Electronic Colloquium on Computational Complexity, Report No. 84 (2003)

Relativized NP search problems and propositional proof systems

Joshua Buresh-Oppenheim Tsuyoshi Morioka

November 27, 2003

Abstract

We consider *Total Functional* **NP** (**TFNP**) search problems. Such problems are based on combinatorial principles that guarantee, through locally checkable conditions, that a solution to the problem exists in an exponentially-large domain, and have the property that any solution has a polynomial-size witness that can be verified in polynomial time. These problems can be classified according to the combinatorial principle that guarantees the existence of a solution; for example, **PPP** is the class of such problems whose totality is assured by the Pigeonhole Principle. We show many strong connections between relativized versions of these search classes and the *computational power*—in particular the proof complexity—of their underlying principles. These connections, along with lower bounds in the propositional proof systems Nullstellensatz and bounded-depth LK, allow us to prove several new relative separations among the classes **PLS**, **PPP**, **PPA**, **PPAD**, and **PPADS**.

1 Introduction

Traditionally the study of computational complexity has been largely a study of decision problems, or the problems of deciding whether the input satisfies a certain property. Consequently, search problems, or the problems of finding an object satisfying a desired property, have been studied in terms of their equivalent decision counterparts. For example, the complexity of finding a Hamiltonian cycle of a graph (if one exists) is studied indirectly via the problem of deciding if the input graph has a Hamiltonian cycle. A justification for this indirect approach is that these search and decision problems are polynomially equivalent, i.e., they are polynomial-time Turing reducible to each other.

However, when a search problem is total, i.e., every instance of it is guaranteed to have a solution, it seems to have no polynomially equivalent decision problem. Such total search problems are commonplace in computer science and mathematics: examples include optimization problems such as the problem of finding a Traveling Salesman tour that is locally optimal with respect to the 2-OPT heuristic, and the problems in game theory such as the problem of finding a Nash equilibrium given payoff matrices for two players. Thus it is important for us to understand the complexity of total search problems, and we need to study them directly.

In the papers [JPY88, Pap94b], total search problems are classified into classes according to the combinatorial principle in the finite domain that guarantees the totality of the problems. These classes contain numerous natural problems, some of which are complete. For example, **PLS**, which is the class of efficient local search heuristics, is characterized by the characterized by the parity principle "no odd-sized graph has a perfect matching"; and **PPP**, which has relevance to cryptographic hash functions, corresponds to the pigeonhole principle "there is no injective mapping from $[n+1]$ to $[n]$." The classes **PPAD** and **PPADS** are defined in a similar manner (**PPAD** was called **PSK** in [Pap94b], and it is given this name in [BCE+98]).

ISSN 1433-8092

Beame et al. [BCE+98] reformulate the search classes in terms of type-2 search problems, or search problems whose input contains not only numbers and strings (type-1 objects) but also functions and relations (type-2 objects) that are presented as oracles. This type-2 approach results in much cleaner definitions of the above search classes: each class essentially becomes a collection of the type-1 instances of a single type-2 problem. Thus the relationship among these classes can be studied through the corresponding type-2 search problems. In many cases we can obtain unconditional separations of type-2 search problems, which imply the separations of the corresponding search classes by an oracle. Since an unrelativized separation of any two search classes implies $\mathbf{P} \neq \mathbf{NP}$, such relative separations are currently the best results we can hope for. Various relative containments and separations among the above five search classes are obtained this way in [BCE+98, Mor01]. Moreover, since the complexity of the type-2 problems is directly related to the 'computational power' of the corresponding combinatorial principles, the type-2 setting also provides us with means to study various combinatorial principles in terms of their computational power, which is an interesting mathematical endeavour by itself. This paper presents new results that are obtained by further exploiting this connection.

We extend the framework of [BCE+98] into a systematic method of formulating type-2 search problems from combinatorial principles. The method is essentially as follows. Let Φ be a first-order existential sentence over an arbitrary language such that Φ holds in every finite structure, and define Q_Φ to be the corresponding type-2 search problem of finding a witness to Φ in a finite structure given as the input. For example, the type-2 problem **PIGEON** of [BCE+98] that characterizes the class **PPP** arises from the following sentence:

$$(\forall x)[\alpha(x) \neq 0] \supset (\exists x, y)[x \neq y \wedge \alpha(x) = \alpha(y)],$$

which states that, if 0 is not in the range of function α, then there must exist two elements that are mapped to the same element by α; this is the injective pigeonhole principle, which holds in every finite structure.

This paper shows that knowledge of the proof complexity of Φ reveals the relative strength of Q_Φ, where the proof complexity of Φ is measured in terms of the size of the shortest proof of the propositional translation of Φ in a given proof system. This close link between proof complexity and computational complexity allows us to derive a number of results on the relative strength of search problems by utilizing the extensive knowledge that has been accumulated in proof complexity research. Our approach is made possible by the direct connection between combinatorial principles and search problems that becomes explicit in the type-2 setting.

Main Result 1: Let Q_Φ and Q_Ψ be two type-2 search problems corresponding to the combinatorial principles Φ and Ψ. If $Q_\Phi \leq_m Q_\Psi$, then there are reductions from the propositional translation of Φ to the propositional translation of Ψ in depth-1.5 tree-like LK and in Nullstellensatz. This result can be seen as a generalization of a technique used in [BCE+98], where a relative separation is proven using a Nullstellensatz degree lower bound.

As corollaries, we obtain relative separations of search classes that have not been known, such as $\mathbf{PLS}^A \not\subseteq \mathbf{PPA}^A$. Our result generalizes the proof techniques of Beame et al. and hence it provides alternative proofs for some of their results via the proof complexity separations. Moreover, since the combinatorial principle characterizing **PPA** has low-complexity proof in Nullstellensatz, it follows that the totality of every **PPA** problems has a low-complexity proof. This is interesting because **PPA** contains the witnessing problems for the fixed point theorems of Brower, Nash, and Kakutani [Pap94b].

We also provides a partial solution for the open question whether **PLS** is contained in **PPP**.

Main Result 2: There is no 'nice reduction' from **ITERATION** (which corresponds to **PLS**) to **PIGEON** (which characterizes **PPP**).

Our third main result is a sufficient condition for Q_Φ to be nonreducible to **ITERATION**.

Main Result 3: If Φ is a combinatorial principle that does not involve the ordering relation, and if Φ fails in an infinite structure, then Q_Φ is not reducible to **ITERATION**.

This generalizes the relative separation in [Mor01], and it implies that, in a relativized world, **PLS** does not contain any of **PPP**, **PPA**, and **PPAD**. This may be interpreted as evidence that efficient local search heuristics are unlikely to exist for these classes. Main result 4 provides an alternative proof for Riis's independence criterion for the bounded arithmetic theory $S^2_2(R)$ [Rii93, Kra95].

This paper is organized as follows. Section 2 introduces basic definitions of search problems and the proof systems that we use. The search classes of [JPY88, Pap94b, BCE+98] are introduced here, and the known relative separations are stated. In Section 3 we show how to translate combinatorial principles in first-order logic into unsatisfiable propositional formulas and unsatisfiable set of polynomials. Section 4 contains our Main Results 1 and 2. Section 5 presents some of the known proof complexity separations, which imply a number of the search problem separations in Section 6. Section 7 is an exposition of Main Result 3. Section 8 contains concluding remarks and some open problems.

2 Preliminaries

Throughout this paper we write V_n to denote the set of all n-bit strings.

2.1 Search Problems

NP is the class of decision problems that are representable as $(\exists y)R(x,y)$, where R is a polynomial-time predicate such that $R(x,y)$ implies $|y| \leq p(|x|)$ for some polynomial p.

The corresponding **NP** search problem Q_R is the problem of finding, given x, y such that $R(x,y)$. The input x is called an *instance* of Q_R and any y satisfying $R(x,y)$ is called a *solution* for instance x. For every x, $Q_R(x) = \{y : R(x,y)\}$ denotes the set of solutions for instance x. Usually we omit the subscript R. We say that Q is *total* if $Q(x)$ is nonempty for all x. **TFNP** is defined to be the class of total NP search problems in [MP91] (see also [Pap94a]); the same class is called **VP** (for Verification of solutions in Polytime) in [Mor01]. A number of interesting subclasses of **TFNP** have been identified and studied: these classes are **PLS** of [JPY88], and **PPP**, **PPA**, **PPAD**, and **PPADS** of [Pap94b, BCE+98]. All of these classes contain natural problems, some of which are complete (under an appropriate notion of reducibility). We will formally define these classes below.

Beame et al. [BCE+98] generalize the notion of search problem so that the instances of search problem Q consist not only of strings, which are type-1 objects, but also functions and relations, which are type-2 objects. More formally, let R be a type-2 relation with arguments $(\alpha_1,\ldots,\alpha_k,x,y)$, where x and y are strings and for each i, $1 \leq i \leq k$, α_i is either a string function or a string relation. R defines a type-2 search problem Q_R in the usual way.

The complexity of type-2 relation, functions, and search problems is measured with respect to a Turing machine that receives the type-1 arguments on its input tape and is allowed to access the type-2 arguments as oracles [Tow90]. In particular, a type-2 function $F(\alpha_1,\ldots,\alpha_k,x)$ is said to be polynomial-time computable if it is computed by a deterministic Turing machine in time polynomial in $|x|$ with oracle access to $\alpha_1,\ldots,\alpha_k$.

2.2 Combinatorial Principles and Search Problems

Beame et al. [BCE+98] introduce several type-2 search problems that correspond to the combinatorial principles that characterize the search classes of [Pap94b]. We extend their approach into a systematic method of defining type-2 search problems from combinatorial principles that are represented as sentences of first-order logic with equality.

Let L be an arbitrary first-order language and let Φ be a sentence over L of the form

$$\Phi \equiv (\exists x_1 \ldots \exists x_k)\phi(x_1,\ldots,x_k)$$

for some quantifier-free ϕ. Let us call such sentences $\exists$-sentences. As usual, we allow the equality symbol $=$ in Φ even though we do not explicitly include it in L. Φ is interpreted in a structure $\mathcal{M}$ which defines the universe of discourse and the meaning of constants, functions, and relations of L. Some symbols of L may be designated as *built-in symbols* with which we associate predetermined interpretation.

Definition 1. *Define a* canonical structure *to be a structure such that (1) the universe of discourse is V_n for some $n \geq 1$; and (2) every* built-in *symbol of L assumes the predetermined interpretation. We abuse the notation and write V_n to denote the canonical structure with the set V_n the universe of discourse.*

Throughout this paper the only built-in symbols we use are $\leq$ and 0, and they are interpreted as the standard ordering of n-bit binary numbers and 0^n, respectively.

Assume that Φ holds in every canonical structure. Then the corresponding witness problem is the following: given a canonical structure V_n, find a tuple $\langle v_1,\ldots,v_k\rangle \in (V_n)^k$ such that $\phi(v_1,\ldots,v_k)$ holds in V_n. We formulate the witness problem as the type-2 search problem Q_Φ whose type-1 argument x specifies the universe of discourse $V_{|x|}$ and whose type-2 arguments are the functions and relations of L. Built-in symbols are not part of the type-2 arguments, since their meaning in $V_{|x|}$ is already fixed. Finally, since only the length of x is used to define $V_{|x|}$, we assume without loss of generality that the type-1 argument of Q_Φ is always of the form 1^n for $n \geq 1$.

Let us introduce the combinatorial principles that are of particular interest in the study of search problems. For readability we present them in implicational form; it is easy to see that all of them are $\exists$-sentences. Moreover, all of them hold in every canonical structure.

$$f(0) = 0 \wedge (\forall x)[x = f(f(x))] \supset (\exists x)[x \neq 0 \wedge x = f(x)] \tag{1}$$

$$(\forall x)[f(x) \neq 0] \supset (\exists x,y)(x \neq y \wedge f(x) = f(y)) \tag{2}$$

$$g(0) = 0 \wedge f(0) \neq 0 \wedge (\forall x)[x = g(f(x))] \wedge (\forall x)[x \neq 0 \supset x = f(g(x))] \supset (\exists x,y)[x \neq y \wedge f(x) = f(y)] \tag{3}$$

$$(\exists x_1,y_1,x_2,y_2)[(x_1 \neq x_2 \vee y_1 \neq y_2) \wedge f(x_1,y_1) = f(x_2,y_2)] \tag{4}$$

$$f(0) > 0 \wedge (\forall x)[f(x) \geq x] \supset (\exists x)[x < f(x) \wedge f(x) = f(f(x))] \tag{5}$$

Principle (1) states that, if the function f matches every element to either a unique partner or itself, and if 0 is matched to itself, then there exists another element that is matched to itself. This is essentially the parity principle 'no odd-sized graph has a perfect matching', and it holds in every structure whose size is even; therefore, it holds in every canonical structure. **LONELY** is the corresponding search problem.

Principle (2) states that if 0 is not in the range of f, then there exist two distinct elements that are mapped to the same element by f; this is the injective, functional pigeonhole principle, and the corresponding search problem is **PIGEON**.

Principle (3) is a weaker variant of the above pigeonhole principle. The additional assumptions essentially state that g is the inverse of f, and 0 is not in the range of f. This is the onto-pigeonhole principle, and the corresponding search problem is **OntoPIGEON**.

Principle (4) is the weak pigeonhole principle, which is similar to (2) but the domain size is the square of the range size. We call the corresponding problem **WeakPIGEON**.

Principle (5) is the iteration principle of [BK94, CK98], and we call the corresponding type-2 problem **ITERATION**. It states that, if f is nondecreasing and $f(0) > 0$, then there exist x such that $f(x) > x$ and $f(x) = f(f(x))$. Note that it contains a built-in ordering $\leq$. It is equivalent to the principle 'every dag with at least one edge has a sink'.

2.3 Reductions

Let Q be a type-2 search problem. Q can be used as an oracle in the following way. A Turing machine M presents a query to Q in the form $(\beta_1, \ldots, \beta_k, 1^m)$, where each of $\beta_1, \ldots, \beta_k$ is a polynomial-time function or relation. In the next step M receives in its answer tape some z that is a solution for $Q(\beta_1, \ldots, \beta_k, 1^m)$.

Let Q_1 and Q_2 be two type-2 search problems. We say Q_1 is *Turing reducible to* Q_2 and write $Q_1 \leq_T Q_2$ iff there exists an oracle Turing machine M that, given an instance $(\alpha_1, \ldots, \alpha_k, 1^n)$ of Q_1, outputs some $z \in Q_1(\alpha_1, \ldots, \alpha_k, 1^n)$ in polynomial-time using $\alpha_1, \ldots, \alpha_k$ and Q_2 as oracles, where each query to Q_2 is of the form $(\beta_1, \ldots, \beta_l, 1^m)$ with $m \in n^{O(1)}$ and with each β_i for each $1 \leq i \leq l$, a function or a relation that is polynomial-time computable using $\alpha_1, \ldots, \alpha_k$ as oracles.

Q_1 is *many-one reducible to* Q_2, written $Q_1 \leq_m Q_2$, if $Q_1 \leq_T Q_2$ by an oracle Turing machine that asks at most one query to Q_2. We write $Q_1 \equiv_m Q_2$ if Q_1 and Q_2 are many-one reducible to each other.

Let Q_1 and Q_2 be type-2 search problems, and assume that the type-2 arguments of Q_1 and Q_2 are $\alpha_1, \ldots, \alpha_k$ and $\beta_1, \ldots, \beta_l$, respectively. Assume that $Q_1 \leq_m Q_2$. Given an instance $(\alpha_1, \ldots, \alpha_k, 1^n)$ of Q_1, the many-one reduction composes a query $(\beta_1, \ldots, \beta_l, 1^m)$ to Q_2, where $m \leq p(n)$ for some polynomial and each β_i is polynomial-time computable using oracles $\alpha_1, \ldots, \alpha_k$.

For each $\overline{v} \in (V_m)^{arity(\beta_i)}$, where the polynomial-time algorithm for β_i gives rise to a decision tree $T^n_{\beta_i, \overline{v}}$, which encodes all possible computations of $\beta_i(\overline{v})$ in terms of $\alpha_1, \ldots, \alpha_k$. More specifically, each internal node of $T^n_{\beta_i, \overline{v}}$ is a query to some α_j with $N = 2^n$ outgoing edges, one for every possible way the query to α_j can be answered. The leaves are labeled by possible values for $\beta_i(\overline{v})$. Note that the height of $T^n_{\beta_i, \overline{v}}$ is polynomial in n and polylogarithmic in N.

These trees have the following two defining properties:
(i) given any set of equations $\{\beta_{s_1}(\overline{v_1}) = w_1, \ldots, \beta_{s_t}(\overline{v_t}) = w_t\}$ that constitute a solution to Q_2, take one path from $T^n_{\beta_{s_i}, \overline{v_i}}$ that leads to a leaf labeled w_i for each i. The queries along these t paths define a portion of $\alpha_1, \ldots, \alpha_k$. This portion constitutes a solution to Q_1.
(ii) each tree is complete in the sense that whenever α_j is queried at an internal node, there is an edge fanning out of that node for each possible value of α_j.

We will use the above properties in the proof of our main theorem

Definition 2. *Let Q be a type-2 search problem. Then $C(Q)$ is defined as*

$$C(Q) = \{Q' : Q' \text{ is type-1 and } Q' \leq_m Q\} \cap \mathbf{TFNP}.$$

Now we are ready to formulate the search classes of [JPY88, P94] in terms of the type-2 search problems.

PPA stands for Polynomial Parity Argument, and it is characterized as $\mathbf{PPA} = C(\mathbf{LONELY})$. This class contains the problems of finding various economic equilibria, some of which are complete. Polynomial Pigeonhole Principle is defined as $\mathbf{PPP} = C(\mathbf{PIGEON})$, and it has relevance in the study of cryptographic hash functions. These two definitions are from [BCE+98]. **PPAD** is an analogue of **PPA** in the directed graphs, and hence the name (**D** is for 'Directed'). $\mathbf{PPAD} = C(\mathbf{OntoPIGEON})$. Beame et al. characterizes **PPAD** by different combinatorial principles, but they are equivalent to the onto-pigeonhole principle, which we use in this paper. Polynomial Local Search is the class of optimization problems for which efficient local-search heuristics exist, and $\mathbf{PLS} = C(\mathbf{ITERATION})$. This characterization is essentially in [CK98] in the context of bounded arithmetic. Also [Mor01] contains a direct proof in a complexity theoretic setting. For more information on these classes, see [JPY88, Yan97] for **PLS** and [Pap94b] for the other classes.

Theorem 3. *[CIY97] Let Q_1 and Q_2 be type-2 search problems defined by $\exists$-sentences. The following are equivalent: (i) $Q_1 \leq_m Q_2$; (ii) for all oracles A, $C(Q_1)^A \subseteq C(Q_2)$; and (iii) there exists a generic oracle G such that $C(Q_1)^G \subseteq C(Q_2)^G$.*

Theorem 4. *[BCE+98] The following hold: (i)* **OntoPIGEON** $\leq_m$ **LONELY***; (ii)* **OntoPIGEON** $\leq_m$ **PIGEON***; (iii)* **LONELY** *and* **PIGEON** *are incomparable, i.e., neither is many-one reducible to the other.*

The above result completely characterizes the relationship among **PPAD**, **PPA**, and **PPP**. However, **PLS** is not discussed in [BCE+98], and progress for resolving the relative complexity of **PLS** is made in [Mor01]:

Theorem 5. *[Mor01]* **OntoPIGEON** *is not many-one reducible to* **ITERATION**.

Thus, **PLS** contains none of **PPP**, **PPA**, and **PPAD** in a relativized world. However, it was still unresolved whether **PLS** is contained in any of the other classes, and we will present below solutions to some of these open problems.

2.4 Proof Systems

We consider two propositional proof systems in this paper. The first is called *propositional LK* or *sequent calculus*. Let $A_1,\ldots,A_k,B_1,\ldots,B_\ell$ be propositional formulas over some set of variables $\overline{X}$ and the connectives $\neg,\vee,\wedge$. A sequent is a syntactic object of the form

$$A_1,\ldots,A_k \longrightarrow B_1,\ldots,B_\ell,$$

with the intended meaning

$$A_1 \wedge \ldots \wedge A_k \supset B_1 \vee \ldots \vee B_\ell.$$

The *depth* of such a sequent is the maximum over the depths of each of the formulas. LK usually includes the following rules (A,B are formulas, $\Delta,\Gamma,\Delta',\Gamma'$ are sets of formulas):

Logical Axiom: $\frac{}{A \longrightarrow A}$

Weakening: $\frac{\Gamma\longrightarrow\Delta}{\Gamma'\longrightarrow\Delta'}$ for $\Gamma \subset \Gamma', \Delta \subset \Delta'$

Contraction: $\frac{A,A,\Gamma\longrightarrow\Delta}{A,\Gamma\longrightarrow\Delta}, \frac{\Gamma\longrightarrow\Delta,A,A}{\Gamma\longrightarrow\Delta,A}$

Exchange: $\frac{A,B,\Gamma\longrightarrow\Delta}{B,A,\Gamma\longrightarrow\Delta}, \frac{\Gamma\longrightarrow\Delta,A,B}{\Gamma\longrightarrow\Delta,B,A}$

$\wedge$-introduction: $\frac{A,B,\Gamma\longrightarrow\Delta}{A\wedge B,\Gamma\longrightarrow\Delta}, \frac{\Gamma\longrightarrow\Delta,A \qquad \Gamma\longrightarrow\Delta,B}{\Gamma\longrightarrow\Delta,A\wedge B}$

$\vee$-introduction: $\frac{\Gamma\longrightarrow\Delta,A,B}{\Gamma\longrightarrow\Delta,A\vee B}, \frac{A,\Gamma\longrightarrow\Delta \qquad B,\Gamma\longrightarrow\Delta}{A\vee B,\Gamma\longrightarrow\Delta}$

$\neg$-introduction: $\frac{\Gamma\longrightarrow\Delta,A}{\neg A,\Gamma\longrightarrow\Delta}, \frac{A,\Gamma\longrightarrow\Delta}{\Gamma\longrightarrow\Delta,\neg A}$

Cut: $\frac{A,\Gamma\longrightarrow\Delta \qquad \Gamma\longrightarrow\Delta,A}{\Gamma\longrightarrow\Delta}$

An LK derivation of B from $A_1,\dots,A_k$ is a sequence of sequents $S_1,\dots,S_m$ where S_m is $\longrightarrow B$ and each S_i either follows via one of the above rules from earlier sequents or is $\longrightarrow A_j$ for some j. A *refutation* is a derivation of $\longrightarrow$. The *size* of a derivation is the sum of the sizes of all formulas mentioned in the derivation, while the depth is the maximum depth of any sequent in the derivation. A derivation is called *tree-like* if each sequent is used at most once to derive a new sequent. LK is well-known to be derivationally sound and complete.

Nullstellensatz is an algebraic proof system. Let F be a field and let $\overline{X}$ be a set of variables. Given polynomials $q_1,\dots,q_m,p \in F[\overline{X}]$, a Nullstellensatz derivation of p from $q_1,\dots,q_m$ is another set of polynomials $r_1,\dots,r_m \in F[\overline{X}]$ such that

$$r_1q_1+\ldots+r_mq_m = p,$$

identically. A *refutation* is a derivation of 1. The *degree* of a Nullstellensatz derivation is the maximum over the degrees of r_iq_i. Nullstellensatz is derivationally sound and complete over any field F in the sense that p can be derived iff it is in the ideal generated by $q_1,\dots,q_m$.

3 Propositional Translations of Type-2 Problems

Let Φ be an $\exists$-sentence over language L. We say that Φ is *basic* if its quantifier-free part is in DNF and contains no nesting of symbols of L. More specifically, if Φ is basic then every atomic formula in Φ is of the form $R(\overline{x})$, $y = f(\overline{x})$, or $x = y$, where R is a predicate symbol and f is a function symbol.

For a type-2 search problem Q_Φ defined by a basic $\exists$-sentence Φ, $CNF(Q_\Phi,n)$ for each n will be the *unsatisfiable* propositional CNF which (falsely) states that Q_Φ is not total. It is the result of a standard translation of $\neg\Phi$ into propositional CNF formulas due to [PW85]. The following is a more detailed description of the translation: There will be a set of variables in $CNF(Q_\Phi,n)$ for each type-2 argument α for Q_Φ. If α is an m-ary relation, then there will be a propositional variable $X^\alpha_{\overline{v}}$ for each m-tuple $\overline{v}$ in the domain of α. If α is a function, we add propositional variables for the relation $graph(\alpha)$: $X^\alpha_{\overline{v}w}$ for each $\overline{v}$ in the domain of α and each w in the target of α.

More specifically, $CNF(Q_\Phi,n)$ is formed as

$$\bigwedge_{x_1,\dots,x_k\in V_n} \phi'(x_1,\dots,x_k),$$

where ϕ' is the CNF that is the negation of ϕ, and where each atom is replaced by its corresponding propositional variable or propositional constants. If an atom contains $=$ or any built-in predicate or function, then it is replaced with either *true* or $false$, depending on their truth value in the canonical structure V_n.

In addition, for each m-ary function in Q_Φ and each m-tuple, we add clauses to $CNF(Q_\Phi,n)$ stating that the function is well-defined on that input: for each $\overline{v}$ in the domain of α, define

$$def_n(X^\alpha_{\overline{v}}) =_{syn} \bigvee_{w\in V_n} X^\alpha_{\overline{v}w}$$

and

$$singledef_n(X^\alpha_{\overline{v}}) =_{syn} \bigwedge_{u,w\in V_n} \neg X^\alpha_{\overline{v}u} \vee \neg X^\alpha_{\overline{v}w}$$

Notice that all the clauses in $CNF(Q_\Phi,n)$ have constant size except the def clauses, which have size $|V_n| = N$. The number of clauses in the CNF is polynomial in N. Let $CNF(Q_\Phi)$ be the family of formulas $\{CNF(Q_\Phi,n)\}_{n\in\mathbb{N}}$.

If we start with Φ that is not basic, we modify it to obtain a basic sentence as follows. If Φ contains a nesting of symbols, say $y = f(g(x))$, then replace it with $(\exists z)[z = g(x) \wedge y = f(z)]$. Treat the other cases $f(x) = g(y)$ and $R(f(x))$ in a similar way. After all atoms are made basic, then make the whole sentence prenex with the quantifi er-free part in DNF. Let Φ' be the resulting $\exists$-sentence. It is clear that $Q_\Phi \equiv_m Q_{\Phi'}$.

We will also need translations of search problems into polynomials. We start with $CNF(Q_\Phi)$: consider those clauses of $CNF(Q_\Phi)$ that correspond to solutions of Q_Φ (call these clauses the *solution clauses*). Each one can be converted into a polynomial in the usual way: each literal forms a linear factor of the polynomial, where a positive literal x becomes a factor $1-x$ and a negative literal $\neg x$ becomes a factor x. In addition, we add polynomials forcing each variable x to take on 0/1 values: $x - x^2$.

We insist each type-2 function α is well-defi ned on each of its domain elements v with the following polynomials:

$$polydef_n(X^\alpha_{\bar{v}}) = \sum_{w \in V_n} X^\alpha_{\bar{v}w} - 1,$$

$$polysingledef_n(X^\alpha_{\bar{v}}) = \{X^\alpha_{\bar{v}w} X^\alpha_{\bar{v}w'} \mid \; w, w' \in V_n;\; w \neq w'\}.$$

Call the set of all the above polynomials $poly(Q_\Phi, n)$. Let $poly(Q_\Phi)$ be the family $\{poly(Q_\Phi, n)\}_{n \in \mathbb{N}}$.

4 Search problem reductions and proof complexity reductions

4.1 Bounded-depth LK

Definition 6. *Let $\mathcal{S}, \mathcal{R}$ be two families of propositional formulas. We say $\mathcal{S}$ has a bounded-depth LK refutation reduction to $\mathcal{R}$, written $\mathcal{S} \leq_{bd-LK} \mathcal{R}$, we can derive a substitution instance $\mathcal{R}'$ of $\mathcal{R}$ from $\mathcal{S}$ in quasi-polynomial-size bounded-depth LK.*

Assume we have a search problem reduction from an instance of Q_Φ of size n to an instance of Q_Ψ of size m, and that the reduction gives rise to decision trees $\mathcal{T} = \{T^n\}$. Call the set of variables of $CNF(Q_\Phi, n)$ $\bar{X}$, and the set of variables of $CNF(Q_\Psi, m)$, $\bar{Y}$. Consider a $Y \in \bar{Y}$ corresponding to the equation $\beta(\bar{v}) = w$, where $\bar{v} \subset V_m$, $w \in V_m$ and β is one of Q_Ψ's type-2 objects. Let $\mathcal{T}(Y)$ be the DNF over $\bar{X}$ that encodes all paths in $T^n_{\beta,\bar{v}}$ that lead to w. For any formula A over $\bar{Y}$, let $\mathcal{T}(A)$ be a substitution instance of A where $\mathcal{T}(Y)$ is substituted for each variable Y.

Theorem 7. *Let Q_Φ, Q_Ψ be two type-2 NP search problems defined by first order sentences. If $Q_\Phi \leq_m Q_\Psi$, then $CNF(Q_\Phi) \leq_{bd-LK} CNF(Q_\Psi)$. In fact, the derivations are tree-like and all formulas mentioned can be represented as decision trees over type-2 objects of Q_Φ with depth polynomial in n.*

Proof. Assume $CNF(Q_\Phi, n)$ is reducible to $CNF(Q_\Psi, m)$ by trees $\mathcal{T}$. We derive $\mathcal{T}(CNF(Q_\Psi, m))$ from $CNF(Q_\Phi, n)$ Let β be one of Q_Ψ's type-2 objects and let $\bar{v} \subset V_m$.

(i) We fi rst show how to derive $D =_{syn} \mathcal{T}(def(Y^\beta_{\bar{v}}))$. D is just $disj(T)$, the disjunction of all paths in $T = T^{\beta,v}_n$. Call the paths $P_1, \ldots, P_k$. Let $\alpha_1(\bar{v}_1), \ldots, \alpha_\ell(\bar{v}_\ell)$ be the queries to Q_Φ that appear in T. We derive the sequent

$$def(X^{\alpha_i}_{\bar{v}}), \ldots, def(X^{\alpha_\ell}_{\bar{v}_\ell}) \longrightarrow D,$$

and then cut the formulas on the left. The derivation will be an upside-down copy of T itself. For a given leaf, let P_i be the path that leads to that leaf. In the proof, the leaf will be labeled with the sequent

$$P_i \longrightarrow P_1, \ldots, P_k.$$

In general, for a node x of T, let P be the path from the root to x and let $\alpha_{i_1}(\bar{v}_{i_1}),\ldots,\alpha_{i_j}(\bar{v}_{i_j})$ be the queries in the subtree of T rooted at x. In the proof, node x will be labeled by

$$P, def(X^{\alpha_{i_1}}_{\bar{\gamma}_1}),\ldots,def(X^{\alpha_{i_k}}_{\bar{\gamma}_j}) \longrightarrow P_1,\ldots,P_k.$$

This sequent is derived from the sequent at the children of x by $\vee$-introduction. Since T has quasi-poly size, so will this derivation.

(ii) Now consider $\mathcal{T}(singledef(Y^{\beta}_{\bar{v}}))$. We show how to derive $D =_{syn} \neg\mathcal{T}(Y^{\beta}_{\bar{v},w}) \vee \neg\mathcal{T}(Y^{\beta}_{\bar{v},w'})$, for each pair $w \neq w'$. The formula $\mathcal{T}(Y^{\beta}_{\bar{v},w})$ is just a disjunction of all paths in $T = T^{\beta,\bar{v}}$ that lead to w, call them $P_1,\ldots,P_\ell$. Similarly, $\mathcal{T}(Y^{\beta}_{\bar{v},w'})$ is the disjunction of all paths in T that lead to w': $R_1,\ldots,R_k$. Any pair of paths P_i,R_j must differ on at least one query. Say that on P_i, we have $\alpha_{ij}(\bar{v}_{ij}) = w_{ij}$ and on R_j, $\alpha_{ij}(\bar{v}_{ij}) = w'_{ij}$. Begin with the sequents

$$P_i,R_j \longrightarrow X^{\alpha_{ij}}_{\bar{\gamma}_j w_{ij}} \qquad \text{and} \qquad P_i,R_j \longrightarrow X^{\alpha_{ij}}_{\bar{\gamma}_j w'_{ij}},$$

for each i,j. By $\neg$- and $\vee$- introduction, we get

$$\neg X^{\alpha_{ij}}_{\bar{\gamma}_j w_{ij}} \vee \neg X^{\alpha_{ij}}_{\bar{\gamma}_j w'_{ij}}, P_i, R_j \longrightarrow .$$

By weakenings and $\vee$-introductions, we get

$$\{\neg X^{\alpha_{ij}}_{\bar{\gamma}_j w_{ij}} \vee \neg X^{\alpha_{ij}}_{\bar{\gamma}_j w'_{ij}}\}_{ij}, \bigvee_i P_i, \bigvee_j R_j \longrightarrow .$$

Finally, by $\neg$-introduction, we get

$$\{\neg X^{\alpha_{ij}}_{\bar{\gamma}_j w_{ij}} \vee \neg X^{\alpha_{ij}}_{\bar{\gamma}_j w'_{ij}}\}_{ij} \longrightarrow D.$$

This derivation again has quasi-polynomial size.

(iii) Finally, consider a clause C of $CNF(Q_\Psi)$ that says that one of the criteria for solution fails. We derive $D = \mathcal{T}(C)$. Assume C mentions variables $Y^{\beta_1}_{\bar{\gamma},w_1},\ldots,Y^{\beta_k}_{\bar{\gamma},w_k}$. For each $1 \leq i \leq k$, let $T_i = T^{\beta_i}_{\bar{\gamma}}$. Let $\mathcal{P}$ be all k-tuples of paths $\{(P_1,\ldots,P_k) \mid P_1 \in T_1,\ldots,P_k \in T_k\}$. If $P \in \mathcal{P}$ violates C, then it must contain a solution to Q_ϕ. We can derive $disj(\mathcal{P})$, the disjunction of all elements of $\mathcal{P}$, just as we derived $disj(T)$ in (i). The inconsistent tuples can be cut out; so can the those tuples that violate C, as follows: any P that violates C must also violate a clause B of $CNF(Q_\phi)$, which rules out some potential solution to Q_ϕ. Starting from $\longrightarrow B$, we get the sequent $B' \longrightarrow$, where B' is a set of literals that are the negations of those literals in B. But B' must be a subset of P. Therefore, we can cut out P from $\longrightarrow \mathcal{P}$. Now we have simply

$$\longrightarrow \mathcal{P}',$$

where $\mathcal{P}'$ is the set of all consistent paths in $\mathcal{P}$ that satisfy C. Write C as $l_1 \vee \ldots, \vee l_k$, where each l_i is either $Y^{\beta_i}_{\bar{\gamma},w_i}$ or its negation. If l_i is a negative literal, let $\mathcal{T}^{comp}(l_i)$ be the disjunction of all the paths in T_i that don't lead to w_i. Let $g(l_i) = \mathcal{T}(l_i)$ if l_i is positive and $\mathcal{T}^{comp}(l_i)$ otherwise. For each path $P \in \mathcal{P}'$, there is an i such that P contains one of the paths in $g(l_i)$. Therefore, it is easy to derive

$$\longrightarrow g(l_1),\ldots,g(l_k)$$

from

$$\longrightarrow \mathcal{P}'.$$

Finally, using $\longrightarrow disj(T_i)$ and the cut rule, we derive

$$\longrightarrow \mathcal{T}(l_1),\ldots,\mathcal{T}(l_k),$$

which is D itself. Again this derivation has quasi-polynomial size. $\square$

4.2 Nullstellensatz

We saw above that a poly-time reduction between search problems yields a quasi-poly-size bounded-depth LK reduction between the corresponding propositional formulas. Here we show a similar connection to the Nullstellensatz system.

Definition 8. *Let F be a field and let $\overline{X}$ and $\overline{Y}$ be infinite sets of variables. Let P_1 be an infinite family of finite subsets of $F[\overline{X}]$ and let P_2 be an infinite family of finite subsets of $F[\overline{Y}]$. We say that P_1 has a degree-d Nullstellensatz reduction to P_2 ($P_1 \leq_{HN(d)} P_2$), if, for any $A \in P_1$ there is a $B \in P_2$ and a set of polynomials $\overline{f_Y} = \{f_Y\}_{Y \in \overline{Y}} \subset F[\overline{X}]$ such that each polynomial in $B(\overline{f_Y}/\overline{Y})$ has a degree-d Nullstellensatz derivation from A. $B(\overline{f_Y}/\overline{Y})$ is just the result of replacing each variable Y in each polynomial of B by f_Y.*

Theorem 9. *Let Q_Φ, Q_Ψ be two type-2 NP search problems defined by first order sentences. If $Q_\Phi \leq_m Q_\Psi$, then $poly(Q_\Phi) \leq_{HN(d)} poly(Q_\Psi)$ for some d that is polynomial in n over any field.*

Proof. Assume we have a search problem reduction from an instance of Q_Φ of size n to an instance of Q_Ψ of size m, and that the reduction is given by decision trees $\mathcal{T} = \{T^n\}$. We present a Nullstellensatz reduction from $poly(Q_\Phi, n)$ to $poly(Q_\Psi, m)$. Call the set of variables of $poly(Q_\Phi, n)$ $\overline{X}$, and the set of variables of $poly(Q_\Psi, m)$ $\overline{Y}$. Consider a $Y \in \overline{Y}$ corresponding to the equation $\beta(\overline{v}) = w$, where $\overline{v} \subset V_m, w \in V_m$ and β is one of Q_Ψ's type-2 objects. We substitute for Y a polynomial $\mathcal{T}(Y)$ that encodes all paths in $T^n_{\beta,\overline{v}}$ that lead to w in the following manner: given one such path p, consider each query on p to one of Q_Φ's type-2 objects α. If α is a function and p insists that $\alpha(\overline{v'}) = w'$, then the polynomial t_p will include the factor $X^\alpha_{\overline{v'}w'}$. If α is a relation and p insists that $\alpha(\overline{v'})$ is true, then t_p will include the factor $X^\alpha_{\overline{v}}$; while if p insists that $\alpha(v')$ is false, t_p will include the factor $1 - X^\alpha_{\overline{v}}$. Essentially, t_p is a polynomial which is 0 on assignments that deviate from p at some point. The polynomial $\mathcal{T}(Y)$ is simply the sum of the polynomials t_p for each such path p. Note that the degree of $\mathcal{T}(Y)$ is polynomial in n.

We now claim that every polynomial in $\mathcal{T}(poly(Q_\Psi, m))$ has a low-degree Nullstellensatz derivation from $poly(Q_\Phi, n)$. $\mathcal{T}(poly(Q_\Psi, m))$ is the result of replacing all variables Y in all polynomials of $poly(Q_\Psi, m)$ by $\mathcal{T}(Y)$. Consider each type of polynomial in $poly(Q_\Psi, m)$ in turn. Observe that the solution clauses of Q_Φ and Q_Ψ enumerate the solutions of each of the search problems. If $g(\overline{Y})$ is a polynomial in $poly(Q_\Psi, m)$ that corresponds to a solution clause, then each term of $\mathcal{T}(g(\overline{Y}))$ includes as a factor a polynomial corresponding to a solution clause of Q_Φ. Since g has constant degree, each term can certainly be derived from $poly(Q_\Phi, n)$ in $poly(n)$-degree Nullstellensatz.

Now let $g \in poly(Q_\Psi, m)$ be one of the 0/1 constraint polynomials: $Y - Y^2$. If X (alternatively, $1 - X$) is a factor in one of the t_p's of $\mathcal{T}(Y)$, then $X - X^2$ is a factor of $t_p - t_p^2$. Now we just have to worry about generating polynomials of the form $t_p t_{p'}$ for $t_p, t_{p'}$ in $\mathcal{T}(Y)$: the paths p and p' are two different paths in the same decision tree. Consider the fi rst place where p and p' differ. If that place queries a type-2 function α of Q_Φ on input $\overline{u}$, then something in $polysingledef_n(X^\alpha_{\overline{u}})$ is a factor of $t_p t_{p'}$. Otherwise, if that place queries a type-2 relation α of Q_Φ on input u, then $X^\alpha_u - (X^\alpha_u)^2$ is a factor of $t_p t_{p'}$. Again, since g has constant degree, $\mathcal{T}(g(\overline{Y}))$ can be derived in polynomial degree.

If $g \in polysingledef_m(Y^\beta_{\overline{v}})$ for some type-2 function β and $\overline{v} \subset V_m$, then g looks like $Y_1 Y_2$, where $Y_1 = Y^\beta_{\overline{v}w}$ and $Y_2 = Y^\beta_{\overline{v}w'}$, $w \neq w'$. Choose any t_p in the sum $\mathcal{T}(Y_1)$ and any $t_{p'}$ in $\mathcal{T}(Y_2)$. The paths p and p' are two different paths in the same decision tree, so the situation is the same as above.

Finally, consider the case where $g = polydef_m(Y^\beta_{\overline{v}})$. Then $\mathcal{T}(g(\overline{Y}))$ is the sum of the t_p's for all paths p in the tree $T = T^m_{\beta,\bar{v}}$. We prove by induction on the height of T that

$$s(T) = \sum_{p \in T} t_p - 1$$

has a Nullstellensatz derivation from $poly(Q_\Phi, n)$ of degree $height(T)$. If $height(T) = 0$, then we consider T to have one path p of length 0 and let $t_p = 1$. Hence $s(T) = 0$. Otherwise, let T have height $k > 0$. Consider the tree T', the subtree of T where every path from the root is truncated at length $k-1$. By induction, $s(T') - 1$ has a Nullstellensatz derivation of degree $k-1$. Consider any leaf l of T' that is not a leaf of T and assume it queries type-2 object α on element $\bar{u}$ in T. Let T'_l be the tree T' with every T-child of l added on. If α is a function, then $s(T'_l) - s(T') = polydef_n(X^{\alpha}_{\bar{u}})t_{p_l}$, where p_l is the path from the root to l. If α is a relation, then $s(T'_l) - s(T') = 0$. We know $s(T) - s(T')$ is just the sum of $s(T'_l) - s(T')$ for all such leaves l. Hence $s(T)$ has a degree k Nullstellensatz derivation and $\mathcal{T}(g(\bar{Y}))$ has a polynomial degree Nullstellensatz derivation. ☐

5 Proof Complexity Separations

In this section we show a number of proof complexity separations which, together with Theorems 7 and 9, imply separations of type-2 search problems. Note that the CNF formulas $CNF(\textbf{PIGEON})$, $CNF(\textbf{WeakPIGEON})$, $CNF(\textbf{LONELY})$, and $CNF(\textbf{WeakPIGEON})$ are equivalent to the CNF formulas whose proof complexity have been studied extensively. $CNF(\textbf{ITERATION})$ is equivalent to the *housesitting principle* of [CEI96, Bus98b].

Lemma 10. *The following separations hold in bounded-depth LK:*
(a) $CNF(\textbf{PIGEON}) \not\leq_{bd-LK} CNF(\textbf{WeakPIGEON})$.
(b) $CNF(\textbf{LONELY}) \not\leq_{bd-LK} CNF(\textbf{PIGEON})$.
(c) $CNF(\textbf{PIGEON}) \not\leq_{bd-LK} CNF(\textbf{ITERATION})$.
(d) $CNF(\textbf{LONELY}) \not\leq_{bd-LK} CNF(\textbf{ITERATION})$.

Proof. [PBI93, KPW95] show that $CNF(\textbf{PIGEON})$ requires exponential-size refutations in any bounded-depth system. [BP96] show (b), and hence $CNF(\textbf{LONELY})$ requires exponential-size bounded-depth LK refutations. On the other hand, [MPW00] show that $CNF(\textbf{WeakPIGEON})$ has quasi-poly-size 0.5-depth LK refutations, and Lemma 11 below shows that $CNF(\textbf{ITERATION})$ has poly-size tree-like resolution refutations. ☐

Lemma 11. $CNF(\textbf{ITERATION})$ *has poly-size tree-like resolution refutation.*

Proof. Fix arbitrary $n \in \mathbb{N}$ and let $N = 2^n$. $CNF(\textbf{ITERATION}, n)$ consists of the following clauses:
(i) $\neg X_{0,0}$
(ii) $\neg X_{i,j}$ for all i, j such that $j < i$
(iii) $\neg X_{i,j} \vee \neg X_{j,j}$ for all i, j such that $i < j$
(iv) $\bigvee_{0 \leq j \leq N-1} X_{i,j}$ for every i
(v) $\neg X_{i,j} \vee \neg X_{i,k}$ for all i, j, k with $j \neq k$
For every $i \geq 1$, define A_i to be the clause $\bigvee_{j \geq i} \neg X_{j,j}$. A_1 is derivable from clauses (i), (iii), and (iv) for $i = 0$. Similarly, for every $i \geq 1$, the clause $X_{i,i} \vee A_{i+1}$ is derived using (ii), (iii), and (iv). Thus, for every $i \geq 2$, A_i is derived by resolving A_{i-1} and $X_{i-1,i-1} \vee A_i$ on $P_{i-1,i-1}$. Finally, the empty clause is derived from $A_n = \neg P_{N,N}$ and $P_{N,N}$, which is derived from (ii) and (iv). ☐

Lemma 12. *The following separations hold for degree-d Nullstellensatz whenever d is polynomial in n:*
(a) $poly(\textbf{PIGEON}) \not\leq_{HN(d)} poly(\textbf{OntoPIGEON})$ *over any field F.*
(b) $poly(\textbf{ITERATION}) \not\leq_{HN(d)} poly(\textbf{OntoPIGEON})$ *over any field F.*
(c) $poly(\textbf{PIGEON}) \not\leq_{HN(d)} poly(\textbf{LONELY})$ *over any field F of characteristic* 2.
(d) $poly(\textbf{ITERATION}) \not\leq_{HN(d)} poly(\textbf{LONELY})$ *over any field F of characteristic* 2.

Proof. [BCE+98, Raz98] prove that $poly(\textbf{PIGEON})$ requires $\Omega(N)$-degree Nullstellensatz refutations over any field. [CEI96, Bus98b] prove the same for $poly(\textbf{ITERATION})$ (they call the principle "housesitting").

On the other hand, $poly(\textbf{OntoPIGEON})$ has constant-degree Nullstellensatz refutations over any field. We have the following polynomials (let X_{ij} say that pigeon i maps to hole j and let Y_{ij} say that hole i maps to pigeon j for $0 \leq i, j < N$):
(i) $(\sum_{j=0}^{N-1} X_{ij}) - 1$ for all i
(ii) $(\sum_{j=0}^{N-1} Y_{ij}) - 1$ for all $i \neq 0$
(iii) X_{i0} for all i
(iv) $X_{ij}(1 - Y_{ji})$ for any i, j
(v) $Y_{ij}(1 - X_{ji})$ for any i, j
(vi) $X_{ij}X_{ij'}$ for any $i, j \neq j'$
Begin by converting each Y_{ij} in (ii) to X_{ji} using (iv) and (v). Now sum up all polynomials in (i) and subtract all polynomials in (ii). What remains is $(\sum_{i=0}^{N} X_{i0}) + 1$. Now we can simply cancel each X_{i0} using (iii).

Finally, $poly(\textbf{LONELY})$ has constant-degree Nullstellensatz refutations over characteristic 2. We have the following polynomials (let X_{ij} say that node i maps to node j for $0 \leq i, j < N$):
(i) $X_{ij} - X_{ij}X_{ji}$ for all $i \neq j$
(ii) X_{ii} for all $i \neq 0$
(iii) $1 - X_{00}$
(iv) $\sum_{j=0}^{N-1} X_{ij}) - 1$ for any i
(v) $X_{ij}X_{ij'}$ for any $i, j \neq j'$
Begin by summing up all polynomials in (i), (ii) and (iv): this yields $(\sum_{i=1}^{N-1} X_{0i}) + 1$. If we add $X_{0j}X_{00} + X_{0j}(1 - X_{00})$ to this, we get simply 1. $\square$

6 Search Problem Separations

The following theorem states many of the separations that follow directly from Theorems 7 and 9, and Lemmas 10 and 12:

Theorem 13. *(a) ([BCE+98])* **PIGEON** $\not\leq_m$ **LONELY**
(b) ([BCE+98]) **LONELY** $\not\leq_m$ **PIGEON**
(c) **PIGEON** $\not\leq_m$ **WeakPIGEON**
(d) ([BCE+98]) **PIGEON** $\not\leq_m$ **OntoPIGEON**
(e) **ITERATION** $\not\leq_m$ **LONELY**
(f) [Mor01] **LONELY** $\not\leq_m$ **ITERATION**
(g) [Mor01] **PIGEON** $\not\leq_m$ **ITERATION**

By Theorem 3, this implies relative separations of all the corresponding search classes.

To (almost) complete the characterization of **PLS**, we prove a slightly weaker separation of **ITERATION** from **PIGEON**:

Definition 14. *We say that Q_1 is nicely reducible to Q_2 if $Q_1 \leq_m Q_2$ and, any instance of Q_1 which contains exactly one solution is reduced to an instance of Q_2 that contains exactly one solution.*

Note that all common examples of reductions are nice reductions. In fact, they almost always preserve the number of solutions in general. Nice reductions are, ostensibly, much less restricted than what [BCE$^+$98] call *strong reductions*.

Lemma 15. *If* **ITERATION** *is nicely reducible to* **PIGEON**, *then* **ITERATION** *is reducible to* **OntoPIGEON**.

Proof. Consider **ITERATION** on a structure of size $N = 2^n$, defined by the type-2 function $succ()$. The corresponding instance of PIGEON has size $M \leq 2^{poly(n)}$, and is defined by the function β. Let $\mathcal{T} = \{T^n_{\beta,p_i}\}_{i=0}^{M-1}$ be the nice reduction from **ITERATION** to **PIGEON**. Begin by creating $\mathcal{T}'$: augment $\mathcal{T}$ so that whenever a tree T_{β,p_i} queries the successor of node x in the **ITERATION** instance, it immediately queries $succ(succ(x))$ whenever $succ(x) > x$. Now prune all branches of these trees that contain a solution to the **ITERATION** instance. At this point, given any path π in T_{β,p_i} and any path π' in T_{β,p_j} such that both paths lead to hole h_k, it must be the case that π and π' are inconsistent. Lemma 4 of [BCE$^+$98] describes how to build a forest of trees $\mathcal{H} = \{H_{\beta,h_i}\}_{i=1}^{M-1}$ such that each tree has height at most polynomial in n and H_{β,h_i} determines which pigeon, if any, maps to hole h_i. If we find that no pigeon maps to hole h_i, we label the leaf by pigeon 0.

We now have the appropriate objects, namely $\mathcal{T}$ and $\mathcal{H}$, to pass to an oracle for **OntoPIGEON**. This oracle will return (1) pigeons p_i and p_j that collide, (2) a pigeon p_i that maps to hole h_0, (3) a pigeon p_i that maps to hole h_k, but hole h_k maps to pigeon p_j, or (4) a hole h_i that maps to pigeon p_k, but p_k maps to hole h_j. Cases (1) and (2) have nothing to do with $\mathcal{H}$, so we can find a solution to **ITERATION** by the correctness of $\mathcal{T}$. In case (3), it must be that p_i and p_j collide under $\mathcal{T}$, so again we can find a solution to **ITERATION**. Finally, case (4) can arise only when $k = 0$ and h_i is left empty by $\mathcal{T}$. Assume that the pertinent path, π, in tree H_{β,h_i} does not reveal a solution to the **ITERATION** instance, otherwise we are done. Create an instance of **ITERATION** that is consistent with π and contains only one solution. Since $\mathcal{T}$ is a nice reduction, the corresponding instance of **PIGEON** has exactly one solution. Hence, there should be no hole, except perhaps h_0, that remains empty. Therefore H_{β,h_i} must have been incorrect, so $\mathcal{T}$ must have been incorrect. $\square$

Theorem 16. **ITERATION** *is not nicely reducible to* **PIGEON**.

Proof. This follows from Lemmas 15 and 12, and Theorem 9. $\square$

7 A Separation Criterion for PLS

We now present a sufficient condition for separating a search Q from **ITERATION**. The condition generalizes all of the relative separations from **ITERATION** that appear in this work and in [CK98, Mor01]. Note that the conclusion of Theorem 17 implies that $C(Q)^A \not\subseteq$ **PLS**^{A} for some oracle A:

Theorem 17. *Let Φ be an $\exists$-sentence over a language L with no built-in predicate and no built-in function. If Φ fails in an infinite structure, then*

$$Q_\Phi \not\leq_T \textbf{ITERATION}.$$

Theorem 17 is similar to Theorem 11.3.1 of [Kra95] below.

Theorem 18. *[Kra95] Let Φ be a $\exists\forall$-sentence over a relational language L without $\leq$. If Φ fails in an infinite structure, then the type-2 problem Q_Φ is not in type-2* $\mathbf{FP}^{\mathbf{NP}}$.

Theorems 17 and 18 are incomparable. Since **ITERATION** is in type-2 $\mathbf{FP}^{\mathbf{NP}}$ trivially, the consequence of Theorem 18 is stronger than that of our Theorem 17. However, it does not apply to search problems defined by $\exists$-sentences (which are also $\exists\forall$-sentences) over functional languages, which is the scope of our Theorem 17. For example, Theorem 18 does not say anything about the complexity of **PIGEON**, since the **PIGEON** principle is not over a relational language. In fact, $\Phi_{\mathbf{PIGEON}}$ is in type-2 $\mathbf{FP}^{\mathbf{NP}}$ trivially: binary search asking 'does there exist $v > k$ witnessing Φ_{PIGEON}?' for various k yields a solution in polynomial-time.

Theorem 17 seems to follow from Theorem 7 and the fact that CNF(**ITERATION**) has small tree-like Resolution refutations, via a lower bound of the style presented in [Rii01, Kra01, DR01]. If Q_Φ is reducible to **ITERATION**, then $CNF(Q_\Phi)$ has tree-like LK refutations of size polynomial in N that mention only formulas that can be represented as poly-depth decision trees over the type-2 objects of Q_Φ. On the other hand, if $CNF(Q_\Phi)$ holds in an infinite model, it shouldn't have such a refutation (for a treatment of similar lower bounds for LK, see [GT03]). Instead, however, we give a more direct proof that does not go through proof complexity.

7.1 Proof of Theorem 17

Throughout this section, we fix the language L to be $L = \{0, \alpha\}$, where α is a unary function, and assume that Φ is an $\exists$-sentence over L of the form $(\exists x)\phi(x)$. The case with arbitrary language and arbitrary $\exists$-sentence is analogous to the current case.

For any $n \geq 1$, a partial function $\rho_n : V_n \mapsto V_n$ is called a *restriction*. Let $\rho = \{\rho_n\}_n$ be a family of restrictions. We denote by $(Q_\Phi)^\rho$ the type-2 search problem Q_Φ such that the oracle for α answers queries consistently with ρ, i.e., on instance $(1^n, \alpha)$, if $v \in dom(\rho_n)$, then the query '$\alpha(v)$' is answered with $\rho_n(v)$. The size of restriction ρ_n is $|dom(\rho_n)| + |ran(\rho_n)|$ and is written $|\rho_n|$. We say that $\{\rho_n\}_n$ is a *polysize family* if $|\rho_n| \in n^{O(1)}$. We say that a restriction $\rho_n : V_n \mapsto V_n$ contains a solution for Q_Φ if the defined part of ρ_n contains a witness to Φ in V_n.

A restriction $\rho_n : V_n \mapsto V_n$ is said to be *safe for* Φ if the following conditions are met: (1) there exists an infinite structure $\mathcal{K} = (K, \alpha_K)$ in which Φ fails; and (2) there exists a one-one mapping $h : V_n \mapsto K$ such that $\rho_n(v) = u$ implies $\alpha_K(h(v)) = h(u)$. Note that, if ρ_n is safe for Φ, then ρ_n does not contain a solution for Q_Φ. We say that a family $\{\rho_n\}_n$ of restrictions is safe for Φ if ρ_n is safe for Φ for every n.

Define T_n to be a decision tree whose internal nodes are labeled with queries '$\alpha(v)$' for some $v \in V_n$ and whose edges are labeled with answers 'u' for some $u \in V_n$. Each internal node specifies an oracle query to α, and if the answer is '$\alpha(v) = u$', then the outgoing edge with label 'u' should be taken, which leads to the node specifying the next query. This procedure terminates when a leaf node is reached. The leaf nodes are unlabeled. Note that, for every path P of T_n, there exists a corresponding restriction π_P specified by the queries and answers on P. We say that T_n solves Q_Φ on n if, for every path P of T_n, the corresponding restriction π_P contains a solution for Q_Φ. Let $Depth_T(n)$ be the depth of T_n, i.e., the maximum length of paths from the root to a leaf node. A family $\{T_n\}_n$ of S-trees is said to be *poly-depth* if $Depth_T(n) \in n^{O(1)}$. Poly-depth families of S-trees constitute a nonuniform version of type-2 **FP**.

Special cases of the following are implicit in [Bus86, Kra95, BCE$^+$98, CK98, Mor01].

Lemma 19. *Let L be a language not containing $\leq$ and let Φ be an $\exists$-sentence over L that fails in an infinite structure. If $\{T_n\}_n$ is a poly-depth family of S-trees over L and $\rho = \{\rho_n\}_n$ is a safe, polysize family of* *restrictions, then, for all sufficiently large n, T_n contains a path P such that $\rho_n \cup \pi_P$ is safe for Φ.*

Note that the conclusion of Lemma 19 implies that T_n does not solve $(Q_\Phi)^\rho$ on n. Proving Lemma 19 is not hard and left to the reader.

Now assume for the sake of contradiction that $Q_\Phi \leq_m$ **ITERATION** by an oracle Turing machine M that solves Q_Φ in polynomial-time by making one query to **ITERATION** and arbitrary many queries to α. Let $k(n) \in n^{O(1)}$ be the running time of M.

Claim 20. *There exists a polysize family $\{\rho_n\}_n$ of restrictions such that, for sufficiently large n, the following hold: (1) ρ_n is safe for Φ; and (2) ρ_n contains the answers to all the queries to α and* **ITERATION** *made by M on $(1^n, \alpha)$.*

Suppose Claim 20 holds and consider M on $(1^n, \alpha)$ for n sufficiently large. We answer all the queries to α and **ITERATION** according to ρ_n asserted to exist by the Claim. At the end of its computation, M is forced to output some $v \in V_n$ as a solution, although no solution is forced by ρ_n. Hence, after M outputs some v, we extend ρ_n to some α such that $\phi(v)$ does not hold in V_n. This completes the proof of Theorem 17.

It remains to prove Claim 20. Fix n sufficiently large so that all the necessary invocations of Lemma 19 hold, and let $k = k(n)$. We divide the computation of M into 3 phases: phase 2 is the **ITERATION**-query of M, and phase 1 and 3 consist of all the α-queries that are asked before and after the **ITERATION**-query, respectively. Our goal is to construct safe restrictions μ_1, μ_2, μ_3 such that (1) μ_3 extends μ_2, which extends μ_1; (2) μ_i contains the answers to all the queries that are asked in the first i phases of M; and (3) $|\mu_i| \in n^{O(1)}$.

In order to construct μ_1, let M' be an oracle Turing machine that simulates M until M writes the **ITERATION**-query, at which point M' halts. Thus, M' simulates phase 1 of M and asks at most k α-queries. Construct a decision tree T' from M' by extracting all possible sequences of α queries that M' asks. By Lemma 19 (with ρ_n empty), T' contains a path P such that π_P is safe for Φ. Let $\mu_1 = \pi_P$.

For phase 2, let $(1^m, \beta)$ be the **ITERATION**-query that M asks, when all preceding α-queries are answered according to μ_1. Our task is to construct μ_2 by extending μ_1 enough so that a solution for **ITERATION**$(1^m, \beta)$ is specified, while keeping μ_2 safe for Φ. By definition of many-one reduction, $\beta : V_m \mapsto V_m$ is computable by an oracle Turing machine M_β in time polynomial in n using α as an oracle. For each $x \in V_m$, let $B(x)$ be the decision tree corresponding to the computations of M_β on x. We say a path P of the decision tree $B(x)$ is *good* if P is consistent with μ_1 and $\mu_1 \cup \pi_P$ is safe. For each $x \in V_m$, let $Good_B(x)$ be the set of all good paths of $B(x)$. By Lemma 19, for all x, $Good_B(x)$ is not empty.

There are three cases to consider.

First Case: $Good_B(0^n)$ contains a path P such that the corresponding computation of M_β makes $\beta(0^n) = 0^n$. We set $\mu_2 = \mu_1 \cup \pi_P$ and return an arbitrary $v \in V_m$ to M as a solution for the **ITERATION**-query $(1^m, \beta)$.

Second Case: For some $x \in V_m$, $Good_B(x)$ contains a path P such that the corresponding computation of M_β makes $\beta(x) = y$ for some $y < x$. We set $\mu_2 = \mu_1 \cup \pi_P$ and return x as a solution for the **ITERATION**-query.

Third Case: the above two cases do not hold. Since the first case does not hold, every path in $Good_B(0^n)$ corresponds to a computation of M_β with $\beta(0^n) > 0^n$. Similarly, since the second case does not hold, every path in $Good_B(1^n)$ leads to $\beta(1^n) = 1^n$. Hence, by the least number principle, there exists $x \in V_m$ such that (1) $Good_B(x)$ contains a path P' that leads to $\beta(x) = y$ for some $y > x$; and (2) for all $z > x$, every path in $Good_B(z)$ leads to $\beta(z) = z$.

Let x, y, and P' be as in the preceding paragraph. Let $Better_B(y)$ as the set of paths P'' of $B(y)$ such that $\pi_{P''}$ is consistent with $\mu_1 \cup \pi_{P'}$ and $\mu_1 \cup \pi_{P'} \cup \pi_{P''}$ is safe for Φ. By Lemma 19, $Better_B(y)$ is not empty. Let P^* be any path in $Better_B(y)$. Set μ_2 to be $\mu_1 \cup \pi_{P'} \cup \pi_{P^*}$ and return x to M as a solution for its **ITERATION**-query. Note that x is a solution because $\beta(x) = y$ and $\beta(y) = y$. This concludes the construction of μ_2.

μ_3 is constructed in essentially the same way as μ_1 by invoking Lemma 19 with μ_2. Finally, let $\rho_n = \mu_3$. The resulting family $\{\rho_n\}_n$ satisfies the conditions in Claim 20.

8 Concluding Remarks and Open Problems

We have obtained a number of search problem separations from proof complexity separations and our Theorems 7 and 9. Note that our proofs of these separations do not depend on the fact that the substitution instance of $CNF(Q_\Psi)$ and $poly(Q_\Psi)$ are uniformly generated by a Turing machine that reduces Q_Φ to Q_Ψ. Hence, all the search problem separations in this paper hold to exclude reductions by nonuniform poly-size circuits. The same is true for the separations obtained in [BCE+98, Mor01].

All the separations we obtained in this paper are with respect to many-one reducibility. Since all the known separations from [BCE+98, Mor01] are known to hold with respect to Turing reducibility, it is an interesting open problem to see if this stronger separation is obtainable directly from proof complexity separation.

We made progress toward resolving the relative complexity of **PLS** by showing **ITERATION** $\not\leq_m$ **LONELY** and **ITERATION** is not nicely reducible to **PIGEON**. We are interested in knowing whether **ITERATION** is many-one reducible to **PIGEON** or not, which still remains open. One difficulty is that the iteration principle is easy for almost all proof systems (except for Nullstellensatz, for which the hardness of the iteration principle allowed us to prove **ITERATION** $\not\leq_m$ **LONELY**) and the pigeonhole principle is hard for almost all proof systems.

From Theorem 9 and the fact that $poly$(**LONELY**) has constant-degree Nullstellensatz refutations, it follows that the totality of every **PPA** problem has low-degree Nullstellensatz proofs. This indicates that the fixed point theorems of Brower, Nash, and Kakutani, whose corresponding search problems are in **PPA**, have low-complexity proofs.

Theorems 7 and 9 constructs propositional refutations from reductions. Does the converse hold? Is it true that the translation of a search problem has a small LK or Nullstellensatz refutation, then the search problem is reducible to, say, **ITERATION** (which is easy for LK) or **LONELY** (which is easy for Nullstellensatz)?

The theories of bounded arithmetic are introduced by Buss in [Bus86] as fragments of Peano Arithmetic with bounds on their reasoning power. Bounded arithmetic is closely related to computational complexity and proof complexity, and our results connecting these two areas naturally have some consequences on bounded arithmetic as well. For the definitions and relevant results, we refer the reader to [Bus86, Kra95, Bus98a].

Theorem 21. *Let $\Psi(a) \in \Sigma^b_\infty(L)$, where L is an arbitrary set of symbols, and define Q_Ψ be a type-2 search problem of witnessing Ψ in canonical structures. Assume that the relativized bounded arithmetic theory $S_2(L)$ proves $\forall x \Psi(a)$. Then* **PIGEON** $\not\leq_m Q_\Psi$ *and* **LONELY** $\not\leq_m Q_\Psi$. *In fact, $Q_\Phi \not\leq_m Q_\Psi$ for any Φ such that $CNF(\Phi)$ requires exponential-size refutations in any bounded-depth LK system.*

Proof. The idea is that, if $S_2(L)$ proves $\forall x \Psi(x)$, then the propositional translations of $\Psi(a)$ has quasipolynomial-size proofs in bounded-depth LK [PW85, Kra95]. From Theorem 7 it follows that, if $Q_\Phi \leq_m Q_\Psi$, then $CNF(\Phi)$ has subexponential-size LK refutations, which contradicts the assumption. ☐

Our Theorem 17 implies the following independence criterion for the relativized S^2_2 by Riis [Rii93] in a similar way Krajicek's theorem (Theorem 18) in [Kra95] implies it.

Theorem 22. *[Rii93] Let L be a language that is disjoint with the language of bounded arithmetic, and let $\Phi = \exists x \phi(x)$ be a sentence of arbitrary quantifier-complexity. If Φ fails in an infinite structure, then the relativized bounded arithmetic theory $S^2_2(L)$ does not prove $\Phi^{<a}$, where $\Phi^{<a}$ is Φ with all quantifiers bounded by a free variable a.*

Proof. Krajicek has a proof of this theorem based on complexity-theoretic separation. Since our proof is similar to his, we only sketch the idea. Let Φ be of the form $\exists x_1 \forall y_1 \ldots \exists x_k \forall y_k \phi(x_1, y_1, \ldots, x_k, y_k)$, with ϕ quantifi er-free. Defi ne a herbrandization Φ_H of Φ as

$$\exists x_1 \exists x_2 \ldots \exists x_k \phi(x_1, f_1(a, x_1), \ldots, x_k, f_k(a, x_1, \ldots, x_k)),$$

where $f_1, \ldots, f_k$ are new functions. Let $L' = L \cup \{f_1, \ldots, f_k\}$. Since there is an infi nite structure in which Φ_H fails, Q_{Φ_H} is not reducible to **ITERATION**. Since **ITERATION** characterizes the $\Sigma^b_1(L')$-consequences of $S^2_2(L')$, $S^2_2(L')$ does not prove

$$\exists x_1 < a \exists x_2 < a \ldots \exists x_k < a[f_1(a, x_1) < a \wedge \ldots \wedge f_k(a, x_1, \ldots, x_k) \supset \phi(x_1, f_1(a, x_1), \ldots, x_k, f_k(a, x_1, \ldots, x_k))].$$

Let M be a model of $S^2_2(L')$ in which the above formula fails. It is not hard to see that $\Phi^{<a}$ fails in this model. $\square$

Acknowledgement. The second author wishes to thank S. Riis for his suggestion that his independence criterion may follow from the main result of [Mor01], which led to Theorems 17 and 22 of this paper. The fi rst author wishes to thank Toni Pitassi for some helpful discussions.

References

[BCE+98] Paul W. Beame, Stephen A. Cook, Jeff Edmonds, ussell Impagliazzo, and Toniann Pitassi. The relative complexity of *NP* search problems. *Journal of Computer and System Sciences*, 57:3–19, 1998.

[BK94] S. R. Buss and J. Krajicek. An application of Boolean complexity to separation problems in bounded arithmetic. *Proceedings of the London Mathematical Society*, 69:1–27, 1994.

[BP96] Paul W. Beame and Toniann Pitassi. An exponential separation between the parity principle and the pigeonhole principle. *Annals of Pure and Applied Logic*, 80:197–222, 1996.

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

[Bus98a] S. R. Buss. First-order proof theory of arithmetic. In S. R. Buss, editor, *Handbook of proof theory*, pages 79–147. Elsevier Science, 1998.

[Bus98b] S. R. Buss. Lower bounds on Nullstellensatz proofs via designs. In Paul W. Beame and Samuel R. Buss, editors, *Proof Complexity and Feasible Arithmetics*, volume 39 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 59–71. American Mathematical Society, 1998.

[CEI96] M. Clegg, J. Edmonds, and R. Impagliazzo. Using the Gröbner basis algorithm to fi nd proofs of unsatisfi ability. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, pages 174–183, Philadelphia, PA, May 1996.

[CIY97] S. A. Cook, R. Impagliazzo, and T. Yamakami. A tight relationship between generic oracles and type-2 complexity. *Information and Computation*, 137(2):159–170, 1997.

[CK98] M. Chiari and J. Krajicek. Witnessing functions in bounded arithmetic and search problems. *The Journal of Symbolic Logic*, 63:1095–1115, 1998.

[DR01] Dantchev and Riis. Tree resolution proofs of the weak pigeon-hole principle. In *Annual IEEE Conference on Computational Complexity (formerly Annual Conference on Structure in Complexity Theory)*, volume 16, 2001.

[GT03] N. Galesi and N. Thapen. The complexity of treelike systems over λ-local formulae. Manuscript, 2003.

[JPY88] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37:79–100, 1988.

[KPW95] J. Krajíček, P. Pudlák, and A Woods. Exponential lower bounds to the size of bounded depth Frege proofs of the pigeonhole principle. *Random Structures and Algorithms*, 7(1), 1995.

[Kra95] J. Krajicek. *Bounded Arithmetic, Propositional Logic, and Complexity Theory*, volume 60 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, 1995.

[Kra01] J. Krajicek. On the weak pigeonhole principle. *Fundamenta Mathematicae*, 170:123–140, 2001.

[Mor01] T. Morioka. Classification of Search Problems and Their Definability in Bounded Arithmetic. Master's thesis, University of Toronto, 2001. Also available as ECCC technical report TR01-82 at http://www.eccc.uni-trier.de.

[MP91] N. Megiddo and C. H. Papadimitriou. On total functions, existence theorems and computational complexity. *Theoretical Computer Science*, 81:317–324, 1991.

[MPW00] Alexis Maciel, Toniann Pitassi, and Alan R. Woods. A new proof of the weak pigeonhole principle. In ACM, editor, *Proceedings of the thirty second annual ACM Symposium on Theory of Computing: Portland, Oregon, May 21–23, [2000]*, pages 368–377, New York, NY, USA, 2000. ACM Press.

[Pap94a] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[Pap94b] C. H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48:498–532, 1994.

[PBI93] Toniann Pitassi, Paul W. Beame, and Russell Impagliazzo. Exponential lower bounds for the pigeonhole principle. *Computational Complexity*, 3(2):97–140, 1993.

[PW85] J. Paris and A. Wilkie. Counting problems in bounded arithmetic. In *Methods in Mathematical Logic: Proceedings of the 6th Latin American Symposium on Mathematical Logic 1983*, volume 1130 of *Lecture notes in Mathematics*, pages 317–340, Berlin, 1985. Springer-Verlag.

[Raz98] A. A. Razborov. Lower bounds for the polynomial calculus. *Computational Complexity*, 7(4):291–324, 1998.

[Rii93] S. Riis. Making infinite structures finite in models of second order bounded arithmetic. In P. Clote and J. Krajicek, editors, *Arithmetic, Proof Theory, and Computational Complexity*, pages 289–319. Oxford University Press, 1993.

[Rii01] S. Riis. A complexity gap for tree resolution. *Computational Complexity*, 10:179–209, 2001.

[Tow90] M. Townsend. Complexity for type-2 relations. *Notre Dame Journal of Formal Logic*, 31(2):241–262, 1990.

[Yan97] M. Yannakakis. Computational complexity. In E. Aarts and J. K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 19–55. John Wiley and Sons Ltd., 1997.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'