Electronic Colloquium on Computational Complexity, Report No. 60 (2024)

Reverse Mathematics of Complexity Lower Bounds

Lijie Chen[*] Jiatu Li[†] Igor C. Oliveira[‡]

April 4, 2024

Abstract

Reverse mathematics is a program in mathematical logic that seeks to determine which axioms are *necessary* to prove a given theorem. In this work, we systematically explore the reverse mathematics of *complexity lower bounds*. We explore *reversals* in the setting of bounded arithmetic, with Cook's theory PV_1 as the base theory, and show that several natural lower bound statements about communication complexity, error correcting codes, and Turing machines are *equivalent* to widely investigated combinatorial principles such as the weak pigeonhole principle for polynomial-time functions and its variants. As a consequence, complexity lower bounds can be formally seen as fundamental mathematical axioms with far-reaching implications.

The proof-theoretic equivalence between complexity lower bound statements and combinatorial principles yields several new implications for the (un)provability of lower bounds. Among other results, we derive the following consequences:

- Under a plausible cryptographic assumption, the classical single-tape Turing machine $\Omega(n^2)$-time lower bound for Palindrome is unprovable in Jeřábek's theory APC_1. The conditional unprovability of this simple lower bound goes against the intuition shared by some researchers that most complexity lower bounds could be established in APC_1.
- While APC_1 proves one-way communication lower bounds for Set Disjointness, it does not prove one-way communication lower bounds for Equality, under a plausible cryptographic assumption.
- An amplification phenomenon connected to the (un)provability of some lower bounds, under which a quantitatively weak $n^{1+\varepsilon}$ lower bound is provable if and only if a stronger (and often tight) n^c lower bound is provable.
- Feasibly definable randomized algorithms can be feasibly defined deterministically (APC_1 is $\forall\Sigma^b_1$-conservative over PV_1) if and only if one-way communication complexity lower bound for Set Disjointness are provable in PV_1.

[*]Miller Institute for Basic Research in Science. University of California at Berkeley. Email: lijiechen@berkeley.edu
[†]Computer Science & Artificial Intelligence Laboratory. Massachusetts Institute of Technology. Email: jiatuli@mit.edu
[‡]Department of Computer Science. University of Warwick. Email: igor.oliveira@warwick.ac.uk

ISSN 1433-8092

Contents

1 Introduction **1**
1.1 Context and motivation . . . 1
1.2 Results . . . 2
1.3 Techniques . . . 8
1.4 Concluding remarks and open problems . . . 11

2 Preliminaries **12**
2.1 Basic notation . . . 12
2.2 Bounded arithmetic . . . 12
2.3 Combinatorial principles . . . 16

3 Equivalence Class for $\mathsf{WPHP}(\mathsf{PV})$ **18**
3.1 Technical lemmas . . . 18
3.2 CC lower bounds for $\mathsf{EQ} \Leftrightarrow \mathsf{WPHP}(\mathsf{PV})$ . . . 18
3.3 Singleton bound for ECC $\Leftrightarrow \mathsf{WPHP}(\mathsf{PV})$ . . . 23
3.4 Summary of the equivalence class . . . 24

4 Equivalence Class for $\mathsf{WPHP}'(\mathsf{PV})$ **24**
4.1 Formalization and technical lemmas . . . 25
4.2 Single-tape lower bounds $\Rightarrow \mathsf{WPHP}'(\mathsf{PV})$ . . . 27
4.3 $\mathsf{WPHP}'(\mathsf{PV}) \Rightarrow$ uniform CC lower bounds . . . 30
4.4 Uniform CC lower bounds $\Rightarrow$ single-tape lower bounds . . . 31
4.5 Uniform one-way CC lower bounds $\Leftrightarrow \mathsf{WPHP}'(\mathsf{PV})$ . . . 32
4.6 Summary of the equivalence class . . . 33

5 Equivalence Class for $\mathsf{WPHPWIT}(\mathsf{PV})$ **34**
5.1 Technical lemmas . . . 34
5.2 $\mathsf{SetDisj}$ lower bounds and $\mathsf{WPHPWIT}(\mathsf{PV})$ . . . 34
5.3 Distance lower bounds for decodable ECC $\Leftrightarrow \mathsf{WPHPWIT}(\mathsf{PV})$ . . . 36
5.4 Summary of equivalence class . . . 37

6 Consequences **38**
6.1 Conditional unprovability of simple lower bounds . . . 38
6.2 On the derandomization of feasibly definable randomized algorithms . . . 39
6.3 Amplification of lower bounds . . . 41
6.4 Connections to TFNP . . . 41

1 Introduction

1.1 Context and motivation

Establishing lower bounds on the amount of resources needed to solve different computational tasks is a central research direction in theoretical computer science. While this endeavor has been highly successful with respect to certain resources (e.g., communication complexity), in many fundamental settings (e.g., circuit size and running time) progress has been much more limited.

This has motivated a sequence of influential works on the difficulty of establishing complexity lower bounds. In particular, several "barrier" results have been proposed, such as relativization [BGS75], natural proofs [RR97], and algebrization [AW09]. While these results provide useful information on the limitations of certain lower bound techniques, they fall short of providing a systematic explanation for the difficulty of proving lower bounds. For instance, there are approaches for proving lower bounds on circuit size that seem to avoid all the aforementioned barriers, often leading to their own ad-hoc explanations for the lack of lower bounds (see, e.g., [CHO$^+$22]). Perhaps more importantly, these barriers to proving lower bounds do not consider a standard notion of proof, and often cannot say anything relevant about approaches that combine different proof strategies.

In recent years, there has been a surge of results aimed at providing a more principled investigation of the difficulty of proving lower bounds. This research is rooted in methods and perspectives from logic, and consider provability in the standard mathematical sense. *A major goal is to identify a suitable logical theory* T *able to formalize the vast majority of known results from algorithms and complexity, and to determine if complexity lower bounds of interest are (un)provable in the same theory* T. In case an unprovability result of this form is possible, it would be a significant achievement in theoretical computer science similar to other major unprovability results, such as the independence of the continuum hypothesis from ZFC set theory [Göd38, Coh63]. While we appear to be far from achieving this goal, modest progress in this direction can still be beneficial. For instance, it can lead to new proofs of existing theorems, a more robust notion of "barrier" for existing techniques, or simply contribute to our understanding of proof complexity, which investigates the inherent hardness of proving mathematical statements, and is another fundamental research direction in complexity theory.

A vast body of work has highlighted certain fragments of Peano Arithmetic collectively known as Bounded Arithmetic as a natural and robust class of theories for the formalization of both basic and advanced results from algorithms and complexity theory (see, e.g., [Kra95, Oja04, Jeř05, CN10, LC11, Lê14, Pic14, BKT14, KO17, MP20, BKKK20, Gay22] and references therein). These theories aim to capture proofs that manipulate and reason with concepts from a given complexity class (e.g., a proof by induction whose inductive hypothesis can be expressed as an NP predicate). The earliest example was the theory $\mathsf{I}\Delta_0$ introduced by Parikh [Par71], who investigated the intuitive concept of feasibility in arithmetic and considered the infeasibility of exponentiation. Some other widely investigated examples include Cook's theory PV_1 [Coo75], which formalizes polynomial-time reasoning; Jeřábek's theory APC_1 [Jeř07a], which extends PV_1 with the dual weak pigeonhole principle for polynomial-time functions, and formalizes probabilistic polynomial-time reasoning; and Buss's theories S^i_2 and T^i_2 [Bus86], which incorporate induction principles corresponding to different levels of the polynomial-time hierarchy.

In connection with the major goal highlighted above, research on the provability of complexity lower bounds has made notable progress on two complementary fronts: the *formalization* of several known results from algorithms and complexity in theories of bounded arithmetic; and results on the *unprovability* of complexity lower bounds in the same theories (often conditional on a computational assumption).

The former direction has a long and rich history that dates back to the work of Razborov [Raz95a, Raz95b], Krajíček [Kra95, Section 15.2], and other researchers from the nineties. For instance, it is known that PV_1 can prove the PCP Theorem [Pic15b], and that APC_1 can prove several well-known circuit lower bounds [MP20], such as monotone circuit lower bounds for k-Clique, and bounded-depth circuit lower bounds for the Parity function. The references cited above describe numerous other formalizations in PV_1 or in its extensions, such as the explicit construction of expander graphs [BKKK20], and the correctness of randomized polynomial-time matching algorithms [LC11].

On the other hand, despite much effort (see, e.g., [Raz95a, Raz95b, Kra11, Pic15a, MP20] and references therein), much less is known about the unprovability of lower bounds in bounded arithmetic. In an exciting recent work, Pich and Santhanam [PS21], and subsequently Li and Oliveira [LO23], obtained the first *unconditional* results showing that certain complexity lower bounds are unprovable in PV_1, APC_1, and even in stronger theories.

Nevertheless, these recent unprovability results can only deal with significantly strong (non-uniform) NP-vs-$\mathsf{coNSUBEXP}$ style *average-case* lower bounds [PS21, LO23]. Since our understanding of average-case complexity in the setting of sub-exponential size co-nondeterministic circuits is rather limited, it is not ruled out that the corresponding lower bound statements could even be false. No approach to attack the unprovability of *worst-case lower bounds*, let alone central goals such as $\mathsf{NP} \not\subseteq \mathsf{P}_{/\mathsf{poly}}$, is currently known.

1.2 Results

Overview. To advance this research program, we focus in this work on the formalization and unprovability of worst-case lower bounds. Our main contributions can be informally described as follows:

(*i*) *Reverse mathematics of lower bounds.* To pinpoint the exact strength of the theories and axioms needed for the formalization of lower bound statements of interest, we systematically explore the *reverse mathematics of lower bounds.* More precisely, we aim to determine the fundamental principles (axioms) needed to establish certain lower bounds, by showing (over a weak base theory) that the lower bound statement and the corresponding principle are *equivalent* (i.e., can be derived from each other). We succeed in showing that natural lower bound statements from communication complexity, error correcting codes, Turing machines, and certain basic principles for polynomial-time functions are all equivalent over PV_1. In light of these results, complexity lower bounds can be formally seen as fundamental mathematical axioms with far-reaching proof-theoretic implications.

(*ii*) *Consequences for the (un)provability of lower bounds.* The novel correspondences between lower bounds and combinatorial principles provided by our results have several consequences for the provability of worst-case lower bounds. Under plausible cryptographic assumptions, we show that certain two-party communication complexity lower bounds and the classical single-tape Turing machine lower bound for computing Palindrome are unprovable in APC_1. This is surprising, given the results and intuition from previous papers suggesting that many sophisticated lower bounds can be established in APC_1. We obtain in some cases a fine-grained understanding of the provability of lower bounds, showing for instance that APC_1 can prove communication lower bounds for Set Disjointness but is unlikely to prove communication lower bounds for Equality. Additionally, our results highlight an interesting amplification phenomenon connected to the (un)provability of lower bounds in certain theories, under which a quantitatively weak lower bound is provable if and only if a stronger lower bound is provable. Finally, we obtain a completeness result that highlights the centrality of proving communication lower bounds for Set Disjointness in PV_1 in connection to the problem of derandomizing feasible arguments.

We now describe our results and technical contributions in detail.

In reverse mathematics (see [Sti20] for a gentle introduction, and [Sim09, DM22] for more details), we begin by establishing a foundational language and a fundamental axiom system known as the *base theory*. This base theory is deliberately chosen to be limited in its capacity to demonstrate the majority of the theorems one wishes to explore. However, it still possesses sufficient strength to facilitate the formulation of essential definitions required to express these theorems. In this paper, we take the first-order theory PV_1 [Coo75, KPT91, Jeř06] as the base theory. Next, we informally describe this theory (see Section 2.2 for a formal description).

Theory PV_1. This is a first-order theory whose intended model is the set $\mathbb{N}$ of natural numbers, together with the standard interpretation for constants and functions symbols such as $0, +, \times$, etc. (Under a standard translation between natural numbers and binary strings, the theory is equally suitable to reason about string-valued computations.) The language (vocabulary) of PV_1 contains a function symbol for each polynomial-time algorithm $f\colon \mathbb{N}^k \to \mathbb{N}$ (where k is any constant). These function symbols, and the axioms defining them, are obtained through Cobham's convenient characterization of polynomial-time functions via an initial set of basic functions, composition, and an operation called bounded recursion on notation [Cob65]. PV_1 also postulates an induction axiom scheme that simulates binary search. For the purpose of this exposition, one can think of PV_1 as a theory with induction for quantifier-free formulas, which correspond to polynomial-time predicates. For a reader that might not be familiar with bounded arithmetic, we note that PV_1 admits equivalent (and perhaps more accessible) formalizations that do not require Cobham's result, such as the one presented in [Jeř06].[1]

As discussed below and in Section 2, the vocabulary of PV_1 allows us to formalize in a natural way many statements of interest in algorithms and complexity. At the same time, PV_1 can be seen as the weakest theory in Buss's hierarchy (see its characterization from [Jeř06]), which means that while versatile enough to express several statements, it might not be strong enough to prove the theorems that we would like to explore. These properties make PV_1 an excellent choice as the base theory in our investigations of the reverse mathematics of lower bounds.

In our results we will consider some pigeonhole principles that have been widely investigated in bounded arithmetic.

Pigeonhole principles for polynomial-time functions. We informally introduce a few variants of the *pigeonhole principle*, referring to Section 2.3 for the formal treatment. The *pigeonhole*

[1] We might use simply PV when referring to the vocabulary of PV_1, since the first-order formulation of the theory [KPT91] (denoted PV_1) and its equational version [Coo75] (denoted PV) share the same vocabulary.

principle $\mathsf{PHP}(f)$ states that an integer function f cannot be injective if its co-domain is smaller than its domain. On the other hand, the *dual pigeonhole principle* $\mathsf{dPHP}(f)$ states that an integer function f cannot be surjective if its co-domain is larger than its domain. We will also need a "composed" version of these two principles, called *witnessing pigeonhole principle* $\mathsf{PHPWIT}(f, g)$, which states that for any two functions $f : [a] \to [b]$ and $g : [b] \to [a]$ such that $a < b$, $f \circ g$ is not the identity function.[2] It is possible to derive $\mathsf{PHPWIT}(f, g)$ from each of the other two principles. We refer to [Jeř07b] and references therein for much more information about them.

In these principles, we allow the function(s) to take extra parameters, meaning that the principle applies to each choice of the parameter (see Section 2.3). For instance, in the case of a function $f(x, y)$ with two input parameters, $\mathsf{PHP}(f)$ is a sentence stating that, for every choice of the input y, the resulting function $f(\cdot, y)$ cannot be injective if its co-domain is smaller than its domain. Informally, from a computational perspective, one can think of the extra input y as a non-uniform advice string when invoking the pigeonhole principle on $f(\cdot, y)$.

Our results will concern *weak* versions of these principles. The meaning of "weak" is that each principle is postulated not for domains and co-domains of different sizes but of *noticeably* different sizes, i.e., of order 2^n versus $(1 + 1/n) \cdot 2^n$. We write $\mathsf{WPHP}(f)$, $\mathsf{dWPHP}(f)$, and $\mathsf{WPHPWIT}(f)$ to denote the sentence corresponding to the weak version of each principle.

In some results, we will need to explicitly forbid functions with extra input parameters. In this case, we use $\mathsf{WPHP}'(f)$ to denote the "uniform" version of the principle.

Finally, we write $\mathsf{WPHP}(\mathsf{PV})$ to denote the set $\{\mathsf{WPHP}(f) \mid f \text{ is a } \mathsf{PV} \text{ function}\}$ of sentences. The sets $\mathsf{WPHP}'(\mathsf{PV})$, $\mathsf{dWPHP}(\mathsf{PV})$, and $\mathsf{WPHPWIT}(\mathsf{PV})$ are defined in an analogous way.

Formalizations. Our equivalences will involve pigeonhole principles (introduced above), communication complexity, time complexity of Turing machines, and bounds for error correcting codes. We now explain how to use the vocabulary of PV_1 to express some of these statements.

For a PV_1 function symbol $f(x, y)$, which we view as a two-party function, we let $\mathsf{LB}^f_{n,m}$ denote the formula stating that the two-party deterministic communication complexity of f on n-bit strings x and y is larger than m. In more detail, for $n, m \in \mathsf{Log}$, this is expressed as[3]

$$\mathsf{LB}^f_{n,m} \triangleq \forall \mathcal{P} \ (\mathsf{CC}(\mathcal{P}, 1^n, 1^m) \to \mathsf{Fail}_f(\mathcal{P}, 1^n)).$$

Here $\mathsf{CC}(\mathcal{P}, 1^n, 1^m)$ denotes that $\mathcal{P}$, viewed as a communication protocol over inputs of length n, always communicates at most m bits. On the other hand, $\mathsf{Fail}_f(\mathcal{P}, 1^n)$ expresses that there is a pair of n-bit inputs (x, y) such that the output of $\mathcal{P}$ on (x, y) is different from $f(x, y)$. In order to explicitly describe the sub-formulas $\mathsf{CC}(\mathcal{P}, 1^n, 1^m)$ and $\mathsf{Fail}_f(\mathcal{P}, 1^n)$, it is necessary to specify the communication transcript of $\mathcal{P}$ on a given input (x, y). The key point is that if $\mathcal{P}$ encodes the next-bit message functions and the final decision of the players in the communication protocol, then a polynomial-time function that is explicitly given $\mathcal{P}$ and the pair (x, y) can produce the transcript of the protocol on (x, y).[4] Consequently, it is not hard to formalize the sentence in PV_1.

[2]This principle is also known as the retraction pigeonhole principle in the literature (see, e.g., [Jeř07b]).

[3]Here, the intended meaning of the notation $n \in \mathsf{Log}$ is that n is the length of a number. Roughly speaking, this allows the statement to reason about numbers bounded by $2^{\mathsf{poly}(n)}$, which corresponds to $\mathsf{poly}(n)$-bit strings. This is standard notation in bounded arithmetic, and we refer to Section 2 for more details.

[4]We describe the different functions involved in the specification of a protocol using Boolean circuits. This does not restrict the protocols because the size of the circuits can be arbitrary. Since an explicitly described circuit can be evaluated on a given input in polynomial time, the formalization can be done with appropriate PV_1 function symbols.

Similarly, we will use $\underrightarrow{\mathsf{LB}}^{f}_{n,m}$ to denote that the one-way two-party deterministic communication complexity of f is larger than m.

Given an explicit single-tape Turing machine M and a constructive time-bound t, let $\mathsf{U}_M(1^t, x)$ be the PV function that simulates the Turing machine M on the input x for t steps and returns the output. For every language $L \in \mathsf{P}$, we define $\mathsf{LB}^{L}_{\text{1-tape}}(M, t)$ as the sentence:

$$\mathsf{LB}^{L}_{\text{1-tape}}(M, t) \triangleq \forall n \in \mathsf{Log}\ \exists x \in \{0,1\}^n\ \big(L(x) \neq \mathsf{U}_M(1^{c(M)\cdot t(n)}, x)\big),$$

where $c(M) \in (0,1)$ is a small constant that depends only on M. (We aim to formalize lower bounds of the form $\Omega(t)$.) (The specification above is informal, as we need to use a PV function symbol for L in the sentence $\mathsf{LB}^{L}_{\text{1-tape}}(M, t)$ to express $L(x)$.)

We can capture the Palindrome lower bound against time $\Omega(t)$ as a set $\mathsf{LB}^{\mathsf{PAL}}_{\text{1-tape}}(t)$ of sentences defined as

$$\mathsf{LB}^{\mathsf{PAL}}_{\text{1-tape}}(t) \triangleq \{\mathsf{LB}^{\mathsf{PAL}}_{\text{1-tape}}(M, t(n)) \mid M \text{ is a single-tape machine}\},$$

where PAL is the language of palindromes, i.e., $\mathsf{PAL} \triangleq \{w \in \{0,1\}^* \mid w = \mathsf{rev}(w)\}$, where $\mathsf{rev}(\cdot)$ is the reversion function $\mathsf{rev}(w_1 w_2 \dots w_k) \triangleq w_k w_{k-1} \dots w_1$.

Due to space constraints, we defer the discussion on error correcting codes to the body of the paper.

Note that some principles and statements are captured by a collection of sentences instead of a single sentence. For this reason, we introduce the following definition. Let T be a theory, and let Φ_1, Φ_2 be sets of formulas. Then Φ_2 is said to be a T-*consequence* of Φ_1, denoted by $\Phi_1 \vdash_{\mathsf{T}} \Phi_2$, if $\Phi_1, \mathsf{T} \vdash \varphi$ for every $\varphi \in \Phi_2$. We say Φ_1 and Φ_2 are equivalent with respect to T, denoted by $\Phi_1 \equiv_{\mathsf{T}} \Phi_2$, if $\Phi_1 \vdash_{\mathsf{T}} \Phi_2$ and $\Phi_2 \vdash_{\mathsf{T}} \Phi_1$.

Equivalences (Reversals). We are now ready to state our results. We start off with an equivalence class for the weak pigeonhole principle $\mathsf{WPHP}(\mathsf{PV})$.

Theorem 1.1 (Informal; see Theorem 3.11 for additional equivalences and the precise statement)**.** *Let $\varepsilon \in (0,1)$ be a constant. The following statements are equivalent with respect to* PV_1:

- *(i)* $\mathsf{WPHP}(\mathsf{PV})$*, i.e., the weak pigeonhole principle for* PV *functions.*
- *(ii)* $\forall n \in \mathsf{Log}\ \mathsf{LB}^{\mathsf{EQ}}_{n,n-1}$*, i.e.,* EQ_n *has communication complexity* $> n-1$.
- *(iii)* $\forall n \in \mathsf{Log}\ \underrightarrow{\mathsf{LB}}^{\mathsf{EQ}}_{n,n^{\varepsilon}}$*, i.e.,* EQ_n *has communication complexity* $> n^{\varepsilon}$ *against one-way protocols.*
- *(iv) The Singleton bound for error correcting codes.*

In other words, the weak pigeonhole principle is not only sufficient but also *necessary* to prove two-party deterministic communication complexity lower bounds for the equality function (which is one of the most well-known results in communication complexity theory, see, e.g. [AB09, Theorem 13.4]). This is also known as a *reversal* in the setting of reverse mathematics, i.e., when the axioms used to prove a theorem are also shown to be necessary to establish the theorem.

Our next result provides an equivalence class for the uniform variant of the weak pigeonhole principle, i.e., $\mathsf{WPHP}'(\mathsf{PV})$. In particular, we show $\mathsf{WPHP}'(\mathsf{PV})$ is equivalent to the classic $\Omega(n^2)$-time lower bound for Palindrome against one-tape Turing machines [Maa84].

Theorem 1.2 (Informal; see Theorem 4.9 for additional equivalences and the precise statement)**.** *Let $\beta \in (0,1)$ be any constant. The following statements are equivalent with respect to* PV_1:

(*i*) $\mathsf{WPHP}'(\mathsf{PV})$*, i.e., the uniform weak pigeonhole principle for* PV *functions.*

(*ii*) $\mathsf{LB}^{\mathsf{PAL}}_{\text{1-tape}}(n^2)$*, i.e., Palindrome requires* $\Omega(n^2)$ *time on single-tape Turing machines.*

(*iii*) $\mathsf{LB}^{\mathsf{PAL}}_{\text{1-tape}}(n^{1+\beta})$*, i.e., Palindrome requires* $\Omega(n^{1+\beta})$ *time on single-tape Turing machines.*

It is possible to extend Theorem 1.2 with communication complexity lower bounds that are analogous to those from Theorem 1.1, but involve *uniform* communication protocols. We omit them here due to space constraints, and refer to Theorem 4.9 for these additional equivalences.

Finally, in our next result we describe an equivalence class for the weak witnessing pigeonhole principle $\mathsf{WPHPWIT}(\mathsf{PV})$.

Theorem 1.3 (Informal; see Theorem 5.7 for additional equivalences and the precise statement)**.** *Let* $\varepsilon \in (0,1)$ *be any constant. The following statements are equivalent with respect to* PV_1:

(*i*) $\mathsf{WPHPWIT}(\mathsf{PV})$*, i.e., the weak witnessing pigeonhole principle for* PV *functions.*

(*ii*) $\forall n \in \mathsf{Log}\ \underrightarrow{\mathsf{LB}}^{\mathsf{SetDisj}}_{n,n-1}$*, i.e.,* $\mathsf{SetDisj}$ *has communication complexity* $> n-1$ *against one-way protocols.*

(*iii*) $\forall n \in \mathsf{Log}\ \underrightarrow{\mathsf{LB}}^{\mathsf{SetDisj}}_{n,n^{\varepsilon}}$*, i.e.,* $\mathsf{SetDisj}$ *has communication complexity* $> n^{\varepsilon}$ *against one-way protocols.*

(*iv*) *The Singleton bound for decodable error correcting codes.*

1.2.1 Consequences

Note that the main novelty of our results is that, as opposed to several previous papers that have formalized complexity lower bounds in different theories, here we establish *equivalences* between lower bound statements and different principles. This is precisely what allows us to derive new consequences on the unprovability of lower bounds that were not previously possible.

Unprovability of simple complexity lower bounds. Since there is evidence that several variants of the pigeonhole principle are not provable in PV_1, our results suggest that this theory is not sufficient to formalize some elementary complexity lower bound results. Moreover, the same argument can be extended to Jeřábek's theory APC_1 [Jeř07a], which is defined as $\mathsf{PV}_1 + \mathsf{dWPHP}(\mathsf{PV})$. We refer to Section 6.1 for the formal statements of the cryptographic assumptions employed in the next result.

Corollary 1.4 (Informal; see Section 6 for the precise statements)**.** *For every* $\varepsilon, \beta \in (0,1)$*, the following results hold*:

(*i*) *Under the existence of collision resistant hash functions* (CHRF)*,* APC_1 *does not prove the sentence* $\forall n \in \mathsf{Log}\ \underrightarrow{\mathsf{LB}}^{\mathsf{EQ}}_{n,n^{\varepsilon}}$*, i.e., that* EQ_n *has communication complexity* $> n^{\varepsilon}$ *against one-way protocols.*

(*ii*) *Under the existence of keyless collision resistant hash functions* (CHRF')*,* APC_1 *does not prove that Palindrome requires* $\Omega(n^{1+\beta})$ *time on single-tape Turing machines.*

Fine-grained understanding of the proof complexity of lower bounds. As a consequence of Theorem 1.1 and Theorem 1.3, the weak witnessing pigeonhole principle is equivalent to one-way communication lower bounds for Set Disjointness, while the weak pigeonhole principle is equivalent to one-way communication lower bounds for Equality. The fact that $\mathsf{WPHPWIT}(\mathsf{PV})$ can be derived from $\mathsf{WPHP}(\mathsf{PV})$ agrees with the status of Set Disjointness as a harder communication problem than Equality [BFS86], which means that it is easier to prove communication complexity lower bounds for Set Disjointness than for Equality. Interestingly, since $\mathsf{WPHPWIT}(\mathsf{PV})$ is available in APC_1 (it follows from $\mathsf{dWPHP}(\mathsf{PV})$), we get that APC_1 proves one-way communication lower bounds for Set Disjointness, while it is unlikely to prove one-way communication lower bounds for Equality (Corollary 1.4).

Amplification of provable lower bounds. Note that Theorem 1.1, Theorem 1.2, and Theorem 1.3 also highlight an interesting phenomenon concerning the (un)provability of lower bounds: a weak lower bound is provable in the base theory PV_1 if and only if a significantly stronger (and in our examples tight) lower bound is provable. Moreover, it follows from Theorem 1.1 that PV_1 proves one-way communication lower bounds for Equality if and only if it proves a lower bound against protocols with an unbounded number of rounds.

Derandomization of feasible arguments and completeness. Finally, we discuss an application of our results to derandomization. Recall that APC_1 is a theory associated with probabilistic feasible reasoning, while PV_1 is associated with (deterministic) feasible reasoning. A significant open problem in bounded arithmetic is whether probabilistic feasible reasoning can be "derandomized", i.e., whether $\mathsf{APC}_1 = \mathsf{PV}_1$ (see Section 6.2 for more details). A related question of particular relevance to complexity lower bounds is whether this equivalence holds with respect to certain classes of sentences. To give an example, note that many complexity lower bounds against deterministic non-uniform computational models (in particular, the lower bounds we considered in the equivalence classes of variants of pigeonhole principles) can be formalized as $\forall\Sigma^b_1$ sentences of the following format:

For every input length $n \in \mathsf{Log}$, $n > n_0$, for every (non-uniform) device A from the model, there is an input $x \in \{0,1\}^n$ such that $A(x) \neq f(x)$.

Consequently, under the hypothesis that APC_1 is $\forall\Sigma^b_1$-conservative over PV_1, we get that every such lower bound that can be proved using feasible probabilistic reasoning (i.e., APC_1 reasoning) can be proved using feasible deterministic reasoning (i.e., PV_1 reasoning). Similarly, we say that a search problem P (represented by an open PV_1-formula) admits a feasible deterministic (resp. randomized) polynomial-time algorithm if $\forall x\ \exists y\ P(x,y)$ is provable in PV_1 (resp. APC_1). The problem of derandomizing feasible definable randomized algorithms can be formalized as:

Is APC_1 $\forall\Sigma^b_1$-conservative over PV_1?

We refer the reader to [Kra24] for a more extensive discussion of this and related questions.

An important property of the witnessing pigeonhole principle is that $\mathsf{APC}_1 = \mathsf{PV}_1 + \mathsf{dWPHP}(\mathsf{PV})$ is $\forall\Sigma^b_1$-conservative over $\mathsf{PV}_1 + \mathsf{WPHPWIT}(\mathsf{PV})$ ([Jeř04, Jeř07a]; see Theorem 2.10), namely any $\forall\Sigma^b_1$-sentence provable in APC_1 is also provable in $\mathsf{PV}_1 + \mathsf{WPHPWIT}(\mathsf{PV})$. This, together with the equivalence between $\forall n \in \mathsf{Log}\ \underrightarrow{\mathsf{LB}}^{\mathsf{SetDisj}}_{n,n^\varepsilon}$ and $\mathsf{WPHPWIT}(\mathsf{PV})$ (Theorem 5.7), leads to the following consequence.

Corollary 1.5. *The following statements are equivalent.*

1. *APC_1 is $\forall\Sigma^b_1$-conservative over PV_1, i.e., feasibly definable randomized algorithms can be feasibly defined deterministically.*

2. $\mathsf{PV}_1 \vdash \mathsf{WPHPWIT}(\mathsf{PV})$, *namely* PV_1 *proves the weak witnessing pigeonhole principle for* PV *functions.*
3. $\mathsf{PV}_1 \vdash \forall n \in \mathsf{Log}\ \underrightarrow{\mathsf{LB}}^{\mathsf{SetDisj}}_{n,n^{\varepsilon}}$, *namely* PV_1 *proves a* $n^{\Omega(1)}$ *communication complexity lower bound for* $\mathsf{SetDisj}$ *against one-way protocols.*

An interpretation of this result is that the one-way communication complexity lower bound for Set Disjointness is an APC_1-*complete lower bound* with respect to PV_1, in the sense that it is provable in APC_1, and if it is provable in PV_1, then every lower bound (formalized as a $\forall\Sigma^b_1$ sentence) provable in APC_1 is also provable in PV_1.

We further elaborate on the aforementioned implications and discuss additional applications of our results in Section 6. In particular, Section 6.4 discusses immediate consequences for subclasses of TFNP and their reducibilities.

Relevant related work. The classical reverse mathematics program was started by Friedman [Fri75] and developed extensively by Simpson (see, e.g., [Sim09]) and other researchers. Similarly, the investigation of the weakest logical theory able to formalize results relevant to algorithms and complexity has been an active area of research for several decades, with S. Cook as one of its earliest and most prominent researchers. The textbook [CN10] provides a comprehensive exposition of the area (see, e.g., [BKKK20, TC21] for more recent papers). While [CN10] does not have a focus on the provability of complexity lower bounds nor on reversals, such as the ones presented here, it is an excellent resource for a reader interested in the broader context of the reserve mathematics of algorithms and complexity theory.

The result that is closest to our contributions is a certain equivalence between the dual weak pigeonhole principle $\mathsf{dWPHP}(\mathsf{PV})$ and circuit lower bounds from [Jeř04]. In more detail, [Jeř04] showed that $\mathsf{dWPHP}(\mathsf{PV})$ is equivalent (over S^1_2) to a statement asserting the existence of a family of Boolean functions with exponential circuit complexity. On the other hand, here we focus on different pigeonhole principles and employ the weaker base theory PV_1.[5] The technique introduced by [Jeř04] to show this reversal is a central tool in some recent papers in complexity theory [Kor21, CHR23, Li23].

Finally, there is a rich literature on pigeonhole principles in bounded arithmetic. We refer to [PWW88, Kra01, Tha02, Jeř07b, KT08, BKT14] and references therein for a partial list.

A complete diagram of implications and equivalence classes appears in Figure 1 on page 48.

1.3 Techniques

In order to explain some of the ideas present in the proofs of these results, we focus on some non-trivial implications within Theorem 1.1, Theorem 1.2, and Theorem 1.3.

$\mathsf{WPHP}(\mathsf{PV})$ and Communication Lower Bounds for EQ. First, we discuss the equivalence between Items (*i*) and (*ii*) in Theorem 1.1. Consider the direction $\mathsf{WPHP}(\mathsf{PV}) \vdash_{\mathsf{PV}_1} \forall n \in \mathsf{Log}\ \mathsf{LB}^{\mathsf{EQ}}_{n,n-1}$. To establish this communication complexity lower bound for the Equality function, we show that the classical *fooling set argument* (see, e.g., [KN97]) can be formalized in $\mathsf{PV}_1 + \mathsf{WPHP}(\mathsf{PV})$. To

[5]It is unclear if Jeřábek's equivalence can be proved over PV_1 instead of S^1_2.

achieve this, we consider a function f with extra parameters such that, as we fix one of its inputs to be the description of a candidate protocol $\mathcal{P}$ communicating at most $n-1$ bits on every input pair (x,y) with $x,y \in \{0,1\}^n$, $f(x,\mathcal{P})$ outputs the transcript of $\mathcal{P}$ over the input pair (x,x). By the weak pigeonhole principle, there are distinct x,y such that $f(x,\mathcal{P}) = f(y,\mathcal{P})$, i.e., the transcripts of $\mathcal{P}$ on (x,x) and (y,y) are identical. We now argue as in the standard proof via the fooling set method, i.e., by analyzing the behavior of $\mathcal{P}$ over the input pair (x,y) to reach a contradiction. The latter requires an inductive argument over the number of rounds of the protocol, which we verify to be feasible with the induction principle available in PV_1.

For the other direction, i.e., to show that $\forall n \in \mathsf{Log}\ \mathsf{LB}^{\mathsf{EQ}}_{n,n-1} \vdash_{\mathsf{PV}_1} \mathsf{WPHP}(\mathsf{PV})$, we argue its contrapositive (in PV_1) as follows. Let f be any PV function symbol, possibly with extra parameters. Suppose that there is a choice of the extra parameters such that f violates the weak pigeonhole principle, i.e., f is an injection from $[(1+1/n)\cdot 2^n]$ to $[2^n]$, for some n. By a standard argument, one can amplify the parameters and show the existence of an *injective* map from $[2^{n'}]$ to $[2^{n'-1}]$, i.e., from n'-bit strings to $(n'-1)$-bit strings, for some n'. We then exploit this map to design a non-trivial one-way communication protocol that allows Alice and Bob to compute Equality over n'-bit strings with communication cost $n'-1$. Finally, we argue that the correctness of this protocol can be proved in PV_1, which establishes that $\neg\forall n \in \mathsf{Log}\ \mathsf{LB}^{\mathsf{EQ}}_{n,n-1}$.

While the argument presented above describes the intuition behind the equivalence between the communication lower bound for Equality and the weak pigeonhole principle, we stress that it needs to be formalized with the reasoning available over the base theory PV_1. Part of the technical work behind this formalization involves setting up appropriate notions and definitions from communication protocols in PV_1. To the best of our knowledge, the formalization of communication complexity in bounded arithmetic has not been explored prior to this work.

To provide some intuition for the amplification phenomenon appearing in results such as Theorem 1.1, we note (as hinted above) that some formulations of the weak pigeonhole principle in PV_1 with different shrinkage are known to be equivalent. Consequently, once one establishes an equivalence between $\mathsf{WPHP}(\mathsf{PV})$ and complexity lower bounds, it is sometimes possible to follow the implications in the equivalence to amplify provable lower bounds. In the case of Theorem 1.1 Item (*iii*), as explained above, the shrinkage is closely related to the communication cost of the protocol.

$\mathsf{WPHP}'(\mathsf{PV})$ and Turing Machine Lower Bounds for Palindrome. The proof of Theorem 1.2 builds on some ideas and intuitions developed in the proof of Theorem 1.1. In our discussion of Theorem 1.2, we first consider the implication from Item (*iii*) to Item (*i*), i.e.,

$$\mathsf{LB}^{\mathsf{PAL}}_{\text{1-tape}}(n^{1+\beta}) \vdash_{\mathsf{PV}_1} \mathsf{WPHP}'(\mathsf{PV}).$$

The intuition is that if, towards a contradiction, $\mathsf{WPHP}'(\mathsf{PV})$ does not hold, we can obtain a PV function f that is a "perfect" hash function, which means that it compresses a string and has no collision. While the mild compression provided by f can be amplified using known techniques from bounded arithmetic, an important issue with any potential use of f in the design of an algorithm for Palindrome is that f is a polynomial-time function that might require much more than quadratic time on a single-tape Turing machine.

A key idea in our proof is to apply the Merkle-Damgård construction [Mer89, Dam89], a technique from cryptography which improves both the *shrinkage* and *running time* of collision-resistant hash functions. In our setting, we show that this construction allows us to obtain a perfect hash function with $n^{1+\gamma}$ time complexity and shrinkage n^{γ}, for a small $\gamma = \gamma(\beta)$. Intuitively, this makes it possible to decide Palindrome in $n^{1+\beta}$ time by hashing the left part and (the reverse of) the right part of the input string, then applying a straightforward algorithm on the hash values.

To formally prove the theorem, we need to check that the algorithm can be effectively implemented in a single-tape Turing machine, which is a rather limited device for computations restricted to sub-quadratic time. Moreover, the correctness of the construction needs to be proved in PV_1, namely with induction only for quantifier-free formulas. These points turn out to require a somewhat delicate analysis. To handle this, we present an explicit description of the different stages of the associated single-tape Turing machine. A careful construction followed by a detailed argument allow us to verify that the aforementioned points do not create issues in the proof.

For the other non-trivial implication, i.e., showing that $\mathsf{WPHP}'(\mathsf{PV}) \vdash_{\mathsf{PV}_1} \mathsf{LB}^{\mathsf{PAL}}_{\mathsf{1\text{-}tape}}(n^2)$, we rely on the close relationship between the weak pigeonhole principle and two-party (deterministic) communication lower bounds, as established in Theorem 1.1. In more detail, we formalize in PV_1 the lower bound for single-tape Turing machines obtained via communication complexity. While this is typically done using randomized communication lower bounds (see, e.g., [KN97, Lemma 12.7]) or non-uniform communication lower bounds, it is also possible to rely on a weaker communication lower bound, where the next-bit message functions of both parties are uniform deterministic polynomial-time algorithms, through a more careful argument. Crucially, the proof can be carried out in $\mathsf{PV}_1 + \mathsf{WPHP}'(\mathsf{PV}_1)$.

$\mathsf{WPHPWIT}(\mathsf{PV})$ and One-Way Communication Lower Bounds. We now explain the equivalence between Item (*i*) and Item (*ii*) in Theorem 1.3. The implication from Item (*ii*) to Item (*i*)

$$\forall n \in \mathsf{Log}\ \underset{\longrightarrow}{\mathsf{LB}}^{\mathsf{SetDisj}}_{n,n-1} \vdash_{\mathsf{PV}} \mathsf{WPHPWIT}(\mathsf{PV})$$

is similar to the "reversals" in Theorem 1.1 and Theorem 1.2. Towards a contradiction, we assume that Item (*i*) does not hold. Then there is a pair of PV functions $f\colon [(1+1/n)\cdot 2^n] \to [2^n]$ and $g\colon [2^n] \to [(1+1/n)\cdot 2^n]$ such that $g \circ f\colon [2^n] \to [2^n]$ is the identity function. Similarly to the proof of Theorem 1.1, we can further construct a pair of functions $f'\colon [2^{n'}] \to [2^{n'-1}]$ and $g'\colon [2^{n'-1}] \to [2^{n'}]$ such that $g' \circ f'$ is the identity function. In other words, (g', f') is a *lossless compression* of n'-bit strings to $(n'-1)$-bit strings. This enables the violation of the one-way communication lower bound, since Alice can send an $(n'-1)$-bit compression of her input that can be decompressed by Bob.

To establish the converse direction $\mathsf{WPHPWIT}(\mathsf{PV}) \vdash_{\mathsf{PV}} \forall n \in \mathsf{Log}\ \underset{\longrightarrow}{\mathsf{LB}}^{\mathsf{SetDisj}}_{n,n-1}$, we need to utilize the properties of Set Disjointness (as in known lower bound proofs) while making sure that the proof can be carried out in $\mathsf{PV}_1 + \mathsf{WPHPWIT}(\mathsf{PV})$. The outline of the argument is as follows. Assume that there is a one-way communication protocol for Set Disjointness with communication complexity at most $n-1$. We notice that the message sent by Alice only depends on her input $x \in \{0,1\}^n$. Therefore, Bob is able to compute $\mathsf{SetDisj}(x, y)$ for every $y \in \{0,1\}^n$ given the message from Alice. Consequently, Bob can compute the bit $x_j = 1 - \mathsf{SetDisj}(x, e^j)$ for every $j \in [n]$, where $e^j \in \{0,1\}^n$ is the indicator string with the j-th coordinate set to 1 and every other coordinate set to 0. This yields a pair of functions constituting a lossless compression of n-bit strings to $(n-1)$-bit strings, thus violates $\mathsf{WPHPWIT}(\mathsf{PV})$:

- $f\colon \{0,1\}^n \to \{0,1\}^{n-1}$, where $f(x)$ is the message sent by Alice given the input x;
- $g\colon \{0,1\}^{n-1} \to \{0,1\}^n$, where the j-th bit of $g(z)$ is the negation of the output of Bob in the communication protocol given the message z from Alice and the input $e^j \in \{0,1\}^n$.

1.4 Concluding remarks and open problems

In relation to the major research goal emphasized in Section 1.1, our results and previous work on the formalization of lower bounds indicate that a suitable logical theory T should contain at least PV_1 and the principles $\mathsf{WPHP}(\mathsf{PV})$ and $\mathsf{dWPHP}(\mathsf{PV})$. In particular, the intuition shared by some researchers that most complexity lower bounds can be established in APC_1 is probably misguided, in light of the existence of simple lower bounds that cannot be proved in this theory under a plausible cryptographic assumption.

We note that, by the main unprovability result of [LO23], $\mathsf{PV}_1+\mathsf{WPHP}(\mathsf{PV})+\mathsf{dWPHP}(\mathsf{PV})$ does not prove that Σ_3-$\mathsf{SIZE}[\mathsf{poly}]$ cannot be approximated by Π_3-$\mathsf{SIZE}[2^{n^\delta}]$ circuits. While this result is unconditional, it would be much more interesting to show the unprovability of lower bounds that are close to major open problems in complexity theory. Even if we consider a weaker theory, such as PV_1, showing the unprovability of $\mathsf{SAT} \notin \mathsf{SIZE}[\mathsf{poly}]$ would represent a significant progress in our understanding of which lower bounds can be feasibly proved.

In terms of open problems, in addition to encouraging the reader to explore equivalences between their favorite lower bound statements and different mathematical principles, we mention below a few directions that we find particularly intriguing or of technical interest.

Note that, as opposed to Theorem 1.1, in Theorem 1.3 we do not establish an equivalence between one-way and unbounded-round communication lower bounds. Is it possible to derive multi-round communication complexity lower bounds from $\mathsf{WPHPWIT}(\mathsf{PV})$?

Does PV_1 prove the nondeterministic time hierarchy theorem, i.e., that $\mathsf{NTIME}[n^c] \nsubseteq \mathsf{NTIME}[n^d]$ for $1 < d < c$? An immediate issue here is that the standard proof of this hierarchy theorem requires a lazy diagonalization argument over input lengths that are exponentially far apart. However, there are alternate arguments that do not need to consider exponentially large intervals of input lengths (see [FS16, FS17] and references therein). If PV_1 cannot establish this result, is there a natural equivalence class that captures this lower bound?

While it seems plausible that explicit lower bounds against Boolean formulas of size $n^{1.99}$ might be provable in PV_1, the situation with respect to formulas of size $n^{2.01}$ (or larger) appears more intricate. In particular, we speculate that near-cubic formula lower bounds for Andreev's function might be equivalent (over the base theory PV_1) to an appropriate variant of the weak witnessing pigeonhole principle $\mathsf{WPHPWIT}$. (Whether this principle and its variants can be established in PV_1 remains an interesting open problem.)

Acknowledgements. We would like to thank Jan Krajíček and Neil Thapen for discussions on pigeonhole principles, and Ryan Williams for discussions on the reverse mathematics of communication complexity. We are also grateful to Jan Krajíček and Zhenjian Lu for sharing several comments about a first draft of the paper. Igor C. Oliveira received support from the Royal Society University Research Fellowship URF\R1\191059; the UKRI Frontier Research Guarantee Grant EP/Y007999/1; and the Centre for Discrete Mathematics and its Applications (DIMAP) at the University of Warwick. Lijie Chen is supported by a Miller Research Fellowship. Jiatu Li is supported by an MIT Akamai Presidential Fellowship.

2 Preliminaries

2.1 Basic notation

Following standard set-theoretic notation, we will identify a number n with the set $\{0, 1, \ldots, n-1\}$. In particular, we will identify $\{0, 1\}$ with 2, as well as a string $x \in \{0, 1\}^n$ with a number $x < 2^n$.

We use $x \| y$ to denote the concatenation of two strings x and y. We use $\Delta(x)$ to denote the Hamming weight of x and $\delta(x)$ to denote the relative Hamming weight of x; similarly, we use $\Delta(x, y)$ and $\delta(x, y)$ to denote the Hamming distance and relative Hamming distance between x and y, respectively.

Let $\mathscr{C}$ be a complexity class. We define $\mathsf{i.o.}\text{-}\mathscr{C}$ to be the complexity class consisting of languages L such that for some $\hat{L} \in \mathscr{C}$, L agrees with $\hat{L}$ on infinitely many input lengths. We assume basic familiarity with standard complexity classes such that $\mathsf{NTIME}[T]$ and $\mathsf{DTIME}[T]$ (see, e.g., [AB09]).

2.2 Bounded arithmetic

Bounded arithmetic is a collective name of several fragments of Peano Arithmetic whose proof-theoretic capabilities are closely linked to complexity classes such as P, the polynomial-time hierarchy, or constant-depth circuits. In this paper, we will mostly work with the theory PV_1 introduced by Cook [Coo75] corresponding to polynomial-time computation. In this subsection, we present the intuition and sketch the formal definition of the theory PV_1 (see [Coo75, Kra95, CN10, Kra19] for the formal definition and more related discussion).

Polynomial-time constructive proofs. One of the main motivations to introduce the theory PV (short for *polynomially verifiable*) is to define a fragment of Peano Arithmetic that captures polynomial-time constructive proofs, in the sense that if $\forall x\, \varphi(x)$ is provable in the theory, the proof provides a uniform method to *verify* $\varphi(a)$ given a that only manipulates polynomial-time computable functions.

For a concrete example, the sentence

$$\forall n \; g(n) \le f(n) \quad \text{where } f(n) \triangleq 2n - 1,\; g(n) \triangleq \begin{cases} 1 & n = 1 \\ 2g(\lfloor n/2 \rfloor) + 1 & n > 1 \end{cases}$$

can be efficiently verified by a structural induction over the definition of g, which is a simple unwinding of the definition of a polynomial-time function. The correctness of the AKS primality test [AKS04]

$$\forall n \; (\mathsf{AKS}(u) = \text{“}\mathsf{prime}\text{”} \leftrightarrow \forall 1 < d < n \; d \nmid n),$$

on the other hand, is not likely to be verifiable in polynomial time. This is because when $\mathsf{AKS}(u) =$ “$\mathsf{composite}$”, there is no polynomial-time algorithm that provides a nontrivial divisor of n to justify the validity of the resulting formula, unless factoring is easy; indeed, one can check that the proof in [AKS04] utilizes functions that appear to be beyond the reach of polynomial-time functions. The upshot is that there are properties of polynomial time computations that might not be feasibly verifiable.

Jumping ahead, a suitable theory for polynomial-time reasoning should be able to *define* all polynomial-time computable functions (even if we cannot feasibly prove all their properties). It will consist of the defining axioms for these functions (encoding how they relate to each other), and support structural induction over the definition of polynomial-time functions. Moreover, the polynomial-time functions introduced in the theory must be *verified* to run in polynomial time. For this, we can pick a finite set of initial functions, which obviously run in polynomial time in the standard model and this fact is accepted through axioms, so that all other functions introduced in the theory must be verified to run in polynomial-time given the information available about the initial functions.

Equational Theory PV. Following the intuition sketched above, Cook [Coo75] defines PV as an equational theory (i.e. the only relation is equality and there are no quantifiers) via Cobham's recursive-theoretic characterization of polynomial-time functions [Cob65]. Cobham [Cob65] proved that the function class $\mathcal{F}$ defined as the minimal set of functions over the natural numbers closed under the following rules is exactly the class of all polynomial-time computable functions.

- (*Base functions*). The constant function $c(x) = 0$; functions $s_0(x) = 2x$ and $s_1(x) = 2x + 1$ for binary encoding of natural numbers; the projection functions $\pi_\ell^i(x_1, \dots, x_\ell) = x_i$; and the function $\#(x, y) = 2^{|x|\cdot|y|}$, where $|x|$ denotes the length of its binary encoding.

- (*Composition*). If h is an ℓ-ary function and $g_1, \dots, g_\ell$ are $\{n_1, n_2, \dots, n_\ell\}$-ary functions, then the composition of f and $g_1, \dots, g_\ell$ is an $(n_1 + n_2 + \cdots + n_\ell)$-ary function defined as

$$f(x_{11}, \dots, x_{1n_1}, \dots, x_{\ell 1}, \dots, x_{\ell n_\ell}) \triangleq h(g_1(x_{11}, \dots, x_{1n_1}), \dots, g_\ell(x_{\ell 1}, \dots, x_{\ell n_\ell})).$$

- (*Limited Recursion on Notation*). For functions $g(\vec{x}), h_0(\vec{x}, y, z), h_1(\vec{x}, y, z)$, and $k(\vec{x}, y)$, a function $f(\vec{x}, y)$ such that

$$\begin{aligned} f(\vec{x}, 0) &\triangleq g(\vec{x}), \\ f(\vec{x}, s_0(y)) &\triangleq h_0(\vec{x}, y, f(\vec{x}, y)), \\ f(\vec{x}, s_1(y)) &\triangleq h_1(\vec{x}, y, f(\vec{x}, y)), \\ f(\vec{x}, y) &\leq k(\vec{x}, y) \text{ for all } \vec{x} \text{ and } y, \end{aligned}$$

  is said to be constructed from g, h_0, h_1, k by limited recursion on (binary) notation.

Here we make a few comments on the definition of $\mathcal{F}$. Compositions involving $\#(x, y)$ can lead to functions with an arbitrary polynomial growth rate on the input length, as $|\#(x, y)| = |x| \cdot |y| + 1$. Also, one can show that the inequality $f(\vec{x}, y) \leq k(\vec{x}, y)$ in limited recursion of notation provides in a sense a verification that $f(\vec{x}, y)$ runs in polynomial time with a function $k(\vec{x}, y)$ that is already known to run in polynomial time. (Indeed, by dropping the inequality $f(\vec{x}, y) \leq k(\vec{x}, y)$ from the definition of limited recursion on notation one can get functions beyond polynomial time.)

The main technical issue to define PV is to properly account for the limited recursion on notation. Clearly, $f(\vec{x}, y) \leq k(\vec{x}, y)$ cannot be proved without defining f in the theory. However, we cannot define f before it is *verified* to run in polynomial time *within the theory*. Cook [Coo75] managed to slacken the inequality constraint to length constraints of the form $|h_i(\vec{x}, y, z)| \leq |z| + |k_i(\vec{x}, y)|$ over h_i for some existing functions k_i ($i \in \{0, 1\}$), which can be described by adding auxiliary functions like size comparison and string concatenation.

To introduce all (countably infinitely many) functions in the theory PV, Cook defines PV *functions* and PV *derivations* by a simultaneous recursion:

- Initial functions (including base functions of Cobham's class $\mathcal{F}$ and auxiliary functions for length comparison) and terms (i.e. composition of functions and open variables) obtained from initial functions are objects of *order* 0. Axioms that define the initial functions are PV-derivations of order 0.
- For every $i \in \mathbb{N}$, an order-$(i+1)$ PV-function is either a PV-function of order i, or a composition of PV-functions of order i, or is defined by *limited induction on notation* by order-i PV-functions, as long as the length inequality constraints $|h_i(\vec{x}, y, z)| \le |z| + |k_i(\vec{x}, y)|$ $(i \in \{0, 1\})$ admit order-i derivations.
- For every $i \in \mathbb{N}$, an order-$(i+1)$ PV-derivation is a sequence of equations involving objects of order-$(i+1)$, where the equations are either order-i PV-derivations or are obtained from deduction rules corresponding to symmetricity of equality, transitivity of equality, substitutions to equality, and the induction over notation. The induction over notation performs structural induction over the PV-functions defined by limited induction over notation, that is:
    - Let f be defined from (h_1, h_2, g, k_1, k_2) using limited induction over notation, and similarly let f' be defined from $(h'_1, h'_2, g', k'_1, k'_2)$. From $h_1 = h'_1$, $h_2 = h'_2$, $g = g'$, $k_1 = k'_1$, and $k_2 = k'_2$, one can deduce $f = f'$.

A PV-function is any order-i function symbol defined above, for an arbitrary $i \in \mathbb{N}$. The theory PV is axiomatized by the union of all equations appearing in order-i PV-derivations, over an arbitrary $i \in \mathbb{N}$.

First-Order Theory PV_1. Cook [Coo75] and a subsequent paper by Krajíček, Pudlák, and Takeuti [KPT91] defined a first-order theory PV_1 axiomatized by all provable equations in PV, as well as an induction axiom scheme postulating that over any quantifier-free formula $\varphi(\vec{x}, y)$:

$$\forall \vec{x}\ \forall b\ \big(\varphi(\vec{x}, 0) \land \neg\varphi(\vec{x}, b) \to \exists a < b\ (\varphi(\vec{x}, a) \land \neg\varphi(\vec{x}, a+1))\big). \tag{1}$$

(Here, $a + 1$ is defined by the PV-function that computes addition.) This sentence is logically equivalent to the standard (bounded) induction axiom:

$$\forall \vec{x}\ \forall b\ \big(\varphi(\vec{x}, 0) \land \forall a < b\ (\varphi(\vec{x}, a) \to \varphi(\vec{x}, a+1)) \to \varphi(\vec{x}, b)\big).$$

It can be proved that PV_1 is conservative over PV [Coo75] (i.e. every provable PV-equation in PV_1 is also provable in PV) and is axiomatizable by universal sentences [KPT91, Kra95].

Conceptually, the induction axiom is *polynomially verifiable* since the definition of the following function h explicitly justifies the existence of a in Equation (1):

- $h(\vec{x}, l, r)$ performs a binary search in the interval $[l, r]$ with the invariant that $\varphi(\vec{x}, l) \land \neg\varphi(\vec{x}, r)$. If $l = r + 1$, $h(\vec{x}, l, r)$ outputs l. If $l < r + 1$, let $m = \lfloor \frac{l+r}{2} \rfloor$ be the midpoint of $[l, r]$, the algorithm outputs $h(\vec{x}, l, m)$ if $\neg\varphi(\vec{x}, m)$ and outputs $h(\vec{x}, m, r)$ otherwise.

This function h can clearly be verified syntactically to run in polynomial time, as the interval is halved in the recursion and the truth-value of $\varphi(\vec{x}, y)$ can be checked in polynomial time given $(\vec{x}, y)$. Moreover, the definition of h clearly indicates that $h(\vec{x}, 0, b)$ will output an a that witnesses the existential quantifier in Equation (1) by a structural induction. Note that the same argument cannot generalize to induction for formulas with quantifiers, as the truth-value of the formula cannot be checked in *verifiable polynomial time* (even if a polynomial-time algorithm exists).

Formalizations in PV_1. In the formalization of lower bounds and upper bounds in PV_1, we will often use *bounded quantifiers*, denoted by $\forall y \leq t(\vec{x})$ or $\exists y \leq t(\vec{x})$ for some term t, which means that the quantifier runs over all numbers $y \leq t(\vec{x})$. Formally, they are defined as abbreviations:

$$\forall y \leq t(\vec{x})\ \varphi(\vec{x}, y) \triangleq \forall y\ (y \leq t(\vec{x}) \to \varphi(\vec{x}, y))$$
$$\exists y \leq t(\vec{x})\ \varphi(\vec{x}, y) \triangleq \exists y\ (y \leq t(\vec{x}) \wedge \varphi(\vec{x}, y)).$$

We will use standard abbreviations to formalize statements in PV_1. For a sentence $\varphi(n)$, the abbreviation $\forall n \in \mathsf{Log}\ \varphi(n)$ means that $\forall v\ \forall n\ (n = |v| \to \varphi(n))$, and $\exists n \in \mathsf{Log}\ \varphi(n)$ means that $\exists v\ \exists n\ (n = |v| \wedge \varphi(n))$. Here $|\cdot|$ denotes the length symbol, and its intended interpretation is that $|v|$ corresponds to the bitlength of the integer v. We often write $x \in \{0,1\}^n$ when formalizing complexity-theoretic statements in bounded arithmetic. Formally, this is an abbreviation for $x < 2^n$ by identifying $\{0,1\}^n$ and 2^n. In order to avoid excessive notation, when referring to quantities such as $a(1+1/c)$ intended to be positive integers in bounded theories, we omit the ceiling function and view it as the smallest integer larger than or equal to $a(1+1/c)$.

Results in this paper involves proofs inside the theory PV_1. Instead of a formal proof from axioms and rules of PV_1, we will write "semi-formal" proofs, typically starting with a sentence like "we argue in PV_1", that describe the idea of the proof and the main technical steps to formalize the idea in PV_1. Conceptually, we need to ensure that every function defined (explicitly or implicitly) during the proof must be verified to run in polynomial time, and all the deductions must be made in a way that is obvious according to the syntactical definition of some (polynomial-time computable) functions; in particular, one can only perform induction to quantifier-free formulas. Similarly, we may also only present an informal statement instead of giving a concrete formalization in $\mathcal{L}(\mathsf{PV})$ when the result is robust to any reasonable formalization.

To characterize the non-constructive parts of a statement, we may formalize an upper or lower bound by a set of sentences instead of a single sentence. For instance, $\mathsf{NP} \not\subseteq \text{i.o.-}\mathsf{P}_{/\mathsf{poly}}$ can be formalized as

$$\text{“}\mathsf{NP} \not\subseteq \text{i.o.-}\mathsf{P}_{/\mathsf{poly}}\text{”} \triangleq \{\text{“}\mathsf{SAT} \notin \text{i.o.-}\mathsf{SIZE}[cn^k]\text{”} \mid c, k \in \mathbb{N}\}, \tag{2}$$

where for some fixed function $n_0 : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, "$\mathsf{SAT} \notin \text{i.o.-}\mathsf{SIZE}[cn^k]$" states that for every $n \in \mathsf{Log}$, $n > n_0(c,k)$, and every circuit $C : \{0,1\}^n \to \{0,1\}$ of size cn^k, C does not compute SAT over input length n. We say $T \vdash$ "$\mathsf{NP} \not\subseteq \text{i.o.-}\mathsf{P}_{/\mathsf{poly}}$" if and only if there exists a function $n_0(\cdot,\cdot)$ such that T proves every sentence in the set in Equation (2). This is a standard approach in bounded arithmetic (see, e.g. [PS21, LO23]).

Toolkits in PV_1. We need the following known formalization of standard tools such as error-correcting codes in bounded theories. Note that we employ the same symbols (Δ and δ) to represent the PV function symbols corresponding to the computation of the Hamming weight and Hamming distance of a string.

Lemma 2.1 (Error-correcting codes, implicit in [Jeř05]). *There is a constant c, a PV-function E, and a PV-function D such that the following is provable in PV_1:*

- (Polynomial rate). $\forall n \in \mathsf{Log}\ \forall x \in \{0,1\}^n\ |E(x)| = cn^c$.
- (Decoding). $\forall n \in \mathsf{Log}\ \forall x \in \{0,1\}^n\ \forall e \in \{0,1\}^{cn^c}\ (\delta(e) \leq 0.1 \to D(E(x) \oplus e) = x)$, *where $\oplus$ denotes bit-wise XOR of two binary strings.*

Lemma 2.2 ($\mathsf{P} \subseteq \mathsf{P}_{/\mathsf{poly}}$, implicit in [Jeř07a, Pic15b])**.** *For every* PV *function* f*, there is a constant* c *such that* PV *proves the following: for every* $n \in \mathsf{Log}$*, there is a circuit* $C : \{0,1\}^n \to \{0,1\}$ *of size at most* cn^c *such that for every* $x \in \{0,1\}^n$*,* $\mathsf{Eval}(x, C) = f(x)$*, where* $\mathsf{Eval}(x, C)$ *is the* PV*-function that evaluates the circuit* C *over input* x*.*

2.3 Combinatorial principles

In this subsection, we define the combinatorial principles we will need and explore their basic proof complexity properties.

Pigeonhole Principles. We start by introducing three versions of pigeonhole principle: (weak) pigeonhole principle, dual (weak) pigeonhole principle, and witnessing pigeonhole principle.

Definition 2.3 ((Weak) pigeonhole principle)**.** Let $a > b > 0$ be numbers and $f : a \times \mathbb{N} \to b$ be a function, where the second input is a parameter. The *pigeonhole principle* of f, denoted by $\mathsf{PHP}^a_b(f)$, refers to the sentence stating that, for every z, $f(\cdot, z)$ is not injective, i.e.,

$$\mathsf{PHP}^a_b(f) \triangleq \forall z\ (\exists u < a\ f(u,z) > b) \vee (\exists x < a\ \exists y < a\ (x \neq y \wedge f(x,z) = f(y,z))).$$

The *weak pigeonhole principle* of f, denoted by $\mathsf{WPHP}(f)$, is defined as

$$\mathsf{WPHP}(f) \triangleq \forall b > 0\ \forall c \in \mathsf{Log}\ \mathsf{PHP}^{b(1+1/c)}_b(f).$$

(When referring to $\mathsf{PHP}^a_b(f)$ and $\mathsf{WPHP}(f)$ as sentences in a formal theory, we consider a and b as formal variables and f as a function symbol of arity two. The same will be the case in the next definitions.)

Definition 2.4 (Dual (weak) pigeonhole principle)**.** Let $b > a > 0$ be numbers and $f : a \times \mathbb{N} \to b$ be a function, where the second input is a parameter. The *dual pigeonhole principle* of f, denoted by $\mathsf{dPHP}^a_b(f)$, refers to the sentence stating that, for every z, $f(\cdot, z)$ is not surjective, i.e.,

$$\mathsf{dPHP}^a_b(f) \triangleq \forall z\ \exists y < b\ \forall x < a\ f(x,z) \neq y.$$

The *dual weak pigeonhole principle* of f, denoted by $\mathsf{dWPHP}(f)$, is defined as

$$\mathsf{dWPHP}(f) \triangleq \forall a > 0\ \forall c \in \mathsf{Log}\ \mathsf{dPHP}^a_{a(1+1/c)}(f).$$

The pigeonhole principle (resp. dual pigeonhole principle) states that an integer function cannot be injective (resp. surjective) if its co-domain is smaller (resp. larger) than its domain. We will also need a "composed" version of these two principles, called *witnessing pigeonhole principle*, which states that for any two functions $f : a \to b$ and $g : b \to a$ (possibly with parameters) such that $a < b$, $f \circ g$ is not the identity function.

Definition 2.5 (Witnessing (weak) pigeonhole principle)**.** Let $b > a > 0$ be numbers, $f : a \times \mathbb{N} \to b$ and $g : b \times \mathbb{N} \to a$ be functions, where the second input of each function is a parameter. The *witnessing pigeonhole principle* of the pair of functions (f, g), denoted by $\mathsf{PHPWIT}^b_a(f, g)$, refers to the sentence stating that for every z_1 and z_2, $f(g(\cdot, z_1), z_2)$ is not an identity function, i.e.,

$$\mathsf{PHPWIT}^b_a(f) \triangleq \forall z_1\ \forall z_2\ (\exists u < b\ g(u, z_1) > a) \vee (\exists y < b\ f(g(y, z_1), z_2) \neq y).$$

The *witnessing weak pigeonhole principle* of (f, g), denoted by $\mathsf{WPHPWIT}(f, g)$, is defined as

$$\mathsf{WPHPWIT}(f, g) \triangleq \forall a > 0\ \forall c \in \mathsf{Log}\ \mathsf{PHPWIT}^{a(1+1/c)}_a(f, g).$$

We use $\mathsf{WPHP}(\mathsf{PV})$ to denote the set $\{\mathsf{WPHP}(f) \mid f \text{ is a } \mathsf{PV} \text{ function}\}$ of formulas, and use $\mathsf{dWPHP}(\mathsf{PV})$ and $\mathsf{WPHPWIT}(\mathsf{PV})$ likewise.

Consequences. Similar to the notion of *reduction* in computational complexity, we use $\varphi \vdash_{\mathsf{T}} \psi$ to denote that ψ is a *consequence* of φ (or ψ *reduces* to φ) with respect to the theory T. In other words, it suffices to prove φ in order to prove ψ when reasoning in the theory T. The formal definition is given below.

Definition 2.6. Let T be a theory and φ_1, φ_2 be formulas. Then φ_2 is said to be a T-*consequence* of φ_1, denoted by $\varphi_1 \vdash_{\mathsf{T}} \varphi_2$, if $\varphi_1, \mathsf{T} \vdash \varphi_2$. We say φ_1 and φ_2 are T-*equivalent*, denoted by $\varphi_1 \equiv_{\mathsf{T}} \varphi_2$, if $\varphi_1 \vdash_{\mathsf{T}} \varphi_2$ and $\varphi_2 \vdash_{\mathsf{T}} \varphi_1$.

Moreover, we can also define the T-consequence relation between sets of formulas by viewing a set as the (infinite) conjunction of the formulas inside the set.

Definition 2.7. Let T be a theory and Φ_1, Φ_2 be sets of formulas. Then Φ_2 is said to be a T-*consequence* of Φ_1, denoted by $\Phi_1 \vdash_{\mathsf{T}} \Phi_2$, if $\Phi_1, \mathsf{T} \vdash \varphi$ for every $\varphi \in \Phi_2$. We say Φ_1 and Φ_2 are T-*equivalent*, denoted by $\Phi_1 \equiv_{\mathsf{T}} \Phi_2$, if $\Phi_1 \vdash_{\mathsf{T}} \Phi_2$ and $\Phi_2 \vdash_{\mathsf{T}} \Phi_1$.

Proof Complexity of Pigeonhole Principles. We present several known results about the proof complexity of the three variants of the pigeonhole principle introduced in Section 2.3.

The following two propositions show that the *witnessing pigeonhole principle* for functions f, g is a consequence of both the dual weak pigeonhole principle for f and the weak pigeonhole principle for g with respect to PV_1.

Proposition 2.8. *Let f, g be arbitrary functions. Then $\mathsf{PHPWIT}^b_a(f,g)$ is a PV_1-consequence of $\mathsf{dPHP}^a_b(f)$. Moreover, $\mathsf{WPHPWIT}(f,g)$ is a PV_1-consequence of $\mathsf{dWPHP}(f)$.*

Proof. We argue in PV_1. Suppose that, towards a contradiction, that $\mathsf{PHPWIT}^b_a(f,g)$ is false. In other words, there exist z_1, z_2 such that for every $y < b$, $f(g(y, z_1), z_2) = y$. We now prove that $\mathsf{dPHP}^a_b(f)$ is false. Let $z \triangleq z_2$. For every $y < b$, we can see that $x \triangleq g(y, z_1)$ satisfies that $f(x, z) = y$, which completes the proof. The "moreover" parts follow straightforwardly. □

Proposition 2.9. *Let f, g be arbitrary functions. Then $\mathsf{PHPWIT}^b_a(f,g)$ is a PV_1-consequence of $\mathsf{PHP}^b_a(g)$. Moreover, $\mathsf{WPHPWIT}(f,g)$ is a PV_1-consequence of $\mathsf{WPHP}(g)$.*

Proof. We argue in PV_1. Suppose that, towards a contradiction, that $\mathsf{PHPWIT}^b_a(f,g)$ is false. In other words, there exist z_1, z_2 such that for every $y < b$, $f(g(y, z_1), z_2) = y$. We now prove that $\mathsf{PHP}^b_a(g)$. Let $z \triangleq z_1$. For every $x_1, x_2 < a$, $x_1 \neq x_2$, we can see that

$$f(g(x_1, z_1), z_2) = x_1 \quad \text{and} \quad f(g(x_2, z_1), z_2) = x_2,$$

which implies that $g(x_1, z_1) \neq g(x_2, z_1)$. This means $\mathsf{PHP}^b_a(g)$ is false. □

Jeřábek [Jeř04, Jeř07a] shows that $\mathsf{PV}_1 + \mathsf{dWPHP}(\mathsf{PV})$ is a $\forall\Sigma^b_1$-conservative extension of $\mathsf{PV}_1 + \mathsf{WPHPWIT}(\mathsf{PV})$. That is, every $\forall\Sigma^b_1$ formula provable in $\mathsf{PV}_1 + \mathsf{dWPHP}(\mathsf{PV})$ is also provable in $\mathsf{PV}_1 + \mathsf{WPHPWIT}(\mathsf{PV})$.

Theorem 2.10 ([Jeř04, Jeř07a]). *$\mathsf{PV}_1 + \mathsf{dWPHP}(\mathsf{PV})$ is $\forall\Sigma^b_1$-conservative over $\mathsf{PV}_1 + \mathsf{WPHPWIT}(\mathsf{PV})$. In particular, if $\mathsf{PV}_1 \vdash \mathsf{WPHPWIT}(\mathsf{PV})$, then $\mathsf{PV}_1 + \mathsf{dWPHP}(\mathsf{PV})$ is $\forall\Sigma^b_1$-conservative over PV_1.*

Note that many almost-everywhere circuit lower bounds (e.g. $\oplus \notin \mathsf{i.o.\text{-}AC}^0[n^k]$) can be formalized by $\forall\Sigma_1^b$ sentences. This means that if the lower bound is provable in $\mathsf{PV}_1 + \mathsf{dWPHP}(\mathsf{PV})$, it can also be proved in $\mathsf{PV}_1 + \mathsf{WPHPWIT}(\mathsf{PV})$.

Uniform Variants of Pigeonhole Principles. In the definition of PHP, dPHP, and PHPWIT, the functions f and g are allowed to take an extra parameter $z \in \mathbb{N}$, which makes the principles work for functions with "non-uniform" advice. Similarly, we can also consider the *uniform* versions of the principles.

Definition 2.11. (Uniform (weak) pigeonhole principle) Let $a > b > 0$ be numbers and $f : a \to b$ be a function. The *uniform pigeonhole principle* of f, denoted by $\mathsf{PHP}'^a_b(f)$, refers to the sentence stating that f is not injective, i.e.,

$$\mathsf{PHP}'^a_b(f) \triangleq \exists x < a\ \exists y < a\ (x \neq y \land f(x) = f(z)).$$

The *uniform weak pigeonhole principle* of f, denoted by $\mathsf{WPHP}'(f)$, is defined as

$$\mathsf{WPHP}'(f) \triangleq \forall b > 0\ \forall c \in \mathsf{Log}\ \mathsf{PHP}'^{b(1+1/c)}_b(f).$$

We can also define dPHP', dWPHP', PHPWIT', and $\mathsf{WPHPWIT}'$ as above. Similarly to the non-uniform case, we can show that PHPWIT' is a consequence of both PHP' and dWPHP'.

Proposition 2.12. *Let f, g be arbitrary functions. Then $\mathsf{PHPWIT}'^b_a(f, g)$ is a PV_1-consequence of $\mathsf{dPHP}'^a_b(f)$. Moreover, $\mathsf{WPHPWIT}'(f, g)$ is a PV_1-consequence of $\mathsf{dWPHP}'(f)$.*

Proposition 2.13. *Let f, g be arbitrary functions. Then $\mathsf{PHPWIT}'^b_a(f, g)$ is a PV_1-consequence of $\mathsf{PHP}'^b_a(g)$. Moreover, $\mathsf{WPHPWIT}'(f, g)$ is a PV_1-consequence of $\mathsf{WPHP}'(g)$.*

3 Equivalence Class for WPHP(PV)

We first explore the equivalence class of weak pigeonhole principle for PV functions. In particular, we will show that $\mathsf{WPHP}(\mathsf{PV})$ is equivalent to deterministic communication lower bounds for EQ against non-uniform protocols with respect to PV_1.

3.1 Technical lemmas

Lemma 3.1 (Implicit in [PWW88, Tha02, Jeř07b])**.** *Let $t(\cdot)$ be any PV function such that $\mathsf{PV}_1 \vdash \forall x\ t(x) \geq 2x$. Then $\mathsf{WPHP}(\mathsf{PV}) \equiv_{\mathsf{PV}_1} \{\forall a\ \mathsf{PHP}^{t(a)}_a(f) \mid f$ is a PV function$\}$.*

3.2 CC lower bounds for EQ ⇔ WPHP(PV)

We start by showing that deterministic communication complexity lower bounds for Equality is equivalent to the weak pigeonhole principle.

Formalization of CC lower bounds. Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a Boolean function. Recall that a communication protocol for f describes a strategy to compute $f(x, y)$ through the interaction between two players called Alice and Bob, where Alice has access to x while Bob has access to y. Their goal is to minimize the communication complexity of the protocol, which is the maximum number of bits transmitted by the players over all input pairs (x, y). We refer to [KN97] for more details.

We describe next a natural way to formalize communication protocols in bounded theories. We will be concerned with protocols where the messages produced by Alice and Bob can be computed by Boolean circuits. Since we will not upper bound the size of the circuits, this assumption is without loss of generality, i.e., every communication protocol can be captured by our formalization.

Let $n \in \mathsf{Log}$, $t \in \mathsf{Log}$, and $x, y \in \{0,1\}^n$ be the inputs for Alice and Bob, respectively. A t-round communication protocol consists of $2t + 2$ Boolean circuits $(S_1, C_1), (S_2, C_2), \ldots, (S_t, C_t), (D, d)$, where $S_i : \{0,1\}^{i-1} \to \{0,1\}$ decides which player to speak in the i-th round given the transcript of the first $i-1$ rounds, and $C_i : \{0,1\}^n \times \{0,1\}^{i-1} \to \{0,1\}$ outputs a single-bit message of the player given the player's input (x for Alice and y for Bob) and the transcript of the first $i-1$ rounds. Let π be the full transcript after t rounds of communication. The circuit $D : \{0,1\}^t \to \{0,1\}$ decides the player to output the answer: If $D(\pi) = 0$, Alice outputs $d(x, \pi)$, and otherwise, Bob outputs $d(y, \pi)$. We say a communication protocol $\mathcal{P}$ computes $f(x, y)$ if $f(x, y) = [D(\pi) = 0] \cdot d(x, \pi) + [D(\pi) = 1] \cdot d(y, \pi)$. The communication complexity of the protocol is the number of bits exchanged by players, and in our formalization, it is simply the number of rounds t.

To formalize communication complexity lower bounds, we define the following functions and formulas in PV_1.

- $\pi(\mathcal{P}, 1^n, x, y, i)$ denotes the first i bits transmitted given the input pair $x, y \in \{0,1\}^n$, which can be defined recursively as

$$\begin{aligned} \pi(\mathcal{P}, 1^n, x, y, 0) &\triangleq \varepsilon \\ \pi(\mathcal{P}, 1^n, x, y, i) &\triangleq \pi(\mathcal{P}, 1^n, x, y, i-1) \| C_i(z, \pi(\mathcal{P}, 1^n, x, y, i-1)), \qquad (i \geq 1) \end{aligned}$$

  where $z = x$ if $S_i(\pi(\mathcal{P}, 1^n, x, y, i-1)) = 0$, and $z = y$ if $S_i(\pi(\mathcal{P}, 1^n, x, y, i-1)) = 1$. For simplicity, we use π_i to denote $\pi(\mathcal{P}, 1^n, x, y, i)$ if it is clear from the context, and use $\pi(x, y)$ or simply π to denote the entire transcript, i.e., $\pi(x, y) \triangleq \pi(\mathcal{P}, 1^n, x, y, t)$ if the protocol has t rounds.
- $p(\mathcal{P}, 1^n, x, y, i)$ denotes the player to speak in the i-th round given input $x, y \in \{0,1\}^n$, defined as $S_i(\pi(\mathcal{P}, 1^n, x, y, i-1))$. We use p_i to denote $p(\mathcal{P}, 1^n, x, y, i)$ if it is clear from the context.
- $\mathsf{Fail}_f(\mathcal{P}, 1^n)$ means the protocol $\mathcal{P}$ fails to compute f, that is,

$$\mathsf{Fail}_f(\mathcal{P}, 1^n) \triangleq \exists x, y \in \{0,1\}^n \ f(x, y) \neq [D(\pi) = 0] d(x, \pi) + [D(\pi) = 1] d(y, \pi).$$

- $\mathsf{CC}(\mathcal{P}, 1^n, 1^m)$ means that the communication complexity of the protocol $\mathcal{P}$ over input length n (for each player) is at most m.
- $\mathsf{LB}^f_{n,m}$ means that there is no communication protocol $\mathcal{P}$ with input length n and communication complexity m that computes f. That is, for $n, m \in \mathsf{Log}$,

$$\mathsf{LB}^f_{n,m} \triangleq \forall \mathcal{P} \ (\mathsf{CC}(\mathcal{P}, 1^n, 1^m) \to \mathsf{Fail}_f(\mathcal{P}, 1^n)).$$

One-way communication protocols. We are also interested in communication protocols with only *one* round, i.e., one of the player transmits a string and then the other decides. We assume that Alice transmits the message and Bob decides. More formally, a one-way communication protocol is defined as two circuits $(g_{\mathsf{A}}, d_{\mathsf{B}})$, where given the input $(x,y) \in \{0,1\}^n \times \{0,1\}^n$, Alice transmits $\pi = g_{\mathsf{A}}(x)$ to Bob and Bob outputs $d_{\mathsf{B}}(y,\pi)$. The protocol is said to compute $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ if $f(x,y) = d_{\mathsf{B}}(y, g_{\mathsf{A}}(x))$, and the communication complexity of the protocol is the output length of the circuit g_{A}.

Similarly to the discussion above, we can define the following formulas.

- $\underrightarrow{\mathsf{CC}}(\mathcal{P}, 1^n, 1^m)$ means that the communication complexity of the protocol $\mathcal{P}$ over input length n (for each player) is at most m, i.e., the output length of g_{A} is at most m.
- $\underrightarrow{\mathsf{Fail}}_f(\mathcal{P}, 1^n)$ means the protocol $\mathcal{P}$ fails to compute f, that is,
$$\underrightarrow{\mathsf{Fail}}_f(\mathcal{P}, 1^n) \triangleq \exists x, y \in \{0,1\}^n \ d_{\mathsf{B}}(y, g_{\mathsf{A}}(x)) \neq f(x,y).$$
- $\underrightarrow{\mathsf{LB}}^f_{n,m}$ means that there is no one-way communication protocol $\mathcal{P}$ with input length n and communication complexity m that computes f. That is, for $n, m \in \mathsf{Log}$,
$$\underrightarrow{\mathsf{LB}}^f_{n,m} \triangleq \forall \mathcal{P} \ (\underrightarrow{\mathsf{CC}}(\mathcal{P}, 1^n, 1^m) \to \underrightarrow{\mathsf{Fail}}_f(\mathcal{P}, 1^n)).$$

EQ lower bounds and WPHP. It is well known that the communication complexity of $\mathsf{EQ}(x,y) \triangleq [x = y]$ is at least n. This lower bound can be formulated in PV_1 by the formula $\mathsf{LB}^{\mathsf{EQ}}_{n,n-1}$, where for $n, m \in \mathsf{Log}$,
$$\mathsf{LB}^{\mathsf{EQ}}_{n,m} \triangleq \forall \mathcal{P} \ (\mathsf{CC}(\mathcal{P}, 1^n, 1^m) \to \mathsf{Fail}_{\mathsf{EQ}}(\mathcal{P}, 1^n)).$$

Theorem 3.2 ($\mathsf{WPHP}(\mathsf{PV}) \Rightarrow \mathsf{LB}^{\mathsf{EQ}}$). *For $n, m \in \mathsf{Log}$, $m < n$, $\mathsf{PHP}^{2^n}_{2^m}(\mathsf{PV}) \vdash_{\mathsf{PV}_1} \mathsf{LB}^{\mathsf{EQ}}_{n,m}$.*

Proof. This follows the standard *fooling set* proof of the communication complexity lower bound (see, e.g., [AB09]). We argue in PV_1. Fix $n \in \mathsf{Log}$ and a protocol $\mathcal{P}$. We need to show that $\mathsf{CC}(\mathcal{P}, 1^n, 1^m)$ implies $\mathsf{Fail}_{\mathsf{EQ}}(\mathcal{P}, 1^n)$.

Suppose that $\mathsf{CC}(\mathcal{P}, 1^n, 1^m)$ is true, we define the circuit $E : \{0,1\}^n \to \{0,1\}^m$ as $E(x) \triangleq \pi(x,x)$, i.e., $E(x)$ outputs the transcript of the protocol after running on the input (x,x). By $\mathsf{PHP}^{2^n}_{2^m}(\mathsf{PV})$, we can see that there exist distinct $x, y \in \{0,1\}^n$ such that $E(x) = E(y)$. (Concretely, we consider the PV function $\mathsf{Eval} : \{0,1\}^n \to \{0,1\}^*$ such that $\mathsf{Eval}(x, C)$ evaluates $C(x)$, and plugin the circuit E.) In other words, $\pi(x,x) = \pi(y,y)$, that is, the transcript of the communication protocol on the input (x,x) is the same as that on the input (y,y).

Claim 3.3 (in PV_1). *If $\pi(x,x) = \pi(y,y)$ then $\pi(x,y) = \pi(x,x) = \pi(y,y)$.*

Let $\pi \triangleq \pi(x,x) = \pi(y,y) = \pi(x,y)$. Given Claim 3.3, we can see that the output of $\mathcal{P}$ on (x,y) is equal to the output of $\mathcal{P}$ on either (x,x) or (y,y): If $D(\pi) = 0$, the output of the protocol is $d(x,\pi)$, which is the same as the output of the protocol on the input (x,x); otherwise, the output of the protocol is $d(\pi, y)$, which is the same as the output of the protocol on the input (y,y). However, as $\mathsf{EQ}(x,y) = 0$ but $\mathsf{EQ}(x,x) = \mathsf{EQ}(y,y) = 1$, the protocol $\mathcal{P}$ cannot be correct, and thus $\mathsf{Fail}_{\mathsf{EQ}}(\mathcal{P}, 1^n)$ follows.

It remains to prove Claim 3.3. Indeed, we will prove that
$$\pi(\mathcal{P}, 1^n, x, y, i) = \pi(\mathcal{P}, 1^n, x, x, i) = \pi(\mathcal{P}, 1^n, y, y, i)$$

for every $i \leq m$. We prove this equation by an induction on i, which utilizes the induction principle for quantifier-free formulas and therefore is admissible in PV_1. For $i = 0$, the equation trivially holds, since the transcript is an empty string.

Now we assume that the equation holds for $i - 1$. That is, the transcripts of the protocol on inputs (x, x), (y, y), and (x, y) in the first $i - 1$ rounds are the same. To prove the equation for i, we only need to show that the bit exchanged in the last round is the same on inputs (x, x), (y, y), and (x, y). Let π_{i-1} be the transcript of the previous rounds.

- If $S_i(\pi_{i-1}) = 0$, i.e., it is Alice to speak. Alice sends $C_i(x, \pi_{i-1})$ on inputs (x, y) and (x, x). Since the full transcripts on inputs (x, x) and (y, y) are the same, we know that on all of these three inputs, the communication transcripts are $\pi_i = \pi_{i-1} \| C_i(x, \pi_{i-1})$.
- Otherwise, it is Bob to speak. Bob sends $C_i(y, \pi_{i-1})$ on inputs (x, y) and (y, y). Since the full transcripts on inputs (x, x) and (y, y) are the same, we know that on all of these three inputs, the communication transcripts are $\pi_i = \pi_{i-1} \| C_i(y, \pi_{i-1})$.

This proves Claim 3.3. $\square$

Theorem 3.4 ($\mathsf{LB}^f \Rightarrow \underrightarrow{\mathsf{LB}}^f$)**.** *For $n, m \in \mathsf{Log}$, $m < n$, and PV function f, $\mathsf{LB}^f_{n,m} \vdash_{\mathsf{PV}_1} \underrightarrow{\mathsf{LB}}^f_{n,m}$.*

Proof. We argue in PV. Suppose, towards a contradiction, that $\underrightarrow{\mathsf{LB}}^f_{n,m}$ is false, then there are circuits $g_{\mathsf{A}} : \{0,1\}^n \to \{0,1\}^m$ and $d_{\mathsf{B}} : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ such that $d_{\mathsf{B}}(y, g_{\mathsf{A}}(x)) = f(x, y)$ for every $x, y \in \{0,1\}^n$. Consider the following protocol $\mathcal{P}$ with m rounds:

- For every $i \in [m]$, $S_i(\cdot) \triangleq 0$, i.e., it is always Alice to speak.
- $D(\cdot) \triangleq 1$, i.e., it is always Bob to decide.
- For every $i \in [m]$, $C_i(x, \cdot)$ outputs the i-th bit of $g_{\mathsf{A}}(x)$.
- $d(y, \pi) \triangleq d_{\mathsf{B}}(y, \pi)$.

It is clear that this protocol solves f on input length n with communication complexity m, which contradicts $\mathsf{LB}^f_{n,m}$. $\square$

Theorem 3.5. *For $n, m \in \mathsf{Log}$, $m < n$, $\underrightarrow{\mathsf{LB}}^{\mathsf{EQ}}_{n,m} \vdash_{\mathsf{PV}_1} \mathsf{PHP}^{2^n}_{2^m}(\mathsf{PV})$.*

Proof. We argue in PV_1. Towards a contradiction, we assume that there is a PV function $h : \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^m$, an $n \in \mathsf{Log}$, and a parameter z such that for all $x, y \in \{0,1\}^n$, $h(x, z) \neq h(y, z)$. Consider the following one-way communication protocol $\mathcal{P}$ for EQ. Let $(x, y) \in \{0,1\}^n \times \{0,1\}^n$ be the input, where Alice holds x and Bob holds y. Alice sends $h(x, z) \in \{0,1\}^m$ to Bob, and Bob accepts if and only if $h(x, z) = h(y, z)$.

Note that to formalize this protocol in PV_1, we need to construct the circuit $C(x) \triangleq h(x, z)$ such that $\forall x\ C(x) = h(x, z)$ is provable, which is possible with Lemma 2.2.

We now show that $\neg(\underline{\mathsf{CC}}(\mathcal{P}, n, m) \to \underline{\mathsf{Fail}}_{\mathsf{EQ}}(\mathcal{P}, n))$ for the protocol $\mathcal{P}$. It is clear that the communication complexity of the protocol is at most m, which means that $\underline{\mathsf{CC}}(\mathcal{P}, n, m)$. It remains to show that the communication protocol is correct, i.e., $\neg\underline{\mathsf{Fail}}_{\mathsf{EQ}}(\mathcal{P})$. That is, for every $x, y \in \{0,1\}^n$,

$$d_{\mathsf{B}}(y, g_{\mathsf{A}}(x)) = \mathsf{EQ}(x, y),$$

which is equivalent to

$$h(x, z) = h(y, z) \leftrightarrow x = y.$$

This holds thanks to the guarantee on h with parameter z, since for all $x, y \in \{0,1\}^n$ with $x \neq y$, $h(x, z) \neq h(y, z)$. $\square$

The three theorems above shows that one-way communication complexity lower bounds for EQ, communication complexity lower bounds for EQ, and weak pigeonhole principle are all equivalent.

Corollary 3.6. *For every $n, m \in \mathsf{Log}$, $m < n$, $\mathsf{LB}^{\mathsf{EQ}}_{n,m} \equiv_{\mathsf{PV}_1} \underrightarrow{\mathsf{LB}}^{\mathsf{EQ}}_{n,m} \equiv_{\mathsf{PV}_1} \mathsf{PHP}^{2^n}_{2^m}(\mathsf{PV})$.*

Fooling Set Method. Indeed, one can observe that the proof of Theorem 3.2 not only works for communication lower bound for EQ, but also works for the fooling set method (see, e.g., [AB09]) in general under a suitable formalization.

Let $n \in \mathsf{Log}$ and $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a Boolean function. A fooling set for f is a subset $S \subseteq \{0,1\}^n \times \{0,1\}^n$ of possible input pairs such that for all distinct input pairs $(x_1, y_1), (x_2, y_2) \in S$, it is not the case that $f(x_1, y_1) = f(x_1, y_2) = f(x_2, y_1) = f(x_2, y_2)$. The fooling set method shows that if f admits a fooling set of size at least 2^m, then the communication complexity of f is at least m. In the proof of Theorem 3.2, we utilized the fact that $S^{\mathsf{EQ}} \triangleq \{(x, x) \mid x \in \{0,1\}^n\}$ is a fooling set for EQ of size 2^n.

To formalize the definition in PV_1, we can define a set S using a Boolean circuit $C : \{0,1\}^n \times \{0,1\}^n$ that decides the membership relation, i.e., $(x, y) \in S$ if and only if $C(x, y) = 1$. (For S^{EQ}, the circuit simply checks whether the two input strings are identical.) The size inequality $|S| \geq 2^k$ can be formalized by a circuit $F : [2^k] \to S$ that is provably injective. Therefore, the fooling set method can be formalized by the following formula:

- $\mathsf{FSet}_{n,m} \triangleq \forall$ circuit $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, circuit $C : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, and circuit $F : [2^{m+1}] \to \{0,1\}^n \times \{0,1\}^n$, if F is an injective map to $S \triangleq C^{-1}(1)$ and $C^{-1}(1)$ is a fooling set, i.e.,
  1. $\forall i \in [2^{m+1}], C(F(i)) = 1$ and
  2. $\forall i, j \in [2^{m+1}], i \neq j \to F(i) \neq F(j)$,
  3. $\forall (x_1, y_1) \neq (x_2, y_2), C(x_1, y_1) = 1 \wedge C(x_2, y_2) = 1$ implies $\neg(f(x_1, y_1) = f(x_1, y_2) = f(x_2, y_1) = f(x_2, y_2))$

  then it follows that $\mathsf{LB}^{f}_{n,m}$.

Proposition 3.7. *For every $n, m \in \mathsf{Log}$ such that $m < n$, $\mathsf{FSet}_{n,m} \vdash_{\mathsf{PV}_1} \mathsf{LB}^{\mathsf{EQ}}_{n,m}$.*

Proof Sketch. Fix $f \triangleq \mathsf{EQ}$, $C(x, y) \triangleq [x = y]$, and $F : [2^{m+1}] \to \{0,1\}^n \times \{0,1\}^n$ such that $F(i)$ outputs (x_i, x_i), where x_i is the lexicographic i-th string of length n. One can easily prove the three requirements in the definition of FSet. Therefore, it follows that $\mathsf{LB}^{\mathsf{EQ}}_{n,m}$. □

Theorem 3.8 ($\mathsf{WPHP}(\mathsf{PV}) \Rightarrow \mathsf{FSet}$)**.** *For every $n, m \in \mathsf{Log}$, $m < n$, $\mathsf{PHP}^{2^{m+1}}_{2^m}(\mathsf{PV}) \vdash_{\mathsf{PV}_1} \mathsf{FSet}_{n,m}$.*

Proof Sketch. Fix $n, m \in \mathsf{Log}$. Assume that there are circuits f, C, F that satisfies the requirements in the definition of FSet. For any protocol $\mathcal{P}$, we need to show that $\mathsf{CC}(\mathcal{P}, 1^n, 1^m)$ implies $\mathsf{Fail}_f(\mathcal{P}, 1^n)$.

Suppose that $\mathsf{CC}(\mathcal{P}, 1^n, 1^m)$ is true. Let $\pi : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ be the circuit that computes the transcript corresponding to an input pair. Furthermore, we define $D : [2^{m+1}] \to \{0,1\}^m$ as $D(i) \triangleq \pi(F(i))$. By $\mathsf{PHP}^{2^{m+1}}_{2^m}(\mathsf{PV})$, there are $i \neq j$ such that $D(i) = D(j)$. Let $(x_i, y_i) \triangleq F(i)$ and $(x_j, y_j) \triangleq F(j)$. Since F is an injective map to $S \triangleq C^{-1}(1)$, (x_i, y_i) and (x_j, y_j)

are distinct input pairs in the fooling set S such that $\pi(x_i, y_i) = \pi(x_j, y_j)$. Therefore, by the definition of the fooling set, we get that

$$\neg(f(x_i, y_i) = f(x_i, y_j) = f(x_j, y_i) = f(x_j, y_j)).$$

However, since the protocol has the same transcript over input pairs (x_i, y_i) and (x_j, y_j), one can prove by induction over the number of rounds that $f(x_i, y_i) = f(x_i, y_j) = f(x_j, y_i) = f(x_j, y_j)$ following the proof of Claim 3.3, which leads to a contradiction. Since the induction hypothesis is captured by a quantifier-free formula, the proof can be formulated in PV_1. $\square$

Since the communication lower bound against EQ is equivalent to the weak pigeonhole principle, the fooling set method is in the same equivalent class.

Corollary 3.9. $\mathsf{WPHP}(\mathsf{PV}) \equiv_{\mathsf{PV}_1} \forall n, m \in \mathsf{Log},\ m < n \to \mathsf{FSet}_{n,m}$.

3.3 Singleton bound for ECC $\Leftrightarrow$ $\mathsf{WPHP}(\mathsf{PV})$

Except for the communication complexity lower bounds, we will show that the Singleton bound for error-correcting codes is equivalent to $\mathsf{WPHP}(\mathsf{PV})$.

The formalization of error-correcting codes in bounded arithmetic is straightforward. Let $E : \{0,1\}^n \to \{0,1\}^m$ be a Boolean circuit. It is said to be an $[n, d, m]$*-code* if for every $x, y \in \{0,1\}^n$, $x \neq y$ implies that $\Delta(E(x), E(y)) \geq d$. Let $n, m, d \in \mathsf{Log}$ and E be a Boolean circuit, we can formalize the definition in PV_1 with:

$$\mathsf{DistECC}(E, n, d, m) \triangleq \forall x, y \in \{0,1\}^n\ (x = y \lor \Delta(E(x), E(y)) \geq d).$$

The *Singleton bound* states that for every $[n, d, m]$-code, $m \geq d + n - 1$. This can be formalized as the following $\forall\Sigma_1^b$-sentence

$$\mathsf{Singleton} \triangleq \forall n, m, d \in \mathsf{Log}\ \forall E : \{0,1\}^n \to \{0,1\}^m\ \mathsf{DistECC}(E, n, d, m) \to m \geq d + n - 1.$$

Theorem 3.10. $\mathsf{WPHP}(\mathsf{PV}) \equiv_{\mathsf{PV}_1} \mathsf{Singleton}$.

Proof. ($\mathsf{WPHP}(\mathsf{PV}) \vdash_{\mathsf{PV}_1} \mathsf{Singleton}$). This follows from the standard proof of the Singleton bound. We argue in $\mathsf{PV}_1 + \mathsf{WPHP}(\mathsf{PV})$. Suppose, towards a contradiction, that there is a circuit $E : \{0,1\}^n \to \{0,1\}^m$ such that $\mathsf{DistECC}(E, n, d, m)$ is true and $m < n + d - 1$. Let $E' : \{0,1\}^n \to \{0,1\}^{m-d+1}$ be the circuit that outputs the first $m-d+1$ bits of E, and $f : \{0,1\}^n \times \mathbb{N} \to \{0,1\}^{m-d+1}$ be the function defined as $f(x, C) \triangleq C(x)$. As $m < n + d - 1$, we know by $\mathsf{WPHP}(f)$ that there must be two distinct strings $x, y \in \{0,1\}^n$ such that $E'(x) = E'(y)$. It then follows that $E(x)$ and $E(y)$ agree on the first $m-d+1$ bits and thus $\Delta(E(x), E(y)) \leq d-1$, which leads to a contradiction to $\mathsf{DistECC}(E, n, d, m)$.

($\mathsf{Singleton} \vdash_{\mathsf{PV}_1} \mathsf{WPHP}(\mathsf{PV})$). Note that by Lemma 3.1, it suffices to prove $\forall a\ \mathsf{PHP}_a^{t(a)}(\mathsf{PV})$ for an arbitrary function $t(a) \geq 2a$. Let c be the constant in Lemma 2.1, and E, D be the PV-functions. Fix $t(a) = 2^{\log^{2c} a}$. For every PV function f and any number a, we need to show that $\mathsf{PV} + \mathsf{Singleton} \vdash \mathsf{PHP}_a^{t(a)}(f)$. (For simplicity, we assume that $a = 2^n$, but it is straightforward to verify that the proof still works for an arbitrary a.)

Since $\mathsf{PHP}_a^{t(a)}(\mathsf{PV})$ is provable for every fixed $a \in \mathbb{N}$, we can assume that n is sufficiently large such that $n^{2c} \geq cn^c$. Suppose, towards a contradiction, that $\mathsf{PHP}_a^{t(a)}(f)$ is not true, which means that there exists a z such that for any distinct $x, y \in \{0,1\}^{n^{2c}}$, $f(x,z) \neq f(y,z)$, where $f(\cdot, z) : \{0,1\}^{n^{2c}} \to \{0,1\}^n$. By Lemma 2.2, we can obtain a circuit $C : \{0,1\}^{n^{2c}} \to \{0,1\}^n$ that computes $f(\cdot, z)$. Consider the circuit $E' : \{0,1\}^{n^{2c}} \to \{0,1\}^{cn^c}$ defined as $E'(x) \triangleq E(C(x))$. We now prove that E' is an $[n^{2c}, 0.2cn^c, cn^c]$ code.

- Let $x, y \in \{0,1\}^{n^{2c}}$ be distinct strings. As the circuit C violates pigeonhole principle, we know that $C(x) \neq C(y)$, which further means that $\Delta(E(C(x)), E(C(y))) \geq 0.2cn^c$ as E is an error-correcting code that corrects a 0.1 fraction of errors.

This clearly violates the Singleton bound as $n^{2c} \geq cn^c$. □

3.4 Summary of the equivalence class

We summarize the equivalence class for $\mathsf{WPHP}(\mathsf{PV})$ in the following theorem.

Theorem 3.11. *Let $\varepsilon \in (0,1)$ be any constant. The following sentences or sets of sentences are equivalent with respect to PV_1.*

1. $\mathsf{WPHP}(\mathsf{PV})$*, i.e., the weak pigeonhole principle for* PV *function symbols.*
2. $\{\forall n \in \mathsf{Log}\ \mathsf{PHP}^{2^n}_{2^n-1}(g) \mid g$ *is a* PV *function*$\}$*. This stands for the weak pigeonhole principle for* PV *functions with single-bit shrinkage.*
3. $\{\forall n \in \mathsf{Log}\ \mathsf{PHP}^{2^n}_{2^{n^\varepsilon}}(g) \mid g$ *is a* PV *function*$\}$*. This stands for the weak pigeonhole principle for* PV *functions with polynomial length shrinkage.*
4. $\forall n \in \mathsf{Log}\ \mathsf{LB}^{\mathsf{EQ}}_{n,n-1}$*, i.e.,* EQ *has communication complexity greater than* $n-1$*.*
5. $\forall n \in \mathsf{Log}\ \mathsf{LB}^{\mathsf{EQ}}_{n,n^\varepsilon}$*, i.e.,* EQ *has communication complexity greater than* n^ε*.*
6. $\forall n \in \mathsf{Log}\ \underrightarrow{\mathsf{LB}}^{\mathsf{EQ}}_{n,n-1}$*, i.e.,* EQ *has communication complexity greater than* $n-1$ *against one-way protocols.*
7. $\forall n \in \mathsf{Log}\ \underrightarrow{\mathsf{LB}}^{\mathsf{EQ}}_{n,n^\varepsilon}$*, i.e.,* EQ *has communication complexity greater than* n^ε *against one-way protocols.*
8. $\forall n, m \in \mathsf{Log}, m < n \to \mathsf{FSet}_{n,m}$*, i.e., the fooling set method.*
9. $\mathsf{Singleton}$*, i.e., the Singleton bound for error correcting codes.*

Proof. (1) $\equiv_{\mathsf{PV}_1}$ (2) $\equiv_{\mathsf{PV}_1}$ (3): This is an easy corollary of Lemma 3.1.

(2) $\vdash_{\mathsf{PV}_1}$ (4): This follows from Theorem 3.2.

(4) $\vdash_{\mathsf{PV}_1}$ (5): Trivial, since quantitatively stronger lower bounds imply weaker lower bounds under any reasonable formalization.

(5) $\vdash_{\mathsf{PV}_1}$ (7) and (4) $\vdash_{\mathsf{PV}_1}$ (6) follow from Theorem 3.4.

(7) $\vdash_{\mathsf{PV}_1}$ (3) and (6) $\vdash_{\mathsf{PV}_1}$ (2) follow from Theorem 3.5.

(1) $\equiv_{\mathsf{PV}_1}$ (8): This is given by Corollary 3.9.

(1) $\equiv_{\mathsf{PV}_1}$ (9): This is given by Theorem 3.10. □

4 Equivalence Class for $\mathsf{WPHP}'(\mathsf{PV})$

In this section, we will show that the $\Omega(n^2)$ time lower bounds for Palindrome against single-tape Turing machines [KN97, Section 12.2] and uniform communication complexity lower bounds for EQ are all equivalent to weak uniform pigeonhole principle.

4.1 Formalization and technical lemmas

We first demonstrate the formalization of lower bounds that we will consider, and present several technical lemmas used in the proofs.

Formalization of single-tape Turing machine lower bounds. Let $Q \triangleq \{0, 1, \ldots, q-1\}$ be the set of states, $\Sigma \triangleq \{\perp, 0, 1\}$ be the alphabet set, and $A \triangleq \{\rightarrow, \leftarrow\}$. A single tape Turing machine M can be defined by a set of states Q, a set of accepting states $S \subseteq Q$, and a transition function $\delta : Q \times \Sigma \to \Sigma \times A \times Q$, which means that if the symbol under the head is σ and the current state is u, $\delta(u, \sigma)$ specifies the new cell symbol $\sigma' \in \Sigma$, the movement $a \in A$, and the state $v \in Q$ afterwards. The head is set to the first character of the input at the beginning of the computation, and the output is defined as the string on the tape (excluding all "$\perp$"s) at the end. The tape of the Turing machine is infinite on both sides. We define the output of the Turing machine as the string in the tape once the machine reaches some accepting state.

Let $\mathsf{U}_M(1^t, x)$ be the PV function that simulates the Turing machine M on the input x for t steps and returns the output. For every language $L \in \mathsf{P}$, every function $c : \mathbb{N} \to \mathbb{Q} \cap [0, 1]$, and every constructive time bound t, we define $\mathsf{LB}^{L}_{\text{1-tape}}(M, t)$ as the $\forall\Sigma^b_1$-sentence:

$$\mathsf{LB}^{L,c}_{\text{1-tape}}(M, t) \triangleq \forall n \in \mathsf{Log}\ \exists x \in \{0,1\}^n\ \left(L(x) \neq \mathsf{U}_M(x, 1^{c(M)\cdot t(n)})\right).$$

We use the function $c(M)$ because we aim to formalize an $\Omega(t(n))$ lower bound whose asymptotic constant might depend on the description length of the machine M. When $c(M) = 1$, we will omit c and simply write $\mathsf{LB}^{L}_{\text{1-tape}}(M, t)$.

We can define the Palindrome lower bound against time $\Omega(t)$ as the set $\mathsf{LB}^{\mathsf{PAL},c}_{\text{1-tape}}(t)$ of sentences defined as

$$\mathsf{LB}^{\mathsf{PAL},c}_{\text{1-tape}}(t) \triangleq \{\mathsf{LB}^{\mathsf{PAL},c}_{\text{1-tape}}(M, t(n)) \mid M \text{ is a single-tape machine}\},$$

where PAL is the language of palindromes, i.e., $\mathsf{PAL} \triangleq \{w \| \mathsf{rev}(w) \mid w \in \{0,1\}^*\}$, where $\mathsf{rev}(\cdot)$ is the reversion function $\mathsf{rev}(w_1 w_2 \ldots w_k) \triangleq w_k w_{k-1} \ldots w_1$.

Technical lemmas. We will need the following technical lemmas before proving the equivalence between the uniform weak pigeonhole principle and the lower bound for PAL.

Lemma 4.1 (Implicit in [PWW88, Jeř07b])**.** *Let $t(\cdot)$ be any PV function such that $\mathsf{PV} \vdash \forall x\ t(x) \geq 2x$. Then $\mathsf{WPHP}'(\mathsf{PV}) \equiv_{\mathsf{PV}_1} \{\forall a\ \mathsf{PHP}'^{t(a)}_a(f) \mid f$ is a PV function$\}$.*

Lemma 4.2. *For every PV-function f, there is a single-tape Turing machine M_f and a constant d such that PV_1 proves that $\forall n \in \mathsf{Log}, \forall \vec{x} \in \{0,1\}^n, \mathsf{U}_{M_f}(\vec{x}, dn^d) = f(\vec{x})$. Moreover, such a Turing machine still exists (potentially with a larger constant d) even if we only have a single tape that is infinite in only one direction.*

Proof Sketch. We prove this by performing a structural induction over all PV function symbols. It is easy to verify that PV_1 proves that every initial function of PV can be computable by a polynomial-time Turing machine with a single tape that is infinite in one direction. Therefore, it remains to consider the functions introduced by composition and limited induction on notation.

Let f be a PV-function introduced by the composition of $g(z_1, \ldots, z_\ell)$ and $h_1(\vec{x}_1), \ldots, h_\ell(\vec{x}_\ell)$. By the induction hypothesis, we know that $g(\vec{z})$ and each $h_i(\vec{x}_i)$, $i \in [\ell]$ can be computed by a polynomial-time Turing machine with only one tape that is infinite in one direction. We assume that (x, y) is encoded as $[x, \perp, y]$ on the tape, where x and y are encoded in binary. Consider the following single-tape Turing machine M_f for f:

- Let $[\vec{x}_1, \perp, \vec{x}_2, \perp, \dots, \perp, \vec{x}_\ell]$ be the input. We assume that the tape is infinite on the left side.
- For every $i \in [\ell]$, the algorithm works as follows. Assume that before the i-th iteration, the tape configuration is $[\vec{x}_i, \perp, \dots, \perp, \vec{x}_\ell, \perp, h_1(\vec{x}_1), \perp, \dots, \perp, h_{i-1}(\vec{x}_{i-1})]$. It computes $h_i(\vec{x}_i)$ using the Turing machine H_i (with an infinite tape on one direction) for h_i on the tape to the left of $\vec{x}_i$ and leads to the tape configuration $[h(\vec{x}_i), \perp, \dots, \perp, \vec{x}_\ell, \perp, h_1(\vec{x}_1), \perp, \dots, \perp, h_{i-1}(\vec{x}_{i-1})]$. After that, we move $h(\vec{x}_i)$ to the rightmost block so that the tape configuration becomes $[\vec{x}_{i+1}, \perp, \dots, \perp, \vec{x}_\ell, \perp, h_1(\vec{x}_1), \perp, \dots, \perp, h_i(\vec{x}_i)]$.
- Now we have $[h_1(\vec{x}_1), \perp, \dots, \perp, h_\ell(\vec{x}_\ell)]$. The algorithm can then run the Turing machine for g to compute $g(h_1(\vec{x}_1), \dots, h_\ell(\vec{x}_\ell)) = f(\vec{x})$.

It is easy to verify that the Turing machine runs in polynomial time and that PV_1 proves the correctness of the Turing machine above using the corresponding function symbol U_{M_f}.

Let $f(\vec{x}, y)$ be introduced by limited recursion on notation from $g(\vec{x}), h_0(\vec{x}, y, z), h_1(\vec{x}, y, z)$, $k_0(\vec{x}, y)$, and $k_1(\vec{x}, y)$ with the following defining axioms

$$f(\vec{x}, 0) = g(\vec{x}); \quad f(\vec{x}, 2y) = h_0(\vec{x}, y, f(\vec{x}, y)); \quad f(\vec{x}, 2y+1) = h_1(\vec{x}, y, f(\vec{x}, y)),$$

and PV-derivations (of smaller order) such that

$$|h_i(\vec{x}, y, z)| \le |z| + |k_i(\vec{x}, y)|$$

for $i \in \{0, 1\}$, where the inequalities are encoded by PV-equations. By the induction hypothesis, we know that g, h_0, h_1, k_0, k_1 can be computed by polynomial-time Turing machines with a single tape that is infinite on one direction. Consider the following single-tape Turing machine M_f that computes f:

- Let $[\vec{x}, \perp, y]$ be the input. We assume that the tape is infinite on the left side.
- M_f first copies $\vec{x}$ to the left, which leads to the tape configuration $[\vec{x}, \perp, \vec{x}, \perp, y]$. Then, it runs the single-tape Turing machine for g to compute $g(\vec{x})$. The tape configuration after this step will be $[f(\vec{x}, 0), \perp, \vec{x}, \perp, y]$.
- Let y_i be the i-th most significant bit of y, $\ell(y, i)$ be the i most significant bits of y, that is, $\ell(y, 0) \triangleq 0$ and $\ell(y, i+1) \triangleq 2\ell(y, i) + y_{i+1}$. For every $i \in [|y|]$, the algorithm works as follows. Assume that the tape configuration before this step is $[f(\vec{x}, \ell(y, i-1)), \perp, \vec{x}, \perp, \vec{y}]$. The algorithm copy $\vec{x}$ and $\ell(y, i-1)$ to the left, which yields the tape configuration

  $$[\vec{x}, \perp, \ell(y, i-1), \perp, f(\vec{x}, \ell(y, i-1)), \perp, \vec{x}, \perp, \vec{y}],$$

  and also remembers y_i in its internal state. The algorithm then runs the Turing machine for h_{y_i} to compute $h_{y_i}(\vec{x}, \ell(y, i-1), f(\vec{x}, \ell(y, i-1))) = f(\vec{x}, \ell(y, i))$. Note that the provable length inequality $|h_i(\vec{x}, y, z)| \le |z| + |k_i(\vec{x}, y)|$ ensures that the running time in the i-th step is $O(n^c)$ for every $i \le |y|$, where c is some universal constant independent of i.

- In the end, $[f(\vec{x}, y), \perp, \vec{x}, \perp, y]$ is on the tape and M_f simply removes $\vec{x}$ and y and terminates.

It is easy to check by induction over quantifier-free formulas that M_f runs in polynomial time and M_f correctly computes f. Moreover, PV_1 proves the correctness of the Turing machine above using the corresponding function symbol U_{M_f}. □

The reason we need the simulation of PV functions with Turing machines with tapes infinite on only one side is that it makes it easier to invoke a Turing machine as a sub-routine. Concretely, as we will need to design single-tape Turing machines in sub-quadratic time, the simulation makes it possible to run the invoked sub-routine near the head while the head is moving back and forth.

4.2 Single-tape lower bounds $\Rightarrow$ $\mathsf{WPHP}'(\mathsf{PV})$

We first show that the uniform weak pigeonhole principle for PV functions is a consequence of an $n^{1+\Omega(1)}$ lower bound against single-tape Turing machines for Palindrome.

Theorem 4.3. $\mathsf{LB}^{\mathsf{PAL}}_{\text{1-tape}}(\beta n^{1+\beta}) \vdash_{\mathsf{PV}_1} \mathsf{WPHP}'(\mathsf{PV})$ *for every constant* $\beta > 0$*.*

The intuition of the theorem is that if, towards a contradiction, $\mathsf{WPHP}'(\mathsf{PV})$ does not hold, we can obtain a PV function that is a "perfect" hash function, which means that it compresses a string with no collision. By applying the Merkle-Damgård construction, we can further obtain a perfect hash function with $n^{1+\Omega(1)}$ time complexity and better shrinkage. This makes it possible to decide Palindrome in $n^{1+\Omega(1)}$ time by hashing the left and right halves of the string and applying a straightforward algorithm on the hash value.

To formally prove the theorem, we need to check that the algorithm can be effectively implemented in single-tape Turing machines, and the correctness can be proved in PV_1, namely with induction only for quantifier-free formulas.

Proof of Theorem 4.3. Let f be an arbitrary PV-function. We will show that $\mathsf{PV}_1 + \mathsf{LB}^{\mathsf{PAL}}_{\text{1-tape}}$ proves $\mathsf{WPHP}'(f)$. By Lemma 4.1, it suffices to show that for every PV-function g, $\mathsf{PV}_1 + \mathsf{LB}^{\mathsf{PAL}}_{\text{1-tape}}$ proves $\forall a > 0\ \mathsf{PHP}'^{2a}_{a}(g)$.

We now start to argue in PV_1. Suppose, towards a contradiction, that there is an a such that $\mathsf{PHP}'^{2a}_{a}(g)$ is false, i.e., for every $x, y < 2a$, we have $g(x) < a$, $g(y) < a$, and $g(x) \neq g(y)$. We assume for simplicity that a is a power of two, and discuss the other cases at the end of the proof. In such case, g is an injective function from n bits to $n-1$ bits. Let M, d be the single-tape Turing machine and the constant in Lemma 4.2 such that $\mathsf{U}_M(x, dn^d) = g(x)$ for every x. Note that we can assume a is larger than any fixed constant, since the uniform pigeonhole principle is true over the standard numbers and provable in PV_1 when the parameters are of constant size.

We will now describe a Turing machine M' with a tape that is infinite on both sides that aims to solve PAL using M. Let x be the input.

Stage 1: Compute the length of the input. The goal of the first step is to compute the length of the input and write it down on both sides of the input. Concretely, the tape configuration after the step will be $[n, \perp, x, \perp, n]$, where $n = |x|$ is encoded in binary.

The algorithm is to scan the entire string from left to right and maintain a binary counter near the head. Let $i = 0, 1, \ldots, n$ be an index. At the end of the i-th iteration, the tape configuration will be $[x', \perp, i, \underline{\perp}, x'']$, where $|x'| = i$, $|x''| = n - i$, and the head is at the underlined $\underline{\perp}$. (We will use $\underline{\sigma}$ to denote the position of the head throughout this subsection.) In the $(i+1)$-th iteration, the algorithm works as follows.

1. Probe the first bit b of x'' and move it to the right of x'. After this step, the tape configuration should be $[x', \underline{b}, \perp, i, \perp, x''']$, where $x'' = b\|x'''$. The time complexity is $O(\log n)$.
2. Increase the counter i by 1. If there is no overflow, the time complexity is $O(\log n)$. Otherwise, we move x', b to the left by one block to make room for the carried bit. Since the overall time for this will be $O(1+2+4+\cdots+n) = O(n)$, the amortized time complexity will be $O(1)$.

At the end of the algorithm, the tape configuration will be $[x, \perp, n]$. We can then perform the algorithm reversely to obtain the tape configuration $[n, \perp, x, \perp, n]$. The overall time complexity for this step is $O(n \log n)$.

Stage 2: Split the string. We will then split the input string x from the middle. The tape configuration after the step will be $[n, \perp, x^1, \perp, x^2, \perp, n]$, where x^1 and x^2 are the first and last $\lfloor n/2 \rfloor$ bits of x, respectively.

We first use the trick of maintaining a counter near the head, as in step 1, to locate the middle of the string and obtain the tape configuration $[n, \perp, x^1, \perp, n, \perp, \lfloor n/2 \rfloor, \underline{\perp}, x^2, \perp, n]$. Concretely, we will maintain in the i-th step the tape configuration $[n, \perp, x', \perp, n, \perp, i, \underline{\perp}, x'', \perp, n]$, where $|x'| = i$, $|x''| = n - i$, and $x'\|x'' = x$. We stop when $i = \lfloor n/2 \rfloor$, which can be checked in-place as $\lfloor n/2 \rfloor$ is equal to n right shifted by 1.

After that, we move x^2 and n to the left bit by bit to obtain the tape configuration $[n, \perp, x^1, \perp, x^2, \perp, n]$. Since $|n| + |i| = O(\log n)$, this will take $O(n \log n)$ time. The overall time complexity in this step will be $O(n \log n)$.

Stage 3: Hashing. Now it remains to decide whether x^1 is equal to the reverse of x^2. Let $n' \triangleq \lfloor n/2 \rfloor$, $\varepsilon \in (0,1)$ be a constant to be determined later, and $m \triangleq n^\varepsilon$. We define the following sequences of hash functions:

- $H_0 : \{0,1\}^m \to \{0,1\}^{m-1}$. $H_0(x) \triangleq g(x)$.
- $H_1 : \{0,1\}^{m+1} \to \{0,1\}^{m-1}$. $H_1(x\|b) \triangleq g(H_0(x)\|b)$.
- $H_2 : \{0,1\}^{m+2} \to \{0,1\}^{m-1}$. $H_2(x\|b) \triangleq g(H_1(x)\|b)$.
- ...
- $H_{n'-m} : \{0,1\}^{n'} \to \{0,1\}^{m-1}$. $H_{n'-m}(x\|b) \triangleq g(H_{n'-m-1}(x)\|b)$.

Let $H \triangleq H_{n'-m}$. The goal of the step is to compute $H(x^1)$ and $H(\mathsf{rev}(x^2))$, where rev denotes the reversion of strings. After this step, the tape configuration should be $[H(x^1), \perp, H(\mathsf{rev}(x^2))]$.

Consider the following algorithm. We first construct the tape configuration $[n, \perp, x^1_{-1}, \perp, x^1_{-1*}, \perp, x^2, \perp, n]$, where x^1_{-1} consists of the first $m-1$ bits of x^1, and x^1_{-1*} consists of the remaining bits. Let $i = 0, 1, \ldots, n'-m$ be an index. At the end of the i-th round, the tape configuration will be $[n, \perp, H_i(x^1_i), \perp, x^1_{i*}, \perp, x^2, \perp, n]$, where x^1_i consists of the first $m+i$ bits of x^1, and x^1_{i*} consists of the remaining $n'-m-i$ bits. For convenience, let H_{-1} denote the identity function. In the i-th step, the algorithm works as follows.

1. At the beginning, the tape configuration is $[n, \perp, H_{i-1}(x^1_{i-1}), \perp, x^1_{(i-1)*}, \perp, x^2, \perp, n]$.

2. Probe the first bit b of $x^1_{(i-1)*}$, and copy $H_{i-1}(x^1_{i-1})$ and b to the left of the string. The tape configuration afterwards will be $[H_{i-1}(x^1_{i-1}), b, \perp, n, \perp, H_{i-1}(x^1_{i-1}), \perp, x^1_{(i-1)*}, \perp, x^2, \perp, n]$.

3. We evaluate $g(H_{i-1}(x^1_{i-1})\|b)$ with the single-tape Turing machine M on a one-way infinite tape, so that it does not affect the right part of the tape. We will end up with the tape configuration $[H_i(x^1_i), \perp, n, \perp, H_{i-1}(x^1_{i-1}), \perp, x^1_{(i-1)*}, \perp, x^2, \perp, n]$.

4. We move $H_i(x^1_i)$ to the right to obtain $[n, \perp, H_i(x^1_i), \perp, x^1_{i*}, \perp, x^2, \perp, n]$

After $n' - m + 1$ steps, we will obtain $[n, \perp, H(x^1), \perp, x^2, \perp, n]$. We can then perform the algorithm reversely to obtain $[n, \perp, H(x^1), \perp, H(\mathsf{rev}(x^2)), \perp, n]$. We can then remove the input length n on both sides in linear time.

The complexity bottleneck is the third step of the algorithm. Since M runs in dn^d time on input length n, the evaluation of g will cost $dn^{d\varepsilon}$ time in each step, therefore the overall time complexity will be bounded by $10n \cdot dn^{d\varepsilon}$. We set $\varepsilon \triangleq \beta/(20d)$ so that the time complexity is at most $n^{1+\beta/10}$ for sufficiently large input lengths.

Step 4: Brute-force. We accepts if and only if $H(x^1) = H(\mathsf{rev}(x^2))$. This can be computed in $O(m^2) = o(n)$ time using a straightforward algorithm on a single-tape Turing machine.

The time complexity of the algorithm M' is at most $\beta n^{1+\beta}$ for $n > n_0$, where $n_0 \in \mathbb{N}$ is some fixed (standard) constant. Let $n \in \mathsf{Log}$ be the smallest number such that $2^{n^\varepsilon} = \alpha$. Recall that we can assume that α is larger than any fixed constants, therefore, we can also assume that $n > n_0$. By $\mathsf{LB}^{\mathsf{PAL}}_{\mathsf{1\text{-}tape}}(\beta n^{1+\beta})$, we know that there exists a string x of length n such that $M'(x) \neq \mathsf{PAL}(x)$. It remains to show that this is impossible.

We skip the analysis of the first two steps and the last step as it is straightforward. Assume that the tape configuration is $[n, \perp, x^1, \perp, x^2, \perp, n]$. Let $\hat{H}(x, i)$ be the PV-function such that $\hat{H}(x, i) = H_i(x)$, recursively defined as

$$\hat{H}(x,i) = \begin{cases} g(x) & i = 0 \\ g(\hat{H}(x', i-1)\|b) & i > 0, x = x'\|b' \\ 0 & \text{otherwise} \end{cases}$$

We can prove by induction (over quantifier-free formulas) that the Turing machine correctly computes $H(x^1)$ and $H(x^2)$, that is, the tape configuration after the computation is $[\hat{H}(x^1, m-n'), \perp, \hat{H}(\mathsf{rev}(x^2), n'-m)]$. For the input length n that we are considering, $m = n^\varepsilon = \log \alpha$. It remains to prove that $\hat{H}$ is a *perfect hash function*, that is for every $\ell \in \mathsf{Log}$ and every $x, y \in \{0,1\}^\ell$, $\hat{H}(x, \ell - m) = \hat{H}(y, \ell - m)$ implies $x = y$.

Fix any $\ell \in \mathsf{Log}$ and $x, y \in \{0,1\}^\ell$. Let x^i and y^i be the first $m + i$ bits of x and y, respectively. We prove by induction on i that if $i \leq \ell - m$, $\hat{H}(x^i, i) = \hat{H}(y^i, i) \to x^i = y^i$. (Note that this is an induction of quantifier-free formula, which can be implemented in PV_1.)

- **(Base Case).** When $i = 0$, the function $\hat{H}(\cdot, i) : \{0,1\}^{m+i} \to \{0,1\}^{m-1}$ is defined as $\hat{H}(x, 0) = g(x)$. Since g violates pigeonhole principle on input length m, we know that $g(x^0) = g(y^0)$ implies $x^0 = y^0$.

- **(Induction).** Assume that $i \leq \ell - m$ and $\hat{H}(x^{i-1}, i-1) = \hat{H}(y^{i-1}, i-1) \to x^{i-1} = y^{i-1}$ is true. Let $x^i = x^{i-1} \| b$ and $y^i = y^{i-1} \| c$. Suppose that $\hat{H}(x^i, i) = \hat{H}(y^i, i)$. We need to show that $x^i = y^i$. By the definition of $\hat{H}$, we know that

$$\hat{H}(x^i, i) = g(\hat{H}(x^{i-1}, i-1) \| b),$$
$$\hat{H}(y^i, i) = g(\hat{H}(y^{i-1}, i-1) \| c).$$

  Since g violates the pigeonhole principle on input length m and $\hat{H}(x^i, i) = \hat{H}(y^i, i)$, we know that $\hat{H}(x^{i-1}, i-1) \| b = \hat{H}(y^{i-1}, i-1) \| c$, which means that $\hat{H}(x^{i-1}, i-1) = \hat{H}(y^{i-1}, i-1)$ and $b = c$. By induction hypothesis, we further know that $x^{i-1} = y^{i-1}$, which, together with $b = c$, implies that $x^i = y^i$.

Finally, we consider the case when α is not a power of two. One can verify that the proof above still works if we replace the "hash function" $H_i : \{0,1\}^{m+i} \to \{0,1\}^{m-1}$ with $H_i : a \times \{0,1\} \to a$, where $m = \lfloor \log a \rfloor$, defined inductively as

$$H_0(x) = g(x)$$
$$H_{i+1}(x, b) = g(H_i(x), b)$$

by identifying each $u \in \{0,1\}^m$ as a number in $2^m \subseteq a$. $\square$

4.3 $\mathsf{WPHP}'(\mathsf{PV}) \Rightarrow$ uniform CC lower bounds

Similarly to the equivalence between $\mathsf{WPHP}(\mathsf{PV})$ and communication complexity lower bounds for EQ, we will show that $\mathsf{WPHP}'(\mathsf{PV})$ is sufficient to prove uniform communication complexity lower bounds for EQ.

Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ be a Boolean function. A *uniform communication protocol for f* with t rounds is described by four PV functions $\mathcal{P} = (S, C, D, d)$, where

- $S(\pi_{i-1}, 1^n, i) \in \{0,1\}$ chooses the player to speak in the i-th round given the transcript π of the first $i-1$ rounds;
- $C(\pi_{i-1}, z, 1^n, i)$ outputs the message to exchange in the i-th round given previous messages π_{i-1} and the input z of the player to speak, as chosen by $S(\pi_{i-1}, 1^n, i)$;
- $D(\pi, 1^n)$ chooses the player to output the answer given the full transcript π after t rounds;
- $d(\pi, z, 1^n)$ outputs the answer given the full transcript π and the input z of the player to decide, as chosen by $D(\pi, 1^n)$.

We define the functions and formulas $\mathsf{CC}(\mathcal{P}, 1^n, 1^m)$, $\pi_i = \pi_i(x, y)$, $\pi = \pi(x, y) = \pi_t(x, y)$, and $\mathsf{Fail}_f(\mathcal{P}, n)$ as in Section 3.2.

Let $\mathsf{LB}'^{f}_{n,m}(\mathcal{P})$ denote the formula that states that if $\mathcal{P}$ is a communication protocol with communication complexity m on input length n, then it must fail to compute f. That is, for every $n, m \in \mathsf{Log}$, every PV function f, and every uniform communication protocol $\mathcal{P}$, we define

$$\mathsf{LB}'^{f}_{n,m}(\mathcal{P}) \triangleq \mathsf{CC}(\mathcal{P}, 1^n, 1^m) \to \mathsf{Fail}_f(\mathcal{P}, 1^n).$$

Theorem 4.4. *Let $n, m \in \mathsf{Log}$, $n > m$. For every uniform communication protocol $\mathcal{P}$, there is a* PV *function f such that* $\mathsf{PHP}'^{2^n}_{2^m}(f) \vdash_{\mathsf{PV}_1} \mathsf{LB}'^{\mathsf{EQ}}_{n,m}(\mathcal{P})$.

Proof Sketch. Similar to the non-uniform case, we use the standard fooling set argument. We argue in PV_1. Let $\mathcal{P} = (g_{\mathsf{A}}, g_{\mathsf{B}}, d_{\mathsf{B}})$ be any uniform communication protocol such that $\mathsf{CC}(\mathcal{P}, n, m)$ is true. Let $f : \{0,1\}^n \to \{0,1\}^m$ be the function that given $x \in \{0,1\}^n$, simulates $\mathcal{P}$ on input (x, x) and outputs the transcript $\pi \in \{0,1\}^m$. Note that f is a PV-function as S, C, D, d are PV-functions. By $\mathsf{PHP}'^{2^n}_{2^m}(f)$, there are $x, y \in \{0,1\}^n$ such that $f(x) = f(y)$. By the same argument as in Theorem 3.2, we can show that the communication transcript for $\mathcal{P}$ given (x, x) is the same as the transcript given (x, y), which means that the protocol must be wrong on one of the inputs. $\square$

Let $m = m(n)$ be a PV function. We can define $\mathsf{LB}'^{f}_{n,m}(\mathsf{PV})$ as the set of sentences

$$\forall n \in \mathsf{Log}\ \mathsf{LB}'^{f}_{n,m(n)}(\mathcal{P})$$

for every uniform communication protocol $\mathcal{P} = (S, C, D, d)$.

Corollary 4.5. $\mathsf{WPHP}'(\mathsf{PV}) \vdash_{\mathsf{PV}_1} \mathsf{LB}'^{\mathsf{EQ}}_{n,n-1}(\mathsf{PV})$.

4.4 Uniform CC lower bounds $\Rightarrow$ single-tape lower bounds

Now we show that CC lower bounds against uniform communication protocol for equality is sufficient to prove the single-tape Turing machine lower bound for Palindrome. This follows from a careful formalization of the standard proof of the Palindrome lower bound in PV_1.

Theorem 4.6. *For every constant $\beta \in (0,1)$, $\mathsf{LB}'^{\mathsf{EQ}}_{n,\beta n}(\mathsf{PV}) \vdash_{\mathsf{PV}_1} \mathsf{LB}^{\mathsf{PAL},c}_{\text{1-tape}}(n^2)$, where $c(M) \triangleq \beta/(15|M|)$, and $|M|$ is the number of bits required to describe the internal state of M.*

Proof. Let M be any single-tape Turing machine and $\beta \in (0,1)$ be a constant. We need to show that $\mathsf{PV}_1 + \mathsf{LB}'^{\mathsf{EQ}}_{n,\beta n}(\mathsf{PV})$ proves $\mathsf{LB}^{\mathsf{PAL},c}_{\text{1-tape}}(M, n^2)$, where $c(M) \triangleq \beta/(15|M|)$.

We argue in PV_1. Suppose, towards a contradiction, that $\mathsf{LB}^{\mathsf{PAL},c}_{\text{1-tape}}(M, n^2)$ is false. This means that there exists an input length $n \in \mathsf{Log}$ such that the single-tape Turing machine M decides Palindrome in $c(M) \cdot n^2$ time.

Let $c = c(M)$ and $t = cn^2$. Without loss of generality, we assume that $n = 3n'$, and consider inputs of the form $x \| 0^{n'} \| y \in \{0,1\}^n$, where $x, y \in \{0,1\}^{n'}$. At the beginning of the computation, the tape contains only the input string $x \| 0^{n'} \| y$. Let $b_1, b_2, \ldots, b_{n'}$ denote the n' cells on the tape corresponding to the middle string $0^{n'}$. During the entire t steps of computation, we know by an averaging argument that for every (x, y), there is an $i \in [n']$ such that the head moves across the border between b_i and the block to its right for at most $t/n' = 3cn$ times. This can be proved in PV_1: As $n' \in \mathsf{Log}$, we can define a PV function that enumerates all $i \in [n']$ and finds the lexicographic first such i, and proves its correctness by induction over quantifier-free formulas. We call any such index i a *splitting point* corresponding to the input $x0^{n'}y$.

Let L_i be the left fragment of the tape from b_i (including b_i) and R_i be the right fragment excluding b_i. Let $k = |M|$ be the number of bits to describe the internal states of the Turing machine M. We now describe a communication protocol that attempts to solve EQ over input length n' with communication complexity $O(3ckn)$.

1. Alice runs the Turing machine M on $x \| 0^{n'} \| \mathsf{rev}(x) \in \{0,1\}^n$, and finds the lexicographic first splitting point i. Similarly, Bob runs the Turing machine M on $y \| 0^{n'} \| \mathsf{rev}(y)$ and finds the lexicographic first splitting point j. Alice transmits i to Bob. If $i \neq j$, Bob rejects. Otherwise, Alice and Bob proceed as follows.

2. Alice and Bob aim to jointly simulate the single-tape Turing machine M over the input $x\|0^{n'}\|\mathsf{rev}(y)$. Alice is responsible for the blocks in L_i and Bob is responsible for the blocks in R_i. At the beginning, Alice and Bob initialize their tape cells according to their corresponding inputs obtained from x and y, respectively.

3. If the head is in the portion of the tape of Alice (resp. Bob), Alice (resp. Bob) simulates the machine by themself. When the head moves across the border of their tapes, say from Alice's tape to Bob's tape, Alice notifies Bob the internal state of the machine using a k-bit message.

4. If the head crosses the splitting point i for more than $3cn$ times, Alice and Bob terminate the protocol and reject the input.

The communication complexity in the first step is only $O(\log n)$. Since the head moves across the border of Alice's and Bob's tape for at most $3cn$ times during the simulation (before they terminate the protocol and reject), the total length of the transcript encoded in binary is at most $5\cdot 3ckn$. Since we choose $c=\beta/(15k)$, the communication complexity of the protocol is at most βn. Moreover, it is easy to verify that the communication protocol is uniform as they are simulating a uniform Turing machine. By $\mathsf{LB}'^{\mathsf{EQ}}_{n,\beta n}(\mathsf{PV})$, we know that the communication protocol cannot be correct; that is, there is a pair of (x,y) such that the communication protocol does not output $\mathsf{EQ}(x,y)$. Consider the following two cases.

- If $x=y$, then $i=j$ and thus Bob will not reject in the first step. Moreover, i is a correct splitting point corresponding to the input $x\|0^{n'}\|\mathsf{rev}(y)$, and therefore the head crosses the splitting point for at most $3cn$ time. By induction on the number of steps, we can prove that the communication protocol correctly simulates M on the input $x\|0^{n'}\|\mathsf{rev}(y)$ and it will accept, as M is a correct algorithm for Palindrome.

- Now we assume that $x\neq y$. If $i\neq j$, Bob will reject in the first step. Also if $i=j$ but i is not a splitting point corresponding to the input $x\|0^{n'}\|\mathsf{rev}(y)$, they will reject according to the Item 4. Otherwise, by induction on the number of steps, we can prove that the communication protocol correctly simulates M on the input $x\|0^{n'}\|\mathsf{rev}(y)$ and it will reject, as M is a correct algorithm for Palindrome and $x\neq y$ if and only if $\mathsf{rev}(x)\neq\mathsf{rev}(y)$.

Note that we only need induction for quantifier-free PV_1 formulas in both cases above. Therefore, it is impossible that the communication protocol does not output $\mathsf{EQ}(x,y)$, which leads to a contradiction. □

4.5 Uniform one-way CC lower bounds $\Leftrightarrow$ $\mathsf{WPHP}'(\mathsf{PV})$

We have shown that uniform CC lower bounds for EQ, single-tape Turing machine lower bounds for Palindrome, and $\mathsf{WPHP}'(\mathsf{PV})$ are all equivalent. Indeed, we can also show that uniform one-way CC lower bounds for EQ is also in the equivalence class.

A uniform one-way communication protocol is defined by two PV-functions $g_{\mathsf{A}}, d_{\mathsf{B}}$, where $g_{\mathsf{A}}(x,1^n)\to\pi$ outputs the message to transmit, and $d_{\mathsf{B}}(x,\pi,1^n)\in\{0,1\}$ outputs Bob's answer. The behavior of a uniform one-way protocol can be defined naturally following the non-uniform case (see Section 3.2). We can define the formulas $\underline{\mathsf{CC}}(\mathcal{P},1^n,1^n)$ and $\underline{\mathsf{Fail}}_f(\mathcal{P},1^n)$ as in Section 3.2.

Let $\underrightarrow{\mathsf{LB}}'^{f}_{n,m}(\mathcal{P})$ denote the formula that if $\mathcal{P}$ is a uniform one-way communication protocol with communication complexity m on input length n, then it must fail to compute f. That is, for every

$n, m \in \mathsf{Log}$, every PV function f, and every uniform communication protocol $\mathcal{P}$, we define

$$\underrightarrow{\underline{\mathsf{LB}}}'^{f}_{n,m}(\mathcal{P}) \triangleq \underline{\mathsf{CC}}(\mathcal{P}, 1^n, 1^m) \to \mathsf{Fail}_f(\mathcal{P}, 1^n).$$

Theorem 4.7 ($\mathsf{LB}'^{f} \Rightarrow \underrightarrow{\underline{\mathsf{LB}}}'^{f}$)**.** *Let $n, m \in \mathsf{Log}$ and f be a PV function. For every uniform one-way protocol $\underline{\mathcal{P}}$, there is a uniform communication protocol $\mathcal{P}$ such that $\mathsf{LB}'^{f}_{n,m}(\mathcal{P}) \vdash_{\mathsf{PV}_1} \underrightarrow{\underline{\mathsf{LB}}}'^{f}_{n,m}(\underline{\mathcal{P}})$.*

Proof Sketch. The proof is essentially the same as Theorem 3.4. Let $\underline{\mathcal{P}} = (g_{\mathsf{A}}, d_{\mathsf{B}})$ be a uniform one-way communication protocol. We can define another uniform communication protocol $\mathcal{P}$, in which Alice transmits the i-th bit of $g_{\mathsf{A}}(x)$ in the i-th round, and Bob outputs $d_{\mathsf{B}}(y, g_{\mathsf{A}}(x))$ after he received all messages from Alice. It is easy to verify that if $\underline{\mathcal{P}}$ violates the lower bound $\underrightarrow{\underline{\mathsf{LB}}}'^{f}_{n,m}$, $\mathcal{P}$ also violates the lower bound $\mathsf{LB}'^{f}_{n,m}$ as it is simulating $\underline{\mathcal{P}}$. $\square$

Theorem 4.8 ($\underrightarrow{\underline{\mathsf{LB}}}'^{\mathsf{EQ}} \Rightarrow \mathsf{WPHP}'$)**.** *Let $n, m \in \mathsf{Log}$, $m < n$, and f be a PV function. For every PV function g, there is a one-way communication protocol $\mathcal{P}$ such that $\underrightarrow{\underline{\mathsf{LB}}}^{\mathsf{EQ}}_{n,m}(\mathcal{P}) \vdash_{\mathsf{PV}_1} \mathsf{PHP}'^{2^n}_{2^m}(g)$.*

Proof Sketch. The proof is essentially the same as Theorem 3.5. For every PV function $g : \{0,1\}^n \to \{0,1\}^m$, we can construct the following uniform one-way communication protocol for EQ: Alice transmits $g(x) \in \{0,1\}^m$ to Bob, and Bob outputs 1 if and only if $g(x) = g(y)$. The correctness of the communication protocol follows from the fact that $\mathsf{PHP}'^{2^n}_{2^m}(g)$ is false, i.e., g is a "perfect" hash function. This violates the lower bound $\underrightarrow{\underline{\mathsf{LB}}}^{\mathsf{EQ}}_{n,m}(\mathcal{P})$. $\square$

4.6 Summary of the equivalence class

Now we summarize the equivalence class for $\mathsf{WPHP}'(\mathsf{PV})$.

Theorem 4.9. *Let $\varepsilon, \beta \in (0,1)$ be arbitrary constants. The following sets of sentences are equivalent with respect to PV_1.*

1. $\mathsf{WPHP}'(\mathsf{PV})$*, i.e., the uniform weak pigeonhole principle for* PV *function symbols.*
2. $\{\forall n \in \mathsf{Log}\ \mathsf{PHP}'^{2^n}_{2^{n-1}}(g) \mid g \text{ is a } \mathsf{PV} \text{ function}\}$*. This stands for the uniform weak pigeonhole principle for* PV *functions with single-bit shrinkage.*
3. $\{\forall n \in \mathsf{Log}\ \mathsf{PHP}'^{2^n}_{2^{n^\varepsilon}}(g) \mid g \text{ is a } \mathsf{PV} \text{ function}\}$*. This stands for the uniform weak pigeonhole principle for* PV *functions with polynomial length shrinkage.*
4. $\{\forall n \in \mathsf{Log}\ \mathsf{LB}'^{\mathsf{EQ}}_{n,n-1}(\mathcal{P}) \mid \mathcal{P} \text{ is a uniform communication protocol}\}$*, i.e.,* EQ *has uniform communication complexity greater than $n-1$.*
5. $\{\forall n \in \mathsf{Log}\ \mathsf{LB}'^{\mathsf{EQ}}_{n,n^\varepsilon}(\mathcal{P}) \mid \mathcal{P} \text{ is a uniform communication protocol}\}$*, i.e.,* EQ *has uniform communication complexity greater than n^ε.*
6. $\{\forall n \in \mathsf{Log}\ \underrightarrow{\underline{\mathsf{LB}}}'^{\mathsf{EQ}}_{n,n-1}(\mathcal{P}) \mid \mathcal{P} \text{ is a uniform one-way communication protocol}\}$*, i.e.,* EQ *has uniform communication complexity greater than $n-1$ against one-way protocols.*
7. $\{\forall n \in \mathsf{Log}\ \underrightarrow{\underline{\mathsf{LB}}}'^{\mathsf{EQ}}_{n,n^\varepsilon}(\mathcal{P}) \mid \mathcal{P} \text{ is a uniform one-way communication protocol}\}$*, i.e.,* EQ *has uniform communication complexity greater than n^ε against one-way protocols.*
8. $\mathsf{LB}^{\mathsf{PAL},c}_{\text{1-tape}}(n^2)$*, i.e., Palindrome requires $\Omega(n^2)$ time on single-tape Turing machines, where the constant factor $c(M) \triangleq 1/(100|M|)$, and $|M|$ is the number of bits used to describe the internal state of the Turing machine.*
9. $\mathsf{LB}^{\mathsf{PAL}}_{\text{1-tape}}(\beta n^{1+\beta})$*, i.e., Palindrome requires $n^{1+\Omega(1)}$ time on single-tape Turing machines.*

Proof. (1) $\equiv_{\mathsf{PV}_1}$ (2) $\equiv_{\mathsf{PV}_1}$ (3): This is an easy corollary of Lemma 4.1.

(2) $\vdash_{\mathsf{PV}_1}$ (4) follows from Theorem 4.4.

(4) $\vdash_{\mathsf{PV}_1}$ (5) and (8) $\vdash_{\mathsf{PV}_1}$ (9): Trivial, since quantitatively stronger lower bounds imply weaker lower bounds under any reasonable formalization.

(4) $\vdash_{\mathsf{PV}_1}$ (6) and (5) $\vdash_{\mathsf{PV}_1}$ (7) follow from Theorem 4.7.

(6) $\vdash_{\mathsf{PV}_1}$ (2) and (5) $\vdash_{\mathsf{PV}_1}$ (3) follow from Theorem 4.8.

(6) $\vdash_{\mathsf{PV}_1}$ (8) follows from Theorem 4.6.

(9) $\vdash_{\mathsf{PV}_1}$ (1) follows from Theorem 4.3. □

5 Equivalence Class for $\mathsf{WPHPWIT}(\mathsf{PV})$

In this section, we explore the equivalence class of witnessing pigeonhole principle for PV functions. We will show that various lower bounds for the *set-disjointness function* are equivalent to the witnessing pigeonhole principle.

5.1 Technical lemmas

Lemma 5.1 (Implicit in [PWW88, Jeřo7b]). *Let $t(\cdot)$ be any PV function such that $\mathsf{PV} \vdash \forall x\ t(x) \geq 2x$. Then $\mathsf{WPHPWIT}(\mathsf{PV}) \equiv_{\mathsf{PV}_1} \{\forall a\ \mathsf{PHPWIT}_a^{t(a)}(f,g) \mid f,g$ are PV functions$\}$.*

5.2 $\mathsf{SetDisj}$ lower bounds and $\mathsf{WPHPWIT}(\mathsf{PV})$

For strings $x, y \in \{0,1\}^n$, we let $\langle x, y\rangle \triangleq \sum_i x_i \cdot y_i$. The set-disjointness function $\mathsf{SetDisj} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is defined as $\mathsf{SetDisj}(x,y) \triangleq [\langle x,y\rangle = 0]$, i.e., $\mathsf{SetDisj}(x,y) = 1$ if and only if there is no $i \in [n]$ such that $x_i = y_i = 1$. It is well known that the communication complexity of $\mathsf{SetDisj}$ is at least n.

$\mathsf{SetDisj}$ lower bounds from $\mathsf{WPHP}(\mathsf{PV})$. Let $n, m \in \mathsf{Log}$, $n < m$. Recall that the (non-uniform) communication complexity lower bound for $\mathsf{SetDisj}$ is formalized as the sentence:

$$\mathsf{LB}_{n,m}^{\mathsf{SetDisj}} \triangleq \forall \mathcal{P}\ (\mathsf{CC}(\mathcal{P}, 1^n, 1^m) \to \mathsf{Fail}_{\mathsf{SetDisj}}(\mathcal{P}, 1^n)).$$

Theorem 5.2. *For $n, m \in \mathsf{Log}$, $m < n$, $\mathsf{PHP}_{2^m}^{2^n}(\mathsf{PV}) \vdash_{\mathsf{PV}_1} \mathsf{LB}_{n,m}^{\mathsf{SetDisj}}$.*

Proof Sketch. This follows the standard fooling set proof of the communication complexity lower bound (see, e.g., [AB09]), which is similar to the proof of Theorem 3.2. For simplicity, we only sketch the proof and identify the differences. Below we use $\overline{x}$ to denote the coordinate-wise complement of a string x.

Let $n \in \mathsf{Log}$ and $\mathcal{P}$ be a protocol such that $\mathsf{CC}(\mathcal{P}, 1^n, 1^m)$. We define the circuit $D : \{0,1\}^n \to \{0,1\}^m$ as $D(x) \triangleq \pi(x, \overline{x})$, where $\langle \overline{x}, x\rangle = 0$. That is, $D(x)$ runs the protocol over the input $(x, \overline{x})$ and outputs the transcript. By $\mathsf{PHP}_{2^m}^{2^n}(\mathsf{PV})$, it follows that for distinct $x, y \in \{0,1\}^n$, $D(x) = D(y)$. We can then prove by induction that $\pi(x, \overline{y}) = \pi(x, \overline{x}) = \pi(y, \overline{y})$.

However, since $\mathsf{SetDisj}(x, \overline{x}) = \mathsf{SetDisj}(y, \overline{y}) = 1$ but $\mathsf{SetDisj}(x, \overline{y}) = 0$, the protocol $\mathcal{P}$ cannot be correct no matter which player will output the answer. This means that $\mathsf{Fail}_{\mathsf{SetDisj}}(\mathcal{P}, n)$ is true, which completes the proof. □

Note that one can also apply Theorem 3.8 and plug in the fooling set construction $S \triangleq \{(x, \overline{x}) \mid x \in \{0,1\}^n\}$ of size n.

We do not know whether the inversion of the theorem holds. It is an interesting open problem to find an equivalence for $\mathsf{LB}^{\mathsf{SetDisj}}$.

One-way lower bounds for $\mathsf{SetDisj}$. We also define the communication complexity lower bound for $\mathsf{SetDisj}$ against one-way (non-interactive) communication protocols. Let $n, m \in \mathsf{Log}$ and $\underrightarrow{\mathsf{LB}}^{\mathsf{SetDisj}}_{n,m}$ be the sentence

$$\underrightarrow{\mathsf{LB}}^{\mathsf{SetDisj}}_{n,m} \triangleq \forall \mathcal{P}\ (\underline{\mathsf{CC}}(\mathcal{P}, 1^n, 1^m) \to \underline{\mathsf{Fail}}_{\mathsf{SetDisj}}(\mathcal{P}, 1^n)).$$

Theorem 5.3 ($\mathsf{WPHPWIT} \Rightarrow \underrightarrow{\mathsf{LB}}^{\mathsf{SetDisj}}$)**.** *Let $n, m \in \mathsf{Log}$, $n > m$. Then* $\mathsf{PHPWIT}^{2^m}_{2^n}(\mathsf{PV}) \vdash_{\mathsf{PV}_1} \underrightarrow{\mathsf{LB}}^{\mathsf{SetDisj}}_{n,m}$.

Proof. We argue in PV_1. Towards a contradiction, we assume that there are $n, m \in \mathsf{Log}$, $n > m$, and a one-way protocol $\mathcal{P}$ with communication complexity at most m that correctly computes $\mathsf{SetDisj}$. In other words, there are functions $g_{\mathsf{A}} : \{0,1\}^n \to \{0,1\}^m$ and $d_{\mathsf{B}} : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ such that for every $x, y \in \{0,1\}^n$, $\mathsf{SetDisj}(x, y) = d_{\mathsf{B}}(y, g_{\mathsf{A}}(x))$. We now define a pair of functions (f_1, f_2) that violates $\mathsf{PHPWIT}^{2^m}_{2^n}(\mathsf{PV})$.

- $f_2 : \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^m$ is defined as $f_2(x, C) \triangleq C(x)$, where the second input is a circuit $C : \{0,1\}^n \to \{0,1\}^\ell$. The second input will be fixed to be g_{A}.
- $f_1 : \{0,1\}^m \times \{0,1\}^* \to \{0,1\}^n$ is defined as follows. Let the second input be fixed as the circuit d_{B}. On the input $y \in \{0,1\}^m$, for any $i \in [m]$, the i-th output bit equals to 1 if and only if $d_{\mathsf{B}}(e^i, y) = 0$, where $e^i \in \{0,1\}^n$ is the string with the i-th index being 1 and other indices being 0.

Then we will show that $\mathsf{PHPWIT}^{2^m}_{2^n}(f_1, f_2)$ does not hold, that is, for every $x \in \{0,1\}^n$, $f_1(f_2(x, g_{\mathsf{A}}), d_{\mathsf{B}}) = x$. For any $i \in [m]$, the i-th bit of $f_1(f_2(x, g_{\mathsf{A}}))$ equals to 1 if and only if $d_{\mathsf{B}}(e^i, g_{\mathsf{A}}(x)) = 0$, which, by the assumption that $\mathcal{P}$ is a correct communication protocol for $\mathsf{SetDisj}$, holds if and only if $\langle e^i, x\rangle = 1$ (i.e. $x_i = 1$). This means that the i-th bit of $f_1(f_2(x, g_{\mathsf{A}}))$ equals to x_i and therefore $f_1(f_2(x, g_{\mathsf{A}})) = x$. $\square$

We know show the inverse of the theorem. Indeed, we can prove the stronger result that *any* one-way communication complexity lower bound implies PHPWIT.

Theorem 5.4 ($\underrightarrow{\mathsf{LB}} \Rightarrow \mathsf{PHPWIT}$)**.** *Let $n, m \in \mathsf{Log}$, $m < n$. Then for every PV function $h\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$,* $\underrightarrow{\mathsf{LB}}^{h}_{n,m} \vdash_{\mathsf{PV}_1} \mathsf{PHPWIT}^{2^m}_{2^n}(\mathsf{PV})$.

Proof. We argue in PV_1. Towards a contradiction, we assume that $\mathsf{PHPWIT}^{2^m}_{2^n}(\mathsf{PV})$ does not hold, that is, there are PV functions $f : \{0,1\}^m \times \{0,1\}^* \to \{0,1\}^n$ and $g : \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^m$, $n, m \in \mathsf{Log}$, $m < n$, and z_1, z_2 such that $f(g(\cdot, z_1), z_2)$ is an identity function. We will construct a one-way communication protocol that computes h on input length $n \in \mathsf{Log}$ with communication complexity at most $m \in \mathsf{Log}$, which leads to a contradiction with $\underrightarrow{\mathsf{LB}}^{h}_{n,m}$. The communication protocol is as follows: On input $(x, y) \in \{0,1\}^n \times \{0,1\}^n$, where Alice holds x and Bob holds y.

- Alice sends $g(x, z_1) \in \{0,1\}^m$ to Bob, that is, the circuit $g_{\mathsf{A}}(x) \triangleq g(x, z_1)$.
- Bob accepts if and only if $h(f(g(x, z_1), z_2), y) = 1$, that is, the decision circuit $d_{\mathsf{B}}(y) \triangleq h(f(g(x, z_1), z_2), y)$ (see Lemma 2.2).

The communication complexity of the protocol is clearly m. Since $f(g(x, z_1), z_2) = x$ for every x, the protocol correctly computes $h(x, y)$. This concludes the proof. $\square$

5.3 Distance lower bounds for decodable ECC $\Leftrightarrow$ $\mathsf{WPHPWIT(PV)}$

We will also show that a distance lower bound for decodable error-correcting codes as an analog of the Singleton bound is equivalent to $\mathsf{WPHPWIT(PV)}$.

An $[n,e,m]$-*decodable error-correcting code* is defined by a pair of circuits (E,D), where $E : \{0,1\}^n \to \{0,1\}^m$, $D : \{0,1\}^m \to \{0,1\}^n$, such that for every error vector $\gamma \in \{0,1\}^m$ of Hamming weight at most e, $D(E(x) \oplus \gamma) = x$. Let $n,e,m \in \mathsf{Log}$ and (E,D) be a pair of circuits, we can formalize the definition in PV_1 with:

$$\mathsf{DistDecECC}(E,D,1^n,1^e,1^m) \triangleq \forall x \in \{0,1\}^n\ \forall \gamma \in \{0,1\}^m\ (\Delta(y) \le e \to D(E(x)\oplus\gamma) = x).$$

A variant of the Singleton bound for decodable ECC states that $m \ge n + \lfloor \log \binom{m}{e} \rfloor$, which can be formalized as

$$\mathsf{Singleton}' \triangleq \forall n,e,m \in \mathsf{Log}\ \forall (E,D)\ \left(\mathsf{DistDecECC}(E,D,1^n,1^e,1^m) \to m \ge n + \left\lfloor \log \binom{m}{e} \right\rfloor\right).$$

To prove the equivalence between $\mathsf{Singleton}'$ and $\mathsf{WPHPWIT}$, we will need the following lemma that gives an explicit encoding of strings with a fixed Hamming weight.

Lemma 5.5. PV_1 *proves that for every* $m,e \in \mathsf{Log}$*, there is a pair of circuits* $(F_{m,e}, G_{m,e})$*, where* $F_{m,e} : [\binom{m}{e}] \to H_{m,e}$*,* $G_{m,e} : H_{m,e} \to [\binom{m}{e}]$*,* $H_{m,e} \triangleq \{x \in \{0,1\}^m \mid |x| = e\}$*, such that* $F_{m,e} \circ G_{m,e}$ *and* $G_{m,e} \circ F_{m,e}$ *are identity functions on their domains, respectively.*

Proof Sketch. We will define $F_{m,e}(k)$ to be the k-th string (in lexicographic order) with Hamming weight e, and $G_{m,e}(x)$ to be the rank of x in lexicographic order. We can construct $F_{m,e}$ and $G_{m,e}$ recursively as follows.

- If $m = e$, $H_{m,e} = \{1^m\}$, we define $F_{m,e}(1) \triangleq 1^m$ and $G_{m,e}(1^m) \triangleq 1$.
- If $e = 0$, $H_{m,e} = \{0^m\}$, we define $F_{m,e}(1) \triangleq 0^m$ and $G_{m,e}(0^m) \triangleq 1$.
- Otherwise, we know by the binomial formula that $\binom{m}{e} = \binom{m-1}{e-1} + \binom{m-1}{e}$.
  - If $k \le \binom{m-1}{e}$, $F_{m,e}(k) \triangleq 0 \| F_{m-1,e}(k)$; otherwise, $F_{m,e}(k) \triangleq 1 \| F_{m-1,e-1}(k - \binom{m-1}{e})$.
  - $G_{m,e}(0\|x) \triangleq G_{m-1,e}(x)$; $G(m,e)(1\|x) \triangleq G_{m-1,e-1}(x) + \binom{m-1}{e}$.

As $m,e \in \mathsf{Log}$, the recursive definition is realizable in PV. Moreover, by induction over quantifier-free formulas, which is admissible in PV_1, we can prove that $F_{m,e}(G_{m,e}(x)) = x$ for every $x \in H_{m,e}$ and $G_{m,e}(F_{m,e}(k)) = k$ for every $k \in [\binom{m}{e}]$. □

Theorem 5.6. $\mathsf{WPHPWIT(PV)} \equiv_{\mathsf{PV}_1} \mathsf{Singleton}'$.

Proof. ($\mathsf{WPHPWIT(PV)} \vdash_{\mathsf{PV}_1} \mathsf{Singleton}'$). This follows from the standard proof of the lower bound. We argue in $\mathsf{PV}_1 + \mathsf{WPHPWIT(PV)}$. Let $h(m,e) \triangleq \lfloor \log \binom{m}{e} \rfloor$ for simplicity. (Note that $\binom{m}{e}$ can be defined in PV_1 as $m,e \in \mathsf{Log}$.) Suppose, towards a contradiction, that there are $n,e,m \in \mathsf{Log}$ and circuits E,D such that $\mathsf{DistDecECC}(E,D,n,e,m)$ but $m < n + h(m,e)$. Let (F,G) be the pair of circuits in Lemma 5.5, where $F : [\binom{m}{e}] \to H_{m,e}$ and $G : H_{m,e} \to [\binom{m}{e}]$. Consider the following pair of circuits (A,B).

- $A : \{0,1\}^{n+h(m,e)} \to \{0,1\}^m$, where the input consists of a string $x \in \{0,1\}^n$ and a number $k \in [\binom{m}{e}]$ encoded in binary, we compute $A(x\|k) \triangleq E(x) \oplus F(k)$.

- $B : \{0,1\}^m \to \{0,1\}^{n+h(m,e)}$. Given the input $y \in \{0,1\}^m$, we compute $x \triangleq D(y)$, $\lambda \triangleq E(x) \oplus y$, and output $B(y) \triangleq x \| G(\lambda)$.

By $\mathsf{WPHPWIT}(\mathsf{PV})$, there exists a string $z \in \{0,1\}^{n+h(m,e)}$ such that $B(A(z)) \neq z$. Let $z = x\|k$ for $x \in \{0,1\}^n$ and $k \in \{0,1\}^{h(m,e)}$, and $\gamma = F(k)$. One can see that

$$B(A(x\|k)) = D(E(x) \oplus \gamma)\|G(F(k)) = D(E(x) \oplus \gamma)\|k,$$

where γ is of Hamming weight at most e. This means that $D(E(x) \oplus \gamma) \neq x$, which violates the correctness of the error-correcting code.

($\mathsf{Singleton}' \vdash_{\mathsf{PV}_1} \mathsf{WPHPWIT}(\mathsf{PV})$). By Lemma 5.1, it suffices to prove that $\mathsf{Singleton}' \vdash_{\mathsf{PV}_1} \forall a\ \mathsf{PHPWIT}_a^{t(a)}(\mathsf{PV})$ for any PV function t such that $t(a) \geq 2a$. Let c be the constant in Lemma 2.1 and (E, D) be the ECC. Fix $t(a) = 2^{\log^{2c}(a)}$. For every pair (f, g) of PV functions and any a, we need to show that $\mathsf{PV} + \mathsf{Singleton}' \vdash \mathsf{PHPWIT}_a^{t(a)}(f, g)$. We assume for simplicity that $a = 2^n$, while the proof clearly generalizes to all a.

Let (f, g) be any pair of PV functions and a be a number. Since $\mathsf{PHPWIT}_a^{t(a)}$ is provable for each fixed $a \in \mathbb{N}$, we can assume that n is sufficiently large such that $n^{2c} \geq cn^c$. We argue in $\mathsf{PV}_1 + \mathsf{Singleton}'$ that $\mathsf{PHPWIT}_a^{t(a)}(f_1, f_2)$ holds. Towards a contradiction, we assume that there exist z_1, z_2 such that for every $u \in \{0,1\}^{n^{2c}}$, $f_1(f_2(u, z_2), z_1) = u$. Let F_1, F_2 be the Boolean circuits such that $F_i(\cdot) = f_i(\cdot, z_i)$ for $i \in \{1, 2\}$ by Lemma 2.2. We define the following $[n^{2c}, 0.1cn^c, cn^c]$ decodable error-correcting code (E', D') that violates the Singleton bound.

- $E' : \{0,1\}^{n^{2c}} \to \{0,1\}^{cn^c}$, defined as $E'(u) = E(F_2(u))$.
- $D' : \{0,1\}^{cn^c} \to \{0,1\}^{n^{2c}}$, defined as $D'(v) = F_1(D(v))$.

We first verify the correctness. For every $u \in \{0,1\}^{n^{2c}}$ and every error vector $\gamma \in \{0,1\}^{cn^c}$ of Hamming weight at most $0.1cn^c$, we have

$$D'(E'(u) \oplus \gamma) = F_1(D(E(F_2(u)) \oplus \gamma)) = F_1(F_2(u)) = u.$$

Moreover, since $n^{2c} > cn^c$, this ECC clearly violates the Singleton bound. □

5.4 Summary of equivalence class

Now we summarize the equivalence class for $\mathsf{WPHPWIT}(\mathsf{PV})$.

Theorem 5.7. *Let $\varepsilon \in (0,1)$ be a constant. The following sentences or sets of sentences are equivalent with respect to PV_1.*

1. $\mathsf{WPHPWIT}(\mathsf{PV})$*, i.e., the weak witnessing pigeonhole principle for* PV *function symbols.*
2. $\{\forall n \in \mathsf{Log}\ \mathsf{PHPWIT}_{2^n-1}^{2^n}(f, g) \mid f, g$ *are* PV *functions*$\}$. *This stands for the witnessing pigeonhole principle for* PV *functions with single-bit stretch/shrinkage.*
3. $\{\forall n \in \mathsf{Log}\ \mathsf{PHPWIT}_{2^{n^\varepsilon}}^{2^n}(f, g) \mid f, g$ *are* PV *functions*$\}$. *This stands for the witnessing pigeonhole principle for* PV *functions with polynomial length stretch/shrinkage.*
4. $\forall n \in \mathsf{Log}\ \underrightarrow{\mathsf{LB}}_{n,n-1}^{\mathsf{SetDisj}}$*, i.e.,* $\mathsf{SetDisj}$ *has communication complexity greater than $n-1$ against one-way protocols.*
5. $\forall n \in \mathsf{Log}\ \underrightarrow{\mathsf{LB}}_{n,n^\varepsilon}^{\mathsf{SetDisj}}$*, i.e.,* $\mathsf{SetDisj}$ *has communication complexity greater than n^ε against one-way protocols.*
6. $\mathsf{Singleton}'$*, i.e., the Singleton bound for decodable error correcting codes.*

Proof. (1) $\equiv_{\mathsf{PV}_1}$ (2) $\equiv_{\mathsf{PV}_1}$ (3): This is an easy corollary of Lemma 5.1.

(2) $\vdash_{\mathsf{PV}_1}$ (4) follows from Theorem 5.3.

(4) $\vdash_{\mathsf{PV}_1}$ (5): Trivial, since quantitatively stronger lower bounds imply weaker lower bounds under any reasonable formalization.

(5) $\vdash_{\mathsf{PV}_1}$ (3) follows from Theorem 5.4.

(1) $\equiv_{\mathsf{PV}_1}$ (6) follows from Theorem 5.6. □

6 Consequences

In this section we highlight some consequences of our results.

6.1 Conditional unprovability of simple lower bounds

We have established the equivalence between natural combinatorial principles and simple complexity lower bounds, such as the communication complexity lower bound for Equality, and the lower bound against single-tape Turing machines for Palindrome. As explained below, this implies that many simple complexity lower bounds are not provable in PV_1 (nor in APC_1) assuming standard cryptographic assumptions.

Definition 6.1 (Collision-Resistant Hash Function)**.** A *collision-resistant hash function* (CRHF) consists of a probabilistic polynomial-time function $\mathsf{Gen}(1^n)$ and a deterministic polynomial-time function $\mathsf{Eval}(k, x)$. $\mathsf{Gen}(1^n)$ generates a key $k \in \{0,1\}^*$ given 1^n, and $\mathsf{Eval}(k, \cdot)\colon \{0,1\}^n \to \{0,1\}^m$ computes the hash function on a given input x and choice of key k, where we assume that $m(n) < n$. The security guarantee is that for any probabilistic polynomial-time adversary A and for n sufficiently large,

$$\Pr_{A,\, k \leftarrow \mathsf{Gen}(1^n)}[A(1^n, k) \text{ outputs distinct } x_1, x_2 \text{ such that } \mathsf{Eval}(k, x_1) = \mathsf{Eval}(k, x_2)] = 1/n^{\omega(1)}.$$

Theorem 6.2 (Folklore, see, e.g., [Kra01])**.** *Assuming the existence of collision-resistant hash functions,* $\mathsf{WPHP}(\mathsf{PV})$ *cannot be proved in* $\mathsf{APC}_1 \triangleq \mathsf{PV}_1 + \mathsf{dWPHP}(\mathsf{PV})$*.*

Proof Sketch. Towards a contradiction, we assume that $\mathsf{WPHP}(\mathsf{PV})$ can be proved in APC_1. It follows by the witnessing theorem for APC_1 (see, e.g., [Tha02, Jeř05]) that there is a polynomial-time probabilistic algorithm for the following search problem that succeeds with at least $1/\mathsf{poly}(n)$ success probability: Given a Boolean circuit $C : \{0,1\}^n \to \{0,1\}^{n-1}$, output two distinct strings $x, y \in \{0,1\}^n$ such that $C(x) = C(y)$. This can be used to break any collision resistant hash function. □

Similarly to Definition 6.1, we review the notion of a *keyless collision-resistant hash function.*

Definition 6.3 (Keyless Collision-Resistant Hash Function)**.** A *keyless collision-resistant hash function* (CRHF') with hash value length $m = m(n) < n$ is a deterministic uniform polynomial-time algorithm $h : \{0,1\}^n \to \{0,1\}^m$ such that for every uniform probabilistic polynomial-time adversary $A(1^n)$, for every large enough input length n. we have

$$\Pr_{A(1^n)}[A(1^n) \text{ outputs distinct } x_1, x_2 \text{ such that } h(x_1) = h(x_2)] = 1/n^{\omega(1)}.$$

Note that the existence of keyless CHRF is a stronger cryptographic assumption than the existence of CHRF.

Theorem 6.4 (Folklore)**.** *Assuming the existence of keyless collision-resistant hash functions,* $\mathsf{WPHP}'(\mathsf{PV})$ *cannot be proved in* APC_1*.*

Proof Sketch. Let h be a keyless collision-resistant hash function. Towards a contradiction, we assume that $\mathsf{WPHP}'(h)$ can be proved in APC_1. Then by the witnessing theorem for APC_1 (see, e.g., [Tha02, Jeř05]), there is a polynomial-time probabilistic algorithm for the following search problem that succeeds with at least $1/\mathsf{poly}(n)$ success probability: Given 1^n, output distinct strings $x, y \in \{0,1\}^n$ such that $h(x) = h(y)$. This breaks the keyless collision resistant hash function h. $\square$

Therefore, our equivalence results have the following corollaries.

Corollary 6.5. *Under the existence of CRHF, the following lower bounds cannot be proved in* APC_1*.*

- $\forall n \in \mathsf{Log}\ \underset{\longrightarrow}{\mathsf{LB}}^{\mathsf{EQ}}_{n,n^{\varepsilon}}$*, i.e.,* EQ *has communication complexity greater than* n^{ε} *against one-way protocols.*
- $\mathsf{Singleton}$*, i.e., the Singleton bound for error correcting codes.*

Corollary 6.6. *Under the existence of keyless CRHF, the following lower bound cannot be proved in* APC_1*.*

- $\mathsf{LB}^{\mathsf{PAL}}_{\text{1-tape}}(\beta n^{1+\beta})$ *for any fixed constant* $\beta \in (0,1)$*, i.e., Palindrome requires* $n^{1+\Omega(1)}$ *time on single-tape Turing machines.*

These results indicate that although APC_1 is expressive enough to formalize sophisticated complexity-theoretic results such as the PCP theorem [Pic15b] and the Razborov-Smolensky lower bound against $\mathsf{AC}^0[p]$ [MP20], it still has some inherent weaknesses that prevent it from proving very simple lower bounds. This may suggest that unprovability of complexity lower bounds against $\mathsf{PV} + \mathsf{dWPHP}(\mathsf{PV}) + \mathsf{WPHP}(\mathsf{PV})$ will be a better evidence that there is no "simple" proof for the lower bound compared to an unprovability result against APC_1.

6.2 On the derandomization of feasibly definable randomized algorithms

While generic derandomization (e.g. $\mathsf{BPP} = \mathsf{P}$) is possible in the computational regime under plausible assumptions [NW94, IW97, TV07], a significant open problem in bounded arithmetic is whether *probabilistic feasible reasoning* can be "derandomized". A recent conditional result [ILW23] shows that APC_1 is *not* conservative over PV_1, which in a sense rules out (under a cryptographic assumption) the possibility of the most generic notion of "derandomizing probabilistic feasible reasoning". Taking a step back, we can consider whether every feasibly definable randomized algorithm is feasibly definable deterministically.

The witnessing theorem for PV_1 [Coo75, Bus86] shows that if a $\forall\Sigma^b_1$-sentence $\forall x\ \exists y \le t(x)$ is provable in PV_1, then the search problem of given x finding a $y \le t(x)$ such that $\phi(x,y)$ holds (under $\mathbb{N}$) can be solved by a deterministic polynomial time algorithm. Similarly, the witnessing theorem for APC_1 (see, e.g., [Tha02, Jeř05]) shows that if a $\forall\Sigma^b_1$-formula $\forall x\ \exists y \le t(x)\ \phi(x,y)$ can be proved in APC_1, then there is a randomized polynomial-time algorithm that given any x outputs a $y \le t(x)$ such that $\phi(x,y)$ is true (in the standard model $\mathbb{N}$). As explained in Section 1.1, we say that a search problem P (represented by an open PV formula) admits a feasible deterministic (resp. randomized) polynomial-time algorithm if $\forall x\ \exists y\ P(x, y)$ is provable in PV_1 (resp. APC_1). The problem of derandomizing feasible definable randomized algorithms can be formalized as follows:

$$\textit{Is } \mathsf{APC}_1\ \forall\Sigma_1^b\textit{-conservative over } \mathsf{PV}_1\textit{?} \qquad (\star)$$

Another facet of $(\star)$ is related to the derandomization of proofs of complexity-theoretic lower bounds. Recall that most complexity lower bounds against deterministic non-uniform computational models can be formalized as $\forall\Sigma_1^b$ sentences of the following format:

For every input length $n \in \mathsf{Log}$, $n > n_0$, for every (non-uniform) device A from the model, there is an input $x \in \{0, 1\}^n$ such that $A(x) \neq f(x)$.

Therefore, the $\forall\Sigma_1^b$-conservation of APC_1 over PV_1 (as in $(\star)$) implies that every such lower bound that can be proved using feasible probabilistic reasoning (i.e., APC_1 reasoning) can be proved using feasible deterministic reasoning (i.e., PV reasoning).

The APC_1-complete lower bound. An important property of the witnessing pigeonhole principle is that $\mathsf{APC}_1 = \mathsf{PV}_1 + \mathsf{dWPHP}(\mathsf{PV})$ is $\forall\Sigma_1^b$-conservative over $\mathsf{PV}_1 + \mathsf{WPHPWIT}(\mathsf{PV})$ (see Theorem 2.10), namely any $\forall\Sigma_1^b$-sentence provable in APC_1 is also provable in $\mathsf{PV}_1 + \mathsf{WPHPWIT}(\mathsf{PV})$. This, together with the equivalence between $\mathsf{LB}^{\mathsf{SetDisj}}_{n,n^\varepsilon}$ and $\mathsf{WPHPWIT}(\mathsf{PV})$, leads to the following consequence.

Corollary 6.7. *The following statements are equivalent.*

1. APC_1 *is* $\forall\Sigma_1^b$*-conservative over* PV_1*, i.e., feasibly definable randomized algorithms can be feasibly defined deterministically.*
2. $\mathsf{PV}_1 \vdash \mathsf{WPHPWIT}(\mathsf{PV})$*, namely* PV_1 *proves the weak witnessing pigeonhole principle for* PV *functions.*
3. $\mathsf{PV}_1 \vdash \forall n \in \mathsf{Log}\ \underrightarrow{\mathsf{LB}}^{\mathsf{SetDisj}}_{n,n^\varepsilon}$*, namely* PV_1 *proves a* $n^{\Omega(1)}$ *communication complexity lower bound for* $\mathsf{SetDisj}$ *against one-way protocols.*

Proof. (1) $\Rightarrow$ (2): Since $\mathsf{WPHPWIT}(f, g)$ is a $\forall\Sigma_1^b$-sentence for every pair of PV functions (f, g) (see Theorem 2.10), and $\mathsf{APC}_1 \vdash \mathsf{WPHPWIT}(f, g)$ (see Proposition 2.8), it follows that the $\forall\Sigma_1^b$-conservation of APC_1 over PV_1 implies the provability of $\mathsf{WPHPWIT}(f, g)$ in PV_1.

(2) $\Leftrightarrow$ (3) is a direct consequence of Theorem 5.7.

(2) $\Rightarrow$ (1): Assume that $\mathsf{PV}_1 \vdash \mathsf{WPHPWIT}(\mathsf{PV})$. Then $\mathsf{PV}_1 + \mathsf{WPHPWIT}(\mathsf{PV})$ and PV_1 are the same theory. Therefore, Theorem 2.10 implies that APC_1 is $\forall\Sigma_1^b$-conservative over PV_1. $\square$

An interpretation of this result is that the one-way communication complexity lower bound for $\mathsf{SetDisj}$ is an APC_1-*complete lower bound* with respect to PV_1. Similarly, every lower bound in the equivalence class of $\mathsf{WPHPWIT}(\mathsf{PV})$, say the Singleton bound for decodable error correcting codes, is APC_1-complete.

The "easiest" communication complexity lower bound. Our results show that the one-way communication complexity lower bound for $\mathsf{SetDisj}$ is the "easiest" communication complexity lower bound, in the sense that it is implied by any communication complexity lower bound. This is formally captured by the following corollary.

Corollary 6.8. *Let f be an arbitrary* PV *function. For $n, m \in \mathsf{Log}$ such that $m < n$,* $\underrightarrow{\mathsf{LB}}^{f}_{n,m} \vdash_{\mathsf{PV}} \underrightarrow{\mathsf{LB}}^{\mathsf{SetDisj}}_{n,m}$*. Moreover, for any constant $\varepsilon \in (0, 1)$,*

$$\forall n \in \mathsf{Log}\ \underrightarrow{\mathsf{LB}}^{f}_{n,n^{\varepsilon}} \vdash_{\mathsf{PV}_1} \forall n \in \mathsf{Log}\ \underrightarrow{\mathsf{LB}}^{\mathsf{SetDisj}}_{n,n-1},$$

that is, $\underrightarrow{\mathsf{LB}}^{\mathsf{SetDisj}}_{n,n-1}$ *is the easiest communication complexity lower bound.*

Proof. This is a direct consequence of Theorem 5.3 and Theorem 5.7. □

Therefore, it suffices to prove *any* communication complexity lower bound against one-way protocols in PV_1 to prove ($\star$), or to derandomize feasible randomized algorithms, or to derandomize lower bounds proved by probabilistic feasible reasoning.

6.3 Amplification of lower bounds

The equivalences between lower bound statements and pigeonhole principles imply that the provability of many weak lower bounds come with an associated amplification phenomenon, in the sense that a weaker lower bound yields a quantitatively stronger lower bound (i.e. in PV_1). Concretely, PV_1 proves the following:

- EQ has non-uniform (resp. uniform) communication complexity $n^{\Omega(1)}$ implies EQ has non-uniform (resp. uniform) communication complexity n.
- Palindrome requires $n^{1+\Omega(1)}$ time by single-tape Turing machines implies it requires $\Omega(n^2)$ time by single-tape Turing machines.
- $\mathsf{SetDisj}$ has one-way communication complexity $n^{\Omega(1)}$ implies $\mathsf{SetDisj}$ has one-way communication complexity n.

Intuitively, this is because there are quantitative connections between these lower bounds and pigeonhole principles, where the complexity corresponds to the stretch or shrinkage of the function in the corresponding pigeonhole principle. For instance, the complexity of EQ is related to the output length of the function for the pigeonhole principle in Theorems 3.2 and 3.5. The amplification results therefore follow from the equivalence of the variants of the pigeonhole principle for different stretch or shrinkage (see Lemma 3.1, 4.1, and 5.1).

6.4 Connections to TFNP

A line of work related to our results is the program of classifying the computational complexity of total functions in NP and in the polynomial-time hierarchy (see, e.g., [KKMP21] and references therein). Recall that many subclasses of TFNP are defined according to natural combinatorial principles; for instance, the class PPP corresponds to the usual pigeonhole principle (with 2^n pigeons and $2^n - 1$ holes), while the class PEPP corresponds to the dual pigeonhole principle (with $2^n - 1$ pigeons and 2^n holes). In this subsection, we rely on an existing connection between bounded arithmetic and TFNP to extract from our equivalences in the context of reverse mathematics efficient reductions among problems in TFNP. The presentation below assumes basic familiarity with TFNP.

Definition 6.9. The class PWPP (Polynomial Weak Pigeonhole Principle; see [Jeř16]) is the set of TFNP problems that are Turing reducible to the following problem: given a circuit $D : \{0,1\}^n \to \{0,1\}^{n-1}$, the solutions are pairs of distinct strings $x, y \in \{0,1\}^n$ such that $D(x) = D(y)$.

Definition 6.10 (Refutation Problem of a Complexity Lower Bound)**.** Let

$$\forall n \in \mathsf{Log}\ \forall C : \{0,1\}^n \to \{0,1\}\ \exists x \in \{0,1\}^n\ \varphi(n, C, x)$$

be a lower bound sentence that is true over the standard model and is formalized by a $\forall\Sigma_1^b$-formula in the language of PV_1, where C encodes an n-input computation device from a fixed computational model $\mathscr{C}$ (e.g., a single-tape machine with a clocked time bound), and $\varphi(n, C, x)$ is a quantifier-free formula which states that C makes a mistake on the input x (with respect to some fixed task, e.g., deciding the language Palindrome).

The *refutation problem* of the lower bound is the following total search problem: given 1^n and a computation device $C \in \mathscr{C}$, the solutions are strings $x \in \{0,1\}^n$ such that $\varphi(n, C, x)$ is true in the standard model, i.e., C makes a mistake on the input x.

We will need the standard Herbrand Theorem from logic (see, e.g., [Bus98, TS00, Koh08]).

Theorem 6.11 (Herbrand Theorem)**.** *Let T be a first-order universal theory (i.e. the axioms are universal sentences), and $\varphi(\vec{x}, y)$ be a quantifier-free formula with free variables as displayed. If $T \vdash \forall \vec{x}\ \exists y\ \varphi(\vec{x}, y)$, then there are finitely many terms $t_1, \ldots, t_k$ in the language of T such that*

$$T \vdash \forall \vec{x}\ \big(\varphi(\vec{x}, t_1(\vec{x})) \vee \ldots \vee \varphi(\vec{x}, t_k(\vec{x}))\big).$$

Theorem 6.12. *Let $\Phi \triangleq \forall n \in \mathsf{Log}\ \forall C : \{0,1\}^n \to \{0,1\}\ \exists x \in \{0,1\}^n\ \varphi(n, C, x)$ be a lower bound formalized by a $\forall\Sigma_1^b$-formula in the language of PV_1. Then:*

- *If $\mathsf{WPHP}(\mathsf{PV}) \vdash_{\mathsf{PV}_1} \Phi$, then the refutation problem of Φ is in PWPP.*
- *If $\Phi \vdash_{\mathsf{PV}_1} \mathsf{WPHP}(\mathsf{PV})$, then the refutation problem of Φ is PWPP-hard under Turing reductions.*

Therefore, if $\mathsf{WPHP}(\mathsf{PV}) \equiv_{\mathsf{PV}_1} \Phi$, then the refutation problem of Φ is PWPP-complete under Turing reductions.

Proof Sketch. This connection is well known to researchers in bounded arithmetic, and as such we only provide a sketch of the argument. Moreover, we only discuss the first bullet, since the second bullet can be proved in similar way.

Suppose that $\mathsf{WPHP}(\mathsf{PV}) \vdash_{\mathsf{PV}_1} \Phi$, which means that $\mathsf{PV}_1 + \mathsf{WPHP}(\mathsf{PV}) \vdash \Phi$. We know by Lemma 3.1 that $\mathsf{PV}_1 + \mathsf{PHP}^{2^n}_{2^n-1}(\mathsf{PV}) \vdash \Phi$. Moreover, let $\mathsf{Eval}(x, C)$ be the PV function that evaluates the Boolean circuit $C\colon \{0,1\}^n \to \{0,1\}^{n-1}$ over the input $x \in \{0,1\}^n$. Then $\mathsf{PV}_1 + \mathsf{PHP}^{2^n}_{2^n-1}(\mathsf{Eval}) \vdash \Phi$ (see Lemma 2.2).

Let f be a new function symbol whose intended interpretation is as follows: given any input D describing a circuit $D\colon \{0,1\}^n \to \{0,1\}^{n-1}$, f outputs a pair (x, y) of distinct strings such that $D(x) = D(y)$. Then $\mathsf{PHP}^{2^n}_{2^n-1}(\mathsf{Eval})$ can be restated as a *universal sentence* in the language of PV_1 extended with the new function symbol f.

Let $\mathsf{PHP}^{2^n}_{2^n-1}(\mathsf{Eval}, f)$ be the universal sentence discussed above, $\mathcal{L}(\mathsf{PV}, f)$ be the language of PV extended with f, and $T \triangleq \mathsf{PV}_1 + \mathsf{PHP}^{2^n}_{2^n-1}(\mathsf{Eval}, f)$. It is easy to argue that the universal theory T proves Φ, since $\mathsf{PHP}^{2^n}_{2^n-1}(\mathsf{Eval})$ can be derived from $\mathsf{PHP}^{2^n}_{2^n-1}(\mathsf{Eval}, f)$. Therefore, by Herbrand Theorem (see Theorem 6.11), there is a constant $k \in \mathbb{N}$ and terms $t_1, \ldots, t_k$ in $\mathcal{L}(\mathsf{PV}, f)$ such that

$$T \vdash \forall n \in \mathsf{Log}\ \forall C : \{0,1\}^n \to \{0,1\}\ \big(\varphi(n, C, t_1(1^n, C)) \vee \ldots \vee \varphi(n, C, t_k(1^n, C))\big).$$

(Note that each term t_i takes 1^n instead of n as input (in addition to C), since $\forall n \in \mathsf{Log}$ is a shorthand for $\forall N\ \forall n = |N|$). Each term t_i is a finite composition of PV functions and of the function symbol f. Consequently, over the standard model $\mathbb{N}$, each $t_i^{\mathbb{N}}$ can be computed by a polynomial-time Turing machine M_i with an f-oracle (see, e.g., [LO23] for a similar argument).

By the soundness of T over the standard model $\mathbb{N}$, for every choice of the input length n and of the circuit C, at least one term $t_i^{\mathbb{N}}$ (i.e. the corresponding oracle Turing machine M_i) correctly solves the refutation problem of Φ on input C. In other words, for every n and every n-input computation device C, there is some i such that $M_i(1^n, C)$ with oracle access to f (a fixed but arbitrary black-box that finds a collision in a given circuit D) runs in polynomial time and outputs an x such that $\varphi(n, C, x)$ is true, i.e., C makes a mistake on the input x. Note that the predicate $\varphi(n, C, x)$ can be decided in polynomial time, since φ is a quantifier-free formula in the language of PV. Consequently, we can efficiently check if a proposed solution x is correct. Since there are finitely many machines M_i, this provides a polynomial-time Turing reduction from the refutation problem of Φ to the complete problem of PWPP. □

This means that our equivalence result (Theorem 3.11) involving WPHP(PV) and complexity lower bounds can be directly translated into the PWPP-completeness of the refutation problems of these lower bounds, which include communication lower bounds for EQ and the Singleton bound for error correcting codes.

We observe that this result also holds in the context of the equivalence class for WPHPWIT, where the corresponding subclass of TFNP will be the class with the complete problem LossyCode that was recently introduced by [Kor22].

Note that the converse of Theorem 6.12 is not necessarily true. Even if there is a polynomial-time Turing reduction between a refutation problem and the complete problem of PWPP, the proof of its correctness may not be formalizable in PV_1.

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach.* Cambridge University Press, 2009.

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in p. *Annals of mathematics*, pages 781–793, 2004.

[AW09] Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *Transactions on Computation Theory* (TOCT), 1(1), 2009.

[BFS86] László Babai, Peter Frankl, and Janos Simon. Complexity classes in communication complexity theory. In *Symposium on Foundations of Computer Science* (FOCS), pages 337–347, 1986.

[BGS75] Theodore P. Baker, John Gill, and Robert Solovay. Relativizatons of the P =? NP Question. *SIAM J. Comput.*, 4(4):431–442, 1975.

[BKKK20] Sam Buss, Valentine Kabanets, Antonina Kolokolova, and Michal Koucký. Expander construction in VNC^1. *Annals of Pure and Applied Logic*, 171(7):102796, 2020.

[BKT14] Samuel R. Buss, Leszek A. Kołodziejczyk, and Neil Thapen. Fragments of approximate counting. *Journal of Symbolic Logic*, 79(2):496–525, 2014.

[Bus86] Samuel R. Buss. *Bounded Arithmetic.* Bibliopolis, 1986.

[Bus98] Samuel R. Buss. *Handbook of Proof Theory.* Elsevier, 1998.

[CHO$^+$22] Lijie Chen, Shuichi Hirahara, Igor C. Oliveira, Ján Pich, Ninad Rajgopal, and Rahul Santhanam. Beyond natural proofs: Hardness magnification and locality. *J. ACM*, 69(4):25:1–25:49, 2022.

[CHR23] Lijie Chen, Shuichi Hirahara, and Hanlin Ren. Symmetric exponential time requires near-maximum circuit size. *Electronic Colloquium on Computational Complexity* (ECCC), TR:23:144, 2023.

[CN10] Stephen A. Cook and Phuong Nguyen. *Logical Foundations of Proof Complexity.* Cambridge University Press, 2010.

[Cob65] Alan Cobham. The intrinsic computational difficulty of functions. *Proc. Logic, Methodology and Philosophy of Science*, pages 24–30, 1965.

[Coh63] Paul J. Cohen. The independence of the continuum hypothesis. *Proceedings of the National Academy of Sciences*, 50(6):1143–1148, 1963.

[Coo75] Stephen A. Cook. Feasibly constructive proofs and the propositional calculus (preliminary version). In *Symposium on Theory of Computing* (STOC), pages 83–97, 1975.

[Dam89] Ivan Bjerre Damgård. A design principle for hash functions. In *Conference on the Theory and Application of Cryptology*, pages 416–427. Springer, 1989.

[DM22] Damir D. Dzhafarov and Carl Mummert. *Reverse mathematics: problems, reductions, and proofs.* Springer Nature, 2022.

[Fri75] Harvey Friedman. Some systems of second order arithmetic and their use. In *Proceedings of the International Congress of Mathematicians*, volume 1, pages 235–242, 1975.

[FS16] Lance Fortnow and Rahul Santhanam. New non-uniform lower bounds for uniform classes. In *Conference on Computational Complexity* (CCC), pages 19:1–19:14, 2016.

[FS17] Lance Fortnow and Rahul Santhanam. Robust simulations and significant separations. *Inf. Comput.*, 256:149–159, 2017.

[Gay22] Azza Gaysin. Proof complexity of CSP. *arXiv:2201.00913*, 2022.

[Göd38] Kurt Gödel. The consistency of the axiom of choice and of the generalized continuum-hypothesis. *Proceedings of the National Academy of Sciences*, 24(12):556–557, 1938.

[ILW23] Rahul Ilango, Jiatu Li, and R. Ryan Williams. Indistinguishability obfuscation, range avoidance, and bounded arithmetic. In *Symposium on Theory of Computing* (STOC), pages 1076–1089. ACM, 2023.

[IW97] Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Symposium on the Theory of Computing* (STOC), pages 220–229, 1997.

[Jeř04] Emil Jeřábek. Dual weak pigeonhole principle, boolean complexity, and derandomization. *Annals of Pure and Applied Logic*, 129(1-3):1–37, 2004.

[Jeř05] Emil Jeřábek. *Weak pigeonhole principle and randomized computation.* PhD thesis, Charles University in Prague, 2005.

[Jeř06] Emil Jeřábek. The strength of sharply bounded induction. *Mathematical Logic Quarterly*, 52(6):613–624, 2006.

[Jeř07a] Emil Jeřábek. Approximate counting in bounded arithmetic. *Journal of Symbolic Logic*, 72(3):959–993, 2007.

[Jeř07b] Emil Jeřábek. On independence of variants of the weak pigeonhole principle. *J. Log. Comput.*, 17(3):587–604, 2007.

[Jeř16] Emil Jeřábek. Integer factoring and modular square roots. *Journal of Computer and System Sciences*, 82(2):380–394, 2016.

[KKMP21] Robert Kleinberg, Oliver Korten, Daniel Mitropolsky, and Christos H. Papadimitriou. Total functions in the polynomial hierarchy. In *Innovations in Theoretical Computer Science Conference* (ITCS), pages 44:1–44:18, 2021.

[KN97] Eyal Kushilevitz and Noam Nisan. *Communication complexity.* Cambridge University Press, 1997.

[KO17] Jan Krajíček and Igor C. Oliveira. Unprovability of circuit upper bounds in Cook's theory PV. *Logical Methods in Computer Science*, 13(1), 2017.

[Koh08] Ulrich Kohlenbach. *Applied Proof Theory - Proof Interpretations and their Use in Mathematics.* Springer Monographs in Mathematics. Springer, 2008.

[Kor21] Oliver Korten. The hardest explicit construction. In *Symposium on Foundations of Computer Science* (FOCS), pages 433–444, 2021.

[Kor22] Oliver Korten. Derandomization from time-space tradeoffs. In Shachar Lovett, editor, *Computational Complexity Conference* (CCC), pages 37:1–37:26, 2022.

[KPT91] Jan Krajíček, Pavel Pudlák, and Gaisi Takeuti. Bounded arithmetic and the polynomial hierarchy. *Annals of Pure and Applied Logic*, 52(1-2):143–153, 1991.

[Kra95] Jan Krajíček. *Bounded Arithmetic, Propositional Logic, and Complexity Theory.* Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1995.

[Kra01] Jan Krajíček. On the weak pigeonhole principle. *Fundamenta Mathematicae*, 1(170):123–140, 2001.

[Kra11] Jan Krajícek. On the proof complexity of the Nisan-Wigderson generator based on a hard NP ∩ coNP function. *Journal of Mathematical Logic*, 11(1), 2011.

[Kra19] Jan Krajíček. *Proof Complexity.* Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2019.

[Kra24] Jan Krajícek. Proof complexity generators. Monograph (In Progress), 2024.

[KT08] Leszek Aleksander Kolodziejczyk and Neil Thapen. The polynomial and linear hierarchies in models where the weak pigeonhole principle fails. *J. Symb. Log.*, 73(2):578–592, 2008.

[LC11] Dai Tri Man Le and Stephen A. Cook. Formalizing randomized matching algorithms. *Log. Methods Comput. Sci.*, 8(3), 2011.

[Li23] Zeyong Li. Symmetric exponential time requires near-maximum circuit size: Simplified, truly uniform. *Electron. Colloquium Comput. Complex.*, TR23-156, 2023.

[LO23] Jiatu Li and Igor C. Oliveira. Unprovability of strong complexity lower bounds in bounded arithmetic. In *Symposium on Theory of Computing* (STOC), 2023.

[Lê14] Dai Tri Man Lê. *Bounded Arithmetic and Formalizing Probabilistic Proofs.* PhD thesis, University of Toronto, 2014.

[Maa84] Wolfgang Maass. Quadratic lower bounds for deterministic and nondeterministic one-tape turing machines. In *Symposium on Theory of Computing* (STOC), pages 401–408. ACM, 1984.

[Mer89] Ralph C Merkle. One way hash functions and des. In *Conference on the Theory and Application of Cryptology*, pages 428–446. Springer, 1989.

[MP20] Moritz Müller and Ján Pich. Feasibly constructive proofs of succinct weak circuit lower bounds. *Annals of Pure and Applied Logic*, 171(2), 2020.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

[Oja04] Kerry Ojakian. *Combinatorics in Bounded Arithmetic.* PhD thesis, Carnegie Mellon University, 2004.

[Par71] Rohit Parikh. Existence and feasibility in arithmetic. *Journal of Symbolic Logic*, 36(3):494–508, 1971.

[Pic14] Ján Pich. *Complexity Theory in Feasible Mathematics.* PhD thesis, Charles University in Prague, 2014.

[Pic15a] Ján Pich. Circuit lower bounds in bounded arithmetics. *Annals of Pure and Applied Logic*, 166(1):29–45, 2015.

[Pic15b] Ján Pich. Logical strength of complexity theory and a formalization of the PCP theorem in bounded arithmetic. *Logical Methods in Computer Science*, 11(2), 2015.

[PS21] Ján Pich and Rahul Santhanam. Strong co-nondeterministic lower bounds for NP cannot be proved feasibly. In *Symposium on Theory of Computing* (STOC), pages 223–233, 2021.

[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.

[Raz95a] Alexander A. Razborov. Bounded arithmetic and lower bounds in boolean complexity. In P. Clote and J. Remmel, editors, *Feasible Mathematics II*, pages 344—386. Birkhäuser, 1995.

[Raz95b] Alexander A Razborov. Unprovability of lower bounds on circuit size in certain fragments of bounded arithmetic. *Izvestiya: mathematics*, 59(1):205, 1995.

[RR97] Alexander A. Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, 1997.

[Sim09] Stephen George Simpson. *Subsystems of second order arithmetic*, volume 1. Cambridge University Press, 2009.

[Sti20] John Stillwell. *Reverse mathematics*. Springer, 2020.

[TC21] Iddo Tzameret and Stephen A. Cook. Uniform, integral, and feasible proofs for the determinant identities. *J. ACM*, 68(2):12:1–12:80, 2021.

[Tha02] Neil Thapen. *The weak pigeonhole principle in models of bounded arithmetic*. PhD thesis, University of Oxford, 2002.

[TS00] A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2 edition, 2000.

[TV07] Luca Trevisan and Salil P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Comput. Complex.*, 16(4):331–364, 2007.

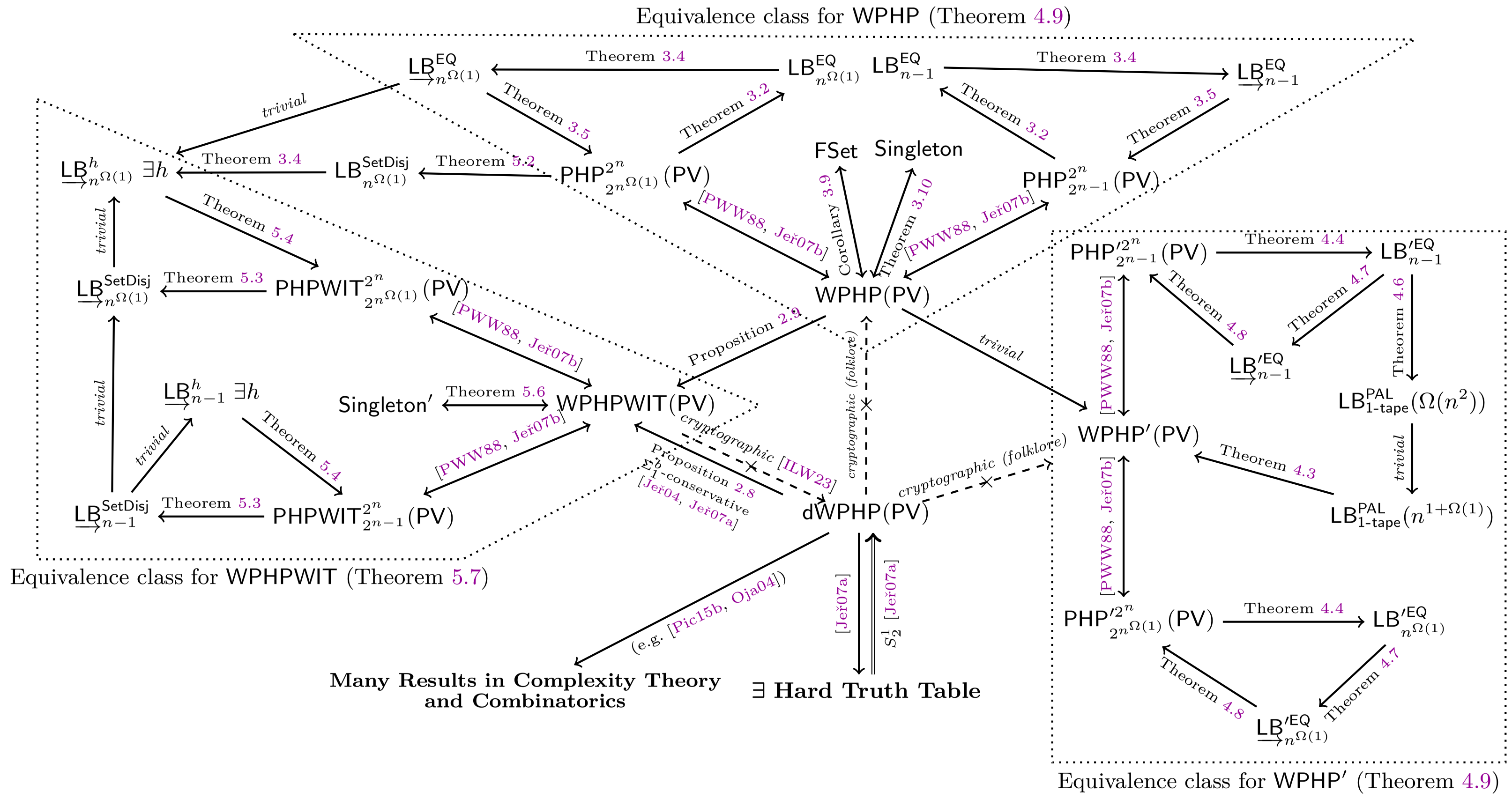
Equivalence class for WPHP (Theorem 4.9)
Equivalence class for WPHPWIT (Theorem 5.7)
Equivalence class for WPHP′ (Theorem 4.9)
$\mathsf{LB}^{\mathsf{EQ}}_{\to n^{\Omega(1)}}$
$\mathsf{LB}^{\mathsf{EQ}}_{n^{\Omega(1)}}$
$\mathsf{LB}^{\mathsf{EQ}}_{n-1}$
$\mathsf{LB}^{\mathsf{EQ}}_{\to n-1}$
$\mathsf{LB}^{h}_{\to n^{\Omega(1)}}\ \exists h$
$\mathsf{LB}^{\mathsf{SetDisj}}_{n^{\Omega(1)}}$
$\mathsf{PHP}^{2^n}_{2^{n^{\Omega(1)}}}(\mathsf{PV})$
$\mathsf{PHP}^{2^n}_{2^n-1}(\mathsf{PV})$
FSet
Singleton
$\mathsf{LB}^{\mathsf{SetDisj}}_{\to n^{\Omega(1)}}$
$\mathsf{PHPWIT}^{2^n}_{2^{n^{\Omega(1)}}}(\mathsf{PV})$
WPHP(PV)
$\mathsf{LB}^{h}_{\to n-1}\ \exists h$
Singleton′
WPHPWIT(PV)
$\mathsf{LB}^{\mathsf{SetDisj}}_{\to n-1}$
$\mathsf{PHPWIT}^{2^n}_{2^n-1}(\mathsf{PV})$
dWPHP(PV)
Many Results in Complexity Theory and Combinatorics
∃ Hard Truth Table
$\mathsf{PHP}'^{2^n}_{2^n-1}(\mathsf{PV})$
$\mathsf{LB}'^{\mathsf{EQ}}_{n-1}$
$\mathsf{LB}'^{\mathsf{EQ}}_{\to n-1}$
$\mathsf{LB}^{\mathsf{PAL}}_{\text{1-tape}}(\Omega(n^2))$
WPHP′(PV)
$\mathsf{LB}^{\mathsf{PAL}}_{\text{1-tape}}(n^{1+\Omega(1)})$
$\mathsf{PHP}'^{2^n}_{2^{n^{\Omega(1)}}}(\mathsf{PV})$
$\mathsf{LB}'^{\mathsf{EQ}}_{n^{\Omega(1)}}$
$\mathsf{LB}'^{\mathsf{EQ}}_{\to n^{\Omega(1)}}$
Theorem 3.4
Theorem 3.5
Theorem 3.2
Theorem 3.2
Theorem 3.4
Theorem 3.5
trivial
Theorem 3.4
Theorem 3.2
Theorem 5.4
Corollary 3.9
Theorem 3.10
[PWW88, Jeřábek07b]
[PWW88, Jeřábek07b]
trivial
Theorem 5.3
[PWW88, Jeřábek07b]
Proposition 2.9
trivial
trivial
trivial
Theorem 5.4
Theorem 5.6
[PWW88, Jeřábek07b]
cryptographic [ILW23]
Proposition 2.8
Σ^b_1-conservative
[Jeř04, Jeř07a]
cryptographic (folklore)
cryptographic (folklore)
Theorem 5.3
(e.g. [Pic15b, Oja04])
[Jeř07a]
S^1_2 [Jeř07a]
Theorem 4.4
Theorem 4.8
Theorem 4.7
Theorem 4.6
[PWW88, Jeř07b]
trivial
Theorem 4.3
[PWW88, Jeř07b]
Theorem 4.4
Theorem 4.8
Theorem 4.7

Figure 1: Landscape of combinatorial principles and lower bounds. (In this diagram, $\Pi \to \Delta$ denotes $\Pi \vdash_{\mathsf{PV}_1} \Delta$, $\Pi \Rightarrow \Delta$ denotes $\Pi \vdash_{S^1_2(\mathsf{PV})} \Delta$, and $\Pi \dashrightarrow\!\!\!\!\times\ \Delta$ denotes $\Pi \nvdash_{\mathsf{PV}_1} \Delta$ (under plausible assumptions).