniform) proof complexity generator hard for EF and possibly for stronger systems. Namely, we take constant $c \geq 1$ and put $m := n + 1$ and $\ell := c \log n$. The proposed generator is $NW_{A,f}$ where A and f are chosen at random. A similar construction but of a one-way function was proposed earlier in [27]: it uses $n \times n$ matrix and c a constant.

One can view the resulting τ-formulas as stating that a system of random sparse equations is unsolvable. The proposal was motivated by my view at the time that randomness of the system will play a role.

5.2 Iterability of NW-like linear maps

A number of lower bound results for weak proof systems as is R, PC or PCR (cf.[65]) were proved in [5, 51, 97] about NW-like maps where the underlying function f is the parity function. That is, the generator $NW_{A,f}$ is a linear map.

We will just state two results that we shall use in one of the applications in Chapter 9 (Section 9.1, in particular). The proofs use the notion of iterability (Definition 4.3.6) and we will not give them as I do not think they can be helpful to understanding strong proof systems. The interested reader is advised to consult the original sources: [51, L.19.4.4] and [97], respectively.

Theorem 5.2.1 ([51, Thm.6.6])

For every $\delta > 0$ there is an $\ell \geq 1$ such that for all sufficiently large n there exists an ℓ-sparse $n^2 \times n$-matrix A such that the linear map $NW_{A,\oplus}$ from $\{0,1\}^n$ into $\{0,1\}^{n^2}$ defined by A is an exponentially hard proof complexity generator for resolution R.

The following theorem was proved actually for $R(\Omega(\log\log))$, a DNF-resolution proof system of [49]. The second item is deduced from the first one by applying a iteration protocol along a complete binary tree of suitable depth as in the proof of the WPHP in bounded arithmetic in [84] or in the construction of pseudo-random function generator in [28].

Theorem 5.2.2 ([97, Thms.2.10 and 2.12])
There is an $\epsilon > 0$ *such that for* $n \geq 1$ *large enough:*

1. *there is a linear map* $\{0,1\}^n \to \{0,1\}^{2n}$ *that is exponentially iterable for resolution* R*,*
2. *there is a linear map from* $\{0,1\}^n \to \{0,1\}^m$ *with* $m := 2^{n^\epsilon}$ *which is an exponentially hard for* R*.*

Let us mention an open problem.

Problem 5.2.3 (Linear generators, [65, Probs.19.4.5 and 19.6.1])
Is the linear map from Theorem 5.2.1 also (exponentially) hard for AC^0*-Frege systems? Is it, in fact, exponentially iterable for the system?*

5.3 Razborov's conjecture

A function$f : \{0,1\}^* \to \{0,1\}$ is an $\mathcal{NP} \cap co\mathcal{NP}$**-map** iff the language whose characteristic function f is in $\mathcal{NP} \cap co\mathcal{NP}$. Note that if f is an $\mathcal{NP} \cap co\mathcal{NP}$-map then the complement of the range of a generator $g := NW_{A,f}$ is in $co\mathcal{NP}$ and hence the associated τ-formulas $\tau(g)_b$ can still be expressed by propositional tautologies. These are translations of

$$\bigvee_{i \in [m]} f(x(J_i(A))) \neq b_i$$

which can be written as

$$\bigvee_{i \in [m]} \neg A_{b_i}(x(J_i(A)), z^i) \tag{5.3.1}$$

where $\exists v(|v| \leq \ell^c) A_a(u,v)$ is an $\mathcal{NP}$-definition of $f(u) = a$, for $a = 0,1$.

Taking advantage of this [97] made the following conjecture.

Conjecture 5.3.1 (Razborov's conjecture [97, Conj.2])
Any generator $NW_{A,f}$ *based on a matrix* A *which is a combinatorial design with the same parameters as in [81] and on any function* f *in* $\mathcal{NP} \cap co\mathcal{NP}$ *that is hard on average for* $\mathcal{P}/poly$*, is hard for* EF*.*

An example of function f that can feature in the conjecture is $B(h^{(-1)}(y))$ where h is a one-way permutation (OWP) and B is a hard bit of h.

There are several sets of parameters in [81] but the parameters mentioned in the conjecture are, I suppose, those used in [81, L.2.5]:

$$d = \log(m)\ ,\ \log(m) \le \ell \le m\ ,\ n = O(\ell^2)\ . \tag{5.3.2}$$

The **hardness on average** of f is measured by the minimum S for which there is a size $\le S$ circuit C such that

$$\mathrm{P}rob_{u\in\{0,1\}^\ell}[f(u) = C(u)]\ \ge\ \frac{1}{2} + \frac{1}{S}\ .$$

The requirement in [81] is that the hardness is $2^{\Omega(\ell)}$, and they also require that it is at least m^2.

Considering all these constrains we are lead to the following set of parameters (for any $\epsilon > 0$):

$$m = 2^{n^\delta}\ ,\ d = \log m\ ,\quad \text{and}\quad \ell = n^{1/3} \tag{5.3.3}$$

where $0 < \delta \le 1/3$ is arbitrary. The huge size of m w.r.t. n means that going through all possible arguments and all possible $\mathcal{NP}$ witnesses in the definition of f takes quasi-polynomial time in m. This yields the following observation.

Lemma 5.3.2

The τ-formulas attached to any generator $NW_{A,f}$ whose parameters satisfy (5.3.3) are provable in quasi-polynomial size $m^{(\log m)^{O(1)}}$ in resolution R.

That leaves rather narrow gap for lower bounds for the τ-formulas.

There are two issues with the formulation of the conjecture we ought to be aware of. The first issue is that generators are, according Definition 3.1.2, computed by circuits on finite domains $\{0,1\}^n$ and this a priori implies that they are maps. That is, the syntactic form of the definition implies that. This is not the case with $g = NW_{A,f}$ with f an $\mathcal{NP} \cap co\mathcal{NP}$-function. Given $\mathcal{NP}$-definitions A_0, A_1 as in (5.3.1) we do not know a priori that they define two complementary $\mathcal{NP}$ sets. Clearly, we cannot express propositionally that $A_0 \cup A_1 = \{0,1\}^*$, i.e. that f is a total function. This is perhaps not such a problem as having only partial f can make the τ-formula only harder to prove.

The disjointness of A_0 and A_1 is expressed by (a sequence of) tautologies. However, we may not be able to prove them shortly, i.e. we may not be able to shortly prove that f has unique values. (One can say that in that case we already have a lower bound so we do not have to trouble ourselves with the τ-formulas.) If we accept this situation the τ formulas could be hard irrespective of how hard it is to compute f. Namely, take two disjoint $\mathcal{NP}$ sets U, V whose disjointness is hard for EF; such a pair exists if EF is not an optimal proof system (cf. [70, 65]). Then for any function f separating U from V EF cannot prove feasibly that $f(u) \neq b$ for either $b = 0, 1$. In fact, in this case oen can take simply $J_1(A) = \cdots = J_m(A)$ and still get hard g.

If the reader started now to see $\mathcal{NP} \cap co\mathcal{NP}$ maps as somewhat opaque object (as I did) let us point out that [97, Conj.1] formulates also a conjecture about Frege systems where function F is p-time (and has a suitable hardness property).

The second issue with the conjecture is the choice of parameters. Note that the size of the τ-formulas will be polynomial in m even if we allow f to come from a larger class $\mathrm{NTime}(m^{O(1)}) \cap co\mathrm{NTime}(m^{O(1)})$. However, this seemingly innocent change leads to a rather dramatic behavior of the conjecture.

Solely for the purpose of stating the next theorem we formulate separately the modification of Conjecture 5.3.1 with the time requirement on f changed.

Statement (R):

Assume that for some $\epsilon > 0$ parameters n, d, ℓ, m satisfy (5.3.3). Let $g = NW_{A,f}$ where A is an $m \times n$ matrix that is an $(l, \log m)$-design and function f is in $NTime(m^{O(1)}) \cap coNTime(m^{O(1)})$.

Then g is hard for EF.

Theorem 5.3.3 ([52, Thm.4.2])

Assume Statement (R) is true. Then EF is not p-bounded.

Proof:

We shall prove the statement contrapositively: assume that EF is p-bounded. Then, in particular, $\mathcal{NP} = co\mathcal{NP}$.

By [52, Thm.3.1(ii)] there is then $L \in \mathcal{NE} \cap co\mathcal{NE}$ that is exponentially hard for $\mathcal{P}/poly$. A direct argument: the lexicographically first string in which is a truth table of a function on $\{0,1\}^\ell$ with any specific exponential hardness on average is in the polynomial-time hierarchy and hence the function it defines is in $\mathcal{E}$ with an oracle access to the p-time hierarchy. But under the hypothesis that $\mathcal{NP} = co\mathcal{NP}$ the function is, in fact, in $\mathcal{NE} \cap co\mathcal{NE}$.

Having such function f, assuming Statement (R) a taking a matrix A with suitable parameters that is constructed in [81], we derive that EF is not p-bounded. That is a contradiction.

q.e.d.

The reader inclined to think positively may conclude that in order to prove lower bounds for EF we only need to establish conditional lower bounds in Statement (R). Less optimistic reader may wonder whether the assumption in Statement (R) plays any role at all if the conclusion holds anyway.

Or perhaps it shows that the parameters in the original conjecture are right and play an essential role. I only wish we had any idea what that role could be; unfortunately it is not discussed in [97].

I think that because of the two issues (more values for f and time constraint on f) it may be better to study the conjecture for some specific f that avoids both of them. For example, take f to be a hard bit of the RSA (e.g. the parity bit), as EF admits p-size proofs of its injectivity, cf. [73] (it is proved there in $S^1_2(\mathrm{PV})$, use the propositional translations).

Let us point out that there are some results about the conjecture for weaker proof systems than is EF:

- The conjecture holds for all proof systems which admit feasible interpolation in place of EF (in fact, it holds under weaker assumptions on A and f), cf. [85, 86].
- The variant of the conjecture with the hardness of f replaced by the requirement that f needs exponential size depth 2 circuits is true for AC^0-Frege systems and a particular definition of the τ-formulas, cf. [40].

We do not present the proofs here as they use special properties of the particular proof systems and cannot be - even in principle - generalized to strong proof systems.

5.4 Limitations of $\mathcal{NP} \cap co\mathcal{NP}$ NW-generators

Statement (R) in the previous section altered the original formulation of Conjecture 5.3.1 by allowing more time to compute the function f the NW-generator uses. Here we stick to the original formulation but consider whether the generator could be actually hard for all proof systems. Such a variant of the conjecture was studied in [58, 62] under the name *Statement (S)*. We shall study it more in Section 8.5. The construction in [58] uses a simplifying technical assumption that the non-deterministic witnesses for (values of) f are unique. This can be arranged by taking for f a hard bit of a OWP. We incorporate it into the formulation of this variant of the conjecture. Recall the notion of the hardness on average for the previous section.

Statement (S):

Assume that for some parameters n, d, ℓ, m satisfy (5.3.3), that is:

$$m = 2^{n^\delta} \ , \ d = \log m \ , \quad \text{and} \quad \ell = n^{1/3}$$

where $0 < \delta \leq 1/3$ is arbitrary. Let h be a p-time OWP with exponential hardness on average and $B(x)$ its hard bit, and assume

$$f(y) := B(h^{(-1)}(y)) \ .$$

Let A_n be $m \times n$ matrices that are $(l, \log m)$-design and such that their i-th row J_i is computable in p-time from i and $1^{(n)}$.

Then $NW_{A_n,f}$ is hard for all proof systems.

Suitable matrices A having the property required in the statement are constructed in [81, L.2.5].

Recall that the infinitely-often symbol $L \in_{i.o.} \mathcal{C}$ used in the second hypothesis of the next theorem denotes that there is a language L' in class $\mathcal{C}$ such that $L_k = L'_k$, the restrictions of the languages to input length k, holds for infinitely many lengths $k \geq 1$. An example of a plausible L satisfying the second hypothesis is TAUT.

Theorem 5.4.1 ([62, L.6.1])

Assume that:

- *OWP exponentially hard on average exist,*
- *there exists $L \in \mathcal{NE} \cap co\mathcal{NE}$ such that $L \notin_{i.o.} \mathcal{NP}/poly$.*

Then Statement (S) is not true; that is, the generator $NW_{A_n,f}$ is not hard for all proof systems.

Proof:

Assume both hypotheses of the theorem and fo the sake of a contradiction also that (S) is true. Setting $k := n^\delta$ we can think of strings from $\{0,1\}^m$ as of the truth-tables of characteristic function of languages on $\{0,1\}^k$; for L a language denote by L_k also its characteristic function restricted to $\{0,1\}^k$.

Note that for any $L \in \mathcal{NE} \cap co\mathcal{NE}$ the set $\{L_k \mid k \geq 1\}$ is in $\mathcal{NP}$. It thus follows from (S) that for all $L \in \mathcal{NE} \cap co\mathcal{NE}$ it holds:

- *For infinitely many $n \geq 1$ and $k = n^\delta$ and $m = 2^k$:*

$$L_k \in \{0,1\}^m \cap rng(NW_{A_n,f}) \ .$$

Now choose $L \in \mathcal{NE} \cap co\mathcal{NE}$ that satisfies the second hypothesis. Take any $L_k \in \{0,1\}^m \cap rng(NW_{A_n,f})$ and $a \in \{0,1\}^n$ such that $L_k = NW_{A_n,f}(a)$. This allows us to compute whether $i \in L$ for $i \in \{0,1\}^k$ by evaluating f on $a(J_i)$. But by the condition on matrices A_n the set $a(J_i)$ can be done by a p-time algorithm from inputs $a, i, 1^{(n)}$ and f is $\mathcal{NP} \cap co\mathcal{NP}$.

This is a contradiction.

q.e.d.

The proof of this theorem relates to other constructions in [62] that we shall discuss in Section 9.2.

Chapter 6

Gadget generator

In this chapter we present a p-time generator defined in [57]. It is this generator we pin on our hopes for future developments.

6.1 The definition

Let

$$f \ : \ \{0,1\}^{\ell} \times \{0,1\}^{k} \ \rightarrow \ \{0,1\}^{k+1}$$

be a p-time function where $\ell = \ell(k)$ depends on k. We shall call any such function a **gadget function**.

Note that w.l.o.g. we could take for gadget functions the circuit value function CV we saw in Section 2.1. Namely, let $CV_{k,a}(u,v)$ be the version of CV which for $u \in \{0,1\}^k$ interprets $v \in \{0,1\}^a$ as (a description of a) circuit C_v with k inputs and $k+1$ and outputs $C_v(u) \in \{0,1\}^{k+1}$.

Definition 6.1.1 (gadget generators, [57])
Let f be a gadget function. The **gadget generator based on** f

$$Gad_f \ : \ \{0,1\}^n \rightarrow \{0,1\}^m$$

where

$$n := \ell + k(\ell+1) \quad and \quad m := n+1$$

is defined as follows:

1. *Input* $x \in \{0,1\}^n$ *is interpreted as* $\ell+2$ *strings*

$$v, u^1, \ldots, u^{\ell+1}$$

where $v \in \{0,1\}^\ell$ *and* $u^i \in \{0,1\}^k$ *for all* i.

2. *Output* $y = Gad_f(x)$ *is the concatenation of* $\ell+1$ *strings* $w^s \in \{0,1\}^{k+1}$ *where* w^s *are defined by the gadget function:*

$$w^s := f(v, u^s) .$$

Denote by f_v the function $\{0,1\}^k \to \{0,1\}^{k+1}$ computed by the gadget function for fixed gadget v. Using this notation the τ-formulas for Gad_f can be written as

$$\tau(\mathrm{Gad}_f)_b = \bigvee_{s \in [\ell+1]} \tau(f_v)_{b^s}$$

where the only common atoms among the formulas $\tau(f_v)_{b^s}$ are those ℓ corresponding to bits of v.

For another view of the τ-formula define, for $e \in \{0,1\}^{k+1}$, an $\mathcal{NP}$ set A_e to be the set

$$A_e := \{v \in \{0,1\}^\ell \mid \exists u \in \{0,1\}^k f_v(u) = e\} .$$

Then the formula $\tau(\mathrm{Gad}_f)_b$ is a tautology iff

$$\bigcap_{s \in [\ell+1]} A_{b^s} = \emptyset .$$

Both these examples show that the $\bigvee$-hardness from Section 4.2 ought to play a significant role in analyzing the gadget generator.

6.2 The $\bigvee$-hardness and gadget size

Let us denote the gadget generator Gad_f based on $f = CV_{k,k^2}$ simply Gad_{sq}. As a circuit of size s can be encoded by $10s \log s$ bits the circuits entering function Gad_{sq} are of size little bit less than quadratic. Note that the generator Gad_{sq} itself is computed in time $\leq n^{3/2}$.

The next theorem shows simultaneously that we can limit the size of the gadget and that non-uniformity of generators is not needed when $\bigvee$-hardness is used instead of the mere hardness.

Theorem 6.2.1 ([57])

Let P be a strong proof system. Assume that there exists a $\mathcal{P}/poly$ generator $g = \{C_k\}_k$ that is $\bigvee$-hard for P.

Then the p-time gadget generator Gad_{sq} based on CV_{k,k^2} is $\bigvee$-hard for P too.

Proof:

Assume P and g satisfy the hypotheses and that (w.l.o.g.) the stretch of g is $n+1$. Assume that circuits C_k computing g_k are encoded by $\ell \leq k^a$ bits, for some constant $a \geq 1$.

Claim 1: *Gad_f with $f := CV_{k,k^a}$ is $\bigvee$-hard for P.*

To see this we use the observation at the end of the last section that the formula $\tau(\text{Gad}_f)_b$ for $b = (b^1, \ldots, b^t) \in \{0,1\}^{n+1}$ is a t-size disjunction, $t = k^a + 1$, of τ-formulas for CV_{k,k^a} and b^i, $i \leq t$. Substitute there for the ℓ gadget atoms corresponding to v the bits of the code, say e, of C_k.

EF can prove in p-size (as they are translations of universal formulas provable in PV_1) formulas expressing the equality between two circuit outputs

$$D(e, u) = C_k(u)$$

where $D(v, u)$ is some canonical circuit computing $CV_{k,k^a}(v, u)$ on the particular input lengths. Because we assume that P is a strong proof system we can use these p-size EF-proofs and transform (using, in particular, item 3 of Theorem 2.1) any proof of the original disjunction for Gad_f into a polynomially longer P-proof of a disjunction of $\tau(g)$-formulas. That is a contradiction with the hypothesis that g is $\bigvee$-hard for P.

Claim 2: *Gad_{sq} is $\bigvee$-hard for P.*

Observe that the generator Gad_f from Claim 1 is computed in time $O(k^{2a}) \leq n^{2-\delta}$, for some $\delta > 0$, and hence encoded by $\leq k^2$ bits. We can now repeat the construction of Claim 1 but using Gad_f instead of g (and Gad_{sq} in place of Gad_f).

q.e.d.

Note that applying Lemma 4.2.4 we can further extend the stretch to $n + n^{1-\delta}$, any $\delta > 0$, if needed.

6.3 Failure of PHP and ideal NW-designs

Gadget generators (and Gad_{sq} in particular) are hard for many if not all proof systems for which super-polynomial lower bounds were shown, cf.[57], [60, Chpts.29-30] and [65]. We will now discuss one specific gadget from [57] that leads to a generator hard for AC^0-Frege systems. It was one of the motivations for the generator proposed for the working conjecture 3.2.2 in [51] (see the end of Section 5.1) and subsequently for a specific gadget generator in [60, Sec.30.3] whose definition we give bellow.

A **PHP-gadget** is a $(k+1) \times k$ $0-1$ matrix A represented by atoms v_{ij}. We interpreted A as a graph of a function $h : [k] \to [k+1]$ and use it to stretch by one bit each block u^s of the input to a block w^s of the output. This will work for a proof system P unless we can rule out in P that h is a bijection (i.e. unless we can prove ontoPHP in P).

The bits of the output are defined (keeping in mind our interpretation of A) by 2-DNF formulas:

$$w_i^s \ := \ \bigvee_{j \in [k]} v_{ij} \wedge u_j^s \ .$$

The following statement was originally proved in [57] (cf. [60, Thm.29.5.2]) for the hardness but the proof also shows without much change the $\bigvee$-hardness. Although it is a result about a weak proof system we spell the proof out explicitly as it motivates Theorem 6.5.1.

Theorem 6.3.1 ([57])

The gadget generator based on the PHP-gadget is exponentially $\bigvee$-hard for AC^0-Frege systems.

Proof:

Let g be the gadget generator with the PHP gadget and consider a disjuction of τ-formulas

$$\bigvee_r \tau(g)_{b^r}$$

as in the definition of the $\bigvee$-hardness, where each $b^r \in \{0,1\}^{n+1}$ is an $(\ell+1)$-tuple of $b^{r,s} \in \{0,1\}^{k+1}$.

Now substitute in it for all gadgets (i.e. for all gadgets for all r) common gadget atoms v. Recall that we write f_v for the gadget function with gadget

v fixed. Hence after the substitution the disjuction becomes

$$\bigvee_{r,s} \tau(f_v)_{b^{r,s}} \ . \tag{6.3.1}$$

It suffices to show that this disjunction requires exponential size AC^0-Frege proofs.

This is done by reducing it to the well-known lower bound for the onto PHP$_k$ formulas in the system, cf. [2, 76, 93] or [65, Chpt.15]. The idea is that we can use v to define the inverse map to f_v. Assuming that v violates the onto PHP, i.e. it is the graph of a bijection between $[k]$ and $[k+1]$, map f_v is a bijection too and the formula

$$u^s_j \ := \bigvee_{i \in [k+1]} v_{ij} \wedge w^s_i \ . \tag{6.3.2}$$

defines its inverse function.

Formally: substituting in each disjunct in (6.3.1) for the input atoms $u^{r,s}_i$ the formulas as (6.3.2) with w^s replaced by $b^{r,s}$ the τ-formula will express that

$$f_v(f^{(-1)}(b^{r,s})) \neq b^{r,s}$$

which implies (by short constant depth Frege proofs) the onto PHP$_k$ formula.

That is a contradiction with the stated lower bound for PHP$_k$.

q.e.d.

Based on Theorem 6.2.1 we adopt as our specific goal to show that generator Gad$_{sq}$ satisfies the working conjecture 3.2.2. However, to be able to work with it we need more specific gadgets than just general circuits of sub-quadratic size. This is supported by the experience with lengths-of-proofs lower bounds for weaker proof systems. There it is always instrumental to have hard examples with some clear combinatorial structure.

In order to study the hardness of Gad$_{sq}$ we thus pick gadgets (i.e. sub-quadratic circuits) of a particular form. The generators using them are thus substitution instances of Gad$_{sq}$. The gadgets try to emulate PHP-gadgets. Saying this dually, we try to look at PHP-gadgets as on ideal NW-designs, namely as on $(0,1)$-designs. Of course, no such designs exists in reality if $m > n$ but we may simply try sparse matrices instead. This leads to the concepts described next.

The generators were defined in [65, pp.431-2] and denoted $\mathrm{nw}_{k,c}$ there. Their gadgets are the small and sparse NW-generators discussed at the end of Section 5.1. Because of its importance (for us) we give now a formal stand-alone definition. We shall use the symbol $\mathrm{nw}_{k,c}$ here for the gadget and symbol Gad_{nw} for the generator using this gadget.

Definition 6.3.2 (NW-like gadgets)

Given $1 \leq k$ *and* $1 \leq c \leq \log k$ *the gadget* $nw_{k,c}$ *is given by the following data:*

- $k+1$ *sets* $J_1, \dots, J_{k+1} \subseteq [k]$*, each of size* c*,*
- 2^c *bits defining the truth-table of a Boolean function* h *with* c *inputs.*

Given gadget $v = nw_{k,c}$ *and* $u \in \{0,1\}^k$ *the gadget-function* f *computes* $w := f(v,u) \in \{0,1\}^{k+1}$ *with the* i*-th bit* $w_i := h(u(J_i))$*.*

Finally, $Gad_{nw} := Gad_f$ *for this* f*.*

Note that the gadget is given by $\leq (k+1)(\log k)c + 2^c \leq O(k(\log k)^2) \leq k^2$ bits so Gad_{nw} is indeed an instance of Gad_{sq}.

Also note that Gad_{nw} is computed by an AC^0-formula and that the following statement is a corollary of (the proof of) Theorem 6.3.1 (originally it was deduced in [60] for the hardness).

Theorem 6.3.3 ([60])

Gad_{nw} for any $1 \leq c \leq \log k$ is exponentially $\bigvee$-hard for AC^0-Frege systems.

6.4 Consistency versus existence

A potential advantage of the $\mathcal{NP} \cap co\mathcal{NP}$ generator from Razborov's conjecture 5.3.1 is that there are non-deterministic witnesses for values of f and that could possibly help in devising a lower bound proof.

Let us a point out an advantage the gadget generator (and the $\bigvee$-hardness) seems to have. To express this we take the viewpoint of model theory as explained in Section 3.5. There we have a non-standard finite string $b \in \{0,1\}^m$ not in the range of the generator and we want to extend the model by adding $a \in \{0,1\}^n$ such that $\mathrm{Gad}_f(a) = b$ in the extension.

If we look at Gad_f just as on a p-time function then it is like adding a solution to a fixed equation $\mathrm{Gad}_f(x) = b$, fixed meaning that it is in the ground model already. But we can also look at it as a system of equations for f_v:

$$\bigwedge_{i\in[\ell+1]} f_v(u^i) = b^i$$

where $b = (b^1, \ldots, b^{\ell+1})$. A potential advantage of this view is that now we do not have f_v given in advance (i.e. in the ground model) as we can also add to the model a new gadget $v := c$. That is, it suffices to show that it is consistent to have a gadget for which the system has a solution.

We shall study in Chapter 7 a particular construction of extensions of the ground model.

6.5 A conditional hardness for uniform proofs

To make a better sense of the previous section (and to justify presenting a result about a weak proof system in Section 6.3) we now prove a conditional statement that a generalization of the gadget generator is hard for all proof systems but w.r.t. *uniform* proofs and τ-formulas.

The hardness hypothesis concerns the following $\mathcal{NP}$ search problem denoted $\mathcal{J}_c$. It is motivated by the principle $\mathrm{dWPHP}_1(f, g)$ (cf. (2.1.1)) and it was defined in [36] with the name WPHPWIT. We use a different name as the parameters are somewhat different (and the name is shorter). The problem is defined as follows:

- *valid inputs*: 3-tuples $(1^{(k)}, D, C)$ where
  - D is a size $\leq k^c$ circuit with k inputs x and $k+1$ outputs y,
  - C is a size $\leq k^c$ circuit with $k+1$ inputs and k outputs,
- *solutions*: any $y \in \{0,1\}^{k+1}$ such that $D(C(y)) \neq y$.

The hardness hypothesis we shall use is the following one.

Hypothesis (J):

There exists a constant $c \geq 1$ such that the search problem $\mathcal{J}_c$ cannot be solved by a p-time function.

At least half of the strings in $\{0,1\}^{k+1}$ are solutions and hence the hypothesis of a universal derandomization [33] implies that for any $c \geq 1$ there is a PRNG with the seed $O(\log k)$ such that at least one string in its range is a solution, and this contradicts (J). A similar situation is discussed in detail in Section 8.4. However, popular as it is, the universal derandomization hypothesis is only a hypothesis and it cannot harm to see what could hold if it is actually false.

Theorem 6.5.1

The hypothesis (J) implies that the following holds for the gadget generator g based on the gadget function CV_{k,k^d}, some constant $d \geq 1$:

- *There are no strong proof system P and p-time functions Π, B such that for infinitely many $n \geq 1$ it holds that*

  $$\Pi(1^{(n)}) : P \vdash \tau(g)_b \ ,$$

  where $b := B(1^{(n)})$.

Proof:

Let $c \geq 1$ be the constant from (J). The gadget generator will take as gadgets size $\leq k^c$ circuits D with k inputs and $k+1$ outputs; the gadget size is thus $\ell := 10ck^c(\log k) \leq k^{c+1}$ for $k >> 1$, and the gadget function is the circuit value function $CV_{k,k^{c+1}}$. To ease on the notation denote the generator simply g, so $g_n : \{0,1\}^n \rightarrow \{0,1\}^m$, with n, m determined by k, ℓ as in Definition 6.1.1.

We shall use the model-theoretic criterion for hardness given in Theorem 3.5.1. Assume for the sake of a contradiction that the conclusion of the theorem does not hold for some P, B and Π.

As we aim at an arbitrary strong proof system P we take $T := T_{\mathrm{PV}}$. Take a non-standard model of true arithmetic $\mathbf{M}$. By the overspill there is a non-standard n such that $\pi := \Pi(1^{(n)})$ is a P-proof of $b := B(1^{(n)})$. Let $\mathbf{M}'$ be the substructure of the corresponding small canonical model $\mathbf{M}_n$ generated by $1^{(n)}$; it contains strings b and π. It is still a model of T as that is a universal theory. Note that the model is generated also from $1^{(k)}$ is it determines n in the prescribed way.

Take theory T' in the language of T_{PV} augmented by two new constants C, D and axiomatized by T, the atomic diagram of $\mathbf{M}'$ two axioms:

$$\forall y \in \{0,1\}^m \ D(C(y)) = y$$

and

$$(1^{(k)}), D, C) \text{ is a valid input to } \mathcal{J}_c \; .$$

Claim: *T' is consistent.*

If T' were inconsistent then Herbrand's theorem would give us a p-time function with parameters from $\mathbf{M}'$ that solves the search problem $\mathcal{J}_c$. All parameters can be themselves generated by p-time functions from $1^{(n)}$ and hence from the input to the problem. That contradicts (J) in $\mathbf{M}$.

Let $\mathbf{M}^*$ be a model of T'. To see that $b = w^1 \dots w^{\ell+1}$ is in the range of g in this model we just need to find an element $a := vu^1 \dots u^{\ell+1} \in \{0,1\}^n$ such that $g_n(a) = b$. That is done entirely analogously as in the proof of Theorem 6.3.1: substitute circuit D for the gadget, $v := D$, and use circuit C to compute the map inverse to that computed by D; that is: $u^i := C(w^i)$ for all $1 \leq i \leq \ell + 1$.

q.e.d.

Let us remark that the use of model theory is certainly not needed. However, in our view it illustrates well an approach that could work in more complicated situations.

Further note that the argument can be straightforwardly extended to show that g is **uniformly pseudo-surjective** for all strong proof systems P by which we mean that that there are no p-time functions Π, S that would compute a P-proof of a disjunction (3.3.1) where strings B^i are computed by S in the sense of (2.2.4). Just continue to use circuit C to find preimages for all B^i (this can be done by one p-time algorithm). This in turn implies by Theorem 2.2.3 that $S^1_2(\mathrm{PV})$ does not prove dWPHP(g) and hence also the negative answer to the dWPHP problem 2.0.1. However, this uses (J) and Corollary 2.1.3(2) implies immediately that (J) solves the conservativity problem 1.0.1 (and hence also the dWPHP problem) in the negative. Pity that (J) is not considered plausible.

Chapter 7

The case of ER

This chapter is devoted to the study of a possible way how to prove that a generator is hard for Extended Frege system EF, equivalently for Extended resolution ER. We shall use the formalism of ER as it has the most rudimentary definition of all proof systems that are p-equivalent to EF (see Section 7.1) and some literature we want to quote uses ER.

ER is a pivotal proof system. In the partitioning of proof systems into four levels in [65, Chpt.22] it separates the bottom two levels, *Algorithmic* and *Combinatorial*, from the top two ones, *Logical* and *Mathematical*, sitting at the bottom of the Logical level.

If one succeeded in proving that ER is not p-bounded it would not imply - at least it is unknown to imply (i.e. we do not know if ER is optimal proof system, cf. [70] or [65, Chpt.21]) - that $\mathcal{NP} \neq co\mathcal{NP}$. But it would be close: any super-polynomial lower bound for the length-of-proofs function $\mathbf{s}_{ER}$ (i.e. for any formulas) implies that $\mathcal{NP} \neq co\mathcal{NP}$ is consistent with theory $S^1_2(\mathrm{PV})$ (which contains PV_1). The reader can find details in [45], [65] or [47].

The qualification *close* seems to be honest not only because $S^1_2(\mathrm{PV})$ contains a significant part of computational complexity theory around $\mathcal{P}$ and $\mathcal{NP}$ but also because of the following scenario. Assume that actually some algorithm M solves SAT in p-time and thus $\mathcal{P} = \mathcal{NP}$, and that you can prove the soundness of M (meaning that if M finds no satisfying assignment then none exists) using induction on $\mathcal{NP}$-predicates but not on $\mathcal{P}$-predicates. Theory $S^1_2(\mathrm{PV})$ proves induction for $\mathcal{P}$ predicates but not for $\mathcal{NP}$ predicates (unless log-space equals to p-time with $\mathcal{NP}$ oracles, cf. [43]). This means that while the classes $\mathcal{P}$ and $\mathcal{NP}$ equal, the concepts of of deterministic and non-deterministic p-time computations is not equivalent from logical perspective, i.e. one cannot replace the latter by the former in proofs. Establishing the consistency of $\mathcal{P} \neq \mathcal{NP} \neq co\mathcal{NP}$ with $S^1_2(\mathrm{PV})$ would thus amount to a form of a logical separation of $\mathcal{P}$, $\mathcal{NP}$ and $co\mathcal{NP}$.

In a more down-to-Earth mood one can view the task to show that some generator is hard for ER as a common consequence (conditional in the last case) of all three conjectures mentioned so far: the working conjecture 3.2.2, the pseudo-surjectivity conjecture 3.3.3 and Razborov's conjecture 5.3.1. Of course, the target is an unconditional result but proving the hardness for ER under a hypothesis of a computational nature that is deemed to be plausible would be, in my view, a significant advance (cf. [55] for a related discussion). The method we shall discuss in Section 7.4 aim at that, cf. the introduction to [60].

7.1 Background on ER and $\mathbf{s}_{ER}$

The underlying Frege system F in the statements below is supposed to use the DeMorgan language $0, 1, \neg, \vee, \wedge$ and have modus ponens among its inference rules. This assumption simplifies the formulation of some statements.

Proof system ER formulated in [103] is p-equivalent not only to Extended Frege EF (by [19]) but to a number of other proof systems. Those of a logical nature examples are SF (Frege system with the substitution rule going back to [24]) by [22, 70], Circuit Frege system CF (cf. Section 3.3) or fragment G^*_1 of the quantified propositional calculus G of [71] (cf. [45, L.4.6.3] or [65, Thm.4.1.3]). A more exotic example is one of implicit proof systems of $[R, R^*]$ (cf. [53] or [65, Sec.7.3] for definition and [104] or [65, L.7.3.4] for proofs of the p-equivalence with ER).

The length-of-proofs function $\mathbf{s}_{EF}$ (which is polynomially related to $\mathbf{s}_{ER}$ by the p-simulation of [19]) is also related to some other proof complexity measures. In particular,

$$\mathbf{k}_{EF}(\alpha) \leq \mathbf{k}_F(\alpha) \leq \mathbf{s}_{EF}(\alpha) \leq O(\mathbf{k}_F(\alpha) + |\alpha|)$$

and

$$\mathbf{k}_F(\alpha) \leq \ell_F(\alpha) \leq O(\mathbf{k}_F(\alpha) + |\alpha|) \ .$$

here $\mathbf{k}_P(\alpha)$ is the minimal number of steps in a P-proof of α while $\ell_P(\alpha)$ is the minimal number of different formulas that need to appear as subformulas in a P-proof of α. The number of steps is perhaps the most natural complexity measure from a proof-theoretical point of view while the number of different formulas is the measure to which many lower bounds proofs actually apply. These inequalities can be found in [19, 46] as well as in [45, 65] (an overview of proof complexity measures is, in particular, in [65, Sec.2.5]).

For our purpose are of interest various characterizations of lower bounds for function $\mathbf{s}_{ER}$, i.e. various frameworks for proving lower bounds for $\mathbf{s}_{ER}$ that are complete in the sense that they can be used, in principle, to prove super-polynomial lower bounds, assuming these are valid. Let us mention a few to illustrate the wider picture.

Extension of models of $\mathbf{PV}_1$.

This was outlined in Section 3.5, another brief overview is in [65, Sec.20.1], more detailed in [72] and in [45].

Forcing expansions of models of V^1_1.

This is a variant of an unpublished construction of A.Wilkie. While the characterization in the previous item holds for any strong proof system this construction was tailored to EF. See [46] or [45, Sec.9.4] for details.

Note that [100, 101] studied a construction of Boolean-valued models of bounded arithmetic aiming at separations of complexity classes; see also overview in [77].

Prover-Liar game.

This is based on a theorem of [44] that an F-proof can be put into a tree-like balanced form without much increase in size or number of steps (cf. also [65, Sec.2.2]). In particular, an F-proof with k steps can be transformed into a tree-like proof with the underlying proof tree having the height $O(\log k)$.

In the game (defined in [13]) Prover P asks Liar L about truth-values of formulas. They start with a formula α: P wants to force L to admit that α is true. L can answer in any way she wants. The game stops with P winning iff

- either L says that α is true, or
- L says that 0 is true or 1 is false, or
- L's answers violate the truth-table of one of the connectives $\neg, \vee, \wedge$.

If P happen to have a tree-like F-proof π^* of formula α and π^* has the height h then he has a winning strategy that beats every L in $\leq O(h)$ rounds.

Namely, P asks about the last formula, i.e. about α. He either wins thanks to the first item above or L claims α is false. L then asks about the premises of the inference. Either L admits that one of them is false or she gets into contradiction with the last item. In this way can P navigate through π^* to an instance of an axiom scheme of F, and asking about the values of formulas substituted in the scheme forces L into a contradiction.

This implies that constructing a strategy for L that survives at least t rounds against any P yields a lower bound $2^{\Omega(t)}$ on the number of steps in any F-proof of α and hence, by one of the inequalities mentioned above, some lower bound for $\mathbf{s}_{EF}(\alpha)$ too. In fact, the opposite in equality is true too: minimal number of rounds P needs in the worst case is proportional to the logarithm of $\mathbf{k}_F(\alpha)$, cf. [13] or [65, L.2.2.3].

A reduction between $\mathcal{NP}$-search problems.

This approach is based on a form of a propositional witnessing theorem and is from [63] (cf. [9] for a related work).

Assume you have a Boolean circuit C with no inputs (other than $0, 1$) and of size s. It is a straight line program how to compute a sequence of s constants. Having variables y_i for the subcircuits the circuit is defined by the set of clauses Def_C from the beginning of Section 3.1. It is obviously satisfiable and hence non-refutable. In particular, if π were a purported R-refutation of Def_C there must be some syntactic error in it. The search problem we are interested, having a rather non-descriptive name $\Gamma(0, s, k)$ in [63], is essentially the problem above except that C and π are not fixed in advance but are inputs to the problem. In particular, $\Gamma(0, s, k)$ is a set of clauses in atoms that describe a potential circuit C of size $\leq s$ (i.e. describe clauses in Def_C) and a potential R-refutation of Def_C having $\leq k$ steps. The definition in [63, Sec.1] is fairly technical and we shall not repeat it here but just note that $\Gamma(0, s, k)$ has size $O(k^5)$ for $k \geq 3s$, contains clauses of width $\leq 3 + 3 \log k$ and is unsatisfiable.

The use of $\Gamma(0, s, k)$ is the following. Assume you have another unsatisfiable set of clauses Δ in n variables disjoint from those of $\Gamma(0, s, k)$ and all clauses of Δ having the width $\leq w$. One can consider a **clause reduction** of Δ to $\Gamma(0, s, k)$: a substitution σ of clauses of literals of Δ for variables of $\Gamma(0, s, k)$ such that the substitution instance of a clause of $\Gamma(0, s, k)$ is either logically valid or contains a clause of Δ. The width of the substitution σ is the maximal size of a clause it uses.

Then it holds:

- If δ has an ER-refutation with k' steps then for some $k \leq O(nk')$ and $s \leq k/3$ there is a clause reduction σ of Δ to $\Gamma(0, s, k)$ hawing width $\max(3, w)$.

[63, Thm.2.1] formulates this as a proof-theoretic reduction (each clause of $\sigma(\Gamma(0, s, k))$ has a short proof from Δ) but can be also stated as a reduction between two oracle $\mathcal{NP}$-search problems, oracle giving an assignment to variables of Δ and the task being to find a false clause. The above is formulated as a criterion for lower bounds (the non-existence of a reduction implies a lower bound for ER) but it can be given as a characterization of $\mathbf{s}_{ER}$, formulating it in the form demanding that the reduction is provable. The details of this approach are quite technical and I refer the interested reader to [63].

Boolean valuations.

The notion of *partial Boolean valuations* defined in [46] does not use non-standard models as the first two approaches but can be seen as a finitary version of forcing (see also [45, Sec.13.3] for some discussion). Below we use the same notation as in [46, 45].

For a set Γ is DeMorgan formulas we say that τ is **F-provable within Γ** iff there is an F-proof π o τ such that all formulas that appear as subformulas in π are in Γ. Note that the minimal cardinality of such Γ is precisely $\ell_F(\tau)$.

A partial Boolean algebra $\mathbf{B}(0, 1, \neg, \vee\wedge)$ is a structure where the operations may be only partil function but whenever an identity axiomatizing the variety of Boolean algebras has both sides defined they must be equal. For axiomatization take any standard one, see [45, Def.13.3.1] for one.

A **partial Boolean valuation** of Γ is a map

$$\nu \ : \ \Gamma \to \mathbf{B}$$

such that constants $0, 1$ get mapped to $0, 1$ of $\mathbf{B}$, and

- $\nu(\neg\alpha) = \neg\nu(\alpha)$, if both sides are defined,
- $\nu(\alpha \vee \beta) = \nu(\alpha) \vee \nu(\beta)$, if both sides are defined, and analogously for $\wedge$.

We shall state the underlying theorem for this method exactly as the approach we propose in the next section can be see as an infinitary version of it.

Theorem 7.1.1 ([46])

For any tautology τ *let* n_τ *be the maximal number* n *such that for every set* Δ *of at most* n *formulas and containing* τ *there is a partial Boolean valuation* $\nu : \Delta \to \mathbf{B}$ *such that* $\nu(\tau) \neq 1_{\mathbf{B}}$*.*

Then:

$$n_\tau \leq O(\ell_F(\tau)) \quad \text{and} \quad \ell_F(\tau) \leq n_\tau^{O(1)} \ .$$

An example of constructions of partial Boolean valuations of large sets of constant depth formulas giving to the PHP formula value different from $1_{\mathbf{B}}$ is in [45, Sec.13.3].

7.2 Expansion of pseudo-finite structures

Bounded arithmetic can be formulated in two different set-ups, one-sorted and two-sorted. The one-sorted set-up is the one of PV_1, T_{PV} or $S^1_2(\mathrm{PV})$: elements of structures are numbers (that represent binary strings) and there are relations and functions (infinitely many of them when language of PV is used) on numbers. In the two-sorted set-up you separate numbers (now representing lengths of strings or position of bits in strings) and bounded sets (that represent by their characteristic functions binary strings). These set-ups are fundamentally equivalent but may be useful in different situations. In particular, the two-sorted set-up allows to ignore that strings ought to be closed under some functions. A gentle introduction to this issue is in [65, Chpt.9], more details are in [45] (however, the reader does not need to know this in order to follow the next).

The models of bounded arithmetics PV_1 or $S^1_2(\mathrm{PV})$ we discussed earlier in the connection to a model-theoretic approach to lengths-of-proofs lower bounds are one-sorted in the sense above. They can be replaced by pseudo-finite structures (which are two-sorted). We recall this framework and then give a novel criterion for ER lower bounds using it. The framework is discussed in some detail in [65, Sec.20.2] and in great detail in [64]. Let us note that [83] used this framework to equivalently reformulate various conjectures about mutual relations of basic complexity classes as statements about model-theoretic properties of pseudo-finite structures (see [64] for other examples and references) and [1, 2] used the framework to a great success for inventing a proof of AC^0 lower bound for parity or proving lower bound for AC^0-Frege proofs of the pigeonhole principle tautologies (this is also described in [65, Sec.20.2]).

The structures we shall be interested in look as follows. Let $\mathbf{M}$ be arbitrary non-standard model of true arithmetic (in the language of PA for definiteness). Let L be a finite first-order language disjoint from the language of $\mathbf{M}$, to avoid a confusion.

We shall consider non-standard finite L-structures that have as their universe some $[n]$, for $n \in \mathbf{M}$ a non-standard element. We shall denote such an L-structure $\mathbf{A}_W$ where W is an interpretation of L on $[n]$ that is definable in $\mathbf{M}$. Note that $\mathbf{A}_W$ is coded by $\leq n^k$ bits, some standard k, so it is coded by an element of $\mathbf{M}$ that is bounded above by 2^{n^k}.

These structures are main examples of **pseudo-finite structures**: infinite structures satisfying the L-theory of all finite L-structures. Useful equivalent definitions are the following two conditions:

- *an infinite L-structure that is elementary equivalent to a non-standard finite L-structure $\mathbf{A}_W$ definable in a non-standard model of true arithmetic* $\mathbf{M}$,
- *an infinite L-structure such that every L-sentence true in it is also true in a finite L-structure.*

The general form of a problem of expansions of pseudo-finite structures related to problems of computational and proof complexity is as follows. Let $L' \supseteq L$ be a finite extension of L and let T' be a first-order L'-theory. Recall that expansion means to interpret symbols not in the original language L over *the same* universe: no new elements are added. The problem then is:

- *Given an L-structure $\mathbf{A}_W$ find its L'-expansion $\mathbf{B}$ such that $\mathbf{B} \models T'$.*

Let us remark, informally, that the existence of such an expansion is related to which T'-proofs are definable over $\mathbf{A}_W$ (in a precise technical sense) and for first-order T' this relates to propositional translations. For some problems it is of interest to have T' a Π^1_1-theory, cf. [64], and [3, 4] even treats arbitrary r.e. theories (sufficiently strong and consistent T') and characterizes the existence of expansions of an end-extension of $\mathbf{A}_W$ (cf. [25] for a more conceptual proof).

For our purposes we want to code by functions and relations in $\mathbf{A}_W$ formulas and circuits. If we have a relation

$$H \subseteq [2] \times [n]^a \times [n]^b \ ,$$

$a, b \geq 1$ standard, we can interpret it is a CNF formula α_H whose atoms p_i are indexed by $i \in [n]^a$, which has $\leq n^b$ clauses D_j indexed by $j \in [n]^b$ and such that atom p_i occurs positively (resp. negatively) in clause D_j iff $H(1, i, j)$ holds (resp. $H(2, i, j)$ holds). On the other hand, any DNF formula with polynomially many (in n) atoms and clauses can be so represented.

We will use here circuits with unbounded fan-in $\bigvee$ and $\bigwedge$ To represent such a circuit with input variables x_i, $i \in [n]^a$, and with $\leq n^c$ nodes y_u indexed by $u \in [n]^c$ we consider a relation

$$C_e \ \subseteq \ [n]^c \times [n]^c$$

determining the underlying graph of the circuits, with an edge from node y to node y' iff y is one of inputs to y', together with mappings

$$C_i \ : \ [n]^c \rightarrow [n]^a \dot{\cup} [2]$$

that labels nodes with in-degree 0 by inputs variables or by on of the two constants $0, 1$, and

$$C_g \ : \ [n]^c \rightarrow [3]$$

that labels gates (nodes with non-zero in-degree) by one of the three connectives $\neg, \vee$ or $\wedge$.

We shall assume that $c \geq a$ and that the relation C_e and maps C_i, C_g are encoded jointly in one relation $C \subseteq [n]^{3c}$ in some canonical way.

The final object we need to represent is a sequence of nodes of C of length $\leq n^d$. A function

$$S \ : \ [n]^d \rightarrow [n]^c$$

represents sequences $y_{u_1}, \ldots, u_{u_t}$ where $t = n^d$, ordered set $\{1, \ldots, t\}$ is identified with lexico-graphically ordered $[n]^d$ and $u_v := S(v)$ for $v \in [n]^d$.

Let us pause and dispose of two technicalities. First, given a relation H we only know its arity $1 + a + b$ but we do not know what a, b are. This can be treated by taking $a = b$ and relations H of odd arity only. Analogously remove the same problem for C and S. Second, first-order functions have one value and not a tuple of values. However, S can be represented by c single-valued d-ary functions computing the individual coordinates of C.

To summarize let us use symbol L_{ER} for any language which has symbols:

- a relation symbol H and functions symbols C, S (for some parameters a, b, c, d as above),

- a relation symbol $\leq$ interpreted in W by the ordering of $\mathbf{M}$,
- constants 1 and n interpreted in W by 1 and n of $\mathbf{M}$.

Note that the syntactic forms of H and S guarantee that they represent a DNF formula and a sequence (of indices from $[n]^c$) resp., but not all relations C represent a valid definition of a circuit. Let T_{ER} bye an L_{ER}-theory axiomatized by:

1. $x \leq y$ is a linear ordering with 1 and n being the minimum and maximum, resp.,
2. *C is a circuit*:
    - if (j, j') is an edge in C_e then $j < j'$ in the lexico-graphic ordering,
    - all nodes j that get assigned by C_g connective $\neg$ have in-degree 1.

Language L' of the expansions we shall consider extends L_{ER} by a function symbol E for a Boolean assignment to variables x_is and y_js. As these are represented by $[n]^a$ and $[n]^c$, resp., we have:

$$E \ : \ [n]^a \ \dot\cup \ [n]^c \to \{0,1\}$$

where $\dot\cup$ denotes the disjoint union. A technicality we shall put aside is that there is no 0 in $[n]$ and that E ought to be represented by two functions E_x and E_y defined on $[n]^a$ and $[n]^c$, respectively.

We will want that expansions satisfy the following L'-theory T':

1. the assignment E violates formula H:

   $$\forall i, j, \ (H(1,i,j) \to E(i) = 0) \ \wedge \ (H(2,i,j) \to E(i) = 1)$$

2. E respects all instructions of C (we will skip the long but simple formula expressing this),
3. the image of S in E satisfies induction:

   $$E(S(\overline{1})) = 0 \vee E(S(\overline{n})) = 1 \vee (\exists u, u', suc(u,u') \wedge E(u) = 1 \wedge E(u') = 0)$$

   where $suc(u, u')$ formalizes that u' is the successor of u in the lexicographic ordering and $\overline{1}$ and $\overline{n}$ are its minimal and maximal elements, r espectively.

Now we are ready to state our criterion.

Theorem 7.2.1

Let $H' \subseteq TAUT$ be a set of DNF formulas. Then the following three statements are equivalent:

1. *Set H' is hard for ER.*

2. *There exists a non-standard model $\mathbf{M}$ of true arithmetic such that every pseudo-finite L_{ER}-structure $\mathbf{A}_W \in \mathbf{M}$, $\mathbf{A}_W = ([n], 0, 1, \leq, H, C, S)$, satisfying*

    - $\mathbf{A}_W \models T_{ER}$,
    - $\mathbf{M} \models \alpha_H \in H'$,

    has an L'-expansion satisfying theory T'.

3. *Statement 2 for all non-standard models $\mathbf{M}$ of true arithmetic.*

Note that the second statement does not say that the expansion is in $\mathbf{M}$ (in fact, it cannot be).

Proof:

Condition 2 is trivially implied by 3 so we need to show that 2 implies 1 and 1 implies 3.

Condition 2 implies 1.

We shall assume that condition 1 fails, i.e. that H' is nto hard for ER, and we shall show that in any nonstandard model $\mathbf{M}$ of true arithmetic there is $\mathbf{A}_W \models T_{ER}$ such that $\mathbf{M} \models \alpha_H \in H'$ but $\mathbf{A}_W$ has no expansion $\mathbf{B}$ satisfying T'.

The assumption mean that for som $k \in \mathbf{N}$ there are arbitrarily large $\beta \in H'$ with $\mathbf{s}_{ER}(\beta) \leq |\beta|^k$. By overspill in $\mathbf{M}$ there are a formula $\beta \in H'$ of non-standard length $n = |\beta|$ and its ER-proof π of size $|\pi| \leq n^k$. Construct (in $\mathbf{M}$) from β, π an L_{ER}-structure $\mathbf{A}_W$ as follows:

1. Let $H \subseteq [2] \times [n] \times [n]$ be a relation coding β. Hence $\mathbf{M} \models \alpha_H = \beta \in H'$.

2. String π is an ER-refutation of the CNF $\neg\beta$ and assume its steps are clauses $D_1, \dots, D_t$ (where $D_i = \emptyset$). Assume further that $y_1, \dots, y_e$ are all extension variables introduced in π and that their definitions specify circuit C_0 whose inputs are variables x of β.

   We now extend C_0 to a bigger circuit C which will have unbounded fan-in (C_0 has fan-in ≤ 2) as follows:

   (a) For each D_j introduce instructions

   $$z_j \ := \ \bigvee_{\ell \in D_j} \ell$$

   where ℓ stands for literals, and

   (b) further introduce instructions:

   $$w_j \ := \ \bigwedge_{r \leq j} z_r \ .$$

   Note that C has $e + 2t \leq 3n$ instructions and its inputs are variables of β, say $x_1, \dots, x_n$.

3. For sequence S take $(w_1, \dots, w_t)$.

The L_{ER}-structure $\mathbf{A}_W$ is $([n], H, C, S)$.

We want to show that $\mathbf{A}_W$ has no expansion $\mathbf{B}$ satisfying T'. Assume for the sake of contradiction that a map E can be added so thatT' is satisfied. Because $D_t = \emptyset$ we have $E(w_t) = 0$. On the other hand, D_1 is either a clause of $\neg\beta$ or an extension axiom; in both case T' implies that $E(w_1) = 1$. Using the S-induction axiom of T' there is some $r < t$ such that

$$E(w_r) = 1 \ \wedge \ E(w_{r+1}) = 0 \ .$$

Now we calculate using only the properties that E evaluates C correctly (we use $\vdash$ as an abbreviation for one equation being implied by one or more in this sense):

- $E(w_r) = 1, E(w_{r+1}) = 0 \vdash E(z_{r+1}) = 0$

- if D_{r+1} was deduced in π using D_u, D_v, $u, v \leq r$, then
  $$E(w_r) = 1 \vdash E(z_u) = 1 \wedge E(z_v) = 1$$
  and also
  $$E(z_u) = 1 \wedge E(z_v) = 1 \vdash E(z_{r+1}) = 1$$
- but we also have
  $$E(w_r) = 1 \wedge E(z_{r+1}) = 1 \vdash E(w_{r+1}) = 1$$

which is a contradiction.

Condition 1 implies 3.

Assume H' is hard for ER, $\mathbf{M}$ is an arbitrary non-standard model of true arithmetic and $\mathbf{A}_W \in \mathbf{M}$ is an L_{ER}-structure satisfying T_{ER} and $\mathbf{M} \models \alpha_H \in H'$.

Let $m := |\alpha_H|$ and take the small canonical model $\mathbf{M}_m \subseteq_e \mathbf{M}$ of theory PV_1 defined in Section 3.5. Its universe is a cut

$$\{u \mid |u| \leq m^k, \text{ some standard } k \}$$

and hence $\alpha_H \in \mathbf{M}_m$. The interpretation of the language of PV is inherited from $\mathbf{M}$.

By the hypothesis that H' is hard for ER we have that α_H has no ER-proof in $\mathbf{M}_m$. Hence by Theorem 3.5.1 the model has an extension $\mathbf{M}'$ to a model of PV_1 in which α_H is falsified by some truth assignment $e \in \mathbf{M}'$ to its atoms.

The evaluation e can be in $\mathbf{M}'$ extended to a unique evaluation of circuit C of $\mathbf{A}_W$ (as PV_1 holds there). Use this evaluation to define map E: it gives the same values to all variables as does e. Because $S \in \mathbf{M}_m \subseteq \mathbf{M}'$ and PV_1 proves open induction, the S-induction axiom of theory t' is satisfied too.

q.e.d.

7.3 A Boolean-valued twist

The fact that model $\mathbf{B}$ in the previous section is supposed to be an expansion of $\mathbf{A}_W$ is used only to guarantee that the L_{ER}-reduct of $\mathbf{B}$ is elementarily equivalent to $\mathbf{A}_W$ (as the two structures are even equal). However, this property is the only one needed to assure that condition 2 implies 1: we need to know that H, C, S of $\mathbf{A}_W$ still obey T_{ER} in the bigger structure. Hence we could set-up the construction as follows:

- first find elementary extension $\mathbf{A}'$ of $\mathbf{A}_W$,
- then expand $\mathbf{A}'$ to $\mathbf{B} \models T'$.

Hence $\mathbf{B}$ is an expansion of an elementary extension of $\mathbf{A}_W$.

We need to generalize this further by allowing both $\mathbf{A}'$ and $\mathbf{B}$ be Boolean-valued structures. Such a structure is defined as usual first-order structure with the truth value of sentences A with parameters determined bottom-up from truth values of atomic sentences but now these atomic sentences have truth values from some complete Boolean algebra $\mathcal{B}$. The truth-value $[\![A]\!] \in \mathcal{B}$ commutes with the Boolean connectives and quantifiers are treated using the equations

$$[\![\exists x A(x)]\!] := \bigvee_{a \in \mathbf{A}} [\![A(a)]\!] \quad \text{and} \quad [\![\forall x A(x)]\!] := \bigwedge_{a \in \mathbf{A}} [\![A(a)]\!] \ .$$

It is well-known that these structures respect first-order logic. In particular, all logically valid sentences get the maximal value $1_{\mathcal{B}}$ (it is convenient to call such sentences **valid** in the Boolean-valued case too) and if B logically follows from $A_1, \dots, A_k$ then

$$\bigwedge_{i \le k} [\![A_i]\!] \ \le \ [\![B]\!]$$

where $\le$ is the canonical partial ordering of $\mathcal{B}$.

We shall say that a Boolean-valued structure $\mathbf{A}'$ is an **elementary extension** of an ordinary first-order structure $\mathbf{A}$ (both with the same language), $\mathbf{A} \preceq \mathbf{A}'$ in notation, iff for all sentences A with parameters from $\mathbf{A}$ it holds:

$$\mathbf{A} \models A \ \Rightarrow \ [\![A]\!] = 1_{\mathcal{B}} \ .$$

With all this we aim at the following statement that will be useful in the next section.

Theorem 7.3.1

Let $\mathbf{M}$ *be a non-standard model of true arithmetic and* $H' \subseteq TAUT$ *a set of DNF formulas. Assume that for any* $\mathbf{A}_W \in \mathbf{M}$ *satisfying*

$$\mathbf{M} \models [\mathbf{A}_W \models T_{ER} \wedge \alpha_H \in H']$$

the following two conditions hold:

- *(A) There is a Boolean-valued* L_{ER}*-structure* $\mathbf{K}$ *such that* $\mathbf{A}_W \preceq \mathbf{K}$*,*
- *(B)* $\mathbf{K}$ *has a Boolean-valued expansion* $\mathbf{B}$ *by map* E *such that all axioms of* T' *have the truth-value* $1_{\mathcal{B}}$*.*

The H' *is hard for ER.*

Proof:

The proof is analogous to the proof why condition 2 implies condition 1 in Theorem 7.2.1. There we needed to use that T_{ER} is still true in (the L_{ER}-reduct of) $\mathbf{B}$ which was trivially true (as the reduct was simply $\mathbf{A}_W$). Here use instead that $\mathbf{A}_W \preceq \mathbf{K}$.

q.e.d.

7.4 Random variables

In this section we recall the method of forcing with random variables from [60] and use it to define a fairly general class of Boolean-valued structures that aim to play the role of structures $\mathbf{K}$ and $\mathbf{B}$ in the previous section. We outline the method precisely but informally and rather swiftly; the interested reader ought to consult [60, Chpt.1] or at least [65, Sec.20.4] for the method set-up (the notation is same as the one used in these references).

We equip the standard model $\mathbf{N}$ by a canonical interpretation of language L_{all} having a name for every relation and every function on $\mathbf{N}$ (this is for a technical convenience). For our non-standard model $\mathbf{M}$ we take any $\aleph_1$-saturated model of true arithmetic in L_{all}.

Let $n \in \mathbf{M}$ be a fixed non-standard element and let L_n be the language consisting of all relations in L_{all} and all functions in L_{all} that map $[n]$ into itself. In particular, all constants for elements of $[n]$ are in L_n as well as all Skolem functions for all formulas on the L_n-structure on $[n]$. Note also that $L_n \supseteq L_{ER}$.

The structure to play the role of $\mathbf{K}$ from the previous section, to be denoted $K(F)$, is determined by:

- A **sample space** Ω which is any infinite set such that $\Omega \in \mathbf{M}$. Elements $\omega \in \Omega$ are **samples**.
- A family $F \subseteq \mathbf{M}$ of partial functions
$$\alpha :\subseteq \Omega \to [n]$$
such that $\alpha \in \mathbf{M}$ and that satisfy:
$$\frac{|\Omega \setminus dom(\alpha)|}{|\Omega|} \text{ is infinitesimal } .$$

Infinitesimal means smaller than $1/t$ for some non-standard t. Note that we do not require that the family F itself is definable in $\mathbf{M}$. The notation $K(F)$ reflects only F as it determines Ω.

The universe of a Boolean-valued L_n-structure $K(F)$ is F. All function symbols of L_n are interpreted quite naturally by composing them with elements from F. For example, for $+$ (truncated at n) $(\alpha+\beta)(\omega) = \alpha(\omega)+\beta(\omega)$ and it is required that this function $\alpha + \beta$ is also in F: the terminology is that F is **L_n-closed**.

Any atomic L_n-sentence A with parameters from F is assigned a subset $\langle\langle A\rangle\rangle \subseteq \Omega$: the set of all $\omega \in \Omega$ such that all parameters from F in A are defined on ω, and A with parameters evaluated at ω is true in the L_n-structure on $[n]$.

The complete Boolean algebra $\mathcal{B}$ we need is the quotient of the Boolean algebra of $\mathbf{M}$-definable subsets of Ω by the ideal of sets of an infinitesimal counting measure, cf. [60, Sec.1.2]. The truth-value $[\![A]\!]$ is the image of $\langle\langle A\rangle\rangle$ in $\mathcal{B}$ in this quotient.

This completes the definition of the Boolean-valued structure $K(F)$ once we specify family F.

To expand $K(F)$ by a k-ary function means to define a function
$$\Theta : F^k \to F$$
that has the following property: for all $\alpha_1, \dots, \alpha_k, \beta_1, \dots, \beta_k \in F$
$$[\![\bigwedge_i \alpha_i = \beta_i]\!] \ \le \ [\![\Theta(\alpha_1, \dots, \alpha_k) = \Theta(\beta_1, \dots, \beta_k)]\!] \ . \qquad (7.4.1)$$
This property is needed to assure that the equality axioms are valid in the expansion.

7.5 Tree models

We are going to describe now a fairly broad class of Boolean valued structures constructed from families of random variables of a particular form. Similar structures turned out to be quite useful in other contexts of proof complexity and bounded arithmetic, cf. [60].

Assume we have $\mathbf{A}_W$, an L_{ER}-structure with a non-standard universe $[n]$ as in the previous section. We may think of $\mathbf{A}_W$ also as a structure in the bigger language L_n defined there. To define a family $F \subseteq \mathbf{M}$ of random variables we shall use the following data $\mathcal{D} \in \mathbf{M}$ consisting of objects (sets and functions) that are elements of $\mathbf{M}$ and hence finite or non-standard finite:

- an infinite set Ω os samples (as before),
- a non-empty set Q of **questions**,
- a non-empty set R of **replies**,
- a partial **reply function** $r :\subseteq \Omega \times Q \to R$.

Given $\mathcal{D}$, the family $\mathcal{T} \subseteq \mathbf{M}$ of (Q, R)**-trees** consists of all labeled trees $T \in \mathbf{M}$ such that:

- T is $|R|$-ary and has the depth at most $(\log n)^k$, for some standard $k \in \mathbf{N}$,
- inner nodes are labeled by elements of Q,
- the $|R|$ edges outgoing from an inner node are labelled by all elements of R,
- leaves are labeled by any elements on $[n]$.

Any $T \in \mathcal{T}$ defines naturally a partial function

$$\alpha_T \ :\subseteq \Omega \to [n]$$

in the following way: given $\omega \in \Omega$ travel in T from the root to a leaf, leaving a node labelled by $q \in Q$ by the edge labelled by $r(\omega, q)$. If you reach a leaf the value $\alpha_T(\omega)$ is the label of that leaf; otherwise $\alpha_T(\omega)$ is undefined. We shall denote by the symbol $\alpha_T(\omega) \uparrow$ the fact that the function is undefined at the sample.

The data $\mathcal{D}$ define family $F_{\mathcal{D}}$ consisting of all partial functions α_T, for all $T \in \mathcal{T}$, *assuming* that the following **Key condition** is satisfied:

- *For every* $\alpha \in F_{\mathcal{D}}$*:*

$$Prob_\omega[\alpha(\omega)\uparrow] \ \ \textit{is infinitesimal}\ ,$$

i.e. $Prob_\omega[\alpha(\omega)\uparrow] \leq 1/\ell$ *for all standard* $\ell \in \mathbf{N}$.

If the Key condition is not met then $\mathcal{D}$ defines no family of random variables.

The lemmas formulated in the rest of the section are variants of statements from [60]; to keep the presentation self-contained we outline proofs briefly.

Lemma 7.5.1 ([60, L. 1.4.2 and 5.5.1])

For every $\mathcal{D}$ *satisfying the key condition it holds:*

$$\mathbf{A}_W \preceq K(F_{\mathcal{D}})\ .$$

Proof:

First note that the definition of the truth-values $[\![\ldots]\!]$ immediately implies

Claim: *Every universal* L_n*-sentence true in* $\mathbf{A}_W$ *is valid in* $K(F_{\mathcal{D}})$.

The next observation is that for any existential L_n-formula $\exists y B(x_1, \ldots, x_k, y)$ (B open) L_n contains a function symbol $f(x_1, \ldots, x_k)$ fo a Skolem function for the formula, i.e. satisfying in $\mathbf{A}_W$ the corresponding Skolem axiom:

$$\forall x_1, \ldots, x_k, y,\ B(x_1, \ldots, x_k, y) \rightarrow B(x_1, \ldots, x_k, f(x_1, \ldots, x_k))\ .$$

By Claim this is valid in $K(F_{\mathcal{D}})$. Because every L_n sentence is equivalent modulo these Skolem axioms to a universal (actually to a quantifier-free) sentence we get the lemma.

q.e.d.

Our task is to expand $K(F_{\mathcal{D}})$ by a function Θ that will interpret function symbol E of L' (i.e. it will assign to variables of H and C values 0 or 1) such that the theory T' from Section 7.2 is satisfied.

We shall assume that Θ is defined in the following way. To ease on the notation let Var denote the set of all variables x_i of H (inputs to C) and all variables y_u, instructions of C (they were indexed by $[n]^a \dot{\cup} [n]^c$ previously). Map Θ is determined by a sequence $\hat{\beta} \in \mathbf{M}$:

$$\hat{\beta}\ :=\ (\beta_v)_{v \in Var}$$

with $\beta_v \in F$ computed by trees T_v, all $v \in Var$. Such Θ is interpreted as a function from F to F as follows:

- *Given* $\alpha_T \in F$ *define tree* S *by:*
  - *append to every leaf in* T *labelled by* $v \in Var$ *tree* T_v*,*
  - *to other leafs append nothing.*

Then we define $\Theta(\alpha_T) := \alpha_S$.

Lemma 7.5.2
For all $\alpha_T \in F$*,* $\Theta(\alpha_T) \in F$ *as well. The equality axioms (7.4.1) are valid in* $(K(F_{\mathcal{D}}), \Theta)$*.*

The following statement shows that we do not need to worry about the third axiom of the theory T' (the S-induction).

Lemma 7.5.3 ([60, L.8.3.2])
For any $\mathcal{D}$ *satisfying the Key condition the S-axiom of* T' *is valid in* $(K(F_{\mathcal{D}}), \Theta)$*, i.e. its truth-value is* $1_{\mathcal{B}}$*.*

Proof:
Sequence S in $\mathbf{A}_W$ is a sequence of $\leq n^d$ nodes of circuit C:

$$y_{u_1}, \ldots, y_{u_s}, \ s \leq n^d \ .$$

Each $\Theta(y_{u_j})$ is computed by a tree T_{u_j} that computes the corresponding element of $\hat{\beta}$. We define tree S as follows:

1. Start with tree T_{u_1}: at leaves labeled by 1 go to item 2, and at leaves labeled by 0 change the label to $i = 1$.

2. To leaves of T_{u_1} labeled by 1 append tree T_{u_s}. At leaves of these appended trees labeled by 1 change the label to $i = s$, and at the leaves labeled by 0 go to item 3.

3. At the leaves referred here from item 2 simulate binary search, using trees T_{u_j} to compute values of $y_{y_{s/2}}$, etc. until an r is found such that T_{u_r} computes while $T_{u_{r+1}}$ computes 0. Then label the leaf by $i = r$.

4. Finally change all labels of the form $i = t$ to t.

Note that the depth of the tree is $\leq d(\log n)d'$, where d' is the maximal depth of a tree T_v, $v \in Var$. Hence $S \in \mathcal{T}$.

Claim: *The element* $\alpha_S \in F$ *witnesses that the* S*-induction axiom is valid in* $(K(F_{\mathcal{D}}), \Theta)$.

q.e.d.

To define Θ we only need to use trees T_v, $v \in Var$. One may be tempted to simplify the data $\mathcal{D}$ in the following way, taking in a sense the minimal data $\mathcal{D}_{min}$ needed, defined as follows:

1. for $\omega \in \Omega$ define
$$\omega^* \ := \ \{\beta_v(\omega)\}_{v \in Var} \in \{0, 1, *\}^{Var}$$
where $*$ represents the case when $\beta_v(\omega)$ is undefined
2. new sample space $\Omega^* := \{\omega^* \mid \omega \in \Omega\}$
3. questions $Q^* := \{v =? \mid v \in Var\}$
4. replies $R^* := \{0, 1\}$
5. reply function $r^* :\subseteq Q^* \times \Omega^* \rightarrow R^*$ by
$$r^*(v =?, \omega^*) \ := \ \omega^*_v$$

and we take Θ^* computed by the depth 0 trees asking $v =?$, for $v \in Var$.

The new family $F_{\mathcal{D}_{min}}$ is smaller and hence there is less opportunity to find a 3-term of α_H that is satisfied by Θ^* (i.e. showing that the first axiom of T' does not hold). On the other hand, if Θ^* claims that a clause is true this smaller family may miss a witness to it, i.e. a true literal in the clause.

Using the economic $\mathcal{D}_{min}$ data may also not be best for analyzing properties of the corresponding family of random variables. As an example may serve PHP-trees where natural trees ask where a pigeon i goes rather than just ask if pigeon i goes to hole j, cf. [60, Chpts.20 an 21].

Chapter 8

Consistency results

In his chapter we prove several consistency results with theory T_{PV}. All are proved by applying the witnessing Theorem 2.2.2, part (a), for Σ^b_2-consequences of T_{PV} and then showing under a hypothesis (some more plausible than other) that the formula in question cannot be witnessed by an S-T computation in a constant number of rounds.

It is in my view important for further development to prove similar consistency results for theory $S^1_2(\mathrm{PV})$. An analogous approach would be to show that dWPHP cannot by witnessed by S-T computations with a polynomial number of rounds. However, there the situation is more complex and the assumption that it is provable in PV_1 that the Student succeeds may be crucial; we discuss this in Section 8.4.

Let us remark that in the relativized case, when we have a function symbol for a generator g but not its definition, a number of unconditional consistency results are known. For example, we cannot witness by a p-time oracle machine with a polynomial advice with an $\mathcal{NP}^R$ oracle, where R is the graph of g, that g is not a bijection between $[a]$ and $[2a]$. Or even with oracle access to functions g, f we cannot witness by a PLS problem defined by a p-time machine with oracle access to f, g that $\mathrm{dWPHP}_1(f, g)$ of Section 2.2 holds. The interested reader can find these and other related results in [45, Secs.11.2-3] and in references given there.

8.1 S-T computations and provability

Consider a Σ_2^b-formula as in (2.2.1):

$$\forall x \exists y (|y| \leq |x|^c) \forall z (|z| \leq |x|^d),\ A(x,y,z)$$

Our main (but not only) example is when A is

$$y < 2x \rightarrow (z < x \rightarrow g(z) \neq y)$$

and (2.2.1) expresses dWPHP(g).

To simplify the notation we shall incorporate bounds to y and z into the formula A, meaning that A has the form

$$A \ :=\ |y| \leq |x|^c \wedge (|z| \leq |x|^d \rightarrow A_0(x,y,z)$$

and the above formula is written simply as

$$\forall x \exists y \forall z,\ A(x,y,z)\ . \tag{8.1.1}$$

The existence of S-T computations witnessing (8.1.1) for A open formula can be characterized analogously to Theorem 2.2.3 by provability in a theory.

Theorem 8.1.1

For formula (8.1.1) with A open the following holds.

1. *The following three conditions are equivalent:*
    - *(a) (8.1.1) can be witnessed by S-T computations in a constant number of rounds,*
    - *(b) T_{PV} proves the formula*

      $$\bigvee_{1 \leq i \leq k} A(x, S_i(x, z_1, \ldots, z_{i-1}), z_i)$$

      where S_i are p-time functions computing the i-th move of S (same as in (2.2.3).),
    - *(c) (8.1.1) is provable in theory T_{PV}.*
2. *The following three conditions are equivalent:*

(a) (8.1.1) can be witnessed by S-T computations in polynomial number of rounds,

(b) T_{PV} proves the formula

$$z \in [x]^{|x|^k} \to \exists i < |x|^k,\ A(x, M(x, z|i), z_i)$$

where M is the machine computing S that always finds a witness in $\leq n^k$ rounds, and $z|i$ has the same meaning as in (2.2.4),

(c) (8.1.1) is provable in theory $T_{PV} + S^1_2(PV)$.

Proof:

Conditions (c) imply conditions (a) by the witnessing theorems alluded to in Section 2.2 (cf. [75] and [42]).

Conditions (a) imply conditions (b) as the formulas in (b) express that (8.1.1) can be witnessed in k or n^k rounds, respectively, and are universal (the formula in 2(b) can be put - provably in PV_1 - into a universal form by using a p-time algorithm finding i). Hence they are axioms of T_{PV}.

That condition 1(b) implies 1(c) is obvious. To get from 2(b) to 2(c) we need to use $S^1_2(\mathrm{PV})$ that proves that there is a maximal $i < |x|^k$ for which there is an evaluation of $z|(i-1)$ such that

$$\forall j < i,\ \neg A(x, M(x, z|j), z_j)\ .$$

Then $M(x, z|(i-1)$ witnesses formula (8.1.1).

q.e.d.

The theorem means that showing the unprovability of a formula of the form (8.1.1) in theories T_{PV} or $T_{\mathrm{PV}} + S^1_2(\mathrm{PV})$ is equivalent to a purely computational complexity task to show that the formula cannot be witnessed by S-T computations with constant or polynomial number of rounds, respectively. As the later assertion (for any A) implies, in particular, that $\mathcal{P} \neq \mathcal{NP}$ all such results have to use some hypothesis. We return to this topic in Section 10.2.

8.2 The dWPHP for the truth-table function

We note first that the truth-table function can be, under an assumption, witnessed by a p-time function.

Lemma 8.2.1

Assume that there exists $L \in \mathcal{E}$ such that $L \notin_{i.o.} Size(2^{\epsilon k})$, for some $\epsilon > 0$. Then the formula $dWPHP(\mathbf{tt}_{s,k})$ with $s = 2^{\epsilon k}$ can be witnessed by a p-time function and hence the theory $T_{\rm PV}$ proves the dWPHP for this function.

Proof:

Assume $L \in \mathcal{E}$ and that $L_k := L \cap \{0,1\}^k$ has no size $2^{\epsilon k}$ circuits for $k >> 1$. The characteristic function of L_k can be, however, constructed from $1^{(2^k)}$ by some p-time function f.

The second part of the statement follows as the fact that f witnesses the dWPHP can be stated as true universal formula, an axiom of $T_{\rm PV}$.

q.e.d.

In this section we give a proof of a conditional result from [66] that theory $T_{\rm PV}$ does not prove the dWPHP for the truth-table function. The hypothesis the statement uses has to contradict the hypothesis of Lemma 8.2.1. In particular, we use the following computational complexity hypothesis.

Hypothesis (H):

There exists a constant $d \geq 1$ such that every language in $\mathcal{P}$ can be decided by circuits of size $O(n^d)$: $\mathcal{P} \subseteq Size(n^d)$.

The hypothesis with $d = 1$ is often attributed to Kolmogorov although it seems he raised it as a possibility and did not present it as a conjecture; see the discussion in [38, Sec.20.2].

As it appears, most experts do not consider it plausible but this should not stop us to investigate it. In particular, there are no technical results that would speak against (H). It implies that that $\mathcal{P} \neq \mathcal{NP}$ as there are languages in the polynomial-time hierarchy that have no size $O(n^d)$ circuits, cf. [39], and moreover implies this by an *upper* bound rather than by a *lower* bound as does the conventional circuit complexity theory. Already this feature ought to attract attention to (H) as we seem to be much better at proving upper bounds while proving lower bounds is in a long term a fiasco.

What some researchers may find less attractive is that (H) also implies that $\mathcal{E} \subseteq \mathrm{Size}(2^{o(n)})$ (use padding), giving a blow to foundations of universal derandomization. Hypothesis (H) is, in my view, good for proof complexity: via [52, Thm.2.1] it implies that either $\mathcal{NP} \neq co\mathcal{NP}$ or that there is no p-optimal proof system.

No we are ready to formulate the result.

Theorem 8.2.2 ([66, Thm.1])

Assume (H). Then for every $0 < \epsilon < 1$ *and* $s = s(k) := 2^{\epsilon k}$ *the formula* $dWPHP(\mathbf{tt}_{s,k})$ *cannot be witnessed by an S-T computation with a constant number of rounds.*

In particular, the theory T_{PV} *does not prove* $dWPHP(\mathbf{tt}_{s,k})$*, i.e. the sentence:*

$$\forall 1^{(m)}(m = 2^k > 1)\exists y \in \{0,1\}^m \forall x \in \{0,1\}^n,\ \mathbf{tt}_{s,k}(x) \neq y \tag{8.2.1}$$

(recall where $n := 10s\log s$*).*

Proof:

Assume that $T_{\rm PV}$ proves the formula. By Theorem 2.2.2 the formula can be witnessed by an S-T computation with a constant $t \geq 1$ number of rounds. Assume the t moves of Student are computed by p-time functions

$$S_1(z), S_2(z, w_1), \ldots, S_t(z, w_1, \ldots, w_{t-1})\ . \tag{8.2.2}$$

Take d the constant guaranteed by (H) and $m >> 0$ large enough. Using these define constants δ_i and m_i by:

$$\delta_i := (2d)^{-i}\ ,\ \text{for } i = 0, \ldots, t \ \text{ and } \ m_i := m^{\epsilon\delta_i}\ .$$

Let us see that the Student cannot succeed in the first round already. Define new function $\hat{S}_1$ that has $m_t + k$ variables and on inputs $1^{(m_t)}$ and $i \in \{0,1\}^k$ computes the i-th bit of $S_1(1^m)$ (padding by the string $1^{(m_t)}$ makes the new function p-time).

Let $C_1'(z, i)$ be a circuit (with the same variables as $\hat{S}_1$) that computes $\hat{S}_1$ guaranteed by hypothesis (H). Define a new circuit C_1 by substituting $1^{(m_t)}$ for z in C_1' and leaving just the k variables for bits of i. Note that by the choice of C_1' circuit C_1 has size $O((m_t + k)^d)$ and thus can be encoded by $\leq m_{t-1}$ bits. Further, by its definition, $\mathbf{tt}_{s,k}(C_1) = b_1$ where $b_1 := S_1(1^{(m)})$.

Now extend the argument to show that S does not succeed in the second round either, i.e. that S_2 does not compute a suitable $b_2 := S_2(1^{(m)}, C_1)$. Define function $\hat{S}_2$ that will now take three inputs: string $1^{(m_{t-1})}$, circuit C_1 (substituted for variables w_1) and $i \in \{0,1\}^k$, and computes the i-th bit of $S_2(1^{(m)}, C_1)$.

Applying (H) again we get a circuit C_2' (now having $2m_{t-1} + k$ variables) computing $\hat{S}_2$, and we define C_2 by substituting $1^{(m_{t-1})}$ for z and bits defining

C_1 for w_1 into C_2'. Note that C_2 is left just with the k variables for bits of i and that it can be encoded by $\leq m_{t-2}$ bits and, crucially, $\mathbf{tt}_{s,k}(C_2) = b_2$.

Continuing in this way we show that Student given by the t-tuple (8.2.2) cannot succeed. The final circuit C_t constructed in the process and witnessing that the last candidate solution b_t is also in $rng(\mathbf{tt}_{s,k})$ can be encoded by m_0 bits. Hence all circuits C_i have size at most $m_0 = m^\epsilon = 2^{\epsilon k}$.

q.e.d.

8.3 The dWPHP for the circuit value function

In this section we state a variant of Theorem 8.2.2 from [30] where the truth-table function is replaced by the circuit value function. The impossibility to witness dWPHP for the truth-table function by S-T computation in a constant number of rounds implies that impossibility for the circuit value function but [30] used different hypotheses than [66], replacing the hypothesis (H) by two new hypotheses (I1) and (I2) formulated below.

Hypothesis (I1) uses the notion of **indistinguishability obfuscation** of [8]. An **indistinguishability obfuscator** with security (S, ϵ) is a p-time randomized algorithm $i\mathcal{O}$ that takes as inputs:

- security parameter λ,
- a circuit C,
- a random string r,

and satisfying two conditions:

1. For all λ ad C the output $i\mathcal{O}(1^{(\lambda)}, C)$ of the algorithm is with the probability $\geq 1/|r|$ a circuit computing the same function as C.

2. For any λ and any two circuits C, C' of size at most $\leq \lambda$ that compute the same function, and for any circuit A of size $S(\lambda)$ (acting as an adversary) it holds that:

$$|\mathrm{P}rob[A(i\mathcal{O}(1^{(\lambda)}, C)) = 1] - \mathrm{P}rob[A(i\mathcal{O}(1^{(\lambda)}, C')) = 1]| \ \leq \ \epsilon(\lambda)$$

Algorithm $i\mathcal{O}$ is **JLS-secure** if it is secure for some $S(n) = n^{\omega(1)}$ and $\epsilon(n) < 2^{-n^{\Omega(1)}}$. We refer the reader to [30] for a more detailed introduction to this notion.

The second hypothesis uses the computational complexity class **AM**, the class of languages having a sound and complete Arthur-Merlin protocol, cf. [7]. The class is a probabilistic analog of $\mathcal{NP}$ and it holds that $\mathcal{NP} \subseteq \mathbf{AM} \subseteq \mathcal{NP}/poly$.

Now we are ready to state the two hypotheses the theorem will assume.

(I1) There exists an indistinguishability obfuscator $i\mathcal{O}$ that is JLS-secure.

(I2) TAUT $\notin_{i.o.}$ **AM**.

Theorem 8.3.1 ([30, Thm.21])
Assume hypotheses (I1) and (I2). Then the formula dWPHP(CV) cannot be witnessed by an S-T computation with a constant number of rounds.

In particular, the theory T_{PV} does not prove dWPHP(CV).

The general idea of the proof is not that difficult but its technical implementations is. We explain here the idea and leave it to the interested reader to read the details in [30, Thm.21].

The starting idea of the proof is a construction, assuming that we have a feasible way how to witness the dPWPHP for the circuit-value function, of an $\mathcal{NP}$ algorithm for TAUT. The $i\mathcal{O}$ is used to get a cryptographic construction of *witness encryption* whose breaking would involve solving a task about SAT. They consider circuit $C[\varphi, y](x)$ which outputs y if x satisfies formula φ and a string of zeros otherwise. Then it is analyzed what happens if the function witnessing dWPHP(CV) is applied to this circuit which is, however, crucially first obfuscated by $i\mathcal{O}$; the non-deterministic algorithm accepts only if the witness is y itself.

The analysis is quite technical already but significant further complications come as the witnessing function provided by the KPT theorem is only computed via an S-T computation in a constant number of rounds. This introduces further (besides $i\mathcal{O}$) probabilistic element that leads eventually to the need for hypothesis (I2) instead of just TAUT $\notin \mathcal{NP}$.

IN particular, assume for the sake of contradiction that dWPHP(CV) can be witness by S-T computations in k rounds. Hence, for some $k \geq 1$, there are k p-time functions $S_1(x), S_2(x, z_1), \ldots, S_k(x, z_1, \ldots, z_{k-1})$ computing the moves of the Student such that in one of the rounds S finds a string outside the range of a given circuit C (expanding n bits to $n < m \leq n^{O(1)}$ bits).

The idea is to show that there are a circuit C and strings $a_1, \ldots, a_k \in \{0,1\}^n$ and $b_1, \ldots, b_k \in \{0,1\}^m$ such that

(a) $b_i = b_j \to a_i = a_j$, for $1 \leq i, j \leq k$,

(b) $C(a_i) = b_i = S_i(C, a_1, \ldots, a_{i-1})$, for all $1 \leq i \leq k$.

Having C and the two k-tuples clearly allows to show that the particular strategies $\{S_i\}_{i \leq k}$ do not work.

The hard part of the proof comes in the construction of these objects and here a particular Arthur-Merlin protocol involving $i\mathcal{O}$ is constructed, and analyzed using (I1) and (I2).

Let us remark that condition (a) is, in principle, not needed as the student has to find a solution even if the teacher answers same questions differently each time (but correctly).

To conclude this section let us discuss the hypotheses used in the theorem. Both are considered by experts plausible and this is an advantage over the hypothesis (H) used in Section 8.2 However, the belief in (I1) is based on a heuristic experience in cryptography (it can be deduced from some hardness assumptions accepted as heuristically verified) rather than from some fundamental theoretical assumption. Hypothesis (I2) is fundamental enough but it alone implies $\mathcal{NP} \neq co\mathcal{NP}$ which is what we are aiming at in the first place (at least if you think of the dWPHP problem as a way to get an insight how to prove the hardness of some generator). In particular, if we think of the results giving the unprovability of dWPHP as weaker versions of the hardness of τ-formulas then we would like to see them proved under a plausible hypothesis about deterministic (probabilistic) computations and stay away from making assumptions relating TAUT and $\mathcal{NP}$ (cf. [55] for a discussion). Of course, these remarks are not meant to lessen in any way the fact how ingenious the construction underlying Theorem 8.3.1 is.

8.4 Revisiting the dWPHP problem

The results in Sections 8.2 and 8.3 settle - under computational hypotheses - the weaker version of the dWPHP problem 2.0.1 when $S^1_2(\mathrm{PV})$ is replaced by PV_1. In particular, by Theorem 8.3.1 the hypotheses (I1) and (I2) imply that $T_{\mathrm{PV}} \supseteq \mathrm{PV}_1$ does not prove $\mathrm{dWPHP}(CV)$. This is complemented by Lemma 8.2.1 that (under a hypothesis about circuit complexity of languages in $\mathcal{E}$) T_{PV} does prove $\mathrm{dWPHP}(f)$ for all p-time functions without parameters (the uniform case). These two results are not in a contradiction because Theorem 2.1.4 holds over $S^1_2(\mathrm{PV})$ but not - as these results show - over T_{PV}. This is further complemented by the unprovability result for the truth-table function in Theorem 8.2.2 under a conflicting hypothesis.

Note also that these results (conditionally) settle also the version of the conservativity problem 1.0.1 when T_{PV} is present: $T_{\mathrm{PV}}+S^1_2(\mathrm{PV})+\mathrm{dWPHP}(\Delta^b_1)$ is $\Sigma^b_1(\mathrm{PV})$-conservative over T_{PV} but it is different unless $\mathcal{NP} \subseteq \mathcal{P}/poly$ (the former follows from Lemma 8.2.1 and Theorem 4.3.2 and the latter follows from [75]).

These results say nothing about the original dWPHP problem 2.0.1 and it is our view that making an advance on this problem holds the key to further advances on the two conjectures 3.2.2 and 3.3.3. In fact, the situation is even more interesting because the problem seems to force us to move to propositional logic: witnessing theorems alone cannot be used to answer the problem in the negative (which is what we expect). This is because, under hypotheses, the dWPHP for p-time generators can be actually witnessed by S-T computations with a p-time student in polynomially many rounds. We have observed this already in Lemma 4.2.6 but let us show this under a weaker hypothesis than is used there. First a simple fact.

Lemma 8.4.1

Assume that the dWPHP for $\mathbf{tt}_{s,k}$ *with any* $s = 2^{\Omega(k)}$ *can be witnessed by an S-T computation with a p-time student in polynomially many rounds. Then this is true for all p-time generators.*

Proof:

This follows essentially from the fact that that $S^1_2 + \mathrm{dWPHP}(\Delta^b_1)$ is axiomatized over $S^1_2(\mathrm{PV})$ by $\mathrm{dWPHP}(\mathbf{tt}_{s,k})$, any $s = 2^{\Omega(k)}$ (Theorem 4.3.2).

In some detail: the hypothesis implies that the universal formula analogous to (2.2.4) expressing that some p-time S solves the witnessing task in n^k rounds is true and hence it is an axiom of T_{PV}. Hence $T_{\mathrm{PV}} + S^1_2(\mathrm{PV})$ proves $\mathrm{dWPHP}(\mathbf{tt}_{s,k})$ and by Theorem 4.3.2 it also proves the dWPHP for all p-time generators. Thus, by Theorem 2.2.2 (adding true universal theory does not change witnessing), all $\mathrm{dWPHP}(g)$ are witnessed in the same way.

q.e.d.

The following lemma follows immediately from Lemmas 8.2.1 and 8.4.1.

Lemma 8.4.2
Assume that there exists $L \in \mathcal{E}$ such that $L \notin_{i.o.} Size(2^{\epsilon k})$, for some $\epsilon > 0$. Then the dWPHP for all p-time generators can be witnessed by an S-T computation with a p-time student in polynomially many rounds.

Hence any proof of the unprovability of dWPHP in $S^1_2(\mathrm{PV})$ ought to use in a substantial way that that the universal formula (2.2.4) expressing that a p-time student witnesses dWPHP in polynomially many rounds is provable in PV_1 by Theorem 2.2.3. This is what lead - via propositional translations into EF proofs (Section 2.3)- to the pseudo-surjectivity conjecture 3.3.3. That move to propositional logic ignored the additional information that circuits computing moves of the student are actually uniform (the non-uniform version relates to extensions of models by Theorem 3.5.3). The uniformity may play a significant role; an example is the construction of the hardcore set in [61] for S-T computations related to Statement (S) (cf. the remark at the end of Section 8.5). Recall also that we noted at the end of Section 6.5 that the hypothesis (J) considered there implies the negative solution to the conservativity problem 1.0.1 and hence also to the dWPHP problem 2.0.1 (but (J) is not considered plausible at present).

8.5 One-way permutations and statement (S)

We have discussed in Section 5.4 Statement (S) which essentially formalizes (modulo some additional technical conditions) that Conjecture 5.3.1 applies to all proof systems, and we proved under some hypotheses that it is not true, cf. Theorem 5.4.1.

In this section we use the hypothesis of the existence of strong OWP and show that it is actually consistent with theory T_{PV}, following the argument in [58]. To ease on technicalities we present here only a sample part of the results from [58], and we simplify a bit the conditions posed in (S) on the NW generator, to avoid the need to formulate precisely relations among various parameters.

The conditions we shall require from the NW generator are the following:

(A1) The parameters n, d, ℓ, m satisfy:

$$m(n) = 2^{n^{o(1)}} \ , \ d = \log m \ , \quad \text{and} \quad \ell = n^{1/3} \ .$$

(A2) Function h be a p-time OWP with exponential hardness on average, $B(x)$ is its hard bit, and assume that function f is defined as $f(y) := B(h^{(-1)}(y))$. In particular, $H_f(\ell)$ is exponential, i.e. $2^{\ell^{\Omega(1)}}$.

(A3) Matrices A_n are $m \times n$ and are $(l, \log m)$-designs, and there is a p-time algorithm that from $i \in \{0,1\}^d$ and $1^{(n)}$ computes the i-th row J_i of A_n.

Let us remark that parts of [58] prove the consistency of a statement with finer relations among the parameters and the hardness of h, using the concept of the *approximating hardness* defined there (we shall not present it here).

The consistency is shown, as the previous two sections, via showing that a certain computational task cannot be solved by an S-T computation in a constant number of rounds. The task is related to the formula expressing the dWPHP for $NW_{A,f}$:

$$\exists y \in \{0,1\}^m \forall x \in \{0,1\}^n \exists i \in [m] \ f(x(J_i(A_n))) \neq y_i$$

(with parameters n, m, A_n universally quantified) but it is not the task to witness this formula.

Instead the consistency is deduced via an elementary model-theoretic construction utilizing the fact that the following formula

$$\forall x \in \{0,1\}^n \exists i \in [m] \ f(x(J_i(A_n))) \neq b_i \tag{8.5.1}$$

cannot be witnessed by an S-T computation in a constant number of rounds for infinitely many $n \geq 1$ and $b \in \{0,1\}^m$, with feasible *nonuniform* student S. It is the use of model theory that forces us to consider non-uniform students (i.e. their moves are computed by circuits) rather than just uniform p-time students as earlier.

The relevant computational task is the following one.

Task (T$_b$): *For a fixed* $b \in \{0,1\}^m \setminus rng(NW_{A_n,f})$*,* T_b *is the task to find, given* $a \in \{0,1\}^n$ *some* $i \in [m]$ *such that* $f(a(J_i(A_n))) \neq b_i$*.*

Here a counter-example to $f(a(J_i(A_n))) \neq b_i$ *is a witness to* $f(a(J_i(A_n))) = b_i$ *using the* $\mathcal{NP}$*-definition of* $f(u)$*, i.e. it is* $v := h^{(-1)}(a(J_i(A_n)))$ *such that* $B(v) = b_i$*.*

Now we state the key lower bound. In its proof we follow closely the presentation in [61, Sec.1] to enable the reader to compare it with a more general argument given there and leading eventually to a hardcore set. The original proof in [58, Thm.3.2] gives a more precise statement using the approximate hardness defined there.

Theorem 8.5.1 ([58, Thm.3.2])

Assume that the parameters n, m, d, ℓ*, the matrices* A_n *and the function* f *obey the conditions (A1)-(A3) stated above. Assume also that* circuits $S_1(x), S_2(x, z_1), \ldots, S_c(x, z_1, \ldots, z_{k-1})$ *compute moves of a student that solves task* T_b *in* c *rounds for all* $b \in \{0,1\}^m \setminus rng(NW_{A_n,f})$*.*

Then for $n >> 1$ *large enough the total size of* $S_1, \ldots, S_c$ *must be exponential* $2^{n^{\Omega(1)}}$*.*

Proof:

Assume that the Student found a solution for $x := a \in \{0,1\}^n$ in the k-th round ($k \leq c$), producing candidate solutions $i = (i_1, \ldots, i_k)$ (with i_k being correct). Call the k-tuple i the *trace* of the computation. Teacher's answers are unique and hence the trace determines them as well. A counting argument establishes the following statement.

Claim 1: *There is* $i \in [m]^k$ *for some* $k \leq c$ *that is the trace of computations on at least a fraction of* $\frac{2}{(3m)^k}$ *of all inputs from* $\{0,1\}^n$*.*

Fix one such trace i and use it to define for any pair $u \in \{0,1\}^\ell$ and $v \in \{0,1\}^{n-\ell}$ string $a(u,v) \in \{0,1\}^n$ putting bits of u into the positions from row J_{i_k} and then filling the remaining positions by bits of v. An averaging argument deduces from Claim 1 the following statement.

Claim 2: *There is* $e \in \{0,1\}^{n-\ell}$ *such that there is at least a fraction of* $1/(3m)^k$ *more strings* $u \in \{0,1\}^\ell$ *determining string* $a(u,e)$ *whose trace is exactly* i *than of those* u *which yield* $a(u,e)$ *whose trace properly contains* i*.*

Fix one such $e \in \{0,1\}^{n-\ell}$. The design property that two distinct rows intersect in at most $\log m$ positions implies that there are, for any row $i \neq i_k$, at most m assignments w to bits in row J_i not determined by e. Any such w defines - together with the fixed e - an assignment to variables in J_i and hence a string $z_w \in \{0,1\}^\ell$. Take the set Y_i of all preimages of all these z_w in the permutation h and note that the total size of all strings in Y_i is $m^{O(1)}$.

We can define now an algorithm for computing f that will use as advice the following data:

- Set system $\{J_i\}_{i \in [m]}$,
- string b,
- trace i,
- string e,
- sets $\{Y_i\}_{i \neq i_k}$,
- circuits $S_1, \dots, S_c$ computing the moves of the Student.

The total size of the advice is $s + m^{O(1)}$, where $s := \sum_{j \le c} |S_j|$.

To define the algorithm take $U \subseteq \{0,1\}^\ell$, the set of those u for which the trace of $a(u,e)$ either equals to i or starts with i. Take $b_0 \in \{0,1\}$ that is the majority value of f on $\{0,1\}^\ell \setminus U$.

The algorithm operates as follows. On input $u \in \{0,1\}^\ell$ it simulates Student's moves in the S-T computation on the string $a := a(u,e) \in \{0,1\}^n$.

1. If any of the candidate solutions produced in the j-th round, some $j \le k$, differs from i_j then algorithm halts and outputs b_0.
2. Otherwise (i.e. the trace of the computation follows i), the algorithm uses sets Y_i in order to simulate Teacher's replies (we use that these are unique and can be tested for their correctness). If the computation followed trace i and reached the k-th step then the algorithm outputs bit $1 - b_{i_k}$.

Note that the algorithm outputs the bit b_0 in all cases except when the computation follows the trace i and reaches the k-th step. If the computation of the Student were to actually stop at that point then the value $1 - b_{i_k}$ is indeed the correct value $f(u)$. If the computation were to continue, we do not have a way to deduce what $f(u)$ is. But the influence of this case can be bounded.

Namely, by the choice of e after Claim 2 the former case happens for at least a fraction of $\frac{1}{(3m)^k}$ more of all u than the latter case. Hence b_0 is the correct value for at least half of $u \notin U$ and the algorithm computes f with an advantage over $1/2$ at least $\frac{1}{(3m)^k}$.

Using the hypothesis of exponential hardness of f this implies that s has to be exponential too.

q.e.d.

Before the next statement recall the notion of a large canonical model from Section 3.5: a cut $\mathbf{M}_n^*$ in a model of true arithmetic $\mathbf{M}$ in the language of T_{PV} whose universe is the set of all elements w of $\mathbf{M}$ of length $|w| \leq 2^{n^{o(1)}}$.

Corollary 8.5.2 ([58, Thm.4.1])

Assume the parameters n, m, d, ℓ, the matrices A_n and the function f obey the conditions (A1)-(A3) stated above.

Let $\mathbf{M}$ be a non-standard model of true arithmetic in the language T_{PV}, n its non-standard element and $b \in \{0,1\}^m$ with $m = m(n)$.

Then the large canonical model $\mathbf{M}_n^*$ has a cofinal extension $\mathbf{M}'$ to a model of T_{PV} such that

$$NW_{A,f}(a) = b$$

for some $a \in \mathbf{M}'$.

Proof:

This is proved by using elementary model theory and witnessing Theorem 2.2.2(part 1).

Take $T \supseteq T_{\mathrm{PV}}$ to be the theory in the language of T_{PV} together with names for all elements of $\mathbf{M}_n^*$ that contains also the atomic diagram of $\mathbf{M}_n^*$ as axioms. It suffices to show that T does not prove that $b \notin rng(NW_{A_n,f})$.

Assume for the sake of a contradiction that it does, i.e. T proves formula (8.5.1). Theorem 2.2.2(part 1) can be applied to T as adding true (here true in $\mathbf{M}_n$) universal sentences (here atomic sentences of the diagram) does not change the witnessing argument based on the KPT theorem. It yields a constant number of terms in the language of T that compute moves of a student solving Task T_b in a constant number of rounds. Each term consists of p-time function and constants from the model that act as advice strings. Hence each move of the student is computed by a circuit of size $2^{n^{o(1)}}$ and their total size is thus also bounded above by $2^{n^{o(1)}}$.

That contradicts Theorem 8.5.1.

q.e.d.

We are ready to state and prove the consistency result.

Theorem 8.5.3 ([58, Thm.4.2(3)])

Assume the parameters n, m, d, ℓ*, the matrices* A_n *and the function* f *obey the conditions (A1)-(A3) stated above. Assume also that* B *is an infinite* $\mathcal{NP}$ *set that has infinitely many elements of lengths* $m(n)$*, for* $n \geq 1$*.*

Then it is consistent with theory T_{PV} *that*

$$rng(NW_{A_n,f}) \cap B \ \neq \ \emptyset \ .$$

Proof:

Assume $y \in B$ is defined by $\exists z(|z| \leq |y|^c) B_0(y, z)$ with B_0 open formula (a p-time relation). Assume for the sake of a contradiction that B is disjoint with the range of the generator. Take a non-standard model $\mathbf{M}$ of true arithmetic in the language of T_{PV} and note that B is disjoint with the range of the generator there as well.

By the hypothesis that B has infinitely many elements of the length $m = m(n)$, we can take non-standard n such that there is $b \in \{0,1\}^m \cap B$. Let $\mathbf{M}_n^*$ be the large canonical model determined by n. In particular, $b \in \mathbf{M}_n^*$ and a witness to $b \in B$ is also in $\mathbf{M}_n^*$.

By Corollary 8.5.2 $\mathbf{M}_n^*$ has a cofinal extension $\mathbf{M}'$ to a model of T_{PV} in which

$$NW_{A_n,f}(a) = b$$

for some $a \in \mathbf{M}'$. This proves the theorem.

q.e.d.

Note that the argument works even if the S-T computation runs in $n^{\Omega(1)}$ many rounds for small enough $\delta > 0$ (small w.r.t. $H_f(\ell)$), cf. [58]).

Let us conclude this section with a few remarks. The reader may wonder why we cannot use the model-theoretic criterion in Theorem 3.5.2 and deduce that $NW_{A_n,f}$ is hard for all proof systems. This is discussed in detail in [58, Sec.5] but the culprit is the fact that f is only $\mathcal{NP} \cap co\mathcal{NP}$ and not deterministic p-time. The fact that

$$NW_{A_n,f}(a) = b$$

in the model $\mathbf{M}'$ does not mean that we have a falsifying assignment for the atoms of the corresponding τ-formula. The τ-formula has the form

$$\bigvee_{i \in [m]} \alpha_i(x, z^i)$$

where $\alpha_i(x, z^i)$ formalizes that z^i witnesses that the value of f on $x(J_i)$ is b_i. What we have in the model is that for each i there is c^i such that (a, c^i) satisfies α_i but we do not have there necessarily string $(a, c^1, \ldots, c^m)$ that aggregates all these individual assignments together. To deduce its existence from the existence of individual assignments needs an instance of sharply bounded **collection scheme** (aka axiom of choice) which is (probably) not available in T_{PV} by [20]. It is available in theory $S^1_2(\mathrm{PV})$ but to extend the argument above to that theory requires to prove a lower bound for S-T computations with polynomially many rounds. However, as discussed in Section 8.4, such a lower bound may not be true without extra assumption that the theory PV_1 proves that the student always succeeds. This issue is also linked with the strong fdp property we used in Section 4.2 as is discussed at length in [58, Sec.5].

We also want to remark that a model playing the same role as the one in Corollary 8.5.2 can be constructed via the method of forcing with random variables we discussed in Section 7.4. This is the **local witness model** of [60, Chpt.31] (with a corrected constructions of a hardcore set in [61]). It is this construction that is not ruled out as a possible approach to arranging that that theory $S^1_2(\mathrm{PV})$ holds in the model, as it is desirable by the discussion above.

Note that [58, Sec.6] explains in detail how the whole situation around the NW generator can be specialized to some proof systems weaker than EF; in particular, to those for which we do not have super-polynomial lower bounds yet.

Finally let us point out that the method used in this section found uses in other contexts, cf. [89, 92, 80].

8.6 S-T computations and a gadget generator

In this section we give a construction generalizing in a sense that of Section 6.5. The construction entails a conditional consistency with the theory T_{PV} of the statement that the range of (a variant of) the gadget generator intersects all infinite $\mathcal{NP}$ sets.

In this construction the generator is only a partial function and its graph is an $\mathcal{NP}$ relation. The construction does not imply the conditional hardness of the corresponding τ-formulas for the same reasons as the consistency of

Statement (S) does not imply the hardness of the NW-generator, as it is discussed at the end of Section 8.5 (a missing collection scheme in$T_{\rm PV}$).

We will talk about partial functions defined by non-deterministic circuits: any circuit $D(x, y, z)$ with k variables x, $k+1$ variables y and further variables z such that the following formula

$$\gamma_D \ := \ (D(x, y^1, z^1) \wedge D(x, y^2, z^2) \to y^1 = y^2) \tag{8.6.1}$$

is a tautology determines a partial function

$$h_D \ : \subseteq \{0,1\}^k \to \{0,1\}^{k+1}$$

defined by:

$$h_D(a) = b \ \text{ iff } \ D(a, b, z) \in \text{SAT} \ .$$

Note that the validity of the formula (8.6.1) guarantees that h_D is a partial *function* and not only a partial *multi-function*.

The hardness hypothesis we shall use says that the following search problem cannot be solved by an S-T computation in a constant number of rounds and $\mathcal{P}/poly$ student. The search problem, denoted $\mathcal{K}(c, P)$, is related to the problem $\mathcal{J}(c)$ of Section 6.5 and it is defined as follows:

- *valid inputs*: 4-tuples $(1^{(k)}, D, C, p)$ where
  - $D(x, y, z)$ is a size $\leq k^c$ circuit with k inputs x, $k+1$ outputs y and further inputs z,
  - p is a P-proof of γ_D,
  - C is a size $\leq k^c$ circuit with $k+1$ inputs and k outputs,
- *solutions*: any $y \in \{0,1\}^{k+1}$ such that $h_D(C(y)) \neq y$.

  (This includes the case when $h_D(C(y))$ is undefined.)

The hardness hypothesis we shall use is the following one.

Hypothesis (K)

There exists a constant $c \geq 1$ and a proof system P such that for no constants $d, t \geq 1$ can the search problem $\mathcal{K}(c, P)$ be solved by an S-T computation in t rounds and with student's moves computed by circuits of size $\leq k^d$, for $k >> 1$.

At least half of the strings in $\{0,1\}^{k+1}$ are solutions and hence, for any fixed $c \geq 1$, a counting argument yields a size $k^{O(1)}$ set $Y \subseteq \{0,1\}^{k+1}$ containing a solution for all inputs. However, the student does not seem to have a way how to pick a right one in $O(1)$ rounds. Note that if he had a polynomial number of rounds he could go through all strings in Y one-by-one and use the teacher to find a correct solution.

To formulate the theorem we shall define first a variant of the gadget generator. Let $c \geq 1$ be a constant. The gadget generator will take as gadgets size $\leq k^c$ circuits $D(x, y, z)$ with k inputs and $k+1$ outputs; the gadget size is thus $\ell := 10ck^c(\log k)$.

The gadget function $f : \{0,1\}^\ell \times \{0,1\}^k \to \{0,1\}^{k+1}$ will now be a partial $\mathcal{NP}$-function defined as follows:

$$f(D,u) = v \text{ iff } \left(\exists \pi (|\pi| \leq |\gamma_D|^e)\ \pi : P \vdash \gamma_D\right) \wedge D(u,v,z) \in \text{SAT} .$$

The existence of π guarantees that at most one v is assigned to (D, u).

Call the resulting (generalization of the) gadget generator simply g^c, so $g^c_n : \{0,1\}^n \to \{0,1\}^m$ where $\ell = \ell(k)$ and hence also $n = n(k)$ and $m = m(k)$ depend on $k \geq 1$. Note that it is now a partial function only but $b \notin rng(g^c)$ is still a $co\mathcal{NP}$ property of b and hence the $\tau(g^c)_b$ formulas are well-defined.

Further note that in the language of T_{PV} the statement that $b = b^1 \dots b^t \in rng(g^c_n)$ can be written as

$$\exists x \in \{0,1\}^n (x = Du^1 \dots u^t) \forall i \in [t]\ f(D,u^i) = b^i \tag{8.6.2}$$

where the $\mathcal{NP}$ statement $f(D,u^i) = b^i$ is a bounded existential formula.

Theorem 8.6.1

Assume the hypothesis (K) and let B be an $\mathcal{NP}$-set having infinitely many elements of size $m(k)$, for $k \geq 1$.

Then it is consistent with T_{PV} that there exists $b \in B$ satisfying the formula (8.6.2).

Proof:

Assume that $c \geq 1$ is a constant and P is a proof system guaranteed to exist by (K).

We shall use the model-theoretic criterion given in Theorem 3.5.1. Take non-standard model of true arithmetic $\mathbf{M}$. By the hypothesis about B there are (in the model) non-standard k, $n = n(k)$ and $b \in \{0,1\}^m \cap B$ for $m := m(k)$. Hence b is also in the corresponding small canonical model $\mathbf{M}_k$.

Take theory T' in L' extending L by three constants C, D, p and consisting of T, the atomic diagram of $\mathbf{M}_k$ and of the axioms:

- $(1^{(k)}, D, C, p)$ is a valid input for $\mathcal{K}(c, P)$,
- $\forall y \in \{0,1\}^m,\ h_D(C(y)) = y$.

Claim: T' *is consistent.*

If T' were inconsistent then the KPT theorem would give us an S-T computation running in $t \geq 1$ rounds (t a fixed standard number) and with student's moves computed p-time functions with parameters from $\mathbf{M}_k$, i.e. by size k^d (some standard $d \geq 1$) circuits in $\mathbf{M}$, that will solve the search problem $\mathcal{K}(c, P)$ on all valid inputs. That contradicts (K) in $\mathbf{M}$.

Let $\mathbf{M}'$ be a model of T'. To see that

$$\mathbf{M}' \ \models\ b \in rng(g_n)$$

in the sense of formula (8.6.2) we just need to find $a = a^1 \dots a^t \in \{0,1\}^n$ witnessing the formula. That is done analogously as in the proof of Theorems 6.3.1 or 6.5.1: substitute circuit D for the gadget and use circuit C to compute $a^i := C(b^i)$.

q.e.d.

The missing collection scheme would be used to pull together all witnesses for all $h_D(a^i) = b^i$, $i \leq t$.

8.7 Feasibly infinite $\mathcal{NP}$-sets

One way how to make the Working conjecture 3.2.2 weaker and hence more tractable is to restrict the class of all infinite $\mathcal{NP}$ sets featuring in the conjecture to some natural subclass of $\mathcal{NP}$.

The restriction we shall define poses a condition on how one can witness that a set is infinite. Take a sound theory $T \supseteq \mathrm{PV}_1$ in a language extending that of T_{PV}. Consider the class of all $\mathcal{NP}$ sets A such that the infinitude of A:

$$Inf_A \ :=\ \forall x \exists y (y > x \wedge y \in A)$$

can be proved in T. Here $y \in A$ is defined by a formula in the language of T_{PV} of the form

$$\exists z(|z| \leq |y|^c) A_0(y, z)$$

with $c \geq 1$ a constant and A_0 open (and hence A_0 defines a p-time relation). Note that Inf_A is an $\forall\exists$-sentence.

The condition that a particular T proves Inf_A yields non-trivial information about A. For example, for $T = T_{\mathrm{PV}}$ Herbrand's theorem implies that there is a p-time function f witnessing Inf_A:

$$\forall x(f(x) > x \wedge f(x) \in A) \ .$$

We shall call $\mathcal{NP}$ sets A for which such p-time function f exists **feasibly infinite**. Note that (using Buss's theorem instead of Herbrand's) A is also feasibly infinite if $S^1_2(\mathrm{PV})$ proves Inf_A.

Theorem 8.7.1 ([68, Thm.7.1])

Assume hypothesis (H) from Section 8.2. Then the Working conjecture 3.2.2 holds relative to the class of feasibly infinite $\mathcal{NP}$ sets: there is a p-time generator g whose range intersects every feasibly infinite $\mathcal{NP}$ set.

Proof:

The proof is a special case of the construction in the proof of Theorem 8.2.2 and the generator g is the truth-table function $\mathbf{tt}_{s,k}$ with $s = s(k) := 2^{k/2}$.

Assume an $\mathcal{NP}$ set A is feasibly infinite and that this it is witnessed by a p-time function f. Take the constant $d \geq 1$ from hypothesis (H) and define parameters:

$$m := |f(1^{(n)})|,\ m' := m^{1/(3d)},\ k := \log m$$

where $n >> 0$ is large enough.

Now take a p-time function $\hat{f}$ with $m' + k$ variables and which on inputs $1^{(m')}$ and $i \in \{0,1\}^k$ computes the i-th bit of $f(1^{(n)})$. The hypothesis (H) gives us a circuit $\hat{C}(z, i)$ that computes $\hat{f}$. Use $\hat{C}$ to define another circuit C by substituting $1^{(m')}$ for z in $\hat{C}$. Hence C has only k variables left (for bits of i) and its size is $O((m'+k)^d) < 2^{k/2}$.

By the definition of C we have $\mathbf{tt}_{s,k}(C) = f(1^{(n)})$ and thus

$$rng(\mathbf{tt}_{s,k}) \cap A \neq \emptyset$$

which is what we wanted to show.

q.e.d.

We can use the theorem and formulate a statement about models of theory T_{PV}.

Corollary 8.7.2 ([68, Cor.7.2])

Assume hypothesis (H). Then there exists a model $\mathbf{M}$ *of theory* T_{PV} *in which the Working conjecture 3.2.2 holds in the following sense:*

- *For* $g := \mathbf{tt}_{s,k}$ *with* $s = s(k) := 2^{k/2}$ *and any standard* $\mathcal{NP}$ *set* A *(i.e. defined without parameters from* $\mathbf{M}$*) it holds:*

$$\mathbf{M} \models rng(g) \cap A = \emptyset \ \rightarrow \ \neg Inf_A \ .$$

Proof:

For any $\mathcal{NP}$ set A the statement $rng(g) \cap A = \emptyset$ is a universl sentence. Hence it is true in the standard model $\mathbf{N}$ iff it is true in all models of T_{PV}. The statement will thus follow if we show the consistency of T_{PV} extended by all sentences $\neg Inf_A$, for all $\mathcal{NP}$ sets A such that $rng(g) \cap A = \emptyset$.

If it were not consistent then the compactness theorem implies that for some $\mathcal{NP}$ set A such that $rng(g) \cap A = \emptyset$ theory T_{PV} proves Inf_A. This uses that a finite number of A_i are all disjoint from $rng(g)$ iff their union is.

But then A is feasibly infinite and that contradicts Theorem 8.7.1.

q.e.d.

Further generalization of (and some problems about) the notion of feasibly infinite $\mathcal{NP}$ sets are discussed in Section 10.3.

Chapter 9

Contexts

In this chapter I want to bring to the attention of the reader several topics to which the theory of proof complexity generators turned out to be related by they are not themselves part of the theory. Each of them (except search problems treated in the last section) appears in one paper each (either entirely devoted to it or describing it as a part of a wider investigation) and it is thus easy to study the original text. For this reason the presentation in this chapter will differ from earlier ones in that we shall describe precisely but informally the underlying idea and key points of proofs or constructions involved, as well as the statements, but refer the reader for details to the respective source papers.

A point I wish to stress is that in all cases the relations between the topic and the proof complexity generators theory can be, I think, generalized and improved, and trying to achieve this may possibly be interesting research topics.

9.1 Essential variables

This section is based on [56].

A pseudo-random generator G mapping short strings x to long strings y are used to reduce the number of random bits a feasible probabilistic algorithm uses. In particular, instead of picking random y the algorithm pics random x and uses $y := G(x)$ for random bits.

The idea of the application we shall discuss in this section is that proof complexity generators may be used quite analogously in the context of certifying the unsolvability by feasible proofs. Assume that we have a generator g with stretch $m(n)$ that is hard for a proof system P. Let $\alpha(y)$ be a formula with m atoms. Now assume that

1. $\alpha(g(x))$ has a short P-proof π, but
2. $\alpha(y) \notin \mathrm{TAUT}$.

We can use this situation to prove in P feasibly any $\tau(g)_b$ for $b \in \{0,1\}^m$ falsifying α as follows:

- *Prove* $\neg\alpha(b)$ *and combine this with proof* π *to deduce* $\tau(g)_b$.

Hence if g is indeed hard for P and short π exists then it follows that α is actually a tautology.

The difference between $\alpha(y)$ and $\alpha(g(x))$ is that the latter formula has a smaller (possibly much smaller) number of *essential variables*. There are more variables in $\alpha(g(x))$ than just x, namely variables encoding the computation of g, and their number can be bigger than m, the number of y variables. However, the values of all these extra variables are determined once

- we know values of the variables x,
- we know that $\alpha(g(x))$ is false.

The word *determine* means that if we write $\alpha(g(x))$ as $\beta(x, z)$ where z are the extra variables, then the implication

$$\neg\beta(x,z) \wedge \neg\beta(x,z') \rightarrow z_i \equiv z'_i$$

is true for all extra variables z_i, z'_i and, in fact, it is provable by a linear size R-proof if α is a DNF. We do not need to formally define what a set of essential variables is as we shall talk in the formal statements below about the substitution $y := g(x)$ provided by the generator g.

In [56] we took for P just resolution R as for this system we have unconditionally hard generators (by Theorems 5.2.2 and 4.3.7) which are p-time - this seems important if we talk about SAT algorithms - and have a large stretch. Note that the condition on having a large stretch, i.e. arranging that the number of essential variables is much smaller than the number of all variables, rules out the PHP-gadget generator which is uniform and exponentially hard for AC^0-Frege systems but has a small stretch by Theorem 6.3.1 (here the affirmative answer to Problem 5.2.3 would be useful as it would allow to extend the results below unconditionally to AC^0-Frege systems.

Two interesting sets of parameter choices for which the idea works are the following two. A note of warning: the parameters k and n of [56] are now called n and m in order to conform with our set-up in which we use n for the number of input bits and m for the number of output bits of a generator. The parameter sets are:

(A) $n := m^\delta$, $s := 2^{m^\epsilon}$,

(B) $n = (\log m)^c$, $s := m^{(\log m)^\mu}$,

and the formal statement about them reads as follows.

Theorem 9.1.1 ([56, Thm.2.1])

1. *For any $\delta > 0$ there are parameter $\epsilon > 0$ and a p-time generator g stretching $n := m^\delta$ bits to m bits and such that whenever $\alpha(y)$ is a 3DNF formula with m atoms and $\alpha(g(x))$ has an R-proof of size $\leq s$, s the parameter in (A), then $\alpha(y)$ is a tautology.*
2. *There are constants $c \geq 1, \mu > 0$ and a generator g computable in time $m^{O(1)}$ and stretching $n := (\log m)^c$ bits to m bits and such that whenever $\alpha(y)$ is a 3DNF formula with m atoms and $\alpha(g(x))$ has an R-proof of size $\leq s$, s the parameter in (B), then $\alpha(y)$ is a tautology.*

A natural question is if one can bound the time of a SAT solver in terms of the minimal number of essential variables rather than in terms of the number of all variables (we restrict in this discussion to 3DNF formulas as in the theorem). It follows from part 2 of the theorem that no SAT solver whose computations can be turned efficiently into at most polynomially longer R-proofs (e.g. those based on some for of the DPLL procedure even augmented by clause learning or restarts of the procedure) can run in time subexponential in the number of essential variables. This is because such computation would yield p-size R-proofs when choosing parameters (B) above and $\mathcal{P} = \mathcal{NP}$ would follow (or some randomized version of this if the original SAT algorithm were randomized). For details and related references see [56, Sec.3].

Note that the generator g we referred to above via Theorems 5.2.2 and 4.3.7 is the truth-table function and hence strings not in its range are truth-tables of hard Boolean functions. This allows us to employ the notion of *natural proofs* from [98] and observe that even the mere fact that $A(g(x))$ is a tautology has an interesting consequence. Namely, assuming the existence of strong pseudo-random generators as in [98], it holds:

- *If* $A(g(x)) \in TAUT$ *then there are at most* $2^m/m^{\omega(1)}$ *falsifying truth assignments for* $A(y)$.

Again, see [56, Sec.3] for details.

A problem left open in [56, Sec.3] is whether substitutions like above can speed-up proofs. Putting it informally:

- *Are there DNF formulas* $A(y)$ *and a generator* g *such that* $A(y)$ *require long R-proofs while the substitution instances* $A(g(x))$ *have short R-proofs?*

9.2 The optimality problem

It is an open problem whether there exists an **optimal proof system** : a proof system P such that its length-of-proof function $\mathbf{s}_P$ has at most polynomial slow-down over $\mathbf{s}_Q$, for any proof system Q. cf.[70] or [65, Optimality problem]. It is known that P is not optimal iff there exists a p-time construable sequence α_k of tautologies (i.e. α_k can be constructed by a p-time algorithm from $1^{(k)}$) such that $\{\alpha_k \mid k \geq 1\}$ is hard for P. All first super-polynomial lower bounds for all proof systems for which some such lower bounds are known were proved for such an explicit sequence. However, for strong proof systems the only candidate p-time sequences $\{\alpha_k\}_k$ we have are based on reflection principles and that is not very helpful for lower bounds as the formulas refer to provability about which we are supposed to prove something, cf.[70, 45, 65].

In this section we shall outline some ideas and results from [62] where the problem to construct such a sequence was approached from the computational complexity perspective, utilizing earlier results about the NW generator and about Statement (S) of Section 5.4. Two search problems more general than just finding hard formulas were studied there. These problems are motivated by the hypothetical situations that we can *prove* that $\mathcal{NP} \neq co\mathcal{NP}$ (task Cert) and that we can *prove* that no optimal proof systems exists (task Find). Note the emphasis on *prove*; that is, not only that it is true but that we can prove it.

To motivate Cert assume that we can prove that $\mathcal{NP} \neq co\mathcal{NP}$ is some theory formalizing mathematics, say ZFC. In particular, we can prove for any $c \geq 1$ that for no proof system P can $\mathbf{s}_P(\alpha)$ be bounded above by $|\alpha|^c$. This statement can be formalized by the following sentence in the language of PV_1 (we use the same notation as in [62]):

$$LB_P(c) \ := \ \forall 1^{(k)} \exists \beta,\ |\beta| \geq k \wedge \beta \in \mathrm{TAUT} \wedge \forall \pi (|\pi| \leq |\beta|^c)\ \pi : P \not\vdash \beta \ .$$

Now note that for a strong proof system P (much weaker assumption on P suffices) if we prove $LB_P(c)$ for $c \geq 2$ then the soundness of P follows: having a proof of falsifiable formula allows to prove anything by a linear size proof. However, a simple use of Gödel's incompleteness implies that ZFC cannot prove the soundness of all proof systems. Hence instead of the provability of LB_P formulas we ought to study their provability under the assumption that P is sound, i.e. the provability of the implications

$$Ref_P \rightarrow LB_P(c) \ .$$

To witness this statement means to either find a hard formula or to find an error of P: a falsifiable formula with a P-proof.

To have one problem rather than one for each P we shall replace proof systems by non-deterministic circuits that are supposed to accept exactly $\mathrm{TAUT} \cap \{0,1\}^k$.

Search problem Cert:

Let $D(x, y)$ be a circuit with k variables x (representing a formula) and $\ell := k^c$ variables y (representing a proof). The search task is:

- Input: a size k^{c^2} circuit $D(x, y)$ with k variables x (representing a formula) and $\ell := k^c$ variables y (representing a proof).
- Output:
  - either a size k falsifiable formula α such that $D(\alpha, y)$ is satisfiable,
  - or a size k tautology β such that $D(\beta, y)$ is unsatisfiable.

Note that the output of $Cert(D)$ certifies that $\exists y D(x,y)$ does not define $\text{TAUT} \cap \{0,1\}^k$.

The second search problem, Find, that we shall define is motivated as follows. Assume you can answer the Optimality problem in the negative and, in fact, that you can give a uniform construction of stronger proof systems. In particular, assume that there is an oracle polynomial time machine that for any proof system P, when having an oracle access to P, defines a stronger proof system $Q(P)$ (i.e. $\mathbf{s}_{Q(P)}$ has a super-polynomial speed-up over $\mathbf{s}_P$) such that we can prove that $Q(P)$ is a proof system:

$$Ref_P \to Ref_{Q(P)}$$

and that it is indeed stronger:

$$Ref_P \to \forall 1^{(k)} \forall \pi (|\pi| \leq k^c)\ \pi : P \not\vdash \| Ref_{Q(P)} \|^k \ .$$

(This formalization uses known facts about relations between simulation and provability of reflection principles and we refer the reader to either of [45, 65] for details.) In particular, any strong proof system simulates $Q(P)$ if it can use $\| Ref_{Q(P)} \|^k$ as extra axioms. In the following search task α represents any possible extra axiom.

Search problem Find:

Let $c_1 \geq c_0 \geq 1$.

- Input: $1^{(k)}$ and a size $\leq k^{c_0}$ tautology α.
- Output: any size k tautology β that has no size $\leq k^{c_1}$ proof in proof system $P + \alpha$.

Let us point out that Find can be reduced to Cert for a suitable c depending on C_0, c_1 (cf. the end of [62]).

The main results in [62] were proved using ideas from [58] discussed in Section 8.5 together with a bit wild idea that the NW-generator can be used not only as a source of τ-formulas but it can also serve as a proof system. yet another search problem was considered in [62] as a technical tool to approach Cert and Find. We shall only state results concerning these two problems.

Theorem 9.2.1 ([62, Cor.4.2])

Assume that an exponentially hard one-way permutation exists. Then there is $c \geq 1$ such that no deterministic time $2^{O(k)}$ algorithm solves Cert *on all input lengths $k \geq 1$.*

Theorem 9.2.2 ([62, Cor.6.2])

Assume that an exponentially hard one-way permutation exists and that Statement (S) holds.

Then there is $c \geq 1$ *such that* Cert *is only partially defined for infinitely many lengths* $k \geq 1$*: there are inputs corresponding to* k *for which the problem has no solution.*

Theorem 9.2.3 ([62, Thm.6.3])

Assume that an exponentially hard one-way permutation exists and that Statement (S) holds.

Then for any strong proof system P *there are constants* $c_1 \geq c_0 \geq 1$ *such that* Find *has no solution for infinitely many lengths* $k \geq 1$.

The interested reader will find all details in [62]. Note that the implications of Statement (S) may seem rather contradictory. On one side it implies $\mathcal{NP} \neq co\mathcal{NP}$ by its formulation and on the other hand it implies, in particular, that TAUT $\in_{i.o.} \mathcal{NP}/poly$ (Theorem 5.4.1). This is caused by the double role the NW generator plays in these constructions: a source of hard formulas and a strong proof system. Some readers may be quick to dismiss Statement (S) as obviously not plausible, citing the second consequence as the reason. I think that we know very little about the power of non-uniformity and of non-deterministic circuits in particular, to jump to such a conclusion.

9.3 Structured WPHP

In this section we shall discuss the idea of structured PHP introduced in [49] and studied in the context of proof complexity generators in [54]. The general idea is simple. Imagine that in a model $\mathbf{M}$ of some theory T you have a bijection $h : [N] \rightarrow [M]$ where $N \neq M$. You can use it to transport structure $\mathbf{A}$ with the universe $[N]$ to structure $h(\mathbf{A})$ with the universe $[M]$. For example, if $N = 2^k$ and $M = 3 \cdot N$ and the $\mathbf{A}$ is a vector space over $\mathbf{F}_2$ then it follows that T cannot prove that that the size of a universe of an $\mathbf{F}_2$-vector space cannot be divisible by 3. Or turning the table around, if you can prove in T that some structure cannot have size M while you can define in T one of size N, then you also disprove in T the existence of a bijection h.

A more delicate variant of the idea involves various small size subsets of $\mathbf{A}$; in the example above take a basis X of the vector space. As any T (containing S^1_2, for example) can count sets of logarithmic size and prove that h preserves the size counting (technically: T proves the PHP for logarithmically small sets) then even if M was a power of 2, say $M = 2^{k+1}$, we get a contradictory situation. Namely, $h(\mathbf{A})$ has basis $h(X)$ which is smaller than it ought to be: $|X| = k < \log M$.

We talked above about a bijection for simplicity of the picture but if $N > M$ and h is an injection then we insert $\mathbf{A}$ into a smaller universe $[M]$, and if $N < M$ and h is a surjective map (this is the case of the dWPHP we are most interested in) then h can be used to pull a structure $\mathbf{B}$ with universe $[M]$ back onto a smaller universe $[N]$. We shall now give an example result for the dWPHP case.

In the context of generators we have $N = 2^n$ and $M = 2^m$ and the universes $[N]$ and $[M]$ are identified with $\{0,1\}^n$ and $\{0,1\}^m$, respectively. We consider relational structures on these universes whose relations are defined by p-size (in n or m, resp.) circuits. We shall call such structures **$\mathcal{P}/poly$-structures**.

A **tournament** is a directed graph $G = (V, E)$ with exactly one edge between any two different vertices. A **dominating set** in G is a set X of its vertices such that

$$\forall i \in V \setminus X \exists j \in X,\ (j, i) \in E\ .$$

Every tournament of size 2^m has a dominating set of size m but by a probabilistic argument [23] showed that there are tournaments of that size having no dominating set of size $m/2$. A $\mathcal{P}/poly$-tournament on $\{0,1\}^m$ having no size $m/2$ dominating set was constructed in [96]; we shall use name E_m for a size $m^{O(1)}$ circuit defining the edge relation for such a tournament.

Now assume we have a generator $g : \{0,1\}^n \to \{0,1\}^m$ with stretch $m = 2n$ and we use it to define a $\mathcal{P}/poly$-tournament $H = (\{0,1\}^n, D_n)$ by

$$D_n(u, v)\ :=\ E_m(g(u), g(v))\ ,\ u, v \in \{0,1\}^n\ .$$

Tournament H has a dominating set X of size n and this fact can be expresses by a formula we shall denote just $\sigma_{n,X}$, leaving the references to E_m and g implicit:

$$\sigma_{n,X}\ :=\ \bigvee_{u \in X} x = u \vee D_n(u, x)$$

where x is an n-tuple of atoms.

Now the observation is that $g(X)$ has size $\leq n$ and hence cannot be dominating in $G = (\{0,1\}^m, E_m)$. If $b \in \{0,1\}^m$ is a vertex that is not dominated by any element of $g(X)$ we can prove that it is not in $rng(g)$: if $b =$

$g(a)$ and a is dominated by $u \in X$ then b is dominated by $g(u)$. Elaborating technical details (to be found in [54]) yields the following theorem.

Theorem 9.3.1 ([54, Thm.2.2])

Assume g is (exponentially) hard for a proof system P that contains R. Then tautologies $\sigma_{n,X}$ are (exponentially) hard for P too.

Let us conclude this section by pointing out two further results from [54] based on the general idea of structured PHP that could be of interest to the reader.

First, the idea of using a violation of WPHP was used in [49] to link proof complexity of WPHP and of Ramsey theorem in DNF-resolution systems R($2g$) and R(g) (defined in the same paper), and in [59] to obtain lower bounds for AC^0-Frege proofs of Ramsey theorem with critical parameters. Perhaps more importantly, the idea was used in [54, Sec.4] to show that WPHP considered as an $\mathcal{NP}$-search problem can be reduced to RAM, an $\mathcal{NP}$-search problem defined there and asking to find a size m homogeneous subgraph in a $\mathcal{P}/poly$-graph on $\{0,1\}^m$, and that also breaking RSA or finding a collision in a family of hash functions can be reduced to RAM too.

The second example uses the idea of implicit proofs [53] but here these are proofs of formulas given themselves implicitly. Such formulas are of exponential size but have succint description bit-by-bit by a p-size circuit. The implicit formulas in question express that search problems WPHP and RAM for $\mathcal{P}/poly$-structures have solutions. The result obtained is that if we can prove a suitable bounds for implicit proofs of these formulas, an upper bound for RAM and a lower bound for WPHP, in a *weak* proof systems (even as weak as R*, the tree-like R) then a lower bound for ordinary *strong* proof system (as is EF) can be derived. The details are too technical to even outline here in a reasonably small space and we refer the interested reader to [54, Secs.5 and 6].

9.4 Incompleteness phenomenon

A construction of a p-time generator g_T utilizing the provability in a first-order theory T was given in [69]. The hardness of the generator for all proof systems is an open question and its answer depends on an issue (Problem

9.4.2) related to the incompleteness of theories able to formalize the syntax of first-order logic. We explain the idea and the statements obtained by it but for the details of the proofs the interested reader is referred to [69].

To avoid discussing how the infinite language of $S^1_2(\mathrm{PV})$ is coded by numbers we take as our basic theory S^1_2 of [10] in its finite language denoted here simply L. Note that S^1_2 is finitely axiomatizable and hence its set of axioms (considered as a set of binary strings) is easily definable by an L-formula. Recall also that Σ^b_1-formulas define in $\mathbf{N}$ exactly $\mathcal{NP}$ sets.

The length $|\Psi|$ of an L-formula Ψ is simply the length of the string encoding the formula. We will consider theories $T \supseteq S^1_2$ in language L that are (i.e. the set of strings encoding axioms of T) p-time. It is a classic observation that every r.e. T has a p-time axiomatization (cf. [21]) so this is not a restriction on the power of T.

We shall denote by $u \subseteq_e v$ the fact that string u is an initial subword of string v, and denote by uv the concatenation of u and v. We will also assume that formulas are encoded in such a way that $\Phi \subseteq_e \Psi$ never holds for two formulas unless they are equal.

Now we are ready to define generator g_T, given a sound and p-time theory $T \supseteq S^1_2$ in language L. The instructions for the computation of the function are:

1. Given length n input u find an L formula $\Phi \subseteq_e u$ having one free variable x and such that $|\Phi| \leq \log n$. (Our assumption about coding of formulas implies that there is at most one such formula.)

    - Output $g_T(u) := \overline{0} \in \{0,1\}^{n+1}$ if Φ does not exist.
    - Otherwise go to instruction 2.

2. Go through all $w \in \{0,1\}^{c+1}$, for $c := |\Phi| + 1$, in the lexicographic ordering and look for a T-proof of size $\leq \log n$ of the following L-sentence Φ^w:

$$\exists y \forall x > y \ \Phi(x) \rightarrow \neg(w \subseteq_e x) \ . \tag{9.4.1}$$

    - Output $g_T(u) := \overline{0} \in \{0,1\}^{n+1}$ if a proof is found for all strings w.
    - Otherwise take for $w_0 \in \{0,1\}^{c+1}$ the first string w for which no proof is found, and go to instruction 3.

3. Output $g_T(u) := w_0 u_0 \in \{0,1\}^{n+1}$, where $u = \Phi u_0$.

It is clear that g_T is p-time generator stretching each input by one bit.

Theorem 9.4.1 ([69, Thm.2.2])

Let $A \subseteq \{0,1\}^*$ *be an infinite* L*-definable set and assume that for some definition* Φ *of* A *theory* T *proves all true sentences* Φ^w *from (9.4.1), for* $w \in \{0,1\}^{c+1}$ *where* $c = |\Phi|$.

Then the range of function g_T *intersects* A*.*

Note that if we apply the theorem to $A := \{0,1\}^* \setminus rng(g)$ we get a version of Gödel's First Incompleteness theorem: no sound, p-time $T \supseteq S^1_2$ is complete. In fact, this shows that for *each* formula Φ defining the complement of $rng(g_T)$ some sentence Φ^w is true but unprovable in T. But this still leaves us a little room: the complement of $rng(g_T)$ is in co$\mathcal{NP}$ and hence definable by a Π^b_1 L-formula but not necessarily by a Σ^b_1-formula.

Problem 9.4.2 ($\mathcal{NP}$-definability [69, Prob.2.4])

For some T *as above, can each infinite* $\mathcal{NP}$ *set be defined by* some L-*formula* Φ*such that all true sentences* Φ^w *as in (9.4.1) are provable in* T*?*

The affirmative answer together with Theorem 9.4.1 would imply that g_T satisfies the working conjecture 3.2.2. Note that it is easy to write *some definition* of the set leading to the unprovability but the problem asks whether *all definitions* must lead to it.

We conclude by noting that the argument can be miniaturized to propositional logic and when that is done the following statement can be proved.

Theorem 9.4.3 ([69, Thm.3.1])

At least one of the following three statements is true:

1. *there is no p-optimal propositional proof system,*
2. $\mathcal{E} \not\subseteq \mathcal{P}/poly$*,*
3. *there exists function* h *that stretches all inputs by one bit, is computable in sub-exponential time* $2^{O((\log n)^{\log\log n})}$ *and its range intersects all infinite* $\mathcal{NP}$ *sets.*

The proof can be found in [69].

9.5 Search problems

An important context for the dWPHP problem 2.0.1 and hence for our topic are witnessing theorems for theories around dWPHP (and WPHP), and consequently also results about formalizations of various complexity-theoretic and combinatorial notions and constructions in these theories.

Recall that we have seen in Section 2.2 that witnessing of true sentences of the form

$$\forall x \exists y (|y| \leq |x|^c) A(x, y) \ , \tag{9.5.1}$$

with A a bounded formula, is closely related to their provability in various theories of bounded arithmetic. These witnessing problems are also called (total) search problems. Of a particular interest are the cases when $A \in \Sigma^b_i$ for small i, say $i = 1, 2, 3$, because for A in low levels of the polynomial-time hierarchy the search problems have a more transparent combinatorial meaning (with more than two quantifier alternations the problems become less clear). In particular, the case $i = 1$ leads to the well-know total $\mathcal{NP}$-search problems (their class is confusingly denoted TFNP with F referring to functions).

In the triangle correspondence among theories, complexity classes and proof systems we touched upon in Chapter 2, a bounded arithmetic theory relates to specific search problems S_i and if a theory proves the totality of another problem as (9.5.1) with $A \in \Sigma^b_i$ then it can be reduced to S_i. The opposite often holds too as many reductions are usually given very explicitly and can be formalized in a suitably weak theory. Proving the totality of a search problem often comes down to proving a combinatorial principle underlying why the problem has always a solution. In addition, proofs of the unprovability of one principle from another that are based on witnessing theorems (these do not change when the true universal theory is added) imply a non-reducibility between the associated search problems.

This is all very well established, in some cases for decades. There are many precise statements about the (mutual) provability of combinatorial principles of various complexities in bounded arithmetic theories in terms of witnessing, reducibilities among them (corresponding to provability over various weak theories) and complete problems in such classes. In addition, there are a number of results formalizing various complexity-theoretic constructions around randomized algorithms, fundamentals of derandomization, cryptographic primitives in bounded arithmetic theories utilizing dWPHP or

WPHP, and in theories PV_1, $S^1_2(\mathrm{PV})$ and $S^1_2 + \mathrm{dWPHP}(\Delta^b_1)$ in particular. Explicit natural search problems related to these results were identified.

For reasons that I do not quite understand complexity theorists prefer to ignore this knowledge and to rediscover (or just reformulate) some of it again using a new terminology. This prevents a sensible discussion of a more recent research in the TFNP area that may be related to our topic (and to the dWPHP in particular) unless you are willing to spend a considerable time and to place the more current research into the context of known results established in bounded arithmetic. This is outside of the scope of this book.

Chapter 10

Further research

We have mentioned in earlier chapters three conjectures:

- the working conjecture 3.2.2
- the pseudosurjectivity conjecture 3.3.3
- Razborov's conjecture 5.3.1

and five specific problems:

- the conservativity problem 1.0.1
- the dWPHP problem 2.0.1
- the Kt problem 4.1.1
- the linear generators problem 5.2.3
- the $\mathcal{NP}$ definability problem 9.4.2

In this concluding chapter we shall discuss various problems and research topics that are motivated by the theory and seem to me be interesting but were not treated in depth (or at all) so far. The order in which we present them is ad hoc and does not reflect the subjective importance we give them.

10.1 Ordinary PHP

Having a generator g, at least $(1 - 2^{n-m} \geq 1/2)$-part of strings in $\{0,1\}^m$ are outside $rng(g_n)$. Intuitively, smaller this part is easier it should be to maintain the hardness of proving the $\tau(g)$-formulas. This suggests to look at a situation when maybe just one string from $\{0,1\}^m$ is missing in $rng(g_n)$. That is, look at **dual ordinary PHP**:

- *if* $g : \{0,1\}^n \to \{0,1\}^n$ *then*

$$\exists y \in \{0,1\}^n \forall x \in \{0,1\}^n \ (x \neq 0 \to g(x) \neq y) \ .$$

This principle, to be denoted **dPHP**, is dual to the ordinary PHP which would say that a map from $\{0,1\}^n$ into $\{0,1\}^n \setminus \{0\}$ cannot be injective in the same way dWPHP is dual to WPHP.

Principle PHP for g implies WPHP for the same g. The principle dWPHP is also weaker (over some basic theory) than dPHP. The Introductory chapter 1 mentioned Macintyre's problem about the provability of Δ_0-PHP in full bounded arithmetic and clearly (over the theory) Δ_0-PHP and Δ_0-dPHP are equivalent.

Furthermore, if we have a generator g we can define $g' : \{0,1\}^n \to \{0,1\}^n$ by restricting the output of g to first n bits. Now assume you could prove feasibly in a proof system P a formula

$$\tau'(g')_{b'} \ := \ \|x \neq 0 \to g'(x) \neq b\|^n$$

for some $b' \in \{0,1\}^n \setminus rng(g'_n)$, for infinitely many n. Then g cannot be hard for P either: to prove in P formula $\tau(g)_b$ for $b = b'b''$ where $|b'| = n$ and $|b''| = m - n$ we can combine a P-proof of $\tau'(g')_{b'}$ with an R-proof of $\|x = 0 \to b \neq g(x)\|^n$.

It thus seems of interest to try to develop a theory around dPHP and the τ'-formulas. There is very little known about Macintyre's problem (cf. the last chapter in [45] for some background). In particular, there do not seem to be good candidate function (with a graph in the p-time hierarchy or perhaps even a p-time function) that would be a good candidate or a function for which PHP or dPHP are not provable in full bounded arithmetic. The well-known relativized results of [2, 76, 93] give no hint for this. To investigate the (restrictions to $\{0,1\}^n$ of the) generators studied in earlier chapters may be a good start.

10.2 Power of S-T computations

Various complexity theoretic hypotheses and conjectures entered our discussion. To mention just some:

- the working conjecture 3.2.2,
- Kolmogorov-type hypothesis (H) in Section 8.2,
- hypotheses (I1) and (I2) in Section 8.3,
- hypotheses about circuit size of languages in $\mathcal{E}$ in Lemmas 4.2.6 and 8.2.1,
- the existence of strong OWP at a number of places starting with Theorem 3.6.2,
- hypothesis (J) about the intractability of a search problem in Section 6.5,
- the impossibility to witness a formula (e.g. dWPHP or statement (S)) by S-T computations in $O(1)$ or $n^{O(1)}$ rounds in Chapter 8,
- hypothesis (K) about unsolvability of a search problem via constant round S-T computations in Section 8.6.

These hypotheses have varying informal standing. Some are considered to be quite plausible based on some mental picture about fundamental notions that is accepted by a lot of people, some are claimed to be plausible because they are useful and there are no counter-examples known at present, and some are deemed unlikely. I think that this informal standing ought to be to some extent ignored and we should keep an open mind.

Most of the hypotheses were used in connections with the S-T computations and one cannot escape the thought that the resulting statements that specific some task can or cannot be solved by S-T computations in a certain number of rounds are at least as fundamental - meaning close to fundamental concepts - as are some of the hypotheses above.

I thus think that the power of S-T computations ought to be studied on its own right and the statements giving upper or lower bounds on the number of rounds ought to be investigated as hypotheses on their own. This has been already started quite some time ago but not really followed up; [74] studied the S-T computability of optimization problems in polynomially many rounds (the original problem to which the KPT theorem was first applied was optimization: finding the largest clique in a graph, cf [75]) and showed that the traveling salesperson problem TSP, as well as MAXSAT and MAX3SAT problems, are complete under a natural notion of reducibility defined there (and the max clique problem is complete among those with small values of the objective function), and conjectured that neither of these problems are solvable in polynomial number of rounds. The paper also established a hierarchy theorem for S-T computations determined by the number of rounds, cf. [74, Thm.1].

To give some specific example of a hypothesis of the sort we referred to above let us pose the following question:

- *What computational complexity consequences has the hypothesis that the dWPHP for p-time generators can be always witnessed by an S-T computation with a p-time student in a polynomial number of rounds but not always in a constant number of rounds?*

We know that good example generators are the circuit value function CV (it has parameters) and the truth-table function $\mathbf{tt}_{s,k}$ (no parameters) with $s = 2^{\Omega(k)}$. However, using these functions to get an insight into the problem may not be the best choice.

10.3 Witnessing the infinitude of $\mathcal{NP}$ sets

We have proved (under a hypothesis (H) about circuit size - see Sections 8.2 and 8.7) the consistency of a weakening of the working conjecture 3.2.2 for a class of feasibly infinite $\mathcal{NP}$ sets.

This class is defined by a condition posed on the computational complexity of witnessing formula

$$\mathrm{Inf}_A \ := \ \forall x \exists y (y > x \wedge y \in A)$$

expressing the infinitude of A. When Inf_A is provable in a bounded arithmetic theory one can bound $|y|$ by $|x|^{O(1)}$ (Parikh's theorem) and hence witnessing Inf_A is a (total) $\mathcal{NP}$ search problem. For all naturally occurring bounded arithmetic theories we have a characterization of their $\forall\Sigma_1^b$-consequences by a specific $\mathcal{NP}$ search problem attached to the respective theory T. This means that if Inf_A is provable in T it can be witnessed by an $\mathcal{NP}$ search problem attached to T. For bounded arithmetic background see [45].

As an example we can take theory T^1_2 of [10] that is based on induction axioms for $\mathcal{NP}$ sets. If this theory proves Inf_A then the formula is witnessed by a PLS problem (the Buss-K.theorem [12]). Hence we can define the class of **PLS-infinite** $\mathcal{NP}$ sets to be those $\mathcal{NP}$ sets A for which there is a PLS problem R with parameter x such that any solution y to R for x witnesses Inf_A.

I find the following question interesting:

- *Show (possibly under a reasonable hypothesis) that the working conjecture 3.2.2 is true for the class of PLS-infinite $\mathcal{NP}$ sets.*

Let us point out in a conclusion of this section that we can define a uniform version of the resultant Res_g^P (Def.3.2.4) w.r.t. to a theory. Given a p-time generator g and a theory $T \supseteq T_{\mathrm{PV}}$ define Res_g^T to be the class of $\mathcal{NP}$ sets A such that

$$T \ \vdash \ rng(g) \cap A = \emptyset \ .$$

The hypothesis that g is p-time is used only in order to arrange that the theory has a function symbol for g and we do not need to talk about its definition.

A natural question is:

- *Give an example of a p-time generator and a theory $T \supseteq T_{PV}$ such that Res_g^T contains only finite sets.*

This section expanded on a casual remark in [68].

10.4 Proof search variant

It was pointed out in [68, Sec.6] that the whole topic of proof complexity generators can be modified for (time complexity of) proof search. The modification is fairly simple: essentially replace everywhere $\mathcal{NP}$ sets by $\mathcal{P}$ sets. To explain this let us use the definition of a proof search algorithm from [67]: a **proof search algorithm** is a pair (A, P) such that A is a deterministic algorithm finding for every tautology σ some its P-proof $A(\sigma)$.

The minimal time any algorithm (A, P) needs on σ is measured by the **information efficiency function**

$$i_P : \mathrm{TAUT} \to \mathbf{N}^+$$

that plays the role analogous to the lengths-of-proofs function in this context. The function is defined using algorithmic information and we refer the interested reader to [67]. For each pps P there is an optimal proof search algorithm (A_P, P) having at most polynomial slow-down over any other algorithm; the time it needs on σ is $2^{O(i_P(\sigma))}$, cf. [67].

Following [68] we can now define a set $H \subseteq \mathrm{TAUT}$ to be search-hard for a proof system P analogously how hardness was defined before. H is **search-hard** iff for any $c \geq 1$ algorithm A_P finds a proof of σ in time bounded above by $|\sigma|^c$ for finitely many formulas $\sigma \in H$ only.

Continuing with the analogy call a p-time generator g with the stretch $n+1$ **search-hard** for P iff the set of tautologies $\tau(g)_b$, $b \notin rng(g)$, is search-hard for P. Then the proof search version of the working conjecture 3.2.2 reads as follows.

- **Conjecture 6.1 of [68]**: *There exist a p-time function g extending each input by one bit such that its range $rng(g)$ intersects all infinite $\mathcal{P}$ sets.*

It would be interesting, I think, if one could prove some results about this conjecture that are not analogous to results about the working conjecture 3.2.2.

In a connection with the gadget generator let us point out that the fdp (Def. 4.2.2) studied in Section 4.2 can be naturally modified for the proof search situation too, cf. [68, Sec.6].

10.5 Exponential time generators

Theorem 4.1.3 pointed out that a consequence of the affirmative answer to the Kt problem 4.1.1 is the separation of $\mathcal{NP}$ and $\mathcal{EXP}$. The same argument yields the following more general observation.

Lemma 10.5.1

Assume that there is a function g stretching (Def. 3.1.2) with stretch $m(n)$ that is computable in exponential time $2^{m^{O(1)}}$ and whose range intersects all infinite $\mathcal{NP}$ sets.

Then $\mathcal{NP} \subset \mathcal{EXP}$.

Thus even if such a function g may not have proof complexity consequences (the $\tau(g)$-formulas are so big that their proofs via exhaustive search is p-size) it would still be very interesting to construct such a function unconditionally.

10.6 Function inversion

Let g be a p-time generator having (for the simplicity of the subsequent formulas) the stretch $n+1$ and assume there is a p-time function h inverting g. This can be written as a formula

$$g(h(y)) \neq y \rightarrow g(x) \neq y \ .$$

Define a strong proof system P that extends EF by adding as axioms all instances of the propositional translations of this formula, i.e. instances of

$$\| g(h(y)) \neq y \rightarrow g(x) \neq y \|^{n+1} \qquad (10.6.1)$$

for all $n \geq 1$.

Generator g is not hard for this proof system P. In fact, P admits p-size proofs of not just infinitely many formulas $\tau(g)_b$, $b \notin rng(g)$, but for *all* of them. To construct a P-proof of such $\tau(g)_b$ substitute $y := b$ into (10.6.1), prove true sentence $\|g(h(y)) \neq y\|^{n+1}(y/b)$ and use modus ponens.

It is thus of great interest w.r.t. the working conjecture 3.2.2 (but also in a relation to the argument in Theorem 6.5.1) whether such function inversion is possible or not. It is not very likely: not only the working conjecture would be false but it would also kill pseudo-random generators and one-way functions and a lot of cryptography along the way.

Hence we hope that a general function inversion is not possible with feasible h, and this hope is based on an intuition that the exhaustive search over the domain of g_n cannot be avoided when computing h.

However, an interesting recent result of [29, 78] shows that the intuition, if true, must incorporate into its reasoning also the *uniformity* of h. Namely, they proved that there is always such *non-uniform* h computed by circuits of size $2^{4n/5}n^{O(1)}$ and hence the circuits do avoid the exhaustive search.

The non-uniformity of h does not allow us to construct a proof system as P above. However, it seems quite important for our topic to understand how - if at all - do the underlying constructions relate to proof complexity and in which bounded arithmetic theory do these constructions formalize.

Bibliography

[1] M. Ajtai, Σ_1^1 - formulas on finite structures, *Annals of Pure and Applied Logic*, **24**, (1983), pp.1-48. 86

[2] M. Ajtai, The complexity of the pigeonhole principle, in: *Proc. IEEE 29th Annual Symp. on Foundation of Computer Science*, (1988), pp. 346-355. 75, 86, 138

[3] M. Ajtai, Generalizations of the Compactness Theorem and Gödel's Completeness Theorem for Nonstandard Finite Structures, in: *Proc. of the 4th International Conference on Theory and Applications of Models of Computation*, (2007), pp.13-33. 87

[4] M. Ajtai, A Generalization of Gödel's Completeness Theorem for Nonstandard Finite Structures, unpublished manuscript (2011). 87

[5] M. Alekhnovich, E. Ben-Sasson, A. A. Razborov, and A. Wigderson, Pseudorandom generators in propositional proof complexity, *SIAM J. on Computing*, **34(1)**, (2004), pp.67-88. 13, 30, 32, 41, 42, 61, 62, 63

[6] E. Allender, Applications of Time-Bounded Kolmogorov Complexity in Complexity Theory, in: Kolmogorov Complexity and Computational Complexity, ed.O.Watanabe, Monographs in Theoretical Computer Science, EATCS Ser., Springer-Verlag, (1992), pp.4-22. 46, 47

[7] L. Babai, Trading group theory for randomness, in: *Proc. 17th Annual ACM Symp. on Theory of Computing* (STOC), (1985), pp. 421-429. ACM Press. 107

[8] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang, On the (im)possibility of obfuscating programs, *J. ACM*, **59(2)**, (2012), pp.6:1 - 6:48. 106

[9] A. Beckmann an S. R. Buss, The NP Search Problems of Frege and Extended Frege Proofs, *ACM Transactions on Computational Logic*, **18, 2**, (2017), Article 11. 84

[10] S. R. Buss, *Bounded Arithmetic*. Naples, Bibliopolis, (1986). 14, 19, 23, 132, 141, 161

[11] S. R. Buss, Axiomatizations and conservation results for fragments of bounded arithmetic, in: *Logic and Computation*, Contemporary Mathematics **106**, (1990), pp.57-84. Providence, American Mathematical Society. 19

[12] S. R. Buss, and J. Krajíček, An application of boolean complexity to separation problems in bounded arithmetic, *Proceedings of the London Mathematical Society*, **69(3)**, (1994), pp. 1-21. 141

[13] S. R. Buss and P. Pudlák, How to lie without being (easily) convicted and the lengths of proofs in propositional calculus, in: *Computer Science Logic'94*, Pacholski and Tiuryn eds., Springer-Verlag LN in Computer Science **933**, (1995), pp.151-162. 83, 84

[14] M. Chiari, and J. Krajíček, Witnessing functions in bounded arithmetic and search problems, *J. of Symbolic Logic*, **63(3)**, (1998), pp. 1095-1115. 21

[15] M. Chiari and J. Krajíček, Lifting independence results in bounded arithmetic, *Archive for Mathematical Logic*, **38(2)**, (1999), pp.123-138. 21

[16] A. Cobham, The intrinsic computational difficulty of functions, in : *Proc. Logic, Methodology and Philosophy of Science*, ed. Y. Bar-Hillel, North-Holland, (1965), pp. 24-30. 19

[17] S. A. Cook, Feasibly constructive proofs and the propositional calculus, in: *Proc.* 7^{th} *Annual ACM Symp. on Theory of Computing* (STOC), (1975), pp. 83-97. ACM Press. 19, 25, 38

[18] S. A. Cook, and P. Nguyen, *Logical foundations of proof complexity*, Cambridge U. Press, (2009). 15

[19] S. A. Cook and R. A. Reckhow, The relative efficiency of propositional proof systems, *J. of Symbolic Logic*, **44(1)**, (1979), pp.36-50. 27, 82, 83

[20] S. A. Cook and N. Thapen, The strength of replacement in weak arithmetic, *ACM Transactions on Computational Logic*, **7:4**, (2006). 116

[21] W. Craig, On Axiomatizability Within a System, *J. of Symbolic Logic*, **18(1)**, (1953), pp.30-32. 132

[22] M. Dowd, *Propositional representations of arithmetic proofs*, PhD Thesis, University of Toronto, (1979). 82

[23] P. Erdös, Some remarks on the theory of graphs, *Bull. of the AMS*, **53**, (1947), pp.292-294. 130

[24] G. Frege, *Begriffsschrift: eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*, Halle, (1879). 82

[25] M. Garlík, A New Proof of Ajtai's Completeness Theorem for Nonstandard Finite Structures, *Archive for Mathematical Logic*, **54(3-4)**, (2015), pp. 413-424. 87

[26] M. Garlík, Failure of Feasible Disjunction Property for k-DNF Resolution and NP-hardness of Automating It, preprint (2020), ArXiv: 2003.10230. 50

[27] O. Goldreich, Candidate one-way functions based on expander graphs, preprint in ECCC, Report 90, (2000). 63

[28] O. Goldreich, S. Goldwasser, and S. Micali, How to construct random functions, *J. Assoc. Comput. Mach.*, **33**, (1986), pp.792–807. 63

[29] S. Hirahara, R. Ilango, R. Williams, Beating Brute Force for Compression Problems, preprint, ECCC, TR23-171, (2023). https://eccc.weizmann.ac.il/report/2023/171/ 143

[30] R. Ilango, J. Li and R. Williams, Indistinguishability Obfuscation, Range Avoidance, and Bounded Arithmetic, Electronic Colloquium on Computational Complexity, Report No. 38 (2023). 106, 107

[31] R. Impagliazzo, V. Kabanets, and A. Wigderson, In Search of an Easy Witness: Exponential Time vs. Probabilistic Polynomial Time, *J.Comp.Syst.Sci.*, **65(4)**, (2002), pp.672-694. 58

[32] R. Impagliazzo, M. Naor, Efficient cryptographic schemes provably as secure as subset sum, *J. of Cryptology*, **9(4)**, (1996), pp.199-216. 44

[33] R. Impagliazzo, and A. Wigderson, P = BPP unless E has subexponential circuits: derandomizing the XOR lemma, in: *Proc. of the 29th Annual ACM Symposium on Theory of Computing* (STOC), (1997), pp. 220-229. 51, 58, 78

[34] E. Jeřábek, *Weak pigeonhole principle, and randomized computation*, Ph.D. thesis, Charles University, Prague, (2005). 15, 22, 36

[35] E. Jeřábek, Dual weak pigeonhole principle, Boolean complexity, and derandomization, *Annals of Pure and Applied Logic*, **129**, (2004), pp.1-37. 15, 20, 21, 22, 36, 53

[36] E. Jeřábek, Approximate counting in bounded arithmetic, *J. of Symbolic Logic*, **72(3)**, (2007), pp.959-993. 15, 21, 77

[37] E. Jeřábek, Approximate counting by hashing in bounded arithmetic, *J. of Symbolic Logic*, **7493)**, (2009), pp.829-860. 15

[38] S. Jukna, *Boolean function complexity*, Springer, 2012. 104

[39] R. Kannan, Circuit-size lower bounds and non-reducibility to sparse sets, *Information and Control*, **55(1-3)**, (1982), pp.40-56. 104

[40] E. Khaniki, Nisan-Wigderson Generators in Proof Complexity: New Lower Bounds, in: 37t h Computational Complexity Conf. CCC 2022, July 20-23, 2022, Philadelphia, PA, USA, vol.234, of LIPIcs, pp. 17:1–17:15, (2022). 67

[41] E. Khaniki, Jump operators, Interactive Proofs and Proof Complexity Generators, preprint, (2023).

[42] J. Krajíček, No Counter-Example Interpretation and Interactive Computation, in: *Logic from Computer Science*, ed. Y. N. Moschovakis, Mathematical Sciences Research Institute Publ. 21, Berkeley, Springer-Verlag, (1992), pp. 287-293. 103

[43] J. Krajíček, Fragments of bounded arithmetic and bounded query classes, *Transactions of the A.M.S.*, **338(2)**, (1993), pp.587-598. 19, 81

[44] J. Krajíček, Lower bounds to the size of constant-depth propositional proofs, *J. of Symbolic Logic*, **59(1)**, (1994), pp.73-86. 83

[45] J. Krajíček, *Bounded arithmetic, propositional logic, and complexity theory*, Encyclopedia of Mathematics and Its Applications, Vol. **60**, Cambridge University Press, (1995). 14, 15, 19, 20, 23, 25, 28, 38, 81, 82, 83, 85, 86, 101, 126, 128, 138, 141

[46] J. Krajíček, On Frege and Extended Frege Proof Systems. in: *Feasible Mathematics II.*, eds. P. Clote and J. Remmel, Birkhauser, (1995), pp.284-319. 83, 85, 86

[47] J. Krajíček, A fundamental problem of mathematical logic, *Annals of the Kurt Gödel Society*, Springer-Verlag, *Collegium Logicum*, **2**, (1996), pp.56-64. 81

[48] J. Krajíček, Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic, *J. Symbolic Logic*, **62(2)**, (1997), pp.457-486. 50

[49] J. Krajíček, On the weak pigeonhole principle, *Fundamenta Mathematicae*, Vol.**170(1-3)**, (2001), pp.123-140. 13, 14, 15, 32, 41, 50, 51, 63, 129, 131

[50] J. Krajíček, Tautologies from pseudo-random generators, *Bulletin of Symbolic Logic*, **7(2)**, (2001), pp.197-212. 13, 32, 38, 40, 41, 43

[51] J. Krajíček, Dual weak pigeonhole principle, pseudo-surjective functions, and provability of circuit lower bounds, *J. of Symbolic Logic*, **69(1)**, (2004), pp.265-286. 13, 32, 33, 34, 35, 36, 37, 52, 54, 55, 56, 61, 62, 63, 74

[52] J. Krajíček, Diagonalization in proof complexity, *Fundamenta Mathematicae*, **182**, (2004), pp.181-192. 13, 66, 104

[53] J. Krajíček, Implicit proofs, *J. of Symbolic Logic*, **69(2)**, (2004), pp.387-397. 82, 131

[54] J. Krajíček, Structured pigeonhole principle, search problems and hard tautologies, *J. of Symbolic Logic*, **70(2)**, (2005), pp.619-630. 13, 129, 131

[55] J. Krajíček, Hardness assumptions in the foundations of theoretical computer science, *Archive for Mathematical Logic*, **44(6)**, (2005), pp.667-675. 82, 108

[56] J. Krajíček, Substitutions into propositional tautologies, *Information Processing Letters*, **101(4)**, (2007), pp.163-167. 13, 123, 124, 125, 126

[57] J. Krajíček, A proof complexity generator, in: *Proc. from the 13th Int. Congress of Logic, Methodology and Philosophy of Science (Beijing, August 2007)*, King's College Publications, London, ser. Studies in Logic and the Foundations of Mathematics. Eds. C.Glymour, W.Wang, and D.Westerstahl, (2009), pp.185-190. 13, 71, 73, 74

[58] J. Krajíček, On the proof complexity of the Nisan-Wigderson generator based on a hard $\mathcal{NP} \cap \mathrm{co}\mathcal{NP}$ function, *J. of Mathematical Logic*, **11(1)**, (2011), pp.11-27. 13, 49, 50, 68, 110, 111, 112, 114, 115, 116, 128

[59] J. Krajíček, A note on propositional proof complexity of some Ramsey-type statements, *Archive for Mathematical Logic*, **50(1-2)**, (2011), pp.245-255. 131

[60] J. Krajíček, *Forcing with random variables and proof complexity*, London Mathematical Society Lecture Note Series, **382**, Cambridge University Press, (2011). 13, 15, 42, 44, 58, 59, 74, 76, 82, 94, 95, 96, 97, 98, 99, 116

[61] J. Krajíček, Pseudo-finite hard instances for a student-teacher game with a Nisan-Wigderson generator, Logical methods in Computer Science, Vol. 8 (3:09) 2012, pp.1-8. 13, 110, 112, 116

[62] J. Krajíček, On the computational complexity of finding hard tautologies, *Bulletin of the London Mathematical Society*, **46(1)**, (2014), pp.111-125. 13, 68, 69, 126, 127, 128, 129

[63] J. Krajíček, Consistency of circuit evaluation, extended resolution and total NP search problems, *Forum of Mathematics, Sigma*, **4**, (2016), e15. 84, 85

[64] J. Krajíček, Expansions of pseudofinite structures and circuit and proof complexity, in: *Liber Amicorum Alberti*, eds.Jan van Eijck, Rosalie Iemhoff and Joost J. Joosten, Tributes Ser. **30**, College Publications, London, (2016), pp.195-203. 86, 87

[65] J. Krajíček, *Proof complexity*, Encyclopedia of Mathematics and Its Applications, Vol. **170**, Cambridge University Press, (2019). 15, 20, 25, 27, 28, 34, 36, 37, 38, 42, 49, 50, 63, 64, 66, 74, 75, 76, 81, 82, 83, 84, 86, 94, 126, 128

[66] J. Krajíček, Small circuits and dual weak PHP in the universal theory of p-time algorithms, *ACM Transactions on Computational Logic*, 22, 2, Article 11 (May 2021). 13, 104, 105, 106

[67] J. Krajíček, Information in propositional proofs and algorithmic proof search, *J. of Symbolic Logic*, vol.87, nb.2, (June 2022), pp.852-869. 13, 141, 142

[68] J. Krajíček, On the existence of strong proof complexity generators, (preliminary version August 2022), 13, 32, 46, 47, 49, 120, 121, 141, 142

ArXiv: http://arxiv.org/abs/2208.11642

[69] J. Krajíček, A proof complexity conjecture and the Incompleteness theorem, *J.Symbolic Logic*, to appear. 13, 131, 132, 133

ArXiv: http://arxiv.org/abs/2303.10637

[70] J. Krajíček and P. Pudlák, Propositional proof systems, the consistency of first-order theories and the complexity of computations, *J. Symbolic Logic*, **54(3)**, (1989), pp.1063-1079. 25, 28, 66, 81, 82, 126

[71] J. Krajíček and P. Pudlák, Quantified Propositional Calculi and Fragments of Bounded Arithmetic, *Zeitschr. f. Mathematikal Logik u. Grundlagen d. Mathematik*, Bd. **36(1)**, (1990), pp. 29-46. 82

[72] J. Krajíček and P. Pudlák, Propositional provability in models of weak arithmetic, in: *Computer Science Logic (Kaiserlautern, Oct. '89)*, eds. E. Boerger, H. Kleine-Bunning and M.M. Richter, Lecture Notes in Computer Science **440**, (1990), pp. 193-210. Springer-Verlag. 39, 83

[73] J. Krajíček and P. Pudlák, Some consequences of cryptographical conjectures for S_2^1 and EF", *Information and Computation*, **140 (1)**, (January 10, 1998), pp.82-94. 43, 67

[74] J. Krajíček, P. Pudlák, and J. Sgall, Interactive Computations of Optimal Solutions, in: B. Rovan (ed.): *Mathematical Foundations of Computer Science* (B. Bystrica, August '90), Lecture Notes in Computer Science **452**, Springer-Verlag, (1990), pp. 48-60. 140

[75] J. Krajíček, P. Pudlák and G. Takeuti, Bounded arithmetic and the polynomial hierarchy, *Annals of Pure and Applied Logic*, **52**, (1991), pp.143–153. 19, 22, 23, 103, 109, 140

[76] J. Krajíček, P. Pudlák,and A. Woods, An Exponential Lower Bound to the Size of Bounded Depth Frege Proofs of the Pigeonhole principle", *Random Structures and Algorithms*, **7(1)**, (1995), pp.15-39. 75, 138

[77] S. Kuroda, Developing Takeuti - Yasumoto forcing, preprint (2018), DOI: https://doi.org/10.48550/arXiv.1804.03798 83

[78] N. Mazor, R. Pass, The Non-Uniform Perebor Conjecture for Time-Bounded Kolmogorov Complexity is False, preprint, ECCC, TR23-175, (2023). 143

https://eccc.weizmann.ac.il/report/2023/175/

[79] L. A. Levin, Universal sequential search problems, *Problems of Information Transmission*, **9**, (1973), pp.265-266. 45

[80] J. Li and I. C. Oliveira, Unprovability of strong complexity lower bounds in bounded arithmetic, in: 55th annual ACM Symposium on Theory of Computing (STOC), (2023), pp.1051-1057. 116

[81] N. Nisan and A. Wigderson, Hardness vs. randomness, *J. Comput. System Sci.*, **49**, (1994), pp.149–167. 58, 61, 64, 65, 67, 68

[82] R. Parikh, Existence and feasibility in arithmetic, *J. of Symbolic Logic*, **36**, (1971), pp.494-508. 14

[83] J. Paris and C. Dimitracopoulos, Truth definitions for Δ_0 formulas, in : *Logic and Algorithmic, l'Enseignement Mathematique*, **30**, (1982), pp.318-329, Genéve. 86

[84] J. Paris, A. J. Wilkie and A. Woods, Provability of the Pigeonhole Principle and the Existence of Infinitely Many Primes, *J. of Symbolic Logic*, **53(4)**, (1988), pp.1235-1244. 14, 56, 63

[85] J. Pich, *Hard tautologies*, MSc Thesis, Charles University in Prague, (2011). 67

[86] J. Pich, Nisan-Wigderson generators in proof systems with forms of interpolation, *Mathematical Logic Quarterly*, **57(4)**, (2011), pp.379-383. 67

[87] J. Pich, *Complexity Theory in Feasible Mathematics*, PhD Thesis, Charles University, (2014). 13

[88] J. Pich, Circuit lower bounds in bounded arithmetics, *Annals of Pure and Applied Logic*, Volume **166(1)**, (2015), pp.29-45.

[89] J. Pich, Learning algorithms from circuit lower bounds, preprint (2020). 116

https://arxiv.org/abs/2012.14095

[90] J.Pich and R.Santhanam, Learning algorithms vs. automatability of Frege systems, preprint (2021). 59

ArXiv:2111.10626

[91] J.Pich and R.Santhanam, Why are proof complexity lower bounds hard? in: *IEEE 60th Annual Symposium on Foundations of Computer Science* (FOCS), D.Zuckerman ed., (2019), pp.1305-1324. 48, 59

[92] J.Pich and R.Santhanam, Strong co-nondeterministic lower bounds for NP cannot be proved feasibly, in: Proc. of the 53rd Annual ACM Symposium on Theory of Computing (STOC), (2021), pp.223-233. 116

https://dl.acm.org/doi/abs/10.1145/3406325.3451117

[93] T. Pitassi, P. Beame, and R. Impagliazzo, Exponential lower bounds for the pigeonhole principle, *Computational complexity*, **3**, (1993), pp.97-308. 75, 138

[94] P. Pudlák, On reducibility and symmetry of disjoint NP-pairs, *Theor. Comput. Science*, **295**, (2003), pp.323-339. 49

[95] A. A. Razborov, Unprovability of lower bounds on the circuit size in certain fragments of bounded arithmetic, *Izvestiya of the R.A.N.*, **59(1)**, (1995), pp.201-224. 52

[96] A. A. Razborov, Formulas of bounded depth in the basis $\&, \oplus$ and some combinatorial problems (Russian), *Vopr. Kibern.*, Moscow, **134**, (1988), pp.149-166. 130

[97] A. A. Razborov, Pseudorandom generators hard for k-DNF resolution polynomial calculus resolution, *Annals of Mathematics*, **181(2)**, (2015), pp.415-472. 13, 41, 62, 63, 64, 66, 67

[98] A. A. Razborov and S. Rudich, Natural proofs, *J.Comp. Syst. Sci.*, **55(1)**, (1997), pp.24-35. 44, 58, 126

[99] S. Rudich, Super-bits, demi-bits, and $\mathcal{NP}/qpoly$-natural proofs, in: *Proc. of the 1st Int.Symp. on Randomization and Approximation Techniques in Computer Science*, LN in Computer Science, Springer-Verlag, **1269**, (1997), pp.85-93. 44, 59

[100] G. Takeuti and M. Yasumoto, Forcing in bounded arithmetic, in: *Gödel'96: Logical Foundations of Mathematics, Computer Science, and Physics*, ed. P.Hajek, LN in Logic 6, A.K.Peters, (1996), pp.120 - 38. 83

[101] G. Takeuti and M. Yasumoto, Forcing in bounded arithmetic II, *Journal of Symbolic Logic*, **63**, (1998), pp.860 - 868. 83

[102] N. Thapen, *The Weak Pigeonhole Principle in Models of Bounded Arithmetic*, PhD thesis, Oxford University, (2002). 22

[103] G. C. Tseitin, On the complexity of derivations in propositional calculus, in: *Studies in mathematics and mathematical logic, Part II*, ed. A.O.Slisenko, (1968), pp.115-125. 82

[104] Z. Wang, Implicit resolution, *Logical Methods in Computer Science*, **9(4-7)**, (2013), pp.1-10. 82

[105] A. Woods, *Some problems in logic and number theory, and their connections*, PhD Thesis, U. of Manchester, (1981). 14

[106] A. C.-C. Yao, Theory andapplications of trapdoor functions, in: *Proc. 23rd Ann. IEEE Symp. on Found. of Comp. Sci.* (FOCS), (1982), pp.80-91. 43

Index

– to be compiled at the end –
–

Special symbols

Symbol are listed approximately by their order of appearance and are given a brief explanation.

- PHP: pigeonhole principle
- WPHP: weak PHP
- dWPHP: dual WPHP
- PA: Peano arithmetic
- $I\Sigma_1$: a subtheory of PA with IND for r.e. sets only
- Δ_0: bounded formulas in the language of PA
- Δ_0PHP: PHP for functions with Δ_0-definable graphs
- Δ_0WPHP: WPHP for functions with Δ_0-definable graphs
- $I\Delta_0 + \Omega_1$: Parikh's bounded arithmetic extended by the Ω_1 axiom
- S^1_2: Buss's most important theory with polynomial induction for $\mathcal{NP}$ sets
- $\mathcal{NP}$: non-deterministic polynomial time
- dWPHP(f): formula stating dWPHP for function f
- dWPHP(Δ^b_1): formula dWPHP(f) for all f Δ^b_1-definable in S^1_2
- BT: theory extending S^1_2 by the scheme dWPHP(Δ^b_1)
- TAUT: propositional tautologies in the DeMorgan language

- $[n]$: $\{1, \ldots, n\}$
- PV_1: Cook's universal theory
- $S^1_2(\mathrm{PV})$: S^1_2 together with PV_1 in the expanded language
- dWPHP(PV): the dWPHP for all p-time algorithms
- $\mathcal{P}/poly$: deterministic non-uniform time
- $CV(y, x)$: the circuit value function evaluating circuit y on input x
- dWPHP(CV): the dWPHP for CV
- $\mathrm{dWPHP}_1(CV, CV)$: the dWPHP_1 for CV
- $\preceq_{\Sigma^b_1}$: Σ^b_1-conservativity
- $\mathcal{E}$: small exponential time $2^{O(n)}$
- $P \vdash_* \alpha_n$: there are p-size P-proofs of formulas α_n,
- $\pi : P \vdash \beta$: π is a P-proof of β.
- $P \supseteq \mathrm{EF}$: P extends EF by a p-time set of extra axioms
- $\mathrm{EF} + A$: EF with extra axioms A
- Ref_P and Con_P: reflection and consistency formulas for P
- $\mathbf{s}_P$: the lengths-of-proofs function
- $\tau(C)_b$ or $\tau(g)_b$: τ-formulas
- Def_C: clauses defining the computation of circuit C
- Res^P_g: resultant, the class of $\mathcal{NP}$ (resp. $\mathcal{NP}/poly$) sets whose disjointness with $rng(g)$ have p-size P-proofs
- CF: circuit Frege system
- WF: weak (PHP) Frege system
- $\mathbf{M}_n, \mathbf{M}^*_n$: small and large canonical models

- Def_C: 3-CNF defining instructions of circuit C
- $\mathrm{Def}_C^{n,m,s}$: as Def_C but specifying the number of inputs, outputs and size
- $\tau(C)_b$: the τ-formulas
- $\tau\mathrm{Fla}(g)$: the set of τ-formulas determined by g
- Res_g^P: resultant
- Kt, K^t: time-bounded Kolmogorov complexity
- U, U^t: time-bounded universal Turing machine
- Kt_A: a function measuring the minimal Kt-complexity of strings in A
- $t \times g$: t independent copies of g
- fdp: feasible disjunction property
- $\mathbf{tt}_{s,k}$: the truth-table function
- $Iter(C/\Theta)$: the circuit obtained by iterating C along protocol Θ
- $Size(s(k))$: the class of languages of circuit complexity $\leq s(k)$
- χ_L: the characteristic function of L
- NW: the Nisan-Wigderson generator
- $NW_{A,f}(x)$: NW generator based on matrix A and function f
- $\partial_A(I)$: boundary of a set I of rows of a matrix
- OWP: one-way permutation
- Gad_f: gadget generator
- $CV_{k,a}$: cuircuit-value function for circuits encoded by $\leq a$ bits and computing a function $\{0,1\}^k \to \{0,1\}^{k+1}$
- f_v: gadget function with gadget v fixed
- Gad_{sq}: gadget generator with gadget function CV_{k,k^2}

- ontoPHP: there is no bijection between $[k]$ and $[k+1]$
- $\mathrm{nw}_{k,c}$: NW-like gadgets
- Gad_{nw}: gadget generator using gadgets $\mathrm{nw}_{k,c}$
- $\in_{i.o.}$: "infinitely often" (a language is a member of a class for infinitely many input lengths)
- $\mathcal{J}$: a particular $\mathcal{NP}$ search problem
- ER: Extended resolution
- $\mathbf{B}$: a partial Boolean algebra
- $\Gamma(0, s, k)$: a search problem
- $\mathbf{A}_W$: a non-standard finite structure coded in a model of true arithmetic
- L_{ER}: a language of pseudo-finite structures $\mathbf{A}_W$ related to ER
- T_{ER}: an L_{ER}-theory
- $\mathcal{B}$: a complete Boolean algebra
- $\mathbf{A} \preceq \mathbf{A}'$: elementary extension of a FO structure by a Boolean-valued one
- $\mathcal{D}$: data used to define family F of random variables
- $\alpha_T(\omega) \uparrow$: α_T is undefined at ω
- $\mathrm{Size}(s(n))$: the class of languages decidable by circuits of size $O(s(n)$
- (H): Kolmogorov's hypothesis
- $i\mathcal{O}$: indistinguishability obfuscation
- T_b: a witnessing task related to the NW generator
- $\mathrm{Size}^A(s(k))$: the class of languages L such that L_k can be computed by a circuit of size $\leq s(k)$ querying oracle A
- $\mathcal{K}(c, P)$: a Σ_2^p-search problem

- Inf_A: a sentence expressing the infinitude of set A
- Cert: a search task
- Find: another search task
- RAM: an $\mathcal{NP}$-search problem based on Ramsey theorem
- R*: tree-like R
- g_T: a generator constructed using provability in theory T
- TFNP: the class of total $\mathcal{NP}$ search problems
- dPHP: dual (ordinary) PHP
- $\tau'(g')_{b'}$: modified τ-formulas for dPHP
- T^1_2: a theory from [10] based on induction for $\mathcal{NP}$ sets
- Res^T_g: resultant w.r.t. to theory T
- i_P: the information efficiency function