On the existence of strong proof complexity generators

Jan Krajíček

Faculty of Mathematics and Physics
Charles University*

Abstract

Cook and Reckhow [5] pointed out that $\mathcal{NP} \neq co\mathcal{NP}$ iff there is no propositional proof system that admits polynomial size proofs of all tautologies. Theory of proof complexity generators aims at constructing sets of tautologies hard for strong and possibly for all proof systems. We focus at a conjecture from [16] in foundations of the theory that there is a proof complexity generator hard for all proof systems. This can be equivalently formulated (for p-time generators) without a reference to proof complexity notions as follows:

- There exist a p-time function g stretching each input by one bit such that its range $rng(g)$ intersects all infinite $\mathcal{NP}$ sets.

We consider several facets of this conjecture, including its links to bounded arithmetic (witnessing and independence results), to time-bounded Kolmogorov complexity, to feasible disjunction property of propositional proof systems and to complexity of proof search. We argue that a specific gadget generator from [18] is a good candidate for g. We define a new hardness property of generators, the $\bigvee$-hardness, and shows that one specific gadget generator is the $\bigvee$-hardest (w.r.t. any sufficiently strong proof system). We define the class of feasibly infinite $\mathcal{NP}$ sets and show, assuming a hypothesis from circuit complexity, that the conjecture holds for all feasibly infinite $\mathcal{NP}$ sets.

Keywords: proof complexity generators, bounded arithmetic, weak pigeonhole principle, time-bounded Kolmogorov complexity, proof search, feasible disjunction property.

1 Introduction

A **propositional proof system** (to be abbreviated *pps*) in the sense of Cook and Reckhow [5] is a polynomial time (p-time, shortly) binary relation $P(x, y)$

*Sokolovská 83, Prague, 186 75, The Czech Republic, krajicek@karlin.mff.cuni.cz

such that $\exists x P(x,y)$ defines exactly TAUT, the set of propositional tautologies (in the DeMorgan language for definiteness). The efficiency of a pps P is measured by the **lengths-of-proofs function** s_P: for $\tau \in$ TAUT put

$$s_P(\tau) \ := \ \min\{|\pi| \mid P(\pi,\tau)\} \ .$$

A pps P for which $s_P(\tau)$ is bounded above by $|\tau|^c$ for some independent $c \geq 1$ is called **p-bounded**. As pointed out by Cook and Reckhow [5], the *$\mathcal{NP}$ vs. co$\mathcal{NP}$ problem* (asking whether the computational complexity class $\mathcal{NP}$ is closed under complementation) can be equivalently restated as a question whether a p-bounded pps exists. The existence of a p-bounded pps is thus a fundamental problem of proof complexity.

A pps P is not p-bounded iff there exists an infinite subset $H \subseteq$ TAUT such that for any $c \geq 1$, for only finitely many $\tau \in H$ it holds that $s_P(\tau) \leq |\tau|^c$. Any such set H will be said to be **hard for** P.

There are essentially only two classes of formulas known that make plausible candidates for being hard for strong pps: reflection principles and τ-formulas coming from proof complexity generators. The former class is a classic topic of proof complexity and its exposition can be found in [22, Sec.19.2].

The latter formulas are constructed as follows. Take a function $g : \{0,1\}^* \to \{0,1\}^*$ that stretches all size n inputs to size $m = m(n) > n$ (and hence the complement of its range $rng(g)$ is infinite) and such that its restriction g_n to $\{0,1\}^n$ is computed by a size $m^{O(1)}$ circuit C_n. For each $b \in \{0,1\}^m \setminus rng(g_n)$ encode naturally (as in the proof of the $\mathcal{NP}$-completeness of SAT) the statement

$$|x| = n \to C_n(x) \neq b$$

by a size $m^{O(1)}$ tautology $\tau(g)_b$. Function g is said to be **hard for** P iff the set $\bigcup_{n\geq 1}\{\tau(g)_b \mid b \in \{0,1\}^{m(n)} \setminus rng(g_n)\}$ is hard for P, and we speak of function g as of a **proof complexity generator** in this context.

We shall actually restrict ourselves here[1] to the rudimentary case of generators g computed in time polynomial in n (except the example of function $\mathbf{tt}_{s,k}$ defined below that is computed in time polynomial in m) and, in fact, Lemma 4.2 shows that non-uniformity is to some extent irrelevant.

The $\tau(g)_b$-formulas were defined in [14] motivated by problems in bounded arithmetic and independently (and with an apparently different motivation) in Alekhnovich et al. [1]. Unfortunately the authors of [1] did not pursue the topic[2] and developing the theory was a rather lonely affair until recently. The theory of proof complexity generators has now a number of facets and it is linked not only to bounded arithmetic and proof complexity but also to various topics in computational complexity theory. To give the reader an idea let us mention

[1] Note that one can allow that output bits of generator g are computed in non-uniform $NTime(m^{O(1)}) \cap coNTime(m^{O(1)})$ and still get tautologies of size polynomial in m expressing that $b \notin rng(g_n)$, cf. Razborov [28, Conj.2], [21, Conj.1] and [17, 20]. There are quite a few facts known about such generators and the interested reader may start with [17, 20, 21].

[2] With the sole exception of Razborov [28] written in 2002/03.

(just by key words and phrases) some topics that have a non-trivial contact with the theory:

- lengths-of-proofs lower bounds, feasible interpolation, implicit proof systems, proof search,
- circuit complexity, the minimum circuit size problem, natural proofs, nondeterministic circuits,
- bounded arithmetic, the Incompleteness theorem, provability of upper and lower bounds, forcing with random variables,
- Nisan-Wigderson generators, structural complexity, $\mathcal{NP}$ search problems, Kolmogorov complexity, learning theory,
- pseudo-randomness, one-way functions, indistinguishability obfuscation,

A more detailed presentation of key points of the theory and of the necessary background requires a text of a book-length but the interested reader may look at [22, Sec.19.4-6] (or at older [19, Chpts.29-30]) for an overview and further references. The introduction to Razborov's [28] is an interesting presentation of his ideas about the topic (including a formulation of a conjecture that stimulated some of my own work) and I regret that he did not continue in this research as it would undoubtedly make the theory richer.

Be as it may, the theory as it is now grew out of the motivation for the formulas in [14]: a logic question about the provability of the dual weak PHP principle (dWPHP) for p-time functions in a weak bounded arithmetic theory S^1_2, cf. [14, Problem 7.7]. The dWPHP(f) says that function f does not map any interval $[0, a]$ onto $[0, 2a]$ (the term $2a$ can be altered to various other values, e.g. to a^2 etc., without changing the logical strength of the principle over S^1_2). Denote the theory resulting from adding to S^1_2 all instances of dWPHP(f) for all (suitably defined) p-time functions f by $S^1_2 + \text{dWPHP}(\Delta^b_1)$. The problem (cf. [14, Problem 7.7]) is:

- *Is $S^1_2 + dWPHP(\Delta^b_1)$ equal to S^1_2? If not, is it at least Σ^b_1-conservative over it?*

This problem has a rather rich background and let me try to outline it in one paragraph. A task inherently difficult for bounded arithmetic (and for feasible algorithms) is to count a number of elements of a finite set. It was discovered by A.Woods [31] that explicit counting may be replaced in many arguments in combinatorics or number theory by the pigeonhole principle PHP for bounded formulas, a statement that no bounded formula defines the graph of a function mapping $[0, a + 1]$ injectively into $[0, a]$. It is still unknown whether this principle (denoted Δ_0-PHP) is provable in bounded arithmetic (the problem is due to MacIntyre). Then Paris, Wilkie and Woods [27] found out that the weak PHP (no bounded formula defines the graph of a function mapping $[0, 2a]$ injectively into $[0, a]$), denoted Δ_0-WPHP, often suffices and that this principle is provable in bounded arithmetic (they used theory $I\Delta_0 + \Omega_1$, nowadays it is replaced by

a more convenient Buss's theory S_2). In a parallel development Buss [3] defined a subtheory S_2^1 of S_2 and proved that functions with $\mathcal{NP}$ graphs provably total in this theory are exactly those that are p-time computable. A final twist before the formulation of our problem was a theorem by A.Wilkie (unpublished but presented in [13, 7.3.7]) that functions with $\mathcal{NP}$ graphs provably total in theory $S_2^1 + \mathrm{dWPHP}(\Delta_1^b)$ are computable in randomized p-time. It occurred to me that one may turn the table around and take the theory $S_2^1 + \mathrm{dWPHP}(\Delta_1^b)$ as a basis for formalizing randomized algorithms and to establish its link with randomized p-time analogous to the link between S_2^1 and deterministic p-time. Because randomized algorithms, and probabilistic constructions and argument more generally, are ubiquitarian in combinatorics and complexity theory I denoted in [14] the theory BT for "basic theory". The link was eventually established by Jeřábek in his PhD Thesis and in a subsequent series of papers [9, 10, 11, 12]. 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.

Right from the beginning there were two working conjectures:

1. *There are generators pseudo-surjective for Extended Frege systems EF*, cf.[14, Conj.7.9, Cor.7.10],[15, Conj.4.1,Cor.4.2].

   This conjecture is related to the provability problem mentioned above and the notion of pseudo-surjectivity implies the hardness as defined earlier. We shall touch upon it in Section 2, the reader can find details in [15, 16].

2. *There is a generator hard for all proof systems*, cf.[16, Sec.2].

We shall concentrate here on the second conjecture and we shall restrict our formulation to uniform generators (i.e computed by algorithms not just by sequences of circuits) having the minimal required stretch $m(n) = n+1$. It is easy to see that truncating any p-time generator to output-size $n+1$ preserves the hardness (over any pps simulating resolution). It also allows for a particularly simple formulation of Conjecture 1.1: by [16, Sec.1] (or [22, L.19.4.1]) the second conjecture can be then restated without any reference to proof complexity notions as follows.

Conjecture 1.1 ([16, Sec.2])
There exist a p-time function g stretching each input by one bit such that its range $rng(g)$ intersects all infinite $\mathcal{NP}$ sets. That is, the complement of $rng(g)$ is $\mathcal{NP}$-immune.

A fundamental question of proof complexity is, in my view, whether the hardness of proving a tautology can be traced back to the hardness of computing some computational task associated with the formula. A paradigm of such a reduction is the method of feasible interpolation that applies to a wide range of proof systems albeit not to strong ones (cf. [22, Chpts.17 and 18]). One can interpret Conjecture 1.1 as stating a reduction of provability hardness to computational hardness for all proof systems in the following sense:

- *short proofs*, here witnesses to the membership in an infinite $\mathcal{NP}$ set A,
- *imply an upper bound on compression* for some strings in A, using g as the decompressing algorithm.

With a bit of imagination a direct parallel between the conjecture and feasible interpolation may be seen when we restrict the conjecture. A natural restriction is by requiring that g is hard for a specific pps P only. This is equivalent to restricting Conjecture 1.1 to $\mathcal{NP}$ sets A from the class of those for which P can prove in polynomial size (the tautologies expressing for all lengths $n \geq 1$) that $A \cap rng(g) = \emptyset$. This class of $\mathcal{NP}$ set is the resultant Res_g^P of [16] and the reader can find details there. Conjecture 1.1 restricted to P then says that Res_g^P contains only finite sets. This looks in form similar to feasible interpolation: there we deduce feasible separability of two $\mathcal{NP}$ sets whose disjointness can be proved efficiently in P, here we deduce the finiteness of an $\mathcal{NP}$ set if it can be proved efficiently in P that it is disjoint from a particular $\mathcal{NP}$ set, namely $rng(g)$. Note also that the conjecture restricted to P implies that P is not p-bounded.

Let us give two examples of potential generators (a third one will be discussed in Sec. 4). An illuminating example of a possibly strong generator is the **truth-table function** $\mathbf{tt}_{s,k}$ sending a size s circuit in k inputs to its truth-table (a size 2^k string), cf.[16] or [22, 19.5]. Circuits of size s can be coded by $10s \log s$ bits and so to make the function stretching we assume that $n := 10s \log s < m(n) := 2^k$ (hence size s circuits are coded by $n < m$ bits). It is computed in (uniform) time $O(sm) = 2^{O(k)}$, so it is p-time if $s = 2^{\Omega(k)}$.

The τ-formulas determined by this generator state circuit lower bounds for particular Boolean functions: $\tau(\mathbf{tt}_{s,k})_b \in$ TAUT iff the function with truth-table b has circuit complexity bigger than s. This makes the formulas attractive but also hard to approach as we know very little about the size of general circuits.

It is known that if a pps P admits any pseudo-surjective function (cf. Section 2) then $\mathbf{tt}_{s,k}$ is pseudo-surjective too (the rate of s depends on the rate of lower bound for P establishing the pseudo-surjectivity) and hence hard for P as well, cf.[16] or [19, Sec.30.1]. The first working conjecture above thus implies that the τ-formulas determined by the truth table function are hard for EF. On the other hand, unless $\mathcal{NE} \cap co\mathcal{NE} \subseteq \mathcal{P}/poly$, this generator cannot be hard for all proof systems[3] and hence it is not a good candidate for Conjecture 1.1, cf. [19, p.198].

Our second example follows [24, Remark 6.1] and concerns time-bounded Kolmogorov complexity. Recall that the complexity measure $K^t(w)$ is the minimal size of a program that prints w in time at most $t(|w|)$, cf. Allender [2]. The point is that a proof complexity generator with stretch $m \geq n + \omega(\log n)$ produces strings w of K^t complexity smaller than $m = |w|$. For example, if

[3]But to find a pps for which it is not hard is likely to be a hard task itself, cf.[19, L.29.2.2].

g stretches n bits to $m = 2n$ bits and runs in p-time $t(n)$ then for all size m strings $w \in rng(g_n)$ and $n >> 0$:

$$K^t(w) \leq n + O(\log n) < 2m/3 \ .$$

In fact, as discussed in [24, 6.1], for a fixed polynomial time $t(n)$ sufficient for the computation of g one can consider the universal Turing machine U^t underlying the definition of K^t as a generator itself[4]. Then for any pps P simulating EF, if some $\tau(U^t)$-formulas have short P-proofs (e.g. by proving tautologies expressing the lower bound $K^t(w) \geq 2m/3$) so do some $\tau(g)$-formulas. That is, if there is any g computable in time t and hard for P then U^t must be hard as well.

The paper is organize as follows. In Section 2 we consider the possibility to disprove (or at least to limit possible g in) Conjecture 1.1 by finding a feasible way to witness that the complement of $rng(g)$ is not empty. This is complemented in Section 5 where we link possible limitation to the stretch g can have to the task to prove lower bounds on time-bounded Kolmogorov complexity. We argue that known results imply that these approaches are not likely to work without proving first super-polynomial lower bounds for (uniform and non-uniform) computations.

In Section 3 we discuss a new definition of hardness, the $\bigvee$-hardness, that strengthens (presumably) the hardness as defined above (but is weaker, also presumably, than notion of pseudo-surjectivity mentioned earlier). The reason for introducing the new notion is that a particular generator from the class of gadget generators introduced in [18] and recalled here in Section 4 is the $\bigvee$-hardest[5] among all generators but (presumably) not the hardest under the definition of the hardness as given above: in [18] we used for this result the notion of iterability that is in strength between hardness and pseudosurjectivity mentioned in Section 2, as it was at hand but that is not good for Conjecture 1.1. Namely, it is known (cf. [16]) that if there is any iterable map for a given pps (containing resolution) then $\mathbf{tt}_{s,k}$ is iterable (for it too and hence hard. But by the remark above $\mathbf{tt}_{s,k}$ is unlikely to be hard for all proof systems.

This new notion of $\bigvee$-hardness is equivalent to the hardness as defined above for a class of pps satisfying the strong feasible disjunction property (Section 3). This class has the property that all pps *not in it* are automatically not p-bounded.

We also indicate in Section 6 how to modify the notion of a generator (and the conjecture and results in Sections 2 and 5) to address the hardness of proof search instead of lengths-of-proofs.

In Section 7 we discuss a way how to restrict Conjecture 1.1 and we show, under a hypothesis, that the conjecture holds relative to all *feasibly infinite* $\mathcal{NP}$ sets: sets for which there is a p-time function picking arbitrarily large elements of the set. The paper is concluded by some remarks in Section 8.

[4] A similar observation was made recently in Ren, Santhanam and Wang [29].

[5] Ren, Santhanam and Wang [29] speak informally about the hardest proof complexity generator but what they define is formally an infinite family of generators.

Basic proof complexity background can be found in [22, Chpt.1], the topic of hard formulas (including a brief introduction to the theory of proof complexity generators) is in [22, Chpt.19]. When we use some proof complexity notions and facts in a formal statement we define them first (and give a reference). But we also use proof complexity background in various informal remarks and there we only refer to the original source and/or to a place in [22] where it can be found.

2 Witnessing the dWPHP

The dWPHP for function g extending n bits to $m = m(n)$ bits is formalized by the formula

$$\forall 1^{(n)} \exists y(|y| = m) \forall x(|x| = n)\ g(x) \neq y\ .$$

Notation $\forall 1^{(n)}$ means that the universal quantifier ranges over all strings $1 \dots 1$ of any length n. To witness this formula means to find a witness y for the existential quantifier given $1^{(n)}$ as input. This task became known recently in complexity theory as the *range avoidance problem*[6].

Witnessing is a classic notion of proof theory[7] and, in particular, many fundamental results in bounded arithmetic are formulated as follows: if theory T proves a formula of a certain syntactic complexity then it can be witnessed (i.e. its leading $\exists$ can be witnessed) by a function from a certain computational class C. Such statements are known[8] for many basic bounded arithmetic theories, many natural syntactic classes of formulas and computational classes of functions.

Unprovability results are generally difficult and usually conditional, and we shall use one below. But in the relativized set-up (in our situation this would mean that g is given by an oracle) many unconditional unprovability results are known and they are usually derived by showing that a principle at hand cannot be witnessed by a function in some particular class C (for dWPHP see the end of this section).

We now give an application of the conditional unprovability result of [23]. Consider theory $T_{\rm PV}$ whose language has a k-ary function symbol f_M attached to every p-time clocked machine M with k inputs, all $k \geq 1$. The symbol f_M is naturally interpreted on $\mathbf{N}$ by the function M computes. The axioms of $T_{\rm PV}$ are all universal sentences in the language true in $\mathbf{N}$ under this interpretation.

The hypothesis used in the unprovability result is this.

Hypothesis (H):

[6] That problem deals with functions computed by circuits and the input to the task is the circuit itself; that is included in the formulation above as g can have parameters (not shown in the notation).

[7] In particular, witnessing of dWPHP is discussed in [16, Sec.7].

[8] 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. For reasons that I do not quite understand complexity theorists prefer to ignore this knowledge and rediscover some of it again in a language avoiding but still simulating logic concepts.

There exists 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 possibility that (H) is true with $d = 1$ is attributed to Kolmogorov but it is not a hypothesis accepted by mainstream complexity theory. However, there are no technical results supporting the skepticism. In fact, (H) has a number of interesting consequences as is $\mathcal{P} \neq \mathcal{NP}$ or $\mathcal{E} \subseteq Size(2^{o(n)})$ (the latter is bad for universal derandomization but it is good for proof complexity, cf. [17, 23]).

The following theorem uses $g := \mathbf{tt}_{s,k}$ with $s = 2^{\epsilon k}$ for a fixed $0 < \epsilon < 1$ for our p-time function. The dWPHP for this function can be expressed by formula:

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

where $n = n(m) := 10s \log s$. We chose m as the natural parameter: m and n are polynomially related and determine each other so this indeed expresses dWPHP.

Theorem 2.1 ([23])
Assume hypothesis (H). Then for every $0 < \epsilon < 1$ and $s = s(k) := 2^{\epsilon k}$ the theory T_{PV} does not prove the sentence (1).

The proof of this theorem in [23] goes by showing that (1) cannot be witnessed in a particular interactive way discussed below. However, we want to stress that the unprovability result itself, perhaps proved from other hypotheses (or unconditionally) not using witnessing methods (but using model theory instead, for example) implies the impossibility to witness (1) in a particular way.

To illustrate the idea simply we start by showing that (1) cannot be witnessed by a p-time function f. The property that f witnesses (1) is itself a universal statement

$$\forall 1^{(m)} \forall x (|x| = n)\ (|f(1^{(m)}| = m \wedge g(x) \neq f(1^{(m)}))$$

and hence, if true, an axiom of T_{PV}. As this axiom easily implies (1) we get a contradiction with Theorem 2.1.

In fact, it is easy to see (as pointed out by one of the referees) that for any specific $0 < \epsilon < 1$ the existence of a p-time witnessing function f for (1) is equivalent to the existence of a language in $\mathcal{E} \setminus Size(2^{\epsilon n})$: the set

$$\{f(1^{(2^\ell)}) \mid \ell \geq 1\}$$

for any potential p-time f consists of the collection of characteristic functions of a language in $\mathcal{E}$ for input lengths $\ell \geq 1$, and vice versa.

Consider now an interactive model of witnessing via constant round Student - Teacher computation. In this model of computation (cf. [26, 25]) p-time student S, given $1^{(n)}$, produces his candidate solution $b_1 \in \{0,1\}^m$. A computationally unlimited teacher T either acknowledges the correctness or she produces a counter-example: $x_1 \in \{0,1\}^n$ s.t. $g(x_1) = b_1$. S then produces his second candidate solution b_2 using also x_1, T either accepts it or gives counter-example

x_2 etc. The requirement is that within a given bound t on the number of rounds S always succeeds. This can be written in a universal way as

$$g(x_1) \neq S(1^{(n)}) \vee g(x_2) \neq S(1^{(n)}, x_1) \vee \ldots \vee g(x_t) \neq S(1^{(n)}, x_1, \ldots, x_{t-1}) \ . \quad (2)$$

Let us remark that S-T protocol with polynomially many rounds $t = m^{O(1)}$ relate to the notion of pseudo-surjectivity mentioned in the Introduction while $O(1)$ rounds correspond to the notion of freeness: the universal statement (2) can be represented by an infinite family of p-size tautologies and the two hardness notions require that these tautologies do not have short proofs, cf.[15, 16] for details.

Let us state the conclusion of this discussion formally.

Theorem 2.2

Assume hypothesis (H). Then dWPHP for function $\mathbf{tt}_{s,k}$ *with parameters as in Theorem 2.1 cannot be witnessed by a Student-Teacher computation with p-time Student and constantly many rounds.*

Hence to witness the non-emptiness of the complement of $\mathbf{tt}_{s,k}$ with parameters as in Theorem 2.1 by a constant round S-T protocol with p-time student would imply arbitrarily high polynomial lower bounds for circuits computing a language in $\mathcal{P}$.

Recently Ilango, Li and Williams [8] proved that the dWPHP for the circuit value function CV (cf. Section 4) is not provable in $T^1_{\rm PV}$ by showing that it cannot be witnessed in by S-T computation with parameters as in Theorem 2.2, assuming a couple of hypotheses of a different nature: that $co\mathcal{NP}$ is not infinitely often in the Arthur-Merlin class AM and a heuristically justified conjecture in cryptography about the security of the indistinguishability obfuscation $i\mathcal{O}$.

One may wonder whether the hypothesis that the dWPHP cannot be in general witnessed by a constant-round (or even with polynomially many rounds) S-T protocol with a p-time student is not more fundamental than the hypotheses above used to derive it.

If we manage to extend the unprovability to theory $T^1_{\rm PV} \cup S^1_2$ then we would rule out witnessing by S-T computation with polynomially many rounds. Extending it further to theory $T^1_{\rm PV} \cup T^1_2$ (or equivalently to $T^1_{\rm PV} \cup S^2_2$) would rule out witnessing by p-time machines accessing an $\mathcal{NP}$ oracle. All these statements need to be conditional as they imply (unconditionally) that $\mathcal{P}$ differs from $\mathcal{NP}$: if $\mathcal{P} = \mathcal{NP}$ then this is implied by a true universal statement in the language of $T^1_{\rm PV}$ (saying that a particular p-time algorithm solves SAT) and hence all true universal closures of bounded formulas are equivalent over $T^1_{\rm PV}$ to universal statements which are axioms of $T^1_{\rm PV}$.

Further note that in the relativized world we have a number of unconditional results about the impossibility to witness dWPHP. As an example let us mention that we cannot witness by non-uniform p-time machine (i.e. using a sequence of polynomial size circuits, cf. Sipser [30]) with an access to an $\mathcal{NP}^R$ oracle where R is the graph of g that g is not a bijection between $[0, a]$ and $[0, 2a]$. Another

example is that even if we have oracle access to g and to another function f we cannot witness by a PLS problem with base data defined by p-time machines with oracle access to f, g that g is not a bijection between $[0, a]$ and $[0, 2a]$ with f being its inverse map. The interested reader can find these results (and all background) in [13, Secs.11.2-3] and in references given there.

3 Feasible disjunction property and $\bigvee$-hardness

We shall propose in this section a notion of hardness that is preserved by more constructions (and, in particularly, by the construction underlying gadget generators in Section 4) than is the original hardness but is presumably weaker than a stronger notion of iterability (mentioned in the introduction) used in [18].

Definition 3.1

A function $g : \{0,1\}^* \to \{0,1\}^*$ that for any $n \geq 1$ stretches all size n inputs to size $m := m(n) > n$ and such that g_n (the restriction of g to $\{0,1\}^n$) is computed by size $m^{O(1)}$ circuits is **$\bigvee$-hard** *for a pps P is for any $c \geq 1$, only finitely many disjunctions*

$$\tau(g_n)_{b_1} \vee \ldots \vee \tau(g_n)_{b_r} \ , \tag{3}$$

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

Note that the definition can be formulated equivalently as saying that the set of all valid disjunctions of the form (3) is hard for P.

A pps P has the **feasible disjunction property** (abbreviated *fdp*) iff whenever a disjunction $\alpha_0 \vee \alpha_1$ of two formulas having no atoms in common has a P-proof of size s then one of α_i has a P-proof of size $s^{O(1)}$. The **strong fdp** is defined in the same way but the starting disjunction can have any arity r: $\bigvee_{i<r} \alpha_i$. The strong fdp plays a role in analysis of a proof complexity generator in [20], see also [22, Subsec.17.9.2]. It is an open problem ([22, Prob.17.9.1]) whether, for example, Frege or Extended Frege systems have the (strong) fdp. Let us note that Garlík [6] proved that proof systems $R(k)$ of [14] have no fdp.

Lemma 3.2 *Assume a pps P has the strong fdp. Then any generator hard for P is also $\bigvee$-hard for P.*

Lemma 3.3 *Assume that g is a function stretching size n inputs to size $n+1$ and such that g_n (the restriction of g to $\{0,1\}^n$) is computed by size $n^{O(1)}$ circuits and is $\bigvee$-hard for a pps P.*

The for all $\delta > 0$ there is g' computed by size $n^{O(1)}$ circuits and stretching size n inputs to size $n + n^{1-\delta}$ that is $\bigvee$-hard for P.

Proof :

Let g' computes g in parallel on n^c many different inputs of size n: it stretches n^{c+1} bits into $n^{c+1} + n^c$ bits. As the τ-formulas for g' are disjunctions of the τ-formulas for g, the lemma follows by taking $c \geq 1$ large enough.

q.e.d.

A strategic choice: use $\bigvee$-hardness

As it was pointed out in [20], for the purpose of proving lengths-of-proofs lower bounds for some pps P we may assume w.l.o.g. that P satisfies the strong fdp: otherwise it is not p-bounded and we are done. This observation, together with Lemma 3.2, justifies the use of $\bigvee$-hardness rather than mere hardness.

The reader skeptical about the choice may interpret the statements contrapositively as sufficient conditions refuting the strong fdp for a particular pps, cf. Lemma 5.4. In particular, it may happen that no strong pps has the strong fdp: but then we can celebrate as $\mathcal{NP} \neq co\mathcal{NP}$.

4 The gadget generator

The class of gadget generators was introduced in [18] and it is defined as follows. Given any p-time function

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

define a **gadget generator based on** f

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

where

$$n := \ell + k(\ell+1) \ \text{ and } \ m := n+1$$

as follows:

1. *The input $\overline{x} \in \{0,1\}^{n}$ 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. *The output $\overline{y} = Gad_f(\overline{x})$ is the concatenation of $\ell+1$ strings $w^s \in \{0,1\}^{k+1}$ where we put*

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

Clearly we may fix f w.l.o.g. to be the **circuit value function** $CV_{a,k}(v,u)$ which from a size a description v of a circuit (denoted also v) with k inputs and $k+1$ outputs and from $u \in \{0,1\}^{k}$ computes the value of v on u, an element of $\{0,1\}^{k+1}$.

It was shown in [18] (see also [22, L.19.4.6]) that if we replace the hardness of a generator by a stronger condition then it suffices to consider circuits v of size $\leq k^{1+\epsilon}$, any fixed $\epsilon > 0$. The proof of this fact in [18] used the notion of iterability mentioned earlier, as it was at hand. However, the same argument gives Theorem 4.1 using presumably weaker notion of $\bigvee$-hardness from Section 3; the proof in [18] was only sketched so we give it here. Recall that a pps P **simulates** Q iff for all $\sigma \in$ TAUT it holds that $s_P(\sigma) \leq s_Q(\sigma)^c$.

Theorem 4.1 (ess.[18])

Let P be a pps simulating EF and having the following properties. 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: $s_P(\beta) \leq (s_P(\alpha) + s_P(\alpha \to \beta))^c$.*

Assume that there exists a p-time function $g : \{0,1\}^* \to \{0,1\}^*$ that stretches all size n inputs to size $m := m(n) > n$ and is $\bigvee$-hard for P.

Then the gadget generator based on $CV_{k^2,k}$ is $\bigvee$-hard (and hence also hard) for P as well.

Proof :

Assume P and g satisfy the hypotheses of the theorem; w.l.o.g. we may assume that $m(n) = n + 1$. Let C_k be a canonical circuit of size polynomial in k that computes g_k and let C_k be encoded by a string $\lceil C_k \rceil$ of size $\ell \leq k^a$, some constant $a \geq 1$.

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

Note that the τ formula for Gad_f and $b = (b^1, \ldots, b^t) \in \{0,1\}^{n+1}$ is a t-size disjunction, $t = k^a + 1$, of τ-formulas for $CV_{k^a,k}$ and b^i, $i \leq t$. Substitute there for (atoms defining) the gadget $v := \lceil C_k \rceil$. Using that EF has p-size proofs[9] of

$$CV_{k^a,k}(\lceil C_k \rceil, u) = C_k(u)$$

and $P \geq EF$, any proof of the original disjunction for Gad_f is turned into a polynomially longer P-proof of a disjuction of τ-formulas for g, contradicting the hypothesis.

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

Note that Gad_f in Claim 1 is computed in time $O(k^{2a})$ which is $\leq n^{2-\delta}$ for some $\delta > 0$. Hence we may perform the same construction as in Claim 1 but using Gad_f instead of g now.

q.e.d.

Notation:

In the rest of paper we shall ease on the notation and we will denote the gadget generator Gad_f based on $f = CV_{k^2,k}$ by Gad_{sq} (sq stands for square).

Note that a circuit of size s can be encoded by $10s \log s$ bits so Gad_{sq} uses as gadgets circuits of size little bit less than quadratic. Observe also that Gad_{sq} is computed in time smaller than $n^{3/2}$.

The next statement shows that non-uniformity is irrelevant in the presence of strong fdp. It is proved analogously as Theorem 4.1 by taking for gadgets circuits needed to compute the generator.

[9] When $CV_{\ell,k}$ is defined naturally by induction on the size of the circuit and the encoding $\lceil C_k \rceil$ uses log-size addresses of subcircuits it would suffice to assume $P \geq R(\log)$, cf.[22] for the proof system.

Lemma 4.2 *Assume a pps P satisfis the hypotheses of Theorem 4.1 and that it admits a $\bigvee$-hard proof complexity generator computed in non-uniform p-time (i.e. by p-size circuits). Then Gad_{sq} is $\bigvee$-hard for P.*

It is known that gadget generators (and Gad_{sq} in particular) are hard for many proof systems for which we know any super-polynomial lower bound, cf.[22]. Our **working hypothesis** is that the generator Gad_{sq} satisfies Conjecture 1.1. But when working with the generator we encounter the same difficulty as in the case of the truth-table generator $\mathbf{tt}_{s,k}$: we know nothing non-trivial about circuits of sub-quadratic size. Furthermore, the experience with lengths-of-proofs lower bounds we have so far suggests that it is instrumental to have hard examples with some clear combinatorial structure. Hence to study the hardness of Gad_{sq} it may be advantageous to consider gadgets (i.e. sub-quadratic circuits) of a special form (technically that would be a substitution instance of Gad_{sq}).

One such specific generator was defined in [22, pp.431-2] and denoted $\mathrm{nw}_{k,c}$ there; its gadget is essentially a slightly over-determined system of sparse equations for a generic function h. Namely the gadget consists of:

- $k+1$ sets $J_1, \ldots, J_{k+1} \subseteq \{x_1, \ldots, x_k\}$, each of size $1 \leq c \leq \log k$,
- together with 2^c bits defining truth table of a Boolean function h with c inputs.

Given gadget v and $u \in \{0,1\}^k$, $f(v,u) \in \{0,1\}^{k+1}$ are the $k+1$ values h computes on values that u gives to variables in sets $J_1, \ldots, J_{k+1}$. This generator for one fixed, non-uniform gadget was the original suggestion for Conjecture 1.1 in [16] but the gadget generator construction allows to avoid the non-uniformity and consider generic case.

5 Stretch and the Kt-complexity

The main aim of proof complexity generators is to provide hard examples and for this purpose the stretch $n+1$ of g in Conjecture 1.1 suffices (and it yields the shortest τ-formulas). A larger stretch is of interest in a connection[10] with the truth-table function $\mathbf{tt}_{s,k}$ discussed earlier.

We may try to limit possible stretch of generators via some considerations involving time-bounded Kolmogorov complexity as we touched upon in the Introduction. We shall use Levin's measure $Kt(w)$: the minimum value of $|d| + \log t$, where program d prints w in time t, cf. Allender [2]. Its advantage over K^t is that it does not require to fix the time in advance. Although statement like $Kt(w) \geq 2m/3$ can presumably not be expressed by a p-size (in m) tautology, certificates for the membership in an $\mathcal{NP}$ set A such that all $w \in A$ satisfy $Kt(w) \geq 2|w|/3$ can be interpreted as proofs of $Kt(w) \geq 2m/3$.

[10] In fact, the need for larger stretch even in this connection seems to be eliminated by the notion of iterability, cf. [16].

Let us consider a function with an extreme stretch: $\mathbf{tt}_{s,k}$ with $s = 100k$. This generator sends $n = 10s \log s \leq O(\log m \log\log m)$ bits to $m = 2^k$ bits and is computed in time $t = O(sm) < m^{3/2}$. Hence both K^t and Kt are bounded above on $rng(\mathbf{tt}_{s,k}) \cap \{0,1\}^m$ by $O(\log m \log\log m)$.

Notation (Allender [2]):
For any set $A \subseteq \{0,1\}^*$ define function $Kt_A : \mathbf{N}^+ \to \mathbf{N}^+$ by

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

if the right-hand side is non-empty, and we leave $Kt_A(m)$ undefined otherwise.

Hence we could limit the extreme stretch as in the above $\mathbf{tt}_{s,k}$ if we could find an infinite $\mathcal{NP}$ set A such that $Kt_A(m) \geq \omega(\log m \cdot \log\log m)$. Unfortunately the next theorem suggests that this is likely not an easy task. Following Allender [2] we define an $\mathcal{NE}$ **search problem** to be a binary relation $R(x,y)$ such that R implicitly bounds $|y|$ by $2^{O(n)}$ for $|x| = n$ and which is decidable in time $2^{O(n)}$ (think of y as an accepting computation of an $\mathcal{NE}$ machine on input x). The search task is: given x, find y such that $R(x,y)$, if it exists. As an example related to our situation let A be an $\mathcal{NP}$ set defined by condition

$$u \in A \ \text{ iff } \ \exists v(|v| \leq |u|^c) S(u,v)$$

with S a p-time relation, and consider $R(x,y)$ with $y = [y_1, y_2]$ defined by:

$$|y_1| = x \wedge |y_2| \leq |y_1|^c \wedge S(y_1, y_2) \ .$$

Theorem 5.1 (Allender [2, Cor.7,Thm.8])
There exists an infinite $\mathcal{NP}$ set A s.t. $Kt_A(m) = \omega(\log m)$ 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.*

Hence ruling out generators with even very large stretch means likely to prove significant computational lower bounds. The following seems to be a natural test question.

Problem 5.2
Is it true that any infinite $\mathcal{NP}$ set A contains a string $w \in A$ with $Kt(w) < |w|$? That is, is it true that the set $\{w \mid Kt(w) \geq |w|\}$ is $\mathcal{NP}$-immune?

Theorem 5.3

1. *If Problem 5.2 has the negative answer then the range of no p-time generator g stretching n bits to $n + \omega(\log n)$ bits can intersect all infinite $\mathcal{NP}$ sets.*

2. *If Problem 5.2 has the affirmative answer then $\mathcal{NP}$ is a proper subclass of $\mathcal{EXP}$.*

Proof :

For the first part note that all strings in the range of g_n (g restricted to $\{0,1\}^n$) have Kt-complexity at most $n + O(\log n)$.

For the second part note that there is a function g computable in time $2^{O(n)}$ such that the range of g_n is the set of $w \in \{0,1\}^{n+1}$ with $Kt(w) \leq n$. We have that $rng(g) \in \mathcal{E}$ and hence $\{0,1\}^* \setminus rng(g)$ is also in $\mathcal{E}$ but it cannot be - assuming the affirmative answer to the problem - in $\mathcal{NP}$. This implies that $\mathcal{E} \not\subseteq \mathcal{NP}$ and hence also $\mathcal{EXP} \not\subseteq \mathcal{NP}$. As $\mathcal{NP} \subseteq \mathcal{EXP}$ we have $\mathcal{NP} \subset \mathcal{EXP}$.

q.e.d.

We would rather like to see the affirmative answer; not only it has nice corollary by the previous theorem, but it is also in the spirit of a potential reduction of provability hardness to computational hardness discussed after Conjecture 1.1. Note that the problem has the affirmative answer for all $\mathcal{NP}$ sets defined in the CSP (constraint satisfaction problem) format: if an instance X of size n has a solution so do instances obtained by taking t disjoint copies (i.e. in disjoint sets of variables) of X, and these have Kt-complexity at most $O(n + \log t + \log tn)$ which is less than the size tn of the new instance if $t > 1$ and $n >> 1$.

Let us consider the stretch of gadget generators. By default it was taken in the definition to be the minimal required stretch but there are other options. One could use as gadgets circuits that map k bits to k' bits where $k' >> k$; for example, $k' = 2k$ or $k' = k^2$ (allowing accordingly a bigger size of gadgets, still polynomial in k). The resulting generator would send n bits to approximately $(k'/k)n$ bits which is about $n^{1+\epsilon}$ for some $\epsilon > 0$, for $k' = k^2$.

However, we want to be conservative with requirements on gadgets. Note that the stretch of gadget generators can be influenced also by taking more strings u^i in the construction of Gad_f than is the minimal number needed, i.e. more than $\ell+1$. In particular, assume we perform the construction of Gad_f but taking $t >> \ell$ strings u^i and w^i. We still want to maintain, as in Theorem 4.1, that the generator is the $\bigvee$-hardest generator; hence we allow only t polynomial in k. Then

$$n := \ell + kt \quad \text{and} \quad m := (k+1)t \ .$$

For $\ell \leq k^{O(1)}$ (as in Gad_{sq}) and taking $t := k^c$ for very large $c \geq 1$ we can arrange that

$$m \ \geq \ n + n^{1-\epsilon}$$

for as small $\epsilon > 0$ as wanted. Denote the generator which extends the definition of Gad_{sq} in this way by Gad^c_{sq}.

Lemma 5.4

Assume that there is an infinite $\mathcal{NP}$ set A such that for some $\delta > 0$:

$$Kt_A(m) \geq m - m^{1-\delta} \ .$$

Assume further that Conjecture 1.1 is true.

Then there is a pps P such that no pps Q simulating P has the strong fdp.

Proof :

Choose $c \geq 1$ so large that the stretch of Gad^c_{sq} is $n^{1-\epsilon}$, $\epsilon = \epsilon(c)$, where

$$m^{1-\delta} = (n + n^{1-\epsilon})^{1-\delta} < n^{1-\epsilon} + 2\log n$$

for $n >> 0$ (taking $c \geq 1$ such that $0 < \epsilon(c) < \delta$ suffices).

Given an infinite $\mathcal{NP}$ set A satisfying the hypothesis define a pps P to be, say, resolution but accepting also witnesses to the membership of $b \in A$ as proofs of $\tau(\mathrm{Gad}^c_{sq})_b$. It is sound as A must be disjoint from the range of Gad^c_{sq}.

If Conjecture 1.1 was true for some g and some Q simulating this P, and Q would satisfy the strong fdp, it would follow by Lemma 3.2 that g is $\bigvee$-hard for Q and hence by Theorem 4.1 (modified trivially for Gad^c_{sq}) that Gad^c_{sq} is $\bigvee$-hard (and hence also hard) for Q. That is a contradiction with how P was defined.

q.e.d.

6 Modifications for proof search hardness

Proof complexity generators, and Conjecture 1.1 in particular, aim primarily at the problem to establish lengths-of-proofs lower bounds. It is easy to modify the concept to aim at time complexity of proof search. Essentially this means to replace everywhere in the previous sections $\mathcal{NP}$ sets by $\mathcal{P}$ sets. To give a little more detail we shall use the definition of a proof search algorithm from [24]: it is a pair (A, P) such that A is a deterministic algorithm that finds for every tautology its P-proof. How much time any algorithm (A, P) has to use on a particular tautology is measured by the information efficiency function $i_P : \mathrm{TAUT} \to \mathbf{N}^+$; it is an inherently algorithmic information concept. For each pps P there is a time-optimal (A_P, P) (it has at most polynomial slow-down over any other proof search algorithm) which is also information-optimal. The reader can find definitions and proofs of these facts in [24].

Define a set $S \subseteq \mathrm{TAUT}$ to be **search-hard for** P 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 $\sigma \in S$ only. Then analogously with the definition of hardness we define g (in the format as in Conjecture 1.1, i.e. p-time stretching each input by one bit) to be **search-hard for** P iff the complement of $rng(g)$ is search-hard for P. It can be shown that the conjecture that there is a uniform generator search-hard for all pps is then equivalent to

Conjecture 6.1 (proof search version of Conjecture 1.1)

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. That is, the complement of $rng(g)$ is $\mathcal{P}$-immune.

There are some more facts known about Kt_A measure for sets in $\mathcal{P}$ (note that Theorem 5.1 was about); for example, Allender [2, Thms.6,8] or Hirahara [7, Thm.3.11]. These results seem to suggest that Conjecture 6.1 may not be any easier to prove than Conjecture 1.1.

Let us conclude this section by noticing that the fdp can be naturally modified for proof search as well: the modification requires that the time A_P needs on α_0 or α_1 is bounded above by a polynomial in time it needs on $\alpha_0 \vee \alpha_1$. However, such a property implies the usual feasible interpolation property. Namely, if π is a P-proof of a disjunction

$$\gamma_0(x, y) \vee \gamma_1(x, z)$$

(the disjuncts are not required to have disjoint sets of variables this time) consider disjunction $\beta \vee (\gamma_0 \vee \gamma_1)$ where β is a propositional sentence that is the conjunctions of 0 with all bits of π. Then A_P (recall from Sec. 6 that (A_P, P) is time-optimal) when given this disjunction reads π and hence proves $\gamma_0 \vee \gamma_1$ and thus also $\beta \vee (\gamma_0 \vee \gamma_1)$. By the search-version of fdp A_P must find in time polynomial in $|\pi|$ a proof of $\gamma_0 \vee \gamma_1$ (as β is false) and thus also of any instance $\gamma_0(a, y) \vee \gamma_1(a, z)$ (this requires that P-proofs are closed under substitution of constants as in Theorem 4.1). By the new property again algorithm A_P, for each a succeeds on either $\gamma_0(a, y)$ or on $\gamma_1(a, z)$ in time polynomial in $|\pi|$. That yields feasible interpolation. This observation means that the proof search variant of fdp cannot hold for any strong proof systems and is subject to same limitations as is feasible interpolation and, in particular, cannot hold for any strong proof systems unless some standard cryptographic assumptions fail. The reader can find all background in [22].

7 Feasibly infinite $\mathcal{NP}$ sets

Two natural ways how to make Conjecture 1.1 weaker and hence more tractable are to either allow generator g from a larger class of functions than just p-time computable or to restrict the requirement of the finiteness only to a subclass of all $\mathcal{NP}$ sets. We have seen in Theorem 5.3 that finding g in exponential time would imply $\mathcal{NP} \subset \mathcal{EXP}$ so such a weakening is definitely interesting although it may not advance proof complexity. In this section we look at how to restrict sensibly the class of $\mathcal{NP}$ sets in the conjecture.

We have seen one such restriction in the Introduction (classes Res_g^P). There is, however, another natural restriction of the class of $\mathcal{NP}$ sets in the conjecture possible. Take a sound theory T whose language extends 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, representing $y \in A$ 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 defining a p-time relation. Hence Inf_A is an $\forall\exists$-sentence.

Knowing that a particular T proves Inf_A yields, in principle, non-trivial information about A. For example, if $T_{\rm PV}$ proves the sentence then by applying Herbrand's theorem we get a p-time function f witnessing it. That is, f finds elements of A:

$$\forall x(f(x) > x \wedge f(x) \in A) \ .$$

We shall call sets A for which such p-time function f exists **feasibly infinite**. This remains true (by Buss's theorem) if $T_{\rm PV}$ is augmented by S^1_2. If $T_{\rm PV}$ is extended by some stronger bounded arithmetic theory then Inf_A will be witnessed by a specific $\mathcal{NP}$ search problem attached to the theory. For example, if we add to $T_{\rm PV}$ induction axioms for $\mathcal{NP}$ sets (theory T^1_2) then Inf_A is witnessed by a PLS problem (by the Buss-K.theorem [4]). The reader can find the bounded arithmetic background in [13].

It is easy to see that Problem 5.2 has the affirmative answer for feasibly infinite $\mathcal{NP}$ sets. Namely applying function $f(x)$ to $x := 1^{(n)}$ produces $y := f(x) \in A$ with $|y| > n$ but $Kt(y) \leq O(\log n)$. For the conjecture we need to work a bit.

Theorem 7.1

Assume hypothesis (H) from Section 2. Then Conjecture 1.1 holds relative to the class of feasibly infinite $\mathcal{NP}$ sets: there is a generator g whose range intersects every feasibly infinite $\mathcal{NP}$ set.

Proof :

The proof is a special case of the construction from [23]. We shall show that generator $\mathbf{tt}_{s,k}$ with $s = s(k) := 2^{k/2}$ satisfies the statement.

Let A be a feasibly infinite $\mathcal{NP}$ set as it is witnessed by a p-time function f. Let $d \geq 1$ be the constant from (H) and put $m' := m^{1/(3d)}$ where $m := |f(1^n)|$ and $n >> 1$, and put also $k := \log m$.

Define the function $\hat{f}$ that has $m' + k$ variables and on inputs $1^{m'}$ and $i \in \{0,1\}^k$ computes the i-th bit of $f(1^n)$; it is a p-time function.

Take a circuit $\hat{C}(z,i)$ that computes $\hat{f}$ of size guaranteed by hypothesis (H) and define new circuit C by substituting $1^{m'}$ for z in $\hat{C}$ and leaving only the k variables for bits of i. Note that C has size $O((m'+k)^d) < 2^{k/2}$. Further, by its definition, $\mathbf{tt}_{s,k}(C) = f(1^n)$; i.e. $rng(\mathbf{tt}_{s,k}) \cap A \neq \emptyset$.

q.e.d.

Corollary 7.2 *Assume hypothesis (H) from Section 2. Then Conjecture 1.1 holds in a model $\mathbf{M}$ of T_{PV}: there is a p-time generator g such that for any $\mathcal{NP}$ set A it holds:*

$$\mathbf{M} \models rng(g) \cap A = \emptyset \ \rightarrow \ \neg Inf_A \ .$$

Proof :

Take the function g from Theorem 7.1. The statement $rng(g) \cap A = \emptyset$ is universal for any $A \in \mathcal{NP}$, so it is true in the standard model $\mathbf{N}$ iff it is true in

all models of $T_{\rm PV}$. It thus suffices to show that $T_{\rm PV}$ together with all sentences $\neg Inf_A$ for these sets A is consistent.

Assume not; then the Compactness theorem and the fact that a finite number of A_i are all disjoint from $rng(g)$ iff their union is imply that for some $\mathcal{NP}$ set A such that $rng(g) \cap A = \emptyset$ theory $T_{\rm PV}$ proves Inf_A. But then it is feasibly infinite and that contradicts Theorem 7.1.

q.e.d.

8 Concluding remarks

I think that it is fundamental for the development of the theory to make a progress on the original problem of the unprovability of dWPHP for p-time functions in S^1_2 discussed in the Introduction. For a start we may try to show the unprovability in $T_{\rm PV}$ (or some of its extension as mentioned at the end of Section 2) under a more mainstream hypothesis than is (H). Note that this presumably requires a different function than $\mathbf{tt}_{s,k}$ we used in Section 2: by remarks there the unprovability of dWPHP for this function implies $\mathcal{E} \subseteq Size(2^{o(k)})$ which contradicts the hypothesis that $\mathcal{E} \not\subseteq Size(2^{\epsilon k})$ for some $\epsilon > 0$ which is - in the eyes of many complexity theorists at least - considered plausible.

However, in my view a real progress will result only from unconditional results. For reasons discussed in the next-to-last paragraph of Section 2 to have a chance to succeed we need to leave theory $T_{\rm PV}$ aside and work with theories PV or S^1_2. This implies that an argument cannot rely just on witnessing theorems as they do not change if $T_{\rm PV}$ is added. The problem becomes essentially propositional and it is exactly this what led in [15, 16] to the notions of freeness and pseudo-surjectivity (of generators for EF) mentioned in Section 2: to show that a p-time generator has this property is essentially equivalent to the unprovability of dWPHP for it in PV or S^1_2, respectively (cf. [15, Sec.6] and [16]).

Acknowledgments: I thank Igor C. Oliveira (Warwick U.) and Jan Pich (Oxford U.) for discussions about the topic. I am indebted to the two anonymous referees for their detailed comments and suggestions.

References

[1] 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.

[2] 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.

[3] S. R. Buss, *Bounded Arithmetic*. Naples, Bibliopolis, (1986).

[4] 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.

[5] S. A. Cook and R. A. Reckhow, The relative efficiency of propositional proof systems, *J. Symbolic Logic*, **44(1)**, (1979), pp.36-50.

[6] M. Garlík, Failure of Feasible Disjunction Property for k-DNF Resolution and NP-hardness of Automating It, preprint (2020), ArXiv: 2003.10230.

[7] S. Hirahara, Unexpected Hardness Results for Kolmogorov Complexity Under Uniform Reductions, in: Proc. of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC), June 2020, pp.1038-1051.

[8] R. Ilango, J. Li and R. Williams, Indistinguishability Obfuscation, Range Avoidance, and Bounded Arithmetic, Electronic Colloquium on Computational Complexity, Report No. 38 (2023).

[9] E. Jeřábek, *Weak pigeonhole principle, and randomized computation*, Ph.D. thesis, Charles University, Prague, (2005).

[10] E. Jeřábek, Dual weak pigeonhole principle, Boolean complexity, and derandomization, *Annals of Pure and Applied Logic*, **129**, (2004), pp.1-37.

[11] E. Jeřábek, Approximate counting in bounded arithmetic, *J. of Symbolic Logic*, **72(3)**, (2007), pp.959-993.

[12] E. Jeřábek, Approximate counting by hashing in bounded arithmetic, *J. of Symbolic Logic*, **7493)**, (2009), pp.829-860.

[13] J. Krajíček, *Bounded arithmetic, propositional logic, and complexity theory*, Encyclopedia of Mathematics and Its Applications, Vol. **60**, Cambridge University Press, (1995).

[14] J. Krajíček, On the weak pigeonhole principle, *Fundamenta Mathematicae*, Vol.**170(1-3)**, (2001), pp.123-140.

[15] J. Krajíček, Tautologies from pseudo-random generators, *Bulletin of Symbolic Logic*, **7(2)**, (2001), pp.197-212.

[16] 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.

[17] J. Krajíček, Diagonalization in proof complexity, *Fundamenta Mathematicae*, **182**, (2004), pp.181-192.

[18] 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.

[19] J. Krajíček, *Forcing with random variables and proof complexity*, London Mathematical Society Lecture Note Series, No. **382**, Cambridge University Press, (2011).

[20] J. Krajíček, On the proof complexity of the Nisan-Wigderson generator based on a hard $NP \cap coNP$ function, *J. of Mathematical Logic*, **11(1)**, (2011), pp.11-27.

[21] J. Krajíček, On the computational complexity of finding hard tautologies, *Bulletin of the London Mathematical Society*, **46(1)**, (2014), pp.111-125.

[22] J. Krajíček, *Proof complexity*, Encyclopedia of Mathematics and Its Applications, Vol. **170**, Cambridge University Press, (2019).

[23] 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).

[24] J. Krajíček, Information in propositional proofs and algorithmic proof search, *J. of Symbolic Logic*, vol.87, nb.2, (June 2022), pp.852-869.

[25] 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.

[26] J. Krajíček, P. Pudlák and G. Takeuti, Bounded arithmetic and the polynomial hierarchy, *Annals of Pure and Applied Logic*, **52**, (1991), pp.143–153.

[27] 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.

[28] A. A. Razborov, Pseudorandom generators hard for k-DNF resolution polynomial calculus resolution, *Annals of Mathematics*, **181(2)**, (2015), pp.415-472.

[29] H.Ren, R.Santhanam and Z.Wang, On the Range Avoidance Problem for Circuits, ECCC Report nb.48, (2022).

[30] M. Sipser, *Introduction to the Theory of Computation*, Cengage Learning (3rd ed.), 2005.

[31] A. Woods, *Some problems in logic and number theory, and their connections*, PhD Thesis, U. of Manchester, (1981).