Automating Cutting Planes Is NP-Hard

Mika Göös*
goos@stanford.edu
Stanford University
USA

Sajin Koroth
skoroth@sfu.ca
Simon Fraser University
Canada

Ian Mertz
mertz@cs.toronto.edu
University of Toronto
Canada

Toniann Pitassi
toni@cs.toronto.edu
University of Toronto / IAS
Canada / USA

ABSTRACT

We show that Cutting Planes (CP) proofs are hard to find: Given an unsatisfiable formula F, it is NP-hard to find a CP refutation of F in time polynomial in the length of the shortest such refutation; and unless Gap-Hitting-Set admits a nontrivial algorithm, one cannot find a *tree-like* CP refutation of F in time polynomial in the length of the shortest such refutation.

The first result extends the recent breakthrough of Atserias and Müller (FOCS 2019) that established an analogous result for Resolution. Our proofs rely on two new lifting theorems: (1) Dag-like lifting for gadgets with *many output bits*. (2) Tree-like lifting that simulates an r-round protocol with gadgets of query complexity $O(r)$.

CCS CONCEPTS

• **Theory of computation → Proof complexity**; **Computational complexity and cryptography**.

KEYWORDS

Proof Complexity, Cutting Planes, Automatability, Lifting theorems

ACM Reference Format:
Mika Göös, Sajin Koroth, Ian Mertz, and Toniann Pitassi. 2020. Automating Cutting Planes Is NP-Hard. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC '20), June 22–26, 2020, Chicago, IL, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3357713.3384248

1 INTRODUCTION

Propositional proof systems are by nature *non-deterministic*: a short refutation of a formula F in a particular proof system constitutes an easy-to-check certificate (an NP-witness) of F's unsatisfiability (which is a coNP-property). The question of efficiently finding such refutations is the foundational problem of *automated theorem proving* with applications to algorithm design, e.g., for combinatorial optimization [FKP19]. The following definition is due to Bonet et al. [BPR00].

> ***Automatability.*** A proof system $\mathcal{P}$ is *automatable* if there is an algorithm that on input an unsatisfiable CNF formula F outputs some $\mathcal{P}$-refutation of F in time polynomial in the length (or size) of the shortest $\mathcal{P}$-refutation of F.

Algorithms. Several basic propositional proof systems are automatable when restricted to proofs of bounded *width* or *degree*. For example, Resolution refutations of width w can be found in time $n^{O(w)}$ for n-variate formulas [BW01]. Efficient algorithms also exist for finding bounded-degree refutations in algebraic proof systems such as Nullstellensatz, Polynomial Calculus [CEI96], Sherali–Adams, and Sum-of-Squares (under technical assumptions) [O'D17, RW17].

Hardness. Without restrictions on width or degree, many of these systems are known *not* to be automatable. For the most basic system, Resolution, a long line of work [Iwa97, ABMP01, AR08, MPW19] recently culminated in an optimal non-automatability result by Atserias and Müller [AM19]. They showed that Resolution is not automatable unless P = NP. Under stronger hardness assumptions non-automatability results are known for Nullstellensatz and Polynomial Calculus [GL10, MPW19] as well as for various Frege systems [KP98, BPR97b, BDG+04].

This work. The above list conspicuously omits to mention any hardness results for the Cutting Planes (CP) proof system (defined in Section 1.1 below). Indeed, we show the first such results:

- (§1.2) It is NP-hard to automate CP. This is an Atserias–Müller style result for CP.
- (§1.3) Under a stronger assumption, it is hard to automate *tree-like* CP.

One reason Cutting Planes has been lacking non-automatability results is because of the shortage of techniques to prove lower bounds on CP refutation length. Virtually the only known method has been to find reductions to *monotone circuit* lower bounds (for example, via monotone feasible interpolation). Our proofs rely on two new *lifting theorems*, one of which bypasses the need for monotone circuit lower bounds. See Section 2 for an overview of our techniques.

*Part of the work done while at Institute for Advanced Study.

STOC '20, June 22–26, 2020, Chicago, IL, USA

ACM ISBN 978-1-4503-6979-4/20/06…$15.00
https://doi.org/10.1145/3357713.3384248

1.1 Cutting Planes

Cook, Coullard, and Turán [CCT87] introduced Cutting Planes as a propositional proof system inspired by a like-named method to solve integer linear programs. The method uses rounding of linear inequalities (Chvátal–Gomory cuts) to reason about the integral solutions to a linear program.

The proof system version of CP is defined as follows. Suppose we are given a CNF formula F over variables $x_1, \dots, x_n$. A (dag-like) *Cutting Planes refutation* of F is a sequence of *lines* $\ell_1, \dots, \ell_m$ (where m is the *length*), each line being a linear inequality, $\sum_i a_i x_i \geq b$, with integer coefficients, $a_i, b \in \mathbb{Z}$. We require that the sequence ends with the contradictory inequality $\ell_m := [\, 0 \geq 1 \,]$ and that each ℓ_i satisfies one of the following:

- *Axiom.* Line ℓ_i is either a boolean axiom ($x_i \geq 0$ or $-x_i \geq -1$) or an encoding of a clause of F (for example, clause $(x_1 \vee \bar{x}_2)$ gets encoded as $x_1 + (1 - x_2) \geq 1$).
- *Derivation.* Line ℓ_i is deduced from two *premises* $\ell_j, \ell_{j'}$ where $j, j' < i$ (perhaps $j = j'$) by an application of a *sound rule*. (A refutation is *tree-like* if each line appears at most once as a premise.)

In the original paper [CCT87] the rules were: (1) deriving from $\ell_j, \ell_{j'}$ any nonnegative integer linear combination of them, and (2) deriving from $\sum a_i x_i \geq b$ the line $\sum (a_i/c) x_i \geq \lceil b/c \rceil$ where $c := \gcd(a_1, \dots, a_n)$. Stronger rules have also been studied, e.g., [CKS90, BCC93], the most general being the *semantic rule*, which allows *any* sound inference: ℓ_i can be derived from $\ell_j, \ell_{j'}$ provided every boolean vector $x \in \{0,1\}^n$ that satisfies both ℓ_j and $\ell_{j'}$ also satisfies ℓ_i. In this paper, we adopt the best of all possible worlds: our lower bounds on CP refutation length will hold even against the semantic system and our upper bounds use the weakest possible rules (in fact, our upper bounds hold for Resolution, which is simulated by every variety of CP).

1.2 Dag-like Result

Our first main result is a CP analogue of the Atserias–Müller theorem [AM19].

Theorem 1 (Dag-like). *There is a polynomial-time algorithm $\mathcal{A}$ that on input an n-variate 3-CNF formula F outputs an unsatisfiable CNF formula $\mathcal{A}(F)$ such that:*

- *If F is* satisfiable*, then $\mathcal{A}(F)$ admits a CP refutation of length at most $n^{O(1)}$.*
- *If F is* unsatisfiable*, then $\mathcal{A}(F)$ requires CP refutations of length at least $2^{n^{\Omega(1)}}$.*

Consequently, it is NP-hard to approximate the minimum CP proof length up to a factor of 2^{n^ε} for some $\varepsilon > 0$. In particular, CP is not automatable unless P = NP.

1.3 Tree-like Result

Our second result is a similar theorem for *tree-like* Cutting Planes. However, we need a stronger hardness assumption (which is morally necessary; see Section 2.2) that we now formulate.

An *n-set system* is a collection $\mathcal{S} = \{S_1, \dots, S_n\}$ where $S_i \subseteq [n]$ for each $i \in [n]$. A subset $H \subseteq [n]$ is a *hitting set* for $\mathcal{S}$ if $H \cap S_i \neq \emptyset$ for all $i \in [n]$. The *hitting set number* of $\mathcal{S}$, denoted $\gamma(\mathcal{S})$, is the minimum size of a hitting set for $\mathcal{S}$. The k-Gap-Hitting-Set promise problem is to distinguish between the cases $\gamma(\mathcal{S}) \leq k$ versus $\gamma(\mathcal{S}) \geq k^2$. A trivial algorithm can solve this problem in time $n^{O(k)}$. It is conjectured that there are no nontrivial algorithms for k as large as $(1-\epsilon)\log n$. Under the Exponential-Time Hypothesis [IP01], the problem is known to be hard up to $k \leq (\log\log n)^{1-o(1)}$ [Lin19]. We need an assumption that is stronger by a hair's breadth.

Conjecture 1. The k-Gap-Hitting-Set problem requires time $n^{\Omega(k)}$ for some $k = k(n)$ with

$$\omega(\log\log n) \;\leq\; k(n) \;\leq\; \log^{1/3} n. \qquad (\dagger)$$

Our second main result says that tree-like CP is not automatable under Conjecture 1.

Theorem 2 (Tree-like). *Let $k = k(n)$ satisfy (†). There is an $n^{o(k)}$-time algorithm $\mathcal{A}$ that on input an n-set system $\mathcal{S}$, outputs a CNF formula $\mathcal{A}(\mathcal{S})$ such that:*

- *If $\gamma(\mathcal{S}) \leq k$, then $\mathcal{A}(\mathcal{S})$ admits a tree-like CP refutation of length at most $n^{o(k)}$.*
- *If $\gamma(\mathcal{S}) \geq k^2$, then $\mathcal{A}(\mathcal{S})$ requires tree-like CP refutations of length at least $n^{\omega(k)}$.*

2 OVERVIEW OF PROOFS

In this section, we explain why both of our main results (dag-like and tree-like) follow from appropriate kinds of *lifting theorems*. Abstractly speaking, a *lifting theorem* is a tool that translates a lower-bound result for a weak model of computation (for us, Resolution) into an analogous lower-bound result for a strong model of computation (for us, Cutting Planes). Starting with Raz and McKenzie [RM99] such theorems now exist for an enormous variety of computational models. In proof complexity alone, prior examples of lifting applications include [BEGJ00, HN12, GP18, dRNV16, GGKS18, GKRS19, dRMN+19]. We provide two more.

2.1 Dag-like Case

Our proof of Theorem 1 builds directly on top of the breakthrough of Atserias and Müller [AM19]. Given an n-variate 3-CNF formula F, they construct a formula $\mathrm{Ref}(F)$, which is an intricate CNF encoding of the claim "F admits a short Resolution refutation." Luckily, the exact details of $\mathrm{Ref}(F)$ are not important for us. We only need a few high-level properties of their construction.

Block-width. The variables of $\mathrm{Ref}(F)$ come partitioned into some number of *blocks*. Given a clause D over the variables of $\mathrm{Ref}(F)$, we define its *block-width* as the number of blocks that D *touches*, that is, contains a variable (or its negation) from that block. The *block-width* of a Resolution refutation is the maximum block-width of any of its clauses.

Lemma 3 (Atserias–Müller [AM19])**.** *There is a polynomial-time algorithm that on input an n-variate 3-CNF formula F outputs an unsatisfiable[1] CNF formula $\mathrm{Ref}(F)$ such that*

[1] Strictly speaking, $\mathrm{Ref}(F)$, as defined in [AM19], may sometimes be satisfiable, in which case its Resolution width/length complexity is understood as ∞. However this case is equivalent to our reformulation, as we can guarantee that $\mathrm{Ref}(F)$ is always unsatisfiable by consider instead the CNF formula $\mathrm{Ref}(F) \wedge T$ where T is some

- *If F is* satisfiable, *then* $\mathrm{Ref}(F)$ *admits a* $n^{O(1)}$*-length* $O(1)$*-block-width Resolution refutation.*
- *If F is* unsatisfiable, *then* $\mathrm{Ref}(F)$ *requires Resolution refutations of block-width at least* $n^{\Omega(1)}$.

Atserias and Müller finish their proof by modifying $\mathrm{Ref}(F)$ slightly via *relativization*, an operation due to Danchev and Riis [DR03] (see also [Gar19]). What this operation achieves is to turn a formula requiring block-width b into a formula requiring Resolution length $2^{\Omega(b)}$. If F is unsatisfiable, relativized-$\mathrm{Ref}(F)$ will have exponential length complexity. On the other hand, if F is satisfiable, relativized-$\mathrm{Ref}(F)$ continues to have a short Resolution refutation, inherited from $\mathrm{Ref}(F)$.

In this paper, in order to make $\mathrm{Ref}(F)$ hard for Cutting Planes (when F is unsatisfiable), we will modify the formula by block-wise *composing (aka lifting)* it with a small gadget, an operation similar to relativization.

Lifting width. Recently, Garg et al. [GGKS18] introduced a new lifting-based lower-bound technique for Cutting Planes: they showed how to lift Resolution width to Cutting Planes length. Namely, if F is an n-variate formula requiring Resolution width w, then for a careful choice of a *gadget* $g\colon \{0,1\}^m \to \{0,1\}$, $m = n^{O(1)}$, the composed formula $F \circ g^n$—obtained from F by substituting each of its variables with a copy of g—has Cutting Planes length complexity $n^{\Theta(w)}$.

What happens if we try to apply the lifting result of [GGKS18] to the formula $\mathrm{Ref}(F)$? When F is unsatisfiable, we indeed do get (using width $\geq$ block-width) that $\mathrm{Ref}(F) \circ g^n$ requires exponential-length CP refutations. However, when F is satisfiable, even though $\mathrm{Ref}(F)$ is promised to have block-width $O(1)$, its usual width still turns out to be $n^{\Omega(1)}$. Therefore the composition with g would blow up the length complexity, not creating the desired gap in CP proof length.

Lifting block-width. Our idea, in short, is to build on [GGKS18] and prove a lifting theorem for block-width (instead of width). Suppose F is a formula whose $n\ell$ variables are partitioned into n many blocks of ℓ variables each (typically $\ell = n^{\Theta(1)}$). We will consider compositions $F \circ g_\ell^n$ with a *multi-output* gadget $g_\ell\colon \{0,1\}^m \to \{0,1\}^\ell$, one gadget for each block; see Section 3.4 for the formal definition. Below, $\mathrm{res}(\cdot)$ denotes Resolution length complexity, $\mathrm{cut}(\cdot)$ denotes Cutting Planes length complexity, and $\mathrm{bw}(\cdot)$ denotes Resolution block-width complexity.

Theorem 4 (Block lifting). *Fix an unsatisfiable CNF formula F having n many blocks of ℓ variables each. There is a gadget $g_\ell\colon \{0,1\}^m \to \{0,1\}^\ell$ where $m := (n\ell)^{\Theta(1)}$ such that*

$$m^{\Omega(\mathrm{bw}(F))} \leq \mathrm{cut}(F \circ g_\ell^n) \leq \mathrm{res}(F \circ g_\ell^n) \leq m^{O(\mathrm{bw}(\Pi))} \cdot |\Pi|,$$

where Π is any Resolution refutation of F of length $|\Pi|$ and block-width $\mathrm{bw}(\Pi)$.

Our main dag-like theorem (Theorem 1) now follows immediately by combining Lemma 3 and Theorem 4. Namely, consider the algorithm $\mathcal{A}$ that on input an n-variate 3-CNF formula F outputs the CNF formula $\mathcal{A}(F) := \mathrm{Ref}(F) \circ g_\ell^k$ where $\mathrm{Ref}(F)$ has $k \leq n^{O(1)}$ many blocks with $\ell \leq n^{O(1)}$ variables each. We only need to note that this composed formula is constructible in polynomial time, which will be evident from the formal definition; see Fact 7 in Section 3.5. Therefore, to prove Theorem 1 it remains to prove Theorem 4, which we do in Section 4.

Relation to monotone circuits. To conclude this subsection, we offer some philosophical musings on the techniques used to prove Theorem 4. Non-automatability results for Cutting Planes have been elusive in part because of the limitations of existing techniques to prove lower bounds on refutation length (as required by the second item in Theorem 1). The only technique available for some twenty years has been *monotone feasible interpolation* [BPR97a, Kra97, HP18], which translates lower bounds for (real) monotone circuits to lower bounds on Cutting Planes length. Historically, the downside with the technique was that it only seemed to apply to highly specialized formulas (e.g., clique-vs-coloring). However, the technique was recently extended to handle a more general class of formulas, random $\Theta(\log n)$-CNFs [HP17, FPPR17]. The only other available lower-bound technique is the aforementioned lifting theorem [GGKS18]. That technique is also powerful enough to prove lower bounds not only on CP length, but also on monotone circuit size. (Whether lifting should be classified under monotone interpolation is up for debate, since this depends on how broadly one defines monotone interpolation.)

In contrast, our Theorem 4 is *not* proved through monotone circuit lower bounds, but through a new weaker model of computation, dubbed *simplex-dags* in Section 3.2. At the heart of monotone interpolation is a characterization of monotone circuits by a two-party communication game [Raz95, Pud10, Sok17]. In this language, our Theorem 4 is obtained not by studying a two-party communication model, but rather a *multi-party* model. Considering a large number of communicating parties is what allows us to analyze multi-output gadgets; we do not know how to do this with only two parties.

2.2 Tree-like Case

Proof of Theorem 2 builds on the important paper by Alekhnovich and Razborov [AR08] (followed up by [GL10, MPW19]). They show that tree-like Resolution is not automatable assuming the fixed parameter hierarchy does not collapse (which is implied by the Exponential-Time Hypothesis). Since tree-like Resolution proofs *can* be found in quasipolynomial-time (we say tree-like Resolution is *quasipolynomially* automatable), they need to assume more than NP-hardness. Our results will inherit this need for a stronger assumption (namely, Conjecture 1), even though tree-like CP is not known to be quasipolynomially automatable.

The reduction of Alekhnovich and Razborov is somewhat complicated, but luckily we will only need as our starting point the following lemma from the follow-up work [MPW19].

Lemma 5 (Mertz et al. [MPW19])**.** *Let $k \leq \log^{1/3} n$. There is a polynomial-time algorithm $\mathcal{B}$ that on input a n-set system $\mathcal{S}$ outputs an unsatisfiable $O(\log n)$-CNF formula $\mathcal{B}(\mathcal{S})$ such that*

- *If $\gamma(\mathcal{S}) \leq k$, then $\mathcal{B}(\mathcal{S})$ admits a Resolution refutation of depth $O(\log n)$.*
- *If $\gamma(\mathcal{S}) \geq k^2$, then $\mathcal{B}(\mathcal{S})$ requires Resolution refutations of depth $\Omega(k \log n)$.*

formula over disjoint variables known to require large width (e.g., Tseitin contradictions [Urq87]).

To prove Theorem 2, our plan is once again to compose the formula $\mathcal{B}(\mathcal{S})$ with a (single-output-bit) gadget in order to lift the Resolution depth gap in Lemma 5 into tree-like CP length gap. To this end, we develop a new lifting theorem for "small" gadgets.

Limitations of existing methods. Let F be an unsatisfiable n-variate formula with Resolution depth complexity $\mathrm{d}(F)$. Existing lifting theorems [BEGJ00, dRNV16] when applied to F would require a gadget g of size $\mathrm{poly}(n)$ that can be computed by a decision tree of depth $\Theta(\log n)$ and hence of size $\mathrm{poly}(n)$. Writing $\text{res-tree}(\cdot)$ for tree-like Resolution length complexity, and $\text{cut-tree}(\cdot)$ for tree-like CP length complexity, the lifting theorems [BEGJ00, dRNV16] show

$$\text{cut-tree}(F \circ g^n) \;=\; \text{res-tree}(F \circ g^n)^{\Theta(1)} \;=\; n^{\Theta(\mathrm{d}(F))}. \qquad (1)$$

The base of the exponent above (namely, $\mathrm{poly}(n)$) is the decision tree size of g. If we applied (1) to Lemma 5, we would only end up with a length gap of $n^{O(\log n)}$ versus $n^{\omega(\log n)}$. But these lengths—and hence running times for the automating algorithm—are enough to solve the k-Gap-Hitting-Set problem, which prevents us from getting a hardness result.

Small gadget lifting. What we need is a lifting theorem for small gadgets, that is, gadgets computed by small decision trees. It is an important open problem whether tree-like lifting is possible with a *constant-size* gadget. In this paper, we are able to use a gadget of decision-tree size depending only on the quantity we want to lift, namely $\mathrm{d}(F)$, and not depending on the number of variables n of F. Our lifting theorem can be seen as a generalization of previous ones, which handled the case $\mathrm{d}(F) = n^{\Omega(1)}$, and can also be viewed as a step towards proving a lifting theorem for significantly smaller gadgets (eventually, constant-size).

Theorem 6 (Small gadget lifting). *For every m there exists a gadget $g\colon \{0,1\}^{\mathrm{poly}(m)} \to \{0,1\}$ of query complexity $O(\log m)$ such that for every unsatisfiable n-variate CNF formula F,*

$$m^{\Theta(\min(\mathrm{d}(F),\, m))} \le \text{cut-tree}(F \circ g^n) \le \text{res-tree}(F \circ g^n) \le m^{O(\mathrm{d}(F))}.$$

Our main tree-like theorem (Theorem 2) now follows by combining Lemma 5 and Theorem 6. Indeed, choose $m := \log^2 n$ and consider the algorithm $\mathcal{A}$ that on input an n-set system $\mathcal{S}$ outputs the formula $\mathcal{A}(\mathcal{S}) := \mathcal{B}(\mathcal{S}) \circ g^{n'}$ where $\mathcal{B}(\mathcal{S})$ has $n' = n^{O(1)}$ variables. We have

$$\begin{aligned} \gamma(\mathcal{S}) \le k \quad &\Longrightarrow \quad \text{cut-tree}(\mathcal{A}(\mathcal{S})) \le m^{O(\log n)} \\ &\qquad = n^{O(\log\log n)} \le n^{o(k)}, \\ \gamma(\mathcal{S}) \ge k^2 \quad &\Longrightarrow \quad \text{cut-tree}(\mathcal{A}(\mathcal{S})) \ge m^{\Omega(k \log n)} \\ &\qquad = n^{\Omega(k \log\log n)} \ge n^{\omega(k)}. \end{aligned}$$

Finally, we note that the composed formula $\mathcal{B}(\mathcal{S}) \circ g^{n'}$ can be constructed in time $n^{o(k)}$. This will be evident from the formal definition (see the full version [GKMP20, Fact 11]), but the intuition is as follows. Each $O(\log n)$-width clause of $\mathcal{B}(\mathcal{S})$ will turn into a whole family $O(\log m \log n)$-width clauses for $\mathcal{B}(\mathcal{S}) \circ g^{n'}$. The family for a particular clause D is obtained by replacing each literal of D in all possible ways by an $O(\log m)$-length root-to-leaf path (of which there are $2^{O(\log m)}$ many) in the decision tree for g. Altogether this will yield $|\mathcal{B}(\mathcal{S})| \cdot (2^{O(\log m)})^{O(\log n)} = n^{O(1)} \cdot n^{o(k)} = n^{o(k)}$ many clauses. Therefore, to prove Theorem 2 it remains to prove Theorem 6; see the full version [GKMP20, Section 6] for the proof.

Relation to real protocols. The lower bound in Theorem 6 holds not only for tree-like Cutting Planes but also for a stronger model of computation, *real communication protocols* [Kra98]. This is not surprising: all existing lower bounds on tree-like CP length have been proved through real protocols (or the even more powerful model of randomized protocols). In a nutshell, our proof of Theorem 6 extends the techniques in a long line of work on tree-like lifting [RM99, BEGJ00, GPW15, dRNV16, GPW17, CFK+19], optimizing the argument in order to get rid of the dependence on the input size n. See the full version [GKMP20] for a detailed overview.

3 DAG-LIKE DEFINITIONS

In this paper, we adopt the standard *top-down* view of proofs [Pud00, AD08]. Namely, we interpret a refutation of an n-variate CNF formula $F := \wedge_{i\in[m]} D_i$ as a way of solving the associated *falsified-clause search problem* $S_F \subseteq \{0,1\}^n \times [m]$. The problem S_F is, on input a truth assignment $x \in \{0,1\}^n$, to find a clause D_j, $j \in [m]$, falsified by x, that is, $D_j(x) = 0$. For example, tree-like Resolution refutations of F are equivalent to decision trees solving S_F [LNNW95]. We proceed to formalize this for dag-like models. The material in Section 3.1 is standard. Section 3.2 introduces a novel model, *simplex-dags*, for which we develop a lifting theorem in Section 4.

3.1 Standard Models

Abstract dags. Fix an abstract search problem $S \subseteq \mathcal{I} \times \mathcal{O}$, that is, on input $x \in \mathcal{I}$ the goal is to find some $o \in S(x) := \{o \in \mathcal{O} : (x,o) \in S\}$. We always work with *total* search problems where $S(x) \neq \emptyset$ for all $x \in \mathcal{I}$. Fix also a family $\mathcal{F}$ of functions $\mathcal{I} \to \{0,1\}$. An *$\mathcal{F}$-dag* solving S is a directed acyclic graph of out-degree ≤ 2 where each vertex v is associated with a function $f_v \in \mathcal{F}$ (here $f^{-1}(1)$ is sometimes called the *feasible set* for v) satisfying the following.

- *Root.* There is a designated root vertex v (in-degree 0) that satisfies $f_v \equiv 1$.
- *Non-leaf.* Every non-leaf v with children u, u' (perhaps $u = u'$) has $f_v^{-1}(1) \subseteq f_u^{-1}(1) \cup f_{u'}^{-1}(1)$.
- *Leaves.* For every leaf v there is some output $o \in \mathcal{O}$ such that $f_v^{-1}(1) \subseteq S^{-1}(o)$.

The *size* of an $\mathcal{F}$-dag is its number of vertices.

Decision-dags and Resolution. Consider instantiating the above template with the n-bit input domain $\mathcal{I} := \{0,1\}^n$ and taking $\mathcal{F}$ to be the set of all conjunctions over the literals $x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n$. We call such $\mathcal{F}$-dags simply *decision-dags*. Apart from the size of a decision-dag another important measure is its *width*: the maximum width of a conjunction used. We define

$$\begin{aligned} \text{dec-dag}(S) &:= \text{least } \textit{size} \text{ of a decision-dag solving } S, \\ \mathrm{w}(S) &:= \text{least } \textit{width} \text{ of a decision-dag solving } S. \end{aligned}$$

When specialized to unsatisfiable CNF search problems $S = S_F$, we recover the usual Resolution proof system. Indeed, $\text{dec-dag}(S_F)$ equals $\mathrm{res}(F)$, the length required to refute F in Resolution, and $\mathrm{w}(S_F)$ equals the Resolution width complexity of F (famously studied in [BW01]).

LTF-dags and Cutting Planes. Consider instantiating $\mathcal{I} := \{0,1\}^n$ and taking $\mathcal{F}$ to be the set of all n-bit *linear threshold functions* (LTFs). Recall that an $f \in \mathcal{F}$ is defined by a vector $a \in \mathbb{R}^{n+1}$ such

that $f(x) = 1$ iff $\sum_{i\in[n]} a_i x_i \geq a_{n+1}$. We call such $\mathcal{F}$-dags simply *LTF-dags*, and define

$$\text{ltf-dag}(S) \;:=\; \text{least } \textit{size} \text{ of an LTF-dag solving } S.$$

When specialized to $S = S_F$, we recover the semantic Cutting Planes proof system. Indeed, ltf-dag(S_F) equals cut(F), the length required to refute F in semantic Cutting Planes.

3.2 Simplex-dags

We now introduce a new type of dag, for which our dag-like lifting theorem is formulated (Section 4). Let $k \geq 1$ and consider a fixed k-partite input domain $\mathcal{I} := \mathcal{I}_1 \times \cdots \times \mathcal{I}_k$. We say that a function $f : \mathcal{I}_1 \times \cdots \times \mathcal{I}_k \to \{0,1\}$ is *monotone* (up to an ordering of the parts $\mathcal{I}_i$; aka *unate*) iff each set $\mathcal{I}_i$ admits a total order $\preceq_i$ such that $f(x) \leq f(y)$ for every pair $x \preceq y$ (meaning $x_i \preceq_i y_i$ for all $i \in [k]$). For example, every n-bit LTF is monotone as an n-partite function: the orderings are determined by the signs of the coefficients appearing in the linear form defining f. We also say that a subset $A \subseteq \mathcal{I}_1 \times \cdots \times \mathcal{I}_k$ is a *(combinatorial) k-simplex* if its indicator function is monotone. Let $\mathcal{F}$ be the set of monotone functions over $\mathcal{I}_1 \times \cdots \times \mathcal{I}_k$; we emphasize that any two $f, f' \in \mathcal{F}$ may not agree on the ordering of any part $\mathcal{I}_i$. We call such $\mathcal{F}$-dags simply *simplex-dags*, and define

$$\text{sim-dag}(S) \;:=\; \text{least } \textit{size} \text{ of a simplex-dag solving } S.$$

Relation to other models. Simplex-dags are a natural k-party generalization of the bipartite case $k = 2$, which was called *triangle-dags* in [GGKS18]. Triangle-dags in turn are equivalent to real circuits and real dag-like protocols [HC99, Pud97, HP18]. Our motivation to consider multi-party models is that they can be vastly weaker than two-party models. Hence one expects it to be easier to prove lower bounds for k-simplex-dags when k is large. For a toy example, consider the n-bit $\textsc{Xor}_n$ function. It is easy to compute for traditional two-party communication protocols regardless of how the n bits are split between the two players. By contrast, for n parties, each holding one input bit, $\textsc{Xor}_n$ is hard to compute.

3.3 Relationships

The complexity measures introduced so far are related as follows:

$$\text{sim-dag}(S_k) \;\leq\; \text{ltf-dag}(S_n) \;\leq\; \text{dec-dag}(S_n) \;\leq\; n^{O(\text{w}(S_n))}.$$

Here $S_n \subseteq \{0,1\}^n \times O$ is any n-bit search problem, and $S_k \subseteq \{0,1\}^{I_1} \times \cdots \times \{0,1\}^{I_k} \times O$ is a k-partite version of S_n obtained from an arbitrary partition $I_1 \sqcup \cdots \sqcup I_k = [n]$. The first inequality follows by noting that each LTF f, defined by $\sum_i a_i x_i \geq a_{n+1}$, is a monotone k-partite function when the i-th part $\{0,1\}^{I_i}$ is ordered according to the partial sum $\sum_{i\in I_i} a_i x_i$ (breaking ties arbitrarily). The second inequality follows since every conjunction is an LTF. The last inequality is standard: the length of any width-w Resolution refutation can be made $n^{O(w)}$ by eliminating repeated clauses (and the same construction works for arbitrary search problems).

3.4 Blocks

Block width. Let $S \subseteq (\{0,1\}^\ell)^n \times O$ be any search problem whose $n\ell$ input bits are partitioned into n blocks of ℓ bits each. For every conjunction C over the variables of S, we define the *block-width* of C as the maximum number of blocks that C *touches*, that is, contains a variable (or its negation) from a block. We define the *block-width* of a decision-dag solving S as the maximum block-width over all conjunctions in the dag. Finally, we define

$$\text{bw}(S) \;:=\; \text{least } \textit{block-width} \text{ of a decision-dag solving } S.$$

Block composition. The *column-index* gadget $\textsc{Ind}_{\ell\times m}$ is defined as $\textsc{Ind}_{\ell\times m}(x,y)$ is the "x-th column of y" where $x \in [m]$ and $y \in \{0,1\}^{\ell\times m}$. We call $y \in \{0,1\}^{\ell\times m}$ the *matrix* and $x \in [m]$ the *pointer* (for decision-dags, we tacitly encode the elements of $[m]$ in binary as $\log m$-bit strings.). Letting $S \subseteq (\{0,1\}^\ell)^n \times O$ be as above, we define a composed search problem

$$S \circ \textsc{Ind}^n_{\ell\times m} \;\subseteq\; [m]^n \times (\{0,1\}^{\ell\times m})^n \times O. \tag{2}$$

Namely, on input $(x,y) \in [m]^n \times (\{0,1\}^{\ell\times m})^n$ the goal is to find an output $o \in S(z)$ for z obtained by applying the gadget on input (x,y) as follows:

$$z := (\textsc{Ind}_{\ell\times m}(x_1,y_1), \ldots, \textsc{Ind}_{\ell\times m}(x_n,y_n)) \in (\{0,1\}^\ell)^n$$

. We shall view the composition (2) as an $(1 + n\ell)$-partite search problem by repartitioning the input domain as

$$[m]^n \times (\{0,1\}^{\ell\times m})^n \;=\; \mathcal{X} \times \prod_{(i,j)\in[n]\times[\ell]} \mathcal{Y}^{ij}$$

where $\mathcal{X} := [m]^n$ and $\mathcal{Y}^{ij} := \{0,1\}^m$. Here we think of player *Alice* as holding $x \in \mathcal{X}$, and for $(i,j) \in [n] \times [\ell]$, player *Bob*^{ij} as holding $(y_i)_j \in \mathcal{Y}^{ij}$, that is, the j-th row of the i-th matrix y_i.

3.5 CNF Encoding

We just defined block-composed search problems $S \circ \textsc{Ind}^n_{\ell\times m}$, but how can we translate such objects back to CNF formulas? The standard recipe is as follows. Fix any search problem $S \subseteq \{0,1\}^n \times O$ (not necessarily of a composed form). A *certificate* for $(x,o) \in S$ is a partial assignment $\rho \in \{0,1,*\}^n$ consistent with x such that for any y consistent with ρ we have $(y,o) \in S$. The size of ρ is the number of its fixed (non-$*$) coordinates. The *certificate complexity* of S is the maximum over all inputs $x \in \{0,1\}^n$ of the minimum over all $o \in S(x)$ of the least size of a certificate for (x,o). For example, if F is an unsatisfiable k-CNF formula, then S_F has certificate complexity at most k. Conversely, any total search problem S of certificate complexity k contains the search problem S_F associated with some unsatisfiable k-CNF formula F as a subproblem (S is at least as hard as S_F). Namely, consider $F := \bigwedge_x \neg C_x$ where C_x is the conjunction that checks if the input is consistent with some fixed size-k certificate for x. Note that F is unsatisfiable because S is total.

For unbounded-width CNF formulas (such as Atserias–Müller's Ref(F)), we need to interpret the above recipe with care. Indeed, fix any unsatisfiable (unbounded-width) CNF formula F with $|F|$ many clauses and such that its $n\ell$ variables are partitioned into n blocks of ℓ variables each. Denote by b the maximum block-width of a clause of F. Then every clause D of F gives rise to a family of certificates for $S_F \circ \textsc{Ind}^n_{\ell\times m}$. Namely, a certificate in the family for D consists of at most $b \log m$ bits (reading b many pointer values associated with the blocks of D) together with $|D|$ many bits read from the pointed-to columns. Thus, altogether, we get at most $|F|m^b$ many certificates, at least one for each input to $S_F \circ \textsc{Ind}^n_{\ell\times m}$. We define $F \circ \textsc{Ind}^n_{\ell\times m}$ as the formula obtained by listing all these

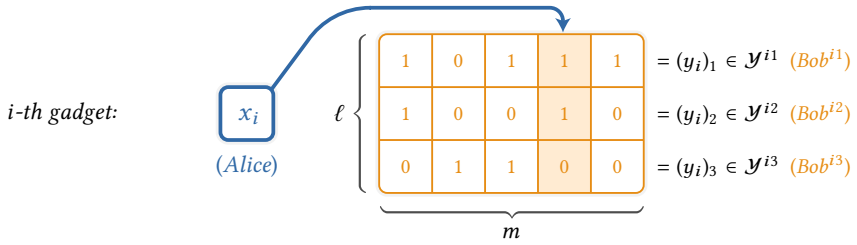

Figure 1: Illustration of the *column-index* gadget

certificates (more precisely, the disjunctions that are the negations of the certificates).

The formula Ref(F) of Atserias and Müller is such that its clauses have block-width 3 [AM19, Appendix A]. Hence $\mathrm{Ref}(F) \circ \mathrm{IND}^n_{\ell\times m}$ has size $n^{O(1)}$ and moreover it is polynomial-time constructible.

Fact 7. *Given an n-variate 3-CNF F, we can construct* $\mathrm{Ref}(F) \circ \mathrm{IND}^n_{\ell\times m}$ *in polynomial time.*

4 DAG-LIKE LIFTING

The purpose of this section is to prove our block-lifting theorem (Theorem 4), which would complete the proof of our main dag-like result (Theorem 1). We restate the block-lifting theorem using the search-problem-centric language of Section 3. Then Theorem 4 is the special case $S := S_F$.

THEOREM 8 (BLOCK LIFTING). *Let $S \subseteq (\{0,1\}^\ell)^n \times O$ be any search problem. For $m := (n\ell)^5$ we have*

$$m^{\Omega(\mathrm{bw}(S))} \le \text{sim-dag}(S \circ \mathrm{IND}^n_{\ell\times m}) \le \text{dec-dag}(S \circ \mathrm{IND}^n_{\ell\times m}) \le m^{O(\mathrm{bw}(\Pi))} \cdot |\Pi|,$$

where Π is any decision-dag solving S of size $|\Pi|$ and block-width bw(Π).

Upper bound. The last inequality is the trivial part of Theorem 8. We only sketch it here. Given a decision-dag Π for S, we construct a decision-dag Π' for $S \circ \mathrm{IND}^n_{\ell\times m}$. For every block-width-$b$ conjunction C in Π, there corresponds a family of exactly m^b many conjunctions in Π'. Namely, the family is constructed by replacing each positive literal x_{ij} (resp. negative literal $\bar{x}_{ij}$) of C with a sequence of $\log m + 1$ many literals that witness the j-th output bit of the i-th gadget being 1 (resp. 0). If C has children C', C'' that only touch blocks touched by C, then every conjunction in the family for C can be directly connected to the families of C', C''. However, if C', C'' touch some block i (there can be at most one) that is untouched by C, then the family for C is connected to the families of C', C'' via decision trees that query the pointer value of the i-th gadget. We have $|\Pi'| \le m^{O(\mathrm{bw}(\Pi))} \cdot |\Pi|$, as desired.

Lower bound. The first inequality is the nontrivial part of Theorem 8. Our proof follows closely the plan from [GGKS18]. However, the proof here is in many ways simpler than the original one. The reason is that we work with multi-party objects (high-dimensional boxes and simplices) rather than two-party objects (rectangles and triangles). For example, one of the key technical lemmas, Lemma 9 ("ρ-structured boxes are ρ-like"), admits a short proof in our multi-party setting, whereas the original lemma for two parties required a long proof involving Fourier analysis. The rest of this section is concerned with proving the simplex-dag lower bound.

4.1 Subcubes from Simplices

Let $\rho \in (\{0,1\}^\ell \cup \{*\})^n$ be a partial assignment that assigns each of the n blocks either an ℓ-bit string or the star symbol. We denote by $\mathrm{free}(\rho) \subseteq [n]$ the subset of blocks assigned a star, and define $\mathrm{fix}(\rho) := [n] \setminus \mathrm{free}(\rho)$. The subcube of strings consistent with ρ is $\mathrm{Cube}(\rho) := \{z \in (\{0,1\}^\ell)^n : z_i = \rho_i, \forall i \in \mathrm{fix}(\rho)\}$. For any set $R \subseteq \mathcal{X} \times \prod \mathcal{Y}^{ij}$ we say that

$$\text{"R is ρ-like"} \quad \text{iff} \quad \mathrm{IND}^n_{\ell\times m}(R) = \mathrm{Cube}(\rho).$$

We formulate a sufficient condition for R to be ρ-like in case R is a *box*, that is a product set.

Definition 1 (Random variables)**.** For a random variable $\boldsymbol{x} \in \mathcal{X}$ we define its *min-entropy* as follows: $\mathbf{H}_\infty(\boldsymbol{x}) := \min_x \log(1/\mathbf{Pr}[\,\boldsymbol{x} = x\,])$. When $\boldsymbol{x}$ is chosen from a set $\mathcal{X}^k$ that is partitioned into k blocks, we define its *blockwise min-entropy* by $\min_{\emptyset\neq S\subseteq[k]} \frac{1}{|S|}\mathbf{H}_\infty(\boldsymbol{x}_S)$ where $\boldsymbol{x}_S$ is the marginal distribution of $\boldsymbol{x}$ over blocks S. We also define the *deficiency* of $\boldsymbol{x} \in \mathcal{X}$ by $\mathbf{D}_\infty(\boldsymbol{x}) := \log|\mathcal{X}| - \mathbf{H}_\infty(\boldsymbol{x}) \ge 0$. For convenience, if X is a set, we denote by $\boldsymbol{X} \in X$ the random variable that is uniform over X. In particular, for $X \subseteq \mathcal{X}^n$ the notation $\boldsymbol{X}_I$ for $I \subseteq [n]$ means "the marginal distribution over coordinates I of the uniform distribution over X". We use $X_I := \{x_I : x \in X\}$ to mean the set that is the projection of X onto coordinates I; thus X_I is the support of $\boldsymbol{X}_I$.

Definition 2 (Structured boxes)**.** Let $R := X \times \prod_{ij} Y^{ij} \subseteq \mathcal{X} \times \prod_{ij} \mathcal{Y}^{ij}$ be a box and $\rho \in (\{0,1\}^\ell \cup \{*\})^n$ a partial assignment. We say R is *ρ-structured* if

(1) Gadgets are fixed according to ρ: $\mathrm{IND}^{\mathrm{fix}(\rho)}_{\ell\times m}(R_{\mathrm{fix}(\rho)}) = \{\rho_{\mathrm{fix}(\rho)}\}$.
(2) X has entropy on the free blocks: $\boldsymbol{X}_{\mathrm{free}(\rho)}$ has blockwise min-entropy $\ge 0.9 \cdot \log m$.
(3) Y^{ij} are large: $\mathbf{D}_\infty(\boldsymbol{Y}^{ij}) \le m^{1/2}$ for $i \in \mathrm{free}(\rho)$, $j \in [\ell]$.

The following key lemma is the reason our dag-lifting result is formulated for k-simplex-dags for large k—we do not know how to prove a multi-output gadget lemma like this for $k = 2$. (The paper [GGKS18] did it for $k = 2$ and single-output gadgets.)

Lemma 9. *Let $R := X \times \prod_{ij} Y^{ij}$ be ρ-structured. There is an $x \in X$ so that $\{x\} \times \prod_{ij} Y^{ij}$ is ρ-like.*

Proof. Assume for simplicity that $\rho = *^n$. Thus our goal is to find an $x \in X$ such that $\text{Ind}^n_{\ell\times m}(\{x\} \times \prod_{ij} Y^{ij}) = (\{0,1\}^\ell)^n$. The key observation is that since each of the $n\ell$ output bits is determined by a different Bob^{ij}, the output bits are independent: $\text{Ind}^n_{\ell\times m}(\{x\} \times \prod_{ij} Y^{ij}) = \prod_{ij} \text{Ind}_{1\times m}(\{x_i\} \times Y^{ij})$. Therefore it suffices to find an $x \in X$ such that for all $i \in [n], j \in [\ell]$,

$$x \text{ is "good" for } Y^{ij}: \qquad \text{Ind}_{1\times m}(\{x_i\} \times Y^{ij}) = \{0,1\}. \tag{3}$$

We claim that a uniform random choice $\boldsymbol{x} \in X$ satisfies all conditions (3) with positive probability. Indeed, for a fixed ij, how many "bad" values $x_i \in [m]$ are there that fail to satisfy (3)? Each bad value x_i implies that the x_i-th bit is fixed in Y^{ij}. But there can be at most $\mathbf{D}_\infty(Y^{ij}) \le m^{1/2}$ fixed such bits. Using $\mathbf{H}_\infty(\boldsymbol{x}_i) \ge 0.9 \cdot \log m$ for $i \in [n]$ and recalling that $m = (n\ell)^5$ we have

$$\mathbf{Pr}[\, \boldsymbol{x}_i \text{ is "bad" for } Y^{ij} \,] \le m^{1/2} \cdot 2^{-0.9 \log m} < 1/(n\ell).$$

A union bound over all the $n\ell$ many conditions (3) completes the proof. □

The following lemma is the culmination of this subsection: Every simplex can be partitioned into ρ-like pieces (and some error sets); see Figure 2. The lemma is a high-dimensional analogue of the *Triangle Lemma* from [GGKS18]. We defer the proof to Appendix A.

Simplex Lemma. *Let $T \subseteq \mathcal{X} \times \prod_{ij} \mathcal{Y}^{ij}$ be a simplex and $k \ge 0$ an error parameter. There exists a disjoint box covering $\bigsqcup_r R^r \supseteq T$ and error sets $X^{\text{err}} \subseteq \mathcal{X}$, $Y^{\text{err},ij} \subseteq \mathcal{Y}^{ij}$, each of density $\le 2^{-k}$, such that for each r one of the following holds:*

- ***Structured case:*** *R^r is ρ^r-structured for some ρ^r that fixes $O(k/\log m)$ blocks. Moreover there exists an "inner" box $R^{\circ,r} \subseteq T \cap R^r$, which is also ρ^r-structured.*
- ***Error case:*** *R^r is covered by error boxes: $R^r \subseteq X^{\text{err}} \times \prod_{ij} \mathcal{Y}^{ij} \cup \bigcup_{ij} \mathcal{X} \times Y^{\text{err},ij} \times \prod_{i'j' \ne ij} \mathcal{Y}^{i'j'}$.*

*Finally, a **query alignment** property holds: for every $x \in \mathcal{X} \setminus X_{\text{err}}$, there exists a subset $I_x \subseteq [n]$ with $|I_x| \le O(k/\log m)$ such that every "structured" R^r intersecting $\{x\} \times \prod_{ij} \mathcal{Y}^{ij}$ has $\text{fix}(\rho^r) \subseteq I_x$.*

4.2 Simplified Proof

To prove (the first inequality of) Theorem 8, fix a simplex-dag Π solving $S \circ \text{Ind}^n_{\ell\times m}$ of size m^d. Our goal is to construct a decision-dag Π' solving S that has block-width $O(d)$. We first present the proof under a simplifying assumption and then remove that assumption in Section 4.3.

(∗) *Assumption:* If we apply Simplex Lemma for $k := 2d \log m$ to any simplex T in Π, then each part in the produced partition $T = \bigsqcup_r T \cap R^r$ satisfies the "structured case".

Using (∗), apply Simplex Lemma (for the above choice of k) to partition all simplicies T in Π. Each resulting structured part $T \cap R^r$ will correspond to a vertex in Π' associated with the partial assignment (or conjunction) ρ^r, that is, with feasible set $\text{Cube}(\rho^r)$. Moreover, we will let the type (root/internal/leaf) of a vertex T in Π dictate the type of the resulting vertices $T \cap R^r$ in Π'. We will add more vertices to Π' shortly in order to connect all the internal vertices, but so far Π' already meets the *root* and *leaf* conditions of a decision-dag solving S, as we note next.

Step 1: Root and leaves. We may assume that for the root of Π, which is associated with the simplex $T := \mathcal{X} \times \prod_{ij} \times \mathcal{Y}^{ij}$, the Simplex Lemma produces the trivial partition consisting of just one $*^n$-structured part, T itself. Hence, the designated root of Π' is defined as the sole part T with an associated feasible set $\text{Cube}(*^n) = (\{0,1\}^\ell)^n$. This meets the *root* condition of a decision-dag.

Consider any part $R^{\circ,r} \subseteq T \cap R^r \subseteq R^r$ with an associated assignment ρ^r, arising from a *leaf* T of Π. Suppose $o \in \boldsymbol{O}$ is a valid solution for T in Π, that is, $T \subseteq (S \circ \text{Ind}^n_{\ell\times m})^{-1}(o)$, or equivalently, $\text{Ind}^n_{\ell\times m}(T) \subseteq S^{-1}(o)$. We claim that o is also a valid solution for the leaf $T \cap R^r$ in Π':

$$\text{Cube}(\rho^r) = \text{Ind}^n_{\ell\times m}(T \cap R^r) \subseteq \text{Ind}^n_{\ell\times m}(T) \subseteq S^{-1}(o).$$

Here the equality uses the fact that $T \cap R^r$ is ρ^r-like (it is sandwiched between two sets that are ρ^r-structured, and hence ρ^r-like by Lemma 9). This meets the *leaf* condition of a decision-dag.

Step 2: Internal. To complete the definition of Π', consider a vertex associated with some part $R^\circ \subseteq T \cap R \subseteq R$, where R° and R are ρ-structured, that arises from a *non-leaf* simplex T of Π. We connect this vertex to the vertices arising from T's two children, L and L'. The connections are made via a *decision tree* $\mathcal{T}$, which we include in Π'. At a high level, the tree will satisfy the following.

(1) *Root:* The root of the tree $\mathcal{T}$ is identified with the vertex $T \cap R$ associated with ρ. That is, $\mathcal{T}$ starts out with the bits in blocks $\text{fix}(\rho) \subseteq [n]$ already queried.
(2) *Non-leaf:* The non-leaf vertices of $\mathcal{T}$ query more bits, one block at a time.
(3) *Leaf:* Every leaf ρ^* of $\mathcal{T}$ extends some assignment τ that arises from the partitions of the children L, L'. Therefore, in Π', we define ρ^* to have τ as its unique child. (This way, the feasible sets satisfy $\text{Cube}(\rho^*) \subseteq \text{Cube}(\tau)$ as required in a decision-dag.)

The tree $\mathcal{T}$ is defined precisely as follows. Since $R^\circ =: X \times \prod_{ij} Y^{ij}$ is ρ-structured, Lemma 9 produces an $x^* \in X$ such that $\{x^*\} \times \prod_{ij} Y^{ij}$ is ρ-like. Using the query alignment property for L and L', there are subsets $I, I' \subseteq [n]$, $|I \cup I'| \le O(k/\log m) \le O(d)$, such that any structured part in the partitions of L and L' that intersects the slice $\{x^*\} \times \prod_{ij} \mathcal{Y}^{ij}$ has their fixed blocks contained in $I \cup I'$. We let $\mathcal{T}$ query all bits in the blocks $(I \cup I') \setminus \text{fix}(\rho)$ in some order, and make the resulting vertices (having queried all bits in blocks $I \cup I' \cup \text{fix}(\rho)$) the leaves of $\mathcal{T}$.

Claim 10. *Every leaf ρ^* of $\mathcal{T}$ satisfies item (3).*

Proof. Since ρ^* extends ρ, and $\{x^*\} \times \prod_{ij} Y^{ij}$ is ρ-like, there is some $y^* \in \prod_{ij} Y^{ij}$ such that $\text{Ind}^n_{\ell\times m}(x^*, y^*) \in \text{Cube}(\rho^*)$. Since $(x^*, y^*) \in R^\circ \subseteq L \cup L'$, we have $(x^*, y^*) \in L$ or $(x^*, y^*) \in L'$. Suppose wlog that $(x^*, y^*) \in L$. Let $L \cap R'$, where R' is τ-structured, be the unique part of L containing (x^*, y^*). But since $\text{fix}(\tau) \subseteq I \subseteq \text{fix}(\rho^*)$ and both τ and ρ^* agree with $\text{Ind}^n_{\ell\times m}(x^*, y^*)$, we conclude that ρ^* extends τ, as required. □

Efficiency. We remark that all the assignments appearing in Π' have block-width $O(d)$. This holds for the vertices coming from partitioning of the simplicies in Π due to our choice of $k \le O(d \log m)$, and it holds for the vertices in the decision trees as they query at

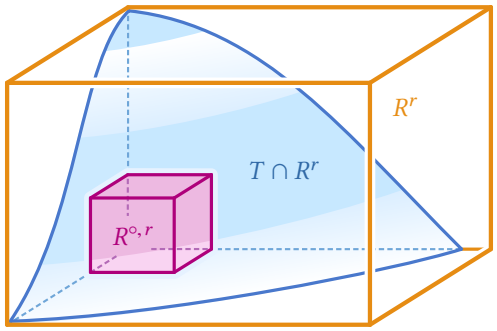

Figure 2: Structured case of Simplex Lemma. The simplex T is partitioned as $T = \bigsqcup_r T \cap R^r$ where each structured part is sandwiched between two ρ^r-structured boxes, $R^{\circ,r} \subseteq T \cap R^r \subseteq R^r$.

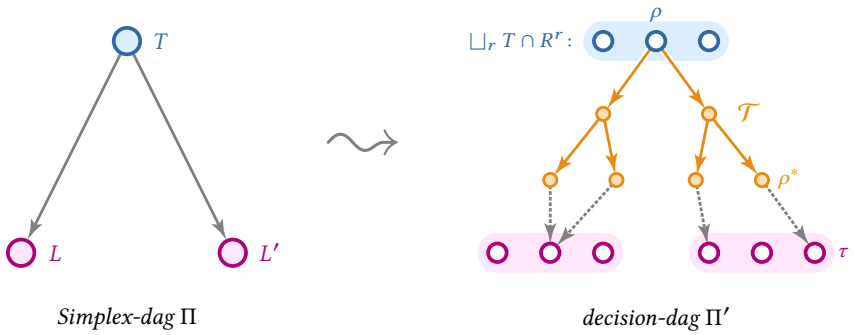

Figure 3: Illustration of turning the Simplex-dag into a decision-dag via a decision tree

most $|I \cup I'| \leq O(d)$ additional blocks. This concludes the (simplified) proof of Theorem 8.

4.3 Accounting for Error

Removing the assumption (∗) is done virtually in the same way as in [GGKS18]. We briefly recall the outline (and refer to [GGKS18, §5.3] for more details if necessary). Instead of partitioning each simplex independently from one another, we instead process them in reverse topological order, $T_1, \ldots, T_{m^d}$ (i.e., if T_i is a descendant of T_j then $i < j$), and before partitioning T_i we first remove all error sets resulting from partitioning of its descendants. More precisely, we initialize an "errorless" box $B := \mathcal{X} \times \prod_{ij} \mathcal{Y}^{ij}$ and then process the simplicies as follows.

Iterate for $i = 1, \ldots, m^d$*:*

(1) Update T_i by removing all the errors accumulated so far: $T_i \leftarrow T_i \cap B$. Note that T_i continues to be a simplex.
(2) Apply Simplex Lemma to obtain a box covering $\bigsqcup_r R^r \supseteq T_i$ with error sets $X^{\text{err}} \subseteq \mathcal{X}$, $Y^{\text{err},ij} \subseteq \mathcal{Y}^{ij}$. Output all the structured parts $R^r \cap T_i$ and discard the error parts.
(3) Update B by removing all the error sets: $B \leftarrow B \setminus (X^{\text{err}} \times \prod_{ij} \mathcal{Y}^{ij} \cup \bigcup_{ij} \mathcal{X} \times Y^{\text{err},ij} \times \prod_{i'j' \neq ij} \mathcal{Y}^{i'j'})$. Note that B continues to be a box.

We can now repeat the simplified proof of Section 4.2 nearly verbatim *using only the structured simplices output by the above process.* When processing Π's root $T_{m^d} \cap B = B$, where $B =: X \times \prod_{ij} Y^{ij}$ is the errorless box at the end of the process, we have that each of X, Y^{ij} has density at least $1 - m^d \cdot 2^{-k} = 1 - m^{-d} \geq 99\%$ by our choice of k. Hence B is $*^n$-structured and we may assume that Simplex Lemma produces the trivial partition for B. This yields the unique root for Π' as before. Another key observation is that the associated errorless box B *grows* when we step from a simplex T_i in Π to either one of its children. Thus every structured part $R^r \cap T_i$ that is output by the above process is wholly covered by the structured parts of T_i's children. This means that our discarding of error sets does not interfere with the construction of the internal trees in Step 2.

A PROOF OF SIMPLEX LEMMA

The proof of Simplex Lemma is a small modification of the proof of the Triangle Lemma in [GGKS18] (the case of 2-dimensional simplices). Since the proof for the latter is somewhat long, we describe here only the required modifications. Our discussion naturally assumes familiarity with the original proof [GGKS18], which analyzed a partitioning procedure called Triangle Scheme (with subroutines Rectangle Scheme and Column Cleanup). The basic difference between the two settings is that instead of partitioning a 2-dimensional simplex over $\mathcal{X} \times \mathcal{Y}$, Bob's input in $\mathcal{Y}$ is further shared over $n\ell$ many Bobs, that is, $\mathcal{Y}$ is replaced with $\prod_{ij} \mathcal{Y}^{ij}$. In this appendix, we explain how to replace all parts involving Bob

with multi-party analogs. There are two: (1) Rectangle Scheme, and (2) Column Cleanup.

(1) Rectangle Scheme. Our first observation is that the Rectangle Scheme, which partitions rectangles $R \subseteq \mathcal{X} \times \mathcal{Y}$, works equally well to partition boxes $B \subseteq \mathcal{X} \times \prod_{ij} \mathcal{Y}$. Indeed, each part output by Rectangle Scheme is obtained from $R := X \times Y$ by restricting the set X arbitrarily and, crucially, restricting Y only via bit-wise restrictions (Round 2 of Rectangle Scheme fixes pointed-to bits in all possible ways). But such bit-wise restrictions when applied to a box $B := X \times \prod_{ij} Y^{ij}$ still result in a box. With this understanding, we may apply Rectangle Scheme to a box.

(2) Column Cleanup. Our biggest modification is to replace the Column Cleanup procedure with a natural multi-party analog. We start with a lemma saying that either a simplex over Bobs' domains contains a box that satisfies the largeness condition of ρ-structuredness (Definition 2), or the simplex can be covered with a small error set.

Claim 11. *Let $T \subseteq \prod_{ij} \mathcal{Y}^{ij}$ be a simplex. Then one of the following holds.*

(i) T contains a box $B := \prod_{ij} Y^{ij}$ where each Y^{ij} has density $\geq 2^{-m^{1/2}}$ (i.e., $\mathbf{D}_\infty(Y^{ij}) \leq m^{1/2}$).

(ii) T is covered by $\bigcup_{ij} Y^{ij,\mathrm{err}} \times \prod_{i'j' \neq ij} \mathcal{Y}^{i'j'}$ where each $Y^{ij,\mathrm{err}}$ has density $\leq 2^{-m^{1/2}}$.

Proof. Consider the largest *cube* $B := \prod_{ij} Y^{ij}$ contained in T, that is, where all the sets $Y^{ij} \subseteq \mathcal{Y}^{ij}$ have the same size. The largest cube can be obtained by the following process: Identify each $\mathcal{Y}^{ij} = \{0,1\}^m$ with $[N]$ according to the reverse of the ordering given to $\mathcal{Y}^{ij}$ by T. (Thus if $x \in T \subseteq [N]^{n\ell}$ and $x' \leq x$ coordinate-wise then $x' \in T$.) Then B equals $[M]^{n\ell}$ where M is the largest number such that $(M, \ldots, M) \in T$. If some (and hence every) Y^{ij} has density $\geq 2^{-m^{1/2}}$ we are in case *(i)*. Otherwise we claim we are in case *(ii)* with $Y^{ij,\mathrm{err}} := Y^{ij}$. Indeed, consider any $x := (M_{11}, \ldots, M_{n\ell}) \in T$. We must have $M_{i^*j^*} \leq M$ for some $(i^*, j^*) \in [n] \times [\ell]$ since otherwise by monotonicity $(M+1, \ldots, M+1) \in T$ contradicting our choice of M. But then $x \in Y^{i^*,j^*} \times \prod_{i'j' \neq i^*j^*} \mathcal{Y}^{i'j'}$, as required. □

We say that a simplex $T \subseteq \prod_{ij} \mathcal{Y}^{ij}$ is *empty-or-heavy* iff $T = \emptyset$ or T satisfies case *(i)* above.

Bob Cleanup

Input: Simplex $T \subseteq \mathcal{X} \times \prod_{ij} \mathcal{Y}^{ij}$
Output: Error sets $Y^{ij,\mathrm{err}} \subseteq \mathcal{Y}^{ij}$ and their combination Y^{err}

1: initialize $Y^{ij,\mathrm{err}} \leftarrow \emptyset$ and write $Y^{\mathrm{err}} := \bigcup_{ij} Y^{ij,\mathrm{err}} \times \prod_{i'j' \neq ij} \mathcal{Y}^{i'j'}$ as a function of the $Y^{ij,\mathrm{err}}$
2: For $I \subseteq [n]$, $\alpha \in [m]^I$, $\gamma \in (\{0,1\}^\ell)^I$, define $Y_{I,\alpha,\gamma} := \left\{ y \in \prod_{ij} \mathcal{Y}^{ij} : g^I(\alpha, y_I) = \gamma \right\}$
3: **while** there are $I, \alpha, \gamma, x \in \mathcal{X}$ s.t. $T' := T \cap (\{x\} \times (Y_{I,\alpha,\gamma} \setminus Y_{\mathrm{err}}))$ is not empty-or-heavy **do** Add to the $Y^{ij,\mathrm{err}}$ all error sets from case *(ii)* for T'
4: **Output** $Y^{ij,\mathrm{err}}$ and Y^{err}

The following claim is the multi party analog of [GGKS18, Claim 10].

Claim 12. *For a simplex $T \subseteq \mathcal{X} \times \prod_{ij} \mathcal{Y}^{ij}$, let $Y^{ij,\mathrm{err}}$, Y^{err} be the outputs of Bob Cleanup. Then:*

- Empty-or-heavy: *For every triple $(I \subseteq [n], \alpha \in [m]^I, \gamma \in (\{0,1\}^\ell)^I)$, and every $x \in \mathcal{X}$, it holds that $T \cap (\{x\} \times (Y_{I,\alpha,\gamma} \setminus Y_{\mathrm{err}}))$ is empty-or-heavy.*
- Size bound: $|Y^{ij,\mathrm{err}}| \leq 2^{m - \Omega(m^{1/2})}$ *for every i, j.*

Proof. The first property is immediate by definition of Bob Cleanup. For the second property, in each while-iteration, at most $2^{m-m^{1/2}}$ elements get added to each $Y^{ij,\mathrm{err}}$. Moreover, there are no more than $2^n \cdot m^n \cdot 2^{n\ell} \cdot m^n = (2m)^{2n\ell}$ choices of I, α, γ, x, and the loop executes at most once for each choice. Thus, $|Y^{ij,\mathrm{err}}| \leq (2m)^{2n\ell} \cdot 2^{m-m^{1/2}} \leq 2^{m-\Omega(m^{1/2})}$. □

This completes the modifications needed to the proof of the Triangle Lemma to handle multiple Bobs.

ACKNOWLEDGEMENTS

We thank Robert Robere for discussions and anonymous STOC reviewers for comments. The first author was supported in part by the NSF grant No. CCF-1412958. The second, third and fourth authors were supported by NSERC, and the fourth author was supported in part by NSF grant No. CCF-1900460.

REFERENCES

[ABMP01] Michael Alekhnovich, Sam Buss, Shlomo Moran, and Toniann Pitassi. Minimum propositional proof length is NP-hard to linearly approximate. *Journal of Symbolic Logic*, 66(1):171–191, 2001. doi:10.2307/2694916.

[AD08] Albert Atserias and Víctor Dalmau. A combinatorial characterization of resolution width. *Journal of Computer and System Sciences*, 74(3):323–334, 2008. doi:10.1016/j.jcss.2007.06.025.

[AM19] Albert Atserias and Moritz Müller. Automating resolution is NP-hard. In *Proceedings of the 60th Symposium on Foundations of Computer Science (FOCS)*, pages 498–509, 2019. doi:10.1109/FOCS.2019.00038.

[AR08] Michael Alekhnovich and Alexander Razborov. Resolution is not automatizable unless W[P] is tractable. *SIAM Journal on Computing*, 38(4):1347–1363, 2008. doi:10.1137/06066850X.

[BCC93] Egon Balas, Sebastián Ceria, and Gérard Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0–1 programs. *Mathematical Programming*, 58(1):295–324, 1993. doi:10.1007/BF01581273.

[BDG+04] Maria Luisa Bonet, Carlos Domingo, Ricard Gavaldà, Alexis Maciel, and Toniann Pitassi. Non-automatizability of bounded-depth Frege proofs. *Computational Complexity*, 13(1-2):47–68, 2004. doi:10.1007/s00037-004-0183-5.

[BEGJ00] Maria Luisa Bonet, Juan Luis Esteban, Nicola Galesi, and Jan Johannsen. On the relative complexity of resolution refinements and cutting planes proof systems. *SIAM Journal on Computing*, 30(5):1462–1484, 2000. doi:10.1137/S0097539799352474.

[BPR97a] Maria Bonet, Toniann Pitassi, and Ran Raz. Lower bounds for cutting planes proofs with small coefficients. *The Journal of Symbolic Logic*, 62(3):708–728, 1997. doi:10.2307/2275569.

[BPR97b] Maria Luisa Bonet, Toniann Pitassi, and Ran Raz. No feasible interpolation for TC^0-frege proofs. In *Proceedings of the 38th Symposium on Foundations of Computer Science (FOCS)*, pages 254–263, 1997. doi:10.1109/SFCS.1997.646114.

[BPR00] Maria Luisa Bonet, Toniann Pitassi, and Ran Raz. On interpolation and automatization for Frege systems. *SIAM Journal on Computing*, 29(6):1939–1967, 2000. doi:10.1137/S0097539798353230.

[BW01] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow—resolution made simple. *Journal of the ACM*, 48(2):149–169, 2001. doi:10.1145/375827.375835.

[CCT87] William Cook, Collette Coullard, and György Turán. On the complexity of cutting-plane proofs. *Discrete Applied Mathematics*, 18(1):25–38, 1987. doi:10.1016/0166-218X(87)90039-4.

[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.

[CFK+19] Arkadev Chattopadhyay, Yuval Filmus, Sajin Koroth, Or Meir, and Toniann Pitassi. Query-to-communication lifting using low-discrepancy gadgets. Technical Report TR19-103, Electronic Colloquium on Computational Complexity (ECCC), 2019. URL: https://eccc.weizmann.ac.il/report/2019/103/.

[CKS90] William Cook, Ravi Kannan, and Alexander Schrijver. Chvátal closures for mixed integer programming problems. *Mathematical Programming*, 47(1):155–174, May 1990. doi:10.1007/BF01580858.

[DR03] Stefan Dantchev and Søren Riis. On relativisation and complexity gap for resolution-based proof systems. In *Proceedings of the 17th International Workshop on Computer Science Logic (CSL)*, pages 142–154. Springer, 2003. doi:10.1007/978-3-540-45220-1_14.

[dRMN+19] Susanna de Rezende, Or Meir, Jakob Nordström, Toniann Pitassi, Robert Robere, and Marc Vinyals. Lifting with simple gadgets and applications to circuit and proof complexity. Technical Report TR19-186, Electronic Colloquium on Computational Complexity (ECCC), 2019. URL: https://eccc.weizmann.ac.il/report/2019/186/.

[dRNV16] Susanna de Rezende, Jakob Nordström, and Marc Vinyals. How limited interaction hinders real communication (and what it means for proof and circuit complexity). In *Proceedings of the 57th Symposium on Foundations of Computer Science (FOCS)*, pages 295–304. IEEE, 2016. doi:10.1109/FOCS.2016.40.

[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:10.1561/0400000086.

[FPPR17] Noah Fleming, Denis Pankratov, Toniann Pitassi, and Robert Robere. Random CNFs are hard for cutting planes. In *Proceedings of the 58th Symposium on Foundations of Computer Science (FOCS)*, 2017. doi:10.2307/2275569.

[Gar19] Michal Garlík. Resolution lower bounds for refutation statements. In *Proceedings of the 44th Mathematical Foundations of Computer Science (MFCS)*, volume 138, pages 37:1–37:13, 2019. doi:10.4230/LIPIcs.MFCS.2019.37.

[GGKS18] Ankit Garg, Mika Göös, Pritish Kamath, and Dmitry Sokolov. Monotone circuit lower bounds from resolution. In *Proceedings of the 50th Symposium on Theory of Computing (STOC)*, pages 902–911. ACM, 2018. doi:10.1145/3188745.3188838.

[GKMP20] Mika Göös, Sajin Koroth, Ian Mertz, and Toniann Pitassi. Automating Cutting Planes is NP-Hard. *Electronic Colloquium on Computational Complexity (ECCC)*, 2020. URL: https://eccc.weizmann.ac.il/report/2020/049/.

[GKRS19] 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)*, pages 38:1–38:19, 2019. doi:10.4230/LIPIcs.ITCS.2019.38.

[GL10] Nicola Galesi and Massimo Lauria. On the automatizability of polynomial calculus. *Theory of Computing Systems*, 47(2):491–506, 2010. doi:10.1007/s00224-009-9195-5.

[GP18] Mika Göös and Toniann Pitassi. Communication lower bounds via critical block sensitivity. *SIAM Journal on Computing*, 47(5):1778–1806, 2018. doi:doi.org/10.1137/16M1082007.

[GPW15] Mika Göös, Toniann Pitassi, and Thomas Watson. Deterministic communication vs. partition number. In *Proceedings of the 56th Symposium on Foundations of Computer Science (FOCS)*, pages 1077–1088. IEEE, 2015. doi:10.1109/FOCS.2015.70.

[GPW17] Mika Göös, Toniann Pitassi, and Thomas Watson. Query-to-communication lifting for BPP. In *Proceedings of the 58th Symposium on Foundations of Computer Science (FOCS)*, pages 132–143, 2017. doi:10.1109/FOCS.2017.21.

[HC99] Armin Haken and Stephen Cook. An exponential lower bound for the size of monotone real circuits. *Journal of Computer and System Sciences*, 58(2):326–335, 1999. doi:10.1006/jcss.1998.1617.

[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.

[HP17] Pavel Hrubes and Pavel Pudlák. Random formulas, monotone circuits, and interpolation. In *Proceedings of the 58th Symposium on Foundations of Computer Science (FOCS)*, pages 121–131, 2017. doi:10.1109/FOCS.2017.20.

[HP18] Pavel Hrubeš and Pavel Pudlák. A note on monotone real circuits. *Information Processing Letters*, 131:15–19, 2018. doi:10.1016/j.ipl.2017.11.002.

[IP01] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

[Iwa97] Kazuo Iwama. Complexity of finding short resolution proofs. In *Mathematical Foundations of Computer Science (MFCS)*, pages 309–318. Springer, 1997. doi:10.1007/BFb0029974.

[KP98] Jan Krajícek and Pavel Pudlák. Some consequences of cryptographical conjectures for s_2^1 and EF. *Information and Computation*, 140(1):82–94, 1998. doi:10.1006/inco.1997.2674.

[Kra97] Jan Krajíček. Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. *Journal of Symbolic Logic*, 62(2):457–486, 1997. doi:10.2307/2275541.

[Kra98] Jan Krajíček. Interpolation by a game. *Mathematical Logic Quarterly*, 44:450–458, 1998. doi:10.1002/malq.19980440403.

[Lin19] Bingkai Lin. A simple gap-producing reduction for the parameterized set cover problem. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 132, pages 81:1–81:15, 2019. doi:10.4230/LIPIcs.ICALP.2019.81.

[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.

[MPW19] Ian Mertz, Toniann Pitassi, and Yuanhao Wei. Short proofs are hard to find. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 132, pages 84:1–84:16. Schloss Dagstuhl, 2019. doi:10.4230/LIPIcs.ICALP.2019.84.

[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.

[Pud97] Pavel Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *The Journal of Symbolic Logic*, 62(3):981–998, 1997. doi:10.2307/2275583.

[Pud00] Pavel Pudlák. Proofs as games. *The American Mathematical Monthly*, 107(6):541–550, 2000. doi:10.2307/2589349.

[Pud10] Pavel Pudlák. On extracting computations from propositional proofs (a survey). In *Proceedings of the 30th Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 8, pages 30–41. Schloss Dagstuhl, 2010. doi:10.4230/LIPIcs.FSTTCS.2010.30.

[Raz95] Alexander Razborov. Unprovability of lower bounds on circuit size in certain fragments of bounded arithmetic. *Izvestiya: Mathematics*, 59(1):205–227, 1995. doi:10.1070/IM1995v059n01ABEH000009.

[RM99] Ran Raz and Pierre McKenzie. Separation of the monotone NC hierarchy. *Combinatorica*, 19(3):403–435, 1999. doi:10.1007/s004930050062.

[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)*, volume 80, pages 80:1–80:13, 2017. doi:10.4230/LIPIcs.ICALP.2017.80.

[Sok17] Dmitry Sokolov. Dag-like communication and its applications. In *Proceedings of the 12th Computer Science Symposium in Russia (CSR)*, pages 294–307. Springer, 2017. doi:10.1007/978-3-319-58747-9_26.

[Urq87] Alasdair Urquhart. Hard examples for resolution. *Journal of the ACM*, 34(1):209–219, 1987. doi:10.1145/7531.8928.