



Separations in Proof Complexity and TFNP

Mika Göös *EPFL* Alexandros Hollender *University of Oxford* Siddhartha Jain *UT Austin* Gilbert Maystre *EPFL*
 William Pires *Columbia University* Robert Robere *McGill University* Ran Tao *Carnegie Mellon University*

July 22, 2024

Abstract. It is well-known that Resolution proofs can be efficiently simulated by Sherali–Adams (SA) proofs. We show, however, that any such simulation needs to exploit huge coefficients: Resolution cannot be efficiently simulated by SA when the coefficients are written in unary. We also show that *Reversible Resolution* (a variant of MaxSAT Resolution) cannot be efficiently simulated by Nullstellensatz (NS).

These results have consequences for total NP search problems. First, we characterise the classes PPADS, PPAD, SOPL by unary-SA, unary-NS, and Reversible Resolution, respectively. Second, we show that, relative to an oracle, $\text{PLS} \not\subseteq \text{PPP}$, $\text{SOPL} \not\subseteq \text{PPA}$, and $\text{EOPL} \not\subseteq \text{UEOPL}$. In particular, together with prior work, this gives a complete picture of the black-box relationships between all classical TFNP classes introduced in the 1990s.

Contents

1	Separations in Proof Complexity	1
2	Separations in TFNP	4
3	Definitions	9
4	Reversible Resolution vs. Nullstellensatz	12
5	Resolution vs. Sherali–Adams	17
6	Proofs of Characterisations	20
7	Intersection Theorems	35
8	Two Further Separations	36
A	Appendix: Coefficient Size in Algebraic Proofs	41
	References	44

1 Separations in Proof Complexity

The main results of this work are two separations between standard propositional proof systems, as summarised in [Figure 1](#). Moreover, these results can be further interpreted as black-box separations in the theory of total NP search problems (TFNP), as we explain later in [Section 2](#). This connection between TFNP and proof complexity, which has proved fruitful in past works and which we further explore here, also yields a new type of result in proof complexity, which we call *intersection theorems*; see [Section 2.4](#).

1.1 Resolution vs. Sherali–Adams

Our first separation is between the most basic and well-studied proof system *Resolution* (see the textbooks [[Juk12](#), [Kra19](#)] for an introduction) and the semi-algebraic proof system *Sherali–Adams* [[SA94](#), [DM12](#)] (see the monograph [[FKP19](#)] for an introduction). Let us briefly recall these systems. Each system aims to refute a given CNF contradiction (unsatisfiable CNF formula) $F := C_1 \wedge \dots \wedge C_m$ over the n boolean variables $x = (x_1, \dots, x_n)$.

Resolution (Res). A Resolution refutation of F starts with the set of clauses of F and repeatedly applies the *resolution rule* $C \vee x_i, D \vee \bar{x}_i \vdash C \vee D$. That is, if we have already deduced premise clauses $C \vee x_i$ and $D \vee \bar{x}_i$ for some i , then we can further deduce the clause $C \vee D$. Once this rule has been applied enough times to produce the empty clause \perp , the refutation is complete. The *size* of the refutation is the number of deduction steps, and its *width* is the maximum width $|C|$ (number of literals) of any clause C appearing in the refutation.

Sherali–Adams (SA). Sherali–Adams refutes unsatisfiable sets of polynomial equations $\{a_i(x) = 0 : i \in [m]\}$ with real coefficients, $a_i \in \mathbb{R}[x]$. A CNF contradiction F can be translated into this language by encoding each clause, say, $C := (x_1 \vee \bar{x}_2 \vee x_3)$, as the equation $(1 - x_1)x_2(1 - x_3) = 0$, and by enforcing each variable x_i to take boolean values by the equation $x_i^2 - x_i = 0$. An SA refutation of $\{a_i(x) = 0\}$ is a polynomial identity of the form¹

$$\sum_{i \in [m]} p_i(x) \cdot a_i(x) = 1 + J(x), \tag{1}$$

where $p_i \in \mathbb{R}[x]$ are polynomials and J is a *conical junta*: a nonnegative linear combination of terms, that is, $J(x) = \sum_j \alpha_j \cdot t_j(x)$ where $\alpha_j \in \mathbb{R}_{\geq 0}$ are nonnegative coefficients and each t_j is a conjunction of literals; for example, $t_j(x) = x_1 \bar{x}_2 x_3 = x_1(1 - x_2)x_3$. The *size* of the refutation is the combined total number of monomials in p_i , a_i , and t_j (viewed as a polynomial) and its *degree* is the maximum of $\deg(p_i) + \deg(a_i)$ and of $\deg(t_j)$ over all i, j .

It is a basic fact that SA is strictly more powerful than Resolution. First, Resolution is *p-simulated* by SA, that is, with only polynomial overhead in proof width/degree and size. Indeed, if F can be refuted by width- w Resolution, then F can be refuted by SA in degree $w + 1$ [[DMR09](#)]. Moreover, if one allows *twin variables* in SA, the simulation can also be made efficient relative to size [[ALN16](#)]. Second, SA is not *p-simulated* by Resolution: there are n -variate CNF contradictions F (e.g., graph pigeonhole principles) that can be refuted by SA in constant degree but such that any Resolution refutation of F requires width $\Omega(n)$ and size $\exp(\Omega(n))$ [[AL19](#)].

Our first result highlights a previously overlooked inefficiency in the way that SA simulates Resolution. We show that any low-degree simulation needs to exploit huge coefficients.

¹This particular form is valid for refuting sets of polynomial equations, and can be easily obtained from more general forms used for refuting sets of polynomial inequalities.

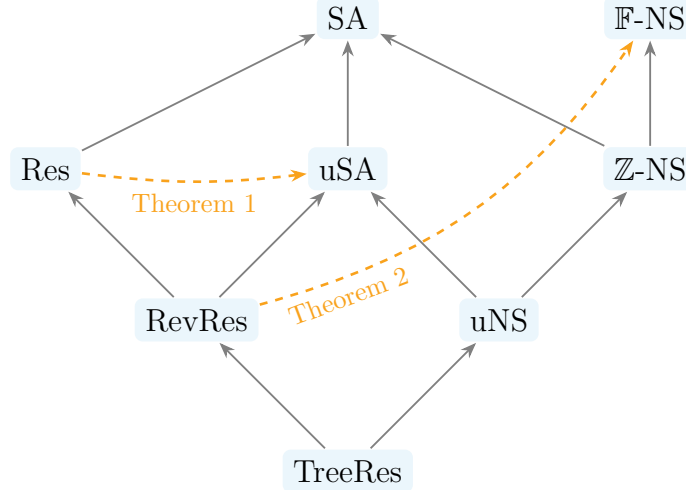


Figure 1: Our new separations of proof systems. An arrow $A \rightarrow B$ means that A is p -simulated by B , that is, with polynomial overhead in width/degree and size (when allowing twin variables). A dashed arrow $A \dashrightarrow B$ means that A is not p -simulated by B .

Theorem 1. *There are n -variate CNF formulas F that can be refuted by constant-width Resolution, but such that any SA refutation of F in degree $n^{o(1)}$ requires coefficients of magnitude $\exp(n^{\Omega(1)})$.*

Theorem 1 is qualitatively tight in that the singly-exponential lower bound $\exp(n^{\Omega(1)})$ cannot be improved much. Namely, if a CNF formula can be refuted by a degree- d SA proof, then there also exist a degree- d SA proof with integer coefficients of magnitude $\exp(n^{O(d)})$ (see [Appendix A](#)). To our knowledge, **Theorem 1** is the first exponential coefficient lower bound for a constant-width CNF formula in any semi-algebraic proof system. (Examples of systems of polynomial equations—not coming from CNFs—requiring even doubly-exponential coefficients were known previously [[O’D17](#), [RW17](#), [Hak21](#)].)

We also note that the conclusion of **Theorem 1** can be slightly strengthened using standard *lifting/xorification* techniques [[Ben09](#), §4] to show that any SA refutation of F must either use exponentially many monomials or exponentially large coefficients. This trade-off has consequences for the *unary Sherali–Adams* (uSA) system where we restrict the coefficients to be integers written in *unary* and where their magnitude counts towards proof size (more precisely, the size of a uSA proof is the sum of the magnitudes of all coefficients appearing in the proof). Thus we conclude that Resolution is not p -simulated by uSA. In particular, this answers a question raised in a concurrent work by Bonacina and Bonet [[BB22](#)]. For comparison, proving a similar lower bound for the Cutting Planes system (separating CP from unary-CP) is a long-standing open problem.

1.2 Reversible Resolution vs. Nullstellensatz

Our second separation is between the standard algebraic proof system Nullstellensatz [[BIK⁺94](#)] and a subsystem of Resolution that we call *Reversible Resolution*. The latter is closely related to fragments of Resolution that have been introduced to model the reasoning used by *MaxSAT* solvers (which find an assignment that satisfies as many clauses as possible). Prior work has defined several distinct such *MaxSAT Resolution* systems [[BLM07](#), [LHdG08](#), [FMSV23](#)]. Our variant is yet slightly different (see [Section 6.3](#) for a comparison to prior systems). Ultimately, our definition is motivated by results that will be discussed in [Section 2](#): Reversible Resolution captures an important TFNP

class, and, moreover, it equals the “intersection” of Resolution and uSA.

Reversible Resolution (RevRes). In this restricted fragment of Resolution we only allow the *symmetric resolution rule* $C \vee x_i, C \vee \bar{x}_i \vdash C$ and its inverse $C \vdash C \vee x_i, C \vee \bar{x}_i$. Moreover, we stipulate that an application of either rule *consumes its premises* in the following sense. The refutation begins with a *multiset* of clauses of F —we may choose the multiplicity of each clause freely at start—and a single application of a deduction rule removes a single occurrence of each premise clause from the multiset and then adds the concluded clauses back to the multiset. Once we produce at least one empty clause, the refutation is complete. The *size* and *width* of the refutation are defined as before.

Nullstellensatz (F-NS). Let \mathbb{F} be a field. An \mathbb{F} -Nullstellensatz refutation of a set of polynomial equations $\{a_i(x) = 0 : i \in [m]\}$ over \mathbb{F} is given by a set of polynomials $\{p_i(x)\} \subseteq \mathbb{F}[x]$ such that

$$\sum_{i \in [m]} p_i(x) \cdot a_i(x) = 1. \quad (2)$$

The *size* of the refutation is the combined total number of monomials in p_i and a_i and its *degree* is the maximum of $\deg(p_i) + \deg(a_i)$ over all i .

Reversible Resolution is p -simulated by uSA. Indeed, the usual simulations of Resolution by SA [DMR09, ALN16] have the neat property that if they are applied to a RevRes proof instead, the resulting coefficients become bounded by the size of the RevRes proof (see also [FMSV23] for a simulation in a closely related MaxSAT system). This also means that RevRes is strictly less powerful than Resolution, as per our first separation result.

It is a classic result that Resolution is not p -simulated by \mathbb{F} -NS over any field \mathbb{F} . This is witnessed by CNF formulas expressing the *sink-of-dag* (SOD) principle [CEI96, Bus98] or the *pebbling* principle [BCIP02, dRNMR19]. Our second result strengthens these classical separations showing that RevRes cannot be simulated by low-degree \mathbb{F} -NS.

Theorem 2. *There are n -variate CNF formulas F that can be refuted by constant-width polynomial-size RevRes, but such that any \mathbb{F} -NS refutation (over any \mathbb{F}) of F requires degree $n^{\Omega(1)}$.*

Again, we note that standard lifting techniques can be used to strengthen the degree lower bound in Theorem 2 to an exponential size lower bound. We conclude that RevRes is not p -simulated by \mathbb{F} -NS. In particular, this strengthens a previous result by Filmus et. al. [FMSV23] who showed that RevRes (actually, their closely related MaxSAT system) is not p -simulated by tree-like Resolution.

1.3 Techniques

Our separation between Resolution and uSA (Section 5) builds on the separation between RevRes and \mathbb{F} -NS (Section 4). We prove the latter separation for $\mathbb{F} = \mathbb{R}$ in a particularly robust form, namely we show that it holds even if we allow some small amount of “error” in the NS proof. We introduce what we call ϵ -approximate Nullstellensatz (ϵ -NS) refutations where we relax the polynomial identity (2) over $\mathbb{F} = \mathbb{R}$ to hold only approximately:

$$\sum_{i \in [m]} p_i(x) \cdot a_i(x) = 1 \pm \epsilon, \quad \forall x \in \{0, 1\}^n. \quad (3)$$

In the above expression and for the remainder of the article, “ $= 1 \pm \epsilon$ ” stands for “ $\in [1 - \epsilon, 1 + \epsilon]$ ”, meaning that the LHS is a polynomial that takes values in $[1 - \epsilon, 1 + \epsilon]$ when evaluated on boolean

inputs. For example, an SA refutation where $J(x) \leq \epsilon$ for all boolean inputs x is also an ϵ -NS refutation (since $J(x) \geq 0$ trivially holds).

We show that there is no low-degree approximate NS proof for the formulas that encode the so called *sink-of-potential-line* (SOPL) principle. These formulas are easy for RevRes, and in fact, we later show they are *complete* for RevRes (see [Theorem 3](#)). Naturally, our lower-bound proof borrows techniques from polynomial approximation theory. We give a randomised *decision-to-search reduction*, showing how a low-degree ϵ -NS refutation of SOPL would imply a low-degree approximating polynomial for the OR function. It is well-known, however, that the n -bit OR requires large approximate polynomial degree, namely $\Omega(\sqrt{n})$. This proof idea is inspired by previous works [[RW92](#), [HN12](#), [GP18](#), [IR21](#)] that followed a similar strategy in the context of communication complexity: they studied randomised reductions from set-disjointness (communication analogue of OR) to various communication search problems. Finally, we also give a separate (non-robust) proof that SOPL is hard for \mathbb{F} -NS over any field \mathbb{F} using the intersection theorem ([Theorem 6](#)).

Our lower bound for ϵ -NS, say with $\epsilon := 1/2$, now helps us prove [Theorem 1](#). We consider an SA refutation (1) of the SOD principle (which is a stronger principle than SOPL). The non-existence of a low-degree ϵ -NS refutation for SOPL immediately implies that in any SA refutation of SOD, the conical junta J has to assume a value at least ϵ on some input: the RHS equals $1 + J(x) \geq 1 + \epsilon = 1.5$ for some x . Our idea is to now *iterate* the ϵ -NS lower-bound argument by combining several SOPL instances inside SOD with the aim of finding large values on the RHS. After i iterations, we show the RHS equals $1 + J(x_i) \geq 1.5^{\Omega(i)}$ for some carefully constructed input x_i that embeds i copies of SOPL. Setting $i = \text{poly}(n)$ concludes the proof.

2 Separations in TFNP

A major motivation for our proof complexity separations in [Section 1](#) is that they have consequences in terms of black-box separations between subclasses of TFNP. Together with prior work, our new separations resolve all the black-box relationships between classes depicted in [Figure 2](#). To explain this connection in detail, we start with a short introduction to TFNP.

2.1 Introduction to TFNP

The class TFNP consists of all *total NP search problems*, that is, search problems where a solution is guaranteed to exist, and where it can be efficiently checked whether a given candidate solution is feasible. Some very important problems lie in TFNP, for example, FACTORING (given a number, compute a prime factor) or NASH (given a bimatrix game, compute a Nash equilibrium).

A crucial observation is that no TFNP problem can be NP-hard, unless $\text{NP} = \text{coNP}$ [[MP91](#)]. Furthermore, it is believed that TFNP is unlikely to have complete problems [[Pud15](#)]. As a result, in order to understand the complexity of important TFNP problems, researchers have defined syntactic subclasses of TFNP, such as PLS [[JPY88](#)], PPAD, PPADS, PPA, PPP [[Pap94](#)]. These subclasses are defined using canonical complete problems that correspond to very simple existence principles.

PLS: Every directed acyclic graph has a sink.

PPAD: Every directed graph with an unbalanced node (outdegree \neq indegree) must have another unbalanced node.

PPADS: Every directed graph with a positively unbalanced node (outdegree $>$ indegree) must have a negatively unbalanced node (outdegree $<$ indegree).

PPA: Every undirected graph with an odd-degree node must have another odd-degree node.

PPP: Every function mapping $[n + 1]$ to $[n]$ must have a collision. (*Pigeonhole Principle*)

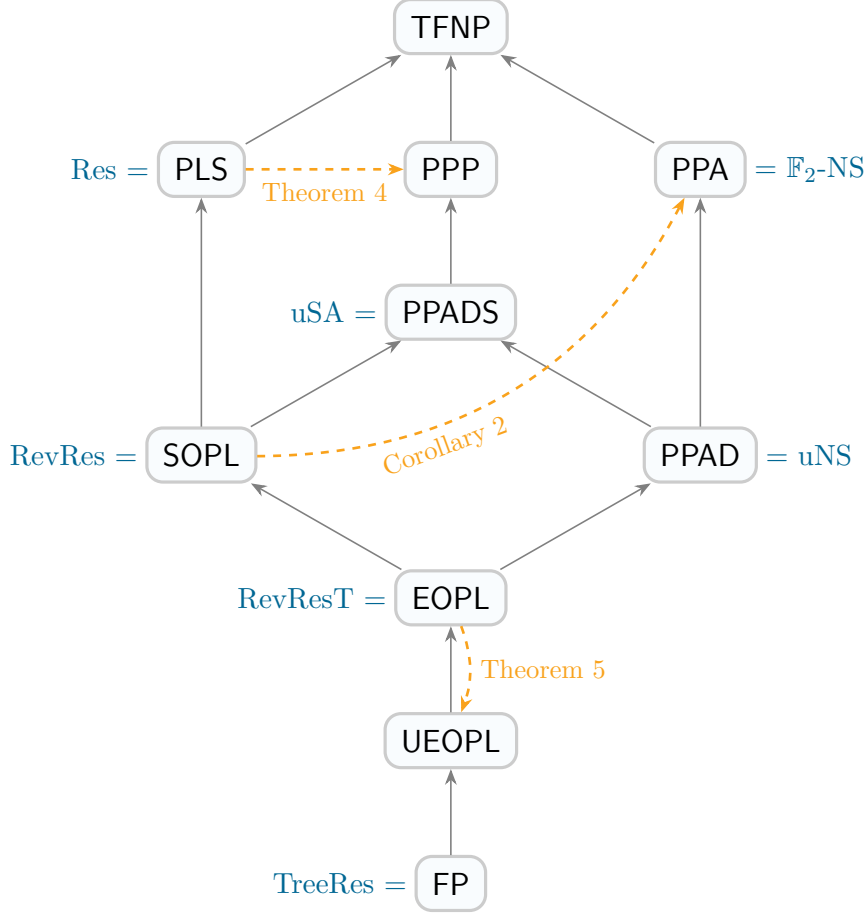


Figure 2: Class inclusion diagram for TFNP. An arrow $A \rightarrow B$ means $A \subseteq B$ relative to all oracles. A dashed arrow $A \dashrightarrow B$ means $A \not\subseteq B$ relative to some oracle. We have only drawn new separations proved in this paper. Together with prior oracle separations [BCE⁺98, Mor01, BM04], this resolves all black-box relationships between the classes featured in the diagram. In the black-box model, some classes can be captured using propositional proof systems, as indicated in blue.

These existence principles naturally give rise to corresponding total search problems. For example, for PPAD that would be: given a directed graph and an unbalanced node in that graph, find another unbalanced node. These problems are defined so that the search space (the set of nodes) has size exponential in the size of the input. Otherwise, it would be trivial to find a solution in polynomial time. In more detail, this is achieved by having the input of the problem consist of a boolean circuit that can be used to compute the neighbours of any given node.

The theory of TFNP classes has been successful in capturing the complexity of many important natural problems. Indeed, in a celebrated result [DGP09, CDT09], it was shown that NASH is complete for PPAD. Following this breakthrough, other problems from game theory [DQS12, CDO15, Meh18] and economics [CSVY08, CDDT09, CPY17] were also proved PPAD-complete. Similarly, PLS has been found to capture the complexity of various interesting problems, mainly ones where a local optimum of some sort is sought [Kre89, Kre90, Sch91, FPT04]. Finally, various problems in fair division are PPA-complete [FG22], while some problems related to cryptography have been shown PPP-complete [SZZ18].

New classes and collapses

More recently, newer classes CLS [DP11], EOPL [FGMS20, HY20], SOPL [GKRS18] were defined, motivated chiefly by problems that were unlikely to be complete for any of the classical classes discussed above. Indeed, it was noted that many interesting problems lie in both PLS and PPAD, but are unlikely to be complete for $\text{PLS} \cap \text{PPAD}$, a seemingly completely artificial class. To remedy this situation, CLS, and later EOPL, were defined as more natural subclasses of $\text{PLS} \cap \text{PPAD}$. However, in a surprising turn of events, it was discovered that $\text{CLS} = \text{PLS} \cap \text{PPAD}$ [FGHS21] and also that $\text{EOPL} = \text{PLS} \cap \text{PPAD}$ and $\text{SOPL} = \text{PLS} \cap \text{PPADS}$ [GHJ⁺22]. In other words, the new classes can be completely defined in terms of the classical ones.

In order to rule out further surprising collapses in the future, it would thus make sense, whenever one defines a new subclass, to also provide some kind of evidence that the new class is indeed new, and does not collapse to existing classes. Clearly, any unconditional separation is completely out of reach, since it would immediately imply that $\text{P} \neq \text{NP}$. However, it turns out that one can indeed prove separations relative to oracles by proving unconditional separations between black-box versions of the classes.

The black-box model

Recall that TFNP subclasses are defined in terms of very simple existence principles that are turned into (white-box) total search problems by having the input be implicitly described by a boolean circuit. Another—sometimes more natural—choice is to have the input be described by a black box, instead of a white box. For example, in the case of PPAD, instead of being given the description of a circuit that can be used to compute neighbours, we can consider the model where we can query an oracle (black-box) to ask for the neighbours of a node.

More formally, a total *query* 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 inputs $x \in \{0, 1\}^n$ there is an output $o \in O_n$ 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 think of an instance $x \in \{0, 1\}^n$ as a very long bitstring that can only be accessed through queries to individual bits. In this context, an efficient algorithm is a deterministic algorithm that, for any $x \in \{0, 1\}^n$, finds a solution o to x by performing a small number of queries to x , namely at most $\text{poly}(\log n)$ queries. Thus, efficient algorithms correspond to decision trees (with leaves labelled by elements of O_n) of depth at most $\text{poly}(\log n)$. Note that this model is non-uniform: the problem admits an efficient algorithm, if for each $n \in \mathbb{N}$, there exists a shallow decision tree solving R_n .

The notion of total search problems as defined above does not quite correspond to TFNP yet, because it is missing the requirement for efficient verification of solutions. We enforce this in the following natural way. A total search problem $R = (R_n)_n$ is in TFNP^{dt} , if for each $o \in O_n$ there is a decision tree T_o with depth $\text{poly}(\log n)$ such that for every $x \in \{0, 1\}^n$, $T_o(x) = 1$ if and only if $(x, o) \in R_n$. We define the class PPAD^{dt} as the set of all TFNP^{dt} problems that have an efficient decision-tree reduction to (the query version of) the canonical complete problem for PPAD. We denote by $\text{PPAD}^{dt}(R_n)$ the decision tree complexity of a reduction from R_n to the canonical PPAD^{dt} -complete problem (see Section 3 for a precise definition). Thus, problem $R = (R_n)_n$ lies in PPAD^{dt} if and only if $\text{PPAD}^{dt}(R_n) = \text{poly}(\log n)$. The decision-tree analogues of the other classes are defined in the same way.

Black-box separations

In the black-box model, it is now possible to prove *unconditional* separations, e.g., that $\text{PPAD}^{dt} \not\subseteq \text{PLS}^{dt}$ by showing that there is no shallow decision-tree reduction from some problem in PPAD^{dt} to

a complete problem for PLS^{dt} . Importantly, a black-box separation also provides some evidence that the separation might hold in the white-box setting too, in the following sense: *any black-box separation implies a corresponding separation in the white-box model relative to some oracle* [BCE⁺98]. Moreover, all existing containment results (including the recent collapses [FGHS21, GHJ⁺22]) also hold in the black-box setting. Thus, a black-box separation is quite significant, since it rules out any collapse using existing techniques.

Previously, Beame et al. [BCE⁺98] proved all possible separations between the classes PPA^{dt} , PPAD^{dt} , PPADS^{dt} , PPP^{dt} . Subsequently, Morioka [Mor01] extended these results by proving that PPAD^{dt} is not reducible to PLS^{dt} . This implies that none of PPA^{dt} , PPAD^{dt} , PPADS^{dt} and PPP^{dt} are contained in PLS^{dt} . Buresh-Oppenheim and Morioka [BM04] further proved that PLS^{dt} is not contained in PPA^{dt} . It has so far remained open whether $\text{PLS}^{dt} \subseteq \text{PPADS}^{dt}$ or $\text{PLS}^{dt} \subseteq \text{PPP}^{dt}$.

Connection to proof complexity

Propositional proof complexity is a major tool for proving black-box separations. There is a natural correspondence between total query search problems and CNF contradictions. In one direction, a CNF contradiction $F := C_1 \wedge \dots \wedge C_m$ over the variables $x = (x_1, \dots, x_n)$ naturally gives rise to a corresponding total search problem $S(F)$: given an assignment $x \in \{0, 1\}^n$, find an unsatisfied clause of F . Formally, we define $S(F) \subseteq \{0, 1\}^n \times [m]$ by $(x, i) \in S(F)$ if and only if $C_i(x) = 0$. Thus, a sequence of unsatisfiable CNF formulas $F = (F_n)$, where F_n has n variables, defines the total search problem $S(F) = (S(F_n))$. Note that $S(F) \in \text{TFNP}^{dt}$ if F_n has width $\text{poly}(\log n)$.

In the other direction, a problem $R = (R_n)$ in TFNP^{dt} can be written equivalently as $S(F)$ for some sequence of CNF contradictions $F = (F_n)$. Specifically, for $R_n \subseteq \{0, 1\}^n \times O_n$ we define the formula $F_n := \bigwedge_{o \in O_n} \neg T_o(x)$, where we note that $T_o(x)$ can naturally be written as a DNF formula of width at most $\text{poly}(\log n)$ (with one term per accepting leaf of T_o), and thus $\neg T_o(x)$ can be written as a CNF formula of the same width.

2.2 New characterisations

The above connection to proof complexity opens up the possibility to characterise search problem classes by propositional proof systems, in the following sense: *the problem $(S(F_n))_n$ lies in class X if and only if the CNF formulas $(F_n)_n$ have small refutations in proof system Y* . To make this more precise, for any proof system P and a CNF formula F , we define

$$P(F) := \min_{P\text{-proof } \Pi \text{ of } F} [\log \text{size}(\Pi) + \deg(\Pi)].$$

Here, $\deg(\Pi)$ should be understood as *width* when P is Resolution (or RevRes) and as *depth* when P is tree-like Resolution. Prior work has established the following characterisations.

- $\text{FP}^{dt}(S(F)) = \Theta(\text{TreeRes}(F))$ [LNNW95].
- $\text{PLS}^{dt}(S(F)) = \Theta(\text{Res}(F))$ [BKT14].
- $\text{PPA}^{dt}(S(F)) = \Theta(\mathbb{F}_2\text{-NS}(F))$ [GKRS18].
- $\text{PPA}_p^{dt}(S(F)) = \Theta(\mathbb{F}_p\text{-NS}(F))$ for every prime p [Kam20].

We contribute the following new characterisations. For one of them, we need to introduce one more proof system, *Reversible Resolution with Terminals* (RevResT), defined in Section 6.3.

Theorem 3. *For any unsatisfiable CNF formula F , we have:*

- $\text{PPAD}^{dt}(S(F)) = \Theta(\text{uNS}(F))$.

- $\text{PPADS}^{dt}(S(F)) = \Theta(\text{uSA}(F))$.
- $\text{SOPL}^{dt}(S(F)) = \Theta(\text{RevRes}(F))$.
- $\text{EOPL}^{dt}(S(F)) = \Theta(\text{RevResT}(F))$.

Together with our proof complexity separations from [Section 1](#), we immediately obtain the following black-box separations (which yield white-box oracle separations as discussed above).

Corollary 1. $\text{PLS}^{dt} \not\subseteq \text{PPADS}^{dt}$.

Corollary 2. $\text{SOPL}^{dt} \not\subseteq \text{PPA}^{dt}$.

Additional characterizations, as well as separation results, were obtained in the subsequent works [\[HKT24, LPR24\]](#).

2.3 Two further separations

We show two more black-box separations involving classes PPP^{dt} and UEOPL^{dt} , which currently lack elegant proof system characterisations. The first separation strengthens [Corollary 1](#).

Theorem 4. $\text{PLS}^{dt} \not\subseteq \text{PPP}^{dt}$.

Theorem 5. $\text{EOPL}^{dt} \not\subseteq \text{UEOPL}^{dt}$.

(An early preprint of this work did not include the above theorems. In an independent work, Bonacina and Thapen [\[BT22\]](#) also proved [Theorem 4](#), deriving it from [Corollary 1](#) using essentially the same proof as we do.)

[Theorem 4](#) settles the last open oracle separation question between the five original TFNP classes introduced in [\[JPY88, Pap94\]](#). This question was re-asked recently by Daskalakis in his Nevanlinna Prize lecture [\[Das19, Open Question 6\]](#). Previously, Buresh-Oppenheim and Morioka [\[BM04\]](#) showed a partial result in the direction of [Theorem 4](#), namely, that there is no reduction from PLS^{dt} to PPP^{dt} that preserves the number of solutions in each instance. Finally, [Theorem 5](#) answers a question of [\[FGMS20\]](#) who introduced the class UEOPL . They conjectured that $\text{EOPL} \not\subseteq \text{UEOPL}$ and asked whether this could be shown relative to an oracle.

2.4 Intersection theorems in proof complexity

Our new characterisations can be combined with the collapses $\text{SOPL} = \text{PLS} \cap \text{PPADS}$ and $\text{EOPL} = \text{PLS} \cap \text{PPAD}$ [\[GHJ⁺22\]](#) (which hold in the black-box model) to produce completely new types of results in propositional proof complexity that we call *intersection theorems*.

Stated plainly, the first of these results says that a CNF formula F admits an efficient (small degree and size) Reversible Resolution refutation *if and only if* if it admits an efficient Resolution refutation *and* an efficient unary Sherali–Adams refutation. In other words, Reversible Resolution is the “intersection” of Resolution and unary Sherali–Adams. We can similarly show that Reversible Resolution with Terminals is the “intersection” of Resolution and unary Nullstellensatz.

Theorem 6. *For any unsatisfiable CNF formula F , we have:*

- $\text{RevRes}(F) = \Theta(\text{Res}(F) + \text{uSA}(F))$.
- $\text{RevResT}(F) = \Theta(\text{Res}(F) + \text{uNS}(F))$.

To our knowledge, these are the first theorems of their type, that is, showing that efficient proofs exist in one system P_0 *if and only if* efficient proofs exist in two other systems P_1 and P_2 . This is all the more striking given that all three of these proof systems are quite natural, being motivated from boolean logic and SAT-solving (Res), linear programming (uSA), and MaxSAT solving (RevRes). Moreover, the proof of this theorem (Section 7) crucially uses *both* perspectives of proof systems and total search problems. Starting with propositional proofs in Resolution and unary Sherali–Adams, we convert them to efficient formulations of $S(F)$ in PLS^{dt} and $PPADS^{dt}$, respectively. We then apply the collapse theorem to argue there is an efficient formulation of $S(F)$ in $SOPL^{dt}$, which we can finally convert back to a RevRes proof. We see no apparent way to prove this theorem directly using classic proof complexity techniques.

2.5 Open problems

In our opinion, exploring the interplay between TFNP and propositional proof complexity holds untapped potential. The results in this work arose from our core belief that *a natural concept introduced in one theory should have a natural counterpart in another theory*. This philosophy suggests many further directions for research and serves as a guiding principle for formulating new beautiful connections between the two theories. For example:

- (1) Can Theorem 1 be strengthened to show that the Sum-of-Squares system needs huge coefficients to simulate Resolution in low degree?
- (2) Can we characterise the class PPP by a proof system?
- (3) Does unary-NS p -simulate \mathbb{Z} -NS for refuting CNF formulas?
- (4) Can we prove other intersection theorems in propositional proof complexity?
- (5) Do Sum-of-Squares and Polynomial Calculus characterise some TFNP classes?
- (6) Are there communication complexity analogues of our results? The recent column [dRGR22] surveys the connections between total search problems and characterisations of various circuit models in the language of communication complexity (via Karchmer–Wigderson games).

We note here that Buss, Fleming and Impagliazzo [BFI22] have recently provided an answer to question (5) by giving a TFNP characterization of Polynomial Calculus. In fact, they show a more general connection: every well-behaved proof system which can prove its own soundness is characterized by a TFNP problem, and vice-versa. This also answers question (2), although ideally we would like to characterize PPP by a more natural proof system than the one obtained through this generic connection.

3 Definitions

In this section we give formal definitions of the total search problems that we consider in this work. We emphasise that unlike the standard uniform setting of TFNP, we will be interested in non-uniform variants of TFNP classes defined by decision trees.

3.1 Decision tree TFNP

Definition 1. A *total (query) search problem* is a sequence of relations $R = \{R_n \subseteq \{0, 1\}^n \times O_n\}$, 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 $\text{poly}(\log n)$ such that for every $x \in \{0, 1\}^n$, $T_o(x) = 1$ iff $(x, o) \in R$.

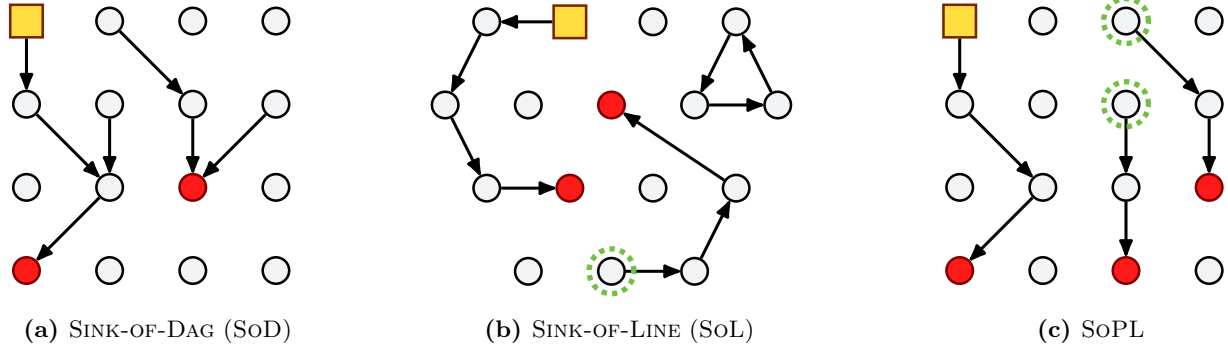


Figure 3: Examples of total search problems. The distinguished source node is drawn as a yellow square. Red nodes are associated with solutions. (For visual clarity, we highlight the actual sink nodes for SoD rather than their predecessors.) Nodes circled in green would be solutions for EoL and EoPL, respectively.

While total search problems are formally defined as sequences $R = (R_n)$, it will often make sense to speak of an individual search problem R_n in the sequence. We will therefore slightly abuse notation and also call R_n a total search problem. It will also be convenient to encode total search problems with inputs and outputs chosen from domains other than $\{0, 1\}^n$. One common example will be total search problems where the inputs are chosen from $[n]^n$. We can simulate this simply by encoding all elements of the non-boolean domain in binary in the usual way. In all examples in this paper, performing this encoding will change the complexities of the involved problems by no more than a $O(\log n)$ factor. We also allow the n -th problem R_n in a sequence to have $\text{poly}(n)$ input bits (instead of n) for notational convenience.

The canonical examples of total search problems in TFNP^{dt} are the search problems associated with an unsatisfiable CNF formula F .

Definition 2. For any unsatisfiable CNF formula $F := C_1 \wedge \dots \wedge C_m$ over n variables, define $S(F) \subseteq \{0, 1\}^n \times [m]$ by $(x, i) \in S(F)$ if and only if $C_i(x) = 0$.

Therefore, given any sequence of unsatisfiable CNF formulas $F = \{F_1, F_2, \dots\}$ we get a total search problem $S(F) = \{S(F_1), S(F_2), \dots\}$ in the natural way. Observe that $S(F) \in \text{TFNP}^{dt}$ if each unsatisfiable CNF formula has width $\text{poly}(\log n)$. Conversely, these examples are also *complete*, in the sense that any search problem in TFNP^{dt} can be re-encoded as unsatisfiable CNF formulas.

Definition 3. 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 $F := \bigwedge_{o \in O} \neg T_o(x)$ where we think of $\neg T_o(x)$ written as a CNF formula (of width determined by the decision tree depth of T_o).

3.2 Search problem zoo

We now define several search problems that will be of interest to us. See also [Figure 3](#) for helpful illustrations of some of them. We start with the problems that are complete for the classical classes introduced in [[JPY88](#), [Pap94](#)].

PPP: PIGEON (PIGEON $_n$). This problem features n pigeons, denoted by $[n]$, and as input we are given, for each pigeon $u \in [n]$ a hole $s_u \in [n - 1]$. The goal is to output

1. $u, v \in [n]$, if $u \neq v$ and $s_u = s_v$. (*pigeon collision*)

PPADS: SINK-OF-LINE (SoL_n). This problem is defined on a set of n nodes, denoted by $[n]$, where the node 1 is “distinguished”. For input, we are given a successor $s_u \in [n]$ for each node $u \in [n]$ and a predecessor $p_u \in [n]$ for each node $u \neq 1$. Given this list of successor/predecessor pointers we create a directed graph G where we add an edge (u, v) if and only if $s_u = v$ and $p_v = u$. We say u is a *proper sink* if it has in-degree 1 and out-degree 0, and it is a *proper source* if it has in-degree 0 and out-degree 1. The goal of the search problem is to output any of the following

1. 1, if 1 is not a proper source node in G , or *(no distinguished source)*
2. $i \neq 1$, if i is a proper sink node in G . *(proper sink)*

PPAD: END-OF-LINE (EoL_n). Same as SoL, except we add the following feasible solution.

3. $i \neq 1$, if i is a proper source node in G . *(proper source)*

PLS: SINK-OF-DAG (SoD_n). This problem is defined on the $[n] \times [n]$ grid, where the node $(1, 1)$ is “distinguished”. As input, for each grid node $u = (i, j) \in [n] \times [n]$, we are given a *successor* $s_u \in [n] \cup \{\text{null}\}$, interpreted as naming a node $(i + 1, s_u)$ on the next row. We say a node u is *active* if $s_u \neq \text{null}$, otherwise it is *inactive*. A node u is a *proper sink* if u is inactive but some active node has u as a successor. The goal of the search problem is to output any of the following

1. $(1, 1)$, if $(1, 1)$ is inactive *(inactive distinguished source)*
2. (n, j) , if (n, j) is active, *(active sink)*
3. (i, j) for $i \leq n - 1$, if (i, j) is active and its successor is a proper sink. *(proper sink)*

For SoD, it is helpful to think of the successors s_u as describing a fan-out 1 dag on an $n \times n$ grid of nodes such that all edges are between adjacent rows. Active nodes are those nodes which have some edge leaving them. Then, if we require that $(1, 1)$ is active and all nodes on row n are inactive, the goal is to find a *proper sink*, that is, an active node with an inactive successor node.

We next define complete problems for the more modern classes introduced in [HY20, FGMS20, GKRS18]. They are variations of the SoD problem where all nodes in the grid have predecessor pointers and we only add an edge if the successor and predecessor pointers agree. In particular, this implies that every node has fan-out *and* fan-in 1.

SOPL: SINK-OF-POTENTIAL-LINE (SoPL_n). As input we are given a *successor* $s_u \in [n] \cup \{\text{null}\}$ for each $u \in [n] \times [n]$ and a *predecessor* $p_u \in [n] \cup \{\text{null}\}$ for each $u \in \{2, \dots, n\} \times [n]$. A node $(i, j) \in [n - 1] \times [n]$ is *active* if $s_{(i,j)} = k \neq \text{null}$ and $p_{(i+1,k)} = j$, otherwise it is *inactive*; a node $(i, j) \in \{n\} \times [n]$ is active if $s_{(i,j)} \neq \text{null}$ and inactive otherwise. A node u is a *proper sink* if u is inactive but some active node has u as a successor. The goal is to output any of the following

1. $(1, 1)$, if $(1, 1)$ is inactive, *(inactive distinguished source)*
2. (n, j) , if (n, j) is active, *(active sink)*
3. (i, j) , if (i, j) is a proper sink. *(proper sink)*

EOPL: END-OF-POTENTIAL-LINE (EoPL_n). Add the following feasible solutions to SoPL. A node (i, j) is a *proper source* if (i, j) is active and, either, $i = 1$ or $1 < i < n$ and there is no active node with (i, j) as a successor.

4. (i, j) , if $(i, j) \neq (1, 1)$ and (i, j) is a proper source. *(proper source)*

UEOPL: UNIQUE-EoPL (UEoPL_n). Add the following feasible solution to EoPL.

5. (i, j) and (i, j') , if $j \neq j'$ and both nodes are active. *(two parallel lines)*

3.3 Reductions and formulations

Given any problem defined above we can consider complexity classes of total search problems obtained by taking reductions to these problems. In this work we are particularly interested in the case where the reduction is defined by a low-depth decision tree.

Definition 4. 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 $f_i: \{0, 1\}^n \rightarrow \{0, 1\}$ and $g_o: \{0, 1\}^n \rightarrow O$ 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

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

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 $S_m^{dt}(R)$ over all m . If $R = (R_n)$ is also a sequence, then we denote by $S^{dt}(R)$ 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 $R = (R_n), S = (S_n)$, we write

$$S^{dt} := \{R : S^{dt}(R) = \text{poly}(\log n)\}.$$

We can now define the decision-tree variants of the standard classes: $\text{PPP}^{dt} = \text{PIGEON}^{dt}$, $\text{PPADS}^{dt} = \text{SOL}^{dt}$, and so on, according to the problems defined in [Section 3.2](#).

4 Reversible Resolution vs. Nullstellensatz

In this section we prove [Theorem 2](#), restated below.

Theorem 2. *There are n -variate CNF formulas F that can be refuted by constant-width polynomial-size RevRes , but such that any \mathbb{F} -NS refutation (over any \mathbb{F}) of F requires degree $n^{\Omega(1)}$.*

We prove [Theorem 2](#) in two ways. First ([Sections 4.1 to 4.3](#)), we give a particularly robust proof in the special case $\mathbb{F} = \mathbb{R}$, which will be useful in [Section 5](#) when we prove our other separation result. Second ([Section 4.4](#)), we give a (non-robust) proof for all \mathbb{F} using the intersection theorem. In both proofs we consider the SoPL principle and show that it does not admit a low-degree NS proof, and that it can be refuted in low-width small-size RevRes .

4.1 Approximate Nullstellensatz

We define a generalisation of \mathbb{R} -NS that we call ϵ -approximate Nullstellensatz (ϵ -NS) where $\epsilon \in (0, 1)$ is an error parameter. An ϵ -NS refutation of a set of real polynomial equations $\{a_i(x) = 0 : i \in [m]\}$ is a set of polynomials $\{p_i(x)\}$ such that

$$\sum_{i \in [m]} p_i(x) \cdot a_i(x) = 1 \pm \epsilon, \quad \forall x \in \{0, 1\}^n \tag{4}$$

where we recall that “ $= 1 \pm \epsilon$ ” stands for “ $\in [1 - \epsilon, 1 + \epsilon]$ ”, meaning that the LHS is a polynomial that takes values in $[1 - \epsilon, 1 + \epsilon]$ when evaluated on boolean inputs. The ϵ -NS system is not a standard proof system in the sense of Cook and Reckhow [CR79]. In particular, it is not hard to show (using the PCP theorem) that testing the condition in (4) is in fact **coNP**-complete. Another feature of the new system is that the error parameter can be efficiently reduced using standard error reduction techniques for polynomial approximation. For example, if we compose any ϵ -NS proof $\sum_i p_i a_i = 1 \pm \epsilon$ with the univariate polynomial $q(z) := z(2 - z)$, we obtain an ϵ^2 -NS proof $q(\sum_i p_i a_i) = 1 \pm \epsilon^2$.

4.2 Lower bound for ϵ -NS

Recall that the input to SOPL_n consists of successor pointers $s_u \in [n] \cup \{\text{null}\}$ and predecessor pointers $p_u \in [n] \cup \{\text{null}\}$ for each grid node $u \in [n] \times [n]$. For the purposes of NS, we encode this input in binary by a string $y \in \{0, 1\}^{n'}$ over $n' = O(n^2 \log n)$ variables. Moreover, we can think of SOPL_n as an unsatisfiable set of polynomial equations $\{a_i(y) = 0\}$ each of degree $O(\log n)$. These equations can be obtained by taking the unsatisfiable CNF encoding of SOPL_n (Definition 7) and encoding each clause as the corresponding polynomial equation in the usual way.

Our goal is to prove the following lemma.

Lemma 1. *Every $\frac{1}{2}$ -NS refutation of SOPL_n requires degree $n^{\Omega(1)}$.*

It suffices to prove the lemma for error $\epsilon := 0.01$, because of efficient error reduction. Fix any ϵ -NS refutation $\sum_i p_i(y) a_i(y) = 1 \pm \epsilon$ of degree k for SOPL_n . Our goal is to show a lower bound on k . We will give a randomised *decision-to-search reduction*, in the style of [RW92, HN12, GP18, IR21], showing that a low-degree ϵ -NS refutation would imply a low-degree approximating polynomial for the $(n - 1)$ -bit OR function. The following well-known fact then concludes the proof.

Fact 1 ([NS94]). Suppose that p is an n -variate real polynomial such that $p(x) = \text{OR}_n(x) \pm 1/3$ for all $x \in \{0, 1\}^n$. Then $\deg(p) \geq \Omega(\sqrt{n})$.

Definition of reduction. We define a depth- d deterministic reduction as a pair (f, u) such that

- (1) $f: \{0, 1\}^{n-1} \rightarrow \{0, 1\}^{n'}$ is a function that maps an input x of OR_{n-1} to an input $y = f(x)$ of SOPL_n . Moreover, each output bit $f_i(x) \in \{0, 1\}$ is a depth- d decision tree function of x .
- (2) For any input x , the only solutions of $y = f(x)$ are active sinks on the last row $\{n\} \times [n]$. We write $\text{Sol}(y) \subseteq \{n\} \times [n]$ for the set of solutions in y . Moreover, $u \in \text{Sol}(y)$ is a solution called the *planted* solution. (Note that u does not depend on x .)
- (3) If $\text{OR}(x) = 0$, then $y = f(x)$ contains a unique solution, namely $\text{Sol}(y) = \{u\}$.
- (4) If $\text{OR}(x) = 1$, then $y = f(x)$ contains at least two solutions, $|\text{Sol}(y)| \geq 2$.

We then define a depth- d *randomised* reduction \mathcal{R} as a probability distribution over depth- d deterministic reductions $(\mathbf{f}, \mathbf{u}) \sim \mathcal{R}$. For every x , we write \mathcal{R}_x for the distribution of $(\mathbf{f}(x), \mathbf{u}) = (\mathbf{y}, \mathbf{u})$. We say that a pair (\mathbf{y}, \mathbf{u}) is *ideal* if it satisfies the following.

Ideal (\mathbf{y}, \mathbf{u}) : Let y be any outcome of \mathbf{y} and consider \mathbf{u} conditioned on $\mathbf{y} = y$, namely, $\mathbf{u}' := (\mathbf{u} \mid \mathbf{y} = y)$. Then \mathbf{u}' is uniformly distributed over $\text{Sol}(y)$; in short, $\mathbf{u}' \sim \text{Sol}(y)$.

We say \mathcal{R} is *ideal* if \mathcal{R}_x is ideal for every x .

Ideal reduction \Rightarrow Approximation to OR. Next, we show that if we had an ideal reduction, we could construct an approximating polynomial for OR. We write i_u for the unique i such that the polynomial equation $a_i(y) = 0$ encodes the SOPL_n constraint that u is not an active sink. Namely, this corresponds to the equation $s_u = 0$, where the bit $s_u \in \{0, 1\}$ of the input y encodes whether or not u is active (see [Definition 7](#)). If we think of $u \in \{n\} \times [n]$ as encoded by an $O(\log n)$ -bit string, we can define an $[n' + O(\log n)]$ -variate polynomial

$$q(y, u) := p_{i_u}(y)a_{i_u}(y) = \sum_i \mathbf{1}[i = i_u] p_i(y) a_i(y). \quad (5)$$

Here, for every i , the indicator function $\mathbf{1}[i = i_u] \in \{0, 1\}$ is computed by an $O(\log n)$ -degree polynomial. This means q has degree $\deg(q) \leq O(k \log n)$. If (\mathbf{y}, \mathbf{u}) is ideal, then

$$\begin{aligned} \mathbb{E}[q(\mathbf{y}, \mathbf{u})] &= \mathbb{E}_{y \sim \mathbf{y}} [\mathbb{E}_{u' \sim (\mathbf{u} | \mathbf{y} = y)} [p_{i_{u'}}(y) a_{i_{u'}}(y)]] \\ &= \mathbb{E}_{y \sim \mathbf{y}} [\mathbb{E}_{u' \sim \text{Sol}(y)} [p_{i_{u'}}(y) a_{i_{u'}}(y)]] \\ &= \mathbb{E}_{y \sim \mathbf{y}} [|\text{Sol}(y)|^{-1} \sum_{u' \in \text{Sol}(y)} p_{i_{u'}}(y) a_{i_{u'}}(y)] \\ &= \mathbb{E}_{y \sim \mathbf{y}} [|\text{Sol}(y)|^{-1} \sum_i p_i(y) a_i(y)] \\ &= \mathbb{E}_{y \sim \mathbf{y}} [|\text{Sol}(y)|^{-1}] \cdot (1 \pm \epsilon) \\ &= (1 \pm \epsilon) \cdot \mathbb{E}[|\text{Sol}(\mathbf{y})|^{-1}] \end{aligned} \quad (6)$$

where we used the fact that $\sum_{u' \in \text{Sol}(y)} p_{i_{u'}}(y) a_{i_{u'}}(y) = \sum_i p_i(y) a_i(y)$, because $a_i(y) = 0$ for all $i \notin \{i_{u'} : u' \in \text{Sol}(y)\}$, given that y satisfies all the SOPL_n constraints, except the equations requiring that u' not be an active sink, for $u' \in \text{Sol}(y)$.

Suppose for a moment we had an ideal depth- d randomised reduction \mathcal{R} . Then, we could construct the polynomial

$$r(x) := \mathbb{E}_{\mathcal{R}_x}[q(\mathbf{y}, \mathbf{u})] = \sum_{f, u} \Pr_{\mathcal{R}}[(\mathbf{f}, \mathbf{u}) = (f, u)] \cdot q(f(x), u).$$

We have $\deg(r) \leq O(dk \log n)$. Moreover, if $\text{OR}(x) = 0$ then $r(x) = 1 \pm \epsilon$; and if $\text{OR}(x) = 1$ then $r(x) \in [0, (1 + \epsilon)/2]$, since $\mathbb{E}[|\text{Sol}(\mathbf{y})|^{-1}] \in [0, 1/2]$. Thus for $\epsilon = 0.01$, if we consider $t(x) := 1 - r^2(x)$ we get that t approximates OR to within error $1/3$. Using [Fact 1](#), we deduce that $k \geq \Omega(\sqrt{n}/(d \log n))$.

In summary, all that remains is to find an ideal reduction of shallow depth. Unfortunately, we do not know how to design an ideal reduction for SOPL . We instead give a reduction that is *locally indistinguishable* from an ideal one, which will suffice for us.

A locally ideal reduction. Consider the following depth-1 randomised reduction \mathcal{R} ; see [Figure 4](#).

1. Let $y = y(x)$ be the input to SOPL_n that has a directed path running down the first column of nodes, starting at distinguished node $(1, 1)$ and terminating at the active sink $u := (n, 1)$ (say u is made active by being assigned 1 as successor). Moreover, we activate a path in y down column $i \geq 2$ iff $x_{i-1} = 1$. Note that y is a depth-1 decision tree function of x , and u does not depend on x at all.
2. Let $\mathbf{y} = \mathbf{y}(x)$ be obtained from y so that, for each row except the first, $i \in [n] \setminus \{1\}$, randomly permute the nodes $\{i\} \times [n]$ on that row (updating the successor/predecessor pointers). Let \mathbf{u} be the sink node that u is mapped to.
3. Output (\mathbf{f}, \mathbf{u}) where $\mathbf{f}(x) := \mathbf{y}(x)$.

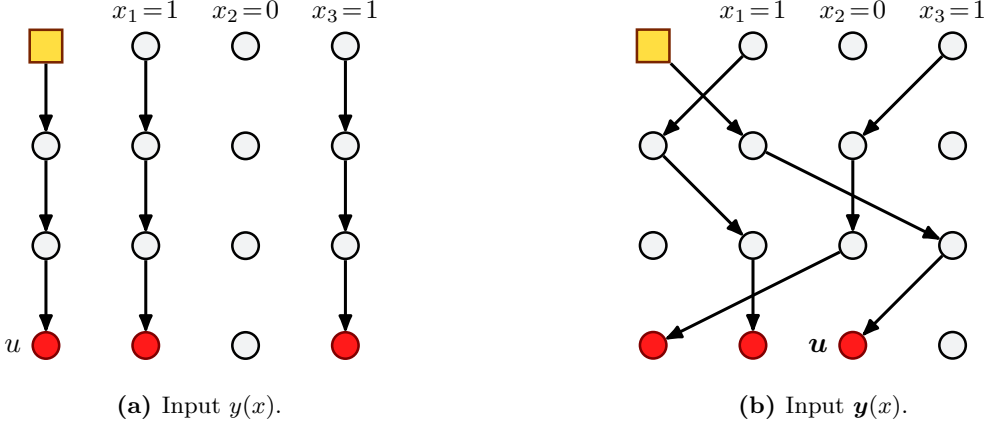


Figure 4: Randomised reduction \mathcal{R} . First, we compute $y(x)$ deterministically from x . This input always contains a path down the left-most column, which terminates at the active sink u (*planted solution*). Moreover, for every i with $x_i = 1$ there is a path down the $(i+1)$ -st column. The number of active sinks is $|\text{Sol}(y)| = 1 + |x|$, where $|x|$ denotes the Hamming weight. In the second step, we randomly permute every row of nodes, except the first one. This yields the random output (\mathbf{y}, \mathbf{u}) of the reduction.

It is easy to check that \mathcal{R} satisfies items (1)–(4) for every outcome of randomness. In particular, we have $|\text{Sol}(\mathbf{y})| = 1 + |x|$. Unfortunately, \mathcal{R} is *not* ideal: \mathbf{u} is always the active sink at the end of the path starting at the distinguished node. What we would really like instead is that \mathcal{R}_x was distributed as the ideal pair $(\mathbf{y}, \mathbf{u}) \sim \mathcal{I}_x$ defined by the following procedure: Sample $(\mathbf{y}, \mathbf{u}') \sim \mathcal{R}_x$; define \mathbf{u} such that for every outcome y , $(\mathbf{u} \mid \mathbf{y} = y) \sim \text{Sol}(y)$; and output (\mathbf{y}, \mathbf{u}) .

Define two functions $\{0, 1\}^{n-1} \rightarrow \mathbb{R}$ by

$$r(x) := \mathbb{E}_{\mathcal{R}_x}[q(\mathbf{y}, \mathbf{u})], \quad (7)$$

$$r'(x) := \mathbb{E}_{\mathcal{I}_x}[q(\mathbf{y}, \mathbf{u})]. \quad (8)$$

We know that r has low degree as a polynomial, $\deg(r) \leq O(k \log n)$, and r' has the ideal output behaviour, $r'(x) = (1 \pm \epsilon) \cdot \mathbb{E}[|\text{Sol}(\mathbf{f}(x))|^{-1}]$ by (6). The following claim shows that, in fact, $r = r'$, and hence we can get the best of both worlds. By the discussion above, we are then able to construct an $O(k \log n)$ -degree approximating polynomial for OR, which concludes the proof of Lemma 1.

Claim 1. *We have $r(x) = r'(x)$ for all $x \in \{0, 1\}^{n-1}$.*

Proof. By linearity of expectation, it suffices to show $\mathbb{E}_{\mathcal{R}_x}[m(\mathbf{y}, \mathbf{u})] = \mathbb{E}_{\mathcal{I}_x}[m(\mathbf{y}, \mathbf{u})]$ for any monomial m of q and every x . Fix a monomial m . We claim that \mathcal{R}_x and \mathcal{I}_x have the same marginal distribution over the variables read by m , which would prove the claim. We may assume that $\deg(m) \leq O(k \log n) \leq o(n)$ because otherwise Lemma 1 is proved. Hence there exist two consecutive rows $i, i+1 \in [n/3, 2n/3]$ such that m does not read any variables associated with either row. Starting with a sample $(\mathbf{y}, \mathbf{u}) \sim \mathcal{R}_x$ we can generate a sample from \mathcal{I}_x as follows: Consider active nodes $A \subseteq \{i\} \times [n]$ and $B \subseteq \{i+1\} \times [n]$ on rows i and $i+1$ in \mathbf{y} and the $|A| = |B| = 1 + |x|$ many directed edges joining them (defined by successor pointers for row i and predecessor pointers for row $i+1$). Reroute these edges by choosing a random bijection $A \rightarrow B$, and denote the resulting input by \mathbf{y}' . Then $(\mathbf{y}', \mathbf{u}) \sim \mathcal{I}_x$. This proves our claim about the marginals, since our modification to the input \mathbf{y} was done outside the variables read by m . \square

4.3 Upper bound for RevRes

Our characterisation of SOPL^{dt} by RevRes in [Section 6.3](#) involves proving that SOPL_n (understood as an $O(\log n)$ -width CNF contradiction) admits an $O(\log n)$ -width polynomial-size RevRes refutation ([Theorem 10](#)). If we want to further optimise this down to a constant-width polynomial-size RevRes refutation, as claimed by [Theorem 2](#), then we can consider instead a *sparse constant-width* variant of SOPL_n . Indeed, the following sparsifying construction is standard and so we only sketch it.

We start by defining a bounded-degree dag G that models the connectivity structure of the $[n] \times [n]$ grid with successor/predecessor pointers. The nodes of G include all the grid nodes $[n] \times [n]$. Moreover, for each $u \in [n-1] \times [n]$ we include in G a *successor tree* S_u that is a full binary tree with n leaves, and has edges directed from the root towards the leaves. Similarly, for each $u \in ([n] \setminus \{1\}) \times [n]$ we include in G a *predecessor tree* P_u whose edges are directed from leaves towards the root. We identify the root nodes of S_u and P_u with u . Moreover, for grid nodes (i, j) and $(i+1, k)$ appearing on consecutive rows, we identify the k -th leaf of $S_{(i,j)}$ and the j -th leaf of $P_{(i+1,k)}$. This completes the description of G . Note that the in/out-degree of every node is at most 2.

We can now define a search problem SOPL_G relative to G . As input, each node u in G gets a successor s_u and a predecessor p_u picked from $\{0, 1\} \cup \{\text{null}\}$. For example, $s_u = 0$ ($s_u = 1$) means that u 's successor is the left (right) child of u in G . The constraints of SOPL_G can now be written in constant width. The RevRes upper bound in [Theorem 10](#) can be adapted to yield a constant-width polynomial-size refutation of SOPL_G . Moreover, the original grid version SOPL_n can be reduced to the graph version SOPL_G using an $O(\log n)$ -depth decision tree reduction; see, for example, [\[FGGR22, §4.2\]](#) for details (but for SoD instead of SOPL). The existence of this reduction implies that SOPL_G needs large ϵ -NS degree, because we showed that SOPL_n does.

This concludes the proof of [Theorem 2](#) in case $\mathbb{F} = \mathbb{R}$.

4.4 Lower bound for \mathbb{F} -NS

We now prove the lower bound in [Theorem 2](#) for any field \mathbb{F} .

Lemma 2. $\mathbb{F}\text{-NS}(\text{SOPL}_n) \geq n^{\Omega(1)}$.

Proof. Prior work has shown that SoD_n (understood as an $O(\log n)$ -CNF) requires $n^{\Omega(1)}$ -degree \mathbb{F} -NS refutations [[Bus98](#), [BCIP02](#), [dRNMR19](#)], and similarly that SoL_n (understood as an $O(\log n)$ -CNF) requires $n^{\Omega(1)}$ -degree \mathbb{F} -NS refutations [[BCE+98](#), [BR98](#)]. Define the CNF formula

$$F_n := \text{SoD}_n \wedge \text{SoL}_n$$

where SoD_n and SoL_n are defined on disjoint sets of variables. The following claim (proved below) states that F_n requires $n^{\Omega(1)}$ -degree \mathbb{F} -NS refutations, or, in other words, $\mathbb{F}\text{-NS}(F_n) \geq n^{\Omega(1)}$.

Claim 2. *Let F and G be two CNF contradictions over disjoint sets of variables. If F and G require \mathbb{F} -NS refutations of degree $\geq d$, then $F \wedge G$ requires \mathbb{F} -NS refutations of degree $\geq d$.*

By the definition of F_n , we have

$$\begin{aligned} \Theta(\text{Res}(F_n)) &= \text{PLS}^{dt}(S(F_n)) = \text{SoD}^{dt}(S(F_n)) \leq O(\log n), \\ \Theta(\text{uSA}(F_n)) &= \text{PPADS}^{dt}(S(F_n)) = \text{SoL}^{dt}(S(F_n)) \leq O(\log n). \end{aligned}$$

By the intersection theorem ([Theorem 6](#)) corresponding to $\text{SOPL} = \text{PLS} \cap \text{PPADS}$ we conclude that $S(F_n)$ has an efficient SOPL-formulation:

$$\Theta(\text{RevRes}(F_n)) = \text{SOPL}^{dt}(S(F_n)) = \text{SoPL}^{dt}(S(F_n)) \leq O(\log n).$$

If we had $\mathbb{F}\text{-NS}(\text{SoPL}_n) \leq n^{o(1)}$, then because $S(F_n)$ reduces to $\text{SoPL}_{n^{O(1)}}$ via an $O(\log n)$ -depth decision tree reduction, we would have $\mathbb{F}\text{-NS}(F_n) \leq n^{o(1)}$, which is a contradiction. \square

Proof of Claim 2. The least degree of an \mathbb{F} -NS refutation of a set of polynomial equations $\mathcal{F} := \{a_i(x) = 0\}$ can be characterised by the maximum d such that \mathcal{F} admits a d -design [Bus98, §2], that is, an \mathbb{F} -linear map $\varphi: \mathbb{F}[x] \rightarrow \mathbb{F}$ satisfying (i) $\varphi(1) = 1$, and (ii) $\varphi(q(x) \cdot a_i(x)) = 0$ for all a_i and $q \in \mathbb{F}[x]$ such that $\deg(q) + \deg(a_i) < d$. Let φ and φ' be d -designs for F and G (encoded as sets of polynomial equations) over variables x and y , respectively. For each monomial $m(x)m'(y)$ in variables x, y , we define $\Phi(m(x)m'(y)) := \varphi(m(x)) \cdot \varphi'(m'(y))$. We can extend this definition linearly into a map $\Phi: \mathbb{F}[x, y] \rightarrow \mathbb{F}$. We claim that Φ is a d -design for $F \wedge G$. Indeed, for (i) we have $\Phi(1) = \varphi(1)\varphi'(1) = 1$. For (ii) it suffices to check the condition for each monomial $q(x, y) = m(x)m'(y)$ and an axiom $a_i(x)$ of F (the case of G is analogous) with $\deg(q) + \deg(a_i) < d$. We have $\Phi(m(x)m'(y) \cdot a_i(x)) = \varphi(m(x) \cdot a_i(x))\varphi'(m'(y)) = 0 \cdot \varphi'(m'(y)) = 0$. \square

5 Resolution vs. Sherali–Adams

In this section we prove [Theorem 1](#), restated below.

Theorem 1. *There are n -variate CNF formulas F that can be refuted by constant-width Resolution, but such that any SA refutation of F in degree $n^{o(1)}$ requires coefficients of magnitude $\exp(n^{\Omega(1)})$.*

We consider the SOD principle. We first show that it requires large coefficients to refute in low-degree SA, and then we recall why it has low-width Resolution refutations.

5.1 Lower bound for SA

We consider the SoD_{n^2} search problem on the grid $[n^2] \times [n^2]$. We think of this large grid as being further subdivided into n^2 many subgrids, each of size $n \times n$. The (i, j) -subgrid consists of nodes

$$((i-1)n, (j-1)n) + [n] \times [n] := \{((i-1)n + i', (j-1)n + j') : (i', j') \in [n] \times [n]\}.$$

Recall that the input to this search problem consists of a successor $s_u \in [n^2] \cup \{\text{null}\}$ for each grid node u . For the purposes of SA, we encode this input by a string $x \in \{0, 1\}^{n'}$ over $n' = O(n^4 \log n)$ variables. Moreover, we can think of SoD_{n^2} as a set of unsatisfiable polynomial equations $\{a_i(x) = 0\}$ each of degree $O(\log n)$. Our goal is to prove the following lemma.

Lemma 3. *Any degree- $n^{o(1)}$ SA proof of SoD_{n^2} requires coefficients of magnitude $\exp(\Omega(n))$.*

Suppose we are given a degree- $n^{o(1)}$ SA refutation of SoD_{n^2} over the reals,

$$\sum_{i \in [m]} p_i(x) a_i(x) = 1 + J(x). \tag{9}$$

Our idea is to apply [Lemma 1](#) iteratively in stages to find a sequence of inputs x_1, \dots, x_n with a RHS value $1 + J(x_i) \geq 2^{\Omega(i)}$. Hence [Lemma 3](#) follows at stage $i = n$, since there are at most $\exp(n^{o(1)})$ many monomials, and so one of them must have a coefficient of exponential magnitude.

We start by preprocessing the SA refutation (9) for technical convenience. We may assume wlog that each term t appearing in $J = \sum_t \alpha_t t$ satisfies the following.

1. t is *node-aligned*: if t reads some variable associated with a node u , then it reads all the $O(\log n)$ variables associated with u . To ensure this, we may replace a term t with an equivalent sum of two terms, $t = tx_i + t\bar{x}_i$, which reads one more variable. Adding more literals to terms like this will only increase the degree of the proof by an $O(\log n)$ factor.

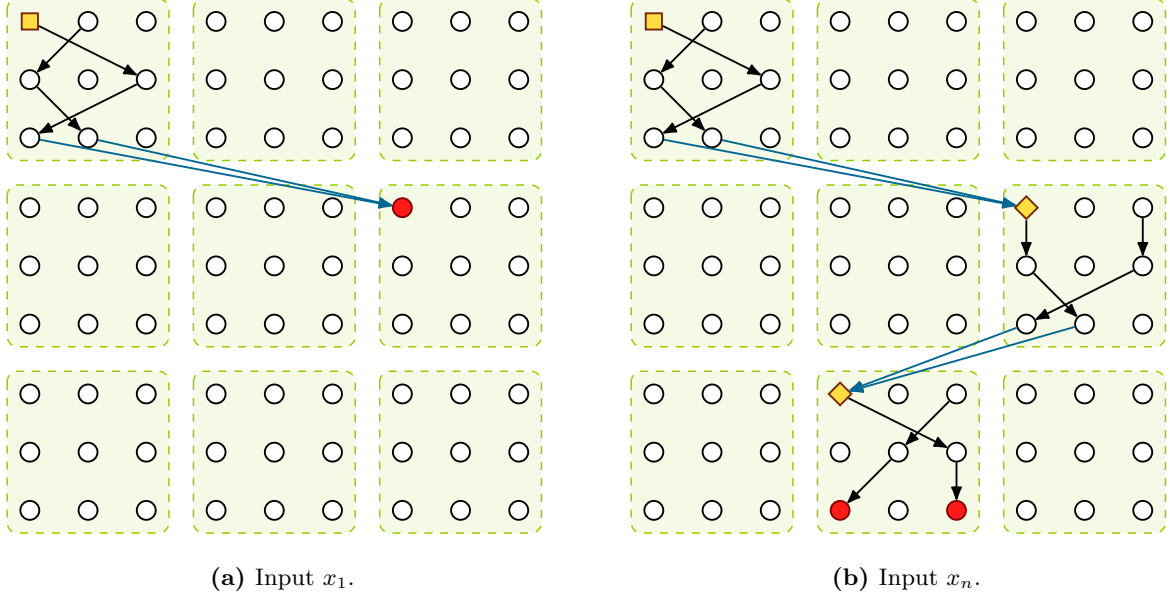


Figure 5: Illustration of the proof of [Lemma 3](#). (a) In the first stage, we construct an input x_1 to SoD_{n^2} that embeds an input y_1 to SoPL_n in the top-left $(1, 1)$ -subgrid, and moreover, all the active sinks of y_1 are assigned as successor the top-left corner of some $(2, j)$ -subgrid. (b) The completed construction after n stages.

2. t is *curious*: if t reads a node u that lies on the last row of a subgrid, that is, $u \in \{in\} \times [n^2]$ for some $i \in [n]$, then t also reads the successor s_u of u (if any) on the next row. Similarly as above, this can be ensured by at most doubling the degree of the proof.
3. t is *non-witnessing*: it does not witness a solution to the search problem. Formally, t witnesses a violation $a_i \neq 0$ if for all x , $t(x) = 1 \Rightarrow a_i(x) \neq 0$ (or contrapositively, $a_i(x) = 0 \Rightarrow t(x) = 0$). To ensure this, if t is witnessing, we can factor² $t = p'_i a_i$ and move t to the LHS of the proof.

First stage. Let y_1 be an input to SoPL_n defined on nodes $[n] \times [n]$. We can embed y_1 inside an input to SoD_{n^2} as follows. We write $(\text{null}^* \leftarrow y_1)$ for the input to SoD_{n^2} where we start with an assignment of null to all nodes $[n^2] \times [n^2]$ (denoted null^*), and then overwrite the top-left $(1, 1)$ -subgrid with the successor pointers in y_1 (aligning the distinguished nodes of SoD_{n^2} and SoPL_n). In this reduction, we can forget the predecessor pointers, as they are not part of the input to SoD . Now every solution of $(\text{null}^* \leftarrow y_1)$ for SoD_{n^2} corresponds naturally to a solution of y_1 for SoPL_n . (A minor detail is that the *active sinks* in y_1 correspond to *proper sinks* in $(\text{null}^* \leftarrow y_1)$.) Using this reduction, we can view our SA refutation of SoD_{n^2} also as a refutation of SoPL_n .

We claim that there is some input y_1 to SoPL_n such that for $x'_1 := (\text{null}^* \leftarrow y_1)$ we have a RHS value $1 + J(x'_1) \geq 1.5$. Suppose not: then the RHS is always in $[1, 1.5]$ for all y_1 , which means we have a low-degree $\frac{1}{2}$ -NS proof of SoPL_n . But this contradicts [Lemma 1](#).

We have now found an input $x'_1 = (\text{null}^* \leftarrow y_1)$ with RHS at least 1.5. Before we iterate this argument in the second stage, we have to clean up x'_1 slightly.

²The existence of such a factorization is easy to see here since both a_i and t are conjunctions of literals. More generally, the existence of such a factorization is guaranteed for any multilinear polynomials a_i and t satisfying $a_i(x) = 0 \Rightarrow t(x) = 0$ on the boolean hypercube, where we simplify expressions using the constraints $x_i^2 - x_i = 0$. One way to prove this is by using the fact that two multilinear polynomials are syntactically identical if and only if they agree on the boolean hypercube.

First stage: Clean-up. Recall that the instances considered in the proof of [Lemma 1](#) consist of some number of directed paths that terminate at sinks $\text{Sol}(y_1) \subseteq \{n\} \times [n]$. We will modify x'_1 by making the nodes $\text{Sol}(y_1)$ point to the same *top-left corner* of a $(2, j)$ -subgrid for some $j \in [n]$. Indeed, let $\rho_j: \text{Sol}(y_1) \rightarrow [n^2]$ be the partial assignment that assigns $(n, (j-1)n) + (1, 1)$ (top-left corner of the $(2, j)$ -subgrid) as the successor of all nodes in $\text{Sol}(y_1)$. Let $(x'_1 \leftarrow \rho_j)$ be the input obtained from x'_1 by applying ρ_j . (We actually have $x'_1 = x'_1 \leftarrow \rho_1$, as this is how we decided to make every node in $\text{Sol}(y_1)$ an active sink in y_1 .) By defining $x_1 := (x'_1 \leftarrow \rho_j)$ for a carefully chosen $j \in [n]$ (see [Figure 5a](#)), we establish the following properties for the start of the next stage.

- (1a) The only solutions in x_1 are proper sinks pointing to the corner of the $(2, j)$ -subgrid.
- (1b) We have $1 + J(x_1 \leftarrow y_2) \geq 1.4$ for any partial assignment y_2 to nodes in the $(2, j)$ -subgrid.

Property [\(1a\)](#) is true by construction and we prove property [\(1b\)](#) below.

Claim 3. *There exists a $j \in [n]$ such that [\(1b\)](#) holds.*

Proof. Let us first prove that for every term t appearing in $J = \sum_t \alpha_t t$, we have

$$t(x'_1) = t(x'_1 \leftarrow \rho_j) \quad \forall j. \tag{10}$$

It suffices to show that any term t in J with $t(x'_1) = 1$ (or $t(x'_1 \leftarrow \rho_j) = 1$) does not read any nodes in $\text{Sol}(y_1)$. Assume for contradiction that such a t reads a node $u \in \text{Sol}(y_1)$. Then, because t is curious, it also reads u 's successor node (note that $s_u \neq \text{null}$ in both x'_1 and $x'_1 \leftarrow \rho_j$) on the next row. This successor node is set to null in x'_1 (and $(x'_1 \leftarrow \rho_j)$) and hence t witnesses that u is a solution (proper sink). But this contradicts our assumption that t is non-witnessing. This proves [\(10\)](#).

Define $J_j := \sum_{t \in T_j} \alpha_t t$ where T_j is the set of terms t in J that do *not* read any node from the $(2, j)$ -subgrid. Note that each t can read from at most $\deg(t) \leq n^{o(1)}$ many different subgrids, and hence if we choose $j \sim [n]$ at random, $\Pr[t \in T_j] \geq 99\%$. We now have

$$\mathbb{E}[1 + J_j(x'_1)] = 1 + \sum_t \Pr[t \in T_j] \alpha_t t(x'_1) \geq 99\% \cdot (1 + J(x'_1)) \geq 99\% \cdot 1.5 \geq 1.4.$$

By averaging, there is some fixed $j \in [n]$ such that $1 + J_j(x'_1) \geq 1.4$. Defining $x_1 := (x'_1 \leftarrow \rho_j)$ for this particular j , we have, for every assignment y_2 to the $(2, j)$ -subgrid,

$$1 + J(x_1 \leftarrow y_2) \geq 1 + J_j(x_1 \leftarrow y_2) = 1 + J_j(x_1) \stackrel{(10)}{=} 1 + J_j(x'_1) \geq 1.4. \quad \square$$

Second stage. Here we start with the input x_1 satisfying [\(1a\)](#)–[\(1b\)](#) for some $j \in [n]$. Let y_2 be any input to SOPL_n . We think of y_2 (ignoring predecessor pointers) as embedded in the $(2, j)$ -subgrid. Consider the input $(x_1 \leftarrow y_2)$ where the distinguished node of y_2 is aligned with corner of the $(2, j)$ -subgrid, which is the only sink in x_1 by [\(1a\)](#). Then every solution of $(x_1 \leftarrow y_2)$ for SOD_{n^2} corresponds to a solution of y_2 for SOPL_n . Hence we can view our SA refutation of SOD_{n^2} as a refutation of SOPL_n (this time in the $(2, j)$ -subgrid). Moreover, we have from [\(1b\)](#) that the RHS of the proof evaluates to $1 + J(x_1 \leftarrow y_2) \geq 1.4$ for all y_2 . If we scale our original SA proof by a factor $1/1.4$, we get another polynomial identity

$$\frac{1}{1.4} \sum_{i \in [m]} p_i(x) a_i(x) = \frac{1}{1.4} (1 + J(x)), \tag{11}$$

where the RHS evaluates to at least 1 on any input of the form $x = (x_1 \leftarrow y_2)$. Using [Lemma 1](#) we can now conclude that there must exist an input $x'_2 = (x_1 \leftarrow y_2)$ such that $\frac{1}{1.4} (1 + J(x'_2)) \geq 1.5$, or equivalently, $1 + J(x'_2) \geq 1.5 \cdot 1.4$.

Second stage: Clean-up. Using exactly the same argument as in the first clean-up stage, we conclude that x'_2 can be cleaned up into x_2 such that for some $j \in [n]$ (different j than in first stage):

- (2a) The only solutions in x_2 are proper sinks pointing to the corner of the $(3, j)$ -subgrid.
- (2b) We have $1 + J(x_2 \leftarrow y_3) \geq 1.4^2$ for any partial assignment y_3 to nodes in the $(3, j)$ -subgrid.

By continuing this argument in the same fashion, we can eventually, at stage n , find an input x_n with $1 + J(x_n) \geq 1.4^n$ (see [Figure 5b](#)). This concludes the proof of [Lemma 3](#).

5.2 Upper bound for Resolution

It is well-known that SoD_n (understood as an $O(\log n)$ -width CNF contradiction) admits an $O(\log n)$ -width Resolution refutation (e.g., [[Kam20](#), Theorem 8.18]). If we want to further optimise this down to a constant-width refutation, as claimed by [Theorem 1](#), then we can consider a sparse variant of SoD_n similarly as we did in [Section 4.3](#). We omit the details.

6 Proofs of Characterisations

In this section we prove [Theorem 3](#), restated below.

Theorem 3. *For any unsatisfiable CNF formula F , we have:*

- $\text{PPAD}^{dt}(S(F)) = \Theta(\text{uNS}(F))$.
- $\text{PPADS}^{dt}(S(F)) = \Theta(\text{uSA}(F))$.
- $\text{SOPL}^{dt}(S(F)) = \Theta(\text{RevRes}(F))$.
- $\text{EOPL}^{dt}(S(F)) = \Theta(\text{RevResT}(F))$.

Recall that the notation $A^{dt}(B)$ for total search problems A, B is the minimum complexity (namely, log size + depth) of an A -formulation of B . Similarly, for a proof system P and CNF formula F the notation $P(F)$ is the minimum of $\log \text{size}(\Pi) + \text{deg}(\Pi)$ where Π is a P -proof of F .

6.1 Unary Nullstellensatz and PPAD

We first argue that unary Nullstellensatz corresponds to the decision tree class PPAD^{dt} .

Theorem 7. *Let F be an unsatisfiable CNF formula. Then,*

- *If F has a degree- d size- L uNS proof, then $S(F)$ has a depth- $O(d)$ $\text{EOL}_{O(L)}$ -formulation.*
- *If $S(F)$ has a depth- d EOL_L -formulation, then F has a degree- $O(d)$ size- $L2^{O(d)}$ uNS proof.*

In particular, $\text{PPAD}^{dt}(S(F)) = \Theta(\text{uNS}(F))$.

Corollary 3. *For any sequence F_n of $\text{poly}(\log n)$ -width CNF formulas, F_n has a degree- $\text{poly}(\log n)$, size- $n^{\text{poly}(\log n)}$ unary Nullstellensatz proof if and only if $S(F) \in \text{PPAD}^{dt}$.*

We prove [Theorem 7](#) in the next two lemmas. The proof of this theorem is itself modelled on a similar characterization of PPA-formulations by \mathbb{F}_2 -Nullstellensatz, proved by [[BCE+98](#), [GKRS18](#)]. It turns out to be easier to show that EOL -formulations imply Nullstellensatz proofs, so we do that first. Furthermore, we will assume that all of our Nullstellensatz proofs are *multilinearized*: that is, we work modulo the $x_i^2 - x_i = 0$ equations, and so the individual degree of any variable in the proof is at most 1. It is well-known that making this assumption will not change the degree or size of the proof by more than a constant factor [[Bus98](#)].

Lemma 4. *Let F be an unsatisfiable CNF formula. If there is a depth- d EoL _{L} -formulation of $S(F)$ then there is a unary Nullstellensatz refutation of F with degree $O(d)$ and size $L2^{O(d)}$.*

Proof. Suppose $F := C_1 \wedge \dots \wedge C_m$ is on n variables x_1, \dots, x_n , and let \overline{C}_i be the negation of C_i represented as a polynomial. Assume that there is a depth- d EoL _{L} -formulation of $S(F)$. Let $V := [L]$ be the set of nodes in the EoL formulation and let $v^* = 1$ denote the distinguished source node. Each node $v \in V$ is equipped with successor and predecessor functions $s_v, p_v : \{0, 1\}^n \rightarrow V$, respectively, each computed by decision trees of depth at most d , as well as a solution decision tree $g_v : \{0, 1\}^n \rightarrow [m]$ that outputs a corresponding solution of $S(F)$. For any input assignment $x \in \{0, 1\}^n$ let G_x denote the directed graph obtained by evaluating all the successor and predecessor decision trees on input x and adding an edge (u, v) iff $s_u(x) = v$ and $p_v(x) = u$.

For each $v \in V$ define the function $S_v : \{0, 1\}^n \rightarrow \{-1, 0, 1\}$ by

$$S_v(x) := \begin{cases} -1 & \text{if } v \neq v^* \text{ is a source in } G_x \\ 1 & \text{if } v \neq v^* \text{ is a proper sink in } G_x \text{ or } v = v^* \text{ and } v^* \text{ is not a source} \\ 0 & \text{otherwise.} \end{cases}$$

We compute S_v for each node v by a depth at most $5d$ decision tree as follows. First, we compute $s_v(x) = u$ and $p_v(x) = w$, and then compute $p_u(x)$ and $s_w(x)$. From this information we can determine the output value of S_v , and we have used at most $4d$ queries. If $S_v = 0$ then the leaf of the decision tree is labelled with 0. Otherwise, if $S_v \neq 0$ then v is a solution to EoL, and so in this case we will also run the decision tree for g_v and label each leaf with either 1 or -1 according to the output value of S_v . Overall this requires at most $5d$ queries.

Now, for any leaf ℓ in the decision tree for S_v let D_ℓ denote the polynomial representation of the conjunction of literals on the path from the root of the tree to ℓ . Observe that we can represent

$$S_v = \sum_{(-1)\text{-leaf } \ell} -D_\ell + \sum_{1\text{-leaf } \ell} D_\ell,$$

where the first sum is over leaves of S_v labelled with -1 and the second is over leaves of S_v labelled with 1. If ℓ is a non-zero leaf then v is a solution to the EoL instance, so let C_ℓ denote the solution of $S(F)$ output by the decision tree g_v at this leaf. Observe that at every non-zero leaf ℓ , the clause C_ℓ must be falsified by the assignment on the path to ℓ , since C_ℓ is a solution to $S(F)$ by the correctness of the EoL formulation and by the fact that we ran the g_v decision tree in S_v . This implies that for each non-zero leaf ℓ of S_v we can write $D_\ell = D'_\ell \overline{C}_\ell$, and thus

$$S_v = \sum_{(-1)\text{-leaf } \ell} -D_\ell + \sum_{1\text{-leaf } \ell} D_\ell = \sum_{(-1)\text{-leaf } \ell} -D'_\ell \overline{C}_\ell + \sum_{1\text{-leaf } \ell} D'_\ell \overline{C}_\ell.$$

If we sum up these polynomials for each $v \in V$ and gather terms then

$$\sum_{v \in V} S_v = \sum_{i=1}^m p_i \overline{C}_i$$

for some polynomials p_i . Note that each polynomial has degree at most $5d$ since they are obtained from the underlying S_v decision trees.

To see that $\sum_{i=1}^m p_i \overline{C}_i$ is a unary Nullstellensatz refutation of F , observe that since each S_v came from an EoL formulation we have

$$\sum_{i=1}^m p_i(x) \overline{C}_i(x) = \sum_{v \in V} S_v(x) = (\# \text{ sinks in } G_x) - (\# \text{ non-distinguished sources in } G_x) = 1$$

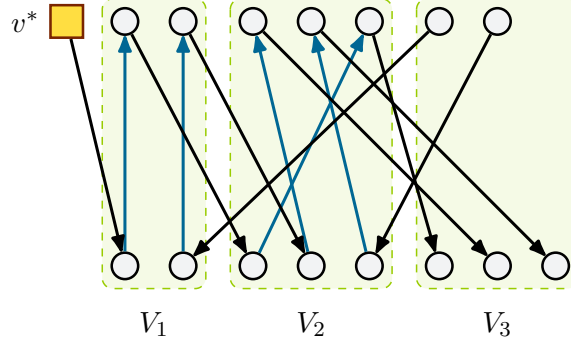


Figure 6: High level illustration of the EOL instance constructed in the proof of [Lemma 5](#). Edges of the outer matching are shown in black, while those from the inner matching are in blue. In this example, the clause corresponding to V_3 is not satisfied by assignment x and as a result no internal edges are added in V_3 . Note that EOL solutions indeed only occur in V_3 .

for any input $x \in \{0, 1\}^n$. Finally, we observe that all coefficients used in this proof are integers, and the number of distinct monomials produced is at most $|V|2^{O(d)} = L2^{O(d)}$ from expanding the depth- d decision trees as polynomials. \square

The more difficult direction is the converse, proved next.

Lemma 5. *Let F be an unsatisfiable CNF formula. If there is a unary Nullstellensatz refutation of F with degree d and size L then there is a depth- $O(d)$ $\text{EOL}_{O(L)}$ -formulation of $S(F)$.*

Proof. Let $F = C_1 \wedge \dots \wedge C_m$ and consider a degree- d , size- L unary Nullstellensatz refutation of F , which we write as

$$\sum_{i=1}^m p_i \bar{C}_i = 1$$

where each p_i is a multilinear polynomial over x_1, \dots, x_n and all coefficients are integers.

To build the EOL formulation, we expand the above proof out into its constituent monomials with multiplicity. That is, for each $i \in [m]$ write the polynomial

$$p_i \bar{C}_i = \sum_j c_{i,j} q_{i,j}$$

where $c_{i,j} \in \mathbb{Z}$ and $q_{i,j}$ is a monomial obtained by expanding the polynomial directly and performing all necessary cancellations. Each node in our EOL formulation will represent one of the above monomials $q_{i,j}$ and is considered a “+” or a “−” node, depending on that monomial’s sign. In total, we create $m + 1$ sets of nodes V^*, V_1, \dots, V_m , defined as follows. The set V^* only contains the distinguished source node v^* , which we consider as a “−” node. For each $i \in [m]$ the set V_i contains a node for each monomial $q_{i,j}$ from the above expansion with multiplicity. So, in particular, we add $|c_{i,j}|$ copies of the monomial $q_{i,j}$ to V_i for each monomial $q_{i,j}$ in the above expansion. Let V denote the set of all nodes produced by this construction. For every node $v \in V$, the decision tree for g_v will query no variables and output C_i if $v \in V_i$ and an arbitrary clause if $v = v^*$; our construction will explicitly prevent the source node v^* from being a solution.

Now, we must describe the successor and predecessor decision trees s_v, p_v at each node. It will be easier to describe the possible edges in G_x on an input $x \in \{0, 1\}^n$; all of the edges are organized into two different matchings as detailed next. See [Figure 6](#) for a high-level illustration.

Outer Matching. In this matching we add edges between nodes in different node groups. All directed edges will be oriented from “−” nodes to “+” nodes. Since the polynomials form a Nullstellensatz refutation over \mathbb{Z} , we know that each time the monomial q appears with a “+” sign, it must also appear with a “−” sign, except for the single 1 term. Thus by treating the distinguished source v^* as “−1”, we can create a perfect matching M on the nodes of V where all matched nodes are between a “+” and a “−” node standing for the same monomial. Since we have gathered terms within the expansions $p_i \overline{C}_i$, all occurrences of monomials q within a set V_i have the same sign, and thus all the edges in this matching will be between nodes in different sets. Formally, in the EoL formulation, for each edge $e = (u, v)$ in M corresponding to a monomial q , we add a directed edge from the “−” to the “+” node if and only if $q(x) = 1$. This condition can be determined by s_u and p_v by querying the variables occurring in q .

Inner Matching. In this matching we add directed edges from “+” nodes to “−” nodes within the same node group. Consider any set V_i . Formally, at each node occurring in the group V_i , we query all variables of the corresponding clause C_i in both the successor and predecessor functions for that node. For any $x \in \{0, 1\}^n$, if $C_i(x) = 1$ then $\overline{C}_i = 0$ and thus $p_i(x)\overline{C}_i(x) = 0$. This means that under the partial restriction ρ consistent with x at the variables of C_i , all monomials remaining in $p_i \overline{C}_i \upharpoonright \rho$ must cancel. We can therefore fix a perfect matching between the negative and positive instances of monomials in V_i under ρ , representing the cancellation of monomials under ρ . Then, each edge of this matching is included in the graph if and only if the monomials corresponding to its endpoints evaluate to 1 at x (note that the two endpoints will both evaluate to the same value, since they are matched under ρ). On the other hand, if $C_i(x) = 0$ then we will simply not add any edges to the internal matching of V_i .

Let $x \in \{0, 1\}^n$ be any assignment to the variables of F . The edges of any node $v \in V_i$ associated with a monomial q are determined by querying the variables of C_i and q . This implies that the depth of each decision tree T_v is at most d , and the size is clearly $O(L)$ since every monomial in the proof is represented as a node.

We now verify correctness of the EoL formulation. Since it is well-defined, on every input x the graph G_x will have a solution. Let v be such a solution (either a sink or proper source node) in G_x . By construction, $v \neq v^*$ since the node v^* is always a source node. This implies that $v \in V_i$ for some $i \in [m]$, and so v must be associated with a monomial q . By the construction of the inner and outer matching, v can only be a source or sink node in V_i if the inner matching is empty. But this can only happen if $C_i(x) = 0$, and thus C_i is a valid solution to $S(F)$. \square

6.2 Unary Sherali–Adams and PPADS

We now show that low-degree unary Sherali–Adams proofs characterise PPADS^{dt} . The proof of this fact follows the proof from the previous section quite closely, but requires some extra work to handle the extra conical junta terms.

Theorem 8. *Let F be an unsatisfiable CNF formula. Then,*

- *If F has a degree- d , size- L unary Sherali–Adams proof, then $S(F)$ has a depth- $O(d)$ $\text{SO}_{O(L)}$ -formulation.*
- *If $S(F)$ has a depth- d SO_L -formulation, then F has a degree- $O(d)$, size- $L2^{O(d)}$ unary Sherali–Adams proof.*

In particular, $\text{PPADS}^{dt}(S(F)) = \Theta(\text{uSA}(F))$.

Corollary 4. *For any sequence F of $\text{poly}(\log n)$ -width CNF formulas, F has a $\text{poly}(\log n)$ -degree, $n^{\text{poly}(\log n)}$ -size unary Sherali–Adams proof if and only if $S(F) \in \text{PPADS}^{dt}$.*

Before we prove the theorem, it will be convenient to have the following simple normal form for Sherali–Adams proofs. Just like in the previous section we will assume that all Sherali–Adams proofs are multilinearized, and it is known that this assumption does not change the degree or size of the proof by more than a constant factor [FKP19].

Lemma 6. *Let F be an unsatisfiable CNF formula. If $\sum_{i=1}^m p_i \bar{C}_i = 1 + J$ is a unary Sherali–Adams refutation of F with degree d and size L , then there is a degree- d , size- L unary Sherali–Adams refutation of F of the form $\sum_{i=1}^m J_i \bar{C}_i = 1 + J_0$ where J_i is a conical junta for each $i = 0, 1, \dots, m$.*

Proof. For each $i \in [m]$ we can expand $p_i = \sum_j c_{i,j} q_{i,j}$ where $c_{i,j}$ are integers and $q_{i,j}$ are monomials. Each monomial $q_{i,j}$ is a conjunction, so the expressions

$$J_i^- = \sum_{j:c_{i,j}<0} |c_{i,j}| q_{i,j}, \quad J_i^+ = \sum_{j:c_{i,j}>0} c_{i,j} q_{i,j}$$

are conical juntas for each $i \in [m]$. Writing $p_i = J_i^+ - J_i^-$, substituting into the Sherali–Adams refutation, and rearranging completes the proof. \square

We now begin the proof of [Theorem 8](#). As before we split the proof into two lemmas, one for each direction of the characterisation. The easier direction is again that an SOL-formulation implies a unary Sherali–Adams proof, and it almost exactly follows the proof of [Lemma 4](#).

Lemma 7. *Let F be an unsatisfiable CNF formula. If there is a depth- d SOL_L -formulation of $S(F)$ then there is a unary Sherali–Adams refutation of F with degree $O(d)$ and size $L2^{O(d)}$.*

Proof. The proof of this lemma is essentially the same as the proof of [Lemma 4](#), so we will simply sketch it and note what needs to be modified. Suppose $F := C_1 \wedge \dots \wedge C_m$ and let \bar{C}_i be the negation of C_i represented as a polynomial. We have an SOL-formulation for $S(F)$, and so we have decision trees computing successors s_v and predecessors p_v for each of the nodes $v \in V$. As in the proof of [Lemma 4](#), for each $v \in V$ we define a depth at most $5d$ decision tree S_v , defined by

$$S_v(x) = \begin{cases} 1 & \text{if } v \neq v^* \text{ is a source in } G_x \\ -1 & \text{if either } v \text{ is a proper sink in } G_x \text{ or if } v = v^* \text{ and } v^* \text{ is not a source} \\ 0 & \text{otherwise,} \end{cases}$$

where we note that we have switched the “ -1 ” and the “ $+1$ ” in the definition of S_v when compared to [Lemma 4](#). As before, $S_v(x)$ can be determined by first running the decision trees for $s_v(x) = u$ and $p_v(x) = w$, then the decision trees for $p_u(x)$, $s_w(x)$, and finally the decision tree for $g_v(x)$ if the node v is a solution to SOL. From this, we can again represent

$$S_v = \sum_{(-1)\text{-leaf } \ell} -D_\ell + \sum_{1\text{-leaf } \ell} D_\ell,$$

where the first sum is over leaves of S_v labelled with -1 and the second is over leaves of S_v labelled with 1 . However, now a node v is only a solution to SOL if $S_v(x) = -1$, and so for each (-1) -leaf ℓ of S_v we can write $D_\ell = D'_\ell \cdot \bar{C}_\ell$ where C_ℓ is the clause of F falsified at that leaf. This allows us to write

$$S_v = \sum_{(-1)\text{-leaf } \ell} -D_\ell + \sum_{1\text{-leaf } \ell} D_\ell = \sum_{(-1)\text{-leaf } \ell} -D'_\ell \cdot \bar{C}_\ell + \sum_{1\text{-leaf } \ell} D_\ell.$$

If we sum up these polynomials for each $v \in V$ and gather terms we get

$$\sum_{v \in V} S_v = \sum_{i=1}^m -J_i \bar{C}_i + J_0$$

for some degree- $O(d)$ conical juntas J_0, J_1, \dots, J_m . As in the proof of [Lemma 4](#) we have that $\sum_v S_v(x) = -1$ and the size and degree calculations are identical. \square

The proof of the converse direction is also similar to the proof of [Lemma 5](#), but requires some more substantial modification when compared to the previous proof. The main issue is how to handle the extra conical junta terms J_0 in the unary Sherali–Adams refutation. As in the proof of [Lemma 5](#), we will create a graph representing all the monomials in the unary Sherali–Adams proof. However, we will do some extra work to ensure that the nodes corresponding to monomials from the conical junta term J_0 will always be source nodes. This ensures that any solutions will occur at nodes corresponding to some falsified clause in the formula.

Lemma 8. *Let F be an unsatisfiable CNF formula. If there is a unary Sherali–Adams refutation of F with degree d and size L then there is a degree- $O(d)$ $\text{SOLO}(L)$ -formulation of $S(F)$.*

Proof. Suppose $F := C_1 \wedge \dots \wedge C_m$ is on n variables and consider a unary Sherali–Adams refutation

$$\sum_{i=1}^m -J_i \bar{C}_i + J_0 = -1$$

of F where each J_i for $i = 0, 1, \dots, m$ are integral conical juntas. For notational convenience, we will let $\bar{C}_0 := -1$ and we will expand each conical junta J_i as a non-negative sum of conjunctions. While this notation is somewhat unusual, it allows us to write the refutation in a uniform way as

$$\sum_{i=1}^m -J_i \bar{C}_i + J_0 = \sum_{i=0}^m \sum_{j=1}^{t_i} -\lambda_{i,j} D_{i,j} \bar{C}_i = -1$$

where t_i is a non-negative integer, $\lambda_{i,j}$ is a positive integer, and $D_{i,j}$ is a conjunction for every i, j .

To build the SOLO formulation, we expand the above proof out into its constituent monomials with multiplicity. As in the proof of [Lemma 5](#), each node in our SOLO formulation will represent a monomial in the proof and is either a “+” or a “−” node, depending on that monomial’s sign. This time, however, we create a group of nodes $V_{i,j}$ for each $i = 0, 1, \dots, m$ and each $j \in [t_i]$, as well as a special group V^* . The group V^* only contains the distinguished node v^* , which we now consider as a “+” node. On the other hand, for each i, j , the group $V_{i,j}$ will correspond to the polynomial $-\lambda_{i,j} D_{i,j} \bar{C}_i$. We expand this polynomial into a sum of monomials $-\lambda_{i,j} D_{i,j} \bar{C}_i = \sum_q c_q q$ for some integers c_q and monomials q , and for each monomial q in this expansion we create $|c_q|$ nodes in $V_{i,j}$, each of which are “+” nodes if $c_q > 0$ and “−” nodes otherwise. Let V denote the set of all nodes produced by this construction. For any node $v \in V$, if $v \in V_{i,j}$ for some $i > 0$ then the solution decision tree g_v will query no variables and simply output C_i as the solution to $S(F)$. Otherwise, g_v will output an arbitrary solution, as in this case by construction of the formulation the node v will never be a solution to SOLO.

Now, we must describe the successor and predecessor decision trees at each node. As in the proof of [Lemma 5](#), it will be easier to describe the possible edges in G_x as all of the edges are organized into two different matchings.

Outer Matching. The definition of the outer matching is the same as in Lemma 5. In this matching we add edges between nodes in different node groups. All directed edges will be oriented from “+” nodes to “−” nodes. Since the polynomials form an SA refutation over \mathbb{Z} , we know that each time the monomial q appears with a “+” sign, it must also appear with a “−” sign, except for the single -1 term. Thus by considering v^* as “+1”, we can create a perfect matching M of the nodes of V where all edges are between a “+” and a “−” node standing for the same monomial. Since we have gathered terms within the expansions of $-\lambda_{i,j}D_{i,j}\overline{C}_i$, all occurrences of monomials q within a single group $V_{i,j}$ have the same sign and thus all the matchings are between nodes in different sets. For each edge e in M , we will add a directed edge between the “+” and the “−” node if and only if $q(x) = 1$; this can be determined by querying all variables in q .

Inner Matching. The inner matching is constructed similarly as in the proof of Lemma 5, but requires some modification. As in that proof, in the inner matching we add directed edges from “−” nodes to “+” nodes within the same node group. However, we will now be careful to force any solution (i.e. a sink node) to occur at a “−” node in G_x . By our construction, the V^* group has no “−” nodes, and all “−” nodes in the group $V_{0,j}$ for any $j \in [t_0]$ will have successors, and thus any sink node must be associated with $V_{i,j}$ for some $i > 0$.

Consider any set of the form $V_{i,j}$, since V^* has a single node corresponding to +1 and so no internal edges will be matched. Formally, at each node occurring in the group $V_{i,j}$, we query all variables of \overline{C}_i and $D_{i,j}$ (note that when $i = 0$, $\overline{C}_0 = 1$ and so we only query $D_{i,j}$ variables). For any assignment $x \in \{0, 1\}^n$, if $C_i(x) = 1$ then $\overline{C}_i(x) = 0$ and thus $D_{i,j}(x)\overline{C}_i(x) = 0$. This means that under the partial restriction ρ consistent with x at the variables of C_i , all monomials in $D_{i,j}\overline{C}_i \upharpoonright \rho$ must cancel to 0. We can therefore fix a directed perfect matching between the negative and positive copies of monomials in $V_{i,j}$, as in the proof of Lemma 5.

On the other hand, if $\overline{C}_i(x) \neq 0$ then $-\lambda_{i,j}D_{i,j}(x)\overline{C}_i(x) = c$ for some integer c . If $i > 0$ then $c \leq 0$, and so in this case, there will be $|c|$ copies of “−” monomials in V_i that are not cancelled by + monomials internally. We can then fix a directed partial matching between monomials accordingly, but leaving the $|c|$ “−” monomials without successors if required (these will become sink nodes). If $i = 0$ then $c \geq 0$ since $\overline{C}_i = -1$, and so in this case there may be more “+” monomials than “−” monomials evaluating to 1. We can therefore fix a directed partial matching between monomials, now leaving some “+” monomials without predecessors (these will become new source nodes), but all “−” monomials will have successors and so they will not become proper sink nodes.

As we have described above, we will need at most d queries in any decision tree in the reduction, and also the number of nodes in the final SOL instance is no more than the size (number of monomials) of the underlying unary Sherali–Adams proof.

We finally verify correctness of the SOL-formulation. This is a well-defined SOL formulation and thus on every input $x \in \{0, 1\}^n$ the graph G_x will have a solution $v \in V$. This must be a sink node by the definition of SOL and therefore, by construction, v must be a “−” node since “+” nodes always have successors by the construction of the outer matching. As we have described in the definition of the inner matching, any “−” node $v \in V_{0,j}$ for any j will have a successor, and thus $v \in V_{i,j}$ for some $i > 0$. But then, by definition of the inner matching, if v is a sink node in $V_{i,j}$ for $i > 0$ then $C_i(x) = 0$ and the label of v is C_i , thus the SOL formulation correctly outputs a solution to $S(F)$. \square

6.3 Reversible Resolution, SOPL, and EOPL

In this section we define the Reversible Resolution systems (RevRes and RevResT), and prove our final characterisations capturing SOPL^{dt} and EOPL^{dt} .

Theorem 9. *Let F be an unsatisfiable CNF formula. Then,*

- *If F has a width- d , size- L Reversible Resolution proof (with Terminals, resp.), then $S(F)$ has a depth- $O(d)$ $\text{SOPL}_{O(L)}$ -formulation (EOPL -formulation, resp.).*
- *If $S(F)$ has a depth- d SOPL_L -formulation (EOPL_L -formulation, resp.), then F has a width- $O(d)$, size- $L^{O(1)}2^{O(d)}$ Reversible Resolution proof (with Terminals, resp.).*

In particular, $\text{SOPL}^{dt}(S(F)) = \Theta(\text{RevRes}(F))$ and $\text{EOPL}^{dt}(S(F)) = \Theta(\text{RevResT}(F))$.

Corollary 5. *For any sequence F of $\text{poly}(\log n)$ -width CNF formulas, F has a $\text{poly}(\log n)$ -width, $n^{\text{poly}(\log n)}$ -size Reversible Resolution proof (with Terminals, resp.) if and only if $S(F) \in \text{SOPL}^{dt}$ ($S(F) \in \text{EOPL}^{dt}$, resp.).*

Reversible Resolution and MaxSAT

We begin by formally defining Reversible Resolution refutations and comparing them to MaxSAT systems from the literature [BLM07, LHdG08, FMSV23].

Definition 5. Let F be an unsatisfiable CNF formula. If C is a clause then the *reversible weakening rule* is the proof rule $C \vdash C \vee x, C \vee \bar{x}$, and the *reversible resolution rule* is the proof rule $C \vee x, C \vee \bar{x} \vdash C$. A *reversible resolution refutation* (RevRes) of F is a sequence of multisets of clauses $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_t$ such that the following holds:

1. Every clause in \mathcal{C}_1 occurs in F , possibly with multiplicity.
2. The multiset \mathcal{C}_t contains the empty clause \perp .
3. For each $i = 1, 2, \dots, t - 1$, the multiset \mathcal{C}_{i+1} is obtained from \mathcal{C}_i by selecting clauses in \mathcal{C}_i and *replacing* them with the result of one of the two reversible rules applied to those clauses.

The proof is a *reversible resolution refutation with terminals* (RevResT) if every clause in \mathcal{C}_t other than \perp is a weakening of a clause from F . The size of the proof is $\sum_{i=1}^t |\mathcal{C}_i|$ — the number of clauses in all configurations. The width of the proof is the maximum width of any clause occurring in any configuration.

The key difference between the reversible resolution rule and the standard resolution rule is that the output of the reversible rule (as a CNF formula) is logically equivalent to the input of the rule. Despite this restriction, it is clear that we can use the reversible rule to simulate tree-like resolution. If we use clauses $C \vee x$ and $D \vee \bar{x}$ to derive $C \vee D$, then we can derive this in RevRes as follows. First, for each literal in $C \vee x$, apply the reversible weakening rule repeatedly to derive $C \vee D \vee x$ (along with some extra clauses which we can ignore). Similarly, derive $D \vee C \vee \bar{x}$. Then apply the reversible resolution rule to these two clauses to derive $C \vee D$.

However, [Theorem 1](#) implies that RevRes *cannot* efficiently simulate Resolution. Intuitively, this is because of property (3) in the definition of a reversible refutation: we must *replace* the clauses used in the rule with new clauses. Therefore we cannot “duplicate” derived clauses for free, which is essential to obtain the full power of Resolution.

Indeed, the RevRes proof system is a slight strengthening of the proof system *MaxSAT Resolution with Weakening* (also denoted MaxResW) studied in the literature on MaxSAT solvers [BLM07,

[LHdG08, FMSV23]. The principal difference between MaxSAT Resolution and standard Resolution is that MaxSAT Resolution seeks to preserve the number of satisfied clauses under any assignment. For completeness, we define the MaxSAT Resolution proof system next.

Definition 6. Let $A = a_1 \vee \dots \vee a_s$ and $B = b_1 \vee \dots \vee b_t$ be clauses over boolean literals a_i, b_j . The *MaxSAT resolution rule* is the proof rule that, given $x \vee A$ and $\bar{x} \vee B$, deduces the following set of clauses:

$$\begin{aligned}
& a_1 \vee \dots \vee a_s \vee b_1 \vee \dots \vee b_t \\
& x \vee A \vee \bigvee_{i=1}^j b_i \vee \bar{b}_{j+1} \quad \forall j = 0, 1, \dots, t \\
& \bar{x} \vee B \vee \bigvee_{i=1}^j a_i \vee \bar{a}_{j+1} \quad \forall j = 0, 1, \dots, s.
\end{aligned}$$

A *MaxRes refutation* of an unsatisfiable CNF F is a sequence of multisets of clauses $\mathcal{C}_1, \dots, \mathcal{C}_t$ where \mathcal{C}_1 contains exactly the clauses in F , \mathcal{C}_t contains a copy of the empty clause \perp , and the configuration \mathcal{C}_i for $i > 1$ is obtained from \mathcal{C}_{i-1} by applying the MaxSAT resolution rule to some clauses in \mathcal{C}_{i-1} and replacing those clauses with the output of the rule. A *MaxResW refutation* is a MaxRes refutation that is also allowed to use the weakening rule $C \vdash C \vee x, C \vee \bar{x}$.

RevRes can simulate MaxResW proofs without much difficulty. The weakening rule in MaxResW is the reversible weakening rule. To simulate the MaxSAT resolution rule, starting from $x \vee A, \bar{x} \vee B$, apply the reversible weakening rule on $x \vee A$ to weaken it with the variable b_1 , obtaining $x \vee A \vee b_1, x \vee A \vee \bar{b}_1$. Then, weaken $x \vee A \vee b_1$ on the variable b_2 to obtain the clauses $x \vee A \vee b_1 \vee b_2, x \vee A \vee b_1 \vee \bar{b}_2$. Repeating in this fashion on all literals in B , and similarly weakening $\bar{x} \vee B$, we obtain $x \vee A \vee B, \bar{x} \vee A \vee B$, and all the extra clauses output by the MaxSAT rule. Finally applying the reversible resolution rule to $x \vee A \vee B$ and $\bar{x} \vee A \vee B$ deduces $A \vee B$.

The converse direction, however, is not clear and could very well be false. A significant difference between RevRes and MaxResW is the fact that MaxResW proofs must have the initial configuration *exactly* equal to F , while RevRes can start with any multiset of clauses from F . As discussed above, this is because the goal of MaxRes is to preserve the number of satisfied clauses under any assignment, while RevRes has no such requirements and simply seeks to prove unsatisfiability.

We can formally interpret this as follows. Suppose we are given an unsatisfiable CNF formula $F = C_1 \wedge \dots \wedge C_m$, where every clause C_i is equipped with a positive integer weight w_i . Since F is unsatisfiable, the maximum possible weight of satisfied clauses in any assignment to the variables of F is at most $\sum_{i=1}^m w_i - 1$. Thus, if we could *prove* that this is true for some choice of weights $w_i > 0$, then we have verified that the formula F is unsatisfiable.

RevRes implements this idea. Given F , we start by choosing positive integer weights w_i for each clause C_i , and make w_i copies of C_i in the initial configuration \mathcal{C}_1 . The two proof rules in RevRes preserve the number of satisfied clauses under any assignment, and so it follows that if $\mathcal{C}_1, \dots, \mathcal{C}_t$ is a RevRes refutation of F then, since \mathcal{C}_t contains at least one instance of \perp , it must be that the maximum weight of satisfied clauses under any assignment is at most $\sum_{i=1}^m w_i - 1$ since \perp is always false. Hence the formula F must be unsatisfiable. Interpreted in this way, RevRes sits between MaxResW and the weighted MaxSAT resolution systems defined in [LHdG08].

Characterisation theorems

Unlike the characterisation theorems for unary Nullstellensatz and unary Sherali–Adams, the easier direction for this characterisation theorem is showing that RevRes proofs imply SOPL-formulations.

Lemma 9. *Let F be an unsatisfiable CNF formula. If there is a RevRes refutation of F with width d and size L , then there is a depth- $(d + 1)$ SOPL $_L$ -formulation of $S(F)$. Furthermore, if there is a RevResT refutation, then there is a depth- $(d + 1)$ EOPL $_L$ -formulation of $S(F)$.*

Proof. We focus on the case of RevRes and then describe what needs to be modified in the case of RevResT. Let $F = C_1 \wedge \dots \wedge C_m$ be an unsatisfiable CNF formula. Let $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_\ell$ be a RevRes refutation of F of the prescribed size and width and let $t := \max_{i \in [\ell]} |\mathcal{C}_i|$. By the size bound we know that $t, \ell \leq L$.

We create an SOPL-formulation of $S(F)$ on a grid of size $L \times L$, although we will only use the subgrid of size $\ell \times t$ and hardwire all other nodes to be inactive. This can be done for each node (i, j) outside of the $\ell \times t$ grid by setting the successor for (i, j) to be null and the predecessor to be arbitrary. The relationship between the grid of the SOPL-formulation and the RevRes proof is straightforward: the node $(i, j) \in [\ell] \times [t]$ corresponds to the j -th clause in the multiset $\mathcal{C}_{\ell-i+1}$. Without loss of generality, we assume \mathcal{C}_ℓ is ordered so that the first clause is \perp , and thus the distinguished node $(1, 1)$ in the SOPL instance corresponds to \perp .

Let $(i, j) \in [\ell] \times [t]$ be any node in the grid and let $C_{i,j}$ denote the corresponding clause in the proof. We define the successor function $s_{i,j} : \{0, 1\}^n \rightarrow [t] \cup \{\text{null}\}$, the predecessor function $p_{i,j} : \{0, 1\}^n \rightarrow [t]$, and the solution function $g_{i,j} : \{0, 1\}^n \rightarrow [m]$. The solution function $g_{i,j}$ queries no variables and outputs $C_{i,j}$ if $C_{i,j} \in F$, and otherwise outputs an arbitrary solution (in the second case, by construction (i, j) will never be a solution to SOPL). To define $s_{i,j}$ and $p_{i,j}$ we introduce some notation. If $C \in \mathcal{C}_i$ and $C' \in \mathcal{C}_{i+1}$ are clauses in adjacent configurations then C' is *derived from* C , written $C \vdash C'$, if either C' is the output of a reversible proof rule applied to C or if no proof rule was applied to C and $C' = C$ is just the same copy of C in the next configuration. For any $x \in \{0, 1\}^n$ define

$$s_{i,j}(x) := \begin{cases} \text{null} & \text{if } C_{i,j}(x) = 1 \\ k & \text{if } i < \ell, C_{i,j}(x) = C_{i+1,k}(x) = 0, \text{ and } C_{i+1,k} \vdash C_{i,j} \\ 1 & \text{if } i = \ell \text{ and } C_{i,j}(x) = 0, \end{cases}$$

and similarly, if $i > 1$, define

$$p_{i,j}(x) := \begin{cases} 1 & \text{if } C_{i,j}(x) = 1 \\ k & \text{if } C_{i,j}(x) = C_{i-1,k}(x) = 0 \text{ and } C_{i,j} \vdash C_{i-1,k}. \end{cases}$$

Intuitively, if $C_{i,j}(x) = 0$ then we will make the successor and predecessors of $C_{i,j}$ point to the unique clauses in the adjacent configurations that are guaranteed to be false. These functions are well-defined since the reversible rules are of the form $C \vee x_i, C \vee \bar{x}_i \vdash C$ and $C \vdash C \vee x_i, C \vee \bar{x}_i$. In particular, under any assignment to the variables, the number of false clauses in the input and output of the rules are equal and at most 1, and thus if C is false then there are unique false clauses in the adjacent configurations that are derived from or used to derive C . Finally, we note that the successor and predecessor functions can each be computed by querying all the variables in $C_{i,j}$ and possibly one more variable (the one that was resolved or weakened on), and thus the decision tree depth of both of these functions is at most $d + 1$.

Now we argue that the SOPL formulation correctly solves $S(F)$. By the definition of the successor and predecessor functions, if any node (i, j) on layer $i < \ell$ is active, then that node has consistent pointers to successor nodes and predecessor nodes on the adjacent layers. This means that the node (i, j) is a solution only if it is an active node on layer $i = \ell$, but such a node is active only if the corresponding clause $C_{i,j} \in \mathcal{C}_1$ is false. But all such clauses occur in F , and in this case the solution function $g_{i,j}$ outputs $C_{i,j}$, which is a correct solution to $S(F)$.

In case we started with a RevResT refutation, we observe that the same argument described above also works for EOPL with one extra observation: any clause in the final configuration \mathcal{C}_t that is falsified under an input x is now a weakening of an input clause of F , and so this is a valid source node solution to the EOPL problem. \square

It remains to prove the converse, which is harder. As a warmup, we begin by showing that the encoding of SOPL (EOPL) as an unsatisfiable CNF formula can be efficiently refuted in RevRes (RevResT resp.). The general case will follow the structure of this proof closely. For the warmup it will be helpful to explicitly write the CNF encoding of SOPL and EOPL (Section 3).

Explicit Encodings for SOPL and EOPL. As we have discussed in Section 2, any total search problem $R_n \subseteq \{0, 1\}^n \times O_n$ has a natural encoding as an unsatisfiable CNF formula by $\bigwedge_{o \in O_n} \neg T_o(x)$ where $T_o(x)$ is the decision tree that checks if $(x, o) \in R_n$. Since T_o is a low-depth decision tree we can encode it as a low-width DNF formula, and thus the resulting CNF formula also has low width. In this section we describe the unsatisfiable CNF formulas corresponding to SOPL_n and EOPL_n explicitly.

The successor and predecessor pointers in the SOPL_n instance will be encoded in binary, so, for the sake of convenience assume $n = 2^\lambda - 1$ for some integer $\lambda \geq 1$ and other cases can be handled similarly. For each node (i, j) the successor and predecessor pointers will be encoded by blocks of boolean variables $s_{i,j} \in \{0, 1\}^\lambda, p_{i,j} \in \{0, 1\}^\lambda$ encoding the value of the pointer in binary. The pointer **null** will always be encoded by the all-0 string. We will abuse notation and often consider $s_{i,j}$ and $p_{i,j}$ as actual elements of $[n] \cup \{\text{null}\}$, rather than as short boolean strings. So, we may write things like $s_{i,j} = k$ for $k \in [n]$ to mean that the bits of $s_{i,j}$ are equal to the binary encoding of k .

As everything is encoded in binary, it will be helpful to introduce the following notation. In general, for a predicate $P : \{0, 1\}^n \rightarrow \{0, 1\}$ we let $\llbracket P \rrbracket$ represent the CNF encoding of P over the n underlying boolean variables. For example, $\llbracket s_{i,j} = \ell \rrbracket$ for $\ell \in [n]$ represents the CNF encoding of the predicate “ $s_{i,j} = \ell$ ” over the boolean variables underlying $s_{i,j}$. Explicitly, $\llbracket s_{i,j} = \text{null} \rrbracket = \bigwedge_{t=1}^\lambda \bar{s}_{i,j,t}$, and similarly $\llbracket s_{i,j} \neq \text{null} \rrbracket$ can be represented by the clause $\bigvee_{t=1}^\lambda s_{i,j,t}$. We can also form more complicated statements, writing e.g. $\llbracket s_{i,j} = k \wedge p_{i+1,k} = j \rrbracket$ to mean the CNF encoding of “the successor of (i, j) is $(i + 1, k)$ and the predecessor of $(i + 1, k)$ is (i, j) ”.

Definition 7. Let n be a positive integer, and for simplicity assume $n = 2^\lambda - 1$ for some integer $\lambda \geq 1$. Consider the following unsatisfiable CNF formula SOPL_n . For each $(i, j) \in \{2, \dots, n-1\} \times [n]$ we have two blocks of λ variables $s_{i,j} \in \{0, 1\}^\lambda, p_{i,j} \in \{0, 1\}^\lambda$ encoding the successor and predecessor pointers of the node (i, j) in binary, where **null** is encoded by 0^λ . For each $j \in [n]$, we additionally have a block of λ variables $s_{1,j} \in \{0, 1\}^\lambda$ encoding the successor of $(1, j)$, a block of λ variables $p_{n,j} \in \{0, 1\}^\lambda$ encoding the predecessor of (n, j) , and a single variable $s_{n,j} \in \{0, 1\}$ encoding whether or not (n, j) is active.

The clauses of SOPL_n are the following:

- For each $j \in [n]$, $\llbracket s_{1,1} \neq j \vee p_{2,j} = 1 \rrbracket$ and $\llbracket s_{1,1} \neq 0 \rrbracket$ *(active distinguished source)*
- For each $j \in [n]$, $\bar{s}_{n,j}$ for each $j \in [n]$, *(inactive sink)*
- For each $(i, j) \in \{1, \dots, n-2\} \times [n]$ and each $a, b \in [n], c \in [n] \cup \{0\}, a \neq c$, *(no proper sinks)*

$$\llbracket s_{i,j} \neq a \vee p_{i+1,a} \neq j \vee s_{i+1,a} \neq b \vee p_{i+2,b} \neq c \rrbracket$$

as well as $\llbracket s_{i,j} \neq a \vee p_{i+1,a} \neq j \vee s_{i+1,a} \neq 0 \rrbracket$. Similarly, for each $a, b \in [n]$,

$$\llbracket s_{n-1,a} \neq b \vee p_{n,b} \neq a \vee s_{n,b} = 1 \rrbracket$$

The EOPL_n formula is obtained by adding the following extra clauses to SOPL_n :

- For each $(i, j) \in \{2, \dots, n-1\} \times [n]$ and each $a, b \in [n]$, $c \in [n] \cup \{0\}$, $c \neq j$, (no proper sources)

$$\llbracket s_{i,j} \neq a \vee p_{i+1,a} \neq j \vee p_{i,j} \neq b \vee s_{i-1,b} \neq c \rrbracket$$

as well as $\llbracket s_{i,j} \neq a \vee p_{i+1,a} \neq j \vee p_{i,j} \neq 0 \rrbracket$. Similarly, for any $a, b \in [n]$ with $a \neq 1$,

$$\llbracket s_{1,a} \neq b \vee p_{2,b} \neq a \rrbracket.$$

From the above definition we can see that both SOPL_n and EOPL_n are polynomial-size, $O(\log n)$ -width CNF formulas, and they are unsatisfiable since the families of clauses simply encode the contradictory statements “the SOPL/EOPL problem has no solution”.

Proofs of Characterisations. Now, before proving that we can refute SOPL_n in RevRes , we first prove a technical lemma that allow us to manipulate binary encodings in RevRes .

Lemma 10. *Let $\lambda > 0$ be a positive integer, and let $n = 2^\lambda - 1$. Let C be a width- k clause that does not depend on a block of boolean variables $z \in \{0, 1\}^\lambda$. Using the reversible weakening rule we can prove, from C , the set of clauses $\{\llbracket C \vee z \neq i \rrbracket : i = 0, \dots, n\}$ in width $k + \lambda$ and size 2^λ . Conversely, from the above set of clauses we can prove C using the reversible resolution rule in the same size and width.*

Proof. Starting from C , apply the reversible weakening rule on the first bit z_1 to obtain $C \vee z_1$ and $C \vee \bar{z}_1$. Weakening each of the results on $z_2, z_3, \dots, z_\lambda$ in turn yields exactly the CNF formula described in the lemma, and the second statement follows from the reversibility of RevRes . \square

Theorem 10. *For each positive integer n , there is a $O(\log n)$ -width, polynomial-size RevRes refutation (RevResT refutation, resp.) of SOPL_n (EOPL_n , resp.).*

Proof. We give the proof for SOPL_n and then describe what needs to be modified for EOPL_n . For each $(i, j) \in [n-1] \times [n]$ and each $k \in [n]$ define the clause $I_{i,j,k} := \llbracket s_{i,j} \neq k \vee p_{i+1,k} \neq j \rrbracket$, and note that $I_{i,j,k}$ has width $2 \log n$ in the variables $s_{i,j}$ and $p_{i+1,k}$. With this notation, the set of clauses

$$I_{i,j} := \{\llbracket s_{i,j} \neq k \vee p_{i+1,k} \neq j \rrbracket \mid k \in [n]\}$$

encodes the statement “the node (i, j) is inactive”. Similarly, for any $j \in [n]$ we define

$$I_{n,j} := \bar{s}_{n,j}$$

encoding that the node (n, j) is inactive, and note that $I_{n,j}$ is a clause in SOPL_n . Thus, for any $i \in [n]$, the collection of clauses $\mathcal{I}_i := \bigcup_{j=1}^n I_{i,j}$ encodes the statement “every node on layer i is inactive”. We now state the main claim of the proof.

Claim 4. *For any $i \in \{2, \dots, n\}$, there is a polynomial-size, $O(\log n)$ -width RevRes proof of \mathcal{I}_{i-1} from \mathcal{I}_i and a polynomial-size collection of clauses from SOPL_n .*

Let us first use the claim to finish the proof of the theorem. We start with the collection of clauses $\mathcal{I}_n = \bigcup_{j=1}^n I_{n,j}$, each of which is a clause from SOPL_n . Applying the claim yields the collection \mathcal{I}_{n-1} in polynomial-size and $O(\log n)$ width from \mathcal{I}_n and a polynomial-size collection of clauses from SOPL_n . Applying the claim $n-2$ more times then yields \mathcal{I}_1 in polynomial-size and $O(\log n)$ width. However, the clauses $I_{1,1} \subseteq \mathcal{I}_1$ are exactly

$$\llbracket s_{1,1} \neq j \vee p_{2,j} \neq 1 \rrbracket$$

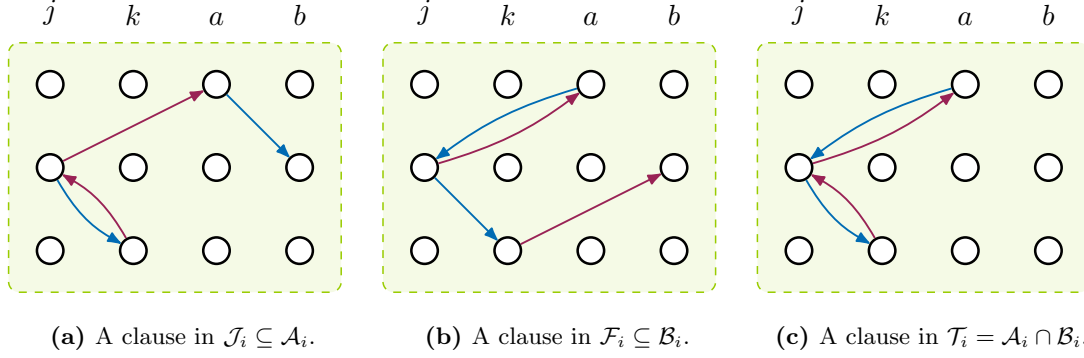


Figure 7: Illustration of the objects in the proof of [Claim 4](#). The blue edges are successor pointers and the red edges are predecessor pointers. Each clause says that at least one of the pointers in the above configuration must not be present.

for each $j \in [n]$. By resolving these clauses with the clauses in $\llbracket s_{1,1} \neq j \vee p_{2,j} = 1 \rrbracket$ in SOPL_n , we can deduce the family of clauses $\llbracket s_{1,1} \neq j \rrbracket$ for all $j \neq 0$, and the clause $\llbracket s_{1,1} \neq 0 \rrbracket$ is already in SOPL_n . Applying [Lemma 10](#) to the clauses $\{\llbracket s_{1,1} \neq i \rrbracket \mid i = 0, \dots, n\}$ deduces the empty clause \perp in $O(\log n)$ width and $O(n)$ size. In sum, the entire proof will have polynomial size and $O(\log n)$ width.

So, it suffices to prove the claim.

Proof of Claim. We show how to prove the general case where $i \leq n - 1$, and the case where $i = n$ is handled by an essentially identical argument. Consider the family of clauses $\mathcal{I}_i = \bigcup_{j=1}^n I_{i,j}$. For each clause $I_{i,j,k}$ apply [Lemma 10](#) to weaken as follows. Initially, we weaken over all values of the predecessor pointer $p_{i,j}$, obtaining the family of clauses $\llbracket I_{i,j,k} \vee p_{i,j} \neq a \rrbracket$ for each $a \in [n]$. Then, from the clause in this family containing $p_{i,j} \neq a$, we weaken over all values of the successor pointer $s_{i-1,a}$, obtaining the family of clauses

$$\begin{aligned} \mathcal{A}_i &= \{\llbracket I_{i,j,k} \vee p_{i,j} \neq a \vee s_{i-1,a} \neq b \rrbracket \mid j, k, a, b \in [n]\} \\ &= \{\llbracket s_{i,j} \neq k \vee p_{i+1,k} \neq j \vee p_{i,j} \neq a \vee s_{i-1,a} \neq b \rrbracket \mid j, k, a, b \in [n]\}. \end{aligned}$$

Partition this family of clauses into two sets as follows. Define

$$\mathcal{T}_i = \{\llbracket s_{i,j} \neq k \vee p_{i+1,k} \neq j \vee p_{i,j} \neq a \vee s_{i-1,a} \neq j \rrbracket \mid j, k, a \in [n]\},$$

which is the subfamily of clauses in \mathcal{A}_i that have $b = j$, and let $\mathcal{J}_i = \mathcal{A}_i \setminus \mathcal{T}_i$ denote the subfamily where $b \neq j$. Next, we show how to use \mathcal{T}_i , along with some clauses in SOPL_n , to deduce \mathcal{I}_{i-1} in width $O(\log n)$ and polynomial size. The clauses \mathcal{J}_i are “junk” clauses that are maintained for the rest of the proof and output along with the bottom clause \perp in the final configuration.

To do this, we exploit the reversibility of RevRes and show how to deduce from \mathcal{I}_{i-1} the collection $\mathcal{T}_i \cup \mathcal{F}_i$ using the reversible weakening rule, where \mathcal{F}_i is a polynomial-size set of clauses all from SOPL_n . By running this proof in reverse and connecting it with the proof described above we prove \mathcal{I}_{i-1} from \mathcal{I}_i , and we can add the clauses \mathcal{F}_i to the initial configuration of the RevRes proof.

This proof is very similar to the proof of \mathcal{A}_i from \mathcal{I}_i . Starting from an arbitrary clause $I_{i-1,a,j} \in \mathcal{I}_{i-1}$, we apply [Lemma 10](#) to weaken the clause on all possible values of the successor pointer $s_{i,j}$, obtaining $\llbracket I_{i-1,a,j} \vee s_{i,j} \neq k \rrbracket$ for all $k \in [n]$. Then, starting from the clause containing $s_{i,j} \neq k$, we weaken on all values of $p_{i+1,k}$, obtaining the family

$$\mathcal{B}_i = \{\llbracket I_{i-1,a,j} \vee s_{i,j} \neq k \vee p_{i+1,k} \neq b \rrbracket \mid j, k, a, b \in [n]\}$$

$$= \{ \llbracket s_{i-1,a} \neq j \vee p_{i,j} \neq a \vee s_{i,j} \neq k \vee p_{i+1,k} \neq b \rrbracket \mid j, k, a, b \in [n] \}$$

We again partition into two sets. The first is, of course, \mathcal{T}_i , which is the case where $j = b$ in \mathcal{B}_i . The second set is $\mathcal{F}_i = \mathcal{B}_i \setminus \mathcal{T}_i$, which is the case where $j \neq b$, and observe that every clause in \mathcal{F}_i is a *no proper sink* clause from SOPL_n .

We can now finish the proof of the claim. Starting from $\mathcal{I}_i \cup \mathcal{F}_i$, use the clauses in \mathcal{I}_i to deduce the set of clauses $\mathcal{T}_i \cup \mathcal{F}_i \cup \mathcal{J}_i$. Then, run the proof deducing $\mathcal{T}_i \cup \mathcal{F}_i$ from \mathcal{I}_{i-1} in reverse to finally deduce $\mathcal{I}_{i-1} \cup \mathcal{J}_i$. The total proof has polynomial size and $O(\log n)$ -width, and therefore the claim is proved. \square

To modify the proof for EOPL_n and RevResT , we make the following changes. First, we observe that all clauses in \mathcal{I}_1 that come from $I_{1,a}$ for $a \neq 1$ are proper source clauses from EOPL_n . All other clauses in the above proof that occur in the final line come from sets of the form \mathcal{J}_i in the proof of the above claim. However, just like the clauses \mathcal{F}_i are proper sink clauses from SOPL_n , the clauses in \mathcal{J}_i are exactly proper source clauses from the EOPL_n . This completes the proof. \square

It remains to modify the previous proof in order to accommodate decision-tree reductions to SOPL and EOPL . To do this we mimic the previous proof, but replace the construction of the sets of clauses in the proof with appropriate queries to the decision trees (which RevRes can simulate) in the reduction.

Before we prove the theorem we introduce some helpful notation for manipulating decision trees. If T is a decision tree then $\mathcal{P}(T)$ is the set of root-to-leaf paths in T . If o is an output (i.e. leaf label) of T , then define $\mathcal{P}_o(T)$ to be the set of root-to-leaf paths in T that output o . Given any path $P \in \mathcal{P}(T)$, let $C_P := \bigvee_{\ell \in P} \neg \ell$ be the negation of the literals along P ; so, $C_P(x) = 1$ iff P is not followed when T is evaluated on x . We also need an appropriate modification of [Lemma 10](#) to arbitrary decision trees, which we prove next.

Lemma 11. *Let C be a width- k clause, and let T be a depth- d decision tree querying a set of variables disjoint from C . Using the reversible weakening rule we can prove, from C , the set of clauses $\{C \vee C_P \mid P \in \mathcal{P}(T)\}$ in width $d+k$ and size at most 2^d . Conversely, from the above set of clauses we can prove C using the reversible resolution rule in the same size and width.*

Proof. This proof is essentially the same as in [Lemma 10](#). Now, starting from C , apply the reversible weakening rule on the first variable x_i queried in the decision tree T to derive the clauses $\{C \vee x_i, C \vee \bar{x}_i\}$. From there we can continue to apply the reversible weakening rule to simulate the queries of the decision tree. For instance, if after the decision tree learns $x_i = 0$ it queries x_j , we apply the reversible weakening rule to x_i to obtain $x_i \vee x_j, x_i \vee \bar{x}_j$. Continuing in this manner we can derive all clauses $C \vee C_P$ for $P \in \mathcal{P}(T)$, and running the proof in reverse yields the lemma. \square

Theorem 11. *Let F be an unsatisfiable CNF formula. If there is a depth- d SOPL_L -formulation (EOPL_L -formulation, resp.) of $S(F)$ then there is a RevRes refutation (with terminals, resp.) of F with width $O(d)$ and size $L^{O(1)}2^{O(d)}$.*

Proof. We follow the proof of [Theorem 10](#) and focus on the case of SOPL . Assume $F = C_1 \wedge \dots \wedge C_m$ is defined on n variables x_1, \dots, x_n . In this proof we think of CNF formulas and sets of clauses interchangeably. In the SOPL_L -formulation of $S(F)$ we have functions

$$s_{i,j} : \{0, 1\}^n \rightarrow [L] \cup \{\text{null}\}, \quad p_{i,j} : \{0, 1\}^n \rightarrow [L] \cup \{\text{null}\}, \quad g_{i,j} : \{0, 1\}^n \rightarrow [m]$$

computing successors, predecessors, and solutions for each internal node, and we identify each function with the depth- d decision tree computing it.

For each $(i, j) \in [L - 1] \times [L]$ consider the CNF formula

$$I_{i,j,k}(x) = \llbracket s_{i,j}(x) \neq k \vee p_{i+1,k}(x) \neq j \rrbracket.$$

In other words, $I_{i,j,k}$ is the analogue of the clause using the same notation from the proof of [Theorem 10](#). We can use the decision trees for $s_{i,j}$ and $p_{i,j}$ to encode $I_{i,j,k}$ as a CNF formula explicitly. To do this, define the decision tree $T_{i,j}$ as follows: take the decision tree $s_{i,j}$ and at each leaf labelled k , simulate the decision tree $p_{i+1,k}$ (skipping queries to variables already made) to obtain an output a , and then output the pair (k, a) . With this decision tree we can define $I_{i,j,k} = \{C_P \mid P \in \mathcal{P}_{(k,j)}(T_{i,j})\}$. As in the proof of [Theorem 10](#), define

$$I_{i,j} := \bigcup_{k=1}^n I_{i,j,k}, \quad \mathcal{I}_i := \bigcup_{j=1}^n I_{i,j},$$

where we recall that we consider CNFs and sets of clauses interchangeably. When $i = L$, then for any $j \in [L]$ define the decision tree $T_{L,j}$ that simulates the decision tree $s_{L,j}$ and outputs 1 if (L, j) is active and 0 otherwise. With this we define $I_{L,j} = \{C_P \mid P \in \mathcal{P}_1(T_{L,j})\}$, and similarly define $\mathcal{I}_L = \bigcup_{j=1}^L I_{L,j}$. In this notation, the set of clauses \mathcal{I}_i again encodes “every node on layer i is inactive”, where now the activity of a node is determined by the underlying decision trees in the formulation.

The main step in this theorem is the following claim.

Claim 5. *For any $i \in \{2, \dots, L\}$, there is a size $L^{O(1)}2^{O(d)}$, $O(d)$ -width RevRes proof of \mathcal{I}_{i-1} from \mathcal{I}_i and a collection of weakenings of clauses from F .*

First we use the claim to finish the proof of the theorem. We begin by deriving from F the clauses \mathcal{I}_L (let us briefly postpone this argument), and then apply the claim $L - 1$ times to derive \mathcal{I}_1 . Let $\mathcal{Q} = \bigcup_{k \neq 0} \mathcal{P}_{(k,1)}(T_{1,1})$ be the set of paths of $T_{1,1}$ that end in a leaf labelled with $(k, 1)$ for some $k \neq 0$, and let $\mathcal{R} = \mathcal{P}(T_{1,1}) \setminus \mathcal{Q}$. Observe that $\mathcal{I}_{1,1} \subseteq \mathcal{I}_1$ is, by definition, the set of clauses $\{C_P \mid P \in \mathcal{Q}\}$.

Consider any path $P \in \mathcal{R}$, and note that P ends in a leaf labelled with (k, a) where either $k = 0$ or $a \neq 1$. Each leaf witnesses that the distinguished node $(1, 1)$ is inactive, and so we can then simulate the decision tree $g_{1,1}$ and learn a solution of $S(F)$. Therefore, for every path $P' \in \mathcal{P}(g_{1,1})$ the clause $C_P \vee C_{P'}$ is either a weakening of a clause in F , or, is trivially true if it contains both a literal and its negation. Therefore, by applying [Lemma 11](#) we can deduce the clause C_P from weakenings of clauses in F in size $2^{O(d)}$ and width $O(d)$. Applying this argument for every $P \in \mathcal{R}$ allows us to deduce the clauses $\{C_P \mid P \in \mathcal{R}\}$. We have now deduced all the clauses $\{C_P \mid P \in \mathcal{P}(T_{1,1})\}$, and so applying [Lemma 11](#) to all of these clauses allows us to deduce \perp .

Let us now describe how to derive from F the clauses

$$\mathcal{I}_L = \{\llbracket (L, j) \text{ is inactive} \rrbracket \mid j \in [L]\} = \bigcup_{j=1}^n \{C_P \mid P \in \mathcal{P}_1(T_{L,j})\}.$$

For any $j \in [L]$ consider the following decision tree $T'_{L,j}$: first run the decision tree $T_{L,j}$ that checks if (L, j) is active and then, if (L, j) is active, simulate the decision tree $g_{L,j}$ to find a solution to $S(F)$. It follows that for any $P \in \mathcal{P}_1(T_{L,j})$ and any $P' \in \mathcal{P}(g_{L,j})$ the clause $C_P \vee C_{P'}$ is a weakening of a clause of F or is trivially true. We can therefore deduce C_P from weakenings of clauses of F using [Lemma 11](#), and repeating this argument for every $j \in [L]$ and every $P \in \mathcal{P}_j$ we can derive every clause in \mathcal{I}_L . So, all that remains is to prove the claim.

Proof of Claim. The proof of this claim is modelled on the proof of the similar claim from the previous theorem. We again do the general case where $i \leq L - 1$; the case where $i = L$ proceeds similarly. Consider the set of clauses \mathcal{I}_i and \mathcal{I}_{i-1} . Our first goal is to derive the analogue of the set \mathcal{A}_i in the proof of [Claim 4](#).

Let $j \in [L]$ be arbitrary and consider any clause $C \in \mathcal{I}_{i,j}$. By definition, there is a $k \neq 0$ such that $C = C_P$ for some $P \in \mathcal{P}_{(k,j)}(T_{i,j})$. Starting from C in the proof apply [Lemma 11](#) to the decision tree $p_{i,j}$ to derive a set of clauses, each of the form $C \vee C_P$, where $P \in \mathcal{P}(p_{i,j})$. Then, for every $a \in [L]$ and any $P' \in \mathcal{P}_a(p_{i,j})$, apply [Lemma 11](#) again to $C \vee C_P$ and the decision tree $s_{i-1,a}$ to obtain $C \vee C_P \vee C_{P'}$ for every $P' \in \mathcal{P}(s_{i-1,a})$. Performing this procedure for all $C \in \mathcal{I}_i$ yields

$$\begin{aligned} \mathcal{A}_i &:= \{ \llbracket s_{i,j} \neq k \vee p_{i+1,k} \neq j \vee p_{i,j} \neq a \vee s_{i-1,a} \neq b \rrbracket \mid j, k, a, b \in [n] \} \\ &= \{ C \vee C_P \vee C_{P'} \mid a \in [L], P \in \mathcal{P}_a(p_{i,j}), P' \in \mathcal{P}(s_{i-1,a}) \}. \end{aligned}$$

We partition \mathcal{A}_i into two sets: the clauses in \mathcal{T}_i where $b = j$, and the clauses in $\mathcal{J}_i = \mathcal{A}_i \setminus \mathcal{T}_i$.

Now, as in the proof of [Claim 4](#), we use \mathcal{T}_i along with some clauses in F to deduce \mathcal{I}_{i-1} , and we again will exploit the reversibility of RevRes to do so. Namely, starting from \mathcal{I}_{i-1} we deduce $\mathcal{T}_i \cup \mathcal{F}_i$, where \mathcal{F}_i is a collection of (weakenings of) clauses from F , and we can then just run the proof in reverse.

Let D be any clause in \mathcal{I}_{i-1} , and note that there is a $j \in [L]$ such that $D = C_P$ for some $P \in \mathcal{P}_{(j,a)}(T_{i-1,a})$. Starting from D , apply [Lemma 11](#) with the decision tree $T_{i,j}$ to obtain a collection of clauses of the form $D \vee C_{P'}$ where $P' \in \mathcal{P}(T_{i,j})$. Let (k, b) be the output of the decision tree $T_{i,j}$ on the path P' . If $b = j$, then the clause $D \vee C_{P'}$ belongs to \mathcal{T}_i . Moreover, if we repeat this argument for all $D \in \mathcal{I}_{i-1}$ then the collection of all such clauses obtained is *exactly* \mathcal{T}_i . This is because from \mathcal{I}_i , the collection of clauses \mathcal{T}_i was obtained by starting from all clauses at leaves of $T_{i,j}$ labelled with (k, j) and then querying $p_{i,j}$ and $s_{i-1,a}$; here, we have performed the exact same queries except we have reversed the order in which we simulated the decision trees $p_{i,j}$ and $s_{i-1,a}$.

On the other hand, if $b \neq j$, then the literals queried on the paths $P \in \mathcal{P}_{(j,a)}(T_{i-1,a})$ and $P' \in \mathcal{P}_{(k,b)}(T_{i,j})$ together witness that the node (i, j) is a proper sink node, and thus is a solution to the SOPL problem. Therefore, at the end of the path $P \cup P'$ we can run the decision tree $g_{i,j}$ to determine a solution to $S(F)$. This means that if $P'' \in \mathcal{P}(g_{i,j})$ is any root-to-leaf path in $g_{i,j}$, then the clause $D \vee C_{P'} \vee C_{P''} = C_P \vee C_{P'} \vee C_{P''}$ must be a weakening of a clause in F (or, again, is trivially true). Let \mathcal{F}_i denote the set of all of these weakenings of clauses of F , obtained by running the above procedure for every clause $D \in \mathcal{I}_{i-1}$. We have therefore shown that from \mathcal{I}_{i-1} we can derive $\mathcal{T}_i \cup \mathcal{F}_i$.

To finish the proof of the claim, we start with the clauses in $\mathcal{I}_i \cup \mathcal{F}_i$, deduce $\mathcal{T}_i \cup \mathcal{J}_i$ from \mathcal{I}_i to obtain the clauses $\mathcal{T}_i \cup \mathcal{J}_i \cup \mathcal{F}_i$, and then deduce \mathcal{I}_{i-1} from $\mathcal{T}_i \cup \mathcal{F}_i$. This yields the clauses $\mathcal{I}_{i-1} \cup \mathcal{J}_i$, and all of these steps required size $L^{O(1)}2^{O(d)}$ and width at most $O(d)$, completing the proof of the claim and the theorem. \square

The above proof can be modified to capture EOPL in the same manner as the proof of [Theorem 10](#). In particular, we can argue via the same techniques that the “junk” clauses in \mathcal{J}_i and the clauses in $\mathcal{I}_1 \setminus \mathcal{I}_{1,1}$ each encode violations of the “no proper source” constraints of EOPL, and thus can be used to deduce weakenings of clauses in F by querying the appropriate solution decision trees $g_{i,j}$. We omit the details. \square

7 Intersection Theorems

We can now finally prove [Theorem 6](#), our intersection theorem for Reversible Resolution. To prove the theorem we use the collapse theorems $\text{SOPL} = \text{PLS} \cap \text{PPADS}$ and $\text{EOPL} = \text{PLS} \cap \text{PPAD}$ [[GHJ⁺22](#)].

In particular, examining the proofs of the collapse theorems from [GHJ⁺22], we can extract the following black-box analogues.

Theorem 12. *Let $R \subseteq \{0, 1\}^n \times O$ be a total search problem, and suppose that there is a depth- d_1 , SoD_{s_1} -formulation of R and a depth- d_2 , SoL_{s_2} -formulation of R . Then there is a depth $O(d)$ SoPL_{s_3} -formulation of R where $d = \max\{d_1, d_2\}$ and $s = \max\{s_1, s_2\}$. \square*

Theorem 13. *Let $R \subseteq \{0, 1\}^n \times O$ be a total search problem, and suppose that there is a depth- d_1 , SoD_{s_1} -formulation of R and a depth- d_2 , EoL_{s_2} -formulation of R . Then there is a depth $O(d)$ EoPL_{s_3} -formulation of R where $d = \max\{d_1, d_2\}$ and $s = \max\{s_1, s_2\}$. \square*

Theorem 6 is now an immediate corollary of the next theorem.

Theorem 14. *Let F be an unsatisfiable CNF formula. Let d_1, d_2, s_1, s_2 be positive integers and let $d = \max\{d_1, d_2\}$ and $s = \max\{s_1, s_2\}$.*

- *If there is a width- d_1 , size- s_1 Resolution proof and a degree- d_2 , size- s_2 unary Sherali–Adams proof of F then there is width $O(d)$ and size $s^{O(1)}2^{O(d)}$ RevRes proof of F .*
- *If there is a width- d_1 , size- s_1 Resolution proof and a degree- d_2 , size- s_2 unary Nullstellensatz proof of F then there is width $O(d)$ and size $s^{O(1)}2^{O(d)}$ RevResT proof of F .*

In particular, $\text{RevRes}(F) = \Theta(\text{Res}(F) + \text{uSA}(F))$ and $\text{RevResT}(F) = \Theta(\text{Res}(F) + \text{uSA}(F))$.

Proof. Since RevRes can be efficiently simulated by both Resolution and unary Sherali–Adams we have $\text{Res}(F) = O(\text{RevRes}(F))$ and $\text{uSA}(F) = O(\text{RevRes}(F))$. For the converse direction, suppose that we have a width- d_1 , size- s_1 Resolution proof and a degree- d_2 , size- s_2 unary Sherali–Adams proof. By [Kam20, Theorem 8.18] there is a depth- $O(d_1)$ $\text{SoD}_{O(s_1)}$ -formulation of $S(F)$ and by Theorem 8, there is a depth- $O(d_2)$ $\text{EoL}_{O(s_2)}$ -formulation for $S(F)$. Applying the above collapse theorem, this implies that there is a depth- $O(d)$ SoPL_{s_3} -formulation of $S(F)$, where $d = \max\{d_1, d_2\}$ and $s = \max\{s_1, s_2\}$. Finally, applying Theorem 9, we obtain a RevRes proof of F with width $O(d)$ and size $s^{O(1)}2^{O(d)}$. We therefore have

$$\text{RevRes}(F) = O(d + \log s) = O(d_1 + d_2 + \log s_1 + \log s_2) = O(\text{Res}(F) + \text{uSA}(F)).$$

A similar proof using Theorem 7 instead yields the characterisation of RevResT. \square

8 Two Further Separations

In this section we prove Theorems 4 and 5, restated below.

Theorem 4. $\text{PLS}^{dt} \not\subseteq \text{PPP}^{dt}$.

Theorem 5. $\text{EOPL}^{dt} \not\subseteq \text{UEOPL}^{dt}$.

The proofs of these theorems rely on a “glueing” technique that was implicitly used in [BCE⁺98] and which we make more explicit in this paper. We use the glueing technique as a tool to alleviate the lack of good proof systems characterizing PPP and UEOPL. In particular, The glueing technique reduces the separation in Theorem 4 to the easier separation $\text{PLS}^{dt} \not\subseteq \text{PPADS}^{dt}$, which we already proved in Corollary 1 and Theorem 5 uses the glueing technique together with a query lower bound for EOPL from [HY20]. This glueing technique was also recently generalized by Jain, Li, Robere and Xun [JLRX24] to prove lower bounds for classes above PPP corresponding to the generalized pigeonhole principles.

8.1 Glueability

Let $R = (R_n)$, $R_n \subseteq \{0, 1\}^n \times O_n$, be a TFNP^{dt} problem. We consider *partial assignments* $x \in \{0, 1, *\}^n$ that define partial inputs to R_n . An index i with $x_i = *$ is interpreted as a boolean variable whose value is not yet assigned. The *size* of a partial assignment is its number of non- $*$ bits. We say that two partial assignments $x, y \in \{0, 1, *\}^n$ are *consistent* if x and y agree on their non- $*$ bits. If x and y are consistent, we can form the partial assignment $x \cup y$ that assigns values to all variables assigned values in x or y . We further say that x is *witnessing* if there exists some solution $o \in O_n$ such that for any $y \in \{0, 1\}^n$ consistent with x we have $o \in R_n(y)$.

Definition 8 (Glueable sets of assignments). A set of partial assignments $P \subseteq \{0, 1, *\}^n$ is *k-glueable* if for each non-witnessing and consistent $p, p' \in P$, their union $p \cup p'$ is non-witnessing, and moreover, if we restrict R_n by the assignment $p \cup p'$, the resulting search problem $(R_n \upharpoonright p \cup p')$ has decision tree complexity greater than k .

This and following definitions are mostly motivated by their use in [Lemma 12](#) and [Lemma 14](#). For instance, in [Lemma 12](#) we consider P to be the set of all partial assignments obtained by collecting leaves pointing to a particular hole in the PPP^{dt}-reduction. The main idea is that the glueability property of P then allows to disambiguate between pigeons to find which (if any) is mapping to the particular hole.

Definition 9 (Completions). Let $x \in \{0, 1, *\}^n$ be a partial assignment and T a decision tree over $\{0, 1\}^n$. The completion $C(T, x)$ of x by T is the set obtained by collecting all the partial assignments corresponding to leaves of T that are consistent with x and taking their union with x . That is, $C(T, x) := \{x \cup p : p \text{ is a leaf of } T \text{ consistent with } x\}$.

Definition 10 (Glueable problem). Let $f: \mathbb{N} \rightarrow \mathbb{N}$ be a function. We say R is *f(k)-glueable* if any set $P \subseteq \{0, 1, *\}^n$ of partial assignments of size at most k , where $k \leq \text{poly}(\log n)$, can be completed by decision trees of depth at most $f(k)$ such that the union of the completions is *k-glueable*. That is, if there exists for each $x \in P$, some decision tree T_x such that $\cup_{x \in P} C(T_x, x)$ is *k-glueable*. We further say that R is *glueable* if it is $\text{poly}(k)$ -glueable.

For example, it is implicit in [[BCE⁺98](#), §3.1] that the PPA^{dt}-complete problem LONELY (given a matching of an odd number of nodes, find an isolated node) is $O(k)$ -glueable. In the case of LONELY, if $x \in \{0, 1, *\}^n$ asserts that node u points to node v , then T_x queries the pointing node for v so that a solution is immediately witnessed if u is isolated. We will shortly prove that SoD and EoPL are glueable, too. In what follows, we slightly depart from the above notation and also consider pointer-like partial assignments (as opposed to assignments over $\{0, 1, *\}$ only). Those are treated naturally; for instance, we can assume that reductions are constrained to query either all or no bits corresponding to a pointer.

8.2 PLS^{dt} $\not\subseteq$ PPP^{dt}

We introduce for convenience the REVERSIBLE-PIGEON problem, which is a variant of PIGEON where a reverse pointer is provided for each hole.

REVERSIBLE-PIGEON (RPIGEON_n). This problem is the same as PIGEON except that we are also given *reverse* pointers $p_u \in [n] \cup \{\text{null}\}$ for each hole $u \in [n - 1]$. The goal is to output any solution of PIGEON or

2. $u \in [n]$ such that $p_{s_u} \neq u$. *(successor/predecessor mismatch)*

This problem is known to be PPADS^{dt} -complete (see, e.g., [GHJ+22, Lemma 1]) so that $\text{RPIGEON}^{dt} = \text{PPADS}^{dt}$. The following key lemma is implicit in [BCE+98, §3.1].

Lemma 12. *If $R \in \text{PPP}^{dt}$ and R is glueable, then $R \in \text{PPADS}^{dt}$.*

Proof. Fix a PIGEON_m -formulation $(f_i, g_{i,i'})_{i,i' \in [m]}$ of R_n that witnesses $R \in \text{PPP}^{dt}$ and let $(T_i, S_{i,i'})$ be decision trees of depth $k = \text{poly}(\log n)$ implementing this reduction. Since R is glueable, it is possible to complete the root-to-leaf paths of each T_i to get a reduction $(T'_i, S_{i,i'})$ of depth $d = \text{poly}(\log n)$ for which the set $P = \cup_{i \in [m]} \text{leaves}(T'_i)$ is k -glueable. (Note that each $S_{i,i'}$ remains unchanged and has depth at most k .) We show how to construct decision trees $(H_j)_{j \in [m]}$ of depth $\leq d^2$ that compute reverse pointers for each hole of the PIGEON_m instance. We start with the following claim.

Claim 6. *Suppose $p \in \text{leaves}(T'_i)$ and $p' \in \text{leaves}(T'_{i'})$ are distinct leaves that are both non-witnessing and labelled with the same hole. Then p and p' are inconsistent.*

Proof. If $i = i'$, then the claim is true since any two distinct leaves of the same tree are inconsistent. Suppose $i \neq i'$ and suppose for contradiction that p and p' are consistent. Then, (i, i') is a valid solution to the PIGEON_m instance $(T'_1(z), T'_2(z), \dots, T'_m(z))$ for any $z \in \{0, 1\}^n$ extending $p \cup p'$. By correctness of the reduction, this further implies that $S_{i,i'}$ can solve $(R_n \upharpoonright p \cup p')$ with at most k queries—but this contradicts the fact that P is k -glueable. \square

Let us write $P_j \subseteq P$ for the set of all non-witnessing partial assignment corresponding to leaves labelled with hole j . The predecessor tree H_j computes as follows. Pick an arbitrary leaf $p \in P_j$ and query all the variables contained in p . At every leaf $x \in \{0, 1, *\}^n$ of the current version of H_j , the next step depends on the set of x -consistent assignments $P_j^x = \{p \in P_j : p \text{ consistent with } x\}$.

1. If $|P_j^x| = 0$, then output label null.
2. If $|P_j^x| = 1$, then output the *unique* $i \in [m]$ (by Claim 6) such that $P_j^x \cap \text{leaves}(T'_i) \neq \emptyset$.
3. If $|P_j^x| \geq 2$, pick an arbitrary $p \in P_j^x$ and recurse by querying its variables, etc.

Note that each predecessor tree H_j has depth at most d^2 : by pairwise inconsistency of P_j , at most d paths are queried each of depth at most d . To complete the RPIGEON_m -formulation of R_n , it remains to specify decision trees $(S_i)_{i \in [m]}$ that transform RPIGEON_m -solutions of type (2) into R_n -solutions. Indeed, suppose $T'_i(z) = j$ but $H_j(z) \neq i$ for some input z to R_n . Then, since H_j decides unambiguously which non-witnessing assignment in P_j is consistent with z (if any), it must be the case that the leaf outputting $T'_i(z) = j$ is not in P_j , which means that it is witnessing. Thus, $S_i(z)$ simply runs $T'_i(z)$ and an R_n -solution must be witnessed during its execution. \square

We note that the method used to disambiguate pigeons in Lemma 12 is common. For instance, it is key to prove the folklore certificate-to-query result $D(f) \leq C^1(f) \cdot C^0(f)$ for boolean functions f . To show Theorem 4, the last missing piece is to show that SoD is glueable. Indeed, if $\text{SoD} \in \text{PPP}^{dt}$, then Lemma 12 would imply that $\text{PLS}^{dt} \subseteq \text{PPADS}^{dt}$, which contradicts Corollary 1. We show that SoD is glueable in Lemma 13 below.

For technical convenience, we consider here a minor variation of how we encode the successor pointers in the input to SoD . We let the input consist of successor pointers $s_u \in [n]$ for each grid node $u \in [n] \times [n]$ as well as an “active” bit $a_u \in \{0, 1\}$, where $a_u = 0$ means that u has a null pointer. This is merely a different way to encode null successors, and indeed, there is a trivial reduction to and from the original SoD problem. The advantage of this new encoding is that it allows for querying the activity of a node *without* querying its successor. This simplifies the completion process in the proof below.

Lemma 13. *SoD is glueable.*

Proof. We show that SoD_n is $O(k)$ -glueable. Fix some partial SoD_n -assignment $x = (s, a)$ of size $k = \text{poly}(\log n)$, that is, $s_u \in [n] \cup \{*\}$ and $a_u \in \{0, 1, *\}$ for each grid node $u \in [n] \times [n]$. The decision tree T completing x starts by checking whether x queries any active node below row $n - k - 1$. If yes, T picks any one such active node and follows the successor path until a sink is found, making the completion witnessing. Note that this step incurs at most $O(k)$ queries. Finally, T ensures that any successor query in x is followed by a query to the active bit of the successor. This costs at most $O(k)$ further queries.

Let P be an arbitrary set of partial assignments each of size at most $k = \text{poly}(\log n)$ and let P' be its completion with respect to the procedure defined above. We first show that P' is k -glueable. Pick any two non-witnessing and consistent $p, p' \in P'$ and suppose toward contradiction that their union $p \cup p'$ is witnessing. If it reveals a SoD solution u of type (1) or (2), then it must be that one of p and p' checks for the active bit of u : a contradiction with the fact that p and p' are non-witnessing. On the other hand, if $p \cup p'$ reveals a solution u of type (3), then it must be that one of p and p' checks for the successor s_u of u , but the completion T forces this check to be followed by a query to the active bit of s_u , making one of the initial partial assignments witnessing as well. Hence $p \cup p'$ is non-witnessing.

We finally argue that $(R_n \upharpoonright p \cup p')$ has query complexity greater than k by describing an adversary that can fool any further k queries to $p \cup p'$ without witnessing a solution. Recall that $p \cup p'$ makes no queries to nodes below row $n - k - 1$. The adversary answers queries as follows. If the successor pointer of an active node is queried, then we answer with a pointer to any unqueried node on the next row and make it active (there always exists one as $k \ll n$). If a node u is queried that is not the successor of any node, we make u inactive ($a_u = 0$ and s_u is arbitrary). This scheme ensures that a solution can only lie on the very last row n , which is not reachable in k queries starting from row $n - k - 1$. \square

8.3 $\text{EOPL}^{dt} \not\subseteq \text{UEOPL}^{dt}$

We prove [Theorem 5](#) using a similar plan as in [Section 8.2](#) above. Namely, we first show ([Lemma 14](#)) that if we have a problem $R \in \text{UEOPL}^{dt}$ that is glueable, then in fact $R \in \text{FP}^{dt}$, that is, R_n admits a shallow decision tree solving it. Second, we show ([Lemma 15](#)) that EOPL_n is glueable. The combination of these two lemmas implies that if $\text{EOPL}^{dt} \subseteq \text{UEOPL}^{dt}$, then $\text{EOPL}^{dt} = \text{FP}^{dt}$. But it is known from prior work [[HY20](#)] (building on [[Ald83](#), [Zha09](#)]) that $\text{EOPL}^{dt} \neq \text{FP}^{dt}$. This proves [Theorem 5](#).

It remains to prove [Lemmas 14](#) and [15](#).

Lemma 14. *If $R \in \text{UEOPL}^{dt}$ and R is glueable, then $R \in \text{FP}^{dt}$.*

Proof. Fix an UEOPL_m -formulation $(f_u, g_{u,u'})_{u,u' \in [m] \times [m]}$ of R_n that witnesses $R \in \text{UEOPL}^{dt}$ and let $(T_u, S_{u,u'})$ be decision trees of depth $k = \text{poly}(\log n)$ implementing this reduction. Note that the leaves of each T_u are labelled by a successor and predecessor pointers in $[m]$. At the cost of doubling the depth of each T_u , we may assume that each leaf is additionally labelled with an “activity” bit, which can be computed by appending to each leaf labelled with successor v the decision tree T_v . Since R is glueable, it is possible to further complete the leaves of each T_u to get a reduction $(T'_u, S_{u,u'})$ of depth $d = \text{poly}(\log n)$ for which the set of leaves $P = \cup_{u \in [m] \times [m]} \text{leaves}(T'_u)$ is k -glueable and each leaf label carries the aforementioned activity bit. Let us say that a node u is *good* for input z if the leaf reached by $T'_u(z)$ is non-witnessing and u is active.

Claim 7. *For every input z , there is at most one good node on each row.*

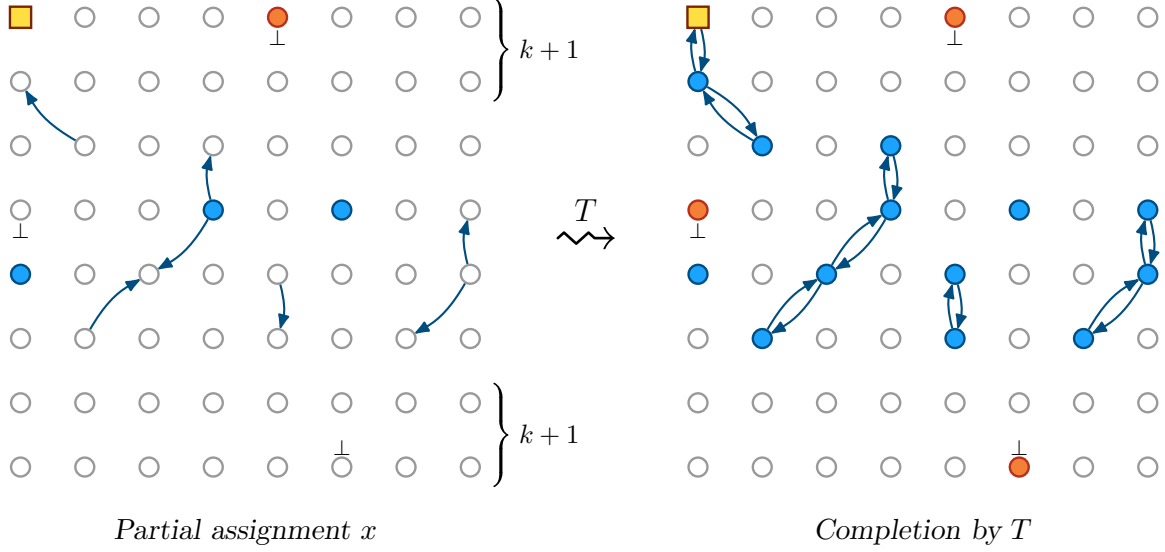


Figure 8: Example of a non-witnessing completion. A node is blue if $a_u = 1$, orange if $a_u = 0$, and white if $a_u = *$ is not queried. The symbol \perp indicates a null pointer. The first and last $k + 1$ rows contain no active nodes, besides those lying on the path starting at the distinguished node $(1, 1)$.

Proof. Fix a row $j \in [m]$ and suppose for the sake of contradiction that the j -th row contains two good nodes u' and u on some input z . Let $p \in \text{leaves}(T'_u)$ and $p' \in \text{leaves}(T'_{u'})$ be the leaves reached on input z . Then p and p' are a pair of non-witnessing and consistent assignments. Thus, (u, u') is a solution to UEOPL_m on any input that extends $p \cup p'$. Hence the depth- k decision tree $S_{u, u'}$ solves the search problem $(R_n \upharpoonright p \cup p')$. But this contradicts the fact that P is k -glueable. \square

Using this claim similarly as in the proof of [Lemma 12](#), we can construct, for each row $j \in [m]$, a decision tree A_j of depth $\leq d^2$ that computes the column-index of a good node on row j or outputs null if the row contains no good node. The main argument is again the disambiguation trick.

We can now design an efficient decision tree for R_n : At the cost of running $O(\log m) = \text{poly}(\log n)$ of the A_j trees, perform a binary search over the m rows to find a good node u on row j such that the next row $j + 1$ contains no good nodes. This means that either (i) the successor of u is inactive, in which case we have found a solution to UEOPL_m and we can use the S -trees to find a solution to R_n , or (ii) the successor u' of u is active and $T'_{u'}(z)$ is witnessing, which solves R_n . \square

We next show that an EOPL^{dt} -complete problem is glueable. Instead of working with EOPL_n , it is convenient again to vary the input encoding. We define EOPL^* as a version of EOPL where in addition to successor/predecessor pointers, we are also given an “activity” bit.

END-OF-POTENTIAL-LINE* (EOPL_n^*). In addition to predecessor/successor pointers, each $u \in [n] \times [n]$ has an activity indicator bit $a_u \in \{0, 1\}$. We add the following solutions to EOPL :

5. u , if u 's activity does not match a_u . *(active bit mismatch)*
6. u , if $a_v = 0$ and $(s_v \neq \text{null or } p_v \neq \text{null})$. *(inactive node with a pointer)*
7. u , if $a_v = 1$ and $(s_v = \text{null or } p_v = \text{null})$. *(active node with a null-pointer)*

Note that EOPL^* is efficiently reducible to and from EOPL , so that EOPL^* is EOPL^{dt} -complete.

Lemma 15. EOPL^* is glueable.

Proof. We show that EoPL^* is $O(k)$ -glueable. Fix some partial EoPL^* -assignment $x = (p, s, a)$ of size $k = \text{poly}(\log n)$. The tree T that completes x proceeds as follows. We start by querying all variables assigned in x . Then we iterate each of the following steps until a solution is found or no further queries are made.

1. *Always query activity bits and reverse pointers.* If we have queried a null-pointer $s_v = \text{null}$ or $p_v = \text{null}$, then we also query the activity bit a_v . This activity bit is $a_v = 0$ unless we have found a solution of type (7).

Moreover, if we have queried a non-null pointer $s_u = v$ (resp. $p_u = v$), then we also query the bits a_u, a_v and the pointer p_v (resp. s_v). Note that both activity bits must be 1 and the reverse pointer must point back, $p_v = u$ (resp. $s_v = u$), as otherwise we can find a solution by making a couple more queries. Indeed, if $a_u = 0$, then there is a solution of type (6). If $a_u = 1$ and $a_v = 0$, then we can find a solution by determining the activity of v : either v is active, which is a mismatch with $a_v = 0$ (type (5)), or v is inactive, which creates a sink. Finally, if $a_u = 1$ and $p_v \neq u$, then u is inactive, which is a mismatch with $a_u = 1$ (type (5)).

2. *Follow the distinguished path.* Follow the successor path starting at the distinguished source node $(1, 1)$ until some node on row $k + 1$ is reached or a sink is found.
3. *Follow early paths.* If we have queried $a_u = 1$ for some node u in the first $k + 1$ rows that does not lie on the path discovered in [Item 2](#), then we follow u 's predecessor path until a solution is found.
4. *Follow late paths.* If we have queried $a_u = 1$ for some node u in the last $k + 1$ rows, then we follow u 's successor path until a solution is found.

This completion adds at most $O(k)$ queries to x . An example of a completion that is non-witnessing is given in [Figure 8](#). It is straightforward to argue that the resulting set of completed assignments is k -glueable using an adversary strategy similar to the one described in the proof of [Lemma 13](#). \square

A Appendix: Coefficient Size in Algebraic Proofs

In this appendix we show that if there are low-degree Nullstellensatz and Sherali–Adams refutations over \mathbb{Z} , then the coefficients in the refutations will also be not too large in magnitude. In particular, if the degree of the proofs are d , then the magnitude of the coefficients can be assumed to be at most $\exp(n^{O(d)})$ without loss of generality. For Sherali–Adams this follows easily as any Sherali–Adams refutation over the reals can be converted into a Sherali–Adams refutation over \mathbb{Z} without badly affecting the coefficient size.

Theorem 15. *Let F be an unsatisfiable CNF formula on n variables and m clauses. If there is a degree- d Sherali–Adams refutation of F then there is a degree- d Sherali–Adams refutation of F over \mathbb{Z} where every coefficient is bounded in magnitude by $\exp(n^{O(d)})$.*

Proof. This is essentially the usual proof of completeness for Sherali–Adams (see e.g. [\[FKP19\]](#)). Consider a degree- d Sherali–Adams refutation of F which, by [Lemma 6](#), we can write as

$$\sum_{i=1}^m -J_i \overline{C}_i + J = -1.$$

We can express the existence of such a proof as a system of integer linear inequalities of the form $Ax = b, x \geq 0$ over $mn^{O(d)} = n^{O(d)}$ variables and over $n^{O(d)}$ constraints where all coefficients of the

matrix A and b are in $\{1, 0, -1\}$, and indeed b has a single non-zero entry with value -1 (see [FKP19, Chapter 2] for an explicit description of the system). By known results on linear programming this implies that the coefficients of the above Sherali–Adams refutation can be assumed to be rational with description length $n^{O(d)}$. Let L be the least common multiple of the denominators all rational numbers occurring in the refutation. By multiplying through by L we obtain the identity

$$\sum_{i=1}^m -LJ_i\bar{C}_i + LJ = -L.$$

We can then add the integer $L - 1$ to both sides (noting that $LJ + L - 1$ is a conical junta) to obtain an integer-coefficient Sherali–Adams refutation with the desired coefficient bound. \square

For Nullstellensatz the proof is slightly different as we need to recruit known bounds for integer solutions to systems of linear equations.

Theorem 16. *Let F be an unsatisfiable CNF formula on n variables and m clauses. If there is a degree- d Nullstellensatz refutation of F over \mathbb{Z} , then there is a degree- d Nullstellensatz refutation over \mathbb{Z} where every coefficient has magnitude at most $\exp(n^{O(d)})$.*

Proof. This follows the standard proof of completeness for Nullstellensatz proofs (see e.g. [Pit96, Bus98]). Write $F = C_1 \wedge \cdots \wedge C_m$ and suppose F has n variables. A degree- d \mathbb{Z} -Nullstellensatz proof of F can be written as

$$\sum_{i=1}^m q_i \bar{C}_i = 1$$

for some integer-coefficient multilinear polynomials q_i . We can express the existence of such a proof as a system of \mathbb{Z} -linear equations $Ax = b$ over $mn^{O(d)}$ variables — roughly one variable for each monomial m of degree at most d — where each coefficient in A and b is small. The result then follows by the known strongly-polynomial time algorithms for finding integer solutions to systems of linear equations over \mathbb{Z} (in particular, via the *Hermite Normal Form* [KB79]).

The system of linear equations is defined as follows. For each $i \in [m]$ and $S \subseteq [n]$ with $|S| \leq d$ we let $\hat{q}_i(S) \in \mathbb{Z}$ denote the coefficient of the monomial $x_S = \prod_{i \in S} x_i$ in the polynomial q_i . Letting $C_{n,d}$ denote all subsets of $[n]$ of size d , we can write the Nullstellensatz refutation as

$$\sum_{i=1}^m \sum_{S \in C_{n,d}} \hat{q}_i(S) x_S \bar{C}_i = 1.$$

From this, we get a system of \mathbb{Z} -linear equations over variables $\hat{q}_i(S)$ for each $i \in [m]$, $S \in C_{n,d}$ enforcing that all monomials in the proof of degree $d > 1$ must cancel out to 0, and the monomials of degree 0 must sum to 1. The system of equations has one constraint for each monomial x_S with $S \in C_{n,d}$ and at most $m|C_{n,d}| \leq mn^{O(d)}$ variables; each coefficient in the system of linear equations is ± 1 from the expansion of $x_S \bar{C}_i$ into a sum of monomials. By reducing to Hermite Normal Form we can find an integer solution to this system with coefficients of size at most $\exp(n^{O(d)})$. \square

Finally, we can consider RevRes proofs. An obvious fact is that any Resolution proof with width w has $n^{O(w)}$ distinct clauses without loss of generality. However, this result fails for RevRes, since we can no longer reuse clauses an unlimited number of times. By combining the previous results with the intersection theorem (Theorem 14), one can also immediately deduce the following result that gives a weak bound on the size of RevRes and RevResT proofs with bounded width. We omit the proof.

Corollary 6. *Let F be an unsatisfiable CNF formula on n variables. If there is a width- d RevRes refutation of F (RevResT resp.) then there is a width- $O(d)$ and size $\exp(n^{O(d)})$ RevRes refutation of F (RevResT resp.).* \square

Acknowledgements

We thank Albert Atserias, Ilario Bonacina, Pritish Kamath, and David Steurer for discussions, and the anonymous reviewers for their suggestions that helped us improve the presentation of the paper. M.G., A.H., S.J., and G.M. were supported by the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number MB22.00026. S.J. did part of the work while being supported by the Quantum Systems Accelerator through DOE. W.P., R.R., and R.T. were supported by NSERC.

References

- [AL19] Albert Atserias and Massimo Lauria. Circular (yet sound) proofs. In *Proceedings of the 22nd Theory and Applications of Satisfiability Testing (SAT)*, pages 1–18. Springer, 2019. doi:10.1007/978-3-030-24258-9_1.
- [Ald83] David Aldous. Minimization algorithms and random walk on the d-cube. *The Annals of Probability*, 11(2):403–413, 1983. URL: <http://www.jstor.org/stable/2243696>.
- [ALN16] Albert Atserias, Massimo Lauria, and Jakob Nordström. Narrow proofs may be maximally long. *ACM Transactions on Computational Logic*, 17(3):1–30, 2016. doi:10.1145/2898435.
- [BB22] Ilario Bonacina and Maria Luisa Bonet. On the strength of Sherali-Adams and Nullstellensatz as propositional proof systems. In *Proceedings of the 37th Symposium on Logic in Computer Science (LICS)*. ACM, aug 2022. doi:10.1145/3531130.3533344.
- [BCE⁺98] Paul Beame, Stephen Cook, Jeff Edmonds, Russell Impagliazzo, and Toniann Pitassi. The relative complexity of NP search problems. *Journal of Computer and System Sciences*, 57(1):3–19, 1998. doi:10.1006/jcss.1998.1575.
- [BCIP02] Joshua Buresh-Oppenheim, Matthew Clegg, Russell Impagliazzo, and Toniann Pitassi. Homogenization and the polynomial calculus. *Computational Complexity*, 11(3-4):91–108, 2002. doi:10.1007/s00037-002-0171-6.
- [Ben09] Eli Ben-Sasson. Size-space tradeoffs for resolution. *SIAM Journal on Computing*, 38(6):2511–2525, 2009. doi:10.1137/080723880.
- [BFI22] Sam Buss, Noah Fleming, and Russell Impagliazzo. TFNP characterizations of proof systems and monotone circuits, 2022. URL: <https://eccc.weizmann.ac.il/report/2022/141/>.
- [BIK⁺94] 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.
- [BKT14] Samuel Buss, Leszek Aleksander Kołodziejczyk, and Neil Thapen. Fragments of approximate counting. *The Journal of Symbolic Logic*, 79(2):496–525, 2014. URL: <http://www.jstor.org/stable/43303745>.
- [BLM07] María Luisa Bonet, Jordi Levy, and Felip Manyà. Resolution for Max-SAT. *Artificial Intelligence*, 171(8-9):606–618, 2007. doi:10.1016/j.artint.2007.03.001.
- [BM04] 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.
- [BR98] Paul Beame and Søren Riis. More on the relative strength of counting principles. In *Proceedings of the DIMACS Workshop on Proof Complexity and Feasible Arithmetics*, volume 39, pages 13–35, 1998.

- [BT22] Ilario Bonacina and Neil Thapen. A separation of PLS from PPP. Technical report, Electronic Colloquium on Computational Complexity (ECCC), 2022. URL: <https://eccc.weizmann.ac.il/report/2022/089/>.
- [Bus98] Samuel Buss. Lower bounds on Nullstellensatz proofs via designs. In *Proof Complexity and Feasible Arithmetics*, pages 59–71. AMS, 1998.
- [CDDT09] Xi Chen, Decheng Dai, Ye Du, and Shang-Hua Teng. Settling the complexity of Arrow-Debreu equilibria in markets with additively separable utilities. In *Proceedings of the 50th Symposium on Foundations of Computer Science (FOCS)*, pages 273–282, 2009. doi:10.1109/FOCS.2009.29.
- [CDO15] Xi Chen, David Durfee, and Anthi Orfanou. On the complexity of Nash equilibria in anonymous games. In *Proceedings of the 47th Symposium on Theory of Computing (STOC)*, pages 381–390, 2015. doi:10.1145/2746539.2746571.
- [CDT09] Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player Nash equilibria. *Journal of the ACM*, 56(3):14:1–14:57, 2009. doi:10.1145/1516512.1516516.
- [CEI96] Matthew Clegg, Jeff Edmonds, and Russell Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the 28th Symposium on Theory of Computing (STOC)*, pages 174–183, 1996. doi:10.1145/237814.237860.
- [CPY17] Xi Chen, Dimitris Paparas, and Mihalis Yannakakis. The complexity of non-monotone markets. *Journal of the ACM*, 64(3):20:1–20:56, 2017. doi:10.1145/3064810.
- [CR79] Stephen Cook and Robert Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44(1):36–50, 1979. doi:10.2307/2273702.
- [CSVY08] Bruno Codenotti, Amin Saberi, Kasturi Varadarajan, and Yinyu Ye. The complexity of equilibria: Hardness results for economies via a correspondence with games. *Theoretical Computer Science*, 408(2–3):188–198, 2008. doi:10.1016/j.tcs.2008.08.007.
- [Das19] Constantinos Daskalakis. Equilibria, fixed points, and computational complexity. In *Proceedings of the International Congress of Mathematicians (ICM)*. World Scientific, 2019. doi:10.1142/9789813272880_0009.
- [DGP09] Constantinos Daskalakis, Paul Goldberg, and Christos Papadimitriou. The complexity of computing a Nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009. doi:10.1137/070699652.
- [DM12] Stefan Dantchev and Barnaby Martin. Rank complexity gap for Lovász-Schrijver and Sherali-Adams proof systems. *computational complexity*, 22(1):191–213, nov 2012. doi:10.1007/s00037-012-0049-1.
- [DMR09] Stefan Dantchev, Barnaby Martin, and Mark Rhodes. Tight rank lower bounds for the Sherali-Adams proof system. *Theoretical Computer Science*, 410(21-23):2054–2063, 2009. doi:10.1016/j.tcs.2009.01.002.
- [DP11] Constantinos Daskalakis and Christos Papadimitriou. Continuous local search. In *Proceedings of the 22nd Symposium on Discrete Algorithms (SODA)*, pages 790–804. SIAM, 2011. doi:10.1137/1.9781611973082.62.

- [DQS12] Xiaotie Deng, Qi Qi, and Amin Saberi. Algorithmic solutions for envy-free cake cutting. *Operations Research*, 60(6):1461–1476, 2012. doi:10.1287/opre.1120.1116.
- [dRGR22] Susanna de Rezende, Mika Göös, and Robert Robere. Proofs, circuits, and communication. *SIGACT News*, 53(1), 2022. doi:10.1145/3532737.3532745.
- [dRNMR19] Susanna de Rezende, Jakob Nordström, Or Meir, and Robert Robere. Nullstellensatz size-degree trade-offs from reversible pebbling. In Amir Shpilka, editor, *Proceedings of the 34th Computational Complexity Conference (CCC)*, volume 137, pages 18:1–18:16. Schloss Dagstuhl, 2019. doi:10.4230/LIPIcs.CCC.2019.18.
- [FG22] Aris Filos-Ratsikas and Paul Goldberg. The complexity of necklace splitting, consensus-halving, and discrete ham sandwich. *SIAM Journal on Computing*, 2022. (to appear). doi:10.1137/20m1312678.
- [FGGR22] Noah Fleming, Mika Göös, Stefan Grosser, and Robert Robere. On semi-algebraic proofs and algorithms. In *Proceedings of the 13th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 215 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 69:1–69:25. Schloss Dagstuhl, 2022. doi:10.4230/LIPIcs.ITCS.2022.69.
- [FGHS21] John Fearnley, Paul W. Goldberg, Alexandros Hollender, and Rahul Savani. The complexity of gradient descent: $\text{CLS} = \text{PPAD} \cap \text{PLS}$. In *Proceedings of the 53rd Symposium on Theory of Computing (STOC)*, pages 46–59, 2021. doi:10.1145/3406325.3451052.
- [FGMS20] John Fearnley, Spencer Gordon, Ruta Mehta, and Rahul Savani. Unique end of potential line. *Journal of Computer and System Sciences*, 114:1–35, 2020. doi:10.1016/j.jcss.2020.05.007.
- [FKP19] 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:10.1561/04000000086.
- [FMSV23] Yuval Filmus, Meena Mahajan, Gaurav Sood, and Marc Vinyals. Maxsat resolution and subcube sums. *ACM Trans. Comput. Log.*, 24(1):8:1–8:27, 2023. doi:10.1145/3565363.
- [FPT04] Alex Fabrikant, Christos Papadimitriou, and Kunal Talwar. The complexity of pure Nash equilibria. In *Proceedings of the 36th ACM Symposium on Theory of Computing (STOC)*, pages 604–612, 2004. doi:10.1145/1007352.1007445.
- [GHJ⁺22] Mika Göös, Alexandros Hollender, Siddhartha Jain, Gilbert Maystre, William Pires, Robert Robere, and Ran Tao. Further collapses in TFNP. In *Proceedings of the 37th Computational Complexity Conference (CCC)*, pages 33:1–33:15, 2022. doi:10.4230/LIPIcs.CCC.2022.33.
- [GKRS18] Mika Göös, Pritish Kamath, Robert Robere, and Dmitry Sokolov. Adventures in monotone complexity and TFNP. In *Proceedings of the 10th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 124, pages 38:1–38:19, 2018. doi:10.4230/LIPIcs.ITCS.2019.38.
- [GP18] Mika Göös and Toniann Pitassi. Communication lower bounds via critical block sensitivity. *SIAM Journal on Computing*, 47(5):1778–1806, 2018. doi:10.1137/16M1082007.

- [Hak21] Tuomas Hakoniemi. Monomial size vs. bit-complexity in sums-of-squares and polynomial calculus. In *Proceedings of the 36th Symposium on Logic in Computer Science (LICS)*. IEEE, 2021. doi:10.1109/lics52264.2021.9470545.
- [HKT24] Pavel Hubáček, Erfan Khaniki, and Neil Thapen. TFNP Intersections Through the Lens of Feasible Disjunction. In Venkatesan Guruswami, editor, *15th Innovations in Theoretical Computer Science Conference (ITCS 2024)*, volume 287 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 63:1–63:24, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: <https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ITCS.2024.63>, doi:10.4230/LIPIcs.ITCS.2024.63.
- [HN12] Trinh Huynh and Jakob Nordström. On the virtue of succinct proofs: Amplifying communication complexity hardness to time–space trade-offs in proof complexity. In *Proceedings of the 44th Symposium on Theory of Computing (STOC)*, pages 233–248. ACM, 2012. doi:10.1145/2213977.2214000.
- [HY20] Pavel Hubáček and Eylon Yogev. Hardness of continuous local search: Query complexity and cryptographic lower bounds. *SIAM Journal on Computing*, 49(6):1128–1172, 2020. doi:10.1137/17m1118014.
- [IR21] Dmitry Itsykson and Artur Riazanov. Proof complexity of natural formulas via communication arguments. In *Proceedings of the 36th Computational Complexity Conference (CCC)*, volume 200, pages 3:1–3:34. Schloss Dagstuhl, 2021. doi:10.4230/LIPIcs.CCC.2021.3.
- [JLRX24] Siddhartha Jain, Jiawei Li, Robert Robere, and Zhiyang Xun. On Pigeonhole Principles and Ramsey in TFNP, 2024. arXiv:2401.12604.
- [JPY88] David Johnson, Christos Papadimitriou, and Mihalis Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37(1):79–100, 1988. doi:10.1016/0022-0000(88)90046-3.
- [Juk12] Stasys Jukna. *Boolean Function Complexity: Advances and Frontiers*, volume 27 of *Algorithms and Combinatorics*. Springer, 2012.
- [Kam20] Pritish Kamath. *Some hardness escalation results in computational complexity theory*. PhD thesis, Massachusetts Institute of Technology, 2020. URL: <https://dspace.mit.edu/handle/1721.1/128290>.
- [KB79] Ravindran Kannan and Achim Bachem. Polynomial algorithms for computing the smith and hermite normal forms of an integer matrix. *SIAM Journal on Computing*, 8(4):499–507, 1979. doi:10.1137/0208040.
- [Kra19] Jan Krajíček. *Proof Complexity*. Cambridge University Press, 2019.
- [Kre89] Mark Krentel. Structure in locally optimal solutions. In *Proceedings of the 30th Symposium on Foundations of Computer Science (FOCS)*, pages 216–221, 1989. doi:10.1109/SFCS.1989.63481.
- [Kre90] Mark Krentel. On finding and verifying locally optimal solutions. *SIAM Journal on Computing*, 19(4):742–749, 1990. doi:10.1137/0219052.

- [LHdG08] Javier Larrosa, Federico Heras, and Simon de Givry. A logical approach to efficient Max-SAT solving. *Artificial Intelligence*, 172(2-3):204–233, 2008. doi:10.1016/j.artint.2007.05.006.
- [LNNW95] László Lovász, Moni Naor, Ilan Newman, and Avi Wigderson. Search problems in the decision tree model. *SIAM Journal on Discrete Mathematics*, 8(1):119–132, 1995. doi:10.1137/S0895480192233867.
- [LPR24] Yuhao Li, William Pires, and Robert Robere. Intersection Classes in TFNP and Proof Complexity. In Venkatesan Guruswami, editor, *15th Innovations in Theoretical Computer Science Conference (ITCS 2024)*, volume 287 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 74:1–74:22, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: <https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ITCS.2024.74>, doi:10.4230/LIPIcs.ITCS.2024.74.
- [Meh18] Ruta Mehta. Constant rank two-player games are PPAD-hard. *SIAM Journal on Computing*, 47(5):1858–1887, January 2018. doi:10.1137/15m1032338.
- [Mor01] 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>.
- [MP91] Nimrod Megiddo and Christos Papadimitriou. On total functions, existence theorems and computational complexity. *Theoretical Computer Science*, 81(2):317–324, 1991. doi:10.1016/0304-3975(91)90200-L.
- [NS94] Noam Nisan and Mario Szegedy. On the degree of boolean functions as real polynomials. *Computational Complexity*, 4(4):301–313, dec 1994. doi:10.1007/bf01263419.
- [O’D17] Ryan O’Donnell. SOS is not obviously automatizable, even approximately. In *Proceedings of the 8th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 67, pages 59:1–59:10. Schloss Dagstuhl, 2017. doi:10.4230/LIPIcs.ITCS.2017.59.
- [Pap94] Christos Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48(3):498–532, 1994. doi:10.1016/s0022-0000(05)80063-7.
- [Pit96] Toniann Pitassi. Algebraic propositional proof systems. In *Descriptive Complexity and Finite Models, Proceedings of a DIMACS Workshop 1996*, volume 31 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 215–244. DIMACS/AMS, 1996. doi:10.1090/dimacs/031/07.
- [Pud15] Pavel Pudlák. On the complexity of finding falsifying assignments for Herbrand disjunctions. *Archive for Mathematical Logic*, 54(7-8):769–783, 2015. doi:10.1007/s00153-015-0439-6.
- [RW92] Ran Raz and Avi Wigderson. Monotone circuits for matching require linear depth. *Journal of the ACM*, 39(3):736–744, jul 1992. doi:10.1145/146637.146684.
- [RW17] Prasad Raghavendra and Benjamin Weitz. On the bit complexity of sum-of-squares proofs. In *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 80:1–80:13, 2017. doi:10.4230/LIPIcs.ICALP.2017.80.

- [SA94] Hanif Sherali and Warren Adams. A hierarchy of relaxations and convex hull characterizations for mixed-integer zero–one programming problems. *Discrete Applied Mathematics*, 52(1):83–106, jul 1994. doi:[10.1016/0166-218x\(92\)00190-w](https://doi.org/10.1016/0166-218x(92)00190-w).
- [Sch91] Alejandro Schäffer. Simple local search problems that are hard to solve. *SIAM Journal on Computing*, 20(1):56–87, 1991. doi:[10.1137/0220004](https://doi.org/10.1137/0220004).
- [SZZ18] Katerina Sotiraki, Manolis Zampetakis, and Giorgos Zirdelis. PPP-completeness with connections to cryptography. In *Proceedings of the 59th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 148–158, 2018. doi:[10.1109/FOCS.2018.00023](https://doi.org/10.1109/FOCS.2018.00023).
- [Zha09] Shengyu Zhang. Tight bounds for randomized and quantum local search. *SIAM Journal on Computing*, 39(3):948–977, 2009. doi:[10.1137/06066775X](https://doi.org/10.1137/06066775X).