Intersection Classes in TFNP and Proof Complexity

Yuhao Li
Columbia University, New York, NY, USA

William Pires
Columbia University, New York, NY, USA

Robert Robere
McGill University, Montreal, Canada

Abstract

A recent breakthrough in the theory of *total NP search problems* (TFNP) by Fearnley, Goldberg, Hollender, and Savani has shown that $\mathsf{CLS} = \mathsf{PLS} \cap \mathsf{PPAD}$, or, in other words, the class of problems reducible to gradient descent are exactly those problems in the intersection of the complexity classes PLS and PPAD. Since this result, two more intersection theorems have been discovered in this theory: $\mathsf{EOPL} = \mathsf{PLS} \cap \mathsf{PPAD}$ and $\mathsf{SOPL} = \mathsf{PLS} \cap \mathsf{PPADS}$. It is natural to wonder if this exhausts the list of intersection classes in TFNP, or, if other intersections exist.

In this work, we completely classify *all* intersection classes involved among the classical TFNP classes PLS, PPAD, and PPA, giving new complete problems for the newly-introduced intersections. Following the close links between the theory of TFNP and propositional proof complexity, we develop new proof systems – each of which is a generalization of the classical Resolution proof system – that characterize all of the classes, in the sense that a query total search problem is in the intersection class if and only if a tautology associated with the search problem has a short proof in the proof system. We complement these new characterizations with black-box separations between all of the newly introduced classes and prior classes, thus giving strong evidence that no further collapse occurs. Finally, we characterize arbitrary intersections and joins of the PPA_q classes for $q \geq 2$ in terms of the Nullstellensatz proof systems.

2012 ACM Subject Classification Theory of computation → Complexity classes; Theory of computation → Proof complexity; Theory of computation → Complexity theory and logic

Keywords and phrases TFNP, Proof Complexity, Intersection Classes

Digital Object Identifier 10.4230/LIPIcs.ITCS.2024.74

Funding *Yuhao Li*: Supported by NSF grants IIS-1838154, CCF-2106429 and CCF-2107187.
William Pires: Supported by NSF grants AF-2212136, IIS-1838154, CCF-2106429 and CCF-2107187.
Robert Robere: Supported by NSERC.

Acknowledgements Part of this work was done while the authors were visiting the Simons Institute for the Theory of Computing at UC Berkeley.

1 Introduction

The class TFNP contains all *total* NP-search problems, which are search problems whose solutions are (a) verifiable in polynomial time, and (b) guaranteed to always exist. This class plays an important role in computational complexity theory as it contains many important search problems that we would like to solve in practice. Two standard examples of such problems are Factoring (given a number, output a prime factor of that number) and Nash (given a bimatrix game, output a Nash Equilibrium for that game). The class TFNP is a semantically defined class which is not believed to have complete problems [33], and thus researchers define syntactic subclasses of TFNP via polynomial-time reductions to certain

fixed TFNP problems. Perhaps the most famous example is the class PPAD, which has as its complete problem Imbalance, where we are given a directed graph with a fixed unbalanced node (i.e. the out-degree of the node is different from the in-degree of the node) and must output any other unbalanced node. This class famously has Nash as a complete problem [14, 10], along with many other problems in game theory and economics [13, 9, 11, 28, 12]. Another prominent example of a TFNP subclass is PLS, which has Sink-of-Dag as its complete problem, where we are given a directed acyclic graph (dag) as input and are asked to find a sink node in that graph. The class PLS was originally defined to capture the complexity of problems solvable by local-search heuristics in a very generic model [26, 34, 30].

The overarching goal of the present work is a systematic study of *intersections* of standard TFNP classes. This is not without precedent in the literature. A recent breakthrough result by [16] proved that $\mathsf{PLS} \cap \mathsf{PPAD} = \mathsf{CLS}$, where CLS is, intuitively, the class of total search problems reducible to computing a local minimum via gradient descent. This result is quite remarkable among class collapses in complexity theory for a number of reasons; perhaps chief among them is that taking the intersection of two complexity classes is an unnatural operation, and usually does not yield classes that can be syntactically defined in an obvious way. For example, while both NP and coNP have syntactic definitions, it is a long-standing open problem in complexity theory whether or not the class $\mathsf{NP} \cap \mathsf{coNP}$ has any complete problems.

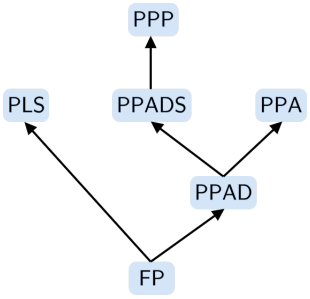
In the theory of TFNP, on the other hand, one can quite easily define *some* complete problem for each intersection class. Formally, if A is a complete problem for the complexity class $\mathsf{A} \subseteq \mathsf{TFNP}$ and B is a complete problem for the class $\mathsf{B} \subseteq \mathsf{TFNP}$, then we can define a new problem Either(A, B) which is complete for $\mathsf{A} \cap \mathsf{B}$ as follows: given instances of *both* problem A and B as input, output a solution to *either* of the instances. It is an easy exercise to see that Either(A, B) is complete for $\mathsf{A} \cap \mathsf{B}$, but, the problem is rather artificially defined. Indeed, while it is easy to reduce the complete problem of CLS to both End-of-Line (another complete problem for PPAD) and Sink-of-Dag, the main contribution of [16] is an intricate reverse reduction from Either(End-of-Line, Sink-of-Dag) back to the defining problem of CLS.

Given the collapse $\mathsf{CLS} = \mathsf{PLS} \cap \mathsf{PPAD}$, it is natural to wonder if other intersections of the commonly-considered TFNP classes also admit nicer complete problems than problems constructed using the Either operation. In the original work that systematically studied subclasses of TFNP [30], Papadimitriou introduced five subclasses of TFNP: the classes PLS [26] and PPAD, which have been defined above, along with the classes:

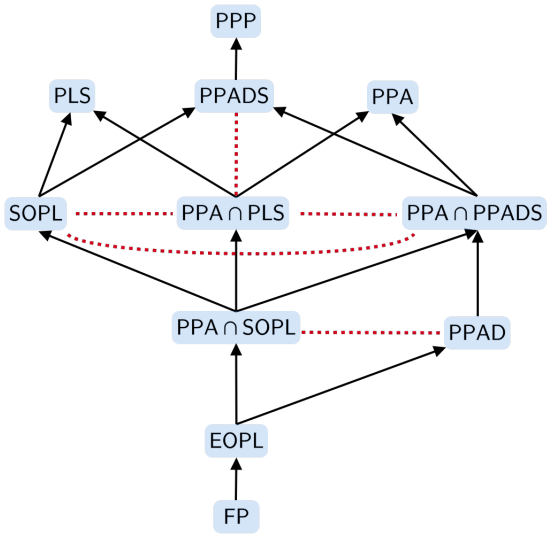
- PPA, defined by the complete problem Lonely: given a graph with an odd number of vertices and where every vertex has degree ≤ 1, output any isolated node.
- PPADS, defined by the complete problem Sink-of-Line: given a directed graph with a fixed *positively* imbalanced node (out-degree > in-degree), output a *negatively* imbalanced node (in-degree < out-degree).
- PPP, defined by the complete problem Pigeon: given a circuit C mapping $\{0,1\}^n \to \{0,1\}^n \setminus 0^n$, find any $x \neq y$ such that $C(x) = C(y)$.

In a follow-up work to [16], Göös et al gave two more examples of intersection results among the original "Big Five" TFNP classes: $\mathsf{SOPL} = \mathsf{PLS} \cap \mathsf{PPADS}$, and $\mathsf{EOPL} = \mathsf{PLS} \cap \mathsf{PPAD} = \mathsf{CLS}$ [19], where EOPL and SOPL are subclasses of TFNP that had been introduced by [17] and [22], respectively. Outside of this, Ishizuka gave a complete problem for $\mathsf{PLS} \cap \mathsf{PPA}$ that was distinct from Either(Sink-of-Dag, Lonely) [25]. Other than these results, no other complete problems are currently known for other intersections among the original TFNP

classes. Furthermore, it is not even known if the other intersection classes simply collapse further to existing TFNP subclasses. For example, while it is known that PPAD is contained in both PPADS and PPA, it is consistent with current knowledge that PPAD = PPADS ∩ PPA (see Figure 1a).



(a) Classical TFNP Classes. A directed arrow from class A to B means that $A \subseteq B$.



(b) Classic TFNP classes with all possible non-trivial intersections. Solid black lines represent strict inclusion w.r.t. a generic oracle, and dotted lines represent new oracle separations proved in this paper.

1.1 Propositional Proof Complexity and Black-Box TFNP

Beyond the fundamental task of understanding the relationships between the standard TFNP classes, another motivation for investigating intersection classes comes from the theory of *propositional proof complexity*, which has a very close relationship with TFNP in the *black-box setting* [6, 1, 29, 4, 8, 21, 20]. In the classical theory of TFNP, the inputs to the problems are typically encoded *succinctly*: for instance, an input to the problem Sink-of-Dag will be an exponentially-large dag that is represented by a polynomial-size circuit C encoding the dag in some topological ordering. Rather than encode these inputs by (white-box) circuits, it is often natural to encode them instead by *black-box* oracles. Formally, a *query* total search problem is a sequence of relations $R_n \subseteq \{0,1\}^n \times O_n$, one for each size $n \in \mathbb{N}$, such that for all $x \in \{0,1\}^n$ there is an $o \in O_n$ such that $(x, o) \in R_n$. Here O_n is a finite set of outputs, and we say that o is a solution to instance x when $(x, o) \in R_n$. We imagine the input x as being given by oracle access to its individual bits $\{x_i\}$, and an efficient algorithm for R_n is given by a query algorithm that makes $\log^{O(1)} n$ many queries to x before outputting a solution to R_n. We enforce efficient verifiability of query total search problems in the natural way: the problem R_n is in TFNP^{dt} if for each $o \in O_n$ the relation $(x, o) \in R_n$ can be computed by a $\log^{O(1)} n$-depth decision tree $T_o(x)$. The corresponding subclasses of TFNP in the black-box model are then defined by efficient *decision-tree reductions* to the corresponding complete problems for the subclasses, and this gives corresponding subclasses $\mathsf{PLS}^{dt}, \mathsf{PPAD}^{dt}$, etc. of query total search problems.

There are two principle reasons for studying black-box models of TFNP. The first is the ability to prove **unconditional separations.** Unlike the white-box model, in the black-box model we are able to prove unconditional separations between the complexity

classes. Formally, any separation in the black-box model implies a separation in the white-box model relative to a generic oracle [1]. Moreover, all known class collapses in TFNP relativize, and so black-box separations rule out any collapse using existing techniques.

The second, and more significant in the eyes of the authors, are **characterizations by propositional proof systems.** There is a close link between the black-box TFNP classes and proof systems in propositional proof complexity. Associated with every unsatisfiable CNF formula $F = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ is the following query total search problem $S(F)$: given an assignment x to the variables of F, output the index of any clause C_i falisfied by x. If we have a sequence of unsatisfiable CNFs F_n on n variables, then this gives a query total search problem $S(F_n)$, which will be in TFNP^{dt} iff the underlying CNF formula has $\log^{O(1)} n$ width. Conversely, if we have *any* total search problem $R_n \in \mathsf{TFNP}^{dt}$, we can write down the corresponding unsatisfiable CNF formula $F_n = \bigwedge_{o \in O_n} \neg T_o(x)$ – obtained by re-encoding the decision-trees into CNF formulas – intuitively stating that R_n has no solution. It turns out that this close relationship is not merely a coincidence: every TFNP^{dt} subclass A^{dt} defined by reductions to a complete problem is *characterized* by a corresponding propositional proof system P_{A}, in the following sense:

A total search problem R_n is in the class A^{dt}
if and only if
The corresponding sequence of CNFs F_n has efficient refutations in the proof system P_{A}.

One can always show the existence of such a proof system [5], but, this abstract construction produces proof systems which are somewhat artificial. It turns out that many of the standard TFNP classes correspond to *natural* proof systems that had already been studied in the literature, prior to this connection being observed. For example, the class PLS^{dt} corresponds exactly to the well-studied *Resolution* proof system, in the sense that a sequence of CNF formulas F_n has $\log^{O(1)} n$-width Resolution refutations iff $S(F_n) \in \mathsf{PLS}^{dt}$ [27, 8]. There have recently been a number of these characterizations shown [21, 27, 20, 15], including characterizations for all of the classical TFNP classes $\mathsf{PPA}, \mathsf{PLS}, \mathsf{PPADS}$, and PPAD [21, 27, 20].

These equivalences provide a two-way street between the theory of TFNP and the theory of propositional proof complexity, allowing us to transform results from one setting into counterpart results in the other. For instance, by recruiting lower bounds in proof complexity, one can construct oracle separations between subclasses of TFNP, and indeed this is the primary mechanism by which these separations are obtained [1, 29, 4, 20]. On the other hand, we only know how to prove the intersection theorems for SOPL and EOPL using the language of TFNP, and this has led to surprising *intersection theorems* for proof systems. For instance, the intersection theorem $\mathsf{SOPL} = \mathsf{PLS} \cap \mathsf{PPADS}$ implies that an unsatisfiable CNF formula F has an efficient *Reversible Resolution* refutation (corresponding to SOPL) if and only if it has both an efficient *Resolution* refutation (PLS) and an efficient *unary Sherali-Adams* refutation (PPADS) [20]. It is not clear at all how to carry out this intersection result for proof systems directly in the language of proof complexity!

A final benefit of the characterization result for proof systems by TFNP subclasses is that it provides a *dictionary*, of sorts, for the existing proof systems. One can easily ask: why do the same proof systems continually appear in many different situations? The classification by TFNP classes due to Fleming, Buss, and Impagliazzo [5] implies that *every* proof system[1]

[1] Satisfying some mild regularity requirements, e.g. being closed under decision tree reductions.

has a corresponding TFNP subclass as above. Thus, one can hope to gain insight into various "boundary" lower-bound questions in propositional proof complexity by learning what their corresponding TFNP subclasses are and studying them. Overall, there is much benefit to using both the study of proof complexity *and* the study of TFNP classes to enhance our knowledge of both fields simultaneously.

1.2 Main Results

The main result of this paper is the systematic classification of all intersections between the classical TFNP classes PPA, PPAD, PPADS, and PLS[2]. In Figure 1b we present the classical TFNP classes with all of their non-trivial intersections. (In fact, our results are stronger than stated in the theorem below. We actually can classify all possible intersections with the generalized classes PPA_q for any *integer* $q \geq 2$.)

▶ **Theorem 1.** *Among the classical* TFNP *classes* PPA, PPAD, PPADS, *and* PLS*, all possible non-trivial intersection classes between them are displayed in Figure 1b. Moreover, no further collapses occur in the black-box model: for every pair of classes* A, B *in Figure 1b where the containment* $\mathsf{A} \subseteq \mathsf{B}$ *is not depicted, there is a generic oracle* O *such that* $\mathsf{A}^O \not\subseteq \mathsf{B}^O$*.*

Our second main contribution is the construction of a *Resolution-style proof system* (cf. Definition 11) corresponding to *every* class depicted in Figure 1b. Informally, a Resolution-style proof system is defined by a set of inference rules $\mathcal{R}$ on clauses of boolean literals. Each rule $R \in \mathcal{R}$ can have any number of *input clauses* and any number of *output clauses*, and crucially we apply the rules via *substitution*: when a rule R is selected to apply to clauses in a proof, we *replace* those clauses with the output of the rule. For example, under this general definition, the standard Resolution system corresponds to $\mathcal{R}$-Resolution with the three rules $\mathcal{R} = \{\text{Rev-Weaken}, \text{Rev-Res}, \text{ Copy}\}$ defined below:

- **Rev-Weaken.** $C \vdash C \vee x, C \vee \overline{x}$,
- **Rev-Res.** $C \vee x, C \vee \overline{x} \vdash C$,
- **Copy.** $C \vdash C, C$.

(Note that we require the "Copy" rule since the inference rules are applied with substitution.) For any $\mathcal{R}$, given an $\mathcal{R}$-Resolution proof Π we define the *width* $w(\Pi)$ of the refutation to be the size of the largest clause in Π, and the *size* $\text{size}(\Pi)$ of the refutation to be the total number of clauses occurring in Π. For any unsatisfiable CNF formula F, we define the complexity measure

$$\text{Res}_{\mathcal{R}}(F) := \min_{\mathcal{R}\text{-Res proofs } \Pi \text{ of } F} w(\Pi) + \log \text{size}(\Pi).$$

▶ **Theorem 2.** *For every class* A *depicted in Figure 1b, there is a fixed set of deduction rules* $\mathcal{R}$ *on clauses such that for any query total search problem* R_n*,* $R_n \in \mathsf{A}^{dt}$ *if and only if the corresponding unsatisfiable CNF formula* F_n *satisfies* $\text{Res}_{\mathcal{R}}(F_n) = \log^{O(1)} n$*.*

This result provides natural, deductive proof systems for *all* of the classical TFNP classes and their intersections[3]. For several of these classes, Resolution-style proof systems were previously known, including PLS [27], PPAD, PPADS [3], SOPL, and EOPL [20]. In

[2] We chose to study these classes in particular, and not PPP, as they have the highest relevance for propositional proof complexity. It is currently unknown whether or not there is a natural propositional proof system corresponding to PPP^{dt}.

[3] With the exception of PPP, for which we still do not have any natural proof system beyond the abstract construction of [5].

particular, we provide the first proof systems capturing the classes $\mathsf{PPA} \cap \mathsf{PLS}$, $\mathsf{PPA} \cap \mathsf{PPADS}$, and $\mathsf{SOPL} \cap \mathsf{PPA}$. For example, for the class $\mathsf{PPA} \cap \mathsf{PLS}$, the corresponding set of deduction rules is $\mathcal{R} = \{\text{Rev-Weaken}, \text{Rev-Res}, \mathbb{F}_2\text{-Copy}\}$, where $\mathbb{F}_2$-Copy is the rule

$$C \vdash C, C, C,$$

which allows us to simultaneously copy C *and* preserve the parity of the number of copies of C.

Our final family of results are a classification of the proof systems related to the PPA_q^{dt} classes, introduced by Papadimitriou [30] and further studied in [23]. For any $q \geq 2$, the PPA_q class corresponds to the following total search problem [23]: given a bipartite graph with a fixed node of degree $\neq q$, output any other such node. It is easy to see that $\mathsf{PPA}_2 = \mathsf{PPA}$, and [23] give a general characterization of PPA_q for any $q \geq 2$ in terms of $\{\mathsf{PPA}_p\}_{p|q}$ where p ranges over all prime numbers that divide q. To properly position our results we must first describe this characterization. If A_0, A_1 are two total search problems then let $A_0 \,\&\, A_1$ denote the following search problem: given x and a bit $b \in \{0,1\}$, solve the problem A_b on input x. Note that this operation gives us the power to solve both problems simultaneously, and thus both A_0 and A_1 reduce to $A_0 \,\&\, A_1$. One of the main results of [23] is the following:

▶ **Theorem 3** (Theorem 1 [23]). *For any positive integer $q \geq 2$,*

$$\mathsf{PPA}_q = \mathsf{PPA}_{p_1} \,\&\, \mathsf{PPA}_{p_2} \,\&\, \cdots \,\&\, \mathsf{PPA}_{p_m},$$

where $p_1, \ldots, p_m$ are the prime factors of q.

We first give proof systems classifying PPA_q^{dt} for *any* positive $q \geq 2$. Our proof systems are slight generalizations of the well-studied *Nullstellensatz* proof system [2, 7].

Consider an unsatisfiable CNF $F = C_1 \wedge \ldots \wedge C_m$. For each clause $C_k = \bigvee_{i \in S} x_i \bigvee_{j \in T} \bar{x}_j$, we let $\bar{C}_k$ be the natural encoding of C_k as a polynomial. That is $\bar{C}_k := \prod_{i \in S}(1 - x_i) \prod_{j \in T} x_j$. For a truth assignment $x \in \{0,1\}^n$, we have $\bar{C}_k(x) = 0$ iff x satisfies C_k.

▶ **Definition 4** (Generalized Nullstellensatz). *Let $q \geq 2$ be a positive integer and consider the ring $\mathbb{Z}_q$. Given a CNF $F = C_1 \wedge \ldots \wedge C_m$ over variables $x_1, \ldots, x_n$, a* generalized Nullstellensatz refutation *of F over $\mathbb{Z}_q$ is given by a list of polynomials $P_1, \ldots, P_m \in \mathbb{Z}_q[x_1, \ldots, x_n]$ such that:*

$$\sum_{j=1}^{m} P_j \bar{C}_j \equiv c \mod q$$

where c is any constant $\not\equiv 0 \mod q$, and the algebra is performed multilinearly (i.e. modulo the ideal $\langle x_i^2 - x_i \rangle_{i=1}^n$). The degree *of the proof is defined to be the maximum degree of any of the $P_j \overline{C}_j$ polynomials, and the* size *of the proof is defined to be the* monomial size *– that is, the number of monomials produced when expanding all $P_j \bar{C}_j$ polynomials out into their constituent monomials, before cancellation. Let $\mathrm{NS}_q^*(F)$ to denote the minimum degree of any generalized Nullstellensatz refutation of F over q. A* Nullstellensatz refutation *is a special case of a generalized Nullstellensatz refutation where $c = 1$ in the above definition, and let $\mathrm{NS}_q(F)$ denote the minimum degree of a Nullstellensatz refutation* mod q.

It is easy to see that if q is prime, then $\mathrm{NS}_q^*(F) = \mathrm{NS}_q(F)$ for every F, since then the ring $\mathbb{Z}_q$ is a field and we can simply divide by c to obtain a Nullstellensatz refutation from a generalized Nullstellensatz refutation. However, when q is a composite number –

thus $\mathbb{Z}_q$ is only a ring and not a field – we show that these proof systems are radically different in power. Namely, the *generalized* Nullstellensatz system NS^*_q characterizes the TFNP class $\mathsf{PPA}^{dt}_q = \&_{p|q}\mathsf{PPA}^{dt}_p$, while the standard Nullstellensatz system NS_q characterizes the intersection class

$$\bigcap_{p|q} \mathsf{PPA}^{dt}_p,$$

where both operations range over all primes p that divide q.

▶ **Theorem 5.** *Let $q \geq 2$ be any positive integer, let $p_1, \ldots, p_m$ denote the prime factors of q, and let F_n be any sequence of $\log^{O(1)} n$-width unsatisfiable CNF formulas. Then*

- $\mathrm{NS}_q(F_n) = \log^{O(1)} n$ *if and only if* $S(F_n) \in \mathsf{PPA}^{dt}_{p_1} \cap \cdots \cap \mathsf{PPA}^{dt}_{p_m}$, *and*
- $\mathrm{NS}^*_q(F_n) = \log^{O(1)} n$ *if and only if* $S(F_n) \in \mathsf{PPA}^{dt}_q = \mathsf{PPA}^{dt}_{p_1} \,\&\, \cdots \,\&\, \mathsf{PPA}^{dt}_{p_m}$.

Theorem 5 follows from Theorem 22 and Theorem 23, and we refer to Section 3 for more details. We believe that it is quite interesting that a seemingly modest modification to the Nullstellensatz system – changing the RHS from 1 to any $c \not\equiv 0$ – allows us to capture the *intersections* or the *joins* of the classes, respectively. A second immediate corollary from Theorem 5 and (the black-box version of) Theorem 1 from [23] is the following:

▶ **Corollary 6.** *For any unsatisfiable CNF F, $\mathrm{NS}^*_{p^k}(F) = \Theta(\mathrm{NS}_p(F))$ for each prime p and $k \geq 1$.*

This corollary is quite remarkable for several reasons. Firstly, it is not hard to see that it *fails* for arbitrary systems of polynomials, and so this is a statement that can truly only apply to CNF formulas. Furthermore, in the proof, we can no longer just take an $\mathrm{NS}^*_{p^k}$ proof whose RHS is c and "divide by c" to obtain an NS_p proof – this strategy will fail if $c \equiv 0 \mod p$. Instead, we must take the $\mathrm{NS}^*_{p^k}$ proof, convert it to a $\mathsf{PPA}^{dt}_{p^k}$ instance, run the transformation underlying [23] to obtain a PPA^{dt}_p-instance, and finally transform back to a NS_p proof. We see no apparent way to prove this corollary directly in the language of proof complexity, giving yet another instance where the theory of TFNP provides powerful tools for understanding proof systems.

1.2.1 Separation Results and Feasible Disjunction

Finally, we remark on our separation results. En-route to proving Theorem 1, we prove several new oracle separation results for black-box TFNP^{dt} subclasses (cf. Theorem 39). All of the separation results follow either from known separations, or, from recruiting existing proof complexity lower-bounds in a new way. The new separation results that we must prove are:

- $\mathsf{PLS}^{dt} \cap \mathsf{PPA}^{dt} \not\subseteq \mathsf{PPADS}^{dt}$,
- $\mathsf{SOPL}^{dt} \cap \mathsf{PPA}^{dt} \not\subseteq \mathsf{PPAD}^{dt}$.

To prove these results we use the known characterizations of PPADS^{dt} and PPA^{dt}_3 by proof systems: namely, *unary Sherali-Adams* and *$\mathbb{Z}_3$-Nullstellensatz*, respectively, and use existing lower-bound techniques for these proof systems. For the sake of argument, let us consider the separation $\mathsf{PLS}^{dt} \cap \mathsf{PPA}^{dt}$. To prove these lower bounds, we do *not* use the new complete problems we construct for the intersection class, and rather prove lower bounds directly against the unsatisfiable CNFs that correspond to, e.g. $\textsc{Either}(\textsc{Sink-of-Dag}, \textsc{Lonely})$. Such CNFs are of the form $\mathrm{SOD}(x) \wedge \mathrm{Lonely}(y)$, where x and y are disjoint sets of variables.

Prior results in the proof complexity literature have shown strong lower bounds for unary Sherali-Adams against both SOD(x) and Lonely(y), and thus we show how to *combine* the lower bounds for both separate formulas into a lower bound for the combined formula.

This is closely related to the notion of *feasible disjunction* in proof complexity [32]. Roughly speaking, a proof system P has *feasible disjunction* if whenever there is an efficient P-proof of a formula $F(x) \wedge G(y)$ for unsatisfiable CNF formulas $F(x), G(y)$ on disjoint variables, then there is an efficient P-proof of either $F(x)$ or $G(y)$. Feasible disjunction is an important property in proof complexity, as it is often a precursor to showing that a proof system has *feasible interpolation* [31], which is one of the primary lower-bound techniques in proof complexity. In our separation result we are therefore showing that, for this particular pair of formulas F and G, unary Sherali-Adams has feasible disjunction (in that lower bounds for F and G can be lifted to lower bounds for the conjunction $F \wedge G$), and we use a similar argument to separate $\mathsf{SOPL}^{dt} \cap \mathsf{PPA}^{dt} \not\subseteq \mathsf{PPAD}^{dt}$ by arguing about Nullstellensatz instead.

Finally, we observe that the proof systems corresponding to any of the intersection classes inside of TFNP^{dt} *cannot* have feasible disjunction. For a guiding example, consider the complexity class $\mathsf{SOPL}^{dt} = \mathsf{PLS}^{dt} \cap \mathsf{PPADS}^{dt}$. From prior work [20], it is known that $\mathsf{SOPL}^{dt} \neq \mathsf{PLS}^{dt}$ and $\mathsf{SOPL}^{dt} \neq \mathsf{PPADS}^{dt}$. This implies that the proof system corresponding to SOPL^{dt} (the *Reversible Resolution* system [20]) has efficient refutations of the principles SOD(x) $\wedge$ SOL(y), but it does *not* have efficient refutations of either SOD(x) or SOL(y), respectively. Thus, intersection results inside of TFNP^{dt} track closely with the feasible disjunction property of the corresponding proof systems. We believe that this is a natural direction for future research.

1.3 Paper Organization

The rest of the paper is organized as follows. In Section 2 we introduce preliminary notions that are used throughout the paper, and also discuss the proof systems that we will need. In Section 3 we prove our classification results for the PPA_q classes, as they will be useful for the rest of the paper. In Section 4 we classify $\mathsf{PPA}_q \cap \mathsf{PPADS}$, in Section 5 we classify $\mathsf{PPA}_q \cap \mathsf{SOPL}$, and in Section 6 we classify $\mathsf{PPA} \cap \mathsf{PLS}$. Finally, in Section 7 we prove the remaining separation results between intersection classes.

2 Proof Systems and Preliminaries

We begin with some general notation that is used throughout the paper. We assume familiarity with the basics of query algorithms, such as the definition of a decision tree, etc. Throughout the paper, we use p to represent a prime and q to represent an integer larger than 1. For a positive integer n, we use $[n]$ to represent $\{1, \cdots, n\}$ and use $[n]_0$ to represent $\{0, \cdots, n\}$.

Occasionally, we use an element x to represent the set $\{x\}$. If $x = 0$, it means x is an empty set $\varnothing$.

2.1 TFNP Preliminaries

We now review some preliminary definitions regarding black-box TFNP classes. Our definitions are borrowed from [20].

▶ **Definition 7.** *A* query total search problem R *is a sequence of relations* $\{R_n \subseteq \{0,1\}^n \times O_n\}$, *where* O_n *are finite sets, such that for all* $x \in \{0,1\}^n$ *there is an* $o \in O_n$ *such that* $(x, o) \in R_n$. *A total search problem* R *is in* TFNP^{dt} *if for each* $o \in O_n$ *there is a decision tree* T_o *with depth* $\mathsf{poly}(\log n)$ *such that for every* $x \in \{0,1\}^n$, $T_o(x) = 1$ *iff* $(x, o) \in R$.

While total search problems are formally defined as sequences $\mathrm{R} = (R_n)$, we (following complexity-theoretic convention) will often refer to the sequence R and an individual problem R_n interchangeably. An important family of query total search problems come from *low-width unsatisfiable CNF formulas.*

▶ **Definition 8.** *For any unsatisfiable CNF formula $F \coloneqq C_1 \wedge \cdots \wedge C_m$ over n variables, define $S(F) \subseteq \{0,1\}^n \times [m]$ by $(x,i) \in S(F)$ if and only if $C_i(x) = 0$.*

Such examples are, in fact, canonical: every query total search problem can be re-written as a total search problem for some unsatisfiable CNF formula:

▶ **Definition 9.** *For any total search problem $R \subseteq \{0,1\}^n \times O$ with solution verifiers T_o, $o \in O$, its encoding as an unsatisfiable CNF formula is given by $F \coloneqq \bigwedge_{o \in O} \neg T_o(x)$ where we think of $\neg T_o(x)$ written as a CNF formula (of width determined by the decision tree depth of T_o).*

We also need notions of reductions between total search problems. The appropriate definition for black-box TFNP are mapping reductions computable by low (i.e. $\log^{O(1)} n$)) depth decision trees.

▶ **Definition 10.** *Let $R \subseteq \{0,1\}^n \times O$ and $S \subseteq \{0,1\}^m \times O'$ be total search problems. An S-*formulation of *R is a decision-tree reduction $(f_i, g_o)_{i \in [m], o \in O'}$ from R to S. Formally, for each $i \in [m]$ and $o \in O'$ there are functions $f_i \colon \{0,1\}^n \to \{0,1\}$ and $g_o \colon \{0,1\}^n \to O$ such that*

$$(x, g_o(x)) \in R \iff (f(x), o) \in S$$

where $f(x) \in \{0,1\}^m$ is the string whose i-th bit is $f_i(x)$. The depth *of the reduction is*

$$d \coloneqq \max\left(\{D(f_i) : i \in [m]\} \cup \{D(g_o) : o \in O'\} \right),$$

where $D(h)$ denotes the decision-tree depth of h. The size *of the reduction is m, the number of input bits to S. The* complexity *of the reduction is $\log m + d$. We write $S^{dt}(R)$ to denote the minimum complexity of an S-formulation of R.*

We extend these notations to sequences in the natural way. If R is a single search problem and $S = (S_m)$ is a sequence of search problems, then we denote by $S^{dt}(R)$ the minimum of $S^{dt}_m(R)$ over all m. If $\mathrm{R} = (R_n)$ is also a sequence, then we denote by $\mathrm{S}^{dt}(\mathrm{R})$ the function $n \mapsto S^{dt}(R_n)$.

Using the previous definition we can now define complexity classes of total search problems via reductions. For total search problems $\mathrm{R} = (R_n), \mathrm{S} = (S_n)$, we write

$$\mathrm{S}^{dt} \coloneqq \left\{ \mathrm{R} : \mathrm{S}^{dt}(\mathrm{R}) = \mathsf{poly}(\log n) \right\}.$$

2.2 Proof Systems

We consider a general definition of a Resolution-style proof system.

▶ **Definition 11.** *A Resolution-style proof system is a proof system described by a set of rule inference rules $\mathcal{R}$. Let F be an unsatisfiable CNF formula over variables $x_1, \ldots, x_n$. A refutation of F is composed of a sequence of multisets of clauses $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_l$ such that the following holds:*

- *$\mathcal{C}_1$ contains only weakenings of clauses of F.*
- *The final multiset contains some non-zero number of copies of $\bot$.*
- *If the proof system is said to be "with Terminals", then all clauses other than $\bot$ in the final multiset $\mathcal{C}_l$ must be weakenings of clauses from F.*
- *For each $i = 1, 2, \ldots, l-1$, the multiset $\mathcal{C}_{i+1}$ is obtained from $\mathcal{C}_i$ by applying one of the rules from the set $\mathcal{R}$. In particular, all the rules we consider consume their premises, that is the premises are removed from $\mathcal{C}_i$, and the conclusions of the rules are added to $\mathcal{C}_{i+1}$*

The size of the proof is given by $\sum_{i=1}^{l} |\mathcal{C}_i|$, and the width of the proof is the maximum width of any clause appearing in some multiset.

In all the Resolution-style proof systems we consider, the set $\mathcal{R}$ will contain the following rules:

Where D is an arbitrary clause. $\dfrac{}{D \vee C}$ Import $C \in F$

$\dfrac{C}{C \vee x_i \quad C \vee \bar{x}_i}$ Rev-Weaken

$\dfrac{C \vee x_i \quad C \vee \bar{x}_i}{C}$ Rev-Res

A key point about the bottom two rules is that under a truth assignment x to the variables, the number of falsified clauses in the premise is always equal to the number of falsified clauses in the conclusion.

▶ **Definition 12** (Resolution). *Resolution is a proof system, without terminals, where the final multiset must contain at least one copy of $\bot$. The set of allowed rules is Rev-Weaken, Rev-Res, Import, and Copy; here, Copy is the rule*

$\dfrac{C}{C \quad C}$ *Copy*

While this isn't the usual definition of Resolution our definition can clearly be simulated by the usual one without changing the width or size by a significant factor.

▶ **Definition 13** (Reversible Resolution [20]). *Reversible Resolution is a proof system, without terminals, where the final multiset must contain at least one copy of $\bot$. The set allowed rules are Rev-Weaken, Rev-Res, Import.*

The original definition of [20], doesn't have an import rule and also requires that all clauses in $\mathcal{C}_1$ are clauses from the CNF formula F rather than weakenings. However, the import rule can be simulated by the Rev-Weaken rule at the cost of blowing up the size of the proof by a polynomial factor in the size and width of the proof. For each clause $D \vee C$ that starts in $\mathcal{C}_1$ or that we import later in the proof, we now put C in the starting multiset, and then proceed one literal from D at time, we use Rev-Weaken to derive $l_1 \vee C$, $l_1 \vee l_2 \vee C$, …, $D \vee C$. Now we can run the proof ignoring any import. This process doesn't change the width of the proof.

▶ **Definition 14** (Weighted Resolution-style proof systems). *A Weighted Resolution-style proof system is a Resolution-style proof system with the following modifications:*

- *Every clause appearing in the proof is labeled with a "+" or a "−" sign.*
- *$\mathcal{C}_1$ contains only clauses from F or weakenings of such clauses, all such clauses are labeled with a "+".*
- *The final multiset contains some number of copies of $+\bot$.*

- *All clauses in the final multiset $\mathcal{C}_l$ are labelled with a "+". In particular, such proof systems do not have a notion of "with Terminals".*

Again, the size of the proof is given by $\sum_{i=1}^{l} |\mathcal{C}_i|$, and the width of the proof is the maximum width of any clause appearing in some multiset.

For all the weighted resolution proofs systems we consider, the set of inference rules $\mathcal{R}$ contains a $+$ and $-$ version of Rev-Res, Rev-Weaken, and Import. These correspond to the rules defined previously, but now either all the premises and conclusions are labeled with a "+" or they are all labeled with a "−". For instance the $+$ Rev-Weaken rule is the following one:

$$\frac{+C}{+C \vee x_i \quad +C \vee \bar{x}_i} + \text{Rev-Weaken}$$

▶ **Definition 15** (Unary Weighted Resolution). *Unary Weighted Resolution is a weighted proof system, where the final multiset must contain at least one copy of $+\bot$. The set of allowed rules is $\pm$Rev-Weaken, $\pm$Rev-Res, $\pm$ Import, Sign-Intro, Sign-Elim. By $\pm$Rev-Weaken we mean both the "+" and "−" versions of Rev-Weaken, and similarly for the two other rules. Here, Sign-Intro and Sign-Elim are the following rules:*

$$\frac{}{+C \quad -C} \textit{ Sign Intro}$$

$$\frac{+C \quad -C}{} \textit{ Sign Elim}$$

Unary Weighted Resolution as a proof system was already studied in [3], where it was shown to be equivalent to the semialgebraic proof system *unary Sherali-Adams*, defined next.

▶ **Definition 16.** *Given a CNF $F = C_1 \wedge \ldots \wedge C_m$, over variables $x_1, \ldots, x_n$, a Unary Sherali-Adams (uSA) refutation of F is given by a set of polynomials $\{P_i\} \cup \{J_0\} \in \mathbb{Z}[x_1, \ldots, x_n]$ such that :*

$$\sum_{i=1}^{m} P_i \bar{C}_i = J_0 + c$$

Where c is a constant such that $c \in \mathbb{N}_{\geq 1}$ is some constant and J_0 is a conical junta, that is a polynomial of the form $\sum_k c_k \prod_{i \in S_k} x_i \prod_{j \in T_k} (1 - x_j)$, where $c_k \in \mathbb{N}$.

The degree of the proof is defined as $\max(\{\deg(P_i) + \deg(C_i) | i \in [m]\} \cup \{\deg(J_0)\})$. The size of the proof is defined as follows: fully expand each of $P_j \bar{C}_i$ and J_0 and before cancellation, sum the absolute value of the coefficient appearing in front of each monomial.

3 Characterization of PPA_q

We first include the definition of problem Lonely_q, which is complete for the class PPA_q for any integer $q \geq 2$ [23].

▶ **Definition 17** (Lonely_q). *The problem is defined by considering a vertex set V, where $|V| \equiv 0 \mod q$. We are also given a set $V^* \subseteq V$ of distinguished nodes, with $|V^*| \leq q - 1$. Each node $v \in V$ has a successor pointer $s_v \in V$. For nodes $v_1, \ldots, v_q$ we add the hyperedge $\{v_1, \ldots, v_q\}$ iff $s_{v_1} = v_2, s_{v_2} = v_3 \ldots, s_{v_q} = v_1$.*

The goal of the search problem is to output either:

- *$v \in V^*$ if v is an hyperedge;*
- *$v \in V \setminus V^*$ if v isn't in an hyperedge, i.e. v is lonely.*

Often, we will abuse notation and say that we add the hyperedge $e := \{v_1, \ldots, v_q\}$ to a Lonely_q instance. This can be done by fixing some ordering on the nodes in V in advance, and then make each node $v \in e$ output the next element in e according to the ordering, unless v is the last node, in which case it outputs the first node according to the ordering.

3.1 Proof systems for PPA_q

3.1.1 $\mathcal{R}_q$-Resolution with Terminals

In this section, we show the equivalence between PPA_q and a new Resolution style proof system called $\mathcal{R}_q$-Resolution with Terminals. The key characteristic of this proof system is that it is allowed to create or remove q copies of an arbitrary clause C, thus preserving the number of copies mod q. This equivalence between PPA_q and $\mathcal{R}_q$-Resolution with Terminals will be useful in later sections, in particular when we examine Resolution-style proof systems for $\mathsf{PPA} \cap \mathsf{PLS}$ and $\mathsf{PPA}_q \cap \mathsf{SOPL}$.

▶ **Definition 18.** *Let $q \geq 2$. The clause deduction rules $\mathbb{F}_q$-Elim and $\mathbb{F}_q$-Intro are defined as follows:*

$$\frac{C \quad \dots \quad C}{} \ \mathbb{F}_q\text{-Elim} \qquad \frac{}{C \quad \dots \quad C} \ \mathbb{F}_q\text{-Intro}$$

The two above rules respectively allow us to remove q copies of an arbitrary clause C from a multiset, or add them.

Define the inference rule set $\mathcal{R}_q = \{$Rev-Res, Rev-Weaken, Import, $\mathbb{F}_q$-Elim, $\mathbb{F}_q$-Intro$\}$. The proof system $\mathcal{R}_q$-Resolution with Terminals, $\mathrm{ResT}_{\mathcal{R}_q}$, is a Resolution-style proof system with Terminals, where the final multiset must contain c copies of $\bot$ for some $1 \leq c < q$.

Let F be an unsatisfiable CNF formula. Any $\mathrm{ResT}_{\mathcal{R}_q}$ refutation of F of size s and width w, can be turned into another proof of size $s^{\mathcal{O}(1)}$ and width w where the final multiset contains only copies of $\bot$.

▶ **Theorem 19.** *Let F be an unsatifiable CNF formula,*

- *If there is a depth d, size s Lonely$_q$ formulation of $S(F)$ then there is $\mathrm{ResT}_{\mathcal{R}_q}$ refutation of F of size $s^{\mathcal{O}(1)}2^{\mathcal{O}(d)}$ and width $\mathcal{O}(d)$.*
- *If there is a width w, size s $\mathrm{ResT}_{\mathcal{R}_q}$ refutation of F then there is depth $w+1$, size $\mathcal{O}(s)$ Lonely$_q$ formulation of $S(F)$.*

3.1.2 Nullstellensatz over $\mathbb{Z}_q$

In this section, we show the equivalence between PPA_q and NS^*_q for all $q \geq 2$. In particular, this implies that NS^*_q and $\mathrm{ResT}_{\mathcal{R}_q}$ are equivalent as proof systems. The equivalence between NS_p and PPA_p was already shown in [21] for $p = 2$ and [27] any prime p. However, this new equivalence allows us to use the characterizations of PPA_q from [23] to study the relative powers of NS^* and NS_q in Section 3.2 for composite q.

▶ **Theorem 20.** *Let F be an unsatisfiable CNF formula,*

- *If there is a depth d, size s Lonely$_q$ formulation of $S(F)$ then there is NS^*_q refutation of F of size $s^{\mathcal{O}(1)}2^{\mathcal{O}(d)}$ and degree $\mathcal{O}(d)$.*
- *If there is a degree d, size s NS^*_q refutation of F then there is depth $\mathcal{O}(d)$, size $s^{\mathcal{O}(1)}$ Lonely$_q$ formulation of $S(F)$.*

3.2 Characterizations of Nullstellensatz

Using the relationship between NS^*_q and PPA_q from the above, we can obtain an interesting characterization of Nullstellensatz over $\mathbb{Z}_q$. In particular, we informally have that $\mathrm{NS}^*_q = \bigcup_{p|q} \mathrm{NS}_p$, while $\mathrm{NS}_q = \bigcap_{p|q} \mathrm{NS}_p$. This roughly states that an unsatisfiable CNF formula

has a low degree NS_q^* refutation if and only if has a low degree refutation in NS_p for **some** prime p diving q. On the other hand, an unsatisfiable CNF formula has a low degree NS_q refutation if and only if has a low degree refutation in NS_p for **all** prime p diving q.

Before proving these statements, note that a consequence of Theorem 3 is that $\mathsf{PPA}_p = \mathsf{PPA}_{p^k}$. While [23] don't prove the above theorem using the language of query TFNP, we can easily extract from their proof the following :

▶ **Lemma 21.** *Let F be an unsatisfiable CNF formula. A size s, depth d Lonely_{p^k} formulation of $S(F)$, can be turned into a size $s^{\mathcal{O}(1)}$ depth $\Theta(d)$ Lonely_p formulation of $S(F)$.*

In the proof of [23] each node v in the Lonely_p instance, corresponds to a subset $\mathcal{S}_v$ of at most p^k vertices from the Lonely_{p^k} instance. The successor function of v is computed by looking at all the hyperedges the nodes of $\mathcal{S}_v$ are. This can be computed by looking at $\mathcal{O}(p^{2k})$ decision trees. Finally a node v is lonely in the Lonely_p instance if and only if all the nodes in S_v were lonely, so when v is a solution, we can just run the decision tree of $v_1 \in S$.

We first have the following theorem regarding NS_q^* :

▶ **Theorem 22.** *Let F be an unsatisfiable CNF formula, and $q \geq 2$*
- *If $\mathrm{NS}_q^*(F) = d$ then $\mathrm{NS}_p(F) = \mathcal{O}(d)$ for some prime $p|q$.*
- *If $\mathrm{NS}_p(F) = d$ for some prime $p|q$, then $\mathrm{NS}_q^*(F) = \mathcal{O}(d)$.*

We also have the following theorem:

▶ **Theorem 23.** *Let F be an unsatisfiable CNF formula, and $q \geq 2$*
- *If $\mathrm{NS}_q(F) = d$ then $\mathrm{NS}_p(F) = \mathcal{O}(d)$ for all primes $p|q$.*
- *If $\mathrm{NS}_p(F) \leq d$ for all primes $p|q$, then $\mathrm{NS}_q(F) = \mathcal{O}(d)$.*

4 Characterizations of $\mathsf{PPA}_q \cap \mathsf{PPADS}$

In this section, we focus on the intersection class $\mathsf{PPA}_q \cap \mathsf{PPADS}$ for any constant integer $q \geq 2$ (so the class $\mathsf{PPA} \cap \mathsf{PPADS}$ is a special case in it). We first provide a graphical problem called SOL_q, which we show is complete for $\mathsf{PPA}_q \cap \mathsf{PPADS}$. This is the first complete problem distinct from $\textsc{Either}(\mathrm{Lonely}_q, \mathrm{SOL})$.

Next, we provide proof systems that characterize $\mathsf{PPA}_q \cap \mathsf{PPADS}$. In particular, we define q-Sherali-Adams (in Section 4.2) and q-Weighted Reversible Resolution (in Section 4.3) and show SOL_q is complete for them.

4.1 A complete problem for $\mathsf{PPA}_q \cap \mathsf{PPADS}$

We first provide the definition of SOL_q. Notably, when $q = 2$ (*i.e.*, consider $\mathsf{PPA} \cap \mathsf{PPADS}$), our definition naturally preserves the graphical structure of the original PPA and PPADS complete problems. It can be rephrased as follows: Given a directed graph in which every node has degree at most two (with some conditions) and a distinguished out-degree-1 source, find another degree-1 node.

▶ **Definition 24** (SOL_q)**.** *The problem is defined on a graph $G = (V, E)$, where $V = V_1 \cup V_2$. For input, we are given a subset $V^* \subseteq V_1$ with $1 \leq |V^*| \leq q - 1$ as distinguished sources, a successor pointer $s_v \in V_1 \cup \{0\}$ for each node $v \in V_1$, a predecessor pointer $p_v \in V \cup \{0\}$ for each node $v \in V_1 \setminus V^*$, and a successor pointer $s_v \in \binom{V_1}{q} \cup \{\varnothing\}$ for each node $v \in V_2$.*

For $v, u \in V$, we add edge $(v, u) \in E$ if and only if $u \in s_v$ and $v \in p_u$.

The goal of the search problem is to output either:

- $v \in V^*$ *if it has out-degree 0, i.e., it is not a source;*
- $v \in V_1 \setminus V^*$ *if it has degree 1, i.e., it is a source* or *a sink; or*
- $v \in V_2$ *if* $deg^+(v) \notin \{0, q\}$ *(equivalently* $deg^+(v) \not\equiv 0 \mod q$*).*

Intuitively, the problem SOL_q defined above is the same way as SOL except that

- The vertex set is split into V_1 and V_2. Where nodes in V_1 have in and out-degree at most 1, and nodes in V_2 have in-degree 0 and out-degree at most q.
- It is allowed to have sources with out-degree at most q (corresponding to nodes in V_2); and
- We are given at least 1, at most $q-1$ distinguished sources; the out-degree of each of them is at most one (meaning that they are all from V_1).

▶ **Theorem 25.** *SOL_q is $\mathsf{PPA}_q \cap \mathsf{PPADS}$-complete.*

4.2 q-Sherali-Adams

In this section we show the equivalence between $\mathsf{PPA}_q \cap \mathsf{PPADS}$ and q-Sherali Adams. Intuitively, a q-Sherali Adams proof is a more restricted Sherali-Adams proof, but taking the proof mod q gives a NS_q^* proof. As such, the following seems like a natural proof system for the intersection of PPA_q (NS_q^*) and PPADS (uSA).

▶ **Definition 26.** *Let $F = C_1 \wedge \ldots \wedge C_m$ be a CNF over variables $x_1, \ldots, x_n$. A q-Sherali-Adams (SA_q) refutation of F is a unary Sherali-Adams refutation, with the further constraints that $1 \leq c \leq q-1$ and the conical junta J_0 can be written as $q * J_0'$. That is, if we write $J_0 = \sum_k c_k \prod_{i \in S_k} x_i \prod_{j \in T_k} (1 - x_j)$ then we always have $c_k \equiv 0 \mod q$.*

We have the following equivalence between SA_q and SOL_q.

▶ **Theorem 27.** *Let F be an unsatisfiable CNF formula,*

- *If there is a depth d, size s SOL_q formulation of $S(F)$ then there is SA_q refutation of F of size $s^{\mathcal{O}(1)}2^{\mathcal{O}(d)}$ and degree $\mathcal{O}(d)$.*
- *If there is a degree d, size s SA_q refutation of F then there is depth $\mathcal{O}(d)$, size $s^{\mathcal{O}(1)}$ SOL_q formulation of $S(F)$.*

4.3 $\pm\mathcal{R}_q$-Weighted Resolution

In this section, we consider a form of Resolution, which we will show characterize $\mathsf{PPA}_q \cap \mathsf{PPADS}$ formulations.

▶ **Definition 28** ($\pm\mathcal{R}_q$-Weighted Resolution)**.** *Let $q \geq 2$. Define the weighted analogues of the $\mathbb{F}_q$-Elim and $\mathbb{F}_q$-Intro rules as follows:*

$$\frac{+C \ldots + C \quad -C \ldots - C}{} \pm\mathbb{F}_q \textit{ Elim}$$

$$\frac{}{+C \ldots + C \quad -C \ldots - C} \pm\mathbb{F}_q \textit{ Intro}$$

where there are q copies of $+$ and $-$ clauses in the above rules.

Define the inference rule set

$$\pm\mathcal{R}_q = \{\textit{Intro}, \pm\textit{Rev-Res}, \pm\textit{Rev-Weaken}, \pm\textit{Import}, \pm\mathbb{F}_q\textit{-Intro}, \pm\mathbb{F}_q\textit{-Elim}\}\,.$$

The proof system $\pm\mathcal{R}_q$-Weighted Resolution, $\mathrm{WRes}_{\pm\mathcal{R}_q}$*, is a weighted Resolution-style system where the final multiset must contain c copies of $+\bot$ where $1 \le c \le q-1$. Furthermore, all clauses in the final multiset must be labeled with a "+", and must be either a weakening of a clause from F or $+C$ must appear a multiple of q times.*

To show the equivalence between $\mathrm{WRes}_{\pm\mathcal{R}_q}$ and $\mathsf{PPA}_q \cap \mathsf{PPADS}$, we will use the PPADS complete problem Inj-PHP, defined as follow :

▶ **Definition 29** (Inj-PHP). *The problem is defined by two sets P and H, where $|P| = |H|+1$. We view P as a set of pigeons and H as a set of holes. As input we are given for each pigeon $p \in P$ a hole s_v, and for each hole $h \in H$ we are given a pigeon s_h.*

We add an edge from $p \in P$ to $h \in H$ (the pigeon p flew into the hole h) if $s_p = h$ and $s_h = p$.

The goal of the search problem is to output a missing pigeon, i.e., $p \in P$ such that p has out-degree 0.

We then have the following characterization of $\mathrm{WRes}_{\pm\mathcal{R}_q}$ proofs.

▶ **Theorem 30.** *Let F be an unsatisfiable CNF formula,*

- *If there are both a Lonely$_q$ and a Inj-PHP formulation of $S(F)$ of depth d and size s, then there is a width $\mathcal{O}(d)$ and size $s^{\mathcal{O}(1)}2^{\mathcal{O}(d)}$ $\mathrm{WRes}_{\pm\mathcal{R}_q}$.*
- *If there is width w and size s $\mathrm{WRes}_{\pm\mathcal{R}_q}$ refutation of F. Then there is depth $\mathcal{O}(w)$, size $s^{\mathcal{O}(1)}$ SOL$_q$ formulation of $S(F)$.*

5 Characterizations of $\mathsf{PPA}_q \cap \mathsf{SOPL}$

In this section, we provide our characterizations (complete problem and proof system) of $\mathsf{PPA}_q \cap \mathsf{SOPL}$ for any constant integer $q \ge 2$.

5.1 A complete problem for $\mathsf{PPA}_q \cap \mathsf{SOPL}$

We define a total search problem EOPL_q and show it is complete for $\mathsf{PPA}_q \cap \mathsf{SOPL}$. Recall that SOPL is essentially PPADS with extra potential on the nodes. We can expect that EOPL_q can be defined from SOL$_q$ in the same way (by adding potential on nodes). However, from the proof system perspective, we would like the potential increase by exactly one along the path. Also in some cases, it would be helpful to think about problems with potential on a grid, as introduced in [19]. These lead to the following definition.

▶ **Definition 31** (EOPL_q). *The problem is defined on a grid $[L] \times [2L]$. As input, we are given a subset $V^* \subseteq [L]$ with $1 \le |V^*| \le q-1$ as distinguished sources, a successor pointer $s_{i,j} \in [L]_0$ for each node $(i,j) \in [L] \times [L]$, a predecessor pointer $p_{i,j} \in [2L]_0$ for each node $(i,j) \in \{2, \cdots, L\} \times [L]$, and a successor pointer $s_{i,\alpha} \in \binom{[L]}{q} \cup \{\varnothing\}$ for each node $(i,\alpha) \in [L] \times ([2L] \setminus [L])$.*

For $i \in [L-1]$ and $\mathbf{a}, \mathbf{b} \in [2L]$, we add an edge between $(i,\mathbf{a})$ and $(i+1,\mathbf{b})$ if and only if $\mathbf{b} \in s_{i,\mathbf{a}}$ and $\mathbf{a} \in p_{i+1,\mathbf{b}}$.

The goal of the search problem is to output either:

- *$(1,j)$ for $j \in V^*$ if $(1,j)$ has out-degree 0, i.e., it is not a source;*
- *(L,j) for $j \in [L]$ if $s_{(L,j)} \ne 0$;*

- $(i,j) \in ([L] \times [L]) \setminus (\{1\} \times V^*)$ *if* (i,j) *has degree 1, i.e., it is a source or a sink; or*
- $(i,\alpha) \in [L] \times ([2L] \setminus [L])$ *if* $deg^+(i,\alpha) \notin \{0,q\}$ *(equivalently* $deg^+(i,\alpha) \neq 0 \mod q$*).*

We often refer to the nodes in the first L rows as type 1 and the nodes in the last L rows as type 2. We note that we can always equivalently consider instances for which the nodes on the last layer have no successor points, so there will be no second type of solution. However, we use the current definition of EOPL_q for simplicity when we work on proof systems later.

▶ **Theorem 32.** EOPL_q *is* $\mathsf{PPA}_q \cap \mathsf{SOPL}$*-complete.*

5.2 $\mathcal{R}_q^-$-Resolution with Terminals

In this section, we show the equivalence between $\mathsf{PPA}_q \cap \mathsf{SOPL}$ and the Resolution style proof system $\mathcal{R}_q^-$-Resolution with Terminals. Intuitively, this proof system should be weaker than both Reversible Resolution (SOPL) and $\mathrm{ResT}_{\mathcal{R}_q}$ (PPA_q). Reversible Resolution, doesn't have the ability to generate copies of an arbitrary clause C, as such the intersection of both proof systems shouldn't have the $\mathbb{F}_q$ Intro rule and it should have terminals. Intuitively, the reason that we are able to keep the $\mathbb{F}_q$ Elim rule, is that Reversible Resolution is a proof system that doesn't have terminals, so it has no need to "clean up" clauses, and thus even if it had the $\mathbb{F}_q$-Elim rule it likely wouldn't be more powerful. For these reasons, it seems natural that the proof system characterizing $\mathsf{PPA}_q \cap \mathsf{SOPL}$ should just be $\mathrm{ResT}_{\mathcal{R}_q}$, but with the $\mathbb{F}_q$-Intro rule removed.

▶ **Definition 33** ($\mathcal{R}_q^-$-Resolution with Terminals)**.** *Define the inference set* $\mathcal{R}_q^- := \mathcal{R}_q \setminus \{\mathbb{F}_q\text{-Intro}\}$*. The Resolution-style proof system* $\mathcal{R}_q^-$*-Resolution with Terminals,* $\mathrm{ResT}_{\mathcal{R}_q^-}$*, is a Resolution-style proof system with terminals where the final multiset must contain* c *copies of* $\bot$*, where* $1 \leq c < q$*.*

We have the following equivalence between $\mathrm{ResT}_{\mathcal{R}_q^-}$ and $\mathsf{PPA}_q \cap \mathsf{SOPL}$.

▶ **Theorem 34.** *Let* F *be an unsatisfiable CNF formula,*

- *If there is a depth* d*, size* s EOPL_q *formulation of* $S(F)$ *then there is* $\mathrm{ResT}_{\mathcal{R}_q^-}$ *refutation of* F *of size* $s^{\mathcal{O}(1)}2^{\mathcal{O}(d)}$ *and width* $\mathcal{O}(d)$*.*
- *If there is a width* w *and size* s $\mathrm{ResT}_{\mathcal{R}_q^-}$ *refutation of* F*. Then there is depth* $\mathcal{O}(w)$*, size* $s^{\mathcal{O}(1)}$ EOPL_q *formulation of* $S(F)$*.*

6 Characterizations of PPA ∩ PLS

6.1 A complete problem for PPA ∩ PLS

Ishizuka [25] has studied the intersection of PPA and PLS, giving a complete problem called PotentialOdd. In this section, we define a new complete problem on the grid, which is a slightly different variant of PotentialOdd. Ultimately, our main motivation here is to show the proof system defined in the subsequent section characterizes the class $\mathsf{PPA} \cap \mathsf{PLS}$, and our complete problem serves as the key bridge.

Intuitively, the problem defined below is the same as EOPL_2 except the nodes in $V_1 = [L] \times [L]$ can have at most three predecessors.

▶ **Definition 35** (MaxOdd)**.** *The problem is defined on a grid* $[L] \times [2L]$*, where* $(1,1)$ *is the distinguished source. As input, we are given a successor pointer* $s_{i,j} \in [L]_0$ *for each node* $(i,j) \in [L] \times [L]$*, a predecessor pointer* $p_{i,j} \in \cup_{k=0}^{3} \binom{[2L]}{k}$ *for each node* $(i,j) \in \{2, \cdots, L\} \times [L]$*, and a successor pointer* $s_{i,j} \in \binom{[L]}{2} \cup \{\varnothing\}$ *for each node* $(i,\alpha) \in [L] \times ([2L] \setminus [L])$*.*

For $i \in [L-1]$ and $\mathbf{a}, \mathbf{b} \in [2L]$, we add an edge between $(i, \mathbf{a})$ and $(i+1, \mathbf{b})$ if and only if $\mathbf{b} \in s_{i,\mathbf{a}}$ and $\mathbf{a} \in p_{i+1,\mathbf{b}}$.

The goal of the search problem is to output either:

- *$(1,1)$ if $(1,1)$ has out-degree 0, i.e., it is not a source;*
- *(L, j) for $j \in [L]$ if $s_{(L,j)} \neq 0$;*
- *$(i, \mathbf{a}) \in ([L] \times [2L]) \setminus (1,1)$ if $(i, \mathbf{a})$ has odd degree; or*
- *$(i, \mathbf{a}) \in [L] \times [2L]$ if $deg^-(i, \mathbf{a}) \neq 0$ and $deg^+(i, \mathbf{a}) = 0$, i.e., $(i, \mathbf{a})$ is a sink.*

To avoid confusion, we note that we can without loss of generality consider instances of MaxOdd such that they are "promised" not to have the first and second type solutions, namely, $(1,1)$ is promised to be a source and $s_{(L,j)}$ is promised to be 0. This follows from an easy argument by locally changing the graph; we omit details of it.

▶ **Theorem 36.** MaxOdd *is* PPA ∩ PLS*-complete.*

6.2 $\mathcal{R}_2^{copy}$-Resolution with Terminals

In this section, we show the equivalence between PPA ∩ PLS and the Resolution style proof system $\mathcal{R}_2^{copy}$-Resolution with Terminals. Despite the fact that the set of inference rules $\mathcal{R}_2^{copy}$ defining the system are relatively simple, the proof of this characterization is (by far) the most technical result in this work.

Intuitively, this proof system should be less powerful than $\mathrm{ResT}_{\mathcal{R}_2}$, where we can create 2 copies of an arbitrary clause C. The proof system should also be weaker than Resolution, where we can't generate copies of an arbitrary clause, but we can make an extra copy of C if it has already been derived. A natural intersection of the Copy rule and the $\mathbb{F}_2$-Intro rule is the following $\mathbb{F}_2$-Copy rule: $C \vdash C, C, C$. This allows us to both copy clauses *and* preserve the parity of each multiset, and as such it seems natural that the following proof system should correspond to the intersection of PPA ($\mathrm{ResT}_{\mathcal{R}_2}$) and PLS (Resolution).

▶ **Definition 37** ($\mathcal{R}_2^{copy}$-Resolution with Terminals)**.** *The $\mathbb{F}_2$-Copy rule is defined to be*

$$\frac{C}{C \quad C \quad C} \ \mathbb{F}_2\text{-Copy}$$

Define the set of inference rules $\mathcal{R}_2^{copy} := \mathcal{R}_2 \cup \{\mathbb{F}_2\text{-Copy}\} \setminus \{\mathbb{F}_2\text{-Intro}\}$. The proof system $\mathcal{R}_2^{copy}$-Resolution with Terminals, $\mathrm{ResT}_{\mathcal{R}_2^{copy}}$, is a Resolution-style proof system with terminals where the final multiset must contain a copy of $\bot$.

We have the following equivalence between $\mathrm{WRes}_{\pm\mathcal{R}_q}$ and $\mathsf{PPA}_q \cap \mathsf{PPADS}$.

▶ **Theorem 38.** *Let F be an unsatisfiable CNF formula,*

- *If there is a depth d, size s* MaxOdd *formulation of $S(F)$ then there is $\mathrm{ResT}_{\mathcal{R}_q^-}$ refutation of F of size $s^{\mathcal{O}(1)}2^{\mathcal{O}(d)}$ and width $\mathcal{O}(d)$.*
- *If there is a width w and size s $\mathrm{ResT}_{\mathcal{R}_2^{copy}}$ refutation of F. Then there is a SOD and a Lonely formulation of $S(F)$, each of depth $\mathcal{O}(w)$ and size $s^{\mathcal{O}(1)}$.*

7 Separations

In this section we provide oracle separations between the newly introduced intersection classes PPA ∩ PLS, PPA ∩ PPADS, PPA ∩ SOPL and the previously defined TFNP classes. These separation results, combined with earlier separation results, imply that no further collapses occur in the black-box setting. The next theorem records the main new separation results.

▶ **Theorem 39.** *We have the following separations in the decision-tree setting (and thus, also with respect to a generic oracle O):*

- $\mathsf{PLS}^{dt} \cap \mathsf{PPA}^{dt} \not\subseteq \mathsf{PPADS}^{dt}$,
- $\mathsf{SOPL}^{dt} \cap \mathsf{PPA}^{dt} \not\subseteq \mathsf{PPAD}^{dt}$.

The theorem clearly implies that the three classes $\mathsf{SOPL}^{dt} = \mathsf{PPADS}^{dt} \cap \mathsf{PLS}^{dt}, \mathsf{PLS}^{dt} \cap \mathsf{PPA}^{dt}$, and $\mathsf{PPADS}^{dt} \cap \mathsf{PPA}^{dt}$ are all distinct in the black-box model, as if any two collapsed then it would either violate the new non-containment above, or, one of the known separations $\mathsf{SOPL}^{dt} \not\subseteq \mathsf{PPA}^{dt}$ [20] or $\mathsf{PPAD}^{dt} \not\subseteq \mathsf{PLS}^{dt}$ [29, 4]. By a similar argument, none of the three classes can collapse downwards to $\mathsf{SOPL}^{dt} \cap \mathsf{PPA}^{dt}$, as this would similarly violate those separation results. We further note that none of the classes $\mathsf{PLS}^{dt}, \mathsf{PPADS}^{dt}$ or PPA^{dt} can collapse downwards to any of $\mathsf{SOPL}^{dt}, \mathsf{PLS}^{dt} \cap \mathsf{PPA}^{dt}$, or $\mathsf{PPADS}^{dt} \cap \mathsf{PPA}^{dt}$, as this would violate the known separations between the classes $\mathsf{PLS}^{dt}, \mathsf{PPADS}^{dt}$, and PPA^{dt} [1, 20]. Finally, if $\mathsf{PPADS}^{dt} \cap \mathsf{PPA}^{dt}$ or $\mathsf{SOPL}^{dt} \cap \mathsf{PPA}^{dt}$ were contained in PPAD^{dt} then the above theorem will be violated, and conversely if $\mathsf{PPAD}^{dt} \subseteq \mathsf{SOPL}^{dt} \cap \mathsf{PPA}^{dt}$ then $\mathsf{PPAD}^{dt} \subseteq \mathsf{PLS}^{dt}$, which violates a known separation due to [29, 4]. Thus, once we prove Theorem 39 the classification theorem Theorem 1 will be completely proved.

7.1 Separations for PLS ∩ PPA

In this section we show the following lemma.

▶ **Theorem 40.** $\mathsf{PLS}^{dt} \cap \mathsf{PPA}^{dt} \not\subseteq \mathsf{PPADS}^{dt}$.

This separation follows by combining several tools from the literature. First, we crucially rely on the following characterization of PPADS^{dt} via low-degree Sherali-Adams refutations.

▶ **Theorem 41** ([20])**.** *A query total search problem $R \in \mathsf{PPADS}^{dt}$ iff the associated family of unsatisfiable CNF formulas F_R has $\log^{O(1)}(n)$-degree, quasipolynomial-size Sherali-Adams refutations over $\mathbb{Z}$ where the coefficients are written in unary.*

We also need the characterization of Sherali-Adams degree by *pseudoexpectation operators*. If F is an unsatisfiable CNF formula, recall that a *degree-d pseudoexpectation operator* $\tilde{\mathbb{E}}$ is a linear functional mapping multilinear polynomials p to $\mathbb{Q}$ such that the following holds:

- $\tilde{\mathbb{E}}[1] = 1$
- For every clause C in F and every monomial m with $\deg(\overline{C}m) \leq d$, $\tilde{\mathbb{E}}[\overline{C}m] = 0$
- For every clause C with $\deg(\overline{C}) \leq d$, $\tilde{\mathbb{E}}[\overline{C}] \geq 0$.

The following result characterizes the minimum degree of Sherali-Adams refutations in terms of the existence of pseudoexpectation operators.

▶ **Theorem 42** ([18])**.** *For any CNF formula F, there is no degree-d Sherali-Adams refutation of F if and only if there is a degree-d pseudoexpectation operator $\tilde{\mathbb{E}}$ for F.*

We also use the following result [24, 18] showing the existence of a pseudoexpectation operator for certain unsatisfiable systems of 3XOR equations.

▶ **Theorem 43.** *There are unsatisfiable systems $Ax = b$ of 3XOR equations on n variables and $m = O(n)$ constraints such that any Sherali-Adams refutation of the CNF encoding of $Ax = b$ requires degree $\Omega(n)$. Furthermore, there is a pseudoexpectation $\tilde{\mathbb{E}}$ witnessing this such that $2^{-\deg(m)} \leq |\tilde{\mathbb{E}}[m]| \leq 1$ for every monomial m.*

Finally, we will use the characterization of PPA^{dt} by Nullstellensatz degree over $\mathbb{F}_2$.

▶ **Theorem 44** ([22]). *Let F be an unsatisfiable CNF formula. If there is a degree-d, size-s Nullstellensatz refutation of F over $\mathbb{F}_2$, then there is a PPA^{dt} formulation of $S(F)$ with depth $O(d)$ and size $O(s2^d)$.*

We combine these results together to prove the above separation.

Proof of Theorem 40. We show that if $\mathsf{PLS}^{dt} \cap \mathsf{PPA}^{dt} \subseteq \mathsf{PPADS}^{dt}$, then $\mathsf{PLS}^{dt} \subseteq \mathsf{PPADS}^{dt}$. This violates the known separation $\mathsf{PLS}^{dt} \not\subseteq \mathsf{PPADS}^{dt}$ due to [20].

Define the unsatisfiable CNF formula $F_n := \mathrm{SoD}_n(x) \wedge \mathrm{G}_n(y)$, where the formulas are understood to be on disjoint sets of variables, and the formula $\mathrm{G}_n(y)$ is the CNF encoding of the 3XOR system from Theorem 43. We first observe that there is a $O(m)$-size, $O(1)$-degree NS_2 refutation of $\mathrm{G}_n(y)$. We may assume without loss of generality that the system of equations $Ay = b$ is minimally unsatisfiable, and thus summing up all equations yields $0 = 1$. Expand $\mathrm{G}_n(y) = \bigwedge_{i=1}^{m} \bigwedge_j C_{ij}$, where $\bigwedge_j C_{ij}$ are the clauses encoding the ith XOR constraint $a_i y = b_i$. Since summing up all equations in the system $Ay = b$ yields the system $0 = 1$, it follows that

$$\sum_{i=1}^{m} \sum_j \overline{C}_{ij} = 1 \bmod 2$$

which is exactly our $O(m)$-size, $O(1)$-degree Nullstellensatz refutation of $\mathrm{G}_n(y)$. By applying Theorem 44, we can therefore write $\mathrm{G}_n(y) = \mathrm{Lonely}_{p(n)}(T(y))$ for some polynomial p and some $O(1)$-depth decision-tree substitution T. Applying the assumption that $\mathsf{PLS}^{dt} \cap \mathsf{PPA}^{dt} \subseteq \mathsf{PPADS}^{dt}$ and the characterization of PPADS^{dt} by Sherali-Adams refutations, we can conclude that F_n has a $\log^{O(1)} n$-degree Sherali-Adams refutation of quasipolynomial size, where every coefficient is written in unary.

Now, writing $\mathrm{SoD}_n = \bigwedge_j D_j$, consider the Sherali-Adams refutation of F_n

$$-1 = \sum_k p_k \overline{D}_k + \sum_{i=1}^{m} \sum_j q_{ij} \overline{C}_{ij} + \mathcal{J}$$

where p_k and q_{ij} are $\log^{O(1)} n$-degree polynomials, and $\mathcal{J}$ is a $\log^{O(1)} n$-degree conical junta, all over the variables x, y. Let $\tilde{\mathbb{E}}$ be the pseudoexpectation for $G_n(y)$ guaranteed by Theorem 40, and note that $1/2^{\deg(m)} \leq |\tilde{\mathbb{E}}[m]| \leq 1$ for every monomial m over the y variables. Extend $\tilde{\mathbb{E}}$ to apply to polynomials over both x and y variables by defining

$$\tilde{\mathbb{E}}\left[\prod_i x_i \prod_j y_j\right] = \prod_i x_i \cdot \tilde{\mathbb{E}}\left[\prod_j y_j\right],$$

and then extending $\tilde{\mathbb{E}}$ to apply to all polynomials over x and y variables by linearity. Applying $\tilde{\mathbb{E}}$ to the Sherali-Adams refutation yields

$$\begin{aligned} -1 &= \tilde{\mathbb{E}}[-1] \\ &= \sum_k \tilde{\mathbb{E}}[p_k \overline{D}_k] + \sum_{i=1}^{m} \sum_j \tilde{\mathbb{E}}[q_{ij} \overline{C}_{ij}] + \tilde{\mathbb{E}}[\mathcal{J}] \\ &= \sum_{i=1}^{m} \sum_j q'_{ij} \overline{C}_{ij} + \mathcal{J}'. \end{aligned}$$

where q'_{ij} and $\mathcal{J}'$ are new polynomials and conical juntas over only x variables, respectively. Finally, we note that since the original Sherali-Adams refutation had degree $\log^{O(1)} n$ and

total coefficient size $n^{\log^{O(1)} n}$, then since $1/2^{\deg(m)} \leq |\tilde{\mathbb{E}}[m]| \leq 1$ for every monomial m, it follows that the new Sherali-Adams refutation also has bounded coefficient size. Thus the above is a quasipolynomial-size, $\log^{O(1)} n$-degree Sherali-Adams refutation of SoD_n, implying $\mathsf{PLS}^{dt} \subseteq \mathsf{PPADS}^{dt}$, which is a contradiction. ◀

7.2 Separations for SOPL ∩ PPA

In this section we show the following lemma.

▶ **Lemma 45.** $\mathsf{SOPL}^{dt} \cap \mathsf{PPA}^{dt} \not\subseteq \mathsf{PPAD}^{dt}$.

This result follows from prior lower-bound results, along with the following characterization of PPAD^{dt} via Nullstellensatz with integer coefficients.

▶ **Theorem 46** ([20])**.** *A query total search problem $R \in \mathsf{PPAD}^{dt}$ iff the associated family of unsatisfiable CNF formulas F_R has a $\log^{O(1)}(n)$-degree, quasipolynomial-size Nullstellensatz refutation over $\mathbb{Z}$ when the coefficients are written in unary.*

References

1 Paul Beame, Stephen A. Cook, Jeff Edmonds, Russell Impagliazzo, and Toniann Pitassi. The relative complexity of NP search problems. *J. Comput. Syst. Sci.*, 57(1):3–19, 1998. doi:10.1006/jcss.1998.1575.

2 Paul Beame, Russell Impagliazzo, Jan Krajíček, Toniann Pitassi, and Pavel Pudlák. Lower bounds on Hilbert's Nullstellensatz and propositional proofs. In *Proceedings of the 35th Symposium on Foundations of Computer Science (FOCS)*, pages 794–806, 1994. doi:10.1109/SFCS.1994.365714.

3 Ilario Bonacina and Maria Luisa Bonet. On the strength of sherali-adams and nullstellensatz as propositional proof systems. In *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '22, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3531130.3533344.

4 Joshua Buresh-Oppenheim and Tsuyoshi Morioka. Relativized NP search problems and propositional proof systems. In *Proceedings of the 19th IEEE Conference on Computational Complexity (CCC)*, pages 54–67, 2004. doi:10.1109/CCC.2004.1313795.

5 Sam Buss, Noah Fleming, and Russell Impagliazzo. TFNP characterizations of proof systems and monotone circuits. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPIcs*, pages 30:1–30:40. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ITCS.2023.30.

6 Sam Buss and Jan Krajícek. An application of boolean complexity to separation problems in bounded arithmetic. *Proceedings of the London Mathematical Society*, 69:1–21, 1994.

7 Samuel R Buss. Lower bounds on nullstellensatz proofs via designs. *Proof complexity and feasible arithmetics*, 39:59–71, 1996.

8 Samuel R. Buss and Alan S. Johnson. Propositional proofs and reductions between NP search problems. *Ann. Pure Appl. Log.*, 163(9):1163–1182, 2012. doi:10.1016/j.apal.2012.01.015.

9 Xi Chen, Decheng Dai, Ye Du, and Shang-Hua Teng. Settling the complexity of arrow-debreu equilibria in markets with additively separable utilities. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 273–282. IEEE Computer Society, 2009. doi:10.1109/FOCS.2009.29.

10 Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player nash equilibria. *J. ACM*, 56(3):14:1–14:57, 2009. doi:10.1145/1516512.1516516.

11 Xi Chen, David Durfee, and Anthi Orfanou. On the complexity of nash equilibria in anonymous games. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 381–390. ACM, 2015. doi:10.1145/2746539.2746571.

12 Xi Chen, Dimitris Paparas, and Mihalis Yannakakis. The complexity of non-monotone markets. *J. ACM*, 64(3):20:1–20:56, 2017. doi:10.1145/3064810.

13 Bruno Codenotti, Amin Saberi, Kasturi R. Varadarajan, and Yinyu Ye. The complexity of equilibria: Hardness results for economies via a correspondence with games. *Theor. Comput. Sci.*, 408(2-3):188–198, 2008. doi:10.1016/j.tcs.2008.08.007.

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

15 Ben Davis and Robert Robere. Colourful TFNP and propositional proofs. In Amnon Ta-Shma, editor, *38th Computational Complexity Conference, CCC 2023, July 17-20, 2023, Warwick, UK*, volume 264 of *LIPIcs*, pages 36:1–36:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.CCC.2023.36.

16 John Fearnley, Paul Goldberg, Alexandros Hollender, and Rahul Savani. The complexity of gradient descent: CLS = PPAD ∩ PLS. *J. ACM*, 70(1):7:1–7:74, 2023. doi:10.1145/3568163.

17 John Fearnley, Spencer Gordon, Ruta Mehta, and Rahul Savani. Unique end of potential line. *J. Comput. Syst. Sci.*, 114:1–35, 2020. doi:10.1016/j.jcss.2020.05.007.

18 Noah Fleming, Pravesh Kothari, and Toniann Pitassi. Semialgebraic proofs and efficient algorithm design. *Foundations and Trends in Theoretical Computer Science*, 14(1-2):1–221, 2019. doi:10.1561/0400000086.

19 Mika Göös, Alexandros Hollender, Siddhartha Jain, Gilbert Maystre, William Pires, Robert Robere, and Ran Tao. Further collapses in TFNP. In Shachar Lovett, editor, *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*, volume 234 of *LIPIcs*, pages 33:1–33:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.CCC.2022.33.

20 Mika Göös, Alexandros Hollender, Siddhartha Jain, Gilbert Maystre, William Pires, Robert Robere, and Ran Tao. Separations in proof complexity and TFNP. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 1150–1161. IEEE, 2022. doi:10.1109/FOCS54457.2022.00111.

21 Mika Göös, Pritish Kamath, Robert Robere, and Dmitry Sokolov. Adventures in monotone complexity and tfnp. *Electron. Colloquium Comput. Complex.*, TR18-163, 2018. URL: https://eccc.weizmann.ac.il/report/2018/163.

22 Mika Göös, Pritish Kamath, Robert Robere, and Dmitry Sokolov. Adventures in monotone complexity and TFNP. In Avrim Blum, editor, *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, volume 124 of *LIPIcs*, pages 38:1–38:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ITCS.2019.38.

23 Mika Göös, Pritish Kamath, Katerina Sotiraki, and Manolis Zampetakis. On the complexity of modulo-q arguments and the chevalley-warning theorem. In *Proceedings of the 35th Computational Complexity Conference*, CCC '20, Dagstuhl, DEU, 2020. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CCC.2020.19.

24 Dima Grigoriev. Linear lower bound on degrees of positivstellensatz calculus proofs for the parity. *Theor. Comput. Sci.*, 259(1-2):613–622, 2001. doi:10.1016/S0304-3975(00)00157-2.

25 Takashi Ishizuka. The complexity of the parity argument with potential. *J. Comput. Syst. Sci.*, 120:14–41, 2021. doi:10.1016/j.jcss.2021.03.004.

26 David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. How easy is local search? *J. Comput. Syst. Sci.*, 37(1):79–100, 1988. doi:10.1016/0022-0000(88)90046-3.

27 Pritish Kamath. *Some hardness escalation results in computational complexity theory*. PhD thesis, Massachusetts Institute of Technology, 2019.

28 Ruta Mehta. Constant rank two-player games are ppad-hard. *SIAM J. Comput.*, 47(5):1858–1887, 2018. doi:10.1137/15M1032338.

29 Tsuyoshi Morioka. Classification of search problems and their definability in bounded arithmetic. Master's thesis, University of Toronto, 2001. URL: https://www.collectionscanada.ca/obj/s4/f2/dsk3/ftp04/MQ58775.pdf.

30 Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. Syst. Sci.*, 48(3):498–532, 1994. doi:10.1016/S0022-0000(05)80063-7.

31 Pavel Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *J. Symb. Log.*, 62(3):981–998, 1997. doi:10.2307/2275583.

32 Pavel Pudlák. On reducibility and symmetry of disjoint NP pairs. *Theor. Comput. Sci.*, 295:323–339, 2003. doi:10.1016/S0304-3975(02)00411-5.

33 Pavel Pudlák. On the complexity of finding falsifying assignments for herbrand disjunctions. *Arch. Math. Log.*, 54(7-8):769–783, 2015. doi:10.1007/s00153-015-0439-6.

34 Alejandro A. Schäffer and Mihalis Yannakakis. Simple local search problems that are hard to solve. *SIAM J. Comput.*, 20(1):56–87, 1991. doi:10.1137/0220004.

A Missing Definitions of TFNP Problems

▶ **Definition 47** (SOL). *The problem is defined on a graph with $[n]$ nodes. As input, we are given a successor pointer $s_v \in [n]$ for every $v \in [n]$, and a predecessor pointer $p_v \in [n]_0$ for every $v \in [n] \setminus \{1\}$.*

For $v, u \in [n]$, we add edge edge (v, u) if and only if $u \in s_v$ and $v \in s_u$.

The goal of the search problem is to output either:

- *The node 1 if $deg^+(1) \neq 1$, i.e., it is not a source.*
- *The node v if $deg^-(v) = 1$ and $deg^+(v) = 0$, i.e., it is a sink.*

▶ **Definition 48** (SOD). *The problem is defined on a grid $[L] \times [L]$, where the node $(1, 1)$ is the distinguished source. As input, we are given a successor pointer $s_{i,j} \in [L]_0$ for each node $(i, j) \in [L] \times [L]$.*

We say a node (i, j) is active if $s_{i,j} \neq 0$, otherwise it is inactive.

The goal of the search problem is to output either

- *$(1, 1)$ if $s_{1,1} = 0$, i.e., it is not a source;*
- *(n, j) if $s_{n,j} \neq 0$, i.e., it is an active sink;*
- *(i, j) if $s_{i,j} \neq 0$ and $s_{i+1,s_{i,j}} = 0$, i.e., (i, j) is active and $(i + 1, s_{i,j})$ is a sink.*

Intuitively, the SOPL defined below is the same as SOD but also has the predecessor pointer.

▶ **Definition 49** (SOPL). *The problem is defined on a grid $[L] \times [L]$, where $(1, 1)$ is the distinguished source. As input, we are given a successor pointer $s_{i,j} \in [L]_0$ for each node $(i, j) \in [L] \times [L]$, and a predecessor pointer $p_{i,j} \in [L]_0$ for each node $(i, j) \in ([L] \setminus \{1\}) \times [L]$.*

We add an edge between (i, j) and $(i + 1, k)$ for $i \in [L - 1]$ and $j, k \in [L]$ if and only if $s_{i,j} = k$ and $p_{i+1,k} = j$.

The goal of the search problem is to output either

- *$(1, 1)$ if $deg^+(1, 1) = 0$, i.e., $(1, 1)$ is a source;*
- *(n, j) if $s_{n,j} \neq 0$, i.e., it is an active sink;*
- *(i, j) if $deg^+(i, j) = 1$ and $deg^+(i + 1, s_{i,j}) = 0$, i.e., (i, j) is active and $(i + 1, s_{i,j})$ is a sink.*