Electronic Colloquium on Computational Complexity, Report No. 190 (2024)

Metamathematics of Resolution Lower Bounds: A TFNP Perspective

Jiawei Li
UT Austin
davidlee@cs.utexas.edu

Yuhao Li
Columbia University
yuhaoli@cs.columbia.edu

Hanlin Ren
University of Oxford
hanlin.ren@cs.ox.ac.uk

Abstract

This paper studies the *refuter* problems, a family of decision-tree TFNP problems capturing the metamathematical difficulty of proving proof complexity lower bounds. Suppose φ is a hard tautology that does not admit any length-s proof in some proof system P. In the corresponding refuter problem, we are given (query access to) a purported length-s proof π in P that claims to have proved φ, and our goal is to find an invalid derivation inside π. As suggested by witnessing theorems in bounded arithmetic, the *computational complexity* of these refuter problems is closely tied to the *metamathematics* of the underlying proof complexity lower bounds.

We focus on refuter problems corresponding to lower bounds for *resolution*, which is arguably the single most studied system in proof complexity. We introduce a new class rwPHP(PLS) in decision-tree TFNP, which can be seen as a randomized version of PLS, and argue that this class effectively captures the metamathematics of proving resolution lower bounds:

- We show that the refuter problems for many resolution size lower bounds fall within rwPHP(PLS), including the classic lower bound by Haken [TCS, 1985] for the pigeonhole principle. In fact, we identify a common technique for proving resolution lower bounds that we call "random restriction + width lower bound", and present strong evidence that resolution lower bounds proved by this technique typically have refuter problems in rwPHP(PLS).
- We then show that the refuter problem for *any* resolution size lower bound is rwPHP(PLS)-hard, thereby demonstrating that the rwPHP(PLS) upper bound mentioned above is tight. It turns out that "rwPHP(PLS)-reasoning" is *necessary* for proving *any* resolution lower bound *at all*!

We view these results as a contribution to the *bounded reverse mathematics* of complexity lower bounds: when interpreted in relativized bounded arithmetic, our results show that the theory $\mathsf{T}^1_2(\alpha) + \mathrm{dwPHP}(\mathsf{PV}(\alpha))$ characterizes the "reasoning power" required to prove (the "easiest") resolution lower bounds. An intriguing corollary of our results is that the combinatorial principle, "the pigeonhole principle requires exponential-size resolution proofs", captures the class of TFNP problems whose totality is provable in $\mathsf{T}^1_2 + \mathrm{dwPHP}(\mathsf{PV})$.

ISSN 1433-8092

Contents

1 Introduction

One of the earliest lower bounds in proof complexity was Haken's landmark result [Hak85] that the pigeonhole principle requires exponential-size proofs in the resolution proof system. Since then, proof complexity has become a vibrant research area with substantial progress in establishing lower bounds for various proof systems, as well as the development of a wide range of lower bound techniques. However, despite decades of efforts, proving nontrivial lower bounds for stronger systems, such as Frege and Extended Frege, remains elusive. It is widely believed that proving lower bounds for Extended Frege is "beyond our current techniques"[1], but what does this even mean? How much, and in which directions, must our techniques expand, to enable us to prove lower bounds for stronger proof systems? These questions call for a study of the *metamathematical* difficulty of proving lower bounds in proof complexity (see, e.g., [PS19, ST21]).

In this paper, we study the metamathematics of resolution lower bounds through the lens of the following type of computational problems. Suppose that we are given a resolution proof Π that claims to have proved the pigeonhole principle, but the length of Π is smaller than the lower bound established in [Hak85]. By Haken's lower bound, Π cannot be a *valid* resolution proof, implying that it must contain an invalid derivation. What is the *computational complexity* of finding such an invalid derivation? We call the following total search problem the "refuter problem"[2] corresponding to Haken's lower bound:

Problem 1.1 (Informal)**.** Given (query access to) a subexponential-size resolution proof Π that claims to be a proof of the pigeonhole principle, find an invalid derivation in Π.

For any proof complexity lower bound stated as "the tautology ϕ requires proof length greater than s in the proof system P", we can define an associated search problem: Given a purported P-proof Π of length at most s that claims to prove ϕ, find an invalid derivation in Π. With appropriate formalization (see Section 1.1), these refuter problems are NP *search problems* and are *total* if and only if their underlying lower bounds are *true*. Therefore, their computational complexity can be studied through the classical theory of TFNP [MP91].

The starting point of this paper is the following principle: the *metamathematics* of proof complexity lower bounds can, and *should*, be understood through the *computational complexity* of the refuter problems. In subsequent discussions, we justify this principle and provide metamathematical motivations for studying the complexity of refuter problems, such as Problem 1.1.

Bounded reverse mathematics. The definition of Problem 1.1 has its roots in *bounded reverse mathematics* [Coo07, Ngu08]. Reverse mathematics explores, for each mathematical theorem of interest, the minimal theory required to prove it. In bounded reverse mathematics, the theories considered come from *bounded arithmetic*, which (roughly speaking) are logical theories formalizing the idea of "reasoning within a complexity class C". The link between these logical theories and complexity classes makes bounded arithmetic, and hence bounded reverse mathematics, an effective framework for studying the metamathematics of complexity theory.

Indeed, there has been a long history of studying the (un)provability of lower bounds in the context of bounded arithmetic: In 1989, Krajíček and Pudlák investigated the unprovability of proof lower bounds [KP89], while Razborov studied the unprovability of circuit lower bounds in 1995 [Raz95a, Raz95b]. Notably, many lower bounds for weak circuit classes and proof systems can be formalized in weak theories [Raz95a, CP90, MP20], while some strong lower bounds are unprovable within them [KP89, Raz95b, Kra97, Kra11b, Pic15, PS21, LO23, CLO24b].

[1] This belief is partly supported by the intuition that proving strong circuit lower bounds (e.g., $\mathsf{NP} \not\subseteq \mathsf{P}/_{\mathrm{poly}}$) seems to be a prerequisite for proving strong proof complexity lower bounds (e.g., for Extended Frege) [Raz15]. However, formalizing such connections has proven challenging [PS23, AKPS24].

[2] This term is adopted from [CTW23], as will be discussed later.

We take a different perspective from the aforementioned line of work: rather than asking whether lower bounds are provable in certain theories, our goal is to *characterize* the exact reasoning power required to prove these lower bounds. That is, we seek to identify the *minimal* theory $\mathcal{T}$ that can prove the given lower bound and to establish the minimality of $\mathcal{T}$ by showing that the axioms used in the proof are indeed *necessary*. The necessity of axioms, i.e., deriving the axiom back from the theorem, is called a *reversal* in reverse mathematics.

Example 1.2. Recently, Chen, Li, and Oliveira [CLO24a] presented several notable reversals related to complexity lower bounds. In their work, they establish that variants of weak pigeonhole principles are *necessary and sufficient* for proving various classical lower bounds. For instance, the fact that one-tape Turing machines require $\Omega(n^2)$ time to recognize palindromes [Maa84] can be proved using the *weak pigeonhole principle*; Moreover, [CLO24a, Theorem 4.9] demonstrates a reversal, proving that this lower bound is, in fact, *equivalent* to the weak pigeonhole principle. The work in [CLO24a] serves as one of the main inspirations of this paper.

Refuter problems. To investigate the metamathematics of a lower bound statement, we first write down the statement in forall-exists form:

- Circuit lower bounds: Let L be a hard language and s be a size lower bound for L. The lower bound statement expresses that *for every* circuit C of size s, *there exists* an input x such that $L(x) \neq C(x)$.
- Proof lower bounds: Let ϕ be a tautology that is hard for some proof system P, and s be a size lower bound for ϕ. The lower bound statement expresses that *for every* purported P-proof Π of size s, *there exists* an invalid derivation step in Π.

Now it becomes evident that Problem 1.1 is exactly the TFNP problem that "corresponds" to Haken's lower bound [Hak85]. In general, a statement

$$\forall x\, \exists y\ V(x, y) \tag{1}$$

would "correspond" to the search problem of finding a valid y given x such that $V(x, y)$ holds; note that the statement is *true* if and only if the search problem is *total*.

This correspondence can be formally justified by the *witnessing theorems* in bounded arithmetic. A witnessing theorem for a theory $\mathcal{T}$ links it to a syntactic subclass $\mathsf{C}_{\mathcal{T}}$ of TFNP, and the theorem states that if (1) is provable in $\mathcal{T}$, then the corresponding (total) search problem lies in the class $\mathsf{C}_{\mathcal{T}}$.[3] For instance, Buss's witnessing theorem [Bus85] states that if (1) is provable in S^1_2, then the corresponding total search problem can be solved in polynomial time. Moreover, Buss and Krajíček [BK94] showed that if (1) is provable in T^1_2, then the corresponding total search problem is solvable in PLS (polynomial local search).

Search problems related to circuit lower bounds have already been studied in the literature [GST07, Pic15, CJSW24, Kor22, CTW23, CLO24a] and are termed "refuter problems" in [CTW23]. We adopt this terminology and refer to the search problems associated with proof lower bounds as "refuter problems" as well.[4]

Total search problems in NP. The above discussion suggests that the metamathematics of lower bounds can be understood through the computational complexity of their refuter problems. Since these problems are total search problems in NP (as long as the lower bounds are true), it is natural to adopt the methodology of TFNP while studying their complexity.

[3] This requires (1) to be a "$\forall\Sigma^b_1$-sentence", meaning that $|x|$ and $|y|$ are polynomially related and $V(x, y)$ is a deterministic polynomial-time relation.

[4] In fact, [CTW23] called these problems "*refutation* problems". We choose to use "*refuter* problems" to avoid confusion with the term "refutation" in proof complexity, which usually refers to a proof showing that a formula is unsatisfiable.

What is the "methodology of TFNP"? Since the seminal work of Megiddo and Papadimitriou [MP91], problems in TFNP have been categorized based on their *proof of totality*. For instance, the class PLS captures NP search problems whose totality is provable from the principle "every DAG has a sink" [JPY88], while the class PPAD captures problems whose totality is provable from "every DAG with an unbalanced node has another one" [Pap94]. Moreover, *completeness* results play the same role as reversals in bounded reverse mathematics. For example, a pivotal result in this direction is the PPAD-completeness of finding a Nash equilibrium in two-player games [CDT09, DGP09]. This result carries an intriguing metamathematical interpretation: Topological arguments (specifically, Brouwer's fixed point theorem [Bro11]) or methods akin to it are *unavoidable* for proving the existence of Nash equilibrium [Nas51], which stands in stark contrast to the linear programming duality methods used for zero-sum games [VS23].

The attentive reader may have already noticed that the above methodology shares a close resemblance to (bounded) reverse mathematics. This similarity can indeed be formally justified by the witnessing theorems mentioned earlier. (Another formal justification is that provability in (universal variants of) bounded arithmetic is equivalent to reducibility in TFNP; see, e.g., [Mül21, Proposition 3.4].) While reading this paper, it is useful to remember that all TFNP results established here can be translated into results in bounded arithmetic and vice versa, conveying the same underlying conceptual message.

This paper. Strongly inspired by the recent work on refuter problems related to *circuit lower bounds* [CJSW24, Kor22, CTW23, CLO24a], we propose investigating refuter problems associated with *proof lower bounds* and studying their complexity in TFNP. As an initial step of this research program, we conduct a case study on the classical *resolution* proof system [Bla37, Rob65].

Remark 1 (Why resolution?). In our view, there are at least two reasons why resolution serves as a suitable "first step" for studying the metamathematics of proof lower bounds:

(i) First, resolution is a well-studied proof system, largely due to its fundamental connections to SAT-solving and automated theorem provers [DP60, DLL62]. Krajíček even estimates that "there are perhaps more papers published about proof complexity of resolution than about all remaining proof complexity topics combined" [Kra19, Chapter 13].

(ii) Second, significant progress has already been made in proving lower bounds against resolution [Hak85, Urq87, CS88, BP96, BW01], suggesting that investigating the metamathematics of resolution lower bounds is a feasible endeavor.

Our results confirm the intuition stated in Item (ii). We introduce a new syntactic subclass of TFNP, denoted as $\mathrm{rwPHP}(\mathsf{PLS})$, and show that:

Theorem 1.3 (Main Result; Informal). *Problem 1.1 is* $\mathrm{rwPHP}(\mathsf{PLS})$*-complete.*

Theorem 1.3 can also be interpreted as conservativeness results showing that a certain fragment of bounded arithmetic "captures" the complexity of proving Haken's lower bounds; see Corollary 4.7 for details.

In fact, our results are more comprehensive than stated in Theorem 1.3 and, in our view, strongly support the claim that $\mathrm{rwPHP}(\mathsf{PLS})$ captures the metamathematics of proving resolution lower bounds:

- First, we investigate several resolution lower bounds proven in the literature, including those against the pigeonhole principle [Hak85, BP96], Tseitin tautologies [Urq87, Sch97], random CNF formulas [CS88], and XOR-lifted formulas [DR03]. We demonstrate that the refuter problems corresponding to all these lower bounds lie within $\mathrm{rwPHP}(\mathsf{PLS})$.[5]

  Notably, all the aforementioned lower bound proofs follow a common proof strategy, which we call "random restrictions + width lower bounds". If a resolution lower bound proof follows this strat-

[5] An interesting exception is the general size-width tradeoff by Ben-Sasson and Wigderson [BW01]; see Section 5.5 for further discussions.

egy, then the corresponding refuter problem is in rwPHP(PLS). This implies that "rwPHP(PLS)-reasoning" is sufficient for implementing one of the most commonly employed strategies for proving resolution lower bounds.

- Complementing the above findings, we prove that for *any* family of hard tautologies for resolution, the corresponding refuter problem is rwPHP(PLS)-hard. Thus, the rwPHP(PLS) upper bound in the previous bullet is indeed tight. Notably, the rwPHP(PLS)-hardness proof in Theorem 1.3 does not rely on the hard tautology being the pigeonhole principle.

  This result carries an intriguing metamathematical implication: "rwPHP(PLS)-reasoning" is necessary for proving *any* resolution lower bound *whatsoever*.

We hope that our results serve as a promising initial step in the bounded reverse mathematics of lower bounds in proof complexity. It is equally intriguing to study (and potentially characterize) the refuter problems for other proof systems such as AC^0-Frege [Ajt94, BIK+92], Cutting Planes [CCT87, Pud97], or Polynomial Calculus [CEI96, Raz98, IPS99], which we leave for future work. An essential goal of this line of research is to understand the complexity of refuter problems for strong proof systems for which we do not know how to prove lower bounds[6]: for which syntactic subclass $\mathcal{P} \subseteq \mathsf{TFNP}$ is "$\mathcal{P}$-reasoning" necessary for proving *any* lower bound for, for example, the $\mathsf{AC}^0[2]$-Frege system?

1.1 Our Settings

Before explaining our results, we first discuss the setting of (decision tree) TFNP and (relativized) bounded arithmetic in which our results take place. In fact, this paper is written with primarily lower bound provers and the TFNP community as target audiences in mind, but the theorems, proofs, and perspectives draw heavy inspiration from bounded arithmetic. Hence, we will mostly state our results and present our proofs in the terminology of TFNP; sometimes after describing a result in TFNP, we will also describe its bounded arithmetic analog. The TFNP parts should be self-contained and require little background from bounded arithmetic. However, we stress that TFNP is just a different language for describing bounded reverse mathematics (over $\forall\Sigma_1^b$-sentences).

We consider TFNP problems in the *decision tree* model ($\mathsf{TFNP}^{\mathsf{dt}}$); this model is sometimes called "type-2 TFNP problems" [BCE+98] when the decision trees are uniform. In this model, we are given an input x of length N and we think of *decision trees of* $\mathrm{polylog}(N)$ *depth* as "efficient". Each possible solution o can be represented by $\mathrm{polylog}(N)$ bits, and there is an efficient procedure $\phi(x, o)$ that verifies whether o is a valid solution for x. (That is, given the purported solution o, $\phi(x, o)$ makes only $\mathrm{polylog}(N)$ queries to x.) The goal is, of course, to find a solution o such that $\phi(x, o)$ holds.

$\mathsf{TFNP}^{\mathsf{dt}}$ corresponds to *relativized* bounded arithmetic where a new predicate α is added into the language. The predicate α is intuitively treated as an oracle (or an exponentially-long input). For example, $\mathsf{PV}(\alpha)$ captures reasoning using P^α-concepts, i.e., *uniform and efficient decision trees over* α.

Remark 2 (Type-1 vs. Type-2 TFNP Problems). In the literature, it is common to define a type-1 TFNP problem in terms of *succinct encodings* of exponentially large objects. For example, a possible definition of a PLS-complete problem is as follows: Given a "neighborhood" circuit $C : \{0,1\}^n \to \{0,1\}^{\mathrm{poly}(n)}$ and a "potential function" circuit $V : \{0,1\}^n \to \{0,1\}^{\mathrm{poly}(n)}$ that together encode a DAG on 2^n nodes, and also an active node (i.e., a node with non-zero out-degree), find a sink of this graph (i.e., a node with non-zero in-degree and zero out-degree). In contrast, the $\mathsf{TFNP}^{\mathsf{dt}}$ / type-2 TFNP problems that we consider simply treat C and V as oracles.

[6]Note that to *characterize* the refuter problems for a proof system $\mathcal{P}$, it is necessary to have proven some lower bounds for $\mathcal{P}$; thus, we can only *speculate* on the complexity of refuter problems for strong proof systems $\mathcal{P}$. Nevertheless, our methodologies might potentially shed light on the metamathematical challenges of proving lower bounds, by providing *hardness* results for such refuter problems.

Any separation of type-2 TFNP problems implies a separation of type-1 TFNP problems *in a relativized world* [BCE+98]. For example, $\mathsf{PLS}^{\mathsf{dt}} \not\subseteq \mathsf{PPA}^{\mathsf{dt}}$ implies an oracle O under which $\mathsf{PLS}^{O} \not\subseteq \mathsf{PPA}^{O}$.

1.1.1 Refuter Problems for Resolution Lower Bounds

This subsection formalizes the refuter problem for resolution lower bounds as a $\mathsf{TFNP}^{\mathsf{dt}}$ problem. We assume familiarity with the resolution proof system. In resolution, every line is a *clause* (i.e., the disjunction of literals) and the only inference rule is the *resolution rule*:

$$\frac{C \vee \ell \qquad D \vee \overline{\ell}}{C \vee D},$$

where C, D are clauses and ℓ is a literal. Sometimes, we will also allow the *weakening* rule that replaces a clause with a consequence of it:

$$\frac{C}{C \vee D}.$$

The *size* of a resolution proof is the number of lines (i.e., clauses) in it. The *width* of a resolution proof is the maximum width of any clause in it, where the *width* of a clause is the number of literals in the clause. Basics about resolution can be found in any textbook on proof complexity, e.g., [Kra19, Section 5].

Size lower bounds for resolution. Let F be a tautology[7] that is *exponentially*-hard for resolution. For example, take F to be the pigeonhole principle which does not have c^n-size resolution proofs for some absolute constant $c > 1$ [Hak85]. The refuter problem, which we denote as

$$\textsc{Refuter}(s(F \vdash_{\mathsf{Res}} \bot) \leq c^n),$$

is defined as follows. The input Π is a purported length-c^n resolution proof of F represented as a list of c^n *nodes*, where each node consists of a clause in the resolution proof and the predecessors of this clause. (For example, if the clause in node i is resolved from the clauses in node j and node k, then the predecessor information would contain two integers (j, k).) A *valid solution* would be the index of any node $i \in [c^n]$ whose derivation is illegal: denoting C_i the clause in node i, then there do not exist clauses C, D and a literal ℓ such that

$$C_i = C \vee D, C_j = C \vee \ell, C_k = D \vee \overline{\ell}.$$

A more formal definition can be found in Section 2.4.

By Haken's lower bound mentioned above [Hak85], every purported resolution proof of length c^n must contain an illegal derivation, thus the above problem is *total*. Let $N := c^n \mathrm{poly}(n)$ denote the bit-length of the input resolution proof, then each node can be described in $\mathrm{poly}(n) \leq \mathrm{polylog}(N)$ bits, hence there is an efficient decision tree that verifies whether a node i is illegal and the above refuter problem is indeed in $\mathsf{TFNP}^{\mathsf{dt}}$.

We can also formalize resolution lower bounds in relativized bounded arithmetic as follows. We add a new symbol α into our language that encodes a length-c^n resolution proof, i.e., for each $i \in [c^n]$, $\alpha(i, \cdot)$ provides information regarding the i-th node of the proof. Let $\mathrm{pf}_F(n, \alpha)$ denote the $\Pi_1^b(\alpha)$-statement expressing that "α encodes a length-c^n resolution proof for F," where F is a hard tautology without such resolution proofs.[8] Note that $\mathrm{pf}_F(n, \alpha)$ is indeed $\Pi_1^b(\alpha)$ since it expresses that *for every* $i \in [c^n]$, the i-th

[7] A DNF D is a *tautology* if and only if the corresponding CNF $\neg D$ is a *contradiction*. A *proof* of D being a tautology is a *refutation* of $\neg D$ being a contradiction. For convenience, we will use the terms "tautology/proof" and "contradiction/refutation" interchangeably.

[8] As a technical detail, we can also allow α to take *parameters* $\vec{z}$ that can be thought of as non-uniformity. That is, for each $i \in [c^n]$, $\alpha(\vec{z}, i, \cdot)$ provides information regarding the i-th node of the proof. We consider the sentence $\mathrm{pf}_F(n, \vec{z}, \alpha)$ which expresses that the proof encoded by $\alpha(\vec{z}, \cdot, \cdot)$ is a valid length-c^n resolution proof for F. To see how the power of, e.g., the dual weak pigeonhole principle with and without parameters can differ, the reader is referred to discussions in [ILW23, Section 4.3].

step of α is correct. The sentence[9]

$$\forall n \in \mathsf{Log}\ \neg \mathrm{pf}_F(n, \alpha)$$

expresses the totality of the refuter problem as defined above; the provability of this sentence in relativized bounded arithmetic corresponds to the complexity of the refuter problem in $\mathsf{TFNP}^{\mathsf{dt}}$.

Width lower bounds for resolution. In this paper, we also study the refuter problems corresponding to *width* lower bounds for resolution. Let F be a tautology without width-w_F resolution proofs, the refuter problem for this width lower bound would be denoted as

$$\textsc{Refuter}(w(F \vdash_{\mathsf{Res}} \bot) \leq w_F).$$

The formalization of width lower bounds is essentially the same as that of size lower bounds, with the only difference that we now impose that every clause in the input resolution proof contains at most w_F literals. This can be done *syntactically* by only allocating w_F literals to each node.

Remark 3 (Further motivations for refutation of width lower bounds). Besides being interesting on their own, the complexity of refuting width lower bounds also serves as a stepping stone for understanding the complexity of refuting *size* lower bounds.

Although we have a fairly good understanding of resolution nowadays, size lower bounds for resolution have been an important open problem in history — in fact, they are milestone achievements in proof complexity. Haken's lower bound [Hak85] for the pigeonhole principle was a breakthrough at its time. But what is the underlying principle for this breakthrough lower bound? Does it correspond to any classical TFNP class such as PPP, PLS, PPAD, PPA, or CLS? Towards its answer, it would be beneficial to dig into the *proofs* of the resolution size lower bounds for PHP.

Haken's original paper [Hak85] employed a "bottleneck counting" argument and the proof was quite involved. Beame and Pitassi [BP96] later introduced a new, simpler proof that elegantly *reduced the size lower bound to a width lower bound* for (a monotone version of) resolution (see Section 3.1 for more details). This size-width connection is not unique for PHP. The groundbreaking paper by Ben-Sasson and Wigderson [BW01] established a generic size-width trade-off for resolution, which had a significant impact on the proof complexity community. Today, studying size-width trade-offs for various proof systems has become standard practice (see e.g. [CEI96, PS12, AH19, Sok20]).

Returning to PHP in the context of resolution, we know that reasoning about size lower bounds can, in some sense, be reduced to reasoning about width lower bounds (we will formalize this very soon!). Thus, understanding the refuter problem for width lower bounds seems like a prerequisite to understanding that for size lower bounds.

1.1.2 Retraction Weak Pigeonhole Principles

This paper demonstrates that the complexity of refuter problems corresponding to resolution size lower bounds is tightly linked to the new complexity class rwPHP(PLS). Therefore, we need to introduce this class before describing our results.

Here, "rwPHP" stands for the *retraction weak pigeonhole principle*:

> For any two functions $f : [N] \to [2N]$ and $g : [2N] \to [N]$, the function $f \circ g : [2N] \to [2N]$ cannot be the identity function.

The term "retraction", borrowed from category theory [Jeř07b], means that the principle concerns a pair of functions f, g where g is a "retraction"; the term "weak" indicates that the domain of g ($[2N]$) is *much* larger than its range ($[N]$). This principle, along with other variants of weak pigeonhole principles, is widely studied in the context of bounded arithmetic [PWW88, Kra01, MPW02, Tha02, Ats03, Kra04, Jeř04, Jeř07b, CLO24a] and total search problems [KKMP21, Kor21, Kor22]; it is sometimes also called

[9] Roughly speaking, the notation $n \in \mathsf{Log}$ means that n is the length of some number, thus allowing one to reason about integers of magnitude $2^{\mathrm{poly}(n)}$. In our particular case, it allows the length of the purported proof to be exponential in n. This is a standard notation in bounded arithmetic.

the "witnessing weak pigeonhole principle (WPHPWIT)" [Jeř07a, CLO24a] and "Lossy-Code" [Kor22]. Clearly, rwPHP corresponds to a $\mathsf{TFNP}^{\mathsf{dt}}$ problem: given (query access to) two functions $f : [N] \to [2N]$ and $g : [2N] \to [N]$, find an input $y \in [2N]$ such that $f(g(y)) \neq y$.

Let $\mathcal{P}$ be a problem in $\mathsf{TFNP}^{\mathsf{dt}}$, then one can define a class rwPHP($\mathcal{P}$) capturing the retraction weak pigeonhole principle where, informally speaking, the retraction function g can be computed in $\mathcal{P}$. In the decision tree model, the inputs of rwPHP($\mathcal{P}$) consist of:

1. (the evaluation table of) a function $f : [N] \to [2N]$, and
2. $2N$ instances of $\mathcal{P}$, denoted as $\{I_y\}_{y\in[2N]}$, where each valid solution *ans* of each I_y is marked with an integer $g_{y,ans} \in [N]$.

The goal is to find an integer $y \in [2N]$ along with a solution *ans* of I_y such that $f(g_{y,ans}) \neq y$. It is not hard to see that if $\mathcal{P} \in \mathsf{TFNP}^{\mathsf{dt}}$ then rwPHP($\mathcal{P}$) $\in \mathsf{TFNP}^{\mathsf{dt}}$ (Fact 2.8). Furthermore, rwPHP($\mathcal{P}$) can be solved by a simple *randomized* algorithm given oracle access to any solver of $\mathcal{P}$.

The class rwPHP(PLS) is defined as the problems reducible to rwPHP($\mathcal{P}$) for a PLS-complete problem $\mathcal{P}$. It can be shown that rwPHP(PLS) does not depend on the exact choice of the PLS-complete problem $\mathcal{P}$ (Fact 2.10).

Witnessing for $\mathsf{T}^1_2 + \mathrm{dwPHP}(\mathsf{PV})$. Although rwPHP(PLS) seems to be new to the TFNP community, it already appeared implicitly in the literature of bounded arithmetic. This class captures the TFNP problems whose totality is provable in $\mathsf{T}^1_2 + \mathrm{dwPHP}(\mathsf{PV})$. In other words, rwPHP(PLS) corresponds to the *witnessing theorem* for $\mathsf{T}^1_2 + \mathrm{dwPHP}(\mathsf{PV})$ (just like how PLS corresponds to a witnessing theorem for T^1_2 [BK94]). This was noticed in [BKT14] where they showed every $\forall\Sigma^b_1$-consequence of $\mathsf{T}^1_2 + \mathrm{dwPHP}(\mathsf{PV})$ *randomly* reduces to PLS; in fact, the same argument implies a deterministic reduction to rwPHP(PLS).

Remark 4 (How Strong is rwPHP(PLS)?).

Since rwPHP(PLS) can be seen as a randomized version of PLS (where the guarantee that "most randomness are good" is provided by the dual weak pigeonhole principle), its position in the $\mathsf{TFNP}^{\mathsf{dt}}$ hierarchy is roughly the same as, but slightly higher than PLS. In particular, in the decision tree setting, it follows from the previous separations ($\mathsf{PLS} \not\subseteq \mathsf{PPP}$ [GHJ+22] and $\mathsf{PLS} \not\subseteq \mathsf{PPA}$ [BM04]) that rwPHP(PLS) is contained in neither PPP nor PPA. Note that there is already a decision tree separation between PLS and the $\mathsf{TFNP}^{\mathsf{dt}}$ problem corresponding to rwPHP (which follows from a resolution width lower bound for rwPHP [PT19, Proposition 3.4]), hence in the decision tree setting, rwPHP(PLS) strictly contains PLS.

We also note that $\mathsf{T}^1_2(\alpha) + \mathrm{dwPHP}(\mathsf{PV}(\alpha))$ is a relatively weak theory in the realm of relativized bounded arithmetic.[a] This theory is a subtheory of both $\mathsf{T}^2_2(\alpha)$ and Jeřábek's (stronger) fragment for approximate counting $\mathsf{APC}_2(\alpha)$ [Jeř09]. It is also "weak" in the sense that unconditional unprovability results are known: it cannot prove the ordering principle [AT14] and the pigeonhole principle [PT19].

[a] The reader might have encountered claims in the literature that even weaker theories such as S^1_2 or APC_1 are "strong", so it might be confusing for a reader unfamiliar with bounded arithmetic that we are claiming $\mathsf{T}^1_2(\alpha) + \mathrm{dwPHP}(\mathsf{PV}(\alpha))$ as a "weak" theory. The reason is *relativization*: In our formalization, the purported resolution proof α has *exponential* size, and we are only allowed to reason about objects in PH (think of AC^0 circuits over α). This is much weaker than the setting where the proof α has *polynomial* size and we are allowed to reason about polynomial-time concepts. This is roughly analogous to classifying the circuit class AC^0 (i.e., relativized PH) as "weak" and $\mathsf{P}/_{\mathrm{poly}}$ as "strong".

1.2 Our Results

Our main results can be categorized into three parts: (1) bounded reverse mathematics (TFNP characterizations) for (several) resolution width lower bounds; (2) bounded reverse mathematics (TFNP characterizations) for (several) resolution size lower bounds; and (3) further applications in TFNP and proof complexity. We will describe the results related to width lower bounds first in Section 1.2.1, not only because they serve as prerequisites for the results regarding size lower bounds (discussed in Section 1.2.2), but also because the techniques therein find additional applications in TFNP and proof complexity (detailed in Section 1.2.3).

Readers interested specifically in the refuter problems corresponding to the pigeonhole principle (i.e., Problem 1.1) will find this paper organized to their advantage. The main body of the paper is structured so that the initial sections primarily focus on presenting the complexities of refuter problems of $\mathrm{PHP}_{(n+1)\to n}$. Specifically, Section 2 contains the minimal preliminaries and definitions, and Section 3 and Section 4 present the proof of Theorem 1.3. It is worth noting, however, that our results extend to many other formulas (XOR-lifted formulas, Tseitin formulas, and random k-CNFs), as shown in Section 5.

1.2.1 Bounded Reverse Mathematics for Resolution Width Lower Bounds

The main message in this subsection is that the refuter problems corresponding to resolution width lower bounds are complete for the well-studied class PLS, the first syntactic subclass of TFNP introduced in the literature [JPY88].

We begin with the results related to the pigeonhole principle. The attentive reader may notice a subtle issue when formulating the refuter problem of width lower bound: $\mathrm{PHP}_{(n+1)\to n}$ already contains an axiom with width n, and the width lower bound for proving it is n as well. Thus, the corresponding width refuter problem becomes trivial. To address this, we instead consider the width refuter problem for a *constant-width analog* of $\mathrm{PHP}_{(n+1)\to n}$, called $\mathrm{EPHP}_{(n+1)\to n}$, which has constant-width axioms and an $n/3$ width lower bound as shown in [BW01]. We characterize the complexity of its corresponding refuter problem:

Theorem 4.2. *$\textsc{Refuter}(w(\mathrm{EPHP} \vdash_{\mathsf{Res}} \bot) < n/3)$ is PLS-complete.*

A similar PLS-completeness result also holds for Tseitin formulas (on expander graphs), where $e(G)$ below is the *expansion* parameter of the graph G (Definition 5.6).

Theorem 5.9. *$\textsc{Refuter}(w(\mathrm{Tseitin} \vdash_{\mathsf{Res}} \bot) < e(G))$ is PLS-complete.*

The techniques used in these results will be further extended to the refuter problems corresponding to black-box TFNP separations, specifically $\mathsf{PLS} \not\subseteq \mathsf{PPP}$ and $\mathsf{PLS} \not\subseteq \mathsf{PPA}$, as described in Theorem 6.11 below.

To tackle Problem 1.1 though, we have to delve into the proofs of the exponential (size) lower bound. A *monotonized* version of the *width* lower bound plays a crucial role in the simplified proof by Beame and Pitassi [BP96]. In particular, they show that any resolution refutation of $\mathrm{PHP}_{(n+1)\to n}$ contains a clause C with "monotone width" of at least $2n^2/9$ (see Section 3.1). We similarly characterize the complexity of its corresponding refuter problem (where the subscript mono denotes the monotone analog of the width refuter problem; the formal definition is provided in Section 3.1):

Theorem 4.3. *$\textsc{Refuter}(w_{\mathsf{mono}}(\mathrm{PHP}_{(n+1)\to n} \vdash_{\mathsf{Res}} \bot) < 2n^2/9)$ is PLS-complete.*

Unsurprisingly, this result serves as a key step toward addressing the size refuter problem for the pigeonhole principle, which will be discussed in the next subsection.

The PLS-hardness parts of all three results above stem from a *unified and simple proof*, detailed in Theorem 4.1. Conversely, the PLS-membership of these refuter problems is established by carefully analyzing the proofs in [BP96, BW01] and demonstrating that "PLS-reasoning" suffices to prove these lower bounds. (In fact, these proofs can be formalized in the theory $\mathsf{T}^1_2(\alpha)$, and the PLS-membership follows directly from the witnessing theorem in [BK94].)

A non-uniform universal PLS-membership. Finally, we establish a *universal* PLS-membership result with respect to *non-uniform* decision tree reductions: for any resolution width lower bound against every unsatisfiable CNF, *as long as the lower bound is correct*, the corresponding refuter problem can be reduced to PLS under *non-uniform* decision tree reductions.

Theorem 5.1. *Let $\mathcal{F}$ be any (possibly non-uniform) family of unsatisfiable CNFs with polynomially many clauses, and let w_0 be any valid resolution width lower bound for $\mathcal{F}$. Then there exists a (non-uniform) decision-tree reduction from* $\textsc{Refuter}(w(\mathcal{F} \vdash_{\mathsf{Res}} \perp) < w_0)$ *to* PLS.

Both the formulation and proof of this result inherently require non-uniformity for at least two reasons: (1) it is computationally hard to check whether an arbitrarily given CNF is unsatisfiable, and (2) even assuming that the given CNF is unsatisfiable, it is hard to calculate the resolution width lower bound. See Section 5.1 for further discussion.

Remark 5 (Uniform vs. non-uniform reductions). Note that if one only cares about non-uniform reductions, then (the PLS-membership parts of) Theorem 4.2 and Theorem 5.9 are merely special cases of Theorem 5.1. Nevertheless, we believe that the uniform PLS-membership results in Theorem 4.2 and Theorem 5.9 are informative, as they actually show that the corresponding lower bounds can be formalized in $\mathsf{T}^1_2(\alpha)$; in fact, the *code* of the Turing machine that implementing the uniform reduction to PLS effectively acts as a *proof* of the width lower bound using a *local search* argument. They are also crucial for the uniform rwPHP(PLS)-memberships for the size refuter problems. However, the decision tree reduction in Theorem 5.1 seems to require $\exp(n)$ bits of non-uniformity, making it *highly* non-uniform.

On the other hand, the non-uniform reduction in Theorem 5.1 implies an intriguing proof complexity upper bound: *Small-width resolution can prove width lower bounds for resolution itself*! (See Section 1.2.3 for more details.) Uniformity is not required for this application, allowing us to derive more proof complexity upper bounds using Theorem 5.1: *every* resolution width lower bound *that is correct* can be proved in low-width resolution. (The size lower bound analog of Theorem 5.1 remains unknown, hence we can only show proof complexity upper bounds for tautologies encoding *specific* resolution size lower bounds.)

1.2.2 Bounded Reverse Mathematics for Resolution Size Lower Bounds

Our main message in this subsection is that the refuter problems corresponding to many resolution size lower bounds are complete for rwPHP(PLS), the TFNP subclass introduced in Section 1.1.2. Indeed, the theorems presented in this subsection suggest that rwPHP(PLS) captures the complexity of proving *the easiest-to-prove* size lower bounds for resolution. Our workflow is the same as before:

- First, we show that for many notable resolution size lower bounds proven in the literature, the corresponding refuter problems reduce to rwPHP(PLS). Specifically, we identify a common technique for proving resolution size lower bounds, which we call "random restriction + width lower bounds", and demonstrate that if a resolution size lower bound can be proven using it, then the corresponding refuter problem generally falls within rwPHP(PLS).
- Next, we present a *unified* rwPHP(PLS)-hardness result: the refuter problems for resolution size lower bounds are rwPHP(PLS)-hard, and the hardness proof *does not* depend on the hard tautology considered. Thus, we conclude the rwPHP(PLS)-completeness of many refuter problems for resolution size lower bounds.

The rwPHP(PLS)-hardness of size lower bound refuters turns out to be more challenging than the PLS-hardness of width lower bound refuters, as discussed in Section 1.4.2.

We begin by showing that Problem 1.1 reduces to rwPHP(PLS):

Theorem 1.4 (Informal version of Theorem 3.16)**.** *There exists an absolute constant $c > 1$ and an efficient decision-tree reduction from the problem* $\textsc{Refuter}(s(\mathrm{PHP}_{(n+1)\to n} \vdash_{\mathsf{Res}} \perp) \le c^n)$ *to* rwPHP(PLS).

In fact, we show that $\mathsf{T}^1_2(\alpha) + \mathrm{dwPHP}(\mathsf{PV}(\alpha))$ proves the sentence

$$\forall n \in \mathsf{Log}\ \neg \mathrm{pf}_{\mathrm{PHP}}(n, \alpha),$$

i.e., α is not a length-c^n resolution proof for PHP, by formalizing the classical proofs in [Hak85, CP90, BP96]; Theorem 3.16 then follows from the witnessing theorem for $\mathsf{T}^1_2(\alpha) + \mathrm{dwPHP}(\mathsf{PV}(\alpha))$. In the technical overview (Section 1.4.1) and the main proof (Section 3.2), we present the reduction from the refuter problem $\textsc{Refuter}(s(\mathrm{PHP} \vdash_{\mathsf{Res}} \bot) \le c^n)$ to rwPHP(PLS) directly, without relying on witnessing theorems.

It turns out that a large variety of resolution size lower bounds can be proven using the paradigm of "random restriction + width lower bounds," including those for XOR-lifted formulas [DR03], Tseitin formulas [Urq87, Sch97], and random CNFs [CS88]. We show that all these lower bounds have corresponding refuter problems in rwPHP(PLS) (see Theorem 5.4, Theorem 5.12, and Theorem 5.13, respectively). These results provide strong evidence that rwPHP(PLS) (or $\mathsf{T}^1_2(\alpha) + \mathrm{dwPHP}(\mathsf{PV}(\alpha))$) captures the "complexity" of this popular proof technique for resolution lower bounds.

We complement the above results by showing that for *every* unsatisfiable family of CNFs $\{F_n\}$ that requires resolution size greater than $s_F(n)$, the corresponding refuter problem $\textsc{Refuter}(s(F_n \vdash_{\mathsf{Res}} \bot) \le s_F(n))$ is hard for rwPHP(PLS).

Theorem 1.5 (Informal version of Theorem 4.4)**.** *For every unsatisfiable family of CNF formulas $\{F_n\}$ and parameter $s_F(n)$ such that every resolution refutation of F_n requires more than $s_F(n)$ clauses, there exists a decision tree reduction of depth* $\mathrm{poly}(n)$ *from* rwPHP(PLS) *to* $\textsc{Refuter}(s(F_n \vdash_{\mathsf{Res}} \bot) \le s_F(n))$.[10]

Note that Theorem 1.5 holds for *every* hard tautology, whereas the rwPHP(PLS) upper bounds such as Theorem 1.4 are only known to hold for some natural families of hard tautologies. For these natural tautologies, we establish a *reversal* in the bounded reverse mathematics of proof complexity lower bounds: The power of "rwPHP(PLS)-reasoning" is *sufficient* for implementing a popular proof strategy that can prove all these resolution lower bounds and, at the same time, is *necessary* for proving *any* resolution lower bound *whatsoever*!

Remark 6. We also note that Theorem 1.5 requires decision tree depth $\mathrm{poly}(n)$ regardless of s_F, and is thus only considered "efficient" when $s_F = 2^{n^{\Omega(1)}}$. However, this is merely an artifact of our definition of "efficiency" in the decision tree setting, i.e., if the input length is N, then depth-polylog(N) decision trees are considered "efficient". In fact, even if $s_F = 2^{n^{o(1)}}$, each node in the purported length-s_F resolution proof still requires $\mathrm{poly}(n)$ bits to represent, so it takes $\mathrm{poly}(n)$ query complexity to *verify* a solution of the refuter problem. Therefore, it still makes sense *in the particular setting of refuter problems* to consider a decision tree reduction *efficient* if its query complexity is at most $\mathrm{poly}(n)$. We interpret Theorem 1.5 to mean that "rwPHP(PLS)-reasoning" is necessary for proving *not only subexponential but any moderately large* size lower bound for resolution.

The proof of Theorem 1.5 is heavily inspired by the NP-hardness of automating resolution [AM20] and the exposition of this result in [dRGN+21]. In these proofs, it was crucial to show that resolution cannot prove lower bounds against itself; in particular, [dRGN+21, Section 5] showed that resolution requires a large (block-)width to prove resolution lower bounds. Notably, the proof in [dRGN+21] is by a reduction from rwPHP, i.e., resolution cannot prove lower bounds against itself because resolution cannot prove rwPHP. We strengthen these results by reducing a stronger problem —rwPHP(PLS) instead of rwPHP — to the refuter problems, thereby obtaining a *tight* characterization of these refuter problems.

Finally, our results provide an intriguing characterization of the provably total NP search problems in $\mathsf{T}^1_2 + \mathrm{dwPHP}(\mathsf{PV})$ (see Corollary 4.7). That is:

> Just as "every DAG has a sink" characterizes the $\forall\Sigma^b_1$-consequences of T^1_2 [BK94], "resolution requires $2^{\Omega(n)}$ size to prove PHP" characterizes the $\forall\Sigma^b_1$-consequences of $\mathsf{T}^1_2 + \mathrm{dwPHP}(\mathsf{PV})$.

1.2.3 Applications

Besides being interesting in itself, our study of refuter problems also reveals several new insights into these well-studied proof complexity lower bounds and TFNP separations. More specifically, we translate

[10] This theorem requires a mild technical condition that $s_F(n)$ should be moderately larger than the size of the rwPHP(PLS) instance; see the formal statement in Theorem 4.4 for details.

our results into different languages using the generic connection between $\mathsf{TFNP}^{\mathsf{dt}}$ and proof complexity via the *false clause search* problem (see, e.g., [dRGR22]): For an unsatisfiable CNF $F = C_1 \wedge \cdots \wedge C_M$, the false clause search problem Search(F) is a $\mathsf{TFNP}^{\mathsf{dt}}$ problem where, given oracle access to an input $x \in \{0,1\}^N$, the goal is to find a clause C_i such that $C_i(x) = \mathsf{false}$. Any $\mathsf{TFNP}^{\mathsf{dt}}$ problem can be written as a false clause search problem for a family of low-width CNFs, and vice versa. In particular, a family of unsatisfiable CNFs has low-width resolution refutations if and only if the corresponding false clause search problem reduces to PLS [Raz95b] (see also [Kam19, Section 8.2.2] for an exposition). See Figure 1 for a diagram that summarizes the translations of our main results in different languages.

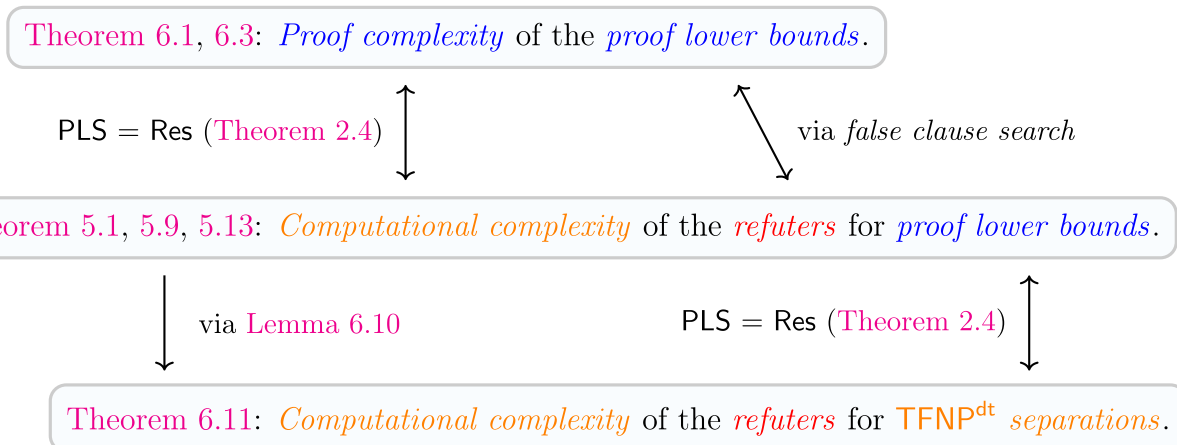


Figure 1: Translations of the main results in different languages. A one-way arrow represents an implication, and a two-way arrow indicates an equivalence.

Proof complexity of proof lower bounds. We first use our results to provide surprisingly efficient proofs for proof complexity lower bounds. Note that a proof complexity lower bound can be expressed by a family of CNFs $\mathcal{F}_{\mathsf{LB}}$ by formulating the corresponding refuter problem as a false clause search problem Search$(\mathcal{F}_{\mathsf{LB}})$ (see, e.g., Section 6.1).

In particular, there exists a family of $\tilde{O}(w)$-width CNFs that encodes a width-w resolution lower bound. Then, since PLS and low-width resolution are equivalent, Corollary 5.2 implies the following *upper bound* on the resolution width required to prove resolution lower bounds.

Theorem 1.6 (Informal version of Theorem 6.1)**.** *Any width-w resolution lower bound can be proved in resolution width $\tilde{O}(w)$.*

We also use our rwPHP(PLS) upper bounds to show that poly(n)-width *random resolution* [BKT14, PT19] can prove exponential-size resolution lower bounds (encoded as poly(n)-width CNFs). In fact, using our results on random k-CNFs (Theorem 5.13), we can show that *most* resolution size lower bounds are provable in low-width random resolution:

Theorem 1.7 (Informal version of Theorem 6.3)**.** *With high probability over a random k-CNF F, the resolution size lower bound $s(F \vdash_{\mathsf{Res}} \bot) > 2^{\Omega(n)}$ can be proved in* random resolution *width of* poly(n).

These results stand in stark contrast with Garlík's result [Gar19] that tautologies encoding any resolution size lower bounds are hard for resolution: We show that either switching to width lower bounds (Theorem 1.6) or considering *random* resolution (Theorem 1.7) makes these lower bound tautologies easy to prove![11]

[11] In our Section 6.1, the lower bound tautologies use *binary encoding*, where (e.g.) the predecessors of every node are encoded by $O(\log N)$ bits. In contrast, Garlík [Gar19] uses *unary encoding* where for every pair of nodes (i, j) (a minor

Complexity of refuting black-box TFNP separations. We also consider the refuter problem for *black-box* TFNP *separations.* Let A, B be two $\mathsf{TFNP}^{\mathsf{dt}}$ classes such that $\mathsf{A} \not\subseteq \mathsf{B}$. Informally, $\mathsf{Ref}(\mathsf{A} \subseteq \mathsf{B})$ is the class of problems reducible to the following kind of "refuter" problems: The input is a purported decision tree reduction from A to B, and the solution is a short witness showing that the reduction is wrong. The refuter problems for $\mathsf{TFNP}^{\mathsf{dt}}$ separations also lie in $\mathsf{TFNP}^{\mathsf{dt}}$, as their totality follows from the correctness of the black-box separation $\mathsf{A} \not\subseteq \mathsf{B}$.

The complexity of such refuter problems measures the strength of the arguments used for black-box separation results. For example, the following corollary conveys a simple but often overlooked fact: *when separating a syntactic* $\mathsf{TFNP}^{\mathsf{dt}}$ *subclass* A *from* B*, it is necessary to incur the totality principle of* B*.*

Corollary 6.9. *For any two* $\mathsf{TFNP}^{\mathsf{dt}}$ *classes* A, B *such that* $\mathsf{A} \not\subseteq \mathsf{B}$*,* $\mathsf{B} \subseteq \mathsf{Ref}(\mathsf{A} \subseteq \mathsf{B})$*.*

Due to the connection between $\mathsf{TFNP}^{\mathsf{dt}}$ and proof complexity, the refuter problem for each black-box TFNP separation naturally aligns with a corresponding refuter problem for a proof complexity lower bound. In particular, we build a uniform reduction from the refuter problems for separations from PLS to the refuter problems for resolution width lower bounds (Lemma 6.10), because showing a $\mathsf{TFNP}^{\mathsf{dt}}$ subclass A is not in PLS is essentially showing a resolution width lower bound for the formula expressing the totality of A.

Note that the false clause search problem for EPHP and Tseitin are in PPP and PPA respectively. Therefore, using our characterization of the resolution width refuter for EPHP (Theorem 3.3) and Tseitin (Theorem 5.9), we conclude that *it is necessary and sufficient to use local search principle to separate* PPP *and* PPA *from* PLS *in the black-box setting.*

Theorem 1.8 (Informal version of Theorem 6.11)**.** $\mathsf{Ref}(\mathsf{PPP} \subseteq \mathsf{PLS}) = \mathsf{Ref}(\mathsf{PPA} \subseteq \mathsf{PLS}) = \mathsf{PLS}$.

1.3 Discussions, Speculations, and Future Directions

This paper initiates a research program that attempts to understand, for every proof system $\mathcal{P}$ of interest, the metamathematics of proving lower bounds against $\mathcal{P}$ through the lens of refuter problems.[12] Our results on resolution suggest that this is a promising direction. There are a plethora of future research directions, both regarding "weak" systems (where we already know strong lower bounds against $\mathcal{P}$-proofs) and "strong" ones (where we are still struggling to prove non-trivial lower bounds against $\mathcal{P}$).

Weak proof systems. It might be feasible to characterize the complexity of refuter problems for weak proof systems. How does the complexity of refuting lower bounds for $\mathcal{P}$ compare with $\mathcal{P}$ itself (or, more precisely, the $\mathsf{TFNP}^{\mathsf{dt}}$ subclass corresponding to $\mathcal{P}$ [BFI23])? In the case that $\mathcal{P}$ is resolution, our work shows that the complexity of refuting *width* lower bounds for $\mathcal{P}$ is exactly $\mathcal{P}$ itself (i.e., PLS), and the complexity of refuting *size* lower bounds is a randomized version of $\mathcal{P}$ (i.e., rwPHP(PLS)). Thus, it seems reasonable to conjecture that for "weak" proof systems, the complexity of proving lower bounds against them is not much higher than themselves.

Moreover, the proof complexity of proof complexity lower bounds is intimately connected to the (non-)automatability of proof systems, see e.g., [AM20, GKMP20, Bel20, dRGN+21, IR22, Gar24, Pap24]. We expect that a thorough understanding of the former would help make progress on the latter as well.

detail is that [Gar19] requires i to be "one level above" j), there is a Boolean variable $x_{i,j}$ indicating whether i is a predecessor of j. As [Gar19] pointed out, a resolution lower bound for the unary-encoded refutation statements implies a similar lower bound for the binary-encoded refutation statements. On the other hand, since we are proving *width* upper bounds and the unary encoding already results in large-width CNFs, we can only afford to use binary encoding (see Remark 8).

[12] We believe the similar research program for circuit lower bounds would also be fruitful, which has already started since [CJSW24, Kor22, CLO24a] if not earlier. We limit our discussions to proof lower bounds here.

Strong proof systems. The situation for strong proof systems seems much more mysterious. For strong proof systems $\mathcal{P}$ (think of $\mathcal{P}$ being Frege or Extended Frege), it is even unclear whether there should be an "easiest-to-prove" lower bound for $\mathcal{P}$ (which would correspond to a syntactic subclass $\mathcal{C}(\mathcal{P}) \subseteq \mathsf{TFNP}^{\mathsf{dt}}$ that characterizes the complexity of proving lower bounds for $\mathcal{P}$). Even if such a $\mathcal{C}(\mathcal{P})$ exists, it is unclear if it is captured within our current landscape of $\mathsf{TFNP}^{\mathsf{dt}}$.[13]

This suggests the following possibility: The reason that we have not been able to prove lower bounds for $\mathcal{P}$ is that $\mathcal{C}(\mathcal{P})$ is a very complicated class, far beyond our current understanding of $\mathsf{TFNP}^{\mathsf{dt}}$ and bounded arithmetic. An even more speculative hypothesis would be that the proof systems $\mathcal{P}$ for which we are able to prove lower bounds are exactly those where $\mathcal{C}(\mathcal{P})$ is not "much" higher than $\mathcal{P}$ themselves. We hope that future work will determine to what extent these hypotheses are correct.

The case of $\mathsf{AC}^0[p]$-Frege (where p is a prime) is of particular interest. Although strong lower bounds for $\mathsf{AC}^0[p]$ *circuits* have been known for decades [Raz87, Smo87], we have not yet succeeded in turning these circuit lower bounds into proof complexity lower bounds against $\mathsf{AC}^0[p]$-Frege (see, e.g., [MP96, BKZ15]). The paper [BIK+97] laid out a research program towards $\mathsf{AC}^0[p]$-Frege lower bounds by studying weaker algebraic proof systems such as the Nullstellensatz [BIK+94] and Polynomial Calculus [CEI96, Raz98]. After a few decades, we have become proficient at proving lower bounds against such algebraic proof systems, but lower bounds against $\mathsf{AC}^0[p]$-Frege remain elusive. Is it because the refuter problems corresponding to $\mathsf{AC}^0[p]$-Frege lower bounds are *fundamentally different* from those for the weaker algebraic proof systems? Does our metamathematical $\mathsf{TFNP}^{\mathsf{dt}}$ perspective bring new insights to this long-standing open question?

1.4 Technical Overview

1.4.1 Refuter Problems in rwPHP(PLS)

In this subsection, we explain how the lower bound proof in [CP90, BP96] yields a reduction from the problem $\textsc{Refuter}(s(\mathrm{PHP}_{(n+1)\to n} \vdash_{\mathsf{Res}} \bot) \le c^n)$ to $\mathrm{rwPHP}(\mathsf{PLS})$. As mentioned before, this is essentially a formalization of the lower bound proof in $\mathsf{T}^1_2(\alpha) + \mathrm{dwPHP}(\mathsf{PV}(\alpha))$, and the reduction follows from the witnessing theorem for this theory. However, this subsection will describe the reduction without invoking the witnessing theorem (nor does the formal proof in Section 3.2 use the witnessing theorem). We hope that by opening up the witnessing theorem as a black box, it would become clearer how each component in the proof corresponds to a component in the reduction to $\mathrm{rwPHP}(\mathsf{PLS})$.

The proof of [CP90, BP96] consists of two components:

- (Random restrictions) First, we carefully design a distribution of random restrictions $\mathcal{R}$ under which the following holds. (1) With high probability over $\rho \leftarrow \mathcal{R}$, any fixed size-$c^n$ resolution proof will simplify to a resolution proof of *width*[14] at most w under ρ (for some parameter w); (2) the pigeonhole principle $\mathrm{PHP}_{(n+1)\to n}$ remains to be the pigeonhole principle (of a slightly smaller size $\mathrm{PHP}_{(n'+1)\to n'}$) under any restriction $\rho \in \mathcal{R}$. Moreover, $\mathcal{R}$ is the uniform distribution over some set of restrictions; we abuse notation and use $\mathcal{R}$ to also denote this set.

- (Width lower bound) Then, we invoke the width lower bound for the pigeonhole principle and show that resolution cannot prove $\mathrm{PHP}_{(n'+1)\to n'}$ in width w.

Given a resolution proof Π of size c^n, the fact that most restrictions simplify Π into a small-width proof can be shown by a *compression argument*: given a clause $C_i \in \Pi$ and a restriction $\rho \in \mathcal{R}$ that does *not* shrink C_i into a clause of width $\le w$, one can describe ρ in ℓ_{comp} bits for some small ℓ_{comp}. In what follows,

[13] Note that the question of where $\mathcal{C}(\mathcal{P})$ sits in the $\mathsf{TFNP}^{\mathsf{dt}}$ hierarchy is merely a restatement of the open problem of determining the proof complexity of proof complexity lower bounds for $\mathcal{P}$. For example, $\mathcal{C}(\mathcal{P})$ is a subclass of PTFNP [GP18a] if and only if Q-EFF (the proof system underlying the definition of PTFNP) can prove lower bounds for $\mathcal{P}$.

[14] In fact, in the case of PHP, we obtain a resolution proof of small *monotone width*. We omit the distinction between width and monotone width in the overview and refer the reader to Section 3 for details.

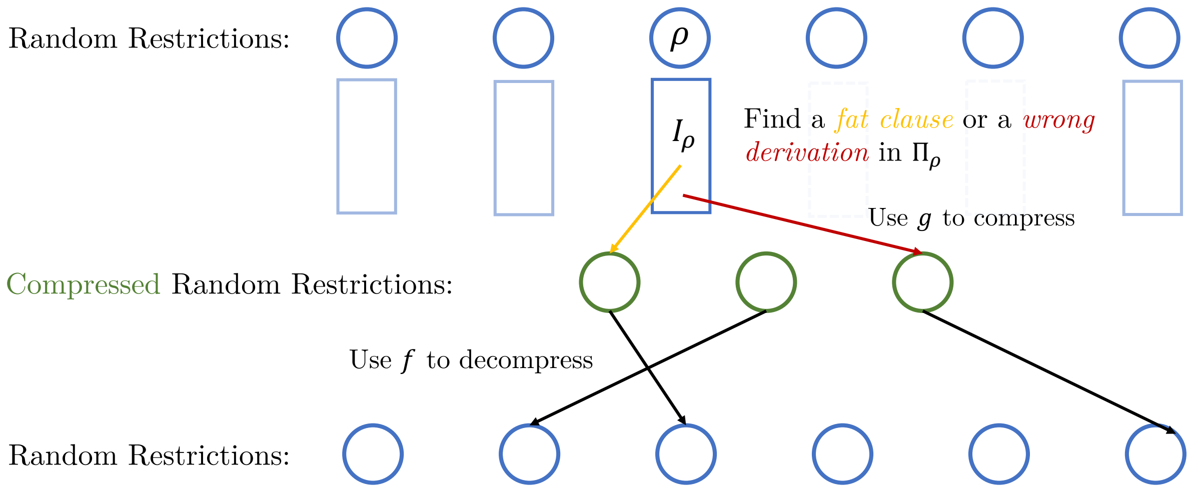


Figure 2: The rwPHP(PLS) instance constructed from $\text{Refuter}(s(\text{PHP}_{(n+1)\to n} \vdash_{\mathsf{Res}} \bot) \le c^n)$.

it suffices if $\ell_{\mathsf{comp}} \le \log|\mathcal{R}| - \log(c^n) - 1$, which is indeed the case under a suitable choice of parameters. Note that for comparison, if such a clause C_i were not known, it would require, information-theoretically, at least $\log|\mathcal{R}|$ bits to encode any restriction $\rho \in \mathcal{R}$. This compression argument implies that a random $\rho \leftarrow \mathcal{R}$ shrinks a fixed clause w.p. $\ge 1 - \frac{1}{2c^n}$, thus the existence of a good ρ shrinking the whole proof Π follows from a union bound over the c^n clauses in Π.

For every restriction ρ, one can compute a proof $\Pi|_\rho$ with each clause $C \in \Pi$ replaced by $C|_\rho$, the restriction of C under ρ; if $\text{width}(C|_\rho) > w$, we truncate $C|_\rho$ to force its width to be at most w. Since $\Pi|_\rho$ is a width-w resolution proof, it follows from the width lower bound that it does not prove $\text{PHP}_{(n'+1)\to n'}$.

Our reduction from $\text{Refuter}(s(\text{PHP}_{(n+1)\to n} \vdash_{\mathsf{Res}} \bot) \le c^n)$ to rwPHP(PLS) works as follows.

- Let $N := |\mathcal{R}|/2$. The function $f : [N] \to [2N]$ takes as inputs (i, s) where $i \in [c^n]$ denotes a node in Π and s is the compressed description of a random restriction ρ that fails to simplify C_i to width w (note that this takes $\log(c^n) + \ell_{\mathsf{comp}} \le \log N$ bits), and outputs the standard encoding of ρ (in $\log|\mathcal{R}| = \log(2N)$ bits). It is easy to see that every restriction ρ outside the range of f would be a good restriction (that successfully shrinks every clause in Π into width w).

- For each $\rho \in \mathcal{R} \cong [2N]$, $\Pi|_\rho$ is a width-w resolution proof. By Theorem 3.8, we can reduce the problem of finding an illegal derivation in $\Pi|_\rho$ to PLS. Call this instance I_ρ.

- Finally, let $i \in [c^n]$ be an illegal derivation in $\Pi|_\rho$ (that can be found in PLS). There are two reasons that the i-th step is illegal in $\Pi|_\rho$: first, it might already be an invalid derivation in Π; second, the width of $C_i|_\rho$ might be greater than w, thus the error happens when we truncate $C_i|_\rho$ to width w. In the second case, let s be the ℓ_{comp}-bit description of ρ given that it does not simplify C_i to width $\le w$, and $g_{\rho,i} := (i, s)$, then $f(g_{\rho,i}) = \rho$.

It follows that once we found any ρ and i such that i is an answer for I_ρ and $f(g_{\rho,i}) \ne \rho$, then the i-th step is illegal for the first reason stated above, i.e., the i-th step in Π is also invalid. See Figure 2 for a high-level overview of this construction.

Random restrictions + width lower bounds. It turns out that the above proof template that combines random restrictions and width lower bounds is very popular in proving resolution lower bounds. Given a hard tautology F, we design a family of restrictions $\mathcal{R}$ such that (1) Any fixed *short* resolution proof will simplify to a *narrow* resolution proof under $\mathcal{R}$, and (2) even after a random restriction in $\mathcal{R}$, F

remains hard for narrow resolution proofs. Note that the family $\mathcal{R}$ is usually carefully chosen according to the hard tautology F; e.g., $\mathcal{R}$ corresponds to *partial matchings* when $F = \text{PHP}$ [BP96] and corresponds to *random edge sets* when $F = \text{Tseitin}$ [Sch97].

As mentioned before, this proof strategy is capable of proving resolution size lower bounds for various hard tautologies, and we can use a similar argument as the above to show that the refuter problems corresponding to these resolution size lower bounds are in rwPHP(PLS). This includes XOR-lifted formulas (Section 5.2), Tseitin tautologies (Section 5.3), and random k-CNFs (Section 5.4).

In fact, it is quite intuitive to formalize "random restrictions + width lower bounds" in $\mathsf{T}^1_2(\alpha)$ + dwPHP($\mathsf{PV}(\alpha)$). Roughly speaking, we first use dwPHP($\mathsf{PV}(\alpha)$) to formalize the compression argument and show that most random restrictions will shrink the resolution proof (represented by α) into a narrow one; then we use $\Sigma^b_1(\alpha)$-MIN (which is available in $\mathsf{T}^1_2(\alpha)$) to prove a resolution width lower bound.

1.4.2 Refuter Problems are rwPHP(PLS)-Hard

In this subsection, we explain the ideas behind the reduction from rwPHP(PLS) to the refuter problems for resolution size lower bounds. In fact, a reduction from rwPHP to the refuter problems is already implicit in the celebrated result on the NP-hardness of automating resolution [AM20] and was made explicit in [dRGN$^+$21]. It turns out that with minor modifications, the same proof can be adapted to reduce not only rwPHP but also rwPHP(PLS) to the refuter problems, thereby proving Theorem 4.4. Hence, the remainder of this subsection will focus on the rwPHP-hardness result from [dRGN$^+$21]; the complete rwPHP(PLS)-hardness result can be found in Section 4.2.

There is a clear intuition behind the reduction: suppose rwPHP *were* false, i.e., there are functions $f : [N] \to [2N]$ and $g : [2N] \to [N]$ such that $f \circ g : [2N] \to [2N]$ is the identity function, then every unsatisfiable CNF F *would* have a resolution refutation of size $\text{poly}(N, n)$. Of course, the ground truth is that such functions f and g should not exist, but a weak proof system (such as resolution itself) might not be aware of this. Suppose the weak system "thinks" that such a pair of functions (f, g) *might* exist, and it can construct a short resolution refutation of F from (f, g), then the weak system should also "think" that F *might* have a short resolution refutation. In summary, if it is hard to refute the existence of (f, g) (which means proving rwPHP), then it is also hard to prove that F does not have a short resolution refutation.

Now, our task becomes the following. We live in a strange world where there is a *surjection* from $[N]$ to $[2N]$; given an *arbitrary* unsatisfiable CNF F, we want to construct a $\text{poly}(N, n)$-size resolution refutation of F. Consider the size-$2^{O(n)}$ brute-force resolution refutation for every unsatisfiable CNF, which is represented by the following proof tree.

- The root (level 0) of the tree contains the empty clause $\bot$.
- For each level $1 \le i \le n$, each clause C at level $i - 1$ is resolved from the two clauses $C \lor x_i$ and $C \lor \overline{x}_i$, both of which sits in level i. The clauses $C \lor x_i$ and $C \lor \overline{x}_i$ are the two *children* of C. Note that each clause at level i has a width of exactly i.
- Finally, every clause C at level n corresponds to an assignment $x_C \in \{0, 1\}^n$ which is the only assignment falsifying C. Since F is unsatisfiable, there is an axiom of F that x_C falsifies. Clearly, C is a *weakening* of this axiom.

We now construct a shorter resolution refutation using the surjection from $[N]$ to $[2N]$. We guarantee that in our short refutation, each level never contains more than N clauses; this implies that our resolution refutation is of size $O(N \cdot n)$. Consider level i where $1 \le i \le n$. If level $i - 1$ contains at most N clauses, then level i contains at most $2N$ clauses: for each clause C_j in level $i-1$, there are two clauses $C'_{2j} := C \lor x_i$ and $C'_{2j+1} := C \lor \overline{x}_i$ in level i. However, since there is a *surjection* from $[N]$ to $[2N]$, it is possible to pick N clauses among these $2N$ ones such that each of the $2N$ clauses appears in these N ones! (The j-th

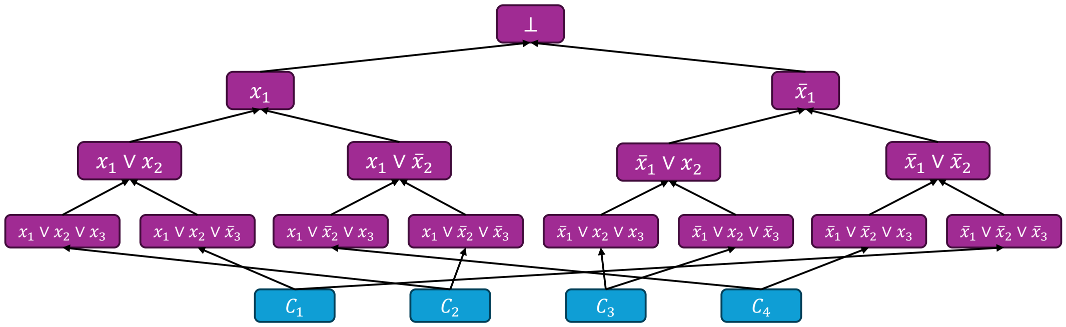


Figure 3: The brute-force resolution proof for a CNF $F = C_1 \wedge C_2 \wedge C_3 \wedge C_4$ when $n = 3$.

picked clause ($j \in [N]$) is $C'_{f(j)}$; the clause C'_j ($j \in [2N]$) appears as the $g(j)$-th picked clause.) Now that level i also contains at most N clauses, we can proceed to the next level and so on.

We stress again that the ground truth is, of course, that there do not exist functions $f : [N] \to [2N]$ and $g : [2N] \to [N]$ such that $f \circ g : [2N] \to [2N]$ is the identity function. However, the point is that given any step in the above resolution refutation that is an invalid derivation, we can pinpoint a "witness" number $x \in [2N]$ such that $f(g(x)) \neq x$.

The above describes the intuition behind the decision tree reduction from rwPHP to the refuter problems of resolution size lower bounds presented in [dRGN+21]. Our reduction from rwPHP(PLS) to the refutation problems proceeds in the same way, except that now g is only a function computable in PLS. Compared with [AM20, dRGN+21], our proof only has one more component: showing that these PLS instances can also be embedded into the above resolution refutation. We refer the reader to the formal proof in Section 4.2 for details.

1.5 Further Related Works

Refuter problems for circuit lower bounds. Our study of the refuter problems for proof lower bounds is strongly influenced by the line of work on refuter problems for circuit lower bounds. Chen, Jin, Santhanam, and Williams [CJSW24] call a lower bound *constructive* if the corresponding refuter problem can be solved in deterministic polynomial time, and they argued that constructivity is a desirable aspect of lower bounds. Chen, Tell, and Williams [CTW23] showed that for many lower bounds against randomized computational models, their refuter problems characterize derandomizing pr-BPP. The main result of Korten [Kor22] can also be seen as the WPHPWIT-hardness of refuter problems for one-tape Turing machine lower bounds. Pich and Santhanam [PS23] showed how to turn proof complexity lower bounds into circuit lower bounds, assuming the refuter problem for the (conjectured) lower bound $\mathsf{SAT} \notin \mathsf{P}/_{\mathrm{poly}}$ is "provably easy" in a certain sense. Finally, the results of Chen, Li, and Oliveira [CLO24a] can be interpreted as the PWPP- and WPHPWIT-completeness of various refuter problems.

It is also worth mentioning that Ebtehaj [Ebt23] studied the refuter problems for $\mathcal{A} \not\subseteq \mathsf{BPP}$ for each (type-1) subclass $\mathcal{A} \subseteq \mathsf{TFNP}$ that is indeed hard. However, [Ebt23] did not obtain any completeness results for such refuter problems.

Unprovability of complexity upper bounds. In parallel to the investigation of unprovability of complexity lower bounds, there is another line of work showing the unprovability of complexity *upper* bounds in fragments of bounded arithmetic [CK07, KO17, BKO20, BM20, CKKO21, ABM23]. For example, Krajíček and Oliveira [KO17] proved that Cook's theory PV cannot prove $\mathsf{P} \subseteq \mathsf{SIZE}[n^k]$, and Atserias, Buss, and Müller [ABM23] proved that the theory V^0_2 cannot prove $\mathsf{NEXP} \subseteq \mathsf{P}/_{\mathrm{poly}}$. These results are equivalent to the *consistency* of lower bounds with fragments of bounded arithmetic, thus in some sense representing progress towards proving circuit lower bounds.[15] Indeed, [CKKO21] presented a general framework for showing such consistency results by proving lower bounds against circuits with a certain uniformity condition called "LEARN-uniformity", and the techniques employed in many of these papers are inspired by uniform circuit lower bounds such as [SW14].

Witnessing theorems. TFNP and bounded arithmetic are connected through *witnessing theorems*: each theory is associated with the class of TFNP problems whose totality is provable in this theory. Perhaps the best-known witnessing theorem is Buss's one [Bus85]: every NP search problem provably total in S^1_2 can be solved in deterministic polynomial time. The class PLS and its generalizations such as CPLS capture the NP search problems provably total in higher levels of bounded arithmetic hierarchy [BK94, KST07, ST11, PT12]; in this sense, witnessing theorems also provide a systematic method for defining new syntactic subclasses of TFNP. Other witnessing theorems considered in the literature include [KNT11, BB17, KT22]. Our paper contributes to this line of research by characterizing the class of NP search problems provably total in $\mathsf{T}^1_2 + \mathrm{dwPHP}(\mathsf{PV})$ by the refuter problems corresponding to many resolution lower bounds, in particular the problem $\textsc{Refuter}(s(\mathrm{PHP}_{(n+1)\to n} \vdash_{\mathsf{Res}} \bot) < c^n)$.

Comparison with the consistency search problem. We note that the refuter problem looks superficially similar to $\textsc{WrongProof}$, the *consistency search* problem for proof systems [BB17, GP18a, Pud20]. Let $\mathcal{P}$ be a proof system, $\textsc{WrongProof}(\mathcal{P})$ is the $\mathsf{TFNP}^{\mathsf{dt}}$ problem that given as input a purported $\mathcal{P}$-proof Π of *an incorrect statement*, asks for the location of an invalid derivation in Π.

Although both $\textsc{WrongProof}$ and our refuter problems take a purported proof as input and ask for an invalid derivation in the proof, we think that these two problems are fundamentally different, because they have different *reasons of totality*. Roughly speaking, the totality of $\textsc{WrongProof}$ is proved by the *soundness* of $\mathcal{P}$, and the totality of $\textsc{Refuter}$ is guaranteed by *lower bound proofs*. We elaborate on this in Appendix B.

Another (superficial) similarity between these two problems is that both problems are used to characterize the provably total NP problems in bounded arithmetic. The consistency search problems for Frege and Extended Frege characterize the $\forall\Sigma^b_1$-consequences of U^1_2 and V^1_2 respectively [BB17], while in this paper we show that the refuter problem for resolution (with a suitable hard tautology) characterizes the $\forall\Sigma^b_1$-consequences of $\mathsf{T}^1_2 + \mathrm{dwPHP}(\mathsf{PV})$.

2 Preliminaries

The first three subsections present standard preliminaries and can be skipped if the reader is familiar. However, the last two subsections introduce new concepts and it is highly recommended to read through (i.e., not skip) them. In particular, Section 2.4 introduces the refuter problems for resolution lower bounds as $\mathsf{TFNP}^{\mathsf{dt}}$ problems, and Section 2.5 defines and discusses the subclass $\mathrm{rwPHP}(\mathsf{PLS})$.

We use 0-indexing: $[n] = \{0, 1, \ldots, n-1\}$. For functions $f : \mathcal{A} \to \mathcal{B}$ and $g : \mathcal{B} \to \mathcal{C}$, their *composition* $g \circ f$ is defined as

$$\forall x \in \mathcal{A}, (g \circ f)(x) = g(f(x)).$$

[15] The "conventional wisdom" seems to believe that the complexity lower bounds are true (for discussions, see `https://rjlipton.com/conventional-wisdom-and-pnp/`, accessed Nov 4, 2024). Hence, unprovability of complexity lower bounds can be seen as the difficulty for proving this "conventional wisdom", while unprovability of complexity upper bounds represents progress towards proving it. One should keep in mind that the opposite opinion makes equal sense: for a believer of complexity *upper* bounds, the unprovability of these upper bounds indicates the difficulty of confirming their belief, while the unprovability of lower bounds implies progress towards it!

2.1 Pigeonhole Principle

Let $m > n$, the *pigeonhole principle* (PHP) states that there is no way to send m pigeons into n holes such that different pigeons are sent to different holes. This is expressed as the following unsatisfiable CNF $\mathrm{PHP}_{m\to n}$. (In the definition below, think of $x_{ij} = 1$ if pigeon i goes to hole j.)

Definition 2.1 ($\mathrm{PHP}_{m\to n}$)**.** $\mathrm{PHP}_{m\to n}$ is the conjunction of the following set of clauses:

- $\bigvee_{j\in[n]} x_{ij}$ for every pigeon $i \in [m]$;
- $\overline{x}_{ij} \vee \overline{x}_{i'j}$ for every two different pigeons $1 \le i < i' \le m$ and every hole $j \in [n]$.

The seminal work of Haken [Hak85] proved that any resolution proof of $\mathrm{PHP}_{(n+1)\to n}$ requires $2^{\Omega(n)}$ size. The proof of this classical theorem has been simplified by several follow-up works [CP90, BP96, BW01].

2.2 Decision Tree TFNP

Let $\mathcal{O} = \{O_N\}_N$ be a family of solution spaces. A *search* problem $\mathcal{P}$ is a family of sets $\{P_N\}_{N\in\mathbb{N}}$, where each P_N is a subset of $\{0,1\}^N \times O_N$. Let $x \in \{0,1\}^N$ be an *input* to $\mathcal{P}$, we say that $o \in \mathcal{O}_N$ is a *solution* of x if $(x, o) \in P_N$. We say $\mathcal{P}$ is *total* if every $x \in \{0,1\}^*$ has at least one solution. We sometimes abuse the notation by calling an individual relation P_N a search problem, and implicitly assume that there is a sequence $\{P_N\}_N$.

We study total search problems in the *decision tree* model. In this model, we think of the input $x \in \{0,1\}^N$ as very long and can only be accessed by querying individual bits. An algorithm (i.e., decision tree) is *efficient* if it only makes $\mathrm{polylog}(N)$ many queries. We will typically consider search problems where $|O_N| \le 2^{\mathrm{polylog}(N)}$, so efficient algorithms will be able to handle solutions $o \in O_N$ in their entirety. A search problem $\mathcal{P}$ is in $\mathsf{FNP}^{\mathsf{dt}}$ if given (oracle access to) an input $x \in \{0,1\}^N$ and a solution $o \in O_N$, there is an efficient decision tree T_o for deciding whether $(x, o) \in P_N$. The class $\mathsf{TFNP}^{\mathsf{dt}}$ consists of all *total* search problems in $\mathsf{FNP}^{\mathsf{dt}}$.

For example, an important $\mathsf{TFNP}^{\mathsf{dt}}$ problem in this paper is the problem Iter, defined as follows.

Problem Iter

<u>Input:</u> A function $S : [N] \to [N]$.
<u>Output:</u> A number $x \in [N]$ is a valid solution if one of the following holds:

- $x = 0$ and $S(0) = 0$;
- $S(x) < x$; or
- $S(x) > x$ and $S(S(x)) = S(x)$.

It is easy to check that Iter is in $\mathsf{FNP}^{\mathsf{dt}}$: Given an output x and oracle access to the function $S : [N] \to [N]$, one can verify whether x is a valid solution by querying at most 2 entries of S; namely $S(x)$ and $S(S(x))$. Since each entry can be represented by at most $\log N$ bits, the query complexity of verifying solutions for Iter is $\mathrm{polylog}(N)$. On the other hand, the totality of Iter expresses the following fact: every DAG has a sink. It turns out that we will also frequently use a reversed version of Iter for simplicity, whose equivalence to Iter is easy to see: Given a function $S : [N] \to [N]$ such that $S(N-1) < N-1$, find some $x \in [N]$ such that 1) either $S(x) > x$ or 2) $S(x) < x$ and $S(S(x)) = S(x)$.

Definition 2.2 (Decision tree reductions)**.** Let $\mathcal{P}, \mathcal{Q}$ be two $\mathsf{TFNP}^{\mathsf{dt}}$ problems, and $d(N)$ be a parameter (typically $\mathrm{polylog}(N)$). A *depth-d* decision tree reduction from $\mathcal{P}$ to $\mathcal{Q}$ consists of two functions (f, g), where each output bit of f, g can be computed from the input x by a depth-d decision tree:

- $f : \{0,1\}^N \to \{0,1\}^{M(N)}$ maps an input x of $\mathcal{P}$ to an input $f(x)$ of $\mathcal{Q}$.

- g maps any valid solution of $f(x)$ (as an instance of $\mathcal{Q}$) into a valid solution of x (as an instance of $\mathcal{P}$).

We say the reduction is *uniform* if both f and g can be computed by uniform Turing machines with query access to x. We allow $M(N)$ to be super-polynomial in N, but we require $M(N) \le \exp(d(N))$.

Usually, for two $\mathsf{TFNP}^{\mathsf{dt}}$ problems $\mathcal{P}, \mathcal{Q}$ we say $\mathcal{P}$ can be (*many-one*) reduced to $\mathcal{Q}$ if there is a polylog(N)-depth decision tree reduction from $\mathcal{P}$ to $\mathcal{Q}$.

The class PLS[16] is the class of problems in $\mathsf{TFNP}^{\mathsf{dt}}$ that has a depth-polylog(N) reduction to ITER. (Note that PLS was originally defined differently [JPY88]; the PLS-completeness of ITER was shown in [Mor01].)

The inputs of most $\mathsf{TFNP}^{\mathsf{dt}}$ problems introduced in this paper will be partitioned into blocks; for example, the input of ITER consists of N blocks where each block consists of $\log N$ bits describing an integer in $[N]$. It will be more convenient to work with the *block-depth* of decision trees, which is the number of different *blocks* that a decision tree queries. For example, solutions of ITER can be verified in block-depth 2. The problems in this paper will have block size polylog(N), hence polylog(N) block-depth is equivalent to polylog(N) (bit-)depth. However, we will upper bound the complexity of our decision trees by block-depth for convenience. Although the distinction of depth and block-depth does not make an essential difference in this paper, many interesting lifting theorems and non-automatability results are recently proved using the notion of block-depth (or block-width) [AM20, GKMP20, dRGN⁺21]. It might be beneficial to have bounds on block-depth, which is usually sharper as the decision trees we construct tend to query many bits in the same block.

We assume all the $\mathsf{TFNP}^{\mathsf{dt}}$ problems discussed in this paper are *paddable*, i.e., for any $N < M$, solving an instance of size N could always be efficiently reduced to solving an instance of size M of the same problem. Most of the common $\mathsf{TFNP}^{\mathsf{dt}}$ problems can be easily formulated in a paddable way.[17]

2.2.1 Connection to Proof Complexity

There is a generic connection between $\mathsf{TFNP}^{\mathsf{dt}}$ and propositional proof complexity via the *false clause search* problem (see, e.g. [dRGR22, BFI23]).

Definition 2.3. For an unsatisfiable CNF $F := C_1 \wedge \cdots \wedge C_m$, Search$(F)$ is the search problem in which an assignment x to F is given via query access, and a solution is a clause C_i of F falsified by x.

Define Search$(\mathcal{F})$ for a family of formula $\mathcal{F} = \{F_n\}_{n\in\mathbb{N}}$ as $\{\text{Search}(F_n)\}_{n\in\mathbb{N}}$ accordingly.

When the width of F is polylog(n), where n is the number of variables in F, Search(F) is a $\mathsf{TFNP}^{\mathsf{dt}}$ problem. In the other direction, for any $\mathsf{TFNP}^{\mathsf{dt}}$ problem $R_n \in \{0,1\}^n \times O_n$, it can be equivalently written as Search(F_n) for some CNF F_n of polylog(n) width. More specifically, let $\{T_o\}_{o\in O_n}$ be the set of efficient decision trees for verifying solutions, $\neg T_o(x)$ can be written as a low-width CNF stating that any accepting path in T_o is falsified by x. We then take

$$F_n = \bigwedge_{o\in O_n} \neg T_o(x), \tag{2}$$

and it is easy to see the equivalence between Search(F_n) and R_n by definition.

Informally, we say a proof system P is characterized by a syntactical $\mathsf{TFNP}^{\mathsf{dt}}$ subclass C if for any family of formula $\mathcal{F} = \{F_n\}$, P has a *small* proof of $\mathcal{F}$ if and only if Search$(\mathcal{F}) \in \mathsf{C}$. Buss, Fleming,

[16] In this paper, most of the times when we mention a syntactic subclass of TFNP (such as PLS) we mean the decision tree version of it (i.e., $\mathsf{PLS}^{\mathsf{dt}}$), and it should be easy to figure out whether we mean the decision tree version or the Turing machine version of this subclass from the context. Therefore, for convenience, we drop the superscript dt when we express syntactic subclasses of $\mathsf{TFNP}^{\mathsf{dt}}$. We still preserve the superscript dt in "$\mathsf{TFNP}^{\mathsf{dt}}$" when we want to emphasize that the underlying model is decision tree TFNP.

[17] There is a similar notion called *instance extension*, which is defined in [BM04].

and Impagliazzo [BFI23] showed that any well-behaved[18] proof system P is characterized by a $\mathsf{TFNP}^{\mathsf{dt}}$ subclass C, and vice versa. In particular, resolution is characterized by PLS.

Theorem 2.4 (Folklore)**.** *Let $\mathcal{F} = \{F_n\}$ be a family of unsatisfiable formula,* $\mathrm{Search}(\mathcal{F}) \in \mathsf{PLS}$ *if and only if $\mathcal{F}$ have a* $\mathrm{polylog}(n)$*-width resolution refutation.*

2.3 Bounded Arithmetic

We introduce the theories T^1_2, $\mathsf{T}^1_2 + \mathrm{dwPHP}(\mathsf{PV})$, as well as their relativized versions. A more comprehensive introduction of bounded arithmetic (including the theories S^i_2 and T^i_2) can be found in [Kra95].

The language of bounded arithmetic consists of the following symbols

$$L_{\mathrm{BA}} := \{0, 1, +, \cdot, <, =, \lfloor \cdot/2 \rfloor, |\cdot|, \#\}.$$

Here, the intended meaning of $|a|$ is the *bit-length* of the binary number a, i.e.,

$$|a| := \begin{cases} \lceil \log_2(a+1) \rceil & \text{if } a > 0; \\ 0 & \text{if } a = 0. \end{cases}$$

The intended meaning of $\#$ ("smash") is

$$x \# y := 2^{|x| \cdot |y|};$$

roughly speaking, this symbol is used to create objects whose size is *polynomial*, instead of only *linear*, in the length of its inputs. These symbols are governed by a list of 32 axioms called BASIC, each of which asserts some basic fact about the intended meanings of these symbols. For instance:

$$a \le b \to a \le b + 1. \qquad \text{(axiom 1 in BASIC)}$$

The complete list of BASIC axioms can be found in [Kra95, Definition 5.2.1].

A *bounded quantifier* is a quantifier of the form

$$\forall y < t(\vec{x}) \quad \text{or} \quad \exists y < t(\vec{x})$$

for some term t. Formally, they are defined as abbreviations:

$$\begin{aligned} \forall y < t(\vec{x})\ \varphi(\vec{x}, y) &:= \forall y\ (y < t(\vec{x}) \to \varphi(\vec{x}, y)); \\ \exists y < t(\vec{x})\ \varphi(\vec{x}, y) &:= \exists y\ (y < t(\vec{x}) \land \varphi(\vec{x}, y)). \end{aligned}$$

A *sharply bounded quantifier* is a quantifier of the form

$$\forall y < |t(\vec{x})| \quad \text{or} \quad \exists y < |t(\vec{x})|.$$

That is, the domain of possible values of y is bounded by the length of a term. Intuitively, sharply bounded quantifiers are "feasible" because, thinking of $t(\vec{x})$ as the description of a polynomial-size object, there are only polynomially many possibilities of y and they can be enumerated in polynomial time.

A formula is *sharply bounded* if all quantifiers in it are sharply bounded quantifiers. A Σ^b_1*-formula* is a formula constructed from sharply bounded formulas using $\land$, $\lor$, sharply bounded quantifiers, and *existential bounded quantifiers* ("$\exists y < t(\vec{x})$"). It can be shown that the languages defined by Σ^b_1-formulas are exactly those computed in NP.

The power of theories in bounded arithmetic comes from their *induction axioms*. Let Φ be a class of formulas, then Φ-IND is the following axiom schema

$$(\phi(0) \land \forall x\ (\phi(x) \to \phi(x+1))) \to \forall x\ \phi(x)$$

[18] Here, a proof system is well-behaved if it is closed under decision tree reduction, and it can prove its own soundness.

for every $\phi \in \Phi$. The definition of T^1_2 is:

$$\mathsf{T}^1_2 := \text{BASIC} + \Sigma^b_1\text{-IND}.$$

That is, when reasoning in T^1_2, it is allowed to use induction axioms over Σ^b_1 formulas (i.e., NP languages).

It is equivalent, and sometimes more convenient to replace Σ^b_1-IND with Σ^b_1-MIN, the *minimization principle* over Σ^b_1 formulas. For a set of formulas Φ, the axiom schema Φ-MIN consists of

$$\phi(a) \to \exists x \leq a \forall y < x\ (\phi(x) \land \neg\phi(y))$$

for every $\phi \in \Phi$. Equivalently, when reasoning in T^1_2, it is allowed to use the fact that there exists a *smallest* x such that $C(x) = 1$, whenever C is a polynomial-size *nondeterministic circuit* and we know some y such that $C(y) = 1$.

The theory PV is an equational theory defined by Cook [Coo75] to capture polynomial-time reasoning. It contains a function symbol for every polynomial-time algorithm, introduced inductively using Cobham's recursion-theoretic characterization of polynomial time [Cob64]. More detailed treatments about PV can be found in [Kra95, CN10, CLO24a]. In the literature, it is common to also use PV to denote the set of function symbols in PV (which corresponds to functions computable in polynomial time).

The *dual weak pigeonhole principle* over PV functions, denoted as dwPHP(PV), is the following axiom schema

$$\forall a > 1 \exists v < a^2 \forall u < a\ f(u) \neq v$$

for every PV-function f with parameters[19]. Roughly speaking, this means that if we have a polynomial-size circuit $f : \{0,1\}^n \to \{0,1\}^{2n}$ (think of $a = 2^n$ above), then there exists some $v \in \{0,1\}^{2n}$ that is not in the range of C. We note that the choice of a^2 above is somewhat arbitrary, as dwPHP(PV) with various parameters are equivalent over $\mathsf{S}^1_2 \subseteq \mathsf{T}^1_2$ [PWW88, Jeř04].

To summarize, when reasoning in the theory T^1_2 + dwPHP(PV), one is allowed to use the following two axiom schemas:

(Σ^b_1-MIN) For a polynomial-size nondeterministic circuit C and some y such that $C(y) = 1$, there exists a *smallest* x such that $C(x) = 1$.

(dwPHP(PV)) For a polynomial-size circuit $C : \{0,1\}^n \to \{0,1\}^{2n}$, there exists a string $y \in \{0,1\}^{2n}$ that is not in the range of C.

Finally, the *relativized* theories $\mathsf{T}^1_2(\alpha)$ and $\mathsf{T}^1_2(\alpha)$ + dwPHP($\mathsf{PV}(\alpha)$) are simply their unrelativized counterparts with a new unary relation symbol α added into the language L_{BA}. (One can think of α as an oracle that encodes an exponentially-long input; for example, $\alpha(i)$ might encode the i-th bit of an exponentially-long resolution proof according to some canonical encoding.) The class of $\Sigma^b_1(\alpha)$ formulas and axioms $\Sigma^b_1(\alpha)$-MIN and dwPHP($\mathsf{PV}(\alpha)$) are relativized in a straightforward way. There are no other axioms involving α except for the induction axioms and dual weak pigeonhole principles. To summarize:

- When reasoning in $\mathsf{T}^1_2(\alpha)$, it is allowed to use $\Sigma^b_1(\alpha)$-MIN, i.e., for any polynomial-size nondeterministic *oracle* circuit C^α and input y such that $C^\alpha(y) = 1$, there exists a *smallest* input x such that $C^\alpha(x) = 1$.
- When reasoning in $\mathsf{T}^1_2(\alpha)$ + dwPHP($\mathsf{PV}(\alpha)$), it is additionally allowed to use the fact that for any polynomial-size *oracle* circuit $C^\alpha : \{0,1\}^n \to \{0,1\}^{2n}$, there exists some $y \in \{0,1\}^{2n}$ that is not in the range of C^α.

[19] This is the standard terminology in bounded arithmetic that means f might depend on some other parameter not shown above. The parameter can be thought of as non-uniformity; cf. Footnote 8.

2.4 Refuter Problems for Resolution Lower Bounds

We provide formal definitions of the refuter problems in the decision tree model. We begin by defining *resolution refutations*; the definition is adapted from [dRGN+21, Section 3.1].

Definition 2.5. Let F be an unsatisfiable CNF with n variables and m clauses; the clauses in F will be called *axioms* and will be denoted as $C_{-m}, \ldots, C_{-1}$ for convenience. A *resolution refutation* of F is a sequence of nodes $C_0, C_1, \ldots, C_{L-1}$, where each node C_i contains the following information.

- A set of literals among $\{x_1, x_2, \ldots, x_n, \overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n\}$. Abusing notation, we also denote the clause consisting of the disjunction of these literals by C_i.
- A *tag* which is one of the following: "resolution" or "weakening".
- Two integers $-m \le j, k < i$ and a variable $a \in \{1, 2, \ldots, n\}$ if the tag is "resolution". This means that C_i is obtained from the clauses C_j and C_k by *resolving* the variable x_a.
- One integer $-m \le j < i$ if the tag is "weakening". This means that C_i is a *weakening* of C_j.

The resolution refutation is valid if the following is true for every $1 \le i \le L$:

- If C_i is marked "resolution", then there are clauses D and E such that $C_j = x_a \vee D$, $C_k = \overline{x}_a \vee E$, and $C_i = D \vee E$.
- If C_i is marked "weakening", then there is a clause D such that $C_i = C_j \vee D$.
- Finally, $C_{L-1} = \bot$ (i.e., contains no literals).

The *length* or *size* of the refutation is L, and the *width* of the refutation is the maximum integer w such that every clause C_i ($-m \le i < L$) in the refutation contains at most w literals.

Resolution is *complete* and *sound*: a CNF F has a resolution refutation (of whatever length) if and only if it is unsatisfiable.

Each node in the resolution refutation would be a *block*; therefore, when we say a decision tree over a resolution refutation has block-depth d, we mean that it only queries (potentially all information in) d nodes of the refutation.

Next, we define the refuter problems.

Definition 2.6. Let $\mathcal{F} = \{F_n\}_{n\in\mathbb{N}}$ be a family of unsatisfiable CNFs where every F_n requires resolution of width greater than w_n and size greater than s_n.

- An input to the problem $\textsc{Refuter}(w(F_n \vdash_{\mathsf{Res}} \bot) \le w_n)$ is a purported resolution refutation of F_n with width at most w_n. (It is easy to syntactically guarantee that the width of the input refutation is at most w_n by allocating only w_n literals for each node.)
- An input to the problem $\textsc{Refuter}(s(F_n \vdash_{\mathsf{Res}} \bot) \le s_n)$ is a purported resolution refutation of F with at most s_n clauses.

The outputs of these problems consist of only one index i, which means the node C_i does not satisfy the validity conditions defined in Definition 2.5. We will call such nodes *invalid derivations* or *illegal derivations*.

Note that each node can be described in $\mathrm{poly}(w, \log n, \log L)$ bits where w is the width of the resolution refutation. Hence, in the typical parameter regime, we will consider resolution refutations whose length is exponential in its width ($L = 2^{w^{\Omega(1)}}$), so that the access to each block is "efficient", i.e., only needs to query polylogarithmic many bits. In particular, the typical parameter regime for size lower bounds is exponential

($L = 2^{n^{\Omega(1)}}$); polynomial width lower bound ($w = n^{\Omega(1)}$) is considered in Section 3 and Section 5, while polylogarithmic width lower bound is considered in Section 6.2. We also assume $L = 2^{n^{O(1)}}$, so that the proof is not extremely redundant.

Since there is a decision tree of block-depth at most 3 verifying whether a given node in the resolution refutation is an invalid derivation, the refuter problems defined above are in $\mathsf{FNP}^{\mathsf{dt}}$. Moreover, if the resolution lower bounds (w_0 or s_0) are indeed true, then the refuter problems defined above are total. Hence, $\textsc{Refuter}(\cdot)$ is a natural family of problems in $\mathsf{TFNP}^{\mathsf{dt}}$.

2.5 $\mathcal{P}$-Retraction Weak Pigeonhole Principle

Recall that rwPHP, the *retraction weak pigeonhole principle*, is the following principle:

Fact 2.7. *Let $g : [2M] \to [M]$ and $f : [M] \to [2M]$ be two functions. Then there must exist some $y \in [2M]$ such that $f(g(y)) \neq y$.*

Roughly speaking, for any TFNP class $\mathcal{P}$, rwPHP($\mathcal{P}$) is the retraction weak pigeonhole principle where the retraction ($g : [2M] \to [M]$) is a (multi-valued) function computable in $\mathcal{P}$. For example:

Problem rwPHP(PLS)

Input: Let $M \leq N/2$. The input consists of the following functions:

- $f : [M] \to [N]$ is a purported "surjection";
- for each $y \in [N]$, $I_y := (L, S_y)$ is an instance of $\textsc{Iter}$, where $S_y : [L] \to [L]$; and
- $g_y : [L] \to [M]$ maps solutions of I_y to integers in $[M]$.

Output: A number $y \in [N]$ and $ans \in [L]$ such that ans is a solution of the $\textsc{Iter}$ instance g_y and $f(g_y(ans)) \neq y$.

In general, for a $\mathsf{TFNP}^{\mathsf{dt}}$ problem $\mathcal{P}$, we define rwPHP($\mathcal{P}$) by replacing each I_y in the above definition with an instance of $\mathcal{P}$.

Fact 2.8. rwPHP($\mathcal{P}$) $\in \mathsf{TFNP}^{\mathsf{dt}}$.

Proof. To verify a solution (y, ans), check that ans is a valid solution for I_y and that $f(g_y(ans)) \neq y$.

The totality (i.e., existence of solutions) can be argued as follows. For each $y \in [N]$, let $ans(y)$ be a solution (say the lexicographically first one) of I_y; since $\mathcal{P}$ is a total problem, $ans(y)$ exists. Let $g'(y) := g_y(ans(y))$. By the retraction weak pigeonhole principle, there exists some $y \in [N]$ such that $f(g'(y)) \neq y$. It follows that $(y, ans(y))$ is a valid solution for rwPHP($\mathcal{P}$). ☐

Fact 2.9. *There is a depth-1 decision tree reduction from $\mathcal{P}$ to* rwPHP($\mathcal{P}$) *and a depth-1 decision tree reduction from* rwPHP *to* rwPHP($\mathcal{P}$).

Proof. To reduce $\mathcal{P}$ to rwPHP($\mathcal{P}$): let I be a $\mathcal{P}$ instance. Define $f(x) = 1$ as a trivial function; for each $y \in [N]$, define the instance $I_y := I$; for every possible answer ans of I_y, let $g_y(ans) = 1$. Clearly, for any answer (y, ans) of the rwPHP($\mathcal{P}$) instance, ans itself would be a valid answer of the $\mathcal{P}$ instance I.

To reduce rwPHP to rwPHP($\mathcal{P}$): let $f : [M] \to [N]$ and $g : [N] \to [M]$ be an rwPHP instance. Fix any (say trivial) $\mathcal{P}$ instance I. For each $y \in [N]$, define the instance $I_y := I$; for every possible answer ans of I_y, let $g_y(ans) = g(y)$. For any answer (y, ans) of the rwPHP($\mathcal{P}$) instance, since $f(g_y(ans)) \neq y$, it follows that $f(g(y)) \neq y$, and hence y is a valid answer of the rwPHP instance (f, g). ☐

Classical techniques (such as Prover-Delayer games) show that the totality of rwPHP requires resolution width $\Omega(M)$ to prove (see also [PT19, dRGN+21]), which means that any decision tree of depth $o(M)$ cannot reduce rwPHP to PLS. It follows from Fact 2.9 that there is a black-box separation between rwPHP(PLS) and PLS itself.

Fact 2.10. *Let $\mathcal{P}$ and $\mathcal{Q}$ be $\mathsf{TFNP}^{\mathsf{dt}}$ problems. If there is a depth-d decision tree reduction from $\mathcal{P}$ to $\mathcal{Q}$, then there is a depth-d decision tree reduction from* $\mathrm{rwPHP}(\mathcal{P})$ *to* $\mathrm{rwPHP}(\mathcal{Q})$.

Proof. Let $f : [M] \to [N]$, $\{I_y\}_{y\in[N]}$, and $\{g_y\}_{y\in[N]}$ be an instance of $\mathrm{rwPHP}(\mathcal{P})$. Define an instance $(f', \{I'_y\}, \{g'_y\})$ of $\mathrm{rwPHP}(\mathcal{Q})$ as follows. The function $f' := f$ stays the same; each I'_y is obtained by running the reduction from $\mathcal{P}$ to $\mathcal{Q}$ on I_y; for each possible answer ans' of each I'_y, we run the reduction to obtain an answer ans of I_y, and return $g'_y(ans') := g_y(ans)$.

Finally, let (y', ans') be a valid answer of the instance $(f', \{I'_y\}, \{g'_y\})$. Let $y := y'$ and run the reduction to obtain an answer ans of I_y from the answer ans' of I'_y, then (y, ans) is a valid answer of $(f, \{I_y\}, \{g_y\})$. □

It follows from Fact 2.10 that, for example, the class $\mathrm{rwPHP}(\mathsf{PLS})$ can be defined from *any* complete problem for PLS.

Amplification for rwPHP. In this paper, when we talk about rwPHP, we always think about a purported "surjection" $f : [M] \to [N]$ where $M = 2N$. This is without loss of generality since the complexity of rwPHP does not depend significantly on the relationship between M and N (unless they are too close to each other). This is also true for $\mathrm{rwPHP}(\mathcal{P})$ provided that $\mathcal{P}$ is closed under *Turing reductions.* We note that many interesting subclasses of TFNP (such as PLS, PPA, PPAD, and PPADS) are indeed closed under Turing reductions [BJ12, Section 6], with the notable exception of PPP in the black-box model [FGPR24].

Theorem 2.11 (Informal)**.** *Suppose $\mathcal{P}$ is closed under Turing reductions. Let* $\mathrm{rPHP}_{M\to N}(\mathcal{P})$ *denote the problem* $\mathrm{rwPHP}(\mathcal{P})$ *where the given "purported surjection" is from $[M]$ to $[N]$.*[20] *Then there is an efficient decision tree reduction from* $\mathrm{rPHP}_{M\to(M+M/\mathrm{polylog}(M))}(\mathcal{P})$ *to* $\mathrm{rPHP}_{M\to M^{100}}(\mathcal{P})$.

We remark that amplification of weak pigeonhole principles is a well-known fact in bounded arithmetic [PWW88, Tha02, Kra04, Jeř04, Jeř07b, CLO24a] and total search problems [Kor21, Kor22]. Since the proof follows from standard arguments in the literature, we postpone it to Appendix A.

2.5.1 Witnessing for $\mathsf{T}^1_2 + \mathrm{dwPHP}(\mathsf{PV})$

As mentioned in the introduction, $\mathrm{rwPHP}(\mathsf{PLS})$ is exactly the class of TFNP problems whose totality can be proved in (the universal variant of) $\mathsf{T}^1_2 + \mathrm{dwPHP}(\mathsf{PV})$. This is an easy corollary of [BK94] and [AT14, Lemma 1] as we explain below.

Suppose that $\phi(x) := \exists y < t_\phi(x)\ \psi(x,y)$ is a $\Sigma^b_1(\alpha)$-sentence and

$$\mathsf{T}^1_2(\alpha) + \mathrm{dwPHP}(\mathsf{PV}(\alpha)) \vdash \forall x\ \phi(x).$$

It is shown in [AT14, Lemma 1] that there is a term $t = t(x)$ and a function symbol $f \in \mathsf{PV}(\alpha)$ such that

$$\mathsf{T}^1_2(\alpha) \vdash \forall x\ (t > 2 \wedge \forall v < t^2 \exists u < t\ (f_x(u) = v \vee \phi(x))). \tag{3}$$

(In fact, this follows from standard manipulations underlying Wilkie's witnessing theorem for $\mathsf{S}^1_2 + \mathrm{dwPHP}(\mathsf{PV})$; see e.g., [Jeř04, Proposition 14].)

To parse Equation 3, note that $f_x : [t] \to [t^2]$ defines a "stretching" function and hence cannot be surjective. The non-existence of u such that $f_x(u) = v$ exactly means that v is not in the range of f_x; hence given Equation 3, $\forall x\ \phi(x)$ follows from the dual weak pigeonhole principle.

The following problem is in PLS by the witnessing theorem for T^1_2 [BK94]. On input (x, v), find either some $u < t(x)$ such that $f_x(u) = v$ or some $y < t_\phi(x)$ such that $\psi(x,y)$ holds (we call such y a

[20] The notation $\mathrm{rPHP}_{M\to N}$ means "retraction pigeonhole principle from M pigeons to N holes". As "weak" conventionally refers to the case where $M = 2N$, the retraction pigeonhole principle with M and N specified are called $\mathrm{rPHP}_{M\to N}$ instead of $\mathrm{rwPHP}_{M\to N}$.

"certificate" for x). Now it is at least easy to see that there is a *randomized* reduction from the TFNP problem corresponding to $\phi(x)$ to PLS: Let $v \leftarrow [t^2]$ be random (which is a non-output of f_x w.h.p.), then the above PLS procedure finds some y that is a certificate for x.

Working slightly harder, we can see that the above is actually a reduction to rwPHP(PLS):

- The "purported surjection" is the function $f_x : [t] \to [t^2]$.
- For each $v \in [t^2]$, there is a PLS instance I_v which captures the problem of given (x, v) outputting either $u \in f_x^{-1}(v)$ or a certificate y for x.

Hence, given any $v \in [t^2]$ and solution *ans* of I_v such that *ans* does not contain the information of $u \in [t]$ such that $f(u) = v$, it must be the case that *ans* leads to a certificate for x. It follows that the TFNP search problem corresponding to ϕ reduces to rwPHP(PLS).

On the other hand, it is easy to see that $\mathsf{T}^1_2 + \mathrm{dwPHP}(\mathsf{PV})$ proves the totality of rwPHP(PLS) and the proof relativizes. Let $(f, \{I_y\}, \{g_y\})$ be an instance of rwPHP(PLS). By dwPHP(f), there exists $y \in [N]$ that is not in the range of f. Since T^1_2 proves the totality of PLS, it also proves the existence of a solution *ans* for I_y. Note that since y is not in the range of f, we have in particular that $f(g_y(ans)) \neq y$. Hence, (y, ans) is a valid solution for this rwPHP(PLS) instance.

3 Refuters for the Pigeonhole Principle

In this section, we study the refuter problems where the family of hard tautologies is the Pigeonhole principle $\mathrm{PHP}_{(n+1)\to n}$. Our main results are the PLS-memberships of width refuter problems for (variants of) $\mathrm{PHP}_{(n+1)\to n}$ and the rwPHP(PLS)-memberships of the size refuter problems for $\mathrm{PHP}_{(n+1)\to n}$. Looking ahead, in Section 4, we will establish *universal* PLS-hardness for width refuters and rwPHP(PLS)-hardness for size refuters, thereby characterizing their complexities in $\mathsf{TFNP}^{\mathsf{dt}}$.

3.1 Refuters for Narrow Resolution Proofs

Historically, proving size lower bounds for resolution has been challenging and considered milestones in proof complexity. The honor of the first super-polynomial size lower bounds belongs to the Pigeonhole Principle ($\mathrm{PHP}_{(n+1)\to n}$). However, in terms of width lower bound, PHP is not satisfactory: $\mathrm{width}(\mathrm{PHP}_{(n+1)\to n} \vdash_{\mathsf{Res}} \bot) = n$, but there is already an *axiom* in PHP that has width n. Therefore, studying the complexity of finding a wide clause is uninteresting, as one of the widest clauses appears directly in the axiom and can be easily located. In what follows, we consider the width refuter problem of a *constant-width analog* of $\mathrm{PHP}_{(n+1)\to n}$, namely, the nondeterministic extension $\mathrm{EPHP}_{(n+1)\to n}$, defined below. However, we will come back to $\mathrm{PHP}_{(n+1)\to n}$ shortly after $\mathrm{EPHP}_{(n+1)\to n}$ and examine the refuter problem for the so-called "monotone width" of $\mathrm{PHP}_{(n+1)\to n}$. The monotone width lower bound for $\mathrm{PHP}_{(n+1)\to n}$ will also serve as a key component in the study of size refuters for $\mathrm{PHP}_{(n+1)\to n}$.

Definition 3.1 ($\mathrm{EPHP}_{(n+1)\to n}$)**.** $\mathrm{EPHP}_{(n+1)\to n}$ is the same as $\mathrm{PHP}_{(n+1)\to n}$ except that we replace every clause $\bigvee_{j\in[n]} x_{ij}$ by a 3-CNF nondeterministic extension; that is, by the following $n+2$ clauses:

$$\overline{y}_{i,0}, (y_{i,0} \vee x_{i,1} \vee \overline{y}_{i,1}), (y_{i,1} \vee x_{i,2} \vee \overline{y}_{i,2}), \cdots, (y_{i,n-1} \vee x_{i,n} \vee \overline{y}_{i,n}), y_{i,n},$$

where $y_{i,0}, \cdots, y_{i,n}$ are newly introduced variables.

Width lower bounds for $\mathrm{EPHP}_{(n+1)\to n}$ were proved by Ben-Sasson and Wigderson [BW01].

Theorem 3.2 ([BW01, Theorem 4.9])**.** *Any resolution refutation of* $\mathrm{EPHP}_{(n+1)\to n}$ *contains a clause* C *with* $w(C) \geq n/3$.

Theorem 3.3. *Refuter*$(w(\mathrm{EPHP} \vdash_{\mathsf{Res}} \bot) < n/3)$ *is in* PLS*. In particular, there is a uniform decision tree reduction of block-depth* 3 *from the refuter problem to* Iter*.*

Proof. We will reduce it to an instance of reversed ITER. This reduction is an analog of a *constructive* version of width lower bound proofs by Beame and Pitassi [BP96], which we will use again later in the proof of Theorem 3.8. We call $(x_{i,j})$ *original* variables and $(y_{i,j})$ *extension* variables.

We consider a set of functions over all the variables, including $n+1$ pigeon functions $\{EP_i\}$ where

$$EP_i := \overline{y}_{i,0} \wedge (y_{i,0} \vee x_{i,1} \vee \overline{y}_{i,1}) \wedge (y_{i,1} \vee x_{i,2} \vee \overline{y}_{i,2}) \wedge \cdots \wedge (y_{i,n-1} \vee x_{i,n} \vee \overline{y}_{i,n}) \wedge y_{i,n}, \tag{4}$$

and $n^2(n+1)/2$ hole functions $\left\{H^j_{(i,i')}\right\}$, where $H^j_{(i,i')} = \overline{x}_{i,j} \vee \overline{x}_{i',j}$. It is easy to see that the semantic meaning of $y_{i,j}$ is whether "the index of the hole that pigeon i goes into belongs to $\{1, \ldots, j\}$" or not.

We say an assignment α over all variables (including the extension variables) is *ℓ-critical* if $EP_\ell(\alpha) = 0$ but all other functions are 1 under this assignment, namely $EP_i(\alpha) = 1$ for all $i \neq \ell$ and $H^j_{(i,i')}(\alpha) = 1$ for all $j \in [n], i, i' \in [n+1]$. If we ignore the extension variables, α is essentially a complete matching between pigeons and holes, except pigeon i is not going anywhere. Given the definition of ℓ-critical assignments, we define a complexity measure for a clause C, denoted by $\mathrm{cri}(C)$:

$$\mathrm{cri}(C) := |\{\ell : \exists \ell\text{-critical assignment } \alpha \text{ such that } C(\alpha) = 0\}|.$$

Note that cri has four important properties:

(I) $\mathrm{cri}(\perp) = n+1$;

(II) $\mathrm{cri}(EP_i) = 1$ for all i and $\mathrm{cri}(H^j_{i,i'}) = 0$ for all j, i, i';

(III) cri is subadditive with respect to resolution derivation, namely, if C is resolved from A and B, then $\mathrm{cri}(C) \leq \mathrm{cri}(A) + \mathrm{cri}(B)$;

(IV) if C is obtained from a weakening of A, then $\mathrm{cri}(C) \leq \mathrm{cri}(A)$.

We first show that $\mathrm{cri}(\cdot)$ can be computed in polynomial time. Then we show that any clause C_i such that $n/3 \leq \mathrm{cri}(C_i) \leq 2n/3$ will give us a solution. The PLS-membership follows from that the standard 1/3-2/3 trick can be implemented via a reduction to reversed ITER.

Lemma 3.4. *For any clause C, $\mathrm{cri}(C)$ can be computed in $\mathrm{poly}(n)$ time.*

Proof. Fix any clause C. We will enumerate ℓ and check the existence of ℓ-critical assignments. The only part that we need to be careful is how we deal with the extension variables.

Imagine that we maintain a complete bipartite graph with $n+1$ nodes on the left and n nodes on the right. We will iteratively delete some edges from this graph based on the requirement of ℓ-critical assignments. We will show that the existence of ℓ-critical assignment can be reduced to the existence of a perfect matching of the final graph.

For an ℓ-critical assignment, EP_ℓ needs to be 0, and all other functions need to be 1, so we have that $x_{\ell,j}$ needs to be 0 for all j (which means pigeon ℓ cannot go into any hole). Note that if pigeon ℓ were matched with some hole, some other pigeons would have no place. So we delete edges between (ℓ, j) for all j.

For some α being an ℓ-critical assignment, we have $C(\alpha) = 0$. This means all literals that appeared in C are fixed to be 0 in the search of α. If C contains a literal $\overline{x}_{\ell,j}$ for some j, then we directly conclude that ℓ-critical assignment does not exist (due to the argument above).

Now assume that C does not contain any literal $\overline{x}_{\ell,j}$. For every literal $x_{i,j}$ in C, in order to falsify C, $x_{i,j}$ is going to be 0, so we delete the edge (i,j). For every literal $\overline{x}_{i,j}$ in C, $x_{i,j}$ is going to be 1, meaning that pigeon i is going to be matched with hole j, so we delete the edge (i, j') for every $j' \neq j$ and (i', j) for every $i' \neq i$.

For every literal $y_{i,j}$ in C, $y_{i,j}$ is going to be 0, so we delete edges (i, j') for all $j' \leq j$. For every literal $\overline{y}_{i,j}$ in C, $y_{i,j}$ is going to be 1, so we delete edges (i, j') for all $j' > j$.

We can conclude that there is an ℓ-critical assignment if and only if the remaining bipartite graph has a perfect matching between pigeons $[n+1] \setminus \{\ell\}$ and holes $[n]$. ⋄

Reduction to Iter: The reversed Iter instance is defined by the following function $S : [L] \to [L]$. For every $i \in [L]$:

- If $\mathrm{cri}(C_i) < \frac{2n}{3}$, then $S(i) = i$.
- Otherwise, if C_i is a weakening from C_j, then let $S(i) = j$.
- Finally, if C_i is resolved from C_j and C_k: If $\mathrm{cri}(C_j) \geq \mathrm{cri}(C_k)$, then $S(i) = j$; otherwise $S(i) = k$.

It is easy to see that this reduction can be implemented in block-depth 3: for example, if C_i is resolved from C_j and C_k, then one only needs to read the i-th, j-th, and k-th node in the resolution refutation.

Note that when we find any solution i of this reversed Iter instance, it satisfies $S(i) < i$ and $S(S(i)) = S(i)$. This means $\mathrm{cri}(C_i) \geq 2n/3$ but $\mathrm{cri}(C_{S(i)}) < 2n/3$. Thus we have $\mathrm{cri}(C_{S(i)}) \in [n/3, 2n/3]$.

Now it remains for us to show that any C such that $n/3 \leq \mathrm{cri}(C) \leq 2n/3$ has width at least $n/3$. For the sake of contraction, assume that $\mathrm{width}(C) < n/3$. This implies that for at most $n/3$ pigeon i, C has some variable related to i, namely, some $x_{i,j}$ or $y_{i,j}$. Since $\mathrm{cri}(C) \geq n/3$, we know that there exists ℓ such that there is an ℓ-critical assignment α for C but C has no variables of the form $x_{\ell,j}$ or $y_{\ell,j}$.

On the other hand, since $\mathrm{width}(C) < n/3$ and $\mathrm{cri}(C) \leq 2n/3$, we know that there is another index ℓ' such that ℓ' is not critical to C and C has no variables of the form $x_{\ell',j}$ or $y_{\ell',j}$. Let k be the hole that is matched with ℓ' in α. Consider the following assignment α': we start from $\alpha' := \alpha$, flip $x_{\ell,k}$ from 0 to 1, and flip $x_{\ell',k}$ from 1 to 0. We further flip all $y_{\ell,j}$ and $y_{\ell',j}$ correspondingly. We have that $EP_\ell(\alpha') = 1$ and $EP_{\ell'}(\alpha') = 0$. Since C doesn't contain any variables related to pigeons ℓ and ℓ', $C(\alpha')$ is still 0. This constructs a witness that ℓ' is critical to C, a contradiction. This finishes the proof. $\square$

It is easy to see that the above reduction is actually a formalization of the width lower bound in $\mathsf{T}^1_2(\alpha)$. Let $n \in \mathsf{Log}$, $M : [2^{O(n)}] \times \mathbb{N} \to \{0,1\}^{\mathrm{poly}(n)}$ be a $\mathsf{PV}(\alpha)$ function symbol that encodes a purported resolution proof of width less than $n/3$, where the second input is a parameter; that is, $M(i,z)$ provides a description of the i-th node in the resolution proof. Let $\mathrm{pf}^{\mathrm{w}}_{\mathrm{EPHP}}(n, M, z)$ denote the $\Pi^b_1(\alpha)$ sentence stating that $M(\cdot, z)$ encodes a valid resolution proof for $\mathrm{EPHP}_{(n+1)\to n}$ (the superscript "w" stands for "width"). Then we have:

Theorem 3.5. *For any* $\mathsf{PV}(\alpha)$ *function symbol* M*, it holds that*

$$\mathsf{T}^1_2(\alpha) \vdash \forall n \in \mathsf{Log}\, \forall z\ \neg \mathrm{pf}^{\mathrm{w}}_{\mathrm{EPHP}}(n, M, z).$$

Proof Sketch. Reason in $\mathsf{T}^1_2(\alpha)$. Assuming $\mathrm{pf}^{\mathrm{w}}_{\mathrm{EPHP}}(n, M, z)$ holds, we will derive a contradiction. A minor technical issue is that we need a PV-definition of the function cri such that the properties (I)-(IV) are true, and that any clause C with $n/3 \leq \mathrm{cri}(C) \leq 2n/3$ has width at least $n/3$. This follows from the formalization of bipartite matching algorithms in PV [LC11].[21]

Let $C_1, C_2, \ldots, C_L$ denote the purported resolution refutation encoded by $M(\cdot, z)$. Using $\Sigma^b_1(\alpha)$-MIN (which is available in $\mathsf{T}^1_2(\alpha)$), there is a smallest integer i such that $\mathrm{cri}(C_i) \geq n/3$. By (II), C_i cannot be an axiom of $\mathrm{EPHP}_{(n+1)\to n}$. By (IV), C_i cannot be a weakening of any clause C_j ($j < i$), as this would contradict the minimality of i. Hence C_i is resolved from C_j and C_k for some $j, k < i$. By (III), $\mathrm{cri}(C_i) \leq \mathrm{cri}(C_j) + \mathrm{cri}(C_k) \leq 2n/3$. Hence the width of C_i is at least $n/3$, contradicting $\mathrm{pf}^{\mathrm{w}}_{\mathrm{EPHP}}(n, M, z)$. $\square$

Monotonized resolution and its width refuter. The first exponential size lower bound for resolution was proven by Haken [Hak85] for the pigeonhole principle PHP. Haken used the so-called "bottleneck counting" argument and the proof was quite involved. A much simpler proof was found by Beame and Pitassi [BP96], where one of the crucial ingredients is the following lemma.

[21] This annoying detail would disappear if we consider the universal variant $\forall\mathsf{T}^1_2(\alpha)$ since these properties are indeed true universal sentences in the standard PV model $\mathbb{N}$ and thus are included in the axioms of $\forall\mathsf{T}^1_2(\alpha)$. As pointed out in [Mül21], it is the provability in *universal* variants of relativized bounded arithmetic that captures reducibility among type-2 TFNP problems.

Lemma 3.6 ([BP96]). *Any resolution refutation of* $\mathrm{PHP}_{(n+1)\to n}$ *contains a clause* C *with* $w(\mathsf{mono}(C)) \geq 2n^2/9$.

Here, for a clause C, $\mathsf{mono}(C)$ is the "monotonized" version of C, which is obtained from C by replacing every negated variable $\overline{x}_{ij}$ with the set of variables $x_{ij'}$ for all $j' \neq j$.[22] Given Lemma 3.6, we could define another variant of the width refuter problem that concerns the original PHP tautologies with no extension variables: we wish to find a clause C such that $\mathsf{mono}(C)$ has a large width, in particular, $w(\mathsf{mono}(C)) \geq 2n^2/9$. The problem is denoted as $\textsc{Refuter}(w_{\mathsf{mono}}(\mathrm{PHP} \vdash_{\mathsf{Res}} \bot) < 2n^2/9)$ and is defined as follows.

Definition 3.7 ($\textsc{Refuter}(w_{\mathsf{mono}}(\mathrm{PHP} \vdash_{\mathsf{Res}} \bot))$)**.** Consider the tautology $\mathrm{PHP}_{(n+1)\to n}$ and let $w := 2n^2/9$. The input instance Π is a purported resolution proof for $\mathrm{PHP}_{(n+1)\to n}$ that consists of clauses $C_{-k}, \dots, C_{-1}, C_0, \dots, C_{L-1}$, where the first $k := n+1+n^2(n-1)/2$ clauses are axioms from $\mathrm{PHP}_{(n+1)\to n}$ and the last clause $C_L = \bot$.

A solution of the given instance is one of the following:

- an index $i \in [L]$ such that $\mathsf{mono}(C_i)$ has at least w literals;[23] or
- an index i such that C_i is an invalid derivation.

The width refuter problem of monotonized resolution proof may not seem natural in the first place. However, converting resolution proof into monotonized resolution proof is an elegant ingredient in the *size lower bounds* of PHP in the proof of Beame and Pitassi [BP96]. Ultimately, our main motivation is for the size refuter of PHP, and the PLS-membership of $\textsc{Refuter}(w_{\mathsf{mono}}(\mathrm{PHP} \vdash_{\mathsf{Res}} \bot) < 2n^2/9)$ is a key step of showing the rwPHP(PLS)-membership of the size refuter ($\textsc{Refuter}(s(\mathrm{PHP} \vdash_{\mathsf{Res}} \bot) < 1.01^n)$). Indeed, the rwPHP($\mathsf{PLS}$)-membership in Section 3.2 uses the following theorem as a black box:

Theorem 3.8. *$\textsc{Refuter}(w_{\mathsf{mono}}(\mathrm{PHP}_{(n+1)\to n} \vdash_{\mathsf{Res}} \bot) < 2n^2/9)$ is in PLS. In particular, there is a uniform decision tree reduction of block-width 3 from this refuter problem to* $\textsc{Iter}$.

The proof is in fact simpler than that of Theorem 3.3, and is a proper constructive translation of the proof by Beame and Pitassi [BP96].

Proof of Theorem 3.8. We call an assignment α to be ℓ-critical if α is a perfect matching between pigeons and holes, except pigeon ℓ is not going anywhere. Formally, for every $i \neq \ell$ we have $x_{i1} \vee \cdots \vee x_{in}$, and for every i, i', j we have $\overline{x}_{ij} \vee \overline{x}_{i'j}$ under α.

Given the definition of ℓ-critical assignments, we define $\mathrm{cri}(\mathsf{mono}(C))$ as follows:

$$\mathrm{cri}(\mathsf{mono}(C)) \coloneqq |\{\ell : \exists \ell\text{-critical assignment } \alpha \text{ that falsifies } \mathsf{mono}(C)\}|.$$

Again, cri has the following four important properties:

- $\mathrm{cri}(\bot) = n+1$;
- $\mathrm{cri}(\mathsf{mono}(C)) \leq 1$ for all axioms C of $\mathrm{PHP}_{(n+1)\to n}$;
- cri is subadditive with respect to resolution derivation, namely, if C is derived from A and B, then $\mathrm{cri}(\mathsf{mono}(C)) \leq \mathrm{cri}(\mathsf{mono}(A)) + \mathrm{cri}(\mathsf{mono}(B))$.

[22] The notion of $\mathsf{mono}(C)$ is tailored to the hard tautology $\mathrm{PHP}_{(n+1)\to n}$. The proof of [BP96] only considers assignments $x \in \{0,1\}^{(n+1)n}$ that defines a bijective mapping from n of the pigeons to all n holes; it is easy to see that every clause C is equivalent to $\mathsf{mono}(C)$ w.r.t. such "critical" assignments x.

[23] Here, unlike the formalization of width lower bounds in Definition 2.6, it is unclear how to syntactically enforce that *the monotonized version* of every clause has width $< w$. Therefore we include the clauses with large monotone width as solutions.

- If C is obtained from a weakening of A, then $\mathrm{cri}(\mathsf{mono}(C)) \le \mathrm{cri}(\mathsf{mono}(A))$.

This time, since we are only concerned with the monotonized version of a clause C and there are no extension variables, it is easier to show that $\mathrm{cri}(\mathsf{mono}(C))$ can be computed in polynomial time.

Claim 3.9. *For any clause C, $\mathrm{cri}(\mathsf{mono}(C))$ can be computed in polynomial time.*

Proof. Fix a pigeon ℓ and we want to check if there is an ℓ-critical assignment for C. We maintain a complete bipartite graph and delete all edges between (ℓ, j) for all hole j. If a variable x_{ij} appears in C, we delete the edge (i, j). Then ℓ-critical assignment exists if and only if there is a perfect matching between pigeons $[n+1] \setminus \{\ell\}$ and holes $[n]$. ◇

Claim 3.10. *For any clause C, the width of $\mathsf{mono}(C)$ is at least $\mathrm{cri}(\mathsf{mono}(C)) \cdot (n - \mathrm{cri}(\mathsf{mono}(C)))$.*

Proof. Let $D \coloneqq \mathsf{mono}(C)$. Let $\mathrm{CriP}(D)$ be the critical pigeons to D, i.e., the set of pigeons $\ell \in [n+1]$ such that there exists an ℓ-critical assignment falsifying D. Then $\mathrm{cri}(\mathsf{mono}(C)) = |\mathrm{CriP}(D)|$. Let $u_1 \in \mathrm{CriP}(D)$ and $u_2 \notin \mathrm{CriP}(D)$. Since $u_1 \in \mathrm{CriP}(D)$, there is a u_1-critical assignment α that falsifies D. Suppose that u_2 is mapped to the hole v_2 in the assignment α. Let β denote the assignment obtained from α by mapping u_1 into v_2 and not mapping u_2 anywhere. Then β is a u_2-critical assignment. Since $u_2 \notin \mathrm{CriP}(D)$, β satisfies D. However, there is only one variable that appears positively in β but negatively in α: namely, x_{u_1,v_2}. Since D is monotone, the literal x_{u_1,v_2} appears in D. Repeating this argument for every $u_1 \in \mathrm{CriP}(D)$ and $u_2 \notin \mathrm{CriP}(D)$, we can see that the width of D is at least $\mathrm{cri}(D) \cdot (n - \mathrm{cri}(D))$. ◇

Given the lemma above, it remains for us to show that finding a clause C such that $\mathrm{cri}(\mathsf{mono}(C)) \in [n/3, 2n/3]$ belongs in PLS. This can be implemented by the standard 1/3-2/3 trick in a potential function way, which is exactly the same as the argument used in the proof of Theorem 3.3. □

3.2 Refuters for Short Resolution Proofs

In this subsection, we investigate Problem 1.1, i.e., the refuter for the following classic resolution *size* lower bound:

Theorem 3.11 ([Hak85])**.** *Any resolution refutation of $\mathrm{PHP}_{(n+1)\to n}$ requires at least $2^{\Omega(n)}$ clauses.*

We show that this refuter problem is in $\mathrm{rwPHP}(\mathsf{PLS})$. As mentioned in Section 1.2.2, this is done by carefully following the proofs of Theorem 3.11. We follow the simplified proof by Beame and Pitassi [BP96], which consists of two steps: a (monotone) width lower bound and a random restriction argument. As the required width lower bound was already studied in Theorem 3.8, we focus on the random restriction argument here.

Let $X := [n+1]$ denote the set of pigeons and $Y := [n]$ denote the set of holes. Recall that the *monotone* version of a clause C, denoted as $\mathsf{mono}(C)$, is obtained from C by replacing every negated variable $\overline{x}_{ij}$ with the set of variables $x_{ij'}$ for all $j' \neq j$.

Definition 3.12. Let $t < n$ be a parameter, $\pi : X \to Y$ be a size-t *matching*, i.e., a partial injective function with $|\mathrm{Domain}(\pi)| = t$. This matching induces a restriction that sets:

- the variable $x_{u,\pi(u)}$ to be 1, for every $u \in \mathrm{Domain}(\pi)$;
- the variable $x_{u,v}$ to be 0, for every $u \in \mathrm{Domain}(\pi)$ and $v \in Y \setminus \{\pi(u)\}$; and
- the variable $x_{u',\pi(u)}$ to be 0, for every $u \in \mathrm{Domain}(\pi)$ and $u' \in X \setminus \{u\}$.

Suppose that $C_0, C_1, \ldots, C_{L-1}$ is a resolution refutation of $\mathrm{PHP}_{(n+1)\to n}$. For each clause C_i, let $\pi(C_i)$ denote the sub-clause of C_i under the above restriction. That is, if π sets some variable in C_i to 1 then $\pi(C_i) = 1$; otherwise $\pi(C_i)$ is obtained from C_i by removing every variable set to 0 by π. Note that the above restriction transforms the unsatisfiable CNF $\mathrm{PHP}_{(n+1)\to n}$ into the unsatisfiable CNF $\mathrm{PHP}_{(n-t+1)\to(n-t)}$. Then, $\pi(C_0), \pi(C_1), \ldots, \pi(C_{L-1})$ is a resolution refutation for $\mathrm{PHP}_{(n-t+1)\to(n-t)}$. Now we claim that the width of the resolution refutation becomes small after being restricted by a random matching π.

Claim 3.13. *If a size-t matching π is chosen uniformly at random over all possible size-t matchings, then with probability at least $1/2$, it holds that for every clause $i \in [L]$, $w(\mathsf{mono}(\pi(C_i))) < W$, where $W := (n+1)^2(1-(1/2L)^{1/t})$.*

Proof. Fix $i \in [L]$, we show that the probability over π that $w(\mathsf{mono}(\pi(C_i))) > W$ is at most $1/(2L)$. The claim then follows from a union bound.

Choose the matching π round by round. There are t rounds, where in each round, we choose an unmatched $u \in X$ and an unmatched $v \in Y$ uniformly at random and match them. If, for the current partial matching π, we have $w(\mathsf{mono}(\pi(C_i))) \geq W$, then the probability that $x_{u,v} \in \mathsf{mono}(\pi(C_i))$ is at least $W/(n+1)^2$. If this is the case, then $\mathsf{mono}(\pi(C_i))$ will become 1 (the always-true clause) after we set $\pi(u) \leftarrow v$, thus it gets "killed." It follows that the probability that C_i never gets killed is at most $(1 - W/(n+1)^2)^t \leq 1/(2L)$. □

Combining Lemma 3.6 and Claim 3.13, we obtain the following size lower bound:

Theorem 3.14. *Any resolution refutation of $\mathrm{PHP}_{(n+1)\to n}$ requires more than $L := 1.01^n$ clauses.*

Proof. Let $t := n/10$, then $W = (n+1)^2(1-(1/2L)^{1/t}) \leq \frac{2}{9}(n-t)^2$. If there is a resolution refutation of $\mathrm{PHP}_{(n+1)\to n}$ of size at most L, then by Claim 3.13, there is a resolution refutation of $\mathrm{PHP}_{(n-t+1)\to(n-t)}$ of monotone width at most $\frac{2}{9}(n-t)^2$. This contradicts Lemma 3.6. □

To derive an upper bound for the complexity of refuter that corresponds to Theorem 3.14, we need a *constructive* version of Claim 3.13. We start by setting up an encoding for the partial matchings and random restrictions that will make it easier to describe our reductions.

A size-t matching can be described by an edge-sequence $(u_0, v_0), (u_1, v_1), \ldots, (u_{t-1}, v_{t-1})$, where for each $j \in [t]$, $u_j \in [n-j+1]$ and $v_j \in [n-j]$. The first edge in this matching connects the u_0-th node in X and the v_0-th node in Y (the first node is the 0-th), the second edge connects the u_1-th unused node in X (i.e., u_1-th node in $X \setminus \{u_1\}$) and the v_1-th unused node in Y, and so on.[24] The space of all possible edge-sequences is denoted by

$$\mathcal{SEQ} := ([n+1] \times [n]) \times ([n] \times [n-1]) \times \cdots \times ([n-t+2] \times [n-t+1]).$$

On the other hand, fix a clause C_i such that $w(\mathsf{mono}(C_i)) \geq W$. Say an edge-sequence s is *bad* for C_i if $w(\mathsf{mono}(s(C_i))) \geq W$, where $\pi_s(C_i)$ is the restriction of C_i under the matching corresponding to π_s. As we argued in Claim 3.13, the number of bad edge-sequences for each C_i is small; we set up another encoding to justify this fact. Any bad edge-sequence can be encoded as a sequence $(e_0, e_1, \ldots, e_{t-1})$,[25] where for each $j \in [t]$, $e_j \in [(n-j+1)(n-j) - W]$. The first edge (u_0, v_0) in this matching is the e_0-th edge, in the lexicographical order, that is not a literal in $\mathsf{mono}(C_i)$; the second edge (u_1, v_1) is the e_1-th edge that still can be chosen (we cannot choose any edge touching either u_0 or v_0) and is not a literal in the current $\mathsf{mono}(\pi_s(C_i))$; and so on. If s is bad, then $w(\mathsf{mono}(\pi_s(C_i)))$ never goes below W, therefore at

[24] Note that the edges are *ordered*, hence each matching corresponds to $t!$ different edge-sequences. In what follows, we will talk about edge-sequences instead of matchings.

[25] To avoid confusion, we use "edge-sequence" to denote elements in $\mathcal{SEQ}$ and "sequence" to denote elements in $\mathcal{BAD}$.

the j-th stage, there are at most $(n-j+1)(n-j)-W$ possible edges to choose. Hence, the space of all possible sequences encoding bad edge-sequences is:

$$\mathcal{BAD} = [(n+1)n - W] \times [n(n-1) - W] \times \cdots \times [(n-t+2)(n-t+1) - W].$$

The following calculation corresponds to a *union bound* over all C_i that the number of bad edge-sequences is small:

$$\begin{aligned}\frac{|\mathcal{BAD}|\cdot L}{|\mathcal{SEQ}|} &= L\cdot\prod_{j=0}^{t-1}\frac{(n+1-j)(n-j)-W}{(n+1-j)(n-j)}\\ &\le L\cdot(1-W/(n+1)^2)^t\\ &\le 1/2. \end{aligned} \tag{5}$$

Fix a clause C_i. For every sequence $b\in\mathcal{BAD}$, let $\mathsf{seq}(C_i,b)\in\mathcal{SEQ}$ denote the bad edge-sequence for C_i corresponding to b; if $\mathsf{seq}(C_i,b)$ does not exist[26], then we denote $\mathsf{seq}(C_i,b)=\bot$. Conversely, any $s\in\mathcal{SEQ}$ is either bad for C_i or not; if s is bad for C_i, then denote $b:=\mathsf{bad}(C_i,s)$ as the sequence $b\in\mathcal{BAD}$ corresponding to s; otherwise we say $\mathsf{bad}(C_i,s):=\bot$. We need the fact that:

Fact 3.15. *Let $s\in\mathcal{SEQ}$ be bad for the clause C_i, then $\mathsf{seq}(C_i,\mathsf{bad}(C_i,s))=s$.*

Now we are ready to establish the rwPHP(PLS) upper bound for the refuter of Theorem 3.14.

Theorem 3.16. *There is a uniform decision tree reduction of block-depth* 3 *from* Refuter(Res(PHP) $>$ 1.01^n) *to* rwPHP(PLS).

That is, there is a uniform decision tree reduction of block-depth 3 *such that the following holds:*

- *given a resolution refutation $\Pi=(C_0,C_1,\dots,C_{L-1})$ for $\mathrm{PHP}_{(n+1)\to n}$, where $L\le 1.01^n$, the reduction computes an instance $(f,\{I_y\},\{g_y\})$ of* rwPHP(PLS)*;*
- *given any valid answer for $(f,\{I_y\},\{g_y\})$, one can compute an invalid derivation $C_i\in\Pi$ in* poly(n) *time.*

Proof. Let $M:=|\mathcal{BAD}|\cdot L$ and $N:=|\mathcal{SEQ}|$, then from Equation 5, we have $M\le N/2$. We will identify numbers in $[M]$ with pairs (i,b) where $i\in[L]$ and $b\in\mathcal{BAD}$, and identify numbers in $[N]$ with edge-sequences in $\mathcal{SEQ}$. The instance $(f,\{I_y\},\{g_y\})$ is defined as follows:

(f) For every $x\in[M]$, we interpret x as a pair (i,b) where $i\in[L]$ and $b\in\mathcal{BAD}$. If $\mathsf{seq}(C_i,b)\ne\bot$, then we let $f(x):=\mathsf{seq}(C_i,b)$; otherwise let $f(x):=0$ (the choice 0 is arbitrary).

(I_y) Fix $y\in[N]=\mathcal{SEQ}$. The edge-sequence y defines a size-t partial matching π_y, which induces a resolution refutation $\Pi|_y=(\pi_y(C_0),\pi_y(C_1),\dots,\pi_y(C_{L-1}))$ of $\mathrm{PHP}_{(n-t+1)\to(n-t)}$. We treat $\Pi|_y$ as a purported resolution refutation with monotone width $<W$; by Theorem 3.8, the problem of finding an invalid derivation in $\Pi|_y$ reduces to Iter via a decision tree of block-width 3. Let I_y be the Iter instance obtained by this reduction.

(g_y) Fix $y\in[N]=\mathcal{SEQ}$ and an answer ans of the Iter instance I_y. Given ans, we can compute a clause that is either invalid or too fat; we then compute $g_y(ans)$ from this clause.

More precisely, we can compute an integer $i\in[L]$ such that either $\mathrm{width}(\mathsf{mono}(\pi_y(C_i)))\ge W$, or $\pi_y(C_i)$ corresponds to an invalid derivation in $\Pi|_y$. In the second case, C_i is also an invalid derivation in Π, thus we can set $g_y(ans)$ to be an arbitrary value (say 0). In the first case, we can set $g_y(ans):=(i,\mathsf{bad}(C_i,y))$.

[26]This may happen when, for example, $w(\mathsf{mono}(C_i))$ is much larger than W and $b_0>(n+1)n-w(\mathsf{mono}(C_i))$.

The block-depth of the decision trees computing f, $\{I_y\}$, and $\{g_y\}$ are 1, 3, and 2 respectively. Clearly, the decision trees are uniform.

Finally, let (y, ans) be a valid solution for the rwPHP(PLS) instance $(f, \{I_y\}, \{g_y\})$ (i.e., $f(g_y(ans)) \neq y$). The edge-sequence $y \in \mathcal{SEQ}$ corresponds to a size-t partial matching π_y and from ans we can read off a clause $i \in [L]$ such that either (1) $\mathrm{width}(\mathsf{mono}(\pi_y(C_i))) \geq W$ or (2) $\pi_y(C_i)$ is an invalid derivation in $\Pi|_y$. If (1) holds, then

$$f(g_y(ans)) = f(i, \mathsf{bad}(C_i, y)) = \mathsf{seq}(C_i, \mathsf{bad}(C_i, y)) = y$$

by Fact 3.15. This contradicts (y, ans) being a valid solution for rwPHP(PLS). It follows that (2) happens and we have found an invalid derivation (namely C_i) in Π. □

As mentioned in Section 1.2.2, the above arguments are essentially a formalization of Haken's lower bounds in the theory $\mathsf{T}^1_2(\alpha) + \mathrm{dwPHP}(\mathsf{PV}(\alpha))$; the decision tree reduction in Theorem 3.16 follows from the witnessing theorem for $\mathsf{T}^1_2(\alpha) + \mathrm{dwPHP}(\mathsf{PV}(\alpha))$ (see Section 2.5.1). In what follows, we make this formalization explicit:

Theorem 3.17. *Let $n \in \mathsf{Log}$, $M : [1.01^n] \times \mathbb{N} \to \{0,1\}^{\mathrm{poly}(n)}$ be a $\mathsf{PV}(\alpha)$ function symbol that encodes a purported resolution proof, where the second input is a parameter. Let $\mathrm{pf}_{\mathrm{PHP}}(n, M, z)$ denote the $\Pi^b_1(\alpha)$ sentence stating that $M(\cdot, z)$ encodes a valid length-1.01^n resolution proof for $\mathrm{PHP}_{(n+1)\to n}$. Then*

$$\mathsf{T}^1_2(\alpha) + \mathrm{dwPHP}(\mathsf{PV}(\alpha)) \vdash \forall n \in \mathsf{Log}\, \forall z\ \neg \mathrm{pf}_{\mathrm{PHP}}(n, M, z).$$

Proof Sketch. Reason in $\mathsf{T}^1_2(\alpha) + \mathrm{dwPHP}(\mathsf{PV}(\alpha))$; assuming $\mathrm{pf}_{\mathrm{PHP}}(n, M, z)$ holds, we will derive a contradiction. We still use C_i to denote the i-th clause of the resolution proof, noticing that given i and z, C_i can be computed by a $\mathsf{PV}(\alpha)$ function (that depends on M). We also use our previous notation such as $\mathcal{BAD}$ and $\mathcal{SEQ}$, and previous parameters $t := n/10$, $L := 1.01^n$, and $W := (n+1)^2(1 - (1/2L)^{1/t}) \leq \frac{2}{9}(n-t)^2$.

First, we use $\mathrm{dwPHP}(\mathsf{PV}(\alpha))$ to select a good random restriction under which each C_i becomes a small-width clause. This random restriction will be encoded as an edge-sequence $s \in \mathcal{SEQ}$. Consider the function

$$\overline{\mathsf{bad}_z}(i, b) := \mathsf{bad}(C_i, b),$$

where $b \in \mathcal{BAD}$ is any sequence encoding a bad edge-sequence. Clearly, $\overline{\mathsf{bad}_z}$ is a function symbol in $\mathsf{PV}(\alpha)$ (which depends on M and has parameter z). By $\mathrm{dwPHP}(\mathsf{PV}(\alpha))$, there is an edge-sequence $s \in \mathcal{SEQ}$ such that for every $i \in [L]$ and $b \in \mathcal{BAD}$, $\overline{\mathsf{bad}_z}(i, b) \neq s$.

Next, we apply s to each clause C_i; denote $\pi_s(C_i)$ the restriction of C_i under the matching corresponding to s. By our choice of s, for every $i \in [L]$, we have $w(\mathsf{mono}(\pi_s(C_i))) < W \leq \frac{2}{9}(n-t)^2$. By Claim 3.10, we have $\mathrm{cri}(\mathsf{mono}(\pi_s(C_i))) > \frac{2(n-t)}{3}$ or $\mathrm{cri}(\mathsf{mono}(\pi_s(C_i))) < \frac{n-t}{3}$ for every $i \in [L]$.

Then we invoke Lemma 3.6 to show that the sequence $\pi_s(C_0), \pi_s(C_1), \ldots, \pi_s(C_{L-1})$ is not a valid resolution proof for $\mathrm{PHP}_{(n-t+1)\to(n-t)}$. Note that this is the step where we use the power of $\mathsf{T}^1_2(\alpha)$. Since $\mathrm{cri}(\mathsf{mono}(\pi_s(C_{L-1}))) = \mathrm{cri}(\bot) = n+1$, by $\Sigma^b_1(\alpha)$-MIN (which is available in $\mathsf{T}^1_2(\alpha)$), there is a smallest integer $i \leq L-1$ such that $\mathrm{cri}(\mathsf{mono}(\pi_s(C_i))) > \frac{2(n-t)}{3}$.

- If $\pi_s(C_i)$ is an axiom, then $\mathrm{cri}(\mathsf{mono}(\pi_s(C_i))) \leq 1$, which is a contradiction.
- If $\pi_s(C_i)$ is a weakening of $\pi_s(C_j)$ where $j < i$, then $\mathrm{cri}(\mathsf{mono}(\pi_s(C_j))) \leq \mathrm{cri}(\mathsf{mono}(\pi_s(C_i)))$, contradicting the minimality of i.
- If $\pi_s(C_i)$ is a resolution of $\pi_s(C_j)$ and $\pi_s(C_k)$ where $j, k < i$, then

  $$\mathrm{cri}(\mathsf{mono}(\pi_s(C_i))) \leq \mathrm{cri}(\mathsf{mono}(\pi_s(C_j))) + \mathrm{cri}(\mathsf{mono}(\pi_s(C_k))).$$

  However, this means either $\mathrm{cri}(\mathsf{mono}(\pi_s(C_j)))$ or $\mathrm{cri}(\mathsf{mono}(\pi_s(C_k)))$ is at least $\frac{n-t}{3}$, a contradiction.

Now, since $\pi_s(C_0), \pi_s(C_1), \ldots, \pi_s(C_{L-1})$ is not a valid resolution proof for $\mathrm{PHP}_{(n-t+1)\to(n-t)}$, we have that $C_0, C_1, \ldots, C_{L-1}$ is not a valid resolution proof of $\mathrm{PHP}_{(n+1)\to n}$ either. This finishes the proof. □

4 Hardness of Refuting Resolution Proofs

In this section, we provide two hardness results for refuter problems: namely, the PLS-hardness for resolution width refuters (Theorem 4.1) and the rwPHP(PLS)-hardness for resolution size refuters (Theorem 4.4). Notably, our hardness results hold for any family of hard tautologies *as long as the lower bounds are true*. This means that "PLS-reasoning" is necessary for proving *any* non-trivial resolution width lower bounds and "rwPHP(PLS)-reasoning" is necessary for proving *any* non-trivial resolution size lower bounds. Of course, this also implies that PLS and rwPHP(PLS) are the *best possible* upper bounds for the refuter problems for *any* non-trivial unsatisfiable family of CNFs (as what we obtained for $\mathrm{PHP}_{(n+1)\to n}$ and more upper bounds in Section 5).

4.1 Hardness of Refuting Narrow Resolution Proofs

We show that the refuter problems for any *true* resolution width lower bounds are PLS-hard. In fact, this holds even for unsatisfiable CNF families that only contain a *single* CNF of *constant* size:

Theorem 4.1. *Let F be any unsatisfiable CNF with a non-trivial resolution width lower bound, i.e., $w(F \vdash_{\mathsf{Res}} \bot) > \mathrm{width}(F)$. Let $\mathcal{F} := \{F\}$ and $w_0 := \mathrm{width}(F)$. Then there is a (uniform) decision-tree reduction of block-depth 2 from* Iter *to* Refuter$(w(F \vdash_{\mathsf{Res}} \bot) \le w_0)$.

Proof. We show a straightforward reduction from the reversed Iter to the width refuter.

Note that F is a fixed CNF so it can be seen as constant size. Hence we can check in constant time that F is unsatisfiable and $\mathrm{width}(F \vdash_{\mathsf{Res}} \bot) > \mathrm{width}(F)$.

Let $S : [L] \to [L]$ such that $S(L) < L$ be any instance of reversed Iter. We will construct a purported resolution refutation Π for F such that any invalid derivation in Π corresponds to an answer for S. Let k be the number of axioms in F. The resolution refutation Π consists of nodes $C_{-k}, \dots, C_{-1}, C_0, \dots, C_L$, where $C_{-k}, \dots, C_{-1}$ are the axioms of F.

For every i such that $S(i) = i$, we let $C_i = C_{-k}$ and define C_i to be a weakening from C_{-k}. This is a valid derivation (and C_i will not be used anymore). The clauses written in all other nodes in $C_1, \dots, C_L$ will be $\bot$. The weakening rules applied among these nodes will encode the successor pointer S:

- For every solution i of the reversed Iter instance (i.e., i such that either $S(i) > i$ *or* ($S(i) < i$ and $S(S(i)) = S(i)$)), the weakening rule applied for C_i will be *invalid*. More specifically, let C_i be a weakening from C_{-k}, then C_i becomes a solution of the Refuter$(w(F \vdash_{\mathsf{Res}} \bot) \le w_0)$ instance.
- For every i such that $S(i) < i$ and $S(S(i)) < S(i)$, we let C_i be a weakening from $C_{S(i)}$. Since both C_i and $C_{S(i)}$ are $\bot$, this is a valid derivation.

This finishes the construction, and the correctness follows from the following two facts immediately: (1) there are no nodes whose width is larger than $\mathrm{width}(F)$; (2) a resolution derivation is invalid if and only if it is a solution of the given reversed Iter instance. The block-depth of our reduction is 2, as we only need to query $S(i)$ and $S(S(i))$. □

Note that the reduction in Theorem 4.1 also works for a family of CNFs $\{F_n\}_{n\in\mathbb{N}}$ with non-trivial width lower bound. Therefore, combined with the PLS-membership results in Section 3.1, we obtain:

Theorem 4.2. *Refuter$(w(\mathrm{EPHP} \vdash_{\mathsf{Res}} \bot) < n/3)$ is PLS-complete.*

Note that every clause generated in the reduction in Theorem 4.1 has monotone width $O(n)$. Hence the same proof also shows the PLS-hardness of *monotone* width refuters (as in Theorem 3.8):

Theorem 4.3. *Refuter$(w_{\mathsf{mono}}(\mathrm{PHP}_{(n+1)\to n} \vdash_{\mathsf{Res}} \bot) < 2n^2/9)$ is PLS-complete.*

4.2 Hardness of Refuting Short Resolution Proofs

In this section, we show the rwPHP(PLS)-hardness of refuters for resolution *size* lower bounds. In particular, for any family of unsatisfiable CNF formulas $\{F_n\}_{n\in\mathbb{N}}$ that requires resolution size $> s_F(n)$, if $s_F(n)$ is not too small, then rwPHP(PLS) reduces to the problem $\textsc{Refuter}(s(F_n \vdash_{\mathsf{Res}} \bot) \le s_F(n))$. This result and our rwPHP(PLS) upper bounds (Theorems 3.16, 5.4, 5.12, and 5.13) complement each other by showing that rwPHP(PLS) is the tightest complexity class in all these results.

Recall that an rwPHP(PLS) instance consists of $\left(f, \{I_y\}_{y\in[2M]}, \{g_y\}_{y\in[2M]}\right)$, where:

- $f : [M] \to [2M]$ is a purported "surjection";
- for each $y \in [2M]$, $I_y := (L, S_y)$ is an instance of $\textsc{Iter}$, where $S_y : [L] \to [L]$; and
- $g_y : [L] \to [M]$ maps solutions of I_y to integers in $[M]$.

We now state and prove the main theorem of this subsection.

Theorem 4.4. *There is a universal constant $C \ge 2$ such that the following holds. Let $L, M \ge 1$ be the parameters of* rwPHP(PLS) *instances and $n \ge 1$.*

For every unsatisfiable CNF formula F over n variables and parameter $s_F \ge C \cdot (nLM + |F|)$ such that every resolution refutation of F requires more than s_F clauses, there is a decision tree reduction of block-depth $O(n)$ from a rwPHP(PLS) *instance to a $\textsc{Refuter}(s(F \vdash_{\mathsf{Res}} \bot) \le s_F)$ instance.*

Proof. Let $\left(f, \{I_y\}_{y\in[2M]}, \{g_y\}_{y\in[2M]}\right)$ be an instance of rwPHP(PLS) and we will reduce it to an instance of $\textsc{Refuter}(s(F \vdash_{\mathsf{Res}} \bot) \le s_F)$. Our goal is to construct a size-s_F resolution refutation Π for F such that any illegal derivation in Π corresponds to a valid solution to the rwPHP(PLS) instance.

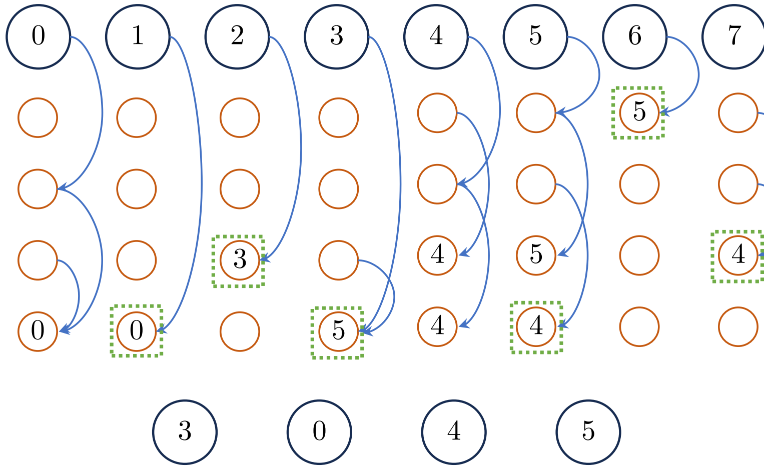
The nodes in Π are partitioned into $n+1$ layers, numbered from layer 0 to layer n. Each layer $t \in [n+1]$ has either two rows or one row: When layer t has two rows, we denote the nodes in the first row by $\left\{D^t_{(y,a)}\right\}$ and those in the second row by $\left\{E^t_i\right\}$; when layer t has only one row, we denote the nodes by $\left\{E^t_i\right\}$. (Therefore, $\left\{E^t_i\right\}$ always denote the *last* row of layer t.) After all these $n+1$ layers of clauses, we put the axioms of F at the very end. It is easy to translate a resolution refutation in this layout into one in the format of Definition 2.5 by decision trees of block-depth 1.

The construction. The layer 0 has one node $E^0_0 := \bot$. For each t from $1, \ldots, n$:

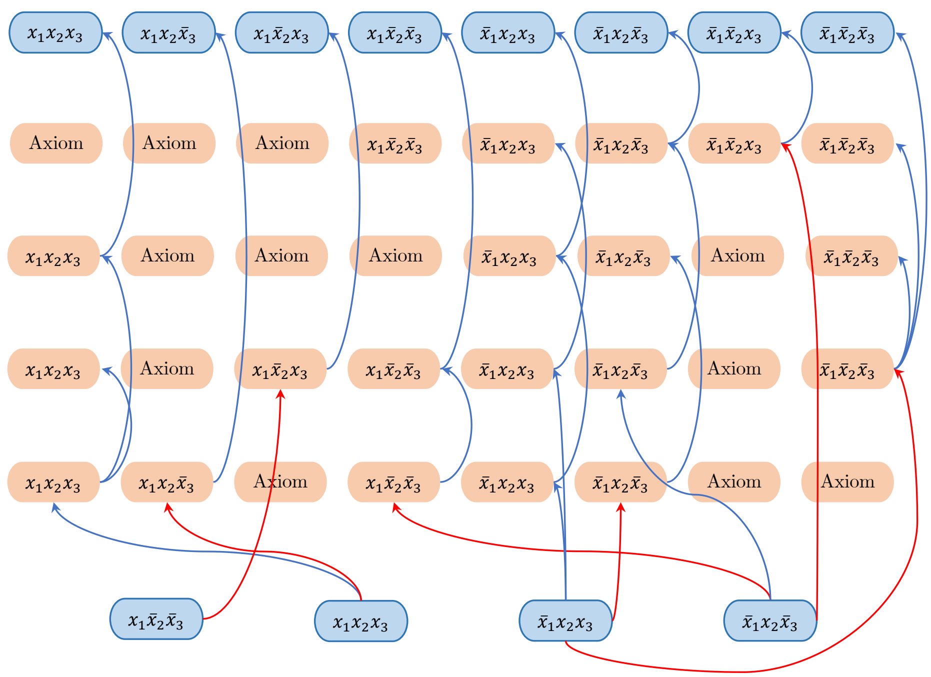
1. Let $E^{t-1}_0, \ldots, E^{t-1}_{k-1}$ be the nodes on the last row of layer $t-1$; we will always guarantee that $k \le M$.
2. **Case 1:** $2k \le M$**.** In this case, layer t will only have one row of nodes, defined as follows. For every node E^{t-1}_i on layer $t-1$, we generate 2 nodes E^t_{2i} and E^t_{2i+1} on layer t, where the clauses written are $E^t_{2i} = E^{t-1}_i \vee x_t$ and $E^t_{2i+1} = E^{t-1}_i \vee \overline{x}_t$. We define E^{t-1}_i to be <u>resolved</u> from E^t_{2i} and E^t_{2i+1}.
3. **Case 2:** $2k > M$**.**
   (a) First, prepare $2M$ nodes $D^t_{(0,0)}, D^t_{(1,0)}, \ldots, D^t_{(2k-1,0)}$. It would be instructive to think of $\{D^t_{(y,a)} : a \in [L]\}$ for each fixed y as a chain and we are now preparing the heads of these $2M$ chains. In what follows, we denote $C_i = D^t_{(i,0)}$ for ease of notation.

   For each $i \in [k]$, let $C_{2i} = E^{t-1}_i \vee x_t$, $C_{2i+1} = E^{t-1}_i \vee \overline{x}_t$, and define E^{t-1}_i to be <u>resolved</u> from C_{2i} and C_{2i+1}. We make sure that there are *exactly* $2M$ clauses on the first row by making several copies of C_{2k-1}: for each $i \in \{2k, \ldots, 2M-1\}$, let $C_i = C_{2k-1}$.

   (b) Generate M nodes on the second row of layer t: for every $i \in [M]$, let $E^t_i := C_{f(i)}$. (Intuitively, if $f : [M] \to [2M]$ *were* a surjection, then every node in $\{C_i\}_{i\in[2M]}$ *would* appear in $\{E^t_i\}_{i\in[M]}$.)



(a) This is an rwPHP(PLS) instance with $M = 4$, compressing the top $2M$ elements to the bottom M elements. The bottom four points represent the function $f : [M] \to [2M]$; i.e., in this example, $f(0) = 3, f(1) = 0, f(2) = 4, f(3) = 5$. Every column is a PLS instance (every vertex without an outgoing edge has a self-loop). Every sink is a solution of the corresponding PLS instance, on which we have $g_y : [L] \mapsto [M]$. The number on every sink represents $f(g_y(\cdot))$, which if different from y, would be a solution of the whole rwPHP(PLS) instance. In this figure, every solution is marked with a dotted green box.



(b) Part of the constructed resolution derivation Π. Initially, we have $2M = 8$ clauses. At the bottom, we have $M = 4$ clauses, which exactly correspond to the 3rd, 0th, 4th, and 5th clauses above. For every node a in a PLS instance, if it is a self-loop, then we let the clause be a weakening from some axiom (and it would never be used again). If a is a solution of the PLS instance, we let it be the weakening of clause $g_y(a)$ at the bottom. Otherwise, we let it be the weakening of $S_y(a)$. All blue arrows are valid weakenings and *all red arrows are invalid weakenings*. The invalid weakenings here will be the (only) solutions to the refuter problem.

Figure 4: The gadget to embed an rwPHP(PLS) instance.

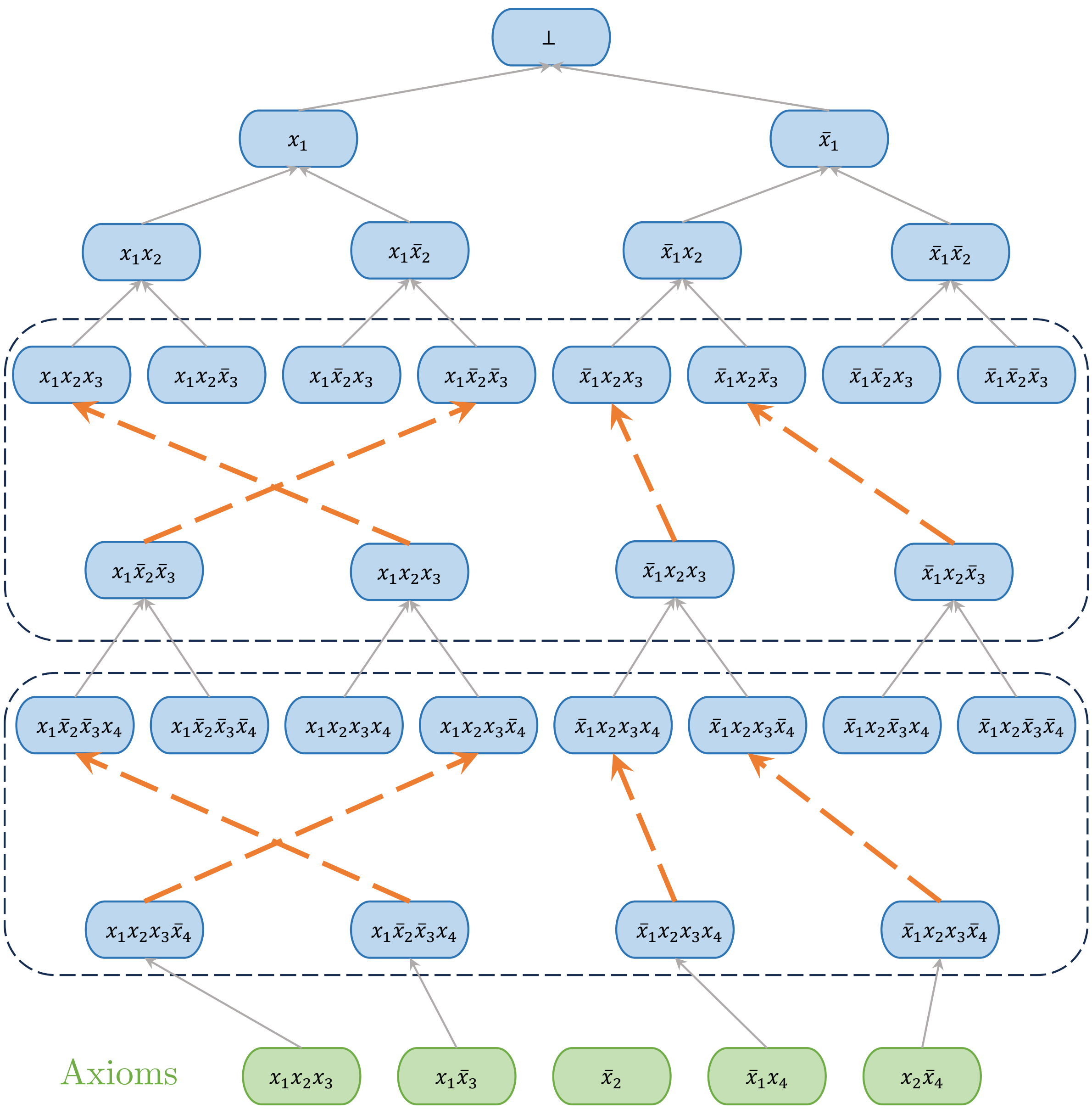


Figure 5: An illustration of our reduction from rwPHP(PLS) to the size refuter problem. All gray arrows are valid resolution derivations (and the last layer is weakening from axioms). Every dashed box uses the gadget to embed an rwPHP(PLS) instance that enforces every layer to have at most $2M$ clauses. Thus the only possible invalid derivations are those inside the gadget which, once found, would directly imply a solution of the original rwPHP(PLS) instance. The overall reduction will produce a purported resolution refutation of size $O(nLM + |F|)$.

(c) Now, for each $y \in [2M]$, we "link" the node $C_y = D^t_{(y,0)}$ to their corresponding $E^t_{f^{-1}(y)}$ on the second row, using the Iter instance I_y. Recall that for each $y \in [2M]$, I_y consists of a function $S_y : [L] \to [L]$, and solutions of I_y are those $a \in [L]$ such that

$$\text{either } S_y(a) < a \text{ or } (S_y(a) > a \text{ and } S_y(S_y(a)) = S_y(a)).$$

As a special case, if $a = 0$ and $S_y(0) = 0$, then 0 also counts as a solution.

Every clause on the chain $\{D^t_{(y,a)} : a \in [L]\}$ will be equal to $C_y = D^t_{(y,0)}$ except those on a "junk" node a such that $S_y(a) = a$ (see Case 2 (c) ii.); these clauses will be weakenings of each other, and the instance I_y dictates the structure of the weakening relationship. For every $a \in [L]$, the node $D^t_{(y,a)}$ is defined as follows:

i. If a is a solution of I_y, then $g_y(a)$ is a purported pre-image of y. The clause written on $D^t_{(y,a)}$ is equal to C_y and we define it to be a weakening of $E^t_{g_y(a)}$. (Note that $E^t_{g_y(a)} = C_{f(g_y(a))}$ by definition, meaning that if the weakening from $E^t_{g_y(a)}$ to $D^t_{(y,a)}$ is an illegal derivation, then $f(g_y(a)) \neq y$.)

ii. If a is not a solution and $S_y(a) = a$, then the clause written on $D^t_{(y,a)}$ is defined to be the weakening of an arbitrary axiom in F (say the first axiom). The node $D^t_{(y,a)}$ is considered a "junk" node and will never be used later.

iii. Otherwise, we have $S_y(a) > a$. The clause written on $D^t_{(y,a)}$ is equal to C_y and we define it to be a weakening of $D^t_{(y,S_y(a))}$.

After constructing all these nodes above, we put the axioms of F at the very end. Each clause E^n_i in the last row of layer n will be a weakening of some axiom in F. In particular, note that each E^n_i has exactly n literals (this can be easily seen from induction) and therefore is satisfied by exactly one assignment α_i. Recall that F is an unsatisfiable CNF formula, so for each E^n_i, there exists an axiom A in F such that α_i falsifies A; hence we can define E^n_i to be a weakening of A.

This finishes the construction.

The above construction gives a resolution refutation Π for F that has size $s_F := O(nLM + |F|)$. The only place in Π where illegal derivations might occur is in Case 2 (c) i. when we define $D^t_{(y,a)}$ to be a weakening of $E^t_{g_y(a)}$. If this is an illegal derivation, then $f(g_y(a)) \neq y$, which means that we have found a valid solution for the rwPHP(PLS) instance. Therefore, the above construction is a correct reduction from rwPHP(PLS) to REFUTER($s(F \vdash_{\mathsf{Res}} \bot) > s_F$), as long as $s_F > C \cdot (nLM + |F|)$ for some large universal constant C.

Finally, we analyze the query complexity of this reduction. It suffices to show that every node $D^t_{(y,a)}$ and E^t_i can be computed in block-depth $O(n)$ from the input rwPHP(PLS) instance. Note that to compute one node, we need to calculate both its origin (i.e., resolved or weakening from which node) and the clause written on it. We use induction on t to show that every clause in layer t can be computed in block-depth $c \cdot (t+1)$ for some universal constant c. Fix a layer t and we argue as follows.

- **(Base case)** If layer t contains only one row, then we can read off the clause E^t_i from the binary representation of i; the node E^t_i is always resolved from E^{t+1}_{2i} and E^{t+1}_{2i+1} (if layer $t+1$ also contains only one row) or C^{t+1}_{2i} and C^{t+1}_{2i+1} (otherwise).

- **(Induction step)** If layer t contains two rows, then we argue as follows.

  1. For $i < 2k$, depending on the parity of i, we have that the clause written on $D^t_{(i,0)}$ is either $E^{t-1}_{\lfloor i/2 \rfloor} \vee x_t$ or $E^{t-1}_{\lfloor i/2 \rfloor} \vee \overline{x}_t$. For $i \geq 2k$, the clause written on $D^t_{(i,0)}$ is always equal to $D^t_{(2k-1,0)}$. For every $i \in [2M]$ and $a \in [L]$, the clause written on $D^t_{(i,a)}$ is either equal to the clause written on $D^t_{(i,0)}$, or equal to some axiom of F, and this can be decided in block-depth 2 (see Case 2 (c) ii.). Since it takes block-depth ct to compute $E^{t-1}_{\lfloor i/2 \rfloor}$, it takes block-depth $ct+2$ to compute the clause written on $D^t_{(i,a)}$.

  2. Every $D^t_{(y,a)}$ (for $y \in [2M]$ and $a \in [L]$) belongs to one of the following three cases:
     - if a is a solution of I_y, then $D^t_{(y,a)}$ is a weakening of $E^t_{g_y(a)}$;
     - if $a \neq 0$ and $S_y(a) = a$, then $D^t_{(y,a)}$ is a weakening of some axiom in F and is a "junk" node;

– otherwise, $D^t_{(y,a)}$ is a weakening of $D^t_{(y,S_y(a))}$.

Therefore, we can use $O(1)$ additional block-depth to determine all information regarding $D^t_{(y,a)}$.

3. Let $i \in [M]$, then $E^t_i = D^t_{(f(i),0)}$, and E^t_i is either resolved from C^{t+1}_{2i} and C^{t+1}_{2i+1} (when $t < n$) or a weakening of some axiom (when $t = n$). This can be computed in constant additional block depth.

It follows that Π can be computed from our input rwPHP(PLS) instance in block-depth $O(n)$. $\square$

Corollary 4.5. *Refuter*$(s(\mathrm{PHP}_{(n+1)\to n} \vdash_{\mathsf{Res}} \bot) \leq 1.01^n)$ *is complete for* rwPHP(PLS).

Proof Sketch. By combining Theorem 3.16 and Theorem 4.4. Note that the reductions have poly(n) block-depth and each block contains poly(n) bits, therefore they are polynomial-time (many-one) reductions. $\square$

The above hardness result in TFNP can be interpreted as a reversal result in bounded reverse mathematics as well. To state this reversal result, we define the following two families of $\forall\Sigma^b_1(\alpha)$-sentences. For $\mathsf{PV}(\alpha)$ function symbols F, I, G, let rwPHP(PLS)(F, I, G) denote the natural $\forall\Sigma^b_1(\alpha)$-sentence expressing the existence of a solution for the rwPHP(PLS)-instance defined by (F, I, G):

- For every auxiliary input z and every t, L, there exists $y \in [2t]$ and $ans \in [L]$ such that ans is a PLS solution for the Iter instance $I_{z,y} : [L] \to [L]$ and that $(G_{z,y}(ans) > t$ or $F_z(G_{z,y}(ans)) \neq y)$.

Similarly, let $\mathsf{Haken}(\alpha)$ denote the family of $\forall\Sigma^b_1(\alpha)$-sentences consisting of

$$\forall n \in \mathsf{Log}\, \forall z\ \neg \mathrm{pf}_{\mathrm{PHP}}(n, M, z)$$

for every $\mathsf{PV}(\alpha)$ function symbol M with parameter z. (Recall that $\mathrm{pf}_{\mathrm{PHP}}(n, M, z)$ is defined in Theorem 3.17.)

Theorem 4.6. *For every* $\mathsf{PV}(\alpha)$ *function symbols* F, I, G, $\mathsf{PV}(\alpha)+\mathsf{Haken}(\alpha)$ *proves* rwPHP(PLS)(F, I, G).

Proof Sketch. Argue in $\mathsf{PV}(\alpha)$. Let Π be the purported resolution proof for PHP as constructed in the proof of Theorem 4.4 from (F, I, G), then Π can be expressed as a $\mathsf{PV}(\alpha)$ function symbol (that depends on F, I, and G). From $\mathsf{Haken}(\alpha)$, we know that there exists an illegal derivation in Π. This illegal derivation can only occur in Case 2 (c) i., and hence it points to a weakening from some $D^t_{(y,ans)}$ to some $E^t_{g_y(ans)}$. This means the existence of a solution (y, ans) of the rwPHP(PLS)-instance (F, I, G). $\square$

We remark that like Theorem 4.4, the proof of the above theorem does not depend on the hard tautology being PHP.

We finish this section by the following nice-looking characterization of $\forall\Sigma^b_1$-consequences (i.e., provably total NP search problems) of $\mathsf{T}^1_2 + \mathrm{dwPHP}(\mathsf{PV})$:

Corollary 4.7. *1. Refuter*$(s(\mathrm{PHP}_{(n+1)\to n} \vdash_{\mathsf{Res}} \bot) \leq 1.01^n)$ *is complete for the class of* NP *search problems provably total in* $\mathsf{T}^1_2 + \mathrm{dwPHP}(\mathsf{PV})$.

2. A $\forall\Sigma^b_1(\alpha)$*-sentence is provable in the theory* $\mathsf{T}^1_2(\alpha) + \mathrm{dwPHP}(\mathsf{PV}(\alpha))$ *if and only if it is provable in the theory* $\mathsf{PV}(\alpha) + \mathsf{Haken}(\alpha)$.

5 Refuters for Other Formulas

This section presents additional upper bounds for the refuter problems associated with resolution lower bounds. We start with a *universal* PLS upper bound for width refuters, showing that any resolution width lower bound *that is true* can be refuted in non-uniform PLS. Then, we provide further examples of resolution lower bounds proven by "random restriction + width lower bounds" and show that the refuter problems for these lower bounds are in rwPHP(PLS). In particular, we present the following three classic resolution lower bounds and show that the refuter problems for all of them are in rwPHP(PLS):

(a) size-width tradeoffs from XOR-lifting [DR03, Kra11a] (Section 5.2);

(b) exponential size lower bounds for the Tseitin formulas [Urq87, Sch97] (Section 5.3); and

(c) exponential size lower bounds for random k-CNFs [CS88, BP96] (Section 5.4).

We believe that the case of random k-CNFs is especially compelling: the *vast majority* of resolution lower bounds have refuters in rwPHP(PLS)!

5.1 Universal Refuters for *Every* Narrow Resolution Proof

This subsection shows a very general result: For *every* (possibly non-uniform) family of unsatisfiable CNFs $\mathcal{F} = \{F_n\}$ and *every* sequence of integers $\{w_n\}$, if for every $n \in \mathbb{N}$, w_n is indeed a resolution width lower bound for F_n, then the refuter problem corresponding to this width lower bound is in PLS under non-uniform decision tree reductions.

We note that such a membership result is *inherently* non-uniform since it is crucial to consider algorithms with unlimited *computational* power. For example, in general, it is not obvious how to decide if w_n is a valid resolution width lower bound for F_n (although it is certainly computable with unlimited computational power). In fact, even checking if F_n is unsatisfiable is itself NP-complete. On the other hand, even though these two tasks are computationally hard, they only require querying at most poly(n) bits of the given resolution proof. Thus, we can still consider these refuter problems in $\mathsf{TFNP}^{\mathsf{dt}}$ and study its query complexity in the non-uniform setting.

Theorem 5.1. *Let $\mathcal{F}$ be any (possibly non-uniform) family of unsatisfiable CNFs with polynomially many clauses. Let $w_0 = w(\mathcal{F} \vdash_{\mathsf{Res}} \bot)$. Then there exists a (non-uniform) decision-tree reduction of block-depth 2 from* Refuter$(w(\mathcal{F} \vdash_{\mathsf{Res}} \bot) < w_0)$ *to* Iter.

Proof. Consider any instance of Refuter$(w(\mathcal{F} \vdash_{\mathsf{Res}} \bot) < w_0)$. Recall from Definition 2.5 that the instance is a purported resolution refutation Π that consists of clauses $C_{-k}, \dots, C_{-1}, C_0, \dots, C_{L-1}$ where $C_{-k}, \dots, C_{-1}$ are the axioms of $\mathcal{F}$ and $C_{L-1} = \bot$. Also, recall that we syntactically ensure the width of Π is $< w_0$ by only allocating $w_0 - 1$ literals for each clause. The key point in the reduction is that, for any clause C_i that is resolved from C_{j_1} and C_{j_2}, if $\text{width}(F \vdash_{\mathsf{Res}} C_i) \geq w_0$, then either $\text{width}(F \vdash_{\mathsf{Res}} C_{j_1}) \geq w_0$, or $\text{width}(F \vdash_{\mathsf{Res}} C_i) \geq w_0$.

The length of the reduced reversed Iter instance is exactly L. Next, we define the successor pointers: for every $i \in [L]$, let C_i be the i-th clause and C_{j_1} and C_{j_2} with $j_1 < j_2 < i$ be the two clauses from which C_i is resolved, then

$$S(i) := \begin{cases} i & \text{if } \text{width}(F \vdash_{\mathsf{Res}} C_i) < w_0; \\ j_1 & \text{if } \text{width}(F \vdash_{\mathsf{Res}} C_{j_1}) \geq w_0; \\ j_2 & \text{otherwise.} \end{cases}$$

Clearly, this is a query-efficient reduction with block-depth 2. It is not time-efficient because it needs to compute whether $\text{width}(F \vdash_{\mathsf{Res}} C) < w_0$ for some clauses C.

To show correctness, we consider any possible solution of the constructed reversed Iter. For any i such that $S(i) > i$, we have either $j_1 > i$ or $j_2 > i$, which means that C_i is an invalid derivation. Now consider any i such that $S(i) < i$ and $S(S(i)) = S(i)$. Since $S(i) < i$, we have that $\text{width}(F \vdash_{\mathsf{Res}} C_i) \geq w_0$. Since $S(S(i)) = S(i)$, we have both $\text{width}(F \vdash_{\mathsf{Res}} C_{j_1}) < w_0$ and $\text{width}(F \vdash_{\mathsf{Res}} C_{j_2}) < w_0$. Thus, the resolution step from C_{j_1} and C_{j_2} to C_i must be an invalid derivation. This finishes the proof. $\square$

Note that Theorem 4.1 already shows a universal PLS-hardness, which even holds for uniform reduction. Combining the the PLS-membership (Theorem 5.1) above, we have the following corollary.

Corollary 5.2. *Let $\mathcal{F}$ be any (possibly non-uniform) family of unsatisfiable CNFs with polynomially many clauses. Let $w_0 := w(\mathcal{F} \vdash_{\mathsf{Res}} \bot)$. Then* Refuter$(w(\mathcal{F} \vdash_{\mathsf{Res}} \bot) < w_0)$ *is* PLS*-complete under (non-uniform) decision tree reductions.*

5.2 Refuters for XOR-Lifted Lower Bounds

We show that for a large family of resolution lower bounds proved by *lifting theorems*, their corresponding refuter problems are in rwPHP(PLS).

Given an unsatisfiable CNF F which is hard for a "weak" proof system, a *lifting theorem* produces another unsatisfiable CNF F' (typically by composing F with some *gadgets*) that is hard for a "stronger" proof system. Lifting is a very influential technique for proving lower bounds in proof complexity, see e.g. [HN12, GP18b, dRNV16, GGKS20, dRMN+20]. This subsection examines one of the simplest lifting theorems for proving lower bounds for resolution, which originated from the technique of "relativization" [DR03, Kra11a] (see also [Kra19, Section 13.2]).

Let $F(z_1, z_2, \ldots, z_n)$ be an unsatisfiable CNF. Roughly speaking, the CNF $F \circ \mathsf{XOR}$ is obtained by replacing each variable z_i with $x_i \oplus y_i$, where x_i and y_i are new variables corresponding to z_i. More formally, the formula $F \circ \mathsf{XOR}$ takes $2n$ Boolean variables $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_n$ as inputs. Denoting $z_i^b = z_i$ if $b = 1$ and $\overline{z}_i$ if $b = 0$; each width-d clause

$$z_{i_1}^{b_1} \vee z_{i_2}^{b_2} \vee \cdots \vee z_{i_d}^{b_d}$$

becomes a set of 2^d width-$2d$ clauses

$$\left\{ \left(x_{i_1}^{r_1 \oplus 1} \vee y_{i_1}^{b_1 \oplus r_1}\right) \vee \left(x_{i_2}^{r_2 \oplus 1} \vee y_{i_2}^{b_2 \oplus r_2}\right) \vee \cdots \vee \left(x_{i_d}^{r_d \oplus 1} \vee y_{i_d}^{b_d \oplus r_d}\right) \right\}_{r_1, r_2, \ldots, r_d \in \{0,1\}}.$$

A classical lifting theorem states that if F requires large resolution *width*, then $F \circ \mathsf{XOR}$ requires large resolution *size*. Here, the "weak" proof system is *narrow* resolution and the "strong" proof system is *short* resolution. More formally:

Theorem 5.3. *Let F be an unsatisfiable CNF that requires resolution width $\geq w$, then $F \circ \mathsf{XOR}$ requires resolution size $\geq 2^{w/3}$.*

The classical proof of this theorem goes through a random restriction argument. Let Π be a purported resolution proof of $F \circ \mathsf{XOR}$ of length $L < 2^{w/3}$. Consider a random restriction ρ as follows: For each index i, with probability $1/2$, we set $\rho_{x_i} = 0/1$ uniformly at random and $\rho_{y_i} = *$; otherwise, we set $\rho_{y_i} = 0/1$ uniformly at random and $\rho_{x_i} = *$. By the construction above, $\Pi|_\rho$ is a resolution proof of F up to substituting some variables by their negations, for any ρ in the support. Moreover, for any clause $C \in \Pi$ of width at least t, C is killed by a random restriction ρ (i.e., $C|_\rho \equiv 1$) w.p. at least $1 - 2^{-\Omega(t)}$. By a union bound over all $L < 2^{w/3}$ clauses in Π, it follows that there is a random restriction ρ killing every clause of width $> w$ in Π. Therefore, $\Pi|_\rho$ is a resolution refutation for F, contradicting the width lower bound for F.

To obtain a reduction to rwPHP(PLS), it would be helpful to rephrase the above proof as a *compression* argument:

Proof of Theorem 5.3. Let $\mathcal{R}$ be the space of the random restrictions in the above proof. Each $\rho \in \mathcal{R}$ can be described in $2n$ bits:

- For each index i, if $\rho_{x_i} = *$, then we write down $0y_i$ (the first bit being 0 indicates that x_i is set to $*$, and the second bit encodes y_i); otherwise we write down $1x_i$.

We call the above encoding the *standard encoding* of a restriction; this encoding is a bijection between $\mathcal{R}$ and $\{0,1\}^{2n}$, showing that $|\mathcal{R}| = 4^n$.

In contrast, if C is a clause of width w and $\rho \in \mathcal{R}$ is a restriction such that $C|_\rho \neq 1$, then given C, such a ρ can be described in $(\log_2 3)w + 2(n-w) < 2n$ bits. This is because for each literal in C (say x_i, y_i, $\overline{x}_i$, or $\overline{y}_i$), if this literal is not simplified to 1, then there are only 3 possible choices for (ρ_{x_i}, ρ_{y_i}); for example, if this literal is x_i, then (ρ_{x_i}, ρ_{y_i}) might be one of $(0,*)$, $(*,0)$, or $(*,1)$, but never $(1,*)$. We call this the *short* encoding of ρ w.r.t. C; note that this encoding only works when $C|_\rho \neq 1$.

Now, let $\Pi = (C_0, C_1, \dots, C_{L-1})$ be a resolution refutation of $F \circ \mathsf{XOR}$ with $L < 2^{w/3}$ clauses. Let $f : [L] \times [3^w 4^{n-w}] \to [4^n]$ be the function that on input (i, ρ'), where $i \in [L]$ and ρ' is the short encoding of a restriction w.r.t. C_i, outputs the standard encoding of ρ' in $\{0,1\}^{2n}$. Since

$$L \times 3^w 4^{n-w} \leq 2^{w/3} \cdot 4^n (3/4)^w < 0.99 \cdot 4^n \text{ (whenever } w \geq 1),$$

it follows from the *dual weak pigeonhole principle* that there exists a $\rho \in \{0,1\}^{2n}$ outside the range of f. This restriction ρ simplifies Π into a width-w resolution proof of F.

In conclusion, if there is a resolution refutation of $F \circ \mathsf{XOR}$ with $< 2^{w/3}$ clauses, then *by the dual weak pigeonhole principle*, there is a resolution refutation of F with width $< w$, contradicting the assumed hardness of F. □

Now we are ready to show the following result: for every unsatisfiable CNF of the form $F \circ \mathsf{XOR}$ whose resolution size lower bound can be derived from Theorem 5.3, the refuter problem for this resolution size lower bound is in rwPHP($\mathcal{P}$), where $\mathcal{P}$ corresponds to the refuter problem for the width lower bound for F. Since the refuter problem corresponding to *every* resolution width lower bound admits a non-uniform reduction to PLS (Corollary 5.2), the refuter problems corresponding to size lower bounds for $F \circ \mathsf{XOR}$ non-uniformly reduce to rwPHP(PLS) as well. Even if we restrict ourselves to uniform reductions, the refuter problems for many interesting width lower bounds reduce to PLS (such as Theorem 3.3), thus the refuter problems for size lower bounds for the corresponding lifted CNFs also reduce to rwPHP(PLS).

Theorem 5.4. *Let $\{F_n\}$ be a family of unsatisfiable CNFs, $w(n)$ be a width lower bound for F_n, and $\mathcal{P}$ denote the problem* Refuter$(w(F_n) > w(n))$*. Then there is a decision tree reduction from* Refuter$(s(F_n \circ \mathsf{XOR}) < 2^{w(n)/3})$ *to* rwPHP($\mathcal{P}$) *with block-depth* 1.

Proof. Let Π be the input instance of Refuter$(s(F_n \circ \mathsf{XOR}) < 2^{w(n)/3})$. That is, $\Pi = (C_0, C_1, \dots, C_{L-1})$ is a purported resolution refutation of $F_n \circ \mathsf{XOR}$ with $L < 2^{w(n)/3}$ clauses, and we want to find an invalid derivation in Π.

Let $f : [0.99N] \to [N]$ be the function defined in the proof of Theorem 5.3, where $N := 4^n$. That is, given a pair (i, ρ') where $i \leq L$ and ρ' is the short encoding of a restriction w.r.t. C_i, $f(i, \rho')$ is the standard encoding of this restriction. The range of f consists of (the standard encodings of) all *bad* restrictions, i.e., those that do *not* simplify Π to a width-w resolution refutation.

Given any restriction $\rho \in \{0,1\}^{2n}$, let $\Pi|_\rho$ denote the restriction of Π under ρ where we force every clause to have width at most w; $\Pi|_\rho$ is a purported width-w resolution refutation for F_n. In particular, for each (ρ, i), the i-th clause of $\Pi|_\rho$ is equal to the restriction of C_i under ρ, truncated at width w. Note that if F_n indeed requires resolution width $> w$, then $\Pi|_\rho$ must be an *invalid* resolution refutation of F_n. Suppose that the i-th clause in $\Pi|_\rho$ is derived illegally, then it could be for the following two reasons:

- Either the derivation of C_i in Π is already illegal;
- or the width of $C_i|_\rho$ is actually $> w$ and the i-th clause in $\Pi|_\rho$ is illegal because it was truncated.

Let $g'_{\rho,i}$ denote the short encoding of ρ w.r.t. C_i, and define $g_{\rho,i} := (i, g'_{\rho,i})$. In the second case, we have a clause C_i and a restriction ρ such that $C_i|_\rho \neq 1$ (in fact, the width of $C_i|_\rho$ is large), thus the short encoding makes sense and, indeed, $f(g_{\rho,i}) = \rho$. In any case, if the short encoding does not make sense (i.e., $C_i|_\rho = 1$), we can set $g_{\rho,i}$ arbitrarily.

Now we have all the ingredients needed in our reduction from the problem of finding an invalid derivation in Π to rwPHP($\mathcal{P}$):

1. a purported "surjection" $f : [0.99N] \to [N]$;
2. a $\mathcal{P}$ instance $\Pi|_\rho$ for every $\rho \in [N]$;
3. for every $\rho \in [N]$ and every solution i of $\Pi|_\rho$ (as a $\mathcal{P}$ instance), a number $g_{\rho,i}$ pointing to a purported pre-image in $f^{-1}(\rho)$.

Every entry $f(i, \rho')$, $\Pi|_\rho(i)$, and $g_{\rho,i}$ only depend on C_i and ρ, thus are computable by a decision tree of block-depth 1.

A solution of the above rwPHP($\mathcal{P}$) instance consists of a restriction ρ and a solution i of $\Pi|_\rho$ such that $f(g_{\rho,i}) \neq \rho$. In this case, the derivation of C_i in Π must be invalid. That is, given a solution of the rwPHP($\mathcal{P}$) instance, we can find an invalid derivation of Π by a decision tree of block-depth 1. $\square$

5.3 Refuters for Tseitin Formulas

Tseitin formulas. Let $G = (V, E)$ be a undirected connected graph, where each vertex $v \in V$ is associated with a value $\tau(v) \in \{0, 1\}$, and each edge $e \in E$ is associated with a Boolean variable x_e. The goal is to assign values to each x_e so that for each vertex $v \in V$, the XOR of edge labels incident to v is equal to $\tau(v)$; that is,

$$\bigoplus_{e \sim v} x_e = \tau(v), \tag{6}$$

where $e \sim v$ denotes that the edge e is incident to the vertex v.

We say τ is an *odd-weighted* function if $\bigoplus_{v \in V} \tau(v) = 1$. It is not hard to see that the above task is impossible if and only if τ is odd-weighted ([Urq87, Lemma 4.1]).

Definition 5.5. The *Tseitin formula* $\mathrm{Tseitin}(G, \tau)$ [Tse83] consists of Equation 6 for every vertex v. When G is a d-regular graph (i.e., every vertex v is incident to exactly d edges), we can write Equation 6 as a d-CNF with 2^{d-1} clauses:

$$\bigwedge_{y_1 \oplus y_2 \oplus \cdots \oplus y_d \neq \tau(v)} ((x_{e_1} \neq y_1) \vee (x_{e_2} \neq y_2) \vee \cdots \vee (x_{e_d} \neq y_d)), \tag{6'}$$

where $e_1, e_2, \ldots, e_d$ are edges incident to v.

For every odd-weighted function τ, $\mathrm{Tseitin}(G, \tau)$ is unsatisfiable; when G is an *expander* graph, $\mathrm{Tseitin}(G, \tau)$ becomes hard for resolution.

Definition 5.6. Let $G = (V, E)$ be an undirected graph. For $S, T \subseteq V$, denote $E(S, T)$ as the set of edges in E with one endpoint in S and the other endpoint in T. The *expansion* of G is defined as:

$$e(G) := \min\{|E(S, V \setminus S)| : S \subseteq V, |V|/3 \leq |S| \leq 2|V|/3\}.$$

This gives rise to a family of popular hard tautologies in proof complexity. The first exponential resolution lower bound for Tseitin formulas was proved by Urquhart [Urq87]; the proof was subsequently simplified by [Sch97, BW01]. We restate the theorem from [BW01] below.

Theorem 5.7 ([BW01, Theorem 4.4]). *For every undirected connected graph G and odd-weighted function $\tau : V \to \{0, 1\}$, any resolution refutation of* $\mathrm{Tseitin}(G, \tau)$ *contains a clause C with $w(C) \geq e(G)$.*

In this paper, we only consider Tseitin formulas on graphs with constant degree $d = O(1)$.

Width Refuters. Similar to the Pigeonhole Principle, we first study the width refuter for Tseitin formulas.

Definition 5.8. Let $\textsc{Refuter}(w(\text{Tseitin} \vdash_{\mathsf{Res}} \bot) < e(G))$ denote the following problem. The input consists of an undirected connected graph $G = (V, E)$ on n vertices with degree $d = O(1)$, an odd-weighted assignment $\tau : V \to \{0, 1\}$, a parameter $e \le |E|$, and a purported resolution refutation Π of $\text{Tseitin}(G, \tau)$ with width less than e. A valid solution is either of the following:

- an index i such that the i-th node in Π is an invalid derivation, or
- a vertex set $S \subseteq V$ such that $|V|/3 \le |S| \le 2|V|/3$ and $|E(S, V \setminus S)| < e$.

(Note: in this $\mathsf{TFNP}^{\mathsf{dt}}$ problem, we think of $\text{poly}(n)$-time algorithms as "efficient", hence an efficient procedure can read the whole graph G, verify that τ is indeed odd-weighted, or count the number of edges between S and $V \setminus S$. When we calculate block-depth, the inputs (G, τ, e) are treated as a single block.)

Remark 7. This definition is different from most refuter problems considered in this paper, as it is not for a single family of tautology, and it does not even *guarantee* that the tautology is hard! Instead, it asks to find either an invalid derivation in the purported proof or a *certificate* of the tautology being easy (i.e., a sparse cut in the graph).

We argue that this is a natural definition. Let $\text{pf}^{\alpha}_{\text{Tseitin}}(G, \tau, e)$ denote the $\Pi_1^b(\alpha)$-sentence "α encodes a width-e proof of $\text{Tseitin}(G, \tau)$" (note that α is treated as an oracle, i.e., a second-order object, while G, τ, and e are inputs, i.e., first-order objects). That is,

$$\text{pf}^{\alpha}_{\text{Tseitin}}(G, \tau, e) := \forall i \ \text{Correct}^{\alpha}(G, \tau, e, i),$$

where $\text{Correct}^{\alpha}(G, \tau, e, i)$ expresses that the i-th step of α, as a width-$(e-1)$ proof of $\text{Tseitin}(G, \tau)$, is correct. Similarly, let $\text{Expander}(G, e)$ denote the Π_1^b-sentence that $e(G) \ge e$. That is,

$$\text{Expander}(G, e) := \forall S \subseteq V \ (|S| \in [(1/3)|V|, (2/3)|V|] \implies |E[S, V \setminus S]| \ge e).$$

The proof in [BW01] actually shows that $\text{Expander}(G, e) \implies \neg\text{pf}^{\alpha}_{\text{Tseitin}}(G, \tau, e)$, which after rearranging is equivalent to:

$$\exists i \ \neg\text{Correct}^{\alpha}(G, \tau, e, i) \lor \exists S \ (|S| \in [(1/3)|V|, (2/3)|V|] \land |E[S, V \setminus S]| < e). \tag{7}$$

It is easy to see that Definition 5.8 is exactly the $\mathsf{TFNP}^{\mathsf{dt}}$ problem corresponding to Equation 7.

Theorem 5.9. *$\textsc{Refuter}(w(\text{Tseitin} \vdash_{\mathsf{Res}} \bot) < e(G))$ is PLS-complete.*

Proof. We will show that there is a (uniform) decision tree reduction of block-width 3 from this problem to $\textsc{Iter}$.

Let $G = (V, E)$ be an undirected graph with purported expansion parameter e. Let $\tau : V \to \{0, 1\}$ be an odd-weighted function. Let Π be a purported resolution refutation that consists of clauses $C_{-k}, \cdots, C_{-1}, C_0, \cdots, C_{L-1}$. where $C_{-k}, \cdots, C_{-1}$ are axioms of the unsatisfiable CNF associated with G and τ. Note that we can syntactically require that each C_i has width at most $e - 1$. Our goal is to find either an invalid derivation in Π or a witness that the expansion of G is, in fact, less than e. In particular, the witness is a vertex set $S \subseteq V$ such that $|V|/3 \le |S| \le 2|V|/3$ and $|E(S, V \setminus S)| < e$.

Similar to before, we first introduce a complexity measure for a clause C. Let $v \in V$ be a vertex. We say an assignment α is v-critical if α only falsifies the constraint associated with v and satisfies all other constraints of the given unsatisfiable CNF. The complexity measure, denoted by $\text{cri}(C)$, is defined as follows.

$$\text{cri}(C) := |\{v \in V : \exists v\text{-critical assignment } \alpha \text{ such that } C(\alpha) = 0\}|.$$

Note that cri has four important properties:

- $\text{cri}(\bot) = n$;

- $\mathrm{cri}(C_i) = 1$ for all $-k \le i \le -1$, namely, $\mathrm{cri}(C) = 1$ for all axioms C;
- cri is subadditive with respect to resolution derivation, namely, if C is resolved from A and B, then $\mathrm{cri}(C) \le \mathrm{cri}(A) + \mathrm{cri}(B)$;
- if C is obtained from a weakening of A, then $\mathrm{cri}(C) \le \mathrm{cri}(A)$.

We first show that $\mathrm{cri}(\cdot)$ can be computed in polynomial time. Then we show that any clause C_i such that $n/3 \le \mathrm{cri}(C_i) \le 2n/3$ will give us a solution. The PLS-membership follows from that the standard 1/3-2/3 trick can be implemented via a reduction to reversed Iter.

Lemma 5.10. *For any clause C, $\mathrm{cri}(C)$ can be computed in $\mathrm{poly}(n)$ time.*

Proof. Fix any clause C. We will enumerate $v \in V$ and check the existence of v-critical assignments.

Note that the aimed assignment α needs to satisfy that $C(\alpha) = 0$, so all literals in C are fixed. For α being a v-critical assignment, the constraint associated with v needs to be falsified. We enumerate an axiom in the constraint associated with v. Since d is a constant, there are only $2^{d-1} = O(1)$ axioms that we need to enumerate.

Fix such an axiom, and set all literals in this axiom to be 0 as well (if setting them to be 0 is not consistent with $C(\alpha) = 0$, then skip this axiom and try the next one). Now we have fixed some variables and left other variables free. Let $\rho \in \{0, 1, *\}^m$ be this partial assignment, where $m = |E|$. Note that $C(\rho) = 0$ and the constraint associated with v has also been falsified. So we only need to check if there is a complement α of ρ such that all other constraints can be satisfied by α. This reduces to checking whether a system of linear equations over $\mathbb{F}_2$ has a solution, which can be done in polynomial time. $\diamond$

Then we show that finding a clause C_i such that $\mathrm{cri}(C_i) \in [n/3, 2n/3]$ can be reduced to Iter.

Reduction to Iter: The instance of a reversed Iter is defined by the following function $S : [L] \to [L]$. For every $i \in [L]$:

- if $\mathrm{cri}(C_i) < \frac{2n}{3}$, then $S(i) = i$;
- otherwise, if C_i is a weakening from C_j, then let $S(i) = j$;
- Finally, let C_i be resolved from C_j and C_k: If $\mathrm{cri}(C_j) \ge \mathrm{cri}(C_k)$, then $S(i) = j$; otherwise $S(i) = k$.

It is easy to see that this reduction can be implemented in block-depth 3: for example, if C_i is resolved from C_j and C_k, then one only needs to read the i-th, j-th, and k-th node in the resolution refutation.

Note that when we find any solution i of this reversed Iter instance, it satisfies $S(i) < i$ and $S(S(i)) = i$. This means $\mathrm{cri}(C_i) \ge 2n/3$ but $\mathrm{cri}(C_{S(i)}) < 2n/3$. Thus we have $\mathrm{cri}(C_{S(i)}) \in [n/3, 2n/3]$.

Correctness of the Reduction: Fix C such that $n/3 \le \mathrm{cri}(C) \le 2n/3$. Let

$$E' = \{(u, v) \in E \mid u \in \mathrm{cri}(C), v \in V \setminus \mathrm{cri}(C)\}.$$

We show that C contains every variable that appears in E'. If not, let $e = (u, v) \in E'$ be a missing variable and suppose without loss of generality that $u \in \mathrm{cri}(C)$ and $v \notin \mathrm{cri}(C)$. Since $u \in \mathrm{cri}(C)$, by definition we know there exists a u-critical assignment α_u such that $C(\alpha_u) = 0$. Let α'_u be the same assignment but flipping $x_{(u,v)}$. Then by definition, we obtain a new assignment α'_u that is v-critical. However, recall that $v \notin \mathrm{cri}(C)$, which leads to a contradiction.

Thus, suppose that C is not obtained by an invalid derivation, then since $\mathrm{width}(C) < e$, we know that $|E'| < e$, which means that $\mathrm{cri}(C)$ is a witness that the expansion of G is in fact less than e.

This finishes the proof. $\square$

Size Refuter. After the PLS-membership of width refuter, we are ready to study the size refuter.

We consider Tseitin formulas where the underlying graph $G = (V, E)$ is an expander. Recall from Definition 5.6 that the expansion of G, denoted as $e(G)$, is the minimum number of edges between S and $V \setminus S$ over every subset $S \subseteq V$ such that $|V|/3 \leq |S| \leq 2|V|/3$. It is proved in [Sch97, BW01] that for every constant-degree expander G with $e(G) \geq n$ and every odd-weighted function $\tau : V \to \{0, 1\}$, the tautology $\mathrm{Tseitin}(G, \tau)$ requires size-$2^{\Omega(n)}$ resolution proof.

Now, analogous to Definition 5.8, we define the refuter problem for the size lower bounds, where the graph G is also given as an input, and a certificate for G not being an expander is also a valid output:

Definition 5.11. Let $\textsc{Refuter}(s(\mathrm{Tseitin} \vdash_{\mathsf{Res}} \bot) < 1.01^{n/d})$ denote the following problem. The input consists of an undirected connected d-regular graph $G = (V, E)$ on n vertices, an odd-weighted assignment $\tau : V \to \{0, 1\}$, and a purported resolution refutation Π for $\mathrm{Tseitin}(G, \tau)$ that contains at most $1.01^{n/d}$ clauses. A valid solution is either of the following:

- an index i such that the i-th node in Π is an invalid derivation, or
- a vertex set $S \subseteq V$ such that $|V|/3 \leq |S| \leq 2|V|/3$ and $|E(S, V \setminus S)| < n$.

Again, when we calculate the block-depth of reductions, we treat (G, τ) as one input block.

Theorem 5.12. *Let $G = (V, E)$ be a d-regular undirected connected graph and $\tau : V \to \{0, 1\}$ be an odd-weighted function. Then, if $e(G) \geq n$, then* $\mathrm{Tseitin}(G, \tau)$ *requires resolution size* $\geq 1.01^{n/d}$.

Moreover, there is a uniform decision tree reduction from $\textsc{Refuter}(s(\mathrm{Tseitin} \vdash_{\mathsf{Res}} \bot) < 1.01^{n/d})$ *to* rwPHP(PLS) *with block-depth* 3.

Proof. We follow the proof in [Sch97], which (also) uses a random restriction argument and a width lower bound. Our exposition about the random restrictions will be careful and slow (since this is relevant to our reduction to rwPHP(PLS)), but we will be sketchy about other parts.

Consider a random restriction as follows. Let $t := n/10$, pick t edges $E' = \{e_1, e_2, \ldots, e_t\}$ uniformly at random, and for each edge e_i assign a uniformly random bit to x_{e_i}. For an edge $e = (x, y)$, each time we assign $x_e \leftarrow 0$, we do nothing with the function τ; each time we assign $x_e \leftarrow 1$, we flip both $\tau(x)$ and $\tau(y)$. After picking these t edges, we reduced the formula $\mathrm{Tseitin}(G, \tau)$ to the formula $\mathrm{Tseitin}(G', \tau')$, where G' is the graph G with edges in E' removed, and τ' is the assignment on vertices we obtained at the end. It is easy to see that $e(G') \geq e(G) - t$, hence by Theorem 5.7, any resolution refutation for $\mathrm{Tseitin}(G', \tau')$ requires width $\geq e(G) - t$.

It would be helpful to rigorously define the space of random restrictions. Fix an ordering $\prec$ (e.g., the lexicographic one) over the $nd = 2|E|$ literals. A restriction is described by a sequence $(i_0, i_1, \ldots, i_{t-1})$ as follows. We first pick the i_0-th literal ℓ_0 according to $\prec$ and set $\ell_0 := 1$. Now we are left with $nd - 2$ literals (as both ℓ_0 and $\bar{\ell}_0$ are set) and we pick the i_1-th literal ℓ_1 among them, according to $\prec$. After setting $\ell_1 := 1$, we are left with $nd - 4$ literals and we pick the i_2-th one, and so on. Each sequence corresponds to a restriction that sets the values of t edges (but note that each restriction corresponds to $t!$ such sequences). The space of random restrictions is denoted as

$$\mathcal{R} := [nd] \times [nd - 2] \times [nd - 4] \times \cdots \times [nd - 2t + 2].$$

Let $w := e(G) - t$ and fix a clause C of width $\geq w$. If we know that a restriction ρ does not kill C, then there is a more efficient way to describe ρ by a sequence $(j_0, j_1, \ldots, j_{t-1})$, as follows. We first pick the j_0-th literal ℓ_0 *among those $nd - w$ ones not in C*, according to $\prec$, and set $\ell_0 := 1$. After this round, there are at most $nd - w - 1$ remaining literals not in C: If $\bar{\ell}_0 \in C$ then there are exactly $nd - w - 1$ such literals (i.e., excluding ℓ_0), otherwise there are $nd - w - 2$ remaining literals (i.e., excluding ℓ_0 and $\bar{\ell}_0$). Anyway, we use $nd - w - 1$ as an upper bound on the number of literals that we can choose after the first round. In the next round, we choose the j_1-th literal ℓ_1 not in C according to $\prec$, set $\ell_1 := 1$, and now there remains at most $nd - w - 2$ literals. In the next round, we choose the j_2-th such literal, and so on. The space of "bad" restrictions that do not kill C is

$$\mathcal{BAD} := [nd - w] \times [nd - w - 1] \times \cdots \times [nd - w - t + 1].$$

Given any C and $b \in \mathcal{BAD}$, we can compute $\mathsf{seq}(C, b) \in \mathcal{R}$ as the "b-th bad restriction corresponding to C".[27] Given any clause C of width $\geq w$ and any restriction $\rho \in \mathcal{R}$ that does not kill C, we can compute an encoding $\mathsf{bad}(C, \rho) \in \mathcal{BAD}$ such that $\mathsf{seq}(C, \mathsf{bad}(C, \rho)) = \rho$. The following calculation corresponds to a "union bound" over $L := 1.01^{n/d}$ clauses in the purported resolution proof:

$$\begin{aligned} L \cdot \frac{|\mathcal{BAD}|}{|\mathcal{R}|} &= L \cdot \prod_{i \in [t]} \frac{nd - e(G) + t - i}{nd - 2i} \\ &\leq L \cdot \left(1 - \frac{e(G) - 3t}{nd - 2t}\right)^t \\ &\leq 1.01^{n/d} \cdot \left(1 - \frac{7}{10d - 2}\right)^{n/10} \leq 1/2. \end{aligned}$$

The lower bound argument proceeds as follows. Let $\Pi = (C_0, C_1, \ldots, C_{L-1})$ be a purported size-L resolution proof for $\mathrm{Tseitin}(G, \tau)$. By the above union bound, there is a restriction $\rho \in \mathcal{R}$ that kills every clause in Π with width $\geq w$. This restriction shrinks Π into $\Pi|_\rho$ which is a width-w resolution proof for $\mathrm{Tseitin}(G', \tau')$, contradicting the width lower bound. Therefore, every resolution proof for $\mathrm{Tseitin}(G, \tau)$ requires more than $1.01^{n/d}$ many clauses.

Finally, we describe the reduction from $\textsc{Refuter}(s(\mathrm{Tseitin} \vdash_{\mathsf{Res}} \bot) < 1.01^{n/d})$ to $\mathrm{rwPHP}(\mathsf{PLS})$:

(f) The function $f : [L] \times \mathcal{BAD} \to \mathcal{R}$ is defined as $f(i, b) := \mathsf{seq}(C_i, b)$.

(I_ρ) For every $\rho \in \mathcal{SEQ}$, we obtain a purported width-w resolution proof $\Pi|_\rho$ for $\mathrm{Tseitin}(G', \tau')$. Every node in $\Pi|_\rho$ can be computed in block-depth 1 from Π. Using Theorem 5.9, we reduce the problem of finding an invalid derivation in Π_ρ to an $\textsc{Iter}$ instance I_ρ, where each node in I_ρ is computed in block-depth 3 from $\Pi|_\rho$.

(g) For every $\rho \in \mathcal{SEQ}$ and every valid solution o of I_ρ, we can compute an index $i \in [L]$ from o such that the i-th step in $\Pi|_\rho$ is an illegal derivation. We let $g_{\rho,o} := (i, \mathcal{BAD}(C_i, \rho))$.

Suppose that (ρ, o) is any solution to the $\mathrm{rwPHP}(\mathsf{PLS})$ instance defined above. Let $i \in [L]$ be computed from o as above, then we claim that the i-th step of Π must be an illegal derivation. Indeed, since o is a solution of I_ρ, the i-th step of $\Pi|_\rho$ must be illegal. On the other hand, if the i-th step of Π is not illegal, then $C_i|_\rho$ is a clause of width $\geq w$, and thus

$$f(g_{\rho,o}) = \mathsf{seq}(C_i, \mathcal{BAD}(C_i, \rho)) = \rho,$$

contradicting that (ρ, o) is a valid solution to the reduced $\mathrm{rwPHP}(\mathsf{PLS})$ instance. $\square$

5.4 Refuters for Random k-CNFs

Finally, we show that resolution lower bounds for random k-CNFs can be refuted in $\mathrm{rwPHP}(\mathsf{PLS})$. More precisely, as in [CS88], we consider the distribution $\mathcal{F}(k, n, m)$ over k-CNFs with n variables and m clauses where each clause is i.i.d. chosen from all $\binom{n}{k}2^k$ ordinary clauses of size k over the n variables. (A clause is *ordinary* if there is no variable x_i such that both x_i and $\bar{x}_i$ occur in this clause.) Let $c \geq 1, \varepsilon > 0$

[27] Note that some $b \in \mathcal{BAD}$ might not correspond to a valid restriction. We can set $\mathsf{seq}(C, b)$ to be an arbitrary value.

be constants, and $\{F_n\}_{n\in\mathbb{N}}$ be a family of k-CNFs, where each F_n is a k-CNF over n variables and cn clauses. In the search problem

$$\textsc{Refuter}(s(F_n) < (1+\varepsilon)^n),$$

we are given query access to a purported resolution refutation Π for F_n that contains at most $(1+\varepsilon)^n$ clauses, and our goal is to locate an invalid derivation in Π.

Theorem 5.13. *For every large enough positive integer k and $c \geq 0.7 \cdot 2^k$, there is a constant $\varepsilon > 0$ such that the following holds. Let $\{F_n\}_{n\in\mathbb{N}}$ be a sequence of random k-CNFs where each F_n is independently chosen according to the distribution $\mathcal{F}(k, n, cn)$. With probability 1, there is a non-uniform decision tree reduction of block-depth 2 from the problem $\textsc{Refuter}(s(F_n) < (1+\varepsilon)^n)$ to* rwPHP(PLS) *that works for all large enough n.*

The unsatisfiability of $\{F_n\}_{n\in\mathbb{N}}$ and the resolution lower bounds for $\{F_n\}_{n\in\mathbb{N}}$ are already shown in the seminal work of Chvátal and Szemerédi [CS88]. We prove Theorem 5.13 by formalizing their resolution lower bound proofs as decision tree reductions to rwPHP(PLS).

The reason that our reduction in Theorem 5.13 is non-uniform is very similar to that in Section 5.1. First, it appears infeasible to decide if $(1+\varepsilon)^n$ is indeed a valid resolution size lower bound for the input formula F_n. Second, the proofs in [CS88] involve some objects that appear to be infeasible to compute given F_n; however, these objects do not depend on the purported size-$(1+\varepsilon)^n$ resolution refutation, thus can be hardwired in a non-uniform decision tree. It might be possible to obtain a "uniform version" of Theorem 5.13 like what we did for Tseitin formulas (Definition 5.8, Remark 7, Definition 5.11), by completely formalizing [CS88] in bounded arithmetic. We choose not to do so because we believe that a non-uniform upper bound of rwPHP(PLS) already supports our claim that rwPHP(PLS) captures the complexity of proving *most* resolution lower bounds; dealing with extra details in [CS88] would only be distracting.

We assume familiarity with the (quite involved) proof in [CS88]. In particular, we need the following definitions and theorems:

- Fix a k-CNF F over n variables and cn clauses. Let $X = \{x_1, x_2, \ldots, x_n\}$ denote the set of variables of F. The "structure" of F can be described by a k-uniform (multi-)hypergraph H over the vertex set X, where each clause F_i of F corresponds to the hyperedge

$$E_i := \{x_j \in X : F_i \text{ contains } x_j \text{ or } \bar{x}_j\}.$$

- Let E' be a subset of hyperedges in H, the *boundary* of E' is the set of all vertices that belong to exactly one hyperedge in E'. We say that H has property $P(a)$ if, for every $m \leq an$, every family of m edges has boundary size at least $m/2$.

- Let $\mathcal{S}$ be a family of subsets of X (note that $\mathcal{S}$ might be a multiset). A *system of distinct representatives* (SDR) of $\mathcal{S}$ is a mapping from each $S \in \mathcal{S}$ to an element in S such that different subsets in $\mathcal{S}$ are mapped to different elements. Alternatively, consider the bipartite graph $(\mathcal{S}, X)$ such that an edge between $S \in \mathcal{S}$ and $x_i \in X$ is drawn if and only if $x_i \in S$, then an SDR of $\mathcal{S}$ is an $\mathcal{S}$-perfect matching of this bipartite graph (that is, every vertex in $\mathcal{S}$ is matched).

- Let S be a subset of vertices in H of size $s := \lfloor bn \rfloor$. We say that S is *good* if there is a subset D of S with $|S \setminus D| \leq (a/32)|S|$ such that every family of at most an edges has an SDR that is disjoint from D. We denote this subset as $D(S)$. We say that H has property $Q(a, b)$ if a random size-s subset $S \subseteq X$ is good with probability at least $1/2$.

- [CS88, Lemma 3] showed that any hypergraph satisfying certain "sparsity" conditions will have properties $P(a)$ and $Q(a, b)$. As a corollary ([CS88, Lemma 4]), for every large enough integers k and $c \geq 0.7 \cdot 2^k$, there are $a, b > 0$ with $b \leq a/8$ such that a random k-uniform hypergraph with

n vertices and cn hyperedges has properties $P(a)$ and $Q(a,b)$ with probability $\geq 1-n^{-2}$ for large enough n.[28]

Now we outline the strategy of [CS88]. Let $F = F_n$ be a k-CNF over n variables and cn clauses whose underlying hypergraph H satisfies $P(a)$ and $Q(a,b)$. We first choose a "special pair" (S,ρ) where S is a random subset of $s := \lfloor bn \rfloor$ vertices and $\rho \in \{0,1\}^{D(S)}$ is a uniformly random restriction on variables in $D(S)$. Then we use a *random restriction* argument to reduce the size lower bound to a *width lower bound*:

Random restriction: Let C be any clause in the purported resolution refutation for F such that the width of C is at least $an/8$. With probability $1-2^{-\Omega(n)}$ over the choice of S, we have $|\mathrm{Vars}(C)\cap S| \geq as/16$, where $\mathrm{Vars}(C)$ denotes the set of variables contained in C. Since $|S\setminus D(S)| \leq as/32$, it follows that $|\mathrm{Vars}(C) \cap D(S)| \geq as/32$, hence the probability over ρ that C is not killed by ρ is at most $2^{-as/32}$. A union bound over all $C \in \Pi$ implies that with high probability over (S,ρ), every clause of width $\geq an/8$ in Π is killed by ρ.

Width lower bound: Now we are left with a purported resolution refutation $\Pi|_\rho$ of width less than $an/8$ for the statement $F|_\rho$. For a clause $C \in \Pi$, let $\mu(C)$ denote the minimum number of clauses from F that logically implies C under ρ. (That is, for any assignment extending ρ, if all these clauses are satisfied, then C is also satisfied.) Every subset of $\leq an/2$ clauses $F' \subseteq F$ can be satisfied by some assignment on $X \setminus D(S)$ (indeed, we can simply choose an SDR for F' that is disjoint from $D(S)$, and fix this SDR), hence $\mu(\bot) > an/2$. On the other hand, every clause in F has μ value at most 1. Let $C' \in \Pi|_\rho$ be the first clause in $\Pi|_\rho$ such that $\mu(C') > an/2$ (recall that $\bot$ is the last clause in $\Pi|_\rho$). One can use a classical argument to show that $an/2 < \mu(C') \leq an$, i.e., the smallest subset of clauses $F' \subseteq F$ that logically implies C' has size between $an/2$ and an. Since H satisfies $P(a)$ and $|F'| \leq an$, the boundary of F' contains at least $|F'|/2 \geq an/4$ variables. It can be shown that C' contains every variable in the boundary of F' but not in S, hence $w(C') \geq an/8$, a contradiction.

Now we are ready to prove Theorem 5.13.

Proof of Theorem 5.13. Let $F = F_n$ be the random k-CNF and $H = H_n$ be the underlying hypergraph for F. Let $a,b > 0$ be constants that arise from [CS88, Lemma 4], we assume that H has properties $P(a)$ and $Q(a,b)$ (this assumption will be justified at the end of the proof). Our reduction needs the following non-uniform advice $\{S_i\}, \{D_i\}, \{\mathcal{R}_{i,\rho}\}$ (of course, they only depend on F and is independent of the purported resolution refutation):

- A list of subsets $S \subseteq X$ with size $s := \lfloor bn \rfloor$ that are good. Since H has property $Q(a,b)$, there are at least $N_{\mathsf{good}} := \binom{n}{s}/2$ such subsets and we only need to encode the first N_{good} ones. For each $i \in [N_{\mathsf{good}}]$, denote the i-th good subset as S_i, we also need the subset $D_i \subseteq S_i$ of size $\geq (1-a/32)s$ such that every family of at most an edges has an SDR disjoint from D_i.
- For each index i and each restriction $\rho \in \{0,1\}^{D_i}$, we compute the subformula $F|_\rho$. The above width lower bound argument (along with properties $P(a)$ and $Q(a,b)$) implies that $F|_\rho$ requires resolution width $> an/8$. Invoking Theorem 5.1, we obtain a non-uniform decision tree reduction from $\textsc{Refuter}(w(F|_\rho) \leq an/8)$ to PLS with block-depth 2, which we denote as $\mathcal{R}_{i,\rho}$.

Let Π be a purported resolution refutation for F consisting of at most $L := (1+\varepsilon)^n$ clauses. Now we describe our reduction from $\textsc{Refuter}(s(F_n) \leq (1+\varepsilon)^n)$ to $\mathrm{rwPHP}(\mathsf{PLS})$:

[28] If this probability is at least $1-n^{-2}$, then we can argue that with probability 1 over an infinite family of random k-CNFs, our reduction to $\mathrm{rwPHP}(\mathsf{PLS})$ is correct on all but finitely many input lengths; see the end of the proof of Theorem 5.13. Although [CS88] only claimed a probability of $1-o(1)$, their proof actually shows a probability of $1-n^{-\Omega(k)}$ where the big Ω hides some absolute constant. This is at least $1-n^{-2}$ when k is large enough; we suspect that our results can be extended to all $k \geq 3$ via a more careful argument.

(f) The function f takes as inputs $i \in [L]$, $\mathit{type} \in \{0,1\}$, and $w \in [\binom{n}{s} \cdot 2^{(1-a/32)s} \cdot 2^{-c'n}]$, where $c' > 0$ is a small enough constant depending on a and b. Essentially, it treats (i, type, w) as the compression of a bad "special pair" (S, ρ) (where S is a good size-s subset and ρ is an assignment over $D(S)$) and decompresses it. We start by checking that $w(C_i) \geq an/8$; if this is not the case then f outputs $\perp$. Next:

- If $\mathit{type} = 0$, then this means $|\mathrm{Vars}(C_i) \cap S| < as/16$. Recall that if $|S| = s$ is chosen uniformly at random, then the probability that $|\mathrm{Vars}(C_i) \cap S| < as/16 \leq 0.5 \cdot |C_i|s/n$ should be at most $2^{-c'n}$ for some small enough constant $c' > 0$. Hence, (S, ρ) can be compressed into $(\log \binom{n}{s} - c'n) + |D(S)|$ bits. We treat w as this compression and recover (S, ρ) from w.
- If $\mathit{type} = 1$, then $|\mathrm{Vars}(C_i) \cap S| \geq as/16$ but C_i is not killed under ρ. In this case, the values of ρ over $\mathrm{Vars}(C_i) \cap D(S)$ can be inferred from C_i. Since $|\mathrm{Vars}(C_i) \cap D(S)| \geq as/16 - as/32 = as/32$, this provides us a way to compress (S, ρ) into $\log \binom{n}{s} + (|D(S)| - as/32) \leq \log \binom{n}{s} + |D(S)| - c'n$ bits. Again, we treat w as this compression and recover (S, ρ) from w.

Now that we obtained (S, ρ), we can find an index $j \in [N_{\mathsf{good}}]$ such that $S = S_j$ (using non-uniformity). If such j does not exist, then f outputs $\perp$; otherwise f outputs (j, ρ).

Hence we have $f : [L] \times \{0,1\} \times [\binom{n}{s} \cdot 2^{(1-a/32)s} \cdot 2^{-c'n}] \to [N_{\mathsf{good}}] \times \{0,1\}^{(1-a/32)s}$. (If f outputs $\perp$ then we can assume that it outputs a default value, say $(0, 0^{(1-a/32)s})$, instead.) Recall that $L = (1+\varepsilon)^n$ and $N_{\mathsf{good}} = \binom{n}{s}/2$, which means if $\varepsilon > 0$ is small enough then

$$\frac{2L \cdot \binom{n}{s} \cdot 2^{(1-a/32)s} \cdot 2^{-c'n}}{N_{\mathsf{good}} \cdot 2^{(1-a/32)s}} \leq 2^{-\Omega(n)} \ll 1, \tag{8}$$

hence f is indeed shrinking. Given an input (i, type, w), its f value can be computed by a non-uniform decision tree of block-depth 1.

($I_{j,\rho}$) Given $j \in [N_{\mathsf{good}}]$ and $\rho \in \{0,1\}^{(1-a/32)s}$, we compute a PLS instance $I_{j,\rho}$ as follows. Abusing notation, we also use ρ to denote the restriction that equals to ρ on D_j and does not restrict any variable outside D_j. Let $\Pi|_\rho$ denote the restriction of Π over ρ, then each clause of $\Pi|_\rho$ can be computed in block-depth 1 from Π. Then we apply the reduction $\mathcal{R}_{i,\rho}$ on $\Pi|_\rho$ to obtain the PLS instance $I_{j,\rho}$.

(g) Let $j \in [N_{\mathsf{good}}]$ and $\rho \in \{0,1\}^{(1-a/32)s}$. Given a valid solution o of $I_{j,\rho}$, we can compute an index $i \in [L]$ from o such that the i-th step in $\Pi|_\rho$ is an illegal derivation. As in the definition of f, we can (assume $w(C_i) \geq an/8$ and) compress (j, ρ) as (i, type, w); then we set $g_{(j,\rho),o} = (i, \mathit{type}, w)$. If $f(i, \mathit{type}, w) \neq (j, \rho)$, then it must be the case that the i-th step in Π is already incorrect (instead of the case that $w(C_i)$ is too large).

The above reduction is correct as long as H has properties $P(a)$ and $Q(a,b)$, and its block-depth is 2. It remains to show that our reduction is correct with probability 1. In fact, for each $N \geq 1$, the probability that for every $n \geq N$, H_n has properties $P(a)$ and $Q(a,b)$ is at least

$$\prod_{n \geq N} (1 - n^{-2}) = \frac{N-1}{N}.$$

It follows that with probability 1 over the family $\{F_n\}_{n \in \mathbb{N}}$, all but finitely many H_n has properties $P(a)$ and $Q(a,b)$. In this case, our reduction will be correct on all but finitely many input lengths. $\square$

5.5 Open Problems: What We *Failed* to Formalize

One interesting problem left open by this work is whether the general size-width trade-offs in [BW01] can be proved in rwPHP(PLS). Ben-Sasson and Wigderson showed that for any unsatisfiable k-CNF F, if F requires resolution width w to refute, then F also requires resolution size $2^{\Omega(w-k)^2/n}$ to refute. This naturally leads to the following conjecture:

Conjecture 5.14 (Informal). *Let F be an unsatisfiable k-CNF with resolution width $> w_F$ and let $s_F := 2^{\Omega(w_F-k)^2/n}$. Let $\mathcal{P}$ denote the problem* Refuter$(w(F \vdash_{\mathsf{Res}} \bot) \leq w_F)$*, then there is an efficient decision-tree reduction from* Refuter$(s(F \vdash_{\mathsf{Res}} \bot) \leq s_F)$ *to* rwPHP$(\mathcal{P})$*. In particular, in the black-box setting, there is always an efficient decision tree reduction from* Refuter$(s(F \vdash_{\mathsf{Res}} \bot) \leq s_F)$ *to* rwPHP(PLS)*.*

Roughly speaking, one obstacle against proving Conjecture 5.14 is that the averaging argument used in the proof of [BW01, Theorem 3.5] seems to rely on "APC_2-style" [Jeř09] approximate counting: one needs to estimate the number of "fat" clauses up to an $(1+\varepsilon)$-multiplicative factor. Therefore, we have been unable to formalize the proof of [BW01, Theorem 3.5] in $\mathsf{T}^1_2+\mathrm{dwPHP}(\mathsf{PV})$ where only "APC_1-style" [Jeř07a] approximate counting is available.

We also leave open the complexity of proving resolution lower bounds by combining monotone circuit lower bounds [Raz85, AB87, Hak95] with feasible interpolation [Raz95b, Kra97, Pud97]. To formalize Razborov's approximation method [Raz85], it seems that we need to iteratively define exponentially many set families (one for each node in the resolution proof) and apply the sunflower lemma [ER60, ALWZ21] to each of them. It is unclear to us how to formalize such arguments in $\mathsf{T}^1_2+\mathrm{dwPHP}(\mathsf{PV})$. (See also [GGKS20] who used lifting techniques to prove monotone circuit lower bounds and resolution lower bounds.)

We showed in Corollary 5.2 that the refuter problem for every true resolution width lower bound is PLS-complete under non-uniform reductions. It would be very interesting to see whether the size lower bound analog holds or not. We propose the following conjecture (which is stronger than the non-uniform version of Conjecture 5.14):

Conjecture 5.15 (Informal). *Let F be an unsatisfiable CNF that requires resolution size $\geq s_F$ to refute. Then the problem* Refuter$(s(F \vdash_{\mathsf{Res}} \bot) < s_F)$ *is* rwPHP(PLS)*-complete under non-uniform decision tree reductions.*

(Note that the *average-case* version of Conjecture 5.15, where F is a random k-CNF and $s_F = 2^{\Omega(n)}$, is already proved in Section 5.3, by formalizing the resolution size lower bounds of [CS88].)

We end this subsection by mentioning a subtle technical issue in our proofs. There are two natural properties in the completeness of resolution (i.e., resolution can prove every true statement within size 2^n): the proof does not require weakening, and it avoids producing duplicate clauses. However, in our current PLS-hardness of refuting resolution width lower bounds and rwPHP(PLS)-hardness of refuting resolution size lower bounds, the resolution proofs produced in our reduction rely on both weakening rules and duplicated clauses. This raises an open question: What is the complexity of the corresponding refuter problems if the proofs are restricted from using either weakening rules or duplicate clauses?

6 Applications

6.1 Proof Complexity of Proof Complexity Lower Bounds

In this subsection, we translate our TFNP upper bounds for the refuter problems into *proof complexity upper bounds* for *proof complexity lower bounds*, showing that resolution lower bounds can actually be proved in weak proof systems! In particular, we show that low-width resolution (itself) can prove lower bounds on resolution width (Theorem 6.1), while low-width *random resolution* (as defined in [BKT14, PT19]) can prove resolution size lower bounds (Theorem 6.3).[29] This stands in stark contrast to the results proven in [AM20, Gar19, dRGN+21] that resolution cannot prove size lower bounds against itself.

Formalization of proof complexity lower bounds as CNFs. Suppose a family of formulas $\mathcal{F} = \{F_n\}$ does not have a width w_F resolution refutation (i.e., $w(\mathcal{F} \vdash_{\mathsf{Res}} \bot) > w_F$). Then we can transform the refuter problem $\textsc{Refuter}(w(\mathcal{F} \vdash_{\mathsf{Res}} \bot) \le w_F)$ into a family of unsatisfiable CNFs $\mathcal{F}^w_{\mathsf{wLB}}$ using via false clause search problem (Equation 2). That is, an unsatisfiable CNF F^w_{wLB} in the family $\mathcal{F}^w_{\mathsf{wLB}}$ is defined as follows:

- The input of F^w_{wLB} is a purported length-L resolution refutation for F_n represented as a list of nodes $C_0, C_1, \ldots, C_{L-1}$ and each node C_i can be encoded in $O(w_F \log n)$ bits.
- Each potential solution *sol* of the refuter problem can be verified by a decision tree of block-depth 3, hence they can be turned into a CNF C_{sol} of width $O(w_F \log n)$. F^w_{wLB} is simply the conjunction of these CNFs.

We can similarly transform a resolution *size* lower bound $s(\mathcal{F} \vdash_{\mathsf{Res}} \bot) > L$ into a family of unsatisfiable CNFs $\mathcal{F}^L_{\mathsf{sLB}}$ via the refuter problem $\textsc{Refuter}(s(\mathcal{F} \vdash_{\mathsf{Res}} \bot) \le L)$. The only difference is that each node consists of an (unbounded-width) clause and thus is encoded in $O(n + \log L)$ bits.

It is easily seen that $\mathcal{F}^w_{\mathsf{wLB}}$ are CNFs of width $O(w_F \log n)$ and $\mathcal{F}^L_{\mathsf{sLB}}$ are CNFs of width $O(n + \log L)$. (When $L = 2^{n^{\Omega(1)}}$, these width parameters are $\mathrm{polylog}(L)$ and can be thought of as "efficient".)

Remark 8 (Comparison with previous formalizations). Similar formalizations of resolution lower bound statements have also appeared in [AM20, Gar19, dRGN+21]. The biggest difference between these formalizations is that in [dRGN+21], the predecessors of each node are represented in binary and as $O(\log N)$ bits; while in [AM20, Gar19], the predecessors are represented in unary and we have tables $L[i,j]$ and $R[i,j]$ denoting whether node j is a predecessor of node i. Note that in the unary representation, it requires an axiom of width L to express that every node u has at least one predecessor $L[u]$ and at least one predecessor $R[u]$. Thus it is impossible to prove resolution width lower bounds in resolution width $O(w \log N) \ll L$. Therefore, we choose to use the binary formalization as in [dRGN+21].

The formalization in [dRGN+21] allows *disabled* nodes in the resolution proof. Our proof complexity upper bounds hold regardless of whether such nodes are allowed in the formalization.

Low-width resolution can prove resolution width lower bounds. First, we show that:

Theorem 6.1. *For every family of unsatisfiable CNFs $\mathcal{F}$, if $w(\mathcal{F} \vdash_{\mathsf{Res}} \bot) > w_F$, then $w(\mathcal{F}^w_{\mathsf{wLB}} \vdash_{\mathsf{Res}} \bot) \le O(w_F \log N)$.*

Theorem 6.1 follows from the proof of Theorem 5.1 and Theorem 2.4: since the refuter problem corresponding to resolution width lower bounds can be solved in PLS and the totality of PLS can be proved in low resolution width, it follows that resolution width lower bounds themselves can be proved in low resolution width. For the sake of intuition, we also present an equivalent but more direct proof using *Prover-Delayer games* [Pud00]. The necessary backgrounds on Prover-Delayer games are presented in Section C.1.

Proof. It suffices to construct a Prover strategy with memory size $O(w_F \log N)$ in the Prover-Delayer game for $\mathcal{F}^w_{\mathsf{wLB}}$. The Prover starts by querying the last node in the purported resolution proof, which should contain the empty clause $\bot$. The Prover maintains the invariant that she is always at some (not disabled) clause C_i such that $w(\mathcal{F} \vdash C_i) > w_F$, i.e., it requires resolution width $> w_F$ to derive C_i from the axioms. Each time the Prover is at some clause C_i:

[29] More precisely, we use Theorem 5.1 to show that low-width resolution can prove *every* resolution width lower bound that is true, and use Theorem 5.13 to show that low-width random resolution can prove *most* resolution size lower bounds.

- Suppose C_i is *resolved* from the clauses C_j, C_k. Then the Prover queries C_j and C_k; if $j \geq i$, $k \geq i$, or the derivation from (C_j, C_k) to C_i is invalid, then she wins the game. Otherwise, since the widths of C_j and C_k are at most w_F (recall that this is guaranteed syntactically by only allocating w_F variables to each clause), one of C_j, C_k must require $> w_F$ width to derive. Suppose it is C_j; that is, $w(\mathcal{F} \vdash C_j) > w_F$. Then the Prover forgets C_i and C_k and only remembers C_j.
- Suppose C_i is a *weakening* of a clause C_j. The Prover queries C_j; if $j \geq i$ or the weakening from C_j to C_i is invalid, then she wins the game. Otherwise, it must be the case that $w(\mathcal{F} \vdash C_j) > w_F$. Then the Prover forgets C_i and only remembers C_j.

Since the index i is always decreasing, the Prover is guaranteed to win the game. The Prover only needs to memorize $O(1)$ resolution nodes, i.e., $O(w_F \log N)$ bits. $\square$

Low-width random resolution can prove resolution size lower bounds. We first define the random resolution system (denoted as rRes):

Definition 6.2 ([BKT14, PT19])**.** An ε*-random* resolution refutation of an unsatisfiable formula F is a distribution $\mathcal{D}$ supported on pairs (Π, B), such that

1. each B is a CNF formula over the variables of F,
2. Π is a resolution refutation of $F \wedge B$, and
3. for any assignment $x \in \{0,1\}^n$, $\Pr_{(\Pi,B)\sim\mathcal{D}}[B(x) = 1] \geq 1 - \varepsilon$.

The *size* $s(F \vdash_{\mathsf{rRes}} \bot)$, and *width* $w(F \vdash_{\mathsf{rRes}} \bot)$ of a random resolution refutation $\mathcal{D}$ for F are the maximum size and width of a proof Π in the support of $\mathcal{D}$, respectively.

We remark that random resolution is not a standard (i.e., Cook–Reckhow) proof system since the distribution $\mathcal{D}$ might potentially require exponentially many bits to describe and it is also unclear how to verify Item 3 above. (In fact, random resolution cannot be simulated by a Cook–Reckhow proof system unless $\mathsf{P} = \mathsf{NP}$ [PT19, Proposition 3.3].) On the other hand, strong lower bounds on both width and size are known for random resolution [PT19], suggesting that it may be classified as a "weak" proof system.

Theorem 6.3. *For every $k \geq 3$ and $c \geq 0.7 \cdot 2^k$, there exists some $\varepsilon > 0$ such that the following holds. Let F be a random k-CNF formula chosen from the distribution $\mathcal{F}(k, n, cn)$, $L := (1+\varepsilon)^n$, and $\mathcal{F}^L_{\mathsf{sLB}}(F)$ be the CNF formula encoding the lower bound that F requires size-L resolution refutation. With probability tending to* 1 *(when $n \to \infty$) over F, $\mathcal{F}^L_{\mathsf{sLB}}(F)$ admits a* $\mathrm{poly}(n)$*-width* γ-random *resolution refutation with* $\gamma := 2^{-\Omega(n)}$.

Similarly, Theorem 6.3 is a corollary of Theorem 5.13: if a search problem reduces to rwPHP(PLS), then it also *randomly* reduces to PLS, and such a random reduction can be translated into a random resolution refutation. Nevertheless, for the sake of intuition, we present an (equivalent) proof that directly constructs the random resolution refutation (Π, B).

Proof Sketch. We assume familiarity with the proofs in Section 5.4. We use the parameters a, b from [CS88, Lemma 4], and denote $s := \lfloor bn \rfloor$. We assume that the properties $P(a)$ and $Q(a, b)$ holds for F; by [CS88, Lemma 4], this is true with high probability over $F \leftarrow \mathcal{F}(k, n, cn)$.

Recall that the variables in $\mathcal{F}^L_{\mathsf{sLB}}(F)$ encode a length-L resolution refutation $C_0, \ldots, C_{L-1}$ of F, where $L := (1+\varepsilon)^n$. Let $S_0, S_1, \ldots, S_{N_{\mathsf{good}}-1}$ denote the first $N_{\mathsf{good}} := \binom{n}{s}/2$ good size-s subsets. For each $j \in [N_{\mathsf{good}}]$, also let D_j denote any subset of S_j of size $\geq (1 - a/32)s$ such that every family of at most an edges has an SDR disjoint from D_j. To sample a pair (Π, B):

1. We first pick a random $j \in [N_{\mathsf{good}}]$ and then pick a string $\rho \leftarrow \{0,1\}^{D_j}$. We also treat ρ as a restriction that fixes every variable in D_j and leaves everything else unchanged.

2. For each $i \in [L]$, let B_i be the decision tree verifying that either $w(C_i) < an/8$ or ρ kills C_i. Note that B_i only depends on the clause C_i, which can be encoded in $\mathrm{poly}(n)$ bits. Let $B \coloneqq \bigwedge_{i\in[L]} B_i$, then B is a $\mathrm{poly}(n)$-width CNF.

   Moreover, following the same calculation as Equation 8, we can show that for any assignment x to the variables in $\mathcal{F}^L_{\mathsf{sLB}}(F)$ (i.e., x encodes a purported resolution refutation of F), the probability over B (i.e., over j and ρ) that $B(x) = 1$ is at least $1 - 2^{-\Omega(n)}$.

3. It remains to argue that there always exists a $\mathrm{poly}(n)$-width resolution refutation Π for $\mathcal{F}^L_{\mathsf{sLB}}(F) \wedge B$. We can use a similar Prover's strategy as described in Theorem 6.1. Recall that for any clause C, $\mu(C)$ denotes the minimum number of clauses from F that logically implies C under ρ, and that $\mu(\perp) > an/2$. The Prover starts from $C_{L-1} = \perp$ and maintains the invariant that she is always at some clause C_i where $\mu(C_i) > an/2$. In addition, when the Prover is at some clause C_i, she also ensures that B_i is satisfied. At some stage, she will encounter some C_i that is resolved from C_j, C_k, such that $\mu(C_i) > an/2$ and $\mu(C_j), \mu(C_k) < an/2$. But due to the width lower bound in Section 5.4, this will imply that either the i-th derivation is invalid, or that B_i is violated.

   It is easy to check that this Prover strategy only requires $\mathrm{poly}(n)$ memory. $\square$

6.2 Complexity of Black-Box TFNP Separations

In this subsection, we introduce a new type of refuter problems — $\mathsf{TFNP}^{\mathsf{dt}}$ *refuter* — which corresponds to the "complexity" of proving black-box $\mathsf{TFNP}^{\mathsf{dt}}$ separations. We present the definition and several basic properties of them in Section 6.2.1. In Section 6.2.2, we relate the $\mathsf{TFNP}^{\mathsf{dt}}$ refuter to the resolution width refuter (Lemma 6.10). Combining this with our results on resolution width refuter for EPHP and Tseitin, we characterize the "complexity" of separating PPA and PPP from PLS in the black-box setting by the class PLS itself.

Notations. For two $\mathsf{TFNP}^{\mathsf{dt}}$ problems $\mathcal{P}, \mathcal{Q}$, we write $\mathcal{P} \le_m \mathcal{Q}$ if there is a *many-one* reduction from $\mathcal{P}$ to $\mathcal{Q}$; if the reduction is also uniform, we write $\mathcal{P} \le^U_m \mathcal{Q}$.

6.2.1 Black-Box TFNP Refuters and its Properties

We start by providing a formal definition of the $\mathsf{TFNP}^{\mathsf{dt}}$ refuter problems. Roughly speaking, in the problem $\textsc{Refuter}_{d,M}(\mathcal{P} \to \mathcal{Q})$, we are given a shallow decision tree that claims to reduce $\mathcal{P}$ to $\mathcal{Q}$, and our goal is to find a witness that this shallow decision tree is incorrect.

Problem $\textsc{Refuter}_{d,M}(\mathcal{P} \to \mathcal{Q})$

Parameters: Two $\mathsf{TFNP}^{\mathsf{dt}}$ problems $\mathcal{P} = \{P_N\}, \mathcal{Q} = \{Q_N\}$ and two functions $d \coloneqq d(N), M \coloneqq M(N)$ such that there is no depth-$d(N)$ decision tree reduction from P_N to $Q_{M(N)}$ for any N.

Input: A purported depth-d decision tree reduction $(f_i, g_o)_{i\in M, o\in O_Q}$ from $P_N = \{0,1\}^N \times O_P$ to $Q_M = \{0,1\}^M \times O_Q$.

Output: A pair (ρ, o^*), where

- $\rho \in \{0,1,*\}^N$ is a partial assignment encoded by specifying the locations and the values of all non-$*$ bits;
- $o^* \in O_Q$ is a solution of the problem Q_M.

The pair (ρ, o^*) satisfy that for any input $x \in \{0,1\}^N$ consistent with ρ,

1. $(f(x), o^*) \in \mathcal{Q}$ and $(x, g_{o^*}(x)) \notin \mathcal{P}$;

2. only bits specified in ρ are ever queried when calculating $g_{o^*}(x)$ and verifying $(f(x), o^*) \in \mathcal{Q}$ and $(x, g_{o^*}(x)) \notin \mathcal{P}$.

In this section, we only consider refuting *low-depth* many-one decision tree reduction. Thus, we always assume the functions $d(N)$ and $\log M(N)$ are poly-logarithmic in N when we write "for any d, M". We also assume $M(N) \geq N$, so we will not consider reductions that are too weak. Note that it is necessary to have o^* as part of the solution for this problem to be in $\mathsf{TFNP}^{\mathsf{dt}}$; otherwise, it might take too many queries to the input (reduction) to find o^*, which is used to refute the reduction later.

We call a $\mathsf{TFNP}^{\mathsf{dt}}$ problem $\mathcal{R}$ *syntactical* if all the decision trees (T_o) for verifying the solution can be replaced by a single polynomial-time oracle Turing machine.[30] Since we mostly care about the black-box separations between *syntactical* TFNP subclasses, we assume all the $\mathsf{TFNP}^{\mathsf{dt}}$ problems in this section are syntactical.

We now present several basic properties regarding the $\mathsf{TFNP}^{\mathsf{dt}}$ separation refuter. First, a weaker reduction, which has a lower depth or smaller instance size, is easier to refute. The proof trivially follows from the definition (where item 2 needs the problem $\mathcal{Q}$ to be paddable).

Lemma 6.4. *1. If $d_1 \leq d_2$, then* $\text{Refuter}_{d_1,M}(\mathcal{P} \to \mathcal{Q}) \leq_m^U \text{Refuter}_{d_2,M}(\mathcal{P} \to \mathcal{Q})$.

2. If $M_1 \leq M_2$, then $\text{Refuter}_{d,M_1}(\mathcal{P} \to \mathcal{Q}) \leq_m^U \text{Refuter}_{d,M_2}(\mathcal{P} \to \mathcal{Q})$.

Even in the easiest parameter settings, i.e., $d = 0, M(N) = N$, the refuter problem $\text{Refuter}_{0,N}(\mathcal{P} \to \mathcal{Q})$ is least as hard as $\mathcal{Q}$ itself, because a valid solution of $\mathcal{Q}$ is always required to witness a mistake given by the input reduction.

Lemma 6.5. $\mathcal{Q} \leq_m^U$ *$\text{Refuter}_{0,N}(\mathcal{P} \to \mathcal{Q})$.*

Proof. Let $y \in \{0,1\}^N$ be an instance of problem $Q_N \in \{0,1\}^N \times O_Q$ and let o_P be an arbitrary fixed solution of problem P_N. We construct a trivial depth-0 reduction $(f_i, g_o)_{i \in N, o \in O_Q}$, where

$$f_i(x) = y_i, \forall i \in N; g_o(x) = o_P, \forall o \in O_Q.$$

Consider such reduction (f_i, g_o) as an instance of $\text{Refuter}_{0,N}(\mathcal{P} \to \mathcal{Q})$, and let (ρ, o^*) be any solution of it. By definition, o^* is a valid solution of instance y. Moreover, our reduction is uniform, though (f_i, g_o) is not. ☐

Finally, we present two useful lemmas, which state that it is easier to refute a reduction when the *difficulty gap* between these two problems becomes larger.

Lemma 6.6. *If $\mathcal{P} \leq_m^U \mathcal{S}$, and $d_1(N), \log M_1(N) = \operatorname{polylog}(N)$, then*

$$\text{Refuter}_{d_1,M_1}(\mathcal{S} \to \mathcal{Q}) \leq_m^U \text{Refuter}_{d_2,M_2}(\mathcal{P} \to \mathcal{Q}),$$

for some $d_2(N), \log M_2(N) = \operatorname{polylog}(N)$.

Proof. Given a depth-d_1 reduction $(f_i, g_o)_{i \in M_1, o \in O_Q}$ from S_N to Q_{M_1}, we compose it with any (uniform) low-depth reduction $(h_i, l_o)_{i \in N, o \in O_S}$ from $P_{N'}$ to S_N. Now we get a depth-d' reduction (f_i', g_o') from $P_{N'}$ to Q_{M_1} with

$$f_i'(x) = f_i(h(x)), \forall i \in [M_1]; \qquad g_o'(x) = l_s(x), s := g_o(h(x)), \forall o \in O_Q.$$

Let $d_2(N') := d', M_2(N') := M_1$, and it is easy to verify that $d_2(N'), \log M_2(N') = \operatorname{polylog}(N')$.

Consider a pair (ρ_P, o^*) that refutes (f_i', g_o'). Let x be any input of $P_{N'}$ that is consistent with ρ_P and define $y := h(x)$. We show how to construct a partial assignment ρ_S consistent with y such that (ρ_S, o^*) refutes (f_i, g_o). We start with setting ρ_S to all $*$ strings, and then execute the process of

[30] A *syntactical* $\mathsf{TFNP}^{\mathsf{dt}}$ problem is essentially a type-2 TFNP (TFNP^2) problem, see [BCE+98].

S1: calculating $g_{o^*}(y)$; **S2:** verifying $(f(y), o^*) \in \mathcal{Q}$; **S3:** verifying $(y, g_{o^*}(y)) \notin \mathcal{S}$.

During the above process, if y_i is queried and y_i has not been specified by ρ_S, we will execute $h_i(x)$ to calculate y_i and then store its value in ρ_S.

For the correctness of our construction, recall that only bits specified in ρ_P are ever queried when

P1: calculating $g'_{o^*}(x)$; **P2:** verifying $(f'(x), o^*) \in \mathcal{Q}$; **P3:** verifying $(x, g'_{o^*}(x)) \notin \mathcal{P}$.

By our construction, process **S1, S2** are sub-procedures of **P1, P2**, and thus they will only query locations of x that are already specified in ρ_P. However, process **S3** might query some locations that are not specified in ρ_P. In this case, it is safe to return arbitrary values for those queries. This is because the correctness of reduction (h_i, l_o) guarantees that there must be $(y, g_{o^*}(y)) \notin \mathcal{S}$.

Finally, note that our whole reduction, including the construction of (f'_i, g'_o) and the execution of process **S1, S2, S3**, can be done in a uniform manner. □

With a similar argument, we can also formalize the other direction.

Lemma 6.7. *If* $\mathcal{S} \leq_m^U \mathcal{Q}$ *and let* $d_1(N), \log M_1(N) = \mathrm{polylog}(N)$*, then*

$$\textit{Refuter}_{d_1,M_1}(\mathcal{P} \to \mathcal{S}) \leq_m^U \textit{Refuter}_{d_2,M_2}(\mathcal{P} \to \mathcal{Q}),$$

for some $d_2(N), \log M_2(N) = \mathrm{polylog}(N)$.

We often consider all low-depth reductions between two $\mathsf{TFNP}^{\mathsf{dt}}$ classes with no valid low-depth reductions possible. So, it is convenient to introduce a new kind of $\mathsf{TFNP}^{\mathsf{dt}}$ subclasses for this type of problem.

Definition 6.8. For two $\mathsf{TFNP}^{\mathsf{dt}}$ classes A, B ($\mathsf{A} \not\subseteq \mathsf{B}$) with $\mathcal{P}, \mathcal{Q}$ being any complete problems of A and B respectively, $\mathsf{Ref}(\mathsf{A} \subseteq \mathsf{B})$ is defined as the class of $\mathsf{TFNP}^{\mathsf{dt}}$ problems that are reducible to $\textsc{Refuter}_{d,M}(\mathcal{P} \to \mathcal{Q})$ for some $d(N), \log M(N) = \mathrm{polylog}(N)$.

This notation is well-defined because Lemma 6.6 and Lemma 6.7 guarantee that the choice of the complete problems does not matter. We also have the following corollary of Lemma 6.4 and Lemma 6.5.

Corollary 6.9. *For any two* $\mathsf{TFNP}^{\mathsf{dt}}$ *classes* A, B *such that* $\mathsf{A} \not\subseteq \mathsf{B}$*,* $\mathsf{B} \subseteq \mathsf{Ref}(\mathsf{A} \subseteq \mathsf{B})$.

6.2.2 Refuter for Separating from PLS

Now we study the complexity of refuting separations between PLS and other classes in $\mathsf{TFNP}^{\mathsf{dt}}$, in particular the separations

$$\mathsf{PPA}^{\mathsf{dt}} \not\subseteq \mathsf{PLS}^{\mathsf{dt}} \quad \text{and} \quad \mathsf{PPP}^{\mathsf{dt}} \not\subseteq \mathsf{PLS}^{\mathsf{dt}}.$$

Our main tool is the equivalence between resolution and PLS via the *false clause search* problem (cf. [dRGR22]): recall that $\mathrm{Search}(\mathcal{F}) \in \mathsf{PLS}$ if and only if $\mathcal{F}$ have a $\mathrm{polylog}(N)$-width resolution refutation (Theorem 2.4).

Studying this equivalence from a computational perspective, we related the $\mathsf{TFNP}^{\mathsf{dt}}$ refuter for PLS with the resolution width refuter.

Lemma 6.10. *For any family of unsatisfiable CNF* $\mathcal{F}$ *that has no* $\mathrm{polylog}(N)$*-width resolution refutation,*

$$\textit{Refuter}_{d,M}(\mathrm{Search}(\mathcal{F}) \to \textit{Iter}) \leq_m \textit{Refuter}(w(\mathcal{F} \vdash_{\mathsf{Res}} \bot) < w_0)$$

for some $w_0 = \mathrm{polylog}(N)$ *that may depend on* d, M.

Furthermore, this reduction is uniform when $\mathcal{F}$ *is a uniform family of unsatisfiable CNFs.*

The proof of Lemma 6.10 follows from the standard procedure that transforms a low-depth decision tree reduction to PLS (i.e., a PLS formulation) to a low-width resolution proof, using the *Prover-Delayer game* [Pud00]. This proof is rather straightforward, but many details have to be taken care of to make sure that the reduction is uniform. To be self-contained, we formally present the transformation from a PLS formulation to a resolution proof in Section C.1;[31] we then prove Lemma 6.10 in Section C.2.

As an application, we combine Lemma 6.10 with our results on resolution width refuter for EPHP and Tseitin formulas. Note that Search(EPHP) and Search(Tseitin) are in PPP and PPA respectively. Therefore, we can reduce the $\mathsf{TFNP}^{\mathsf{dt}}$ refuter for $\mathsf{PPP}^{\mathsf{dt}} \not\subseteq \mathsf{PLS}^{\mathsf{dt}}$ and $\mathsf{PPA}^{\mathsf{dt}} \not\subseteq \mathsf{PLS}^{\mathsf{dt}}$ to the resolution width refuters for EPHP and Tseitin respectively.

Theorem 6.11. *Let $\mathcal{P}, \mathcal{Q}$ be any complete problems for* PPP *and* PPA *respectively, then for any d, M, both $\textsc{Refuter}_{d,M}(\mathcal{P} \to \textsc{Iter})$ and $\textsc{Refuter}_{d,M}(\mathcal{Q} \to \textsc{Iter})$ are* PLS*-complete via uniform reductions.*

In particular, $\mathsf{Ref}(\mathsf{PPP} \subseteq \mathsf{PLS}) = \mathsf{Ref}(\mathsf{PPA} \subseteq \mathsf{PLS}) = \mathsf{PLS}$.

Equivalently, Theorem 6.11 says that *local search* arguments are both *necessary* and *sufficient* for separating PPP and PPA from PLS in the black-box setting.

Proof. Note that Lemma 6.5 already gives the PLS-hardness result, we will focus on showing that for any d, M, $\textsc{Refuter}_{d,M}(\mathcal{P} \to \textsc{Iter})$ and $\textsc{Refuter}_{d,M}(\mathcal{Q} \to \textsc{Iter})$ are in PLS via uniform reductions.

We start with PPP versus PLS. Note that Search(EPHP) is in PPP, thus, by Lemma 6.6, we have

$$\textsc{Refuter}_{d,M}(\mathcal{P} \to \textsc{Iter}) \leq_m^U \textsc{Refuter}_{d',M'}(\text{Search}(\text{EPHP}) \to \textsc{Iter}),$$

where $d', \log M'$ are also poly-logarithmic in N. Combining with Lemma 6.10, there is

$$\textsc{Refuter}_{d,M}(\mathcal{P} \to \textsc{Iter}) \leq_m^U \textsc{Refuter}(w(\text{EPHP} \vdash_{\mathsf{Res}} \bot) < w_0)$$

for some $w_0 = \text{polylog}(N)$ that may depend on d, M.

Recall that Theorem 3.3 shows that $\textsc{Refuter}(w(\text{EPHP} \vdash_{\mathsf{Res}} \bot) < w_0)$ is in PLS via a uniform reduction when $w_0 = n/3$. The same reduction to $\textsc{Iter}$ would still work when $w_0 = \text{polylog}(N)$, and the only issue is to make sure that the $\text{cri}(C)$ function could be calculated "efficiently" in this different parameter regime. Note that when $w_0 = \text{poly}(n)$ (and $N = 2^{\Omega(n)}$)), a $\text{poly}(n)$ time procedure (Lemma 3.4) would be considered as time efficient; however, when $w_0 = \text{polylog}(n)$ and N being quasi-polynomial in n, only a $\text{polylog}(n)$ running time is acceptable. Since $|C| \leq w_0 = \text{polylog}(n)$, it suffices to prove that the following claim.

Claim 6.12. $\text{cri}(C)$ *can be calculated in* $\text{polylog}(n)$ *time when* $|C| = \text{polylog}(n)$.

Proof. We modify the algorithm described in the proof of Lemma 3.4. First, notice that we do not have to enumerate all possible $\ell \in [n+1]$, because only $\text{polylog}(n)$ pigeons are *involved* in the clause C, where we say a pigeon ℓ is *involved* in C if a literal related to ℓ appears in C. Any pigeons that are not involved in C would be equivalent, thus, we only need to consider any one of them.

For a fixed ℓ, deciding whether an ℓ-critical assignment exists for C is reduced to the following graph problem: Given a complete bipartite graph with n pigeons on the left and n holes on the right, $\text{polylog}(n)$ sets of edges are then deleted, determine whether a perfect matching still exists in the end. Each deleted set can be described by a triple (i, j_1, j_2), representing the set $\{(i, j) : j_1 \leq j \leq j_2\}$.

It is not difficult to design an $\text{polylog}(n)$ time algorithm for this problem by exploiting the sparsity:

1. We first ignore all pigeons with full degree n, because they could always be matched in the end.

2. Suppose we have $t_1 = \text{polylog}(n)$ pigeons left after the first step. We then ignore all pigeons with the degree at least $t + 1$ for the same reason.

[31] This transformation is a well-known folklore among the *Proof Complexity and* TFNP community. However, to the best of the authors' knowledge, it has not yet been formally written down in any previous literature.

3. We have $t_2 = \text{polylog}(n)$ pigeons left now, and there are at most $t_1 \cdot t_2 = \text{polylog}(n)$ edges connected to those pigeons. So, we can run the standard maximum matching algorithm on the subgraph of the remaining pigeons. The original graph has a perfect matching if and only if all t_2 pigeons could be matched. $\diamond$

A similar argument works for PPA. We use the fact that Search(Tseitin(G, τ)) is in PPA when the graph G has a constant degree. We will fix a family of strongly explicit expander graph G and an odd-weighted function τ, rather than giving them as input as we did in Section 5.3. For example, we can take G as a 2D-grid with a boundary being wrapping around, and $\tau(v) = 1$ only if v is some designated vertex (say $(1, 1)$). Then, we claim that the $\text{cri}(C)$ function (defined differently for the Tseitin formula in Theorem 5.9) can also be calculated in $\text{polylog}(n)$ time when $|C| = \text{polylog}(n)$ by exploiting the sparsity of C. We omit the proof this claim here. Finally, using the same proof of Theorem 5.9, we show that $\textsc{Refuter}(w(\text{Tseitin} \vdash_{\mathsf{Res}} \bot) < w_0$ is in PLS via a uniform reduction when $w_0 = \text{polylog}(N)$, which concludes the proof. $\square$

Acknowledgments

Jiawei thanks Igor C. Oliveira for introducing him to refuters and their connections with TFNP and thanks Robert Robere and Noah Fleming for knowledge of proof complexity.

Yuhao thanks Toniann Pitassi for the knowledge and guidance in the field of proof complexity and thanks Robert Robere for introducing him to beneficial intuition about proof systems and $\mathsf{TFNP}^{\mathsf{dt}}$ classes.

Hanlin thanks Svyatoslav Gryaznov and Iddo Tzameret for helpful discussions regarding [dRGN+21] and proof complexity in general, and Rahul Santhanam and Ján Pich for beneficial conversations.

We thank Lijie Chen, Jiatu Li, and Igor C. Oliveira for sending us a preliminary version of [CLO24a]. We thank Michal Garlík for helpful discussions on [Gar19]. We thank Ján Pich and anonymous referees for their helpful suggestions that improve the presentation of this paper.

References

[AB87] Noga Alon and Ravi B. Boppana. The monotone circuit complexity of Boolean functions. *Comb.*, 7(1):1–22, 1987. doi:10.1007/BF02579196. 50

[ABM23] Albert Atserias, Sam Buss, and Moritz Müller. On the consistency of circuit lower bounds for non-deterministic time. In *STOC*, pages 1257–1270. ACM, 2023. doi:10.1145/3564246.3585253. 16

[AH19] Albert Atserias and Tuomas Hakoniemi. Size-degree trade-offs for Sums-Of-Squares and Positivstellensatz proofs. In *CCC*, volume 137 of *LIPIcs*, pages 24:1–24:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICS.CCC.2019.24. 6

[Ajt94] Miklós Ajtai. The complexity of the pigeonhole principle. *Comb.*, 14(4):417–433, 1994. doi:10.1007/BF01302964. 4

[AKPS24] Noel Arteche, Erfan Khaniki, Ján Pich, and Rahul Santhanam. From proof complexity to circuit complexity via interactive protocols. In *ICALP*, volume 297 of *LIPIcs*, pages 12:1–12:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.ICALP.2024.12. 1

[ALWZ21] Ryan Alweiss, Shachar Lovett, Kewen Wu, and Jiapeng Zhang. Improved bounds for the sunflower lemma. *Annals of Mathematics*, 194(3):795–815, 2021. doi:10.4007/annals.2021.194.3.5. 50

[AM20] Albert Atserias and Moritz Müller. Automating resolution is NP-hard. *J. ACM*, 67(5):31:1–31:17, 2020. doi:10.1145/3409472. 10, 12, 15, 16, 19, 51

[AT14] Albert Atserias and Neil Thapen. The ordering principle in a fragment of approximate counting. *ACM Trans. Comput. Log.*, 15(4):29:1–29:11, 2014. doi:10.1145/2629555. 7, 24

[Ats03] Albert Atserias. Improved bounds on the weak pigeonhole principle and infinitely many primes from weaker axioms. *Theor. Comput. Sci.*, 295:27–39, 2003. doi:10.1016/S0304-3975(02)00394-8. 6

[BB17] Arnold Beckmann and Sam Buss. The NP search problems of Frege and Extended Frege proofs. *ACM Trans. Comput. Log.*, 18(2):11:1–11:19, 2017. doi:10.1145/3060145. 17, 67

[BCE+98] Paul Beame, Stephen A. Cook, Jeff Edmonds, Russell Impagliazzo, and Toniann Pitassi. The relative complexity of NP search problems. *J. Comput. Syst. Sci.*, 57(1):3–19, 1998. doi:10.1006/JCSS.1998.1575. 4, 5, 54

[Bel20] Zoë Bell. Automating regular or ordered resolution is NP-hard. *Electron. Colloquium Comput. Complex.*, TR20-105, 2020. URL: https://eccc.weizmann.ac.il/report/2020/105. 12

[BFI23] Sam Buss, Noah Fleming, and Russell Impagliazzo. TFNP characterizations of proof systems and monotone circuits. In *ITCS*, volume 251 of *LIPIcs*, pages 30:1–30:40. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ITCS.2023.30. 12, 19, 20, 67

[BIK+92] Paul Beame, Russell Impagliazzo, Jan Krajíček, Toniann Pitassi, Pavel Pudlák, and Alan R. Woods. Exponential lower bounds for the pigeonhole principle. In *STOC*, pages 200–220. ACM, 1992. doi:10.1145/129712.129733. 4

[BIK+94] Paul Beame, Russell Impagliazzo, Jan Krajíček, Toniann Pitassi, and Pavel Pudlák. Lower bound on Hilbert's Nullstellensatz and propositional proofs. In *FOCS*, pages 794–806. IEEE Computer Society, 1994. doi:10.1109/SFCS.1994.365714. 13

[BIK+97] Samuel R. Buss, Russell Impagliazzo, Jan Krajíček, Pavel Pudlák, Alexander A. Razborov, and Jirí Sgall. Proof complexity in algebraic systems and bounded depth Frege systems with modular counting. *Comput. Complex.*, 6(3):256–298, 1997. doi:10.1007/BF01294258. 13

[BJ12] Samuel R. Buss and Alan S. Johnson. Propositional proofs and reductions between NP search problems. *Ann. Pure Appl. Log.*, 163(9):1163–1182, 2012. doi:10.1016/J.APAL.2012.01.015. 24, 64

[BK94] Samuel R. Buss and Jan Krajíček. An application of Boolean complexity to separation problems in bounded arithmetic. *Proceedings of the London Mathematical Society*, s3-69(1):1–21, 1994. doi:10.1112/plms/s3-69.1.1. 2, 7, 8, 10, 17, 24

[BKO20] Jan Bydzovsky, Jan Krajícek, and Igor C. Oliveira. Consistency of circuit lower bounds with bounded theories. *Log. Methods Comput. Sci.*, 16(2), 2020. doi:10.23638/LMCS-16(2:12)2020. 16

[BKT14] Samuel R. Buss, Leszek Aleksander Kołodziejczyk, and Neil Thapen. Fragments of approximate counting. *J. Symb. Log.*, 79(2):496–525, 2014. doi:10.1017/JSL.2013.37. 7, 11, 50, 52

[BKZ15] Samuel R. Buss, Leszek Aleksander Kołodziejczyk, and Konrad Zdanowski. Collapsing modular counting in bounded arithmetic and constant depth propositional proofs. *Trans. Amer. Math. Soc.*, 367(11):7517–7563, 2015. doi:10.1090/S0002-9947-2015-06233-3. 13

[Bla37] Archie Blake. *Canonical expressions in Boolean algebra*. PhD thesis, The University of Chicago, 1937. 3

[BM04] Josh Buresh-Oppenheim and Tsuyoshi Morioka. Relativized NP search problems and propositional proof systems. In *CCC*, pages 54–67. IEEE Computer Society, 2004. doi:10.1109/CCC.2004.1313795. 7, 19

[BM20] Jan Bydzovsky and Moritz Müller. Polynomial time ultrapowers and the consistency of circuit lower bounds. *Arch. Math. Log.*, 59(1-2):127–147, 2020. doi:10.1007/S00153-019-00681-Y. 16

[BP96] Paul Beame and Toniann Pitassi. Simplified and improved resolution lower bounds. In *FOCS*, pages 274–282. IEEE, 1996. doi:10.1109/SFCS.1996.548486. 3, 6, 8, 9, 13, 15, 18, 26, 27, 28, 29, 39

[Bro11] L. E. J. Brouwer. Über abbildung von mannigfaltigkeiten. *Mathematische Annalen*, 71:97–115, 1911. In German. doi:10.1007/BF01456931. 3

[Bus85] Samuel R. Buss. *Bounded arithmetic.* Princeton University, 1985. 2, 17

[BW01] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow - resolution made simple. *J. ACM*, 48(2):149–169, 2001. doi:10.1145/375827.375835. 3, 6, 8, 18, 25, 42, 43, 45, 50

[CCT87] William J. Cook, Collette R. Coullard, and György Turán. On the complexity of cutting-plane proofs. *Discret. Appl. Math.*, 18(1):25–38, 1987. doi:10.1016/0166-218X(87)90039-4. 4

[CDT09] Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player Nash equilibria. *J. ACM*, 56(3):1–57, 2009. doi:10.1145/1516512.1516516. 3

[CEI96] Matthew Clegg, Jeffery Edmonds, and Russell Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *STOC*, pages 174–183, 1996. doi:10.1145/237814.237860. 4, 6, 13

[CJSW24] Lijie Chen, Ce Jin, Rahul Santhanam, and Ryan Williams. Constructive separations and their consequences. *TheoretiCS*, volume 3, February 2024. doi:10.46298/theoretics.24.3. 2, 3, 12, 16

[CK07] Stephen A. Cook and Jan Krajíček. Consequences of the provability of $\mathsf{NP} \subseteq \mathsf{P}/_{\text{poly}}$. *J. Symb. Log.*, 72(4):1353–1371, 2007. doi:10.2178/JSL/1203350791. 16

[CKKO21] Marco Carmosino, Valentine Kabanets, Antonina Kolokolova, and Igor C. Oliveira. LEARN-uniform circuit lower bounds and provability in bounded arithmetic. In *FOCS*, pages 770–780. IEEE, 2021. doi:10.1109/FOCS52979.2021.00080. 16, 17

[CLO24a] Lijie Chen, Jiatu Li, and Igor C. Oliveira. Reverse mathematics of complexity lower bounds. In *FOCS*, 2024. URL: https://eccc.weizmann.ac.il/report/2024/060/. 2, 3, 6, 7, 12, 16, 21, 24, 57

[CLO24b] Lijie Chen, Jiatu Li, and Igor Carboni Oliveira. On the unprovability of circuit size bounds in intuitionistic S^1_2. *Electron. Colloquium Comput. Complex.*, pages TR24–083, 2024. URL: https://eccc.weizmann.ac.il/report/2024/083. 1

[CN10] Stephen A. Cook and Phuong Nguyen. *Logical Foundations of Proof Complexity*, volume 11. Cambridge University Press, 2010. doi:10.1017/CBO9780511676277. 21

[Cob64] Alan Cobham. The intrinsic computational difficulty of functions. In *Proc. Logic, Methodology, and the Philosophy of Science*, pages 24–30, 1964. 21

[Coo75] Stephen A. Cook. Feasibly constructive proofs and the propositional calculus (preliminary version). In *STOC*, pages 83–97. ACM, 1975. doi:10.1145/800116.803756. 21

[Coo07] Stephen A. Cook. Bounded reverse mathematics, 2007. Plenary lecture for CiE 2007. 1

[CP90] Stephen A. Cook and Toniann Pitassi. A feasibly constructive lower bound for resolution proofs. *Inf. Process. Lett.*, 34(2):81–85, 1990. doi:10.1016/0020-0190(90)90141-J. 1, 9, 13, 18

[CS88] Vasek Chvátal and Endre Szemerédi. Many hard examples for resolution. *J. ACM*, 35(4):759–768, 1988. doi:10.1145/48014.48016. 3, 10, 39, 46, 47, 48, 50, 52

[CTW23] Lijie Chen, Roei Tell, and Ryan Williams. Derandomization vs refutation: A unified framework for characterizing derandomization. In *FOCS*, pages 1008–1047. IEEE, 2023. doi:10.1109/FOCS57990.2023.00062. 1, 2, 3, 16

[DGP09] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a Nash equilibrium. *SIAM J. Comput.*, 39(1):195–259, 2009. doi:10.1137/070699652. 3

[DLL62] Martin Davis, George Logemann, and Donald W. Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, 1962. doi:10.1145/368273.368557. 3

[DP60] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, 1960. doi:10.1145/321033.321034. 3

[DR03] Stefan S. Dantchev and Søren Riis. On relativisation and complexity gap for resolution-based proof systems. In *CSL*, volume 2803 of *Lecture Notes in Computer Science*, pages 142–154. Springer, 2003. doi:10.1007/978-3-540-45220-1_14. 3, 10, 39, 40

[dRGN+21] Susanna F. de Rezende, Mika Göös, Jakob Nordström, Toniann Pitassi, Robert Robere, and Dmitry Sokolov. Automating algebraic proof systems is NP-hard. In *STOC*, pages 209–222. ACM, 2021. doi:10.1145/3406325.3451080. 10, 12, 15, 16, 19, 22, 23, 51, 57

[dRGR22] Susanna F. de Rezende, Mika Göös, and Robert Robere. Proofs, circuits, and communication. *SIGACT News*, 53(1):59–82, 2022. doi:10.1145/3532737.3532746. 11, 19, 55

[dRMN+20] Susanna F. de Rezende, Or Meir, Jakob Nordström, Toniann Pitassi, Robert Robere, and Marc Vinyals. Lifting with simple gadgets and applications to circuit and proof complexity. In *FOCS*, pages 24–30. IEEE, 2020. doi:10.1109/FOCS46700.2020.00011. 40

[dRNV16] Susanna F. de Rezende, Jakob Nordström, and Marc Vinyals. How limited interaction hinders real communication (and what it means for proof and circuit complexity). In *FOCS*, pages 295–304. IEEE, 2016. doi:10.1109/FOCS.2016.40. 40

[Ebt23] Mohammad Hossein Ebtehaj. Variants of pseudo-deterministic algorithms and duality in TFNP. Master's thesis, University of Waterloo, 2023. URL: http://hdl.handle.net/10012/19721. 16

[ER60] Paul Erdös and Richard Rado. Intersection theorems for systems of sets. *Journal of the London Mathematical Society*, 1(1):85–90, 1960. doi:10.1112/jlms/s1-35.1.85. 50

[FGPR24] Noah Fleming, Stefan Grosser, Toniann Pitassi, and Robert Robere. Black-box PPP is not Turing-closed. In *STOC*, pages 1405–1414. ACM, 2024. doi:10.1145/3618260.3649769. 24

[Gar19] Michal Garlík. Resolution lower bounds for refutation statements. In *MFCS*, volume 138 of *LIPIcs*, pages 37:1–37:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICS.MFCS.2019.37. 11, 12, 51, 57

[Gar24] Michal Garlík. Failure of feasible disjunction property for k-DNF resolution and NP-hardness of automating it. In *CCC*, volume 300 of *LIPIcs*, pages 33:1–33:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.CCC.2024.33. 12

[GGKS20] Ankit Garg, Mika Göös, Pritish Kamath, and Dmitry Sokolov. Monotone circuit lower bounds from resolution. *Theory Comput.*, 16:1–30, 2020. doi:10.4086/TOC.2020.V016A013. 40, 50

[GHJ+22] Mika Göös, Alexandros Hollender, Siddhartha Jain, Gilbert Maystre, William Pires, Robert Robere, and Ran Tao. Separations in proof complexity and TFNP. In *FOCS*, pages 1150–1161. IEEE, 2022. doi:10.1109/FOCS54457.2022.00111. 7

[GKMP20] Mika Göös, Sajin Koroth, Ian Mertz, and Toniann Pitassi. Automating cutting planes is NP-hard. In *STOC*, pages 68–77. ACM, 2020. doi:10.1145/3357713.3384248. 12, 19

[GP18a] Paul W. Goldberg and Christos H. Papadimitriou. Towards a unified complexity theory of total functions. *J. Comput. Syst. Sci.*, 94:167–192, 2018. doi:10.1016/J.JCSS.2017.12.003. 13, 17, 67

[GP18b] Mika Göös and Toniann Pitassi. Communication lower bounds via critical block sensitivity. *SIAM J. Comput.*, 47(5):1778–1806, 2018. doi:10.1137/16M1082007. 40

[GST07] Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. If NP languages are hard on the worst-case, then it is easy to find their hard instances. *Comput. Complex.*, 16(4):412–441, 2007. doi:10.1007/S00037-007-0235-8. 2

[Hak85] Armin Haken. The intractability of resolution. *Theor. Comput. Sci.*, 39:297–308, 1985. doi:10.1016/0304-3975(85)90144-6. 1, 2, 3, 5, 6, 9, 18, 27, 29

[Hak95] Armin Haken. Counting bottlenecks to show monotone $\mathsf{P} \neq \mathsf{NP}$. In *FOCS*, pages 36–40. IEEE Computer Society, 1995. doi:10.1109/SFCS.1995.492460. 50

[HN12] Trinh Huynh and Jakob Nordström. On the virtue of succinct proofs: Amplifying communication complexity hardness to time-space trade-offs in proof complexity. In *STOC*, pages 233–248. ACM, 2012. doi:10.1145/2213977.2214000. 40

[ILW23] Rahul Ilango, Jiatu Li, and R. Ryan Williams. Indistinguishability obfuscation, range avoidance, and bounded arithmetic. In *STOC*, pages 1076–1089. ACM, 2023. doi:10.1145/3564246.3585187. 5

[IPS99] Russell Impagliazzo, Pavel Pudlák, and Jirí Sgall. Lower bounds for the polynomial calculus and the gröbner basis algorithm. *Comput. Complex.*, 8(2):127–144, 1999. doi:10.1007/S000370050024. 4

[IR22] Dmitry Itsykson and Artur Riazanov. Automating OBDD proofs is np-hard. In *MFCS*, volume 241 of *LIPIcs*, pages 59:1–59:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.MFCS.2022.59. 12

[Jeř04] Emil Jeřábek. Dual weak pigeonhole principle, Boolean complexity, and derandomization. *Ann. Pure Appl. Log.*, 129(1-3):1–37, 2004. doi:10.1016/j.apal.2003.12.003. 6, 21, 24

[Jeř07a] Emil Jeřábek. Approximate counting in bounded arithmetic. *J. Symb. Log.*, 72(3):959–993, 2007. doi:10.2178/JSL/1191333850. 7, 50

[Jeř07b] Emil Jeřábek. On independence of variants of the weak pigeonhole principle. *J. Log. Comput.*, 17(3):587–604, 2007. doi:10.1093/LOGCOM/EXM017. 6, 24

[Jeř09] Emil Jeřábek. Approximate counting by hashing in bounded arithmetic. *J. Symb. Log.*, 74(3):829–860, 2009. doi:10.2178/JSL/1245158087. 7, 50

[JPY88] David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. How easy is local search? *J. Comput. Syst. Sci.*, 37(1):79–100, 1988. doi:10.1016/0022-0000(88)90046-3. 3, 8, 19

[Kam19] Pritish Kamath. *Some hardness escalation results in computational complexity theory.* PhD thesis, Massachusetts Institute of Technology, 2019. URL: https://hdl.handle.net/1721.1/128290. 11

[KKMP21] Robert Kleinberg, Oliver Korten, Daniel Mitropolsky, and Christos H. Papadimitriou. Total functions in the polynomial hierarchy. In *ITCS*, volume 185 of *LIPIcs*, pages 44:1–44:18, 2021. doi:10.4230/LIPIcs.ITCS.2021.44. 6

[KNT11] Leszek Aleksander Kołodziejczyk, Phuong Nguyen, and Neil Thapen. The provably total NP search problems of weak second order bounded arithmetic. *Ann. Pure Appl. Log.*, 162(6):419–446, 2011. doi:10.1016/J.APAL.2010.12.002. 17

[KO17] Jan Krajíček and Igor C. Oliveira. Unprovability of circuit upper bounds in Cook's theory PV. *Log. Methods Comput. Sci.*, 13(1), 2017. doi:10.23638/LMCS-13(1:4)2017. 16

[Kor21] Oliver Korten. The hardest explicit construction. In *FOCS*, pages 433–444. IEEE, 2021. doi:10.1109/FOCS52979.2021.00051. 6, 24

[Kor22] Oliver Korten. Derandomization from time-space tradeoffs. In *CCC*, volume 234 of *LIPIcs*, pages 37:1–37:26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CCC.2022.37. 2, 3, 6, 7, 12, 16, 24

[KP89] Jan Krajíček and Pavel Pudlák. Propositional provability and models of weak arithmetic. In *CSL*, volume 440 of *Lecture Notes in Computer Science*, pages 193–210. Springer, 1989. doi:10.1007/3-540-52753-2_40. 1

[Kra95] Jan Krajíček. *Bounded arithmetic, propositional logic, and complexity theory*, volume 60 of *Encyclopedia of mathematics and its applications.* Cambridge University Press, 1995. doi:10.1017/CBO9780511529948. 20, 21

[Kra97] Jan Krajíček. Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. *J. Symb. Log.*, 62(2):457–486, 1997. doi:10.2307/2275541. 1, 50

[Kra01] Jan Krajíček. On the weak pigeonhole principle. *Fundamenta Mathematicae*, 170(1-2):123–140, 2001. URL: http://eudml.org/doc/282141. 6

[Kra04] Jan Krajíček. Dual weak pigeonhole principle, pseudo-surjective functions, and provability of circuit lower bounds. *J. Symb. Log.*, 69(1):265–286, 2004. doi:10.2178/jsl/1080938841. 6, 24

[Kra11a] Jan Krajíček. A note on propositional proof complexity of some Ramsey-type statements. *Arch. Math. Log.*, 50(1-2):245–255, 2011. doi:10.1007/S00153-010-0212-9. 39, 40

[Kra11b] Jan Krajíček. On the proof complexity of the Nisan-Wigderson generator based on a hard $\mathsf{NP} \cap \mathsf{coNP}$ function. *J. Math. Log.*, 11(1), 2011. doi:10.1142/S0219061311000979. 1

[Kra19] Jan Krajíček. *Proof Complexity.* Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2019. doi:10.1017/9781108242066. 3, 5, 40

[KST07] Jan Krajíček, Alan Skelley, and Neil Thapen. NP search problems in low fragments of bounded arithmetic. *J. Symb. Log.*, 72(2):649–672, 2007. doi:10.2178/JSL/1185803628. 17

[KT22] Leszek Aleksander Kołodziejczyk and Neil Thapen. Approximate counting and NP search problems. *J. Math. Log.*, 22(3):2250012:1–2250012:31, 2022. doi:10.1142/S021906132250012X. 17

[LC11] Dai Tri Man Le and Stephen A. Cook. Formalizing randomized matching algorithms. *Log. Methods Comput. Sci.*, 8(3), 2011. doi:10.2168/LMCS-8(3:5)2012. 27

[LO23] Jiatu Li and Igor C. Oliveira. Unprovability of strong complexity lower bounds in bounded arithmetic. In *STOC*, pages 1051–1057. ACM, 2023. doi:10.1145/3564246.3585144. 1

[Maa84] Wolfgang Maass. Quadratic lower bounds for deterministic and nondeterministic one-tape Turing machines (extended abstract). In *STOC*, pages 401–408. ACM, 1984. doi:10.1145/800057.808706. 2

[Mor01] Tsuyoshi Morioka. Classification of search problems and their definability in bounded arithmetic. Master's thesis, University of Toronto, 2001. URL: https://hdl.handle.net/1807/16458. 19

[MP91] Nimrod Megiddo and Christos H. Papadimitriou. On total functions, existence theorems and computational complexity. *Theor. Comput. Sci.*, 81(2):317–324, 1991. doi:10.1016/0304-3975(91)90200-L. 1, 3

[MP96] Alexis Maciel and Toniann Pitassi. Towards lower bounds for bounded-depth Frege proofs with modular connectives. In *Proof Complexity and Feasible Arithmetics*, volume 39 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 195–227. DIMACS/AMS, 1996. doi:10.1090/DIMACS/039/12. 13

[MP20] Moritz Müller and Ján Pich. Feasibly constructive proofs of succinct weak circuit lower bounds. *Ann. Pure Appl. Log.*, 171(2), 2020. doi:10.1016/j.apal.2019.102735. 1

[MPW02] Alexis Maciel, Toniann Pitassi, and Alan R. Woods. A new proof of the weak pigeonhole principle. *J. Comput. Syst. Sci.*, 64(4):843–872, 2002. doi:10.1006/JCSS.2002.1830. 6

[Mül21] Moritz Müller. Typical forcings, NP search problems and an extension of a theorem of Riis. *Ann. Pure Appl. Log.*, 172(4):102930, 2021. doi:10.1016/J.APAL.2020.102930. 3, 27

[Nas51] John Nash. Non-cooperative games. *Annals of mathematics*, 54(2):286–295, 1951. doi:10.2307/1969529. 3

[Ngu08] Phuong Nguyen. *Bounded reverse mathematics*. PhD thesis, University of Toronto, 2008. 1

[Pap94] Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. Syst. Sci.*, 48(3):498–532, 1994. doi:10.1016/S0022-0000(05)80063-7. 3

[Pap24] Theodoros Papamakarios. Depth-d frege systems are not automatable unless $\mathsf{P} = \mathsf{NP}$. In *CCC*, volume 300 of *LIPIcs*, pages 22:1–22:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.CCC.2024.22. 12

[Pic15] Ján Pich. Circuit lower bounds in bounded arithmetics. *Ann. Pure Appl. Log.*, 166(1):29–45, 2015. doi:10.1016/J.APAL.2014.08.004. 1, 2

[PS12] Toniann Pitassi and Nathan Segerlind. Exponential lower bounds and integrality gaps for tree-like Lovász–Schrijver procedures. *SIAM J. Comput.*, 41(1):128–159, 2012. doi:10.1137/100816833. 6

[PS19] Ján Pich and Rahul Santhanam. Why are proof complexity lower bounds hard? In *FOCS*, pages 1305–1324. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00080. 1

[PS21] Ján Pich and Rahul Santhanam. Strong co-nondeterministic lower bounds for NP cannot be proved feasibly. In *STOC*, pages 223–233. ACM, 2021. doi:10.1145/3406325.3451117. 1

[PS23] Ján Pich and Rahul Santhanam. Towards $\mathsf{P} \neq \mathsf{NP}$ from Extended Frege lower bounds. *arXiv math.LO*, 2023. doi:10.48550/arXiv.2312.08163. 1, 16

[PT12] Pavel Pudlák and Neil Thapen. Alternating minima and maxima, Nash equilibria and Bounded Arithmetic. *Ann. Pure Appl. Log.*, 163(5):604–614, 2012. doi:10.1016/J.APAL.2011.06.014. 17

[PT19] Pavel Pudlák and Neil Thapen. Random resolution refutations. *Comput. Complex.*, 28(2):185–239, 2019. doi:10.1007/S00037-019-00182-7. 7, 11, 23, 50, 52

[Pud97] Pavel Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *J. Symb. Log.*, 62(3):981–998, 1997. doi:10.2307/2275583. 4, 50

[Pud00] Pavel Pudlák. Proofs as games. *Am. Math. Mon.*, 107(6):541–550, 2000. doi:10.2307/2589349. 51, 56, 68, 69, 70

[Pud20] Pavel Pudlák. Reflection principles, propositional proof systems, and theories. *arXiv*, abs/2007.14835, 2020. doi:10.48550/arXiv.2007.14835. 17, 67

[PWW88] Jeff B. Paris, A. J. Wilkie, and Alan R. Woods. Provability of the pigeonhole principle and the existence of infinitely many primes. *J. Symb. Log.*, 53(4):1235–1244, 1988. doi:10.1017/S0022481200028061. 6, 21, 24

[Raz85] Alexander A. Razborov. Lower bounds on the monotone complexity of some Boolean function. In *Soviet Math. Dokl.*, volume 31, pages 354–357, 1985. URL: https://www.mathnet.ru/eng/dan9192. 50

[Raz87] Alexander A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical Notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987. 13

[Raz95a] Alexander A. Razborov. Bounded arithmetic and lower bounds in Boolean complexity. In *Feasible Mathematics II*, pages 344–386. Birkhäuser Boston, 1995. doi:10.1007/978-1-4612-2566-9_12. 1

[Raz95b] Alexander A. Razborov. Unprovability of lower bounds on circuit size in certain fragments of bounded arithmetic. *Izvestiya: mathematics*, 59(1):205, 1995. doi:10.1070/IM1995v059n01ABEH000009. 1, 11, 50

[Raz98] Alexander A. Razborov. Lower bounds for the polynomial calculus. *Comput. Complex.*, 7(4):291–324, 1998. doi:10.1007/S000370050013. 4, 13

[Raz15] Alexander A. Razborov. Pseudorandom generators hard for k-DNF resolution and polynomial calculus resolution. *Annals of Mathematics*, 181(2):415–472, 2015. doi:10.4007/annals.2015.181.2.1. 1

[Rob65] John Alan Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41, 1965. doi:10.1145/321250.321253. 3

[Sch97] Uwe Schöning. Resolution proofs, exponential bounds, and Kolmogorov complexity. In *MFCS*, volume 1295 of *Lecture Notes in Computer Science*, pages 110–116. Springer, 1997. doi:10.1007/BFB0029954. 3, 10, 15, 39, 42, 45

[Smo87] Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *STOC*, pages 77–82. ACM, 1987. doi:10.1145/28395.28404. 13

[Sok20] Dmitry Sokolov. (Semi)algebraic proofs over $\{\pm 1\}$ variables. In *STOC*, pages 78–90, 2020. doi:10.1145/3357713.3384288. 6

[ST11] Alan Skelley and Neil Thapen. The provably total search problems of bounded arithmetic. *Proceedings of the London Mathematical Society*, 103(1):106–138, 2011. doi:10.1112/plms/pdq044. 17

[ST21] Rahul Santhanam and Iddo Tzameret. Iterated lower bound formulas: a diagonalization-based approach to proof complexity. In *STOC*, pages 234–247. ACM, 2021. doi:10.1145/3406325.3451010. 1

[SW14] Rahul Santhanam and Ryan Williams. On uniformity and circuit lower bounds. *Comput. Complex.*, 23(2):177–205, 2014. doi:10.1007/S00037-014-0087-Y. 17

[Tha02] Neil Thapen. *The weak pigeonhole principle in models of bounded arithmetic*. PhD thesis, University of Oxford, 2002. 6, 24

[Tse83] Grigori S Tseitin. On the complexity of derivation in propositional calculus. *Automation of reasoning: 2: Classical papers on computational logic 1967–1970*, pages 466–483, 1983. doi:10.1007/978-3-642-81955-1_28. 42

[Urq87] Alasdair Urquhart. Hard examples for resolution. *J. ACM*, 34(1):209–219, 1987. doi:10.1145/7531.8928. 3, 10, 39, 42

[VS23] Bernhard Von Stengel. Zero-sum games and linear programming duality. *Mathematics of Operations Research*, 2023. doi:10.1287/moor.2022.0149. 3

A Amplification for rwPHP($\mathcal{P}$)

In this section, we prove Theorem 2.11, showing that the relationship between N and M does not influence the complexity of rwPHP($\mathcal{P}$) (provided that M and N are not too close to each other), thus our choice of $N = 2M$ is indeed without loss of generality. This result requires $\mathcal{P}$ to be closed under Turing reductions [BJ12], as defined below:

Definition A.1 (Turing Reductions in $\mathsf{TFNP}^{\mathsf{dt}}$)**.** Let $\mathcal{P}, \mathcal{Q}$ be problems in $\mathsf{TFNP}^{\mathsf{dt}}$. We say there is a time-t (uniform) Turing reduction from $\mathcal{Q}$ to $\mathcal{P}$ if there is a time-t oracle Turing machine $R^{x,\mathcal{P}}$ that solves $\mathcal{Q}$ in the following manner. Let $x \in \{0,1\}^N$ be the input to $\mathcal{Q}$. Besides work tapes and a query tape for access to x, R has another query tapes for access to a $\mathcal{P}$ oracle. Each query q to $\mathcal{P}$ is described as $(1^{t'}, L, M_q^x)$, where L is the length of the query input, and M_q^x is a time-t' Turing machine with query access to x. The input to $\mathcal{P}$ is defined as the L-bit string whose i-th bit is $M_q^x(i)$. The answer to this query would be any valid solution for this L-bit string as an input to $\mathcal{P}$. Finally, for every $x \in \{0,1\}^N$ and every valid computational history of R (i.e., every query to $\mathcal{P}$ is answered correctly), the output of R should be a valid output of x for $\mathcal{Q}$.

Assumption A.2. $\mathcal{P}$ is closed under Turing reductions. More precisely, for some function $\gamma(t)$ (think of $\gamma(t) \le \mathrm{poly}(t)$), if a $\mathsf{TFNP}^{\mathsf{dt}}$ problem $\mathcal{Q}$ admits a time-t Turing reduction to $\mathcal{P}$, then $\mathcal{Q}$ also admits a uniform depth-$\gamma(t)$ decision tree reduction to $\mathcal{P}$.

Recall that $\mathrm{rPHP}(\mathcal{P})_{M\to N}$ denotes the rwPHP($\mathcal{P}$) problem where the purported "surjection" is $f : [M] \to [N]$.

Fact A.3. *Let $M < N_1 \le N_2$, then there is a depth-1 decision tree reduction from $\mathrm{rPHP}(\mathcal{P})_{M\to N_2}$ to $\mathrm{rPHP}(\mathcal{P})_{M\to N_1}$.*

Theorem A.4 (Formal version of Theorem 2.11)**.** *Let $N \ge 2M$ and $\varepsilon > 0$ be parameters, and let $d := \gamma(O(\varepsilon^{-1})) \cdot \gamma(O(\log \frac{N}{M}))$. If Assumption A.2 holds for $\mathcal{P}$, then there is a depth-d decision tree reduction from $\mathrm{rPHP}(\mathcal{P})_{M\to(1+\varepsilon)M}$ to $\mathrm{rPHP}(\mathcal{P})_{M\to N}$.*

We prove Theorem A.4 in two steps: in Lemma A.5 we reduce rwPHP with stretch $(1+\varepsilon)$ (i.e., $\mathrm{rPHP}(\mathcal{P})_{M\to(1+\varepsilon)M}$) to rwPHP with stretch 2, and in Lemma A.6 we reduce rwPHP with stretch 2 to rwPHP with arbitrarily large stretch. Theorem A.4 follows easily from Lemma A.5 and A.6.

Lemma A.5. *Let $M \ge 1$, $\varepsilon > 0$ be parameters. Suppose that Assumption A.2 holds for $\mathcal{P}$. Then there is a depth-$\gamma(O(\varepsilon^{-1}))$ decision tree reduction from $\mathrm{rPHP}(\mathcal{P})_{M\to\lfloor(1+\varepsilon)M\rfloor}$ to $\mathrm{rPHP}(\mathcal{P})_{M\to 2M}$.*

Proof. Without loss of generality, assume that both $\varepsilon \cdot M$ and $d := 1/\varepsilon$ are integers. Let $(f, \{I_y\}, \{g_y\})$ be an instance of $\mathrm{rPHP}(\mathcal{P})_{M\to(1+\varepsilon)M}$ and we want to reduce it to an instance $(f', \{I'_y\}, \{g'_y\})$ of $\mathrm{rPHP}(\mathcal{P})_{M\to 2M}$. Recall that:

- $f : [M] \to [(1+\varepsilon)M]$ is the purported "surjection".
- For every $y \in [(1+\varepsilon)M]$, I_y is a $\mathcal{P}$ instance where every possible answer ans of I_y is labelled with an integer $g_y(ans) \in [M]$.
- The goal is to find some $y \in [(1+\varepsilon)M]$ and a solution ans of I_y such that $f(g_y(ans)) \neq y$.

For every $k \in [d]$ (recall $d = 1/\varepsilon$), define $f_k : [M + k\varepsilon M] \to [M + (k+1)\varepsilon M]$ as the following function: on input $x \in [M + k\varepsilon M]$, if $x < M$, then $f_k(x) := f(x)$; otherwise $f_k(x) := x + \varepsilon M$. The function f' in our reduction is simply $f' := f_{d-1} \circ f_{d-2} \circ \cdots \circ f_0$. Intuitively, if (a weak theory thinks that) $f : [M] \to [(1+\varepsilon)M]$ is a surjection, then (it also thinks that) $f' : [M] \to [2M]$ is a surjection.

Next we define the instances $\{I'_y\}$ and the functions $\{g'_y\}$. Roughly speaking, the input instance $\{I_y\}$ and $\{g_y\}$ defines a *$\mathcal{P}$-computable multi-function* (also denoted as) $g : [(1+\varepsilon)M] \to [M]$, which is a purported inverse of f. By padding g, we obtain $\mathcal{P}$-computable multi-functions $g_k : [M + (k+1)\varepsilon M] \to [M + k\varepsilon M]$ for each $k \in [d]$, and each g_k is a purported inverse of f_k. We compose these multi-functions g_k to obtain a single multi-function $g : [2M] \to [M]$ that can be computed by a Turing reduction to $\mathcal{P}$. Details follow.

Consider a Turing machine with oracle access to I_y and $\mathcal{P}$ that on input $y \in [2M]$, operates as follows. Let $y_d := y$. For each k from $d-1$ downto 0:

- if $y_{k+1} \geq (1+\varepsilon)M$, then we define $y_k := y_{k+1} - \varepsilon M$;
- otherwise we query $\mathcal{P}$ to obtain a valid answer ans_k for $I_{y_{k+1}}$ and let $y_k := g(ans_k)$.

Finally, the machine outputs y_0 (as a purported preimage of y under f').

The computational history of this Turing machine defines a total search problem L_{his} as follows. The input to L_{his} consists of $(f, \{I_y\}, \{g_y\})$, as well as some $y \in [2M]$. The output consists of a sequence $(ans_0, ans_1, \ldots, ans_{d-1})$. Denote $y_d = y$ and

$$y_k = \begin{cases} y_{k+1} - \varepsilon M & \text{if } y_{k+1} \geq (1+\varepsilon)M, \\ g_{y_{k+1}}(ans_k) & \text{otherwise} \end{cases}$$

for each $k \in [d]$. We accept the output if for every k such that $y_{k+1} < (1+\varepsilon)M$, ans_k is a valid solution for $I_{y_{k+1}}$; otherwise we reject the output.

Clearly, the above Turing machine itself is a time-$O(d)$ Turing reduction from L_{his} to $\mathcal{P}$. Since $\mathcal{P}$ is closed under Turing reductions, there is also a depth-$\gamma(O(d))$ mapping reduction from L_{his} to $\mathcal{P}$. That is, there is a depth-$\gamma(O(d))$ decision tree that on input $(f, \{I_y\}, \{g_y\})$ as well as $y \in [2M]$, outputs a $\mathcal{P}$ instance (that we call I'_y), and a mapping that given any valid answer ans of I'_y, finds a valid sequence $(ans_0, ans_1, \ldots, ans_{d-1})$. We compute each $\{y_k\}_{k\in[d+1]}$ as above and define $g'_y(ans) := y_0$.

This finishes the description of our reduction from $\mathrm{rPHP}(\mathcal{P})_{M\to(1+\varepsilon)M}$ to $\mathrm{rPHP}(\mathcal{P})_{M\to 2M}$; it is easy to see that it has depth $\gamma(O(\varepsilon^{-1}))$. Now, given a valid solution (y', ans') for $(f', \{I'_y\}, \{g'_y\})$, we can compute a valid solution (y, ans) for $(f, \{I_y\}, \{g_y\})$ as follows. First, since ans' is a valid solution for $I'_{y'}$, we can unpack ans' to obtain a sequence $(ans_0, ans_1, \ldots, ans_{d-1})$. Then we define each $\{y_k\}_{k\in[d+1]}$ as above (starting with $y_d = y'$). Also, for every $k \in [d+1]$, define $f_{\geq k} := f_{d-1} \circ \cdots \circ f_k$, then $f_{\geq k}$ is a purported surjection from $[M + k\varepsilon M]$ to $[2M]$. (As special cases, $f_{\geq d} : [2M] \to [2M]$ is the identity function and $f_{\geq 0} = f'$.) Since (y', ans') is a valid solution, we know that $f'(g'_{y'}(ans')) \neq y'$, which translates to $f_{\geq 0}(y_0) \neq y_d$. Since $f_{\geq d}(y_d) = y_d$, there is some integer $k \in [d]$ such that $f_{\geq k}(y_k) \neq y_d$ but $f_{\geq k+1}(y_{k+1}) = y_d$. We argue that (y_{k+1}, ans_k) is a valid solution for $(f, \{I_y\}, \{g_y\})$:

- First, it must be the case that $y_{k+1} < (1+\varepsilon)M$. If $y_{k+1} \geq (1+\varepsilon)M$, then $y_k = y_{k+1} - \varepsilon M \geq M$ and thus $f_k(y_k) = y_{k+1}$. It follows that
$$y_d \neq f_{\geq k}(y_k) = f_{\geq k+1}(f_k(y_k)) = f_{\geq k+1}(y_{k+1}) = y_d, \tag{9}$$
a contradiction.
- Since $(ans_0, ans_1, \ldots, ans_{d-1})$ is a valid sequence, ans_k is a valid solution for $I_{y_{k+1}}$.
- Finally, if $f(g(ans_k)) = f(y_k) = y_{k+1}$, then (9) holds, which is a contradiction. Therefore, it must be the case that $f(g_{y_{k+1}}(ans_k)) \neq y_{k+1}$ and thus (y_{k+1}, ans_k) is a valid solution for $(f, \{I_y\}, \{g_y\})$. ☐

Lemma A.6. *Let $N \geq 2M$. Suppose that Assumption A.2 holds for $\mathcal{P}$. There is a depth-$\gamma(O(\log \frac{N}{M}))$ decision tree reduction from $\mathrm{rPHP}(\mathcal{P})_{M\to 2M}$ to $\mathrm{rPHP}(\mathcal{P})_{M\to N}$.*

Proof. The proof is similar to that of Lemma A.5. Without loss of generality, we may assume that $d := \log \frac{N}{M}$ is an integer (i.e., N/M is a power of 2). Let $(f, \{I_y\}, \{g_y\})$ be an instance of $\mathrm{rPHP}(\mathcal{P})_{M\to 2M}$ and we want to reduce it to an instance $(f', \{I'_y\}, \{g'_y\})$ of $\mathrm{rPHP}(\mathcal{P})_{M\to N}$. Recall that:

- $f : [M] \to [2M]$ is the purported "surjection".
- For every $y \in [2M]$, I_y is a $\mathcal{P}$ instance where every possible answer ans of I_y is labelled with an integer $g(ans) \in [M]$.
- The goal is to find some $y \in [2M]$ and a solution ans of I_y such that $f(g_y(ans)) \neq y$.

For every integer $k \in [d]$, we put 2^k copies of the instance $(f, \{I_y\}, \{g_y\})$ in parallel and obtain the instance $(f_k, \{(I_k)_y\}, \{g_{k,y}\}$ of $\mathrm{rPHP}(\mathcal{P})_{(2^k M) \to (2^{k+1} M)}$. More precisely:

<u>(Definition of f_k)</u> Let $x \in [2^k M]$ be the input, and let $x = x_0 \cdot M + x_1$ where $x_0 \in [2^k]$ and $x_1 \in [M]$. We define $f_k(x) := x_0 \cdot 2M + f(x_1)$.

<u>(Definition of $(I_k)_y$ and $g_{k,y}$)</u> Let $y \in [2^{k+1} M]$ be the input, and let $y = y_0 \cdot 2M + y_1$ where $y_0 \in [2^k]$ and $y_1 \in [2M]$. We define $(I_k)_y := I_{y_1}$, and for every ans that is a possible solution of $(I_k)_y = I_{y_1}$, define $g_{k,y}(ans) := y_0 \cdot M + g_y(ans)$.

The mapping from $(f, \{I_y\}, \{g_y\})$ to $(f_k, \{(I_k)_y\}, \{g_{k,y}\})$ can be computed by a depth-1 decision tree. Given a valid solution (y, ans) for $(f_k, \{(I_k)_y\}, \{g_{k,y}\})$, we write $y = y_0 \cdot 2M + y_1$ where $y_0 \in [2^k]$ and $y_1 \in [2M]$. Since ans is a solution of $(I_k)_y = I_{y_1}$ and

$$y_0 \cdot 2M + y_1 = y \neq f_k(g_{k,y}(ans)) = y_0 \cdot 2M + f(g(ans)) \implies f(g_y(ans)) \neq y_1,$$

it follows that (y_1, ans) is also a valid solution for $(f, \{I_y\}, \{g_y\})$. Therefore, there is a depth-1 decision tree reduction from $\mathrm{rPHP}(\mathcal{P})_{M \to 2M}$ to $\mathrm{rPHP}(\mathcal{P})_{(2^k M) \to (2^{k+1} M)}$.

Now, we compose the instances $(f_k, \{(I_k)_y\}, g_{k,y})$ for every $k \in [d]$ to obtain the instance $(f', \{I'_y\}, \{g'_y\})$. In particular, the "surjection" $f' : [M] \to [2^d M]$ is defined as $f' := f_{d-1} \circ f_{d-2} \circ \cdots \circ f_0$.

To define I'_y and g'_y, consider the Turing machine with oracle access to $\mathcal{P}$ that, on input $y \in [2^d M]$, operates as follows. Let $y_d := y$. For each k from $d-1$ to 0, the machine queries $\mathcal{P}$ to obtain a valid answer ans_k for $(I_k)_{y_{k+1}}$, and then sets $y_k := g_{k,y_{k+1}}(ans_k)$. Finally, the machine outputs the number $y_0 \in [M]$.

We define a total search problem L_{his} based on the computational history of this machine. The input of L_{his} consists of $M, N, (f, \{I_y\}, \{g_y\})$, as well as some $y \in [2^d M]$; note that given these inputs, one can define the $\mathrm{rPHP}(\mathcal{P})_{(2^k M) \to (2^{k+1} M)}$ instances $(f_k, \{(I_k)_y\}, \{g_{k,y}\})$ as before. The output consists of a sequence $(ans_0, ans_1, \ldots, ans_{d-1})$. Denoting $y_d = y$ and $y_k = g_{k,y_{k+1}}(ans_k)$ for every $k \in [d]$, accept the output if for every $k \in [d]$, ans_k is a valid solution for $(I_k)_{y_{k+1}}$; otherwise reject the output.

Clearly, the above Turing machine itself is a time-$O(d)$ Turing reduction from L_{his} to $\mathcal{P}$. Since $\mathcal{P}$ is closed under Turing reductions, there is also a depth-$\gamma(O(d))$ mapping reduction from L_{his} to $\mathcal{P}$. Therefore, there is a depth-$\gamma(O(d))$ decision tree that on input $(f, \{I_y\}, \{g_y\})$ as well as $y \in [2^d M]$, outputs a $\mathcal{P}$ instance (that we call I'_y), and a mapping that given any valid answer ans of I'_y, finds a valid sequence $(ans_0, ans_1, \ldots, ans_{d-1})$. We define $g'_y(ans) := g_{0,y}(ans_0)$.

This finishes the description of our reduction from $\mathrm{rPHP}(\mathcal{P})_{M \to 2M}$ to $\mathrm{rPHP}(\mathcal{P})_{M \to N}$; it is easy to see that it has depth $\gamma(O(\log \frac{N}{M}))$. Now, given a valid solution (y', ans') for $(f', \{I'_y\}, \{g'_y\})$, we can compute a valid solution (y, ans) for $(f, \{I_y\}, \{g_y\})$ as follows. First, since ans' is a valid solution for $I'_{y'}$, we can unpack ans' to obtain a sequence $ans_0, ans_1, \ldots, ans_{d-1}$. Let $y_d = y'$ and $y_k = g_{k,y_{k+1}}(ans_k)$ for every k from $d-1$ downto 0, then $f'(y_0) \neq y_d$. For every $k \in \{0, 1, \ldots, d\}$, let $f_{\geq k} := f_{d-1} \circ f_{d-2} \circ \cdots \circ f_k$; notice that $f_{\geq k}$ is a purported surjection from $[2^k M]$ to $[2^d M]$. (Note that as special cases, $f_{\geq 0} = f'$ and $f_{\geq d} : [2^d M] \to [2^d M]$ is the identity function.) Since $f_{\geq 0}(y_0) \neq y_d$ but $f_{\geq d}(y_d) = y_d$, there is an integer $k \in [d]$ such that $f_{\geq k}(y_k) \neq y_d$ but $f_{\geq (k+1)}(y_{k+1}) = y_d$. We claim that (y_{k+1}, ans_k) is a valid solution to the instance $(f_k, \{(I_k)_y\}, \{g_{k,y}\})$.

- Since $(ans_0, ans_1, \ldots, ans_{d-1})$ is a valid solution of L_{his} on input $(f, \{I_y\}, \{g_y\}, y')$, ans_k is a valid solution for $(I_k)_{y_{k+1}}$.

- Suppose $f_k(g_{k,y_{k+1}}(ans_k)) = y_{k+1}$, then $f_{\geq k}(y_k) = f_{\geq(k+1)}(f_k(g_{k,y_{k+1}}(ans_k))) = f_{\geq(k+1)}(y_{k+1})$. However, the RHS is equal to y_d while the LHS is not equal to y_d. Therefore it must be the case that $f_k(g_{k,y_{k+1}}(ans_k)) \neq y_{k+1}$.

It follows that given a valid solution for $(f', \{I'_y\}, \{g'_y\})$, one can always find some k and a valid solution for $(f_k, \{(I_k)_y\}, \{g_{k,y}\})$. That is, there is a depth-$\gamma(O(\log \frac{N}{M}))$ reduction from solving $\mathrm{rPHP}(\mathcal{P})_{M\to N}$ to solving one of $\{\mathrm{rPHP}(\mathcal{P})_{(2^k M)\to(2^{k+1}M)}\}_{k\in[d]}$. Composing this with the aforementioned depth-1 reduction from $\mathrm{rPHP}(\mathcal{P})_{M\to 2M}$ to $\mathrm{rPHP}(\mathcal{P})_{(2^k M)\to(2^{k+1}M)}$ completes our reduction. □

B Comparing Refuter with WrongProof(Res)

We discuss the similarities and differences between the refuter problems and the WrongProof problem. We first recall the formal definition of WrongProof(Res) [BB17, GP18a]:

Problem WrongProof(Res)

Input: A CNF F with n variables and k clauses; a purported resolution refutation Π for F represented as $C_{-k}, \ldots, C_{-1}, C_0, C_1, \ldots, C_{L-1}$, where $C_{-k}, \ldots, C_{-1}$ are axioms of F, $C_{L-1} = \bot$, and $L = 2^{n^{\Omega(1)}}$; and a purported satisfying assignment $\alpha \in \{0,1\}^n$.

Output: A number $i \in [L]$ such that C_i is obtained by an invalid resolution derivation, or a number $-k \leq j \leq -1$ such that α does not satisfy C_j.

At first glance, the Refuter problem looks similar to the WrongProof problem. First, both problems take as input a purported (but not correct) resolution proof. Second, both are looking for an invalid derivation as a solution. Moreover, when we consider the resolution proof system (and consider refuting *width* lower bounds), both WrongProof and Refuter are PLS-complete.

However, we think that they are fundamentally different. One primary difference is the reason of totality: When introduced to a (non-promise) TFNP problem, the initial inquiry ought to be: *why is the problem total?* The totality of WrongProof(Res) follows from the *reflection principle* for resolution [Pud20, BFI23], i.e., it is impossible to derive $\bot$ from a satisfiable CNF. The same reasoning holds for every sound proof system, regardless of their power. However, the totality of Refuter is far from trivial: *They rely on non-trivially proven width or size lower bounds.*

Furthermore, for comparison with Refuter, we include a proof that WrongProof(Res) is PLS-complete (this is a folklore result, see e.g., [BFI23]). The proof is seemingly similar to that of Theorem 4.1 and Theorem 5.1, but there are crucial differences. For example, the reduction from WrongProof(Res) to PLS is uniform, since the totality of WrongProof(Res) relies on simpler reasoning. In contrast, the uniform PLS-membership of Refuter$(w(F \vdash_{\mathsf{Res}} \bot))$ crucially relies on nice properties of the family of CNFs (e.g., EPHP), and it is possible that for some families, the refuter problem cannot be uniformly reduced to PLS at all. This demonstrates another difference between WrongProof and Refuter.

Lemma B.1. *WrongProof*(Res) *is in* PLS.

Proof. Let $(C_{-k}, \ldots, C_{-1}, C_0, \ldots, C_{L-1})$ be a purported resolution refutation of a CNF F, and α be a purported satisfying assignment of F. We will reduce this WrongProof(Res) instance to an instance $S : \{-k, \ldots, L-1\} \to \{-k, \ldots, L-1\}$ of reversed Iter.

It would be convenient to think of a clause C_i as "active" if $C_i(\alpha) = 0$. An invalid derivation in the resolution refutation corresponds to an edge from an active node to an inactive node. For every $i \in \{-k, \ldots, L-1\}$, if $C_i(\alpha) = 1$ (i.e., C_i is inactive), then we define $S(i) = i$. Otherwise, if $i < 0$ (i.e., C_i is an axiom *not* satisfied by α), then we define $S(i) = 0$, making i a solution since $S(i) > i$. Otherwise, suppose C_i is derived from C_j (i.e., C_i is a *weakening* of C_j, or C_i is *resolved* from C_j and some other C_k), then we define $S(i) = j$. If i is a solution for the reversed Iter instance S, then either $j < i$ or j is inactive (which means $S(j) = j$), and in either case i is a valid solution for WrongProof(Res). □

Remark 9. The proof above is easy, but one can see that the crucial components are 1) the resolution proof system is sound, and 2) a resolution proof is a "DAG"-like structure. This proof strategy can potentially be easily extended to other proof systems with similar properties.

Lemma B.2. *WrongProof*(Res) *is* PLS-*hard.*

Proof. We will reduce any reversed Iter instance to an instance of WrongProof(Res). The construction below is very similar to the proof of Theorem 4.1. In fact, all clauses and derivations in the construction of the proof of Theorem 4.1 are sound except for the solutions of the given reversed Iter instance.

Let F be any satisfiable CNF with k clauses $C_{-k}, \dots, C_{-1}$ and α be any satisfying assignment of F. Without loss of generality assume there are two clauses C_{-2} and C_{-1} that we can apply a valid resolution step and call the resolved clause D. Let $S : [L] \to [L]$ be an instance of reversed Iter where $S(L) < L$. We construct a purported resolution refutation $\Pi = (C_{-k}, \dots, C_{-1}, C_0, \dots, C_{L-1})$ as follows:

- For every i such that $S(i) = i$, we let $C_i := D$ to be *resolved* from C_{-2} and C_{-1}.
- For every i that is a solution for S, let $C_i := \perp$ be a *weakening* from an axiom (say C_{-k}). Note that this weakening step is invalid and C_i becomes a solution for the WrongProof(Res) instance.
- Finally, for every i such that $S(i) < i$ and $S(S(i)) < S(i)$, let $C_i := \perp$ be a *weakening* of $C_{S(i)}$. Note that $C_{S(i)}$ is also $\perp$, hence this is a valid derivation.

It is easy to see that the invalid derivations in Π correspond exactly to the solutions of S. ☐

C Prover-Delayer Games, PLS, and the Proof of Lemma 6.10

In Section C.1, we provide a self-contained description of the transformation from a PLS formulation to a low-width resolution proof using *Prover-Delayer* game, along with several properties of this transformation that are useful when proving Lemma 6.10. We then prove Lemma 6.10 in Section C.2.

C.1 From PLS to Resolution using Prover-Delayer Game

Introduced by Pudlák [Pud00], the *Prover-Delayer game* provides an elegant characterization of resolution width. There are two players in the game, the *Prover* (she) and the *Delayer* (he). Fixing an unsatisfiable CNF formula F, and let $x = (x_1, \dots, x_n)$ be the variables in F. At first, the Prover's memory is empty. Then, in each step, she can either

- *query* the Delayer for the value of a certain variable, and add that value to her memory;
- *forget* the value of a certain variable stored in her memory; or
- *output* a clause of F that is falsified by the partial assignment stored in her memory, which means she wins the game.

We assume the Delayer also has access to Prover's memory. If the Prover queries a variable that is currently in its memory, then the Delayer's answer must be consistent with the memory; otherwise, his answer could be arbitrary. Note that if the Prover queries a variable, forgets it, and queries it again, the Delayer is allowed to answer different values to these two queries of the same variable.

Of course, Prover can always win the game by querying all variables without forgetting any of them. However, for the connection with resolution width, her goal is to win the game with the minimum memory size, where the memory size is the maximum number of variables she remembered during the whole execution of the game. The Delayer is *adversarial* to Prover's goal, i.e., wants her to spend as much memory as possible.

The following theorem shows that the minimum resolution width of an unsatisfiable CNF is characterized by the minimum Prover memory in the corresponding Prover-Delayer game.

Theorem C.1 ([Pud00])**.** *For any unsatisfiable CNF formula F, there exists a width-w resolution refutation of F if and only if there is a winning strategy for Prover using memory size w in the Prover-Delayer game for F.*

In this section, we prove the "if" direction in the previous lemma and highlight some nice properties of the obtained low-width resolution proof that will be helpful when proving Lemma 6.10.

Making Prover's strategy uniform. Note that both Prover's and Delayer's strategies could be quite non-uniform in the Prover-Delayer game model described above. Here, we twist the model a little bit by allowing the Prover to explicitly store several *state registers* in its memory, besides a partial assignment of variables. These internal state registers are also counted in the memory size of her strategy. Later, in the proof of Lemma 6.10, it is more convenient to describe a uniform Prover's strategy with state registers.

From PLS to Prover-Delayer game. A PLS formulation of a search problem $\mathrm{Search}(F_n)$ is a decision tree reduction $(f_i, g_o)_{i,o\in M}$ from $\mathrm{Search}(F_n)$ to $\textsc{Iter}_M$. Let $x = (x_1, \dots, x_n)$ be the variables in F_n, then we have $S(v) \coloneqq f_v(x)$, where S is the successor function in the $\textsc{Iter}_M$ instance reduced from $\mathrm{Search}(F_n)$.

Lemma C.2 (Folklore)**.** *Given a* PLS *formulation $(f, g)_{i,o\in M}$ of depth d for* $\mathrm{Search}(F_n)$*, there exists a Prover's strategy of memory size $O(d + \log M)$ for F_n.*

Proof. We say the Prover queries a decision tree T if she evaluates $T(x)$, and stores the queried variables in her memory in each step. Now we describe the Prover's strategy in what follows.

1. The Prover starts from the node 0 of the $\textsc{Iter}_M$ instance and queries the decision tree f_0. If $f_0(x) = 0$, then 0 is a valid solution for the $\textsc{Iter}_M$ instance, hence she can query $g_0(x)$ to obtain a falsified clause in F_n. Otherwise, we say that the Prover is currently at node $v = f_0$ and previously visited node 0.

2. Assume the Prover is at node $v \in [M]$, and the previous node she visited is u. She queries the decision tree f_v and obtains the next node $w = f_v(x)$.

3a. If $w \le v$, then she has found a solution of the $\textsc{Iter}_M$ instance. In particular, if $w < v$ then the solution is v; if $w = v$ then the solution is u. Note that all the variables queried by f_u, f_v, g_u are still in her memory. Therefore, the clause F returned by $g_v(x)$ (if $w < v$) or $g_u(x)$ (if $w = v$) must be falsified by the variables in her memory.

3b. If $w > v$, then the Prover forgets all the variables that are queried in f_u and not queried in f_v. She then updates the current node as w and the previous node as v, and loops back to **Step 2**.

The Prover's strategy will always end, since the index of v increases in every step. The Prover needs to remember at most $3d$ variables at any time, and $O(d + \log M)$ bits to remember the current state to execute this strategy. ☐

By further examining the proof of Lemma C.2, we obtain several properties that are useful for the proof of Lemma 6.10 later.

Observation C.3. *In Lemma C.2, the Prover's strategy can be implemented in a uniform manner if the* PLS *formulation (f, g) is given via oracle access.*

Lemma C.4. *In Lemma C.2, there exists an efficient binary encoding of Prover's memory, such that:*

1. *The encoding has bit-length* $\mathrm{poly}(d, \log M)$*.*
2. *It is computationally efficient to transform Prover's memory into an encoding and vice versa.*

3. *The encoding of the Prover's memory is lexicographically increasing as the Prover's strategy proceeds.*

4. *We say an encoding is* invalid *if it is in the wrong format, or Prover's internal state registers are inconsistent with the partial assignment of variables w.r.t.* (f, g). *There is an efficient uniform algorithm for checking whether an encoding is invalid given oracle access to* (f, g).

Proof. The Prover's internal state registers should store the index of the current node, the previous node, and its current location in the decision tree it is querying. This part is lexicographically increasing since we always have $w > v$ in **Step 3b**. Our encoding consists of these internal state registers followed by the partial assignment of variables.

It is trivial to construct such an encoding scheme satisfying conditions 1,2,3, and easy to check the correctness of its format. To check the validity of an encoding, we can query the (at most 3) decision tree paths corresponding to the Prover's internal state registers, and check whether they are consistent with the partial assignment. □

From Prover-Delayer games to resolution proofs

Lemma C.5 ([Pud00]). *Given a Prover's strategy of memory cost* w *for* F_n*, there exists a width-*w *resolution proof refuting* F_n.

Proof. Without loss of generality, we assume the Prover will not query a variable that is already in her memory.

We simulate the Prover's strategy. Initially, there is no variable stored in her memory, and it corresponds to the empty clause $\perp$ at the end of the resolution proof. We then generate the resolution proof recursively by maintaining the Prover's memory and the current node of the resolution proof.

In each step, let ρ be the current partial assignment of variables stored in the Prover's memory. Define $C(\rho)$ to be the only clause that is falsified by ρ, using only the variables that are set in ρ. For example, if ρ is $\{x_1 = 1, x_2 = 0\}$, the $C(\rho) = \neg x_1 \vee x_2$. The procedure will guarantee that $C(\rho)$ is the clause of the current node in the resolution proof.

The Prover has three possible actions given ρ and its internal state register:

FORGET If the Prover decides to forget x_i, then we generate a new node C' with clause $C(\rho_{-i})$, where ρ_{-i} is the partial assignment by *forgetting* the value of x_i from ρ. We mark that the current node is derived by a *weakening* step from node C'. We then update the Prover's memory and continue our process at node C' recursively.

QUERY If the Prover queries x_i, then we generate two new nodes C^0, C^1 with clauses $C(\rho) \vee x_i$ and $C(\rho) \vee \neg x_i$ respectively. We mark that the current node is derived by a *resolution* step from node C^0 and C^1. We first recursively proceed to C^0 by updating Prover's memory with $x_i = 0$, and then proceed to C^1 with $x_i = 1$.

OUTPUT If the Prover outputs a falsified clause D, it must be the case that D is a sub-clause of $C(\rho)$. If D is equal to $C(\rho)$, then we simply stop; otherwise, we add a new node for the clause D and add one or more intermediate *weakening* steps towards the current node.

It is easy to verify that during the process, $C(\rho)$ is always the clause of the current node in the resolution proof. This process will stop since the Prover will stop, and its correctness is guaranteed by the Prover's correctness. Finally, note that the largest clause ever generated in the resolution proof is upper bounded by the memory size of the Prover. □

C.2 Proof of Lemma 6.10

Lemma 6.10. *For any family of unsatisfiable CNF $\mathcal{F}$ that has no* $\mathrm{polylog}(N)$*-width resolution refutation,*

$$\textit{Refuter}_{d,M}(\mathrm{Search}(\mathcal{F}) \to \textit{Iter}) \leq_m \textit{Refuter}(w(\mathcal{F} \vdash_{\mathsf{Res}} \bot) < w_0)$$

for some $w_0 = \mathrm{polylog}(N)$ *that may depend on* d, M*.*

Furthermore, this reduction is uniform when $\mathcal{F}$ is a uniform family of unsatisfiable CNFs.

Proof. Let $\mathcal{F} = \{F_n\}_{n \in \mathbb{N}}$. Suppose we are given a purported depth-d reduction (f, g) from $\mathrm{Search}(F_n)$ to $\textsc{Iter}_M$, i.e., (f, g) is a PLS formulation for $\mathrm{Search}(F_n)$. We now use (f, g) to construct a purported resolution refutation $(C_0, \ldots, C_{L-1})$ for F_n with width $w_0 = \mathrm{poly}(d, \log M)$, while satisfying the following two conditions.

1. Given any index $i \in [L]$, the i-th node C_i can be calculated in $\mathrm{polylog}(n)$ queries to (f, g) uniformly.
2. If the i-th node is invalid, one can recover a pair (ρ, o^*) that refutes (f, g) uniformly given i.

As a high-level plan, we first apply the procedures described in Lemma C.2 which transforms the PLS formulation (f, g) to a Prover's strategy of $O(d + \log M)$ memory for the Prover-Delayer game. We then use the procedure described in Lemma C.5 to convert such a strategy into a width-$O(d+\log M)$ resolution proof for F_n.

We now specify the details in these two steps to make sure the two conditions are met. The first condition can be achieved by letting an index $i \in [L]$ to be an encoding of the Prover's memory in a single step, as described in Lemma C.4.[32]

By Observation C.3, given any valid index (encoding), one can calculate the next action of the Prover using $O(d)$ number of queries to (f, g). Then, for all three actions $\{\mathbf{FORGET}, \mathbf{QUERY}, \mathbf{OUTPUT}\}$, we can also calculate the indices of the one or two previous nodes in the resolution proof uniformly. For any invalid index (encoding), we pad a trivially correct resolution node using axioms from the beginning.

To see the second condition above, note that the procedure described in Lemma C.5 generates an invalid node of the resolution proof only when the Prover is taking an **OUTPUT** step. That is, when the Prover outputs a falsified clause D, D might not be a sub-clause of $C(\rho)$, so the *weakening* steps from D to $C(\rho)$ will be wrong. This happens because a solution o of the $\textsc{Iter}_M$ instance is found by the Prover, but querying g_o does not lead to a clause D in F_n that is falsified by the current partial assignment ρ in her memory.

Note that the partial assignment ρ and o^* used to refute (f, g) can be recovered from the index (encoding) of the invalid node. We then complement the partial assignment ρ to ρ' by setting all unassigned variables that appear in D to satisfy clause D. By definition, the pair (ρ', o) is a valid solution to refute the reduction (f, g).

Finally, if $\mathcal{F}$ is a uniform family of formulas, then the whole reduction is also uniform. □

[32] Note that the encoding described in Lemma C.4 is lexicographically increasing, but we can easily make it lexicographically decreasing by flipping all the bits.