Approximate counting and NP search problems

Leszek Aleksander Kołodziejczyk* and Neil Thapen†

November 29, 2021

Abstract

We study a new class of NP search problems, those which can be proved total using standard combinatorial reasoning based on approximate counting. Our model for this kind of reasoning is the bounded arithmetic theory APC_2 of [Jeřábek 2009]. In particular, the Ramsey and weak pigeonhole search problems lie in the new class. We give a purely computational characterization of this class and show that, relative to an oracle, it does not contain the problem CPLS, a strengthening of PLS.

As CPLS is provably total in the theory T^2_2, this shows that APC_2 does not prove every $\forall\Sigma^b_1$ sentence which is provable in bounded arithmetic. This answers the question posed in [Buss, Kołodziejczyk, Thapen 2014] and represents some progress in the programme of separating the levels of the bounded arithmetic hierarchy by low-complexity sentences.

Our main technical tool is an extension of the "fixing lemma" from [Pudlák, Thapen 2017], a form of switching lemma, which we use to show that a random partial oracle from a certain distribution will, with high probability, determine an entire computation of a P^{NP} oracle machine. The introduction to the paper is intended to make the statements and context of the results accessible to someone unfamiliar with NP search problems or with bounded arithmetic.

1 Introduction

An *NP search problem* is specified by a polynomial-time relation $R(x, y)$ and a polynomial $p(x)$. Given an input x, a solution to the problem is any y such that $R(x, y)$ holds and $|y| < p(|x|)$ (where $|x|$ is the length of a string x). We only consider *total* problems, where a solution is guaranteed to exist for all x. The class of all such problems is called TFNP, standing for *total functional* NP [31].

Subclasses of TFNP are sometimes described as consisting of all search problems which can be proved to be total by some particular combinatorial lemma or style of argument [31, 35]. For example, the class PPA "is based on the lemma that every graph has an even number of odd-degree nodes" [3]. Often, the particular lemma or argument can be represented by a specific axiomatic theory. In

[*] Institute of Mathematics, University of Warsaw, lak@mimuw.edu.pl

[†] Institute of Mathematics of the Czech Academy of Sciences, thapen@math.cas.cz

this paper we study the class, which we call APPROX, of problems that can be proved total using reasoning based on *approximate counting*. Our model of this kind of reasoning is the theory APC_2 developed by Jeřábek in [20, 21], which provides machinery to count the size of a set well enough to distinguish between sets of size a and $(1 + \varepsilon)a$, for a given in binary (but not between sets of size a and $a + 1$), and to formalize a certain amount of induction in this language. In this way it can carry out the standard proofs of, for example, the finite Ramsey theorem and the tournament principle [21]. We give a non-logical characterization of APPROX, as the class of TFNP problems reducible in a certain sense to a version of the weak pigeonhole principle for P^{NP} functions. Examples of problems in the class are natural problems associated with the finite Ramsey theorem, the usual weak pigeonhole principle, and the ordering principle.

Our main result is that, in the relativized setting, a search problem known as CPLS [28] is not in APPROX. Here CPLS is a natural strengthening of a complete problem for the class PLS of search problems (*polynomial local search*, see Section 1.2 below).

This work is mainly motivated by an open problem in logic. Our result answers a question about a hierarchy of theories collectively known as bounded arithmetic. For each $i \in \mathbb{N}$, the theory T^i_2 is axiomatized by induction for formulas at level i in the polynomial hierarchy. It is a long-standing open problem whether the theories T^i_2 can be separated by sentences expressing that various NP search problems are total – known as $\forall\Sigma^b_1$ sentences. In other words: does the class of provably total NP search problems get strictly bigger as i increases? This is open for $i \geq 2$.

APC_2 lies between T^1_2 and T^3_2. In [11] we pointed out that the NP search problems typically used in arguments separating T^1_2 from T^i_2 for $i \geq 2$ could be proved total using approximate counting. This led us to state the following open problem, which is an important special case of the more general one: is there any i such that T^i_2 proves the totality of more NP search problems than APC_2 does?

Our result in this paper implies that the answer is yes. The totality of CPLS, which is provable in T^2_2, is not provable in APC_2, and thus T^3_2 proves strictly more NP search problems total than APC_2 does. This makes APC_2 one of the strongest natural theories that has been separated from theories higher up in the bounded arithmetic hierarchy – in fact, from T^i_2 for the lowest possible i – in terms of NP search problems. Intuitively speaking, the conclusion is that the power of T^2_2, $T^3_2, \ldots$ to prove many NP search problems total is based on more than just a limited ability to count.

While our motivation is from logic, an important part of the methods we use are complexity-theoretic, and may be independently interesting to complexity theorists. Our main technical tool is the "fixing lemma" from [42]. This is related to the *switching lemma* of Håstad [19], which is used in complexity to separate depth $d + 1$ circuits from depth d circuits. The fixing lemma is a simplified, but more widely applicable, version of this result. It shows that a random partial assignment can, with high probability, determine the value of a CNF. We strengthen it slightly, to show that a random partial oracle can determine an entire computation of a P^{NP} oracle machine. The proof of our version is almost identical, and many definitions are identical, to what appears in [42]. In the more technical parts of Sections 4-6 will assume the reader has access to that paper.

In the rest of this introductory section we give an overview of bounded arithmetic and the theory APC_2, describe the structure of TFNP from this point of view, and outline how we handle relativization and reductions. The paper is then structured as follows.

Section 2. We define CPLS and our search-problem class APPROX, formally state our main Theorems 12 and 13, obtain some corollaries, and give an outline of the proofs.

Section 3. We prove Theorem 12, that the class APPROX captures the $\forall\Sigma_1^b$ consequences of APC_2. This section contains the technical work in logic.

Section 4. We state and prove our version of the fixing lemma.

Section 5. We use the fixing lemma to show that a P^{NP} computation is determined by a random partial oracle, and derive Theorem 13, that CPLS is not in APPROX.

Section 6. We briefly sketch an alternative way to prove our main result about bounded arithmetic, that APC_2 does not prove the totality of CPLS, by going through propositional proof complexity rather than NP search problems.

Section 7. We mention some open problems.

Acknowledgements. The first author was partially supported first by grant 2013/09/B/ST1/04390 and then by grant 2017/27/B/ST1/01951 of the National Science Centre, Poland. The second author was partially supported by GA ČR project 19-05497S. Part of this research was carried out during the first author's visit to Prague in 2017, funded by ERC grant 339691. The Institute of Mathematics of the Czech Academy of Sciences is supported by RVO:67985840.

We are grateful to Pavel Pudlák and Pavel Hubáček for discussions about this work.

1.1 Bounded arithmetic

Fix a language L_{PV} containing a symbol for every function or relation computed by a polynomial-time machine. Then a total NP search problem can be identified with a true L_{PV} sentence of the form $\forall x\, \exists y < t(x)\, R(x,y)$, where R is a polynomial-time relation, t is a polynomial-time function, and x and y range over natural numbers written in binary notation. Let T be any sound theory. The set of such sentences provable in T then defines a class of search problems. For the class to have some reasonable properties, T should not be too weak; and to get classes of the kind usually studied in complexity theory, it should not be too strong.

Natural examples of suitable theories T come from bounded arithmetic, which has close ties to computational complexity. For the purposes of this paper, we will take such theories to be given by a *base theory* fixing some basic properties of the symbols of L_{PV}, together with one or more axiom schemes that allow us to do stronger kinds of reasoning, typically induction. All axioms are universal closures of *bounded formulas*, that is, formulas in which all quantifiers appear in the form $\forall x{<}t$ or $\exists x{<}t$.

In more detail, a PV *formula* is a quantifier-free formula of L_{PV}. A Σ^b_i *formula* is one of the form

$$\exists x_1{<}t_1(\bar{z})\,\forall x_2{<}t_2(\bar{z},x_1)\ \ldots\ \varphi(\bar{z},\bar{x})$$

where φ is a PV formula, the bounds t_j are L_{PV}-terms, quantifiers may appear in alternating $\exists$ and $\forall$ blocks, and there are at most i blocks. Such formulas define precisely the Σ^p_i relations, that is, those at level i in the polynomial hiearchy. The Π^b_i formulas are defined dually. The *universal closure* of a formula $\varphi(\bar{z})$ is the sentence $\forall\bar{z}\,\varphi(\bar{z})$. Given a class of formulas Γ, we write $\forall\Gamma$ for the set of universal closures of formulas from Γ. Thus, for instance, a $\forall$PV sentence states that some polynomial-time computable property holds for all inputs.

We will consider two base theories, both containing only $\forall$PV sentences. The first and more usual one is the theory PV, which comes from Cobham's characterization of the polynomial-time functions as a function algebra [16, 15]. We will not define PV here, as the details are not important, but it can be thought of as a minimal theory in which all polynomial-time functions are well-behaved.

The second, stronger but simpler theory, which we denote $\forall\mathrm{PV}(\mathbb{N})$, consists simply of all $\forall$PV sentences which are true under the standard interpretation in $\mathbb{N}$. This is simpler to understand than PV because there is no list of axioms to keep in mind, and also works more naturally for defining NP search problems. Our results translate easily between the two, and a reader unfamiliar with bounded arithmetic will not go very wrong by reading PV as $\forall\mathrm{PV}(\mathbb{N})$ throughout – see Subsection 1.3.

The important family of theories T^i_2, for $i \geq 0$, is defined as

$$T^i_2 := \mathrm{PV} + \Sigma^b_i\text{-IND}$$

where Σ^b_i-IND is the usual induction scheme for Σ^b_i formulas with parameters. PV already proves induction for quantifier-free formulas (as that sort of induction can be witnessed by polynomial-time binary search) so in this setting T^0_2 is the same as PV. We write T_2 for the union of this family.

We can now state a fundamental theorem. By the Σ^b_i*-definable* functions of a theory we mean the functions with Σ^b_i graphs which the theory proves are total.

Theorem 1 ([9]). *For $i \geq 0$, the Σ^b_{i+1}-definable functions of T^i_2 are precisely the $\mathrm{P}^{\Sigma^p_i}$ functions, that is, those that are polynomial-time computable with an oracle from level i of the polynomial hierarchy.*

We will also use a related family of theories S^i_2, for $i \geq 1$, which is defined by replacing the Σ^b_i-IND scheme in T^i_2 with the apparently weaker scheme Σ^b_i-LIND in which inductions can only run for polynomially many steps (in the binary length of a parameter). We have $T_2^i \subseteq S_2^{i+1} \subseteq T_2^{i+1}$ [9]. Theorem 1 remains true if T_2^i is replaced by S_2^{i+1} and/or the base theory PV is replaced by $\forall\mathrm{PV}(\mathbb{N})$.

In addition to being related to computational complexity by Theorem 1, bounded arithmetic is a natural environment in which to ask questions about the provability or consistency of theorems or conjectures from complexity theory. For recent examples see [38, 34].

We now make the above definitions slightly more complicated. As in complexity theory, we typically cannot expect to show that two theories of bounded arithmetic are distinct without either making some extra assumption or working relative to some oracle. We will use oracles. We redefine L_{PV} to include a unary relation symbol α standing for "an arbitrary oracle", and function and relation symbols for all polynomial-time machines with oracle access to α. Other formula classes and theories are redefined to use this extended language. In particular, $\forall\mathrm{PV}(\mathbb{N})$ becomes the set of $\forall$PV sentences which are true in $\langle \mathbb{N}, A\rangle$ for *every* oracle A interpreting the symbol α. Strictly speaking, we should change the names to $\mathrm{PV}(\alpha)$, $\Sigma_i^b(\alpha)$, $T_2^i(\alpha)$ etc. However, since we never use the unrelativized versions, we simplify notation by keeping the old names. The results mentioned above still hold.

An open problem. By adapting oracle separation results for the polynomial hierarchy, it has been shown that the strength of the (relativized) theories T_2^i increases strictly with i: for each i there is a $\forall\Sigma_{i+1}^b$ sentence provable in T_2^{i+1} but not in T_2^i [13]. A pressing open question in proof complexity[1] is whether this remains true if we measure the strength of theories only by their $\forall\Sigma_k^b$ consequences for some fixed k, in particular for $k = 1$.

We write $\forall\Sigma_k^b(T)$ for the $\forall\Sigma_k^b$ consequences of a theory T. In particular $\forall\Sigma_1^b(T)$ is a class of total NP search problems (as long as T is sound). From [25, 14] we know that

$$\forall\Sigma_1^b(\mathrm{PV}) \subsetneq \forall\Sigma_1^b(T_2^1) \subsetneq \forall\Sigma_1^b(T_2^2)$$

and from [14, 43] we know that for any $i, k \geq 1$,

$$\text{if } \forall\Sigma_k^b(T_2^i) = \forall\Sigma_k^b(T_2^{i+1}) \text{ then } \forall\Sigma_k^b(T_2^i) = \forall\Sigma_k^b(T_2).$$

The following is open for $k \leq 2$:

$$\text{does } \ \forall\Sigma_k^b(T_2^2) = \forall\Sigma_k^b(T_2) \ ? \tag{1}$$

The answer is expected to be negative, even for $k = 0$, by analogy with the Π_1 separation between $\mathrm{I}\Sigma_i$ and $\mathrm{I}\Sigma_{i+1}$ given by the second incompleteness theorem. The case $k = 1$ seems to be particularly approachable, as classes $\forall\Sigma_1^b(T)$ have a natural computational interpretation in terms of NP search problems.

[1] This is essentially equivalent to a question in propositional proof complexity about separating bounded-depth Frege systems by formulas of fixed depth, and in particular finding a family of small-width CNF's which have short refutations in bounded-depth Frege but require long refutations in Res(log).

Approximate counting. Jeřábek [20, 21] developed a bounded arithmetic theory for approximate counting. Following [11] we call this theory[2] APC_2 and define it as T^1_2 together with the *surjective weak pigeonhole principle* (sWPHP) for P^{NP} functions, which asserts that no such function can be a surjection from n to $2n$, for any $n > 0$. APC_2 can formalize many arguments in finite combinatorics that use approximate counting, such as the standard proofs of the finite Ramsey theorem and the tournament principle, as well as some probabilistic reasoning. It lies between T^1_2 and T^3_2 in strength, as this instance of the weak pigeonhole principle is provable in T^3_2.

In [11] we asked the analogue of question (1) for APC_2 in place of T^2_2. More specifically:

$$\text{does } \ \forall\Sigma^b_1(\mathrm{APC}_2) = \forall\Sigma^b_1(T_2) \ ? \tag{2}$$

We expected the answer to be "no", but the opposite did not seem completely implausible. Approximate counting is a powerful tool in finite combinatorics, and typical combinatorially natural examples of hard $\forall\Sigma^b_1$ statements that have been used to separate T^1_2 from T_2 were known to be provable in APC_2 [11]. Moreover it was shown in [12], by formalizing Toda's theorem, that all of bounded arithmetic collapses to the analogue of APC_2 if we add a parity quantifier to the language.

Both [11] and later [2] showed unprovability results for various natural subtheories of APC_2, but these fell well short of answering (2). In fact, they were obtained using a $\forall\Sigma^b_1$ sentence that is actually provable in APC_2.

1.2 TFNP

As already discussed, in our language a total NP search problem is simply a true $\forall\Sigma^b_1$ sentence, that is, one of the form $\forall x\, \exists y{<}t(x)\, R(x,y)$ where $R(x,y)$ is a PV formula and t is an L_{PV}-term. This represents the search-task of finding a witness y, given x. We will often assume that the bound $y < t(x)$ is implicit in $R(x,y)$, and we will usually write $R(x,y)$ or just R as a name for the search problem.

As before, polynomial-time is defined relative to an oracle symbol α, and we will occasionally use notation like $R(x,y;\alpha)$ to emphasize the specific oracle being used. The oracle leads to a slight complication in what we mean when we say a search problem is total: $\forall x\, \exists y{<}t(x)\, R(x,y;\alpha)$ must be true in $\langle \mathbb{N}, A\rangle$ for *every* oracle A interpreting α. This behaviour is essentially the same as what is called a total type-2 NP search problem in e.g. [3, 10].

We define two important notions of reducibility between search problems Q and R. We will introduce one more in Subsection 2.2.

Definition 2. $Q(x,y)$ is *polynomial-time many-one reducible*, or simply *reducible*, to $R(x',y')$, written $Q \leq R$, if there are polynomial-time functions f

[2] Our definition is slightly different from Jeřábek's in [21], which uses a variant of the surjective weak pigeonhole principle with a smaller difference between domain and range. However, the theories prove the same $\forall\Sigma^b_2$ statements, which is all that matters for this paper.

and g and a polynomial-time relation P (all of which may query the oracle α) such that

$$R(f(x), y'; P(x, \cdot)) \to Q(x, g(x, y'); \alpha)$$

holds for all x, y' and α, where $P(x, \cdot)$ represents the oracle $\{z : P(x, z)\}$. Two problems are *equivalent* if they are reducible to one another.

Definition 3. $Q(x, y)$ is *polynomial-time Turing-reducible* to $R(x', y')$ if there is a polynomial-time relation P and a polynomial-time oracle machine M which, on input x, makes a series of (adaptive) queries to $R(x', y'; P(\langle x, x'\rangle, \cdot))$. If all replies are correct, then M outputs some y such that $Q(x, y; \alpha)$.

We are interested in search problem classes corresponding to bounded arithmetic theories, in the sense that the class captures the search problems proved total by the theory (but see Section 1.3 below). There has been a research programme, motivated partly by the logical separation question discussed above, to characterize these classes.

- PV corresponds to FP, the class of search problems which can be solved in deterministic polynomial time [16, 9].

- T^1_2 corresponds to PLS [23, 13]. A PLS problem is given by polynomial-time *neighbourhood* and *cost* functions N_x and C_x and *domain* predicate F_x, such that $0 \in F_x$ and if $y \in F_x$, then $|y| \leq |x|^k$ for some fixed k. A solution to an instance x is any $y \in F_x$ such that either $N_x(y) \notin F_x$ or $C_x(N_x(y)) \geq C_x(y)$. Such a y exists because costs cannot decrease indefinitely. A complete problem for the class is to find a local minimum for a function on a bounded-degree graph.

- T^2_2 corresponds to CPLS [28], a generalization of PLS described below.

- For $k \geq 1$, T^k_2 corresponds to a class GI_k defined by the k-turn game induction principle [43] (see also [5, 6]). Roughly speaking, a complete problem for GI_k is: given a sequence of k-round 2-player games, winning strategies for opposite players in the first and last games, and reductions between neighbouring games in the sequence, find an error in one of the strategies or one of the reductions. Equivalent search problems include further generalizations of PLS and principles about feasible Nash equilibria [41], and LLI_k, the k-round linear local improvement principle [24].

The theory T^i_2 is equivalent to a natural formalization of "every $\mathrm{P}^{\Sigma^p_i}$ machine has a computation on every input", essentially by Theorem 1. The search problems above can thus be thought of as the projections onto TFNP of increasingly strong computation models. This can be taken further: there are two "second-order" bounded arithmetic theories, U^1_2 and V^1_2, which are equivalent (with respect to their $\forall\Sigma^b_1$ consequences) to similar statements about computations of, respectively, PSPACE and EXPTIME machines [9, 24]. In terms of NP search problems, from [24, 7] we have:

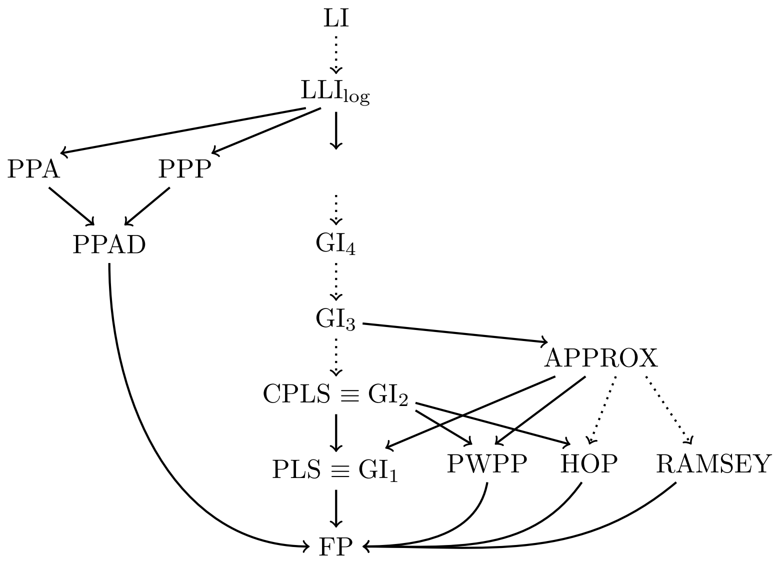

Figure 1: A diagram showing some inclusions between selected classes in TFNP, including our results. Some classes are named by their complete problems. Solid arrows are strict inclusions, relative to some oracle. Dotted arrows are inclusions not known to be strict. For separations among PPA, PPP, PPAD, PLS see [3, 32, 8, 10]. For other references see Sections 1.2 and 2.2. CPLS separates GI$_3$ from APPROX and also separates GI$_2$ from HOP and PWPP. WEAKPIGEON separates APPROX from PLS. HOP separates APPROX from PWPP. Solid arrows from PPP to HOP [11] and from PPP to PWPP have been omitted for readability.

- U_2^1 corresponds to LLI$_{\log}$, the linear local improvement principle with polynomially many rounds.[3]

- V_2^1 corresponds to LI, the local improvement principle.

One can think of the classes $\mathrm{FP} \subseteq \mathrm{GI}_1 \subseteq \mathrm{GI}_2 \subseteq \cdots \subseteq \mathrm{LLI}_{\log} \subseteq \mathrm{LI}$ as forming a backbone for TFNP, arising natural from the hierarchy of bounded arithmetic theories or of computation models – see Figure 1. This could be extended even beyond bounded arithmetic, to say $\forall\Sigma_1^b(\mathrm{PA})$ or $\forall\Sigma_1^b(\mathrm{ZFC})$ [4, 44], or by using potentially stronger systems of reasoning, as in [17].

However, experience suggests[4] that it is difficult to find any natural "combinatorial" NP search problem that is not already provably total in U_2^1, and thus reducible to LLI$_{\log}$. In particular, U_2^1 is strong enough to formalize the counting arguments needed to prove the totality of complete problems for the well-known

[3] Unfortunately LLI$_{\log}$ is rather complicated to describe or use. The authors believe that it is equivalent to the simpler game induction principle GI, which is like the principle GI$_k$ of [43] but with polynomially many rounds.

[4] A related issue in propositional proof complexity is the difficulty of finding candidates for separating the Frege and extended Frege proof systems [1].

classes PPA and PPP introduced in [35]. Thus, all problems in those classes are reducible to $\mathrm{LLI}_{\log}$.

On the other hand the bijective pigeonhole principle, called OntoPIGEON in the search problem literature, is a complete problem for the class PPAD [8] which is contained in both PPA and PPP. Standard proof-complexity lower bound arguments for the pigeonhole principle [27, 39] show that this problem is not provably total in any theory T^k_2 and not reducible to any GI_k.

Finally let us mention the search problem class PWPP [22], based on the injective weak pigeonhole principle search problem WEAKPIGEON, which is contained in GI_2 and PPP but not in PLS [30, 25]. We will discuss this and two other search problems, RAMSEY and HOP, in Section 2.

It is open whether the hierarchy $\mathrm{GI}_2 \subseteq \mathrm{GI}_3 \subseteq \dots$ is strict. This is essentially the same problem as the separation of $\forall\Sigma^b_1(T^{i+1}_2)$ from $\forall\Sigma^b_1(T^i_2)$ discussed above.

1.3 True and provable reductions

In the previous section, we did not explicitly say what it means for a search problem class to correspond to a theory T. The obvious meaning, that the class is precisely $\forall\Sigma^b_1(T)$, potentially has a problem. Namely, such a class does not have the desirable property of being closed under many-one reductions, unless the reductions work provably in T. There may even be two PV formulas R_1 and R_2 which "semantically" define the same relation on $\mathbb{N}$, and thus the same search problem by the usual complexity-theoretic definition, but are such that T proves that one is total but not the other.

There are two natural ways around this issue. One is to define our class as the closure of $\forall\Sigma^b_1(T)$ under many-one reductions. The other is to stick to theories T that contain the set $\forall\mathrm{PV}(\mathbb{N})$ of all true $\forall$PV sentences, and exploit the fact that the statement that a reduction works is such a sentence.

The next lemma shows that, for theories of the kind we consider, these two approaches have the same result. In this paper we prefer the second one.

Lemma 4 (folklore, see also [18])**.** *Let $Q(x, y)$ be an NP search problem. Let T be a bounded arithmetic theory containing* PV *and with axioms closed under substituting polynomial-time relations for oracles. Then (1) and (2) below are equivalent. If T contains S^1_2, then (3) is also equivalent.*

1. *Q is provably total in $T + \forall\mathrm{PV}(\mathbb{N})$.*

2. *$Q \leq R$ for some TFNP problem R provably total in T.*

3. *Q is Turing reducible to a TFNP problem R provably total in T.*

Proof. Suppose (1) holds. We have that $T + \forall z\, \varphi(z) \vdash \forall x\, \exists y\, Q(x, y)$ for some PV formula φ such that $\forall z\, \varphi(z) \in \forall\mathrm{PV}(\mathbb{N})$. Hence $T \vdash \forall x\, [\exists z\, \neg\varphi(z) \vee \exists y\, Q(x, y)]$. Since T is a bounded theory, by Parikh's theorem [36] we may add some term $t(x)$ bounding both existential quantifiers. Therefore $T \vdash \forall x\, \exists y < t(x)\, R(x, y)$ where $R(x, y)$ is the formula $\neg\varphi(y) \vee Q(x, y)$. Now R is an NP search problem, provably total in T, and Q is reducible to R in $\mathbb{N}$ using the identity function, since $\varphi(y)$ is true for every y and every oracle. Hence (2) holds.

Now suppose (2) holds. Then (3) is immediate. For (1), from the definition of a reduction, there are PV function symbols f, g and a relation symbol P such that, for every oracle A,

$$\langle \mathbb{N}, A\rangle \vDash \forall x\, \forall y' \,[R(f(x), y'; P(x, \cdot)) \to Q(x, g(x, y'); A)].$$

Now define the search problem $R^*(x', y'; \alpha) := R(x', y'; P(x, \cdot))$. Then $\forall$PV($\mathbb{N}$) proves $\forall x'\, \exists y'\, R^*(x', y') \to \forall x\, \exists y\, Q(x, y)$, and by the property of closure under subsitution for the oracle, T proves that R^* is total. Hence we have (1).

Lastly we show that (3) implies (1) under the stronger assumption. Turing reducibility means that there is a polynomial-time oracle machine M which, on input x, makes oracle queries to R and, if the replies are correct, outputs y such that $Q(x, y)$. Formally, for every A, $\langle \mathbb{N}, A\rangle \vDash \forall x\, \forall w\, \varphi(x, w)$ for a PV relation $\varphi(x, w)$ expressing that: if w is a computation of M on input x, and every oracle query x' in w has a reply y' in w such that $P(x', y')$, then $Q(x, \mathrm{output}(w))$. But T proves that, for all x, such a w exists, since Σ^b_1-LIND is enough to construct w query by query. Hence $T + \forall\mathrm{PV}(\mathbb{N}) \vdash \forall x\, \exists y\, Q(x, y)$. □

2 Main definitions and results

2.1 Coloured polynomial local search

We study a search problem introduced in [28]. We will need several results about it from [42], so we take the definition verbatim from there.

Let a, b, c be parameters. Consider a levelled directed graph whose nodes consist of all pairs (i, x) from $[0, a) \times [0, b)$. We refer to (i, x) as *node x on level i*. If $i < a - 1$, this node has a single neighbour in the graph, node $f_i(x)$ on level $i + 1$. Every node in the graph is coloured with some set of colours from $[0, c)$. The principle CPLS, *coloured polynomial local search*, says that the following three statements cannot all be true:

(i) Node 0 on level 0 has no colours.

(ii) For every node x on every level $i < a - 1$, and for every colour y, if the neighbour $f_i(x)$ of x on level $i + 1$ has colour y, then x also has colour y.

(iii) Every node x on the bottom level $a - 1$ has at least one colour, $u(x)$.

When the parameters a, b, c are universally quantified, CPLS is expressed as a $\forall\Sigma^b_1$ sentence about an oracle α encoding the functions f_i and u and a predicate G, where $G_i(x, y)$ means "node x on level i has colour y".

To describe it explicitly as a search problem: the inputs are the parameters a, b, c and a solution is a witness that one of items (i)-(iii) above fails. That is, a colour y such that $G_0(0, y)$; or a node (i, x) and a colour y such that $G_{i+1}(f_i(x), y) \wedge \neg G_i(x, y)$; or a node $(a - 1, x)$ such that $\neg G_{a-1}(x, u(x))$.

To see that the principle is true, or equivalently that the search problem is total, suppose that (i)-(iii) hold simultaneously. Then we can reach a contradiction by arguing inductively on i that for all i, some node on level i has no colours. This argument can be formalized as a proof of CPLS in T^2_2. Moreover, this has a kind of converse, in that it is shown in [28] that CPLS is complete for the search-problem class $\forall\Sigma^b_1(T^2_2)$ with respect to many-one reductions.

It is worth pointing out that CPLS is a generalization of PLS in the sense that it simplifies to a PLS-complete problem if the parameters are restricted in a certain way, for instance if we fix the number of colours c to 1. In that situation, given a, b, we define the domain F of the PLS problem by putting $(i, x) \in F$ if and only if $\neg G_i(x, 0)$ holds, that is, if x does not have the unique possible colour 0 on level i. For $(i, x) \in F$, let $N(i, x) := (i + 1, f_i(x))$ and $C(i, x) := a - i$.

Thus, one way of looking at CPLS is that the parameter a is a bound on possible costs, b is a bound on the number of potential neighbours of a given element, and c is a bound on the number of potential reasons why an element could fail to be in the domain. It can be shown that CPLS becomes a PLS problem whenever one of the three parameters is constrained to be is at most polylogarithmic in the maximum of the two others.

2.2 Retraction WPHP and Σ^p_2 search problems

The *retraction weak pigeonhole principle* [21] asserts that, for $n > 0$, given two functions $f : n \to 2n$ and $g : 2n \to n$ there must be some $v < 2n$ such that $f(g(v)) \neq v$. It is true, because otherwise simultaneously f would be a surjection and g an injection. If f and g are polynomial time, this principle naturally gives rise to a problem in TFNP. We will be in a situation where f and g are P^{NP}, and for this we will define a more complex kind of search problem.

Definition 5. A Σ^p_2 *search problem* is specified by a coNP relation $R(x, y)$ and a polynomial bound q such that $\forall x\, \exists y < 2^{q(|x|)}\ R(x, y)$. We will often assume that the bound q is implicit in R and will not write it. The problem represents the search-task of finding such a y, given x.

As this definition makes sense outside the context of bounded arithmetic, we have written it in standard complexity-theory notation. But we could alternatively define a Σ^p_2 search problem as a true $\forall\Sigma^b_2$ sentence, in the style of our syntactical definition of TFNP problems.

A basic example of such a search problem is: for a fixed P^{NP} machine M, given an input, find a computation of M on this input. Here we assume that a computation includes witnesses for all NP queries that get the answer YES, so that the property of being a (correct) computation is thus coNP. We specify precisely what we mean, as it will be important in what follows.

Definition 6. We define a Π^b_1 formula *"w is a computation of M on input v"*. The formula interprets w as a sequence $\bar{q}, \bar{r}, \bar{y}$ of respectively NP queries, YES/NO replies and witnesses to replies. It expresses that

1. For each i, q_i is the i-th query asked by M in a computation on input v, assuming the previous replies were $r_1, \ldots, r_{i-1}$,
2. For each i, if reply r_i is YES then y_i witnesses this,
3. For all sequences $\bar{z}$ of possible counterexamples, for each i, if reply r_i is NO then z_i is not a counterexample to this.

The machine only accesses the oracle α via the NP queries. We say that w is a *precomputation of M on input v* if it satisfies 1. and 2. above.

Note that being a precomputation of M on a given input is a PV formula, so it makes sense to speak of precomputations also when α is only partially defined (as long as the defined part is large enough to verify 2. above). Note also that it is implicit in clause 1. of the definition that each query asked in a computation of M depends only on the input and the previous YES/NO replies to queries, not on the witnesses to the previous replies.

We now give our main definitions.

Definition 7. rWPHP$_2$ is a class of Σ_2^p search problems. A problem in the class is specified by P$^{\mathrm{NP}}$ machines computing functions $f_x(u)$ and $g_x(v)$, where we treat one argument x as a parameter. The functions f_x and g_x are constrained to take values less than $2x$ and x respectively. An input to the problem is a size parameter x. A solution is a pair $\langle v, w \rangle$ such that $v < 2x$, w is a computation of $f_x(g_x(v))$ in the sense of Definition 6, and the output of w is not v.

Definition 8. An NP search problem $Q(x, y)$ is PLS *counterexample reducible* to a Σ_2^p search problem $R(x', y')$ if there is a PLS problem $P(x'', y'')$ and polynomial time functions d and e with the following property: for any x, y', y'' such that $P(\langle x, y' \rangle, y'')$, either

1. $d(x, y'')$ witnesses that $R(e(x), y')$ is false, or
2. $Q(x, d(x, y''))$.

As in Definition 2, the oracle called by R is allowed to be a polynomial-time variant of the oracle α called by Q. Precisely, there is a polynomial-time relation A querying α such that, in the description above, R queries $A(x, \cdot)$ as its oracle rather than α.

Definition 9. The search problem class APPROX consists of all NP search problems which are PLS counterexample reducible to an rWPHP$_2$ problem.

The definition of PLS counterexample reducibility should be understood as follows. We are given x and want to find y such that $Q(x, y)$. We create an input $e(x)$ to R and are given a purported solution y' for which it is claimed that $R(e(x), y')$ – since R is a Σ_2^p search problem, this is a coNP claim which we cannot check directly. We then use $\langle x, y' \rangle$ as input for our PLS problem P, and find a solution y''. Then either 1. or 2. above holds, that is, either the coNP claim about R was false and $d(x, y'')$ is a counterexample, or $d(x, y'')$ is a solution to our original problem.

As far as we know, this notion of reducibility has not been studied before. We give some examples. Unfortunately, in the examples we know which are relatively simple, one part or another of the definition becomes trivial.

Firstly, every PLS problem $Q(x, y)$ is PLS counterexample reducible to the trivial Σ_2^p search problem defined by letting $R(x', y')$ hold for all $y' < 2^{|x'|}$. In the reduction, both d and e are the identity mapping and $P(\langle x, y'\rangle, y'')$ holds exactly if $Q(x, y'')$ does.

Now consider four search problems:

1. The Σ_2^p problem TOURNAMENT: given x and a binary oracle α representing a tournament on the set of vertices $[0, x)$, find y coding a set of at most $\lceil \log x \rceil$ vertices which dominates the tournament, that is, coding vertices $y_1, \ldots, y_k$ such that for all $z < x$ we have $\alpha(y_i, z)$ for some i.

2. The TFNP problem HOP (the *Herbrandized ordering principle*[5] [11]): given x and oracles for a binary relation $\preccurlyeq$ and a unary function h, find either a witness that $\preccurlyeq$ restricted to $[0, x)$ is not a total ordering, or a witness that h is not the $\preccurlyeq$-immediate predecessor function on $[0, x)$.

3. The TFNP problem RAMSEY: given x and an oracle R for a graph on $[0, x)$, find y encoding a set $s \subseteq [0, x)$ of cardinality $\lfloor \log x/2 \rfloor$ such that $[s]^2$ is homogeneous with respect to R.

4. The TFNP problem WEAKPIGEON[6]: given x and an oracle g for a function from $[0, 2x)$ to $[0, x)$, find $v, v' < 2x$ such that $v \neq v$ and $g(v) = g(v')$. The class of problems that are reducible to WEAKPIGEON was given the name PWPP in [22].

HOP is reducible to TOURNAMENT in the following sense. We are given a size x and a binary relation $\preccurlyeq$ on $[0, x)$ for which we want to solve HOP. Consider $\preccurlyeq$ as a tournament on $[0, x)$ and give it as input to TOURNAMENT. Suppose $s = \{y_1, \ldots, y_k\}$ is a purported solution to TOURNAMENT, with k polylogarithmic in x. By polynomial-time search we can find either a witness in s that $\preccurlyeq$ is not a total ordering (thus solving HOP), or a $\preccurlyeq$-least element y_i of s. In the second case, compute $y' := h(y_i)$. If $y' \not\preccurlyeq y_i$, then y_i is a solution to HOP. Otherwise, comparing y' to each $y_j \in s$ will either give us a witness that $\preccurlyeq$ is not a total ordering or reveal that y' witnesses that s is not a correct solution to TOURNAMENT.

The above is a description of a PLS counterexample reduction of HOP to TOURNAMENT. The function e translating input to HOP into input to TOURNAMENT is the identity. We then compute from s either a solution to HOP

[5] *Herbrandized* refers to the presence of the predecessor function h which turns the task of finding the $\preccurlyeq$-smallest element of $[0, x)$, which is naturally a Σ_2^p problem, into a TFNP problem. The *generalized iteration principle* of [14] is similar in spirit, as is the *graph ordering principle* in the propositional proof complexity literature.

[6] This is different in a non-essential way from how this problem is defined in [22], where it takes as input a circuit for the function g.

or a witness that s is not a solution to TOURNAMENT. However we can do this computation in polynomial time, while Definition 8 more generally allows it to be done by a call to a PLS problem P (assisted by a polynomial time "decoding" function d).

Our final examples of PLS counterexample reducibility are related to the class APPROX. We will show in Theorem 12 below that this class coincides with the class of NP search problems that are provably total using approximate counting. This has the following consequence.

Corollary 10 (of Theorem 12)**.** *The NP search problems* HOP*,* RAMSEY *and* WEAKPIGEON *are in* APPROX*. Therefore they are* PLS *counterexample reducible to* rWPHP_2 *problems.*

Proof. The problems RAMSEY [40, 21], HOP [11] and WEAKPIGEON are all provably total in APC_2. (This is also true for a stronger version of HOP in which $\preceq$ is only required to be a partial ordering, not a total ordering [11].) □

We are only aware of a simple direct proof of the reduction in the case of WEAKPIGEON. Suppose we are given a polynomial-time function g and a parameter x as input. To solve WEAKPIGEON we want to find a collision in g, which we interpret as a function from $2x$ to x. Let a function f be given by a P^{NP} machine inverting g, as follows. On input $u < x$, f queries whether $\exists v \! < \! 2x \, (g(v) = u)$. If the answer is NO, f gives up and outputs 0. If the answer is YES, f outputs the maximal such v, found by binary search. Consider the problem in rWPHP_2 specified by f, g and the parameter x. Recall that a solution is a pair $\langle v, w \rangle$ such that $v < 2x$, w is a computation of $f(g(v))$ in the sense of Definition 6, and the output of w is not v.

In the PLS counterexample reduction of WEAKPIGEON to this rWPHP_2 problem, the auxiliary functions d and e will be the identity. The reduction procedure P will once again be polynomial-time. To describe it, suppose we are given v and a precomputation w of f on input $u = g(v)$, with output $v' \neq v$ (we can ignore the computation of $g(v)$, since g is polynomial time). We want to either find a collision in g, or a witness that w is not a computation of f, that is, that some NO reply recorded in w is wrong.

First suppose that $v' = 0$ and was output by f because the reply to the first query "$\exists z \! < \! 2x \, (g(z) = u)$?" was NO. Then v is a witness that this reply is wrong. Otherwise, v' was found by binary search. In this case, we follow w until the first place where the binary search interval excludes v. Then, if the interval is strictly below v, we can use v to witness that the most recent NO reply was wrong. If it is above v, then this has to be the result of a YES answer, with a witness $v'' > v$ recorded in w. Thus we have found a collision, since $g(v'') = g(v) = u$.

Finally let us record here a fact which appears in Figure 1.

Proposition 11. *Relative to an oracle,* HOP *is not reducible to* WEAKPIGEON *and therefore is not in* PWPP*.*

This is implicit in the proof in [11] that T_2^1+iWPHP does not prove HOP is total. It explicitly follows from [33], which generalizes this result and works over the stronger base theory $\forall\mathrm{PV}(\mathbb{N})$.

2.3 Results

Recall that APPROX was defined in the last subsection as the set of NP search problems which are PLS counterexample reducible to an rWPHP_2 problem.

Theorem 12. $\forall\Sigma_1^b(\mathrm{APC}_2 + \forall\mathrm{PV}(\mathbb{N})) = \mathrm{APPROX}$.

In other words, APPROX captures the class of TFNP problems which are provably total using approximate counting. This is proved in Lemmas 18 and 20 in Section 3, by applying standard witnessing techniques from bounded arithmetic to the definition of the theory APC_2.

Theorem 13. CPLS *is not in* APPROX*.*

This is proved in Section 5, using a lemma about random oracles proved in Section 4. We briefly sketch the proof. We first fix an alleged PLS counterexample reduction of CPLS to a problem from rWPHP_2 specified by a pair of P^{NP} functions f and g, then choose a large size parameter n and use it to set suitable values for the parameters a, b, c of CPLS. We define a notion of a "legal" partial oracle, which in particular is one which does not contain any witness to CPLS. We adapt a lemma on random restrictions from [42] to show that with high probability a random partial oracle ρ from a certain distribution will "fix" YES or NO replies to all NP queries made in a P^{NP} computation, in the sense that these replies will never become wrong in any legal extension of ρ (Lemma 30). It follows that most partial oracles ρ from this distribution will fix computations of $(f \circ g)(v)$ in this sense on most inputs v. This is enough for ρ to determine a solution $\langle v, w\rangle$ to our instance of rWPHP_2 for which it is difficult to find a counterexample (Lemma 31). Finally we again adapt a proof from [42] to show, by an Adversary argument in which the Adversary's strategy uses only legal extensions of ρ, that our PLS reduction is not able to find a witness to CPLS from $\langle v, w\rangle$.

Our main result about bounded arithmetic, answering the question posed in [11], is an immediate consequence of Theorems 12 and 13:

Corollary 14. *The principle* CPLS *is not provable in* APC_2*. Since it is provable in T_2^2, it follows that, in the relativized setting,* APC_2 *does not prove all* $\forall\Sigma_1^b$ *consequences of full bounded arithmetic T_2.*

This naturally also limits the strength of theories that are provable in APC_2, such as the following one based on the usual (non-Herbrandized) ordering principle.

Corollary 15. *Consider the theory consisting of T_2^1 together with axioms stating that for every* PV *formula $R(x, y)$ and every a, if R is a partial ordering on $[0, a)$ then $[0, a)$ contains an R-minimal element. This theory, in the relativized setting, is strictly weaker than T_2^2.*

Proof. This theory is provable in T^2_2 by straightforward induction on a. It is also provable in APC_2 by an entirely different proof involving a reduction to the tournament principle, as is shown in [11] (by an argument due to Jeřábek). By Corollary 14, the theory does not prove CPLS, hence is weaker than T^2_2. □

Both CPLS and the ordering principle have short proofs in the *resolution* propositional proof system, and the argument above could also be used to show that the ordering principle is not "complete" for resolution, in the sense that there are things with short proofs in resolution which do not follow from it. But it is not clear what the most suitable notion of "follow from" is here.

Corollary 16. CPLS *is not polynomial-time Turing reducible to* RAMSEY *or* HOP*.*

Proof. RAMSEY and HOP are in APPROX by Corollary 10. If CPLS were polynomial-time Turing reducible to either of these problems, then Lemma 4 would imply that CPLS is provable in $\mathrm{APC}_2 + \forall\mathrm{PV}(\mathbb{N})$, contradicting Theorem 12 and Theorem 13. □

3 Witnessing and definability

This section contains our main technical work in logic: a proof of Theorem 12 via two lemmas corresponding to the two containments in the statement of the theorem. The proofs assume some familiarity with bounded arithmetic.

Intuitively, APC_2 is a combination of T^1_2 and the weak pigeonhole principle, and what we show is that the NP search problems provably total in APC_2 arise as a combination of PLS (which is known to correspond to T^1_2 [13]) and the weak pigeonhole principle, with an important difference that, while a proof can make many "calls" to WPHP, our reductions only allow one call. We also introduce the following technical condition on reductions, which we will need in our non-reducibility proof in Section 5.

Definition 17. We say that a NP search problem Q is *cleanly* PLS counterexample reducible to a Σ^p_2 search problem R, if Q is PLS counterexample reducible to R as in Definition 8 with the extra condition that the function e, which produces inputs to R from inputs to Q, does not make any oracle calls.

It may be helpful to think of such an e as a translation between size parameters, for which the structure of the oracle does not matter.

Lemma 18. *Every* NP *search problem provably total in* $\mathrm{APC}_2 + \forall\mathrm{PV}(\mathbb{N})$ *is cleanly* PLS *counterexample reducible to an* rWPHP_2 *problem.*

Proof. Let $Q(x, y)$ be an NP search problem. Assume that

$$\mathrm{APC}_2 + \forall z\, \varphi(z) \vdash \forall x\, \exists y\, Q(x, y),$$

where $\varphi(z)$ is a PV formula such that $\mathbb{N} \vDash \forall z\, \varphi(z)$ for all oracles. Thus we have

$$\mathrm{APC}_2 \vdash \exists y\, Q(x, y) \vee \exists z\, \neg\varphi(z).$$

Writing out the definition of APC_2, this means

$$T^1_2 + \forall b, c\, \exists v\!<\!2b\, \forall u\!<\!b\;\, e(c,u) \neq v \vdash \exists y\, Q(x,y) \vee \exists z\, \neg\varphi(z)$$

where the formula on the left is sWPHP for a universal P^{NP} machine $e(c,u)$ running code c on input u with time bound $|c|$. The quantifier $\forall b, \ldots$ should strictly speaking be $\forall b\!\neq\!0, \ldots$ but we suppress this here and below for the sake of readability. Replacing T^1_2 with the stronger theory S^2_2 and moving sWPHP to the right hand side gives

$$S^2_2 \vdash [\exists b, c\!<\!s'(x)\; \forall v\!<\!2b\, \exists u\!<\!b\;\, e(c,u) = v] \vee \exists y\, Q(x,y) \vee \exists z\, \neg\varphi(z), \qquad (3)$$

where we have also used Parikh's theorem [36] to bound b and c by some term $s'(x)$. We may assume $s'(x)$ has the form $2^{|x|^k}\!+x$ for some $k \in \mathbb{N}$, where "$+x$" is included to make it possible to recover x from $s'(x)$ as required below.

The formula in square brackets asserts that sWPHP fails for e for parameters b, c less than s'. By a standard technical fact stated and proved below as Lemma 19, we can amplify this failure to a P^{NP} function F which is a surjection from $(s')^5$ to $2(s')^5$, with no parameters. This lets us remove one block of existential quantifiers from (3) to give, setting the term s to be $(s')^5$,

$$S^2_2 \vdash [\forall v\!<\!2s\, \exists u\!<\!s\;\, F(u) = v] \vee \exists y\, Q(x,y) \vee \exists z\, \neg\varphi(z). \qquad (4)$$

To match the definition of rWPHP_2, we define a P^{NP} function of two arguments a, u by $f_a(u) = \min(F(u), 2a-1)$. Then (4) is equivalent to

$$S^2_2 \vdash [\forall v\!<\!2s\, \exists u\!<\!s\;\, f_s(u) = v] \vee \exists y\, Q(x,y) \vee \exists z\, \neg\varphi(z). \qquad (5)$$

The sentence in (5) is $\forall\Sigma^b_2$, so by Buss' witnessing theorem for S^2_2 ([9]) there is a P^{NP} machine which, provably in T^1_2, maps the input parameters x, v to a triple $\langle u, y, z\rangle$ witnessing one of the three existential quantifiers. Let g be defined so that $g_{s(x)}(v)$ first computes x from $s(x)$ and then outputs the first component u of this witnessing function applied to $\langle x, v\rangle$, as long as $u < s$; otherwise, g outputs 0. We have

$$T^1_2 \vdash [\forall v\!<\!2s\;\, f_s(g_s(v)) = v] \vee \exists y\, Q(x,y) \vee \exists z\, \neg\varphi(z).$$

Now, $f_s(g_s(v)) = v$ can be written as a Π^b_2 formula:

$$\forall w\, [w \text{ is a computation of } f_s(g_s(v)) \rightarrow \text{output}(w) = v],$$

where w is suitably bounded by a term in x, and "w is a computation of $f_s(g_s(v))$" is a Π^b_1 formula as in Definition 6, describing the P^{NP} machine that first computes g and then computes f on the output.

So we have

$$\begin{aligned} T^1_2 \vdash \forall v\!<\!2s\, \forall w\, [w \text{ is not a computation of } f_s(g_s(v)) \vee \text{output}(w) = v] \\ \vee \exists y\, Q(x,y) \vee \exists z\, \neg\varphi(z). \end{aligned}$$

The formula in square brackets is now Σ^b_1, so by the PLS witnessing theorem for T^1_2 ([13]) there is a PLS problem $P(x'', y'')$ witnessing this whole sentence. That is, if we solve P on input $x'' = \langle x, v, w\rangle$ and find y'' such that $P(x'', y'')$, then one of the following holds:

1. w is not a precomputation of $f_s(g_s(v))$, or has output v,

2. y'' is a tuple containing a witness that some NO reply in w is wrong,

3. y'' is a tuple containing a witness to $\exists y\, Q(x, y)$.

We know that y'' cannot contain a witness to the last disjunct $\exists z\, \neg\varphi(z)$ as by assumption $\mathbb{N} \vDash \forall z\, \varphi(z)$.

Using the notation of Definition 8, letting d be the function that outputs the witness of incorrectness in case 2., and the witness to $\exists y\, Q(x, y)$ in case 3., and setting $e(x) = s(x)$, we see that Q is cleanly PLS counterexample reducible to the rWPHP$_2$ problem given by f and g. □

To complete the proof of Lemma 18 we need a technical lemma from [45, Section 2] about "amplifying" failures of WPHP.

Lemma 19. *Let e be a P^{NP} function. There is a P^{NP} function F such that provably in S^2_2, if $e(c, \cdot)$ is a surjection $b \to 2b$ then $F(a, b, c, \cdot)$ is a surjection $b \to 2^{|a|}b$. Furthermore there is a P^{NP} function F' such that provably in S^2_2, if $e(c, \cdot)$ is a surjection $b \to 2b$ for some $0 < b, c < s$, then F' is a surjection $s^5 \to 2s^5$ (with no parameters).*

Proof. Let $h(a, b, c, u)$ be the function that, assuming $u < 2^{|a|}b$, interprets u as a pair $\langle u_0, u_1\rangle$ with $u_0 < b, u_1 < 2^{|a|}$, and outputs the number $e(c, u_0)2^{|a|} + u_1$. If $e(c, \cdot)$ is a surjection from b onto $2b$, then $h(a, b, c, \cdot)$ is a surjection from $2^{|a|}b$ onto $2^{|a|+1}b$. Consider now the function $F(a, b, c, u)$ which, given $u < b$, applies the composition of $e(c, \cdot), h(1, b, c, \cdot), h(2, b, c, \cdot), h(4, b, c, \cdot), \ldots, h(2^{|a|-1}, b, c, \cdot)$, to u, in the order shown. If $e(c, \cdot)$ is a surjection from b onto $2b$, then $F(a, b, c, \cdot)$ is a surjection from b onto $2^{|a|}b$. To prove this, we consider any fixed number $v < 2^{|a|}b$ and show by reverse induction on $i < |a| - 1$ that v can be obtained by applying the composition of $h(2^i, b, c, \cdot), h(2^{i+1}, b, c, \cdot), \ldots, h(2^{|a|-1}, b, c, \cdot)$ to some argument u below $2^i b$. This is where we need the scheme Σ^b_2-LIND, which is available in S^2_2.

For the last claim, we note that if we code quadruples as $\langle\langle\cdot, \cdot\rangle, \langle\cdot, \cdot\rangle\rangle$ where $\langle\cdot, \cdot\rangle$ is Cantor's pairing function, then for each k larger than a fixed natural number any quadruple of numbers less than k has code less than k^5. We treat the argument $x < s^5$ of F' as a quadruple $\langle a', b', c', u'\rangle$ and set $F'(\langle a', b', c', u'\rangle) := F((a' + 1)^6, b', c', u')$. Since such arguments x will cover all quadruples of numbers below s, it follows from the properties of F that F' contains every number less than $2^{6|s|}b$ in its range, and in particular every number less than $2s^5$. □

The next lemma is the converse to Lemma 18.

Lemma 20. *If Q is an* NP *search problem* PLS *counterexample reducible to an* rWPHP$_2$ *problem, then Q is provably total in* $\mathrm{APC}_2 + \forall\mathrm{PV}(\mathbb{N})$.

Proof. Let Q be an NP search problem that is PLS counterexample reducible to the rWPHP$_2$ problem given by the functions f and g. Let $P(x'', y''), d, e$ be as in the definition of PLS counterexample reducibility. Consider the PV formula $\xi(x, v, w, y, y'')$ defined by

$$v < 2e(x) \wedge P(\langle x, \langle v, w\rangle\rangle, y'') \wedge y = d(x, y'')$$
$$\wedge\, [y \text{ does not witness that } w \text{ is not a computation of } f_e(g_e(v)) \neq v],$$

with y'' suitably bounded. We view ξ as defining an NP search problem with input x and output $\langle v, w, y, y''\rangle$. Notice that $\xi(x, v, w, y, y'')$ implies that $Q(x, y)$, so we have $Q \leq \xi$.

We claim that APC$_2$ proves that ξ is total. To see this, work in APC$_2$ and consider some input x. By sWPHP(PV$_2$), there is some $v < 2e(x)$ which is outside of the range of $f_{e(x)}$ on inputs below $e(x)$. By T^1_2, there is some computation of $f_{e(x)}(g_{e(x)}(v))$, say w, which by the choice of v must produce an output different from v. Again by T^1_2, there is a solution to P on input $\langle x, \langle v, w\rangle\rangle$, say y''. Clearly, $y = d(x, y'')$ cannot witness that w is not a computation of $f_{e(x)}(g_{e(x)}(v))$ with output different from v, so $\xi(x, v, w, y, y'')$ holds. This proves the claim.

We have shown that Q is reducible to a problem provably total in APC$_2$. It follows from Lemma 4 that Q is provably total in $\mathrm{APC}_2 + \forall\mathrm{PV}(\mathbb{N})$. □

Lemmas 18 and 20 have the following additional consequence.

Corollary 21. *The class* APPROX *can equivalently be defined as the set of* NP *search problems which are* cleanly PLS *counterexample reducible to an* rWPHP$_2$ *problem.*

4 Fixing lemma

This section contains our main technical result in complexity, Lemma 27, which is an extension of the "fixing lemma" from [42]. There, the fixing lemma is a limited switching lemma which says the following: given suitable parameters a, b, c for CPLS, for a well-chosen probability distribution on partial restrictions to an oracle α encoding $(f_i)_{i<a-1}$, u, $(G_i)_{i<a}$, a random restriction has a relatively high probability of determining the value of a narrow CNF in propositional variables standing for bits of α. Importantly, the restriction does not reveal a witness to CPLS; in particular, the (unsatisfiable) CNF asserting that there is no witness to CPLS has to be determined to be true.

In our application in the proof of Theorem 13, we want to fix answers to the NP queries made in a P$^{\mathrm{NP}}$ computation. Each query is (the negation of) a CNF, but now there are many of them, and they are made adaptively depending on earlier replies. So we cannot use the lemma from [42] directly. Instead we adapt the proof to show that given a low-depth decision tree labelled with CNFs, with high probability a random restriction fixes the truth values of all CNFs along some branch. This is the basic content of Lemma 27 below.

Our definitions are essentially as the same as in [42], and so is one of the proofs. We will repeat some definitions almost verbatim, but will only give high-level descriptions of some other definitions and of the unchanged proof details.

We think of the bits of the oracle as propositional variables. So, for example, for each node (i, x) there are $\log b$ variables $(f_i(x))_0, \ldots, (f_i(x))_{\log b-1}$ expressing the value of $f_i(x)$. A total oracle is defined by a total assignment to all variables. We will be working with partial oracles, which we will also call *partial assignments* or *restrictions*.

We copy in full the definition of a *random restriction* from [42]. First, a *path* in a partial assignment β is a maximal sequence $(i, x_0), \ldots, (i+k, x_k)$ of nodes such that $f_{i+j}(x_j) = x_{j+1}$ in β for each $j \in [0, k)$. Note that a path may consist of a single node (i, x) if x is neither in the domain of f_i nor in the range of f_{i-1}. If all functions f_i are partial injections, then every node is on some unique path.

Definition 22. ([42, Definition 5.7]) Fix parameters $0 < p, q < 1$. Let $\mathcal{R}_{p,q}$ be the distribution of random restrictions chosen as follows.

R1. For each pair $i < a$ and $x < b$, with probability $(1-p)$ include (i, x) in a set Z. For each $i < a$, choose f_i uniformly at random from the partial injections from the domain $\{x < b : (i, x) \in Z\}$ into b.

R2. Set colours on the path beginning at $(0, 0)$ so that $G_i(x, y) = 0$ for all y for all nodes (i, x) on that path.

R3. For every other path π, with probability $(1-q)$ *colour* π randomly with one colour. That is, choose uniformly at random a colour y and, for every node (i, x) on π, set $G_i(x, y) = 1$ and then set $G_i(x, y') = 0$ for all $y' \neq y$.

R4. Finally consider each node $(a-1, x)$ on the bottom level. It is on some path π. If π was coloured at step R3, then set $u(x) = y$ where y is the unique colour assigned to π (that is, $G_{a-1}(x, y) = 1$). Otherwise leave $u(x)$ undefined.

We will also use $\mathcal{R}_{p,q}$ to denote the support of this distribution.

We take the definitions of *legal restrictions* and *good restrictions* from [42, Definition 5.4 and Lemma 5.8]. *Legal* restrictions are those that meet a minimal standard of "niceness" – on every path either no colour variables are set, or they are all set in one of a few particular ways which do not immediately witness CPLS. Among restrictions $\rho \in \mathcal{R}_{p,q}$, the only ones that are not legal are those that contain a path connecting $(0, 0)$ with some node of the form $(a-1, x)$. Our lower bound in the next section will make use of a game played between a Prover, who is trying to witness CPLS by making oracle queries, and an Adversary who is trying to answer queries in a way that does not witness CPLS. It will turn out that the Adversary can restrict herself to answers that come from legal restrictions. In effect, we do not have to worry about the evaluation of formulas under restrictions which are not legal.

A *good* restriction is a legal one which is of typical size, measured in various ways – in particular no path is very long, and there is a reasonable fraction of variables unset at every level. A *bad* restriction is one which is not good.

It may be useful to keep in mind what the analogous definitions would be if we were dealing with the more familiar pigeonhole principle PHP instead of CPLS. A legal restriction would be any restriction representing a partial injection. With probability parameter p, a random restriction would choose holes independently with probability $1-p$, and then randomly map some pigeons to the chosen holes. A good restriction would be a legal one that leaves at least, say, a fraction $p/2$ of holes unset.

We choose a suitable large n and fix our parameters as $a = b = n$, $c = \lfloor n^{1/7} \rfloor$, $p = n^{-4/7}$ and $q = n^{-2/7}$, where b and c are powers of 2.

Lemma 23. ([42, Lemma 5.8]) *The probability that a random restriction is bad is exponentially small in n.*

Definition 24. Let ρ be a restriction. We say that a CNF B is:

- *fixed to* 0 by ρ if ρ falsifies B, that is, if for some conjunct in B each literal in the conjunct is set to 0 by ρ,
- *fixed to* 1 by ρ if it is not fixed to 0 by any legal extension of ρ.

Note that a legal restriction can fix a CNF to at most one truth value. The following proposition is therefore obvious.

Proposition 25. *If ρ fixes a CNF B then every extension of ρ also fixes B to the same value.*

It follows from the proof of the "fixing lemma" [42, Lemma 5.9] that, for k reasonably small compared to n, the probability that a given k-CNF is fixed by a random restriction is relatively high – in fact, the probability that it is *not* fixed is $O(kn^{-1/7})$. We need a slightly more general version that also bounds some conditional probabilities.

Lemma 26 (conditional fixing lemma)**.** *Let $A_1, \ldots, A_m$ be a sequence of k-CNFs and $e_1, \ldots, e_m$ be a sequence of* 0/1 *values such that*

$$\Pr[\rho \textit{ is bad} \,|\, \rho \textit{ fixes each } A_i \textit{ to } e_i] < 1/2.$$

Let B be a k-CNF. Then

$$\Pr[\rho \textit{ does not fix } B \,|\, \rho \textit{ is good and } \rho \textit{ fixes each } A_i \textit{ to } e_i] < 12kn^{-1/7}.$$

Proof. If we remove the CNFs A_i, this is essentially the "fixing lemma" of [42, Lemma 5.9] and our proof is almost identical. Define

$$F = \{\rho \in \mathcal{R}_{p,q} : \rho \text{ fixes each } A_i \text{ to } e_i\}$$
$$G = \{\rho \in \mathcal{R}_{p,q} : \rho \text{ is good}\}$$
$$S = \{\rho \in F \cap G : \rho \text{ does not fix } B\}$$

so that our S is the intersection with F of the set S defined in [42]. The assumption gives us that $\Pr[F \cap G]/\Pr[F] \geq 1/2$ and our goal is to show that $\Pr[S]/\Pr[F \cap G]$ is small.

Every $\rho \in S$ does not falsify B but does have some legal extension which falsifies B. Exactly as in [42] we define a function θ on S by $\theta(\rho) = \sigma'$, where σ' is a certain minimal legal extension of ρ. We have, over $\rho \in S$,

1. $\Pr[\theta(\rho)]/\Pr[\rho] \geq \frac{1}{2}n^{1/7}$
2. θ is at most $3k$-to-one
3. $\theta(\rho) \in F$ (although it may happen that $\theta(\rho) \notin F \cap G$).

Items 1 and 2 are proved as in [42]. Item 3 is immediate from Proposition 25.

Now partition S as $S_0, \ldots, S_{3k-1}$ where $S_i = \{\rho \in S : \rho \text{ is the } i\text{th preimage of } \theta(\rho)\}$. Then

$$\begin{aligned}\Pr[S_i] = \sum_{\rho \in S_i} \Pr[\rho] = \sum_{\rho \in S_i} \Pr[\theta(\rho)] \frac{\Pr[\rho]}{\Pr[\theta(\rho)]} &\leq 2n^{-1/7} \sum_{\rho \in S_i} \Pr[\theta(\rho)] \\ &\leq 2n^{-1/7} \Pr[F]\end{aligned}$$

where for the last inequality we use that $\sum_{\rho \in S_i} \Pr[\theta(\rho)] \leq \Pr[F]$, since θ is an injection from S_i to F. This step is the main difference from [42], which uses only that θ is an injection from S_i to $\mathcal{R}_{p,q}$, giving the weaker bound $\sum_{\rho \in S_i} \Pr[\theta(\rho)] \leq \Pr[\mathcal{R}_{p,q}] = 1$.

It follows that $\Pr[S] = \Pr[S_0] + \cdots + \Pr[S_{3k-1}] \leq 6kn^{-1/7} \Pr[F]$. Hence $\Pr[S]/\Pr[F \cap G] \leq 12kn^{-1/7}$ as required. □

Lemma 27. *Consider a complete binary decision tree in which each internal node is labelled with a k-CNF and has outgoing edges for NO and YES answers. A node z and a restriction ρ are* compatible *if ρ is good and, for every CNF B on the path down from the root to z, ρ fixes B to the value specified by the outgoing edge along the path.*

Let $\varepsilon = \Pr[\rho \text{ is bad}]$. A node z is big *if $\Pr[\rho \text{ is compatible with } z] > \varepsilon$. Let S_d be the set of good restrictions ρ which are compatible with some big node at depth d. Then*

$$\Pr[S_d] \geq 1 - d \cdot 12kn^{-1/7} - 2^{d+1}\varepsilon.$$

This will be used in the next section, where the decision tree will model a computation of a P^{NP} machine. In particular d and k will be polylogarithmic in n and ε will be exponentially small in n. It follows from the lemma that at least one node on the bottom level, and thus at least one computation of the machine, is compatible with some ρ.

Proof. We use induction on d. For the base case $d = 0$, first observe that every good restriction is compatible with the root. It follows that the root is big, as thanks to Lemma 23 we may assume that $\varepsilon < 1/2$. Hence S_0 is just the set of good restrictions.

At depth d in the tree, by the definition of compatibility each restriction in S_d is compatible with exactly one big node. Consider any such big node z. It is labelled with a k-CNF B and has a NO child z_0 and a YES child z_1. Define P_z as

$$\Pr[\rho \text{ is not compatible with either } z_0 \text{ or } z_1 \mid \rho \text{ is compatible with } z]$$

This is equal to the probability that ρ does not fix B, under the condition that ρ is good and $\rho \in C_z$, where C_z is the set of restrictions which correctly fix all CNFs above z. Notice that $\Pr[\rho \text{ is bad}] = \varepsilon$, so in particular $\Pr[\rho \text{ is bad and } \rho \in C_z] \leq \varepsilon$. On the other hand $\Pr[\rho \text{ is good and } \rho \in C_z] > \varepsilon$, since z is big. Hence of restrictions $\rho \in C_z$, the fraction which are good is at least $1/2$. This is the condition we need to apply Lemma 26, which gives $P_z < 12kn^{-1/7}$.

Hence, summing over big nodes at level d, the probability that ρ is compatible with some (not necessarily big) node at depth $d+1$ is at least

$$\sum_{z \in \{0,1\}^d,\, z \text{ big}} (1 - P_z) \Pr[\rho \text{ is compatible with } z] \geq (1 - 12kn^{-1/7}) \Pr[S_d].$$

To obtain S_{d+1} we must finally remove the restrictions which are compatible with non-big nodes at depth $d+1$. But there are at most 2^{d+1} such nodes, so the probability of being compatible with any of them is at most $2^{d+1}\varepsilon$. A straightforward calculation shows that

$$(1-12kn^{-1/7})(1-d\cdot 12kn^{-1/7}-2^{d+1}\varepsilon)-2^{d+1}\varepsilon \geq 1-(d+1)\cdot 12kn^{-1/7}-2^{d+2}\varepsilon,$$

which completes the inductive step. □

5 Non-reducibility

Consider a restriction ρ and a Σ^b_1 formula $\exists y < t\, \theta(a, y)$, where θ is a PV formula and a is some number. We say that this formula *is witnessed* in ρ if there is some $b < t$ such that $\theta(a, b)$ holds in ρ. That is, if you run the computation verifying $\theta(a, b)$ and answer queries to α with values from ρ, these values are all defined and the computation is accepting.

Recall from Definition 6 that a precomputation of a P^{NP} machine contains a correct witness for every YES reply, but may be wrong about NO replies.

Definition 28. Let ρ be a restriction. A precomputation w of a P^{NP} machine M is *fixed* by ρ if both of the following hold.

1. For every NP query in w with a YES reply, the witness provided by w is correct in ρ.
2. No NP query in w with a NO reply is witnessed in any legal $\sigma \supseteq \rho$.

We say that ρ *fixes a precomputation of M on input v* if there is some such w. For a function f computed by a P^{NP} machine, we write $\rho \Vdash w\colon f(x) = y$ if ρ fixes a precomputation w of f on input x that outputs y, and we write $\rho \Vdash f(x) = y$ if ρ fixes some such w.

If w is fixed by ρ then ρ fixes, in the sense of Definition 24, each DNF representing an NP query made in w. Note that w does not have to be a computation of M relative to any complete oracle α extending ρ. In fact, if ρ is legal and w contains a query like "is there a witness to CPLS?", then w *cannot* be a computation of M relative to any complete α, because α will contain a witness to a YES answer, but the answer given in w will be NO.

The symbol $\Vdash$ is intentionally chosen to be the same one as in forcing. In fact, one could formulate the concept of fixing in terms of a forcing relation, with the restrictions as forcing conditions. However, attempting to preserve all the trappings of forcing in the context of finite combinatorics leads to some annoying issues, so in this paper we do not explore this possibility further.

Lemma 29. *For a* P^{NP} *function* f*, a restriction* ρ *and an input* x*, there is at most one* y *such that* $\rho \Vdash f(x) = y$*.*

Proof. The progress of a P^{NP} precomputation depends only on the YES/NO replies to NP queries, not on the witnesses chosen. In all precomputations of $f(x)$ fixed by ρ these replies are necessarily the same. □

Below a "suitable" n is one for which $n^{1/7}$ is a power of two.

Lemma 30. *Let* M *be a* P^{NP} *machine, running on inputs* x *with* $|x|$ *polylogarithmic in* n*. For all suitable large enough* n*, for every such input* x*,*

$$\mathrm{Pr}_{\rho\sim\mathcal{R}_{p,q}}[\rho \textit{ fixes a precomputation of } M \textit{ on } x] \geq 1 - n^{-1/6}.$$

Proof. We can model a run of M on v as a decision tree $\mathcal{T}_M$. The height d of $\mathcal{T}_M$ is bounded by the running time of M. At each node the tree makes an NP query; by negating the reply, we can view this as a query to a k-CNF, where k is some obvious syntactic upper bound on the time needed to verify a witness to the query. Since M is a P^{NP} machine, k can be chosen polynomial in the running time of M. So we can apply Lemma 27 with $k = d = |n|^c$ for some $c \in \mathbb{N}$. This gives the lower bound

$$1 - |n|^{2c} n^{-1/7} - 2^{|n|^c+1}\varepsilon \tag{6}$$

on the probability that ρ is compatible with one of the leaves of $\mathcal{T}_M$. By Lemma 23 the probability ε that ρ is bad is exponentially small in n, so the bound in (6) is at least $1 - n^{-1/6}$ for n sufficiently large.

Finally, suppose ρ is compatible with a leaf of $\mathcal{T}_M$. We form a precomputation w by answering queries with the replies given on the path from the root to the leaf. For each YES reply, it follows from the definition of fixing a DNF to 1 (that is, fixing a CNF to 0) that ρ provides enough information to verify at least one witness to the reply; we make some such witness part of w. □

Lemma 31. *Let a search problem in* rWPHP_2 *be given by* P^{NP} *functions* $f_x(u)$ *and* $g_x(v)$*. Let* $s(n)$ *be quasipolynomial in* n*. Then for all suitable large enough* n*,*

$$\mathrm{Pr}_{\rho\sim\mathcal{R}_{p,q}}[\textit{there exist } v \neq v' \textit{ such that } \rho \Vdash f_s(g_s(v)) = v'] \geq 1 - 3n^{-1/6}.$$

Proof. Choose n sufficiently large. Let M be the P^{NP} machine which takes input n, v and computes $f_s(g_s(v))$ by first computing g and then f. As s is quasipolynomial in n, we may assume that M satisfies the assumption of Lemma 30 on input size. We will write just f and g below, suppressing the parameter s.

Consider the machine M running on inputs $v < 2s$. By Lemma 30, for any fixed v, a random ρ fixes a precomputation of M on v with probability at least $1 - n^{-1/6}$. It follows that with probability at least $1 - 3n^{-1/6}$ a random ρ simultaneously fixes precomputations for at least $2/3$ of all inputs v (as otherwise the fraction of pairs (ρ, v) in which ρ does not fix a precomputation on v would be more than $n^{-1/6}$). Fix such a ρ. In particular, there are at least $s + 1$ many distinct inputs $v_0, \ldots, v_s$ for which ρ fixes precomputations $w_0, \ldots, w_s$.

The machine M first computes $u = g(v)$, which is necessarily less than s, and then computes $f(u)$. Hence, by the pigeonhole principle, there is some u for which there exist distinct i, j such that $\rho \Vdash w_i : g(v_i) = u$ and $\rho \Vdash w_j : g(v_j) = u$. (Here and below we are abusing our notation slightly, as w_i and w_j are really precomputations of g followed by f.)

But, by Lemma 29, there must be a single v' such that $\rho \Vdash w_i : f(u) = v'$ and $\rho \Vdash w_j : f(u) = v'$. At least one of v_i and v_j is distinct from v'; without loss of generality suppose v_i is. Let $v = v_i$ and $w = w_i$. Thus we have $\rho \Vdash w : f(g(v)) = v'$ and $v' \neq v$, as required. □

Now consider the following Prover-Adversary game, given by an NP search problem $Q(x, y)$ and a Σ_2^p search problem $R(x', y')$. At the start of the game, the Prover queries R for some input x', with $|x'|$ polynomial in $|x|$, and the Adversary gives a reply y'. Then the Prover repeatedly queries bits of the oracle α, and the Adversary replies. The Prover is limited in the number of bits of α he can remember at once, and can also forget bits to save memory. The Prover wins when the partial oracle in his memory either witnesses $Q(x, y)$ for some y, or witnesses that $R(x', y')$ is false. This game models PLS counterexample reducibility, in the following sense.

Lemma 32. *Suppose an* NP *search problem $Q(x, y)$ is cleanly* PLS *counterexample reducible to a Σ_2^p search problem $R(x', y')$. Then for all inputs x the Prover can win the game using only polynomially many (in $|x|$) bits of memory.*

Proof. This is an immediate consequence of the definitions of PLS counterexample reducibility (Definition 8) and PLS. Using the notation of Definition 8, the Prover first asks for y' such that $R(e(x), y')$. Since the reduction is clean, the value of $e(x)$ does not depend on any oracle queries (Definition 17). He then sets $x'' = \langle x, y' \rangle$ and simulates the (exponential time, but polynomial memory) task of solving the PLS problem $P(x'', y'')$ by starting with $y'' = 0$ and then repeatedly setting y'' to $N_{x''}(y'')$, finding domain elements of smaller and smaller cost, until either the costs stop decreasing or y'' leaves the domain $F_{x''}$. Once y'' satisfying $P(x'', y'')$ is found, the Prover simulates the polynomial-time computation of $y := d(x, y'')$. Then he simulates two additional polynomial-time computations in order to check whether $Q(x, y)$ holds and whether y witnesses that $R(x', y')$ is false. By the definition of PLS counterexample reducibility, one of the two possibilities must hold, thus allowing the Prover to win the game.

During the entire game, the Prover never needs to remember more bits of the oracle than are necessary to fix simultaneously the cost and membership of the domain of one candidate solution to P, the computation of its neighbour, the cost of the neighbour, and possibly computations of d and the witnessing for Q and R. □

We can now prove Theorem 13, that CPLS is not in the class APPROX.

Proof of Theorem 13. Assume that CPLS is in APPROX. Then by Corollary 21 it is cleanly PLS counterexample reducible to an instance of rWPHP_2 given by some functions f and g. This means that, by Lemma 32, the Prover can win the Prover-Adversary game in which Q is CPLS and R is rWPHP_2, using only polynomially many bits of memory. We obtain a contradiction by describing a strategy for the Adversary that defeats any Prover with small memory.

The Prover first makes his query x' to rWPHP_2. The Adversary then picks a restriction ρ from $\mathcal{R}_{p,q}$ for which there exist a precomputation w and numbers $v, v' < 2x'$ with $v \neq v'$ such that $\rho \Vdash w : f_{x'}(g_{x'}(v)) = v'$. By Lemma 31 such a ρ exists, and by Lemma 23 we may further assume that it is good. The Adversary replies with $\langle v, w \rangle$.

Then, using the goodness of ρ and the limited size of the Prover's memory, the Adversary is able to have in hand throughout the game a legal $\sigma \supseteq \rho$ which contains all bits in the Prover's current memory. Such a σ can never witness CPLS, because it is legal. However, a legal σ also cannot witness that $\langle v, w \rangle$ is not a solution to rWPHP_2, because the only way to do this would be to witness that one of the NO replies in w is wrong, which is impossible by the choice of ρ. The details of the strategy are as in the proof of [42, Theorem 5.10]. □

6 Reformulation in propositional logic

In this section we sketch another way of presenting our main result about bounded arithmetic, that CPLS, considered as a $\forall\Sigma^b_1$ principle, is not provable in APC_2. We will use propositional proof complexity and in particular the well-known Paris-Wilkie translation of relativized bounded arithmetic into propositional logic [37].

Suppose φ is bounded formula of L_{PV}, and that we have specified values $\bar{n}$ for all free variables in φ. We can write a propositional formula $\langle \varphi \rangle$ with the same semantics as φ, if we interpret propositional variables x_n as bits $\alpha(n)$ of the oracle. Below we will use *narrow* to mean "of width polylogarithmic in $\bar{n}$".

If φ does not mention the oracle α, then its translation $\langle \varphi \rangle$ is the propositional constant $\top$ or $\bot$, depending on whether φ is true or false in $\mathbb{N}$. If φ is $\alpha(n)$, then $\langle \varphi \rangle$ is the propositional variable x_n. If φ is a PV formula, then $\langle \varphi \rangle$ is a narrow CNF — we can take it to be the conjunction of clauses expressing "some oracle reply in w is false" over all possible rejecting computations w

of the polynomial-time machine deciding φ. If φ is a Π^b_1 formula $\forall x < n\, \theta(x)$, then again $\langle\varphi\rangle$ is a narrow CNF, namely the conjunction, over $m < n$, of the translations $\langle\theta(x)\rangle$ with $x \mapsto m$.

The translation theorem we will use follows from the translation of T^1_2 into treelike Res(log) refutations from [26] and the connection between treelike Res(log) and narrow resolution [29]. It can also be shown via PLS witnessing, as described in [11].

Theorem 33. *Let $\varphi(\bar{n})$ be a Π^b_1 formula and suppose $T^1_2 \vdash \forall\bar{n}\, \neg\varphi(\bar{n})$. Then the translations $\langle\varphi\rangle$ have narrow resolution refutations.*

Now suppose for a contradiction that $\mathrm{APC}_2 \vdash \mathrm{CPLS}$. Consider CPLS as described in Section 4, with parameters $a = b = n$ and $c = \lfloor n^{1/7}\rfloor$ and the structure of the problem given entirely by the oracle. Let $Q(n, y)$ assert that y is a solution to CPLS on input n. We may bound y by some term $t(n)$, such that $\mathrm{APC}_2 \vdash \forall n \exists y < t\, Q(n, y)$. By the proof of Lemma 18, there exist P^{NP} machines f, g defining an instance of rWPHP_2, and a term $s(n)$, such that

$$T^1_2 \vdash \forall n \forall v < 2s\, \forall w\, [w \text{ is not a computation of } f_s(g_s(v)) \vee \mathrm{output}(w) = v \\ \vee \exists y < t\, Q(n, y)].$$

Let M be the P^{NP} machine which takes input n, v and computes $f_s(g_s(v))$ by first computing g and then f. We think of v as the "real input" to M and of n as a parameter, and write $\mathrm{Comp}_M(v, w)$ for the Π^b_1 formula from Definition 6 expressing that w is a computation of M on input v. Noting that the expression on the right above is $\forall\Sigma^b_1$, we can apply Theorem 33 to conclude that the family of narrow CNFs

$$\Phi_{n,v,w} := \langle v < 2s\rangle \wedge \langle\mathrm{Comp}_M(v, w)\rangle \wedge \langle\mathrm{output}(w) \neq v\rangle \wedge \bigwedge_{y<t} \langle\neg Q(n, y)\rangle$$

has narrow resolution refutations, that is, of width polylogarithmic in n, v, w.

Fix a suitable large n. By definition, no legal restriction σ can falsify any clause in the last conjunct $\bigwedge_{y<t}\langle\neg Q(n, y)\rangle$, as otherwise for some y there is an accepting computation of $Q(n, y)$ over σ, so σ witnesses CPLS.

By Lemma 31, with high probability for a random ρ from $\mathcal{R}_{p,q}$ there exist $v < 2s$ and a precomputation w of M on s with $\mathrm{output}(w) \neq v$ such that w is fixed by ρ, meaning that all witnesses in w to YES answers are correct in ρ and no query with a NO answer has a witness in any legal extension of ρ. It follows that for such v, w, no clause in the first three conjuncts of $\Phi_{n,v,w}$ is false in any legal extension of ρ. By Lemma 23 we can pick a good ρ for which such v, w exist.

By the Prover-Adversary construction in the proof of [42, Theorem 5.10], we can exploit the limited width of the refutation of $\Phi_{n,v,w}$ to find a legal extension of ρ which falsifies one of the conjuncts of $\Phi_{n,v,w}$. This is a contradiction.

7 Open problems

The *random resolution* propositional proof system was introduced in [11]. Very roughly speaking, a refutation of a CNF F in this system is a refutation of $F \wedge A$, where A is any CNF which is true with high probability.

Suppose a sentence $\forall n\, \varphi(n)$, with φ a Σ^b_1 formula, is provable in the subtheory of APC_2 consisting of T^1_2 together with the surjective WPHP only for polynomial time functions. It was shown in [11] that this implies that the translations $\langle \neg\varphi(n) \rangle$ have narrow refutations in random resolution.

Open Problem 1. Is there a natural propositional proof system which captures, in a similar way, the $\forall\Sigma^b_1$ consequences of full APC_2?

Ideally, one would want to show not only that APC_2 proofs translate into the system, but also something in the opposite direction, for example, that if $\langle \neg\varphi(n) \rangle$ has small, suitably uniform refutations in the system, then $\forall n\, \varphi(n)$ is provable in APC_2. Some system with these properties could be constructed using the Paris-Wilkie translation and our arguments in Section 6, but it would be rather unnatural and awkward.

It is consistent with what we know that narrow random resolution, or possibly random resolution with no width restriction, already provides a positive answer to Open Problem 1. So, we can ask:

Open Problem 2. Is there a $\forall\Sigma^b_1$ sentence which is provable in APC_2 but whose propositional translations do not have narrow random resolution refutations?

A candidate is the Herbrandized ordering principle HOP, which is provable in APC_2 [11] but not in the subtheory mentioned above [2].

What makes this problem interesting is that, so far, our only tool for proving lower bounds on random resolution is the fixing lemma of [42]. For a typical random restriction, it is a small step from proving this to proving our conditional fixing lemma from Section 4, which implies unprovability in APC_2. But showing a separation seems to require finding a principle and a random restriction for which one lemma holds, but not the other. The restrictions used to show unprovability of HOP in [2] may be useful here.

Finally we mention a rather obvious question: is every problem in APPROX reducible to CPLS? This is subsumed in the old open problem, discussed in the introduction, of separating the classes GI_k or the theories T^k_2: it is possible that every search problem reducible to any GI_k is already reducible to CPLS.

References

[1] James Aisenberg, Maria Luisa Bonet, Sam Buss, Adrian Crăciun and Gabriel Istrate. *Short Proofs of the Kneser-Lovász Coloring Principle.* Information and Computation 261:2, pp. 296-310, 2019.

[2] Albert Atserias and Neil Thapen. *The Ordering Principle in a Fragment of Approximate Counting.* ACM Transactions on Computational Logic 15:4, article 29, 2014.

[3] Paul Beame, Stephen Cook, Jeff Edmonds, Russell Impagliazzo and Toniann Pitassi. *The relative complexity of NP search problems.* Journal of Computer and System Sciences 57:1, pp. 3-19, 1998.

[4] Arnold Beckmann. *A Characterisation of Definable NP Search Problems in Peano Arithmetic.* Proceedings of WoLLIC 2009, pp. 1-12, 2009.

[5] Arnold Beckmann and Samuel Buss. *Polynomial Local Search in the Polynomial Hierarchy and Witnessing in Fragments of Bounded Arithmetic.* Journal of Mathematical Logic 9:1, pp. 103-138, 2009.

[6] Arnold Beckmann and Samuel Buss. *Characterizing Definable Search Problems in Bounded Arithmetic via Proof Notations.* In Ways of Proof Theory, ONTOS Series in Mathematical Logic, pp. 65-134, 2010.

[7] Arnold Beckmann and Samuel Buss. *Improved Witnessing and Local Improvement Principle for Second-Order Bounded Arithmetic.* ACM Transactions on Computational Logic 15:1, article 2, 2014.

[8] Joshua Buresh-Oppenheim and Tsuyoshi Morioka. *Relativized NP search problems and propositional proof systems.* Proceedings of the 19th IEEE Conference on Computational Complexity (CCC), pp. 54-67, 2004.

[9] Samuel Buss. *Bounded Arithmetic.* Bibliopolis, 1986.

[10] Samuel Buss and Alan Johnson. *Propositional Proofs and Reductions between NP Search Problems.* Annals of Pure and Applied Logic 163':9, pp. 1163-1182, 2012.

[11] Samuel Buss, Leszek Aleksander Kołodziejczyk and Neil Thapen. *Fragments of approximate counting.* Journal of Symbolic Logic 79:2, pp. 496-525, 2014.

[12] Samuel Buss, Leszek Aleksander Kołodziejczyk and Konrad Zdanowski. *Collapsing modular counting in bounded arithmetic and constant depth propositional proofs.* Transactions of the AMS 367, pp. 7517-7563, 2015.

[13] Samuel Buss and Jan Krajíček. *An application of Boolean complexity to separation problems in bounded arithmetic.* Proceedings of the London Mathematical Society 3:1, pp. 1-21, 1994.

[14] Mario Chiari and Jan Krajíček. *Witnessing functions in bounded arithmetic and search problems.* Journal of Symbolic Logic 63:3, pp. 1095-1115, 1998.

[15] Alan Cobham. *The intrinsic computational difficulty of functions.* In Logic, Methodology and Philosophy of Science, Proceedings of the Second International Congress, Y. Bar-Hillel, ed., pp. 24-30, 1965.

[16] Stephen Cook. *Feasibly Constructive Proofs and the Propositional Calculus.* Proceedings of the Seventh Annual ACM Symposium on Theory of Computing, pp. 83-97, 1975.

[17] Paul Goldberg and Christos Papadimitriou. *Towards a Unified Complexity Theory of Total Functions.* Journal of Computer and System Sciences 94, pp. 167-192, 2018.

[18] Jiří Hanika. *Herbrandizing search problems in Bounded Arithmetic.* Mathematical Logic Quarterly 50:6, pp. 577-586, 2004.

[19] Johan Håstad. Computational limitations for small depth circuits. Ph.D. thesis, MIT, 1986.

[20] Emil Jeřábek. *Approximate counting in bounded arithmetic.* Journal of Symbolic Logic 72:3, pp. 959-993, 2007.

[21] Emil Jeřábek. *Approximate counting by hashing in bounded arithmetic.* Journal of Symbolic Logic 74:3, pp. 829-860, 2009.

[22] Emil Jeřábek. *Integer factoring and modular square roots.* Journal of Computer and System Sciences 82:2, pp. 380-394, 2016.

[23] David Johnson, Christos Papadimitriou and Mihalis Yannakakis. *How easy is local search?* Journal of Computer and System Sciences 37:1, pp. 79-100, 1988.

[24] Leszek Aleksander Kołodziejczyk, Phuong Nguyen and Neil Thapen. *The provably total NP search problems of weak second-order bounded arithmetic.* Annals of Pure and Applied Logic 162:2, pp. 419-446, 2011.

[25] Jan Krajíček. *No Counter-Example Interpretation and Interactive Computation.* Logic from Computer Science, MSRI volume 21, pp. 287-293, 1992.

[26] Jan Krajíček. *On the weak pigeonhole principle.* Fundamenta Mathematicae 170, pp. 123-140, 2001.

[27] Jan Krajíček, Pavel Pudlák and Alan Woods. *An exponential lower bound to the size of bounded depth Frege proofs of the pigeonhole principle.* Random Structures and Algorithms 7:1, pp. 15-39, 1995.

[28] Jan Krajíček, Alan Skelley and Neil Thapen. *NP search problems in low fragments of bounded arithmetic.* Journal of Symbolic Logic 72:2, pp. 649-672, 2007.

[29] Massimo Lauria. *A note about k-DNF resolution.* Information Processing Letters 137, pp. 33-39, 2018.

[30] Alexis Maciel, Toniann Pitassi and Alan Woods. *A new proof of the weak pigeonhole principle.* Journal of Computer and System Sciences 64:4, pp. 843-872, 2002.

[31] Nimrod Megiddo and Christos Papadimitriou. *On total functions, existence theorems and computational complexity.* Theoretical Computer Science 81:2, pp. 317-324, 1991.

[32] Tsuyoshi Morioka. *Classification of Search Problems and Their Definability in Bounded Arithmetic.* Master's thesis, University of Toronto, 2001.

[33] Moritz Müller. *Typical forcings, NP search problems and an extension of a theorem of Riis.* Annals of Pure and Applied Logic 172:4, article 102930, 2021.

[34] Moritz Müller and Ján Pich. *Feasibly constructive proofs of succinct weak circuit lower bounds.* Annals of Pure and Applied Logic 171:2, article 102735, 2020.

[35] Christos Papadimitriou. *On the complexity of the parity argument and other inefficient proofs of existence.* Journal of Computer and system Sciences 48:3, pp. 498-532, 1994.

[36] Rohit Parikh. *Existence and Feasibility in Arithmetic.* Journal of Symbolic Logic 36:3, pp. 494-508, 1971.

[37] Jeff Paris and Alex Wilkie. *Counting problems in bounded arithmetic.* Methods in mathematical logic, Springer Lecture Notes in Mathematics 1130, pp. 317-340, 1985.

[38] 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.

[39] Toniann Pitassi, Paul Beame and Russell Impagliazzo. *Exponential lower bounds for the pigeonhole principle.* Computational complexity 3:2, pp. 97-140, 1993.

[40] Pavel Pudlák. *Ramsey's theorem in bounded arithmetic.* Proceedings of Computer Science Logic '90, Lecture Notes in Computer Science vol. 533, Springer, 1991, pp. 308-317.

[41] Pavel Pudlák and Neil Thapen. *Alternating Minima and Maxima, Nash Equilibria and Bounded Arithmetic.* Annals of Pure and Applied Logic 163, pp. 604-614, 2012.

[42] Pavel Pudlák and Neil Thapen. *Random resolution refutations.* Computational Complexity 28:2, 185-239, 2019.

[43] Alan Skelley and Neil Thapen. *The provably total search problems of bounded arithmetic.* Proceedings of the London Mathematical Society 103:1, pp. 106-138, 2011.

[44] Amirhossein Tabatabai. *Computational Flows in Arithmetic.* arXiv preprint 1711.01735, 2017.

[45] Neil Thapen. *A model-theoretic characterization of the weak pigeonhole principle.* Annals of Pure and Applied Logic, vol 118, pages 175-195, 2002.