Electronic Colloquium on Computational Complexity, Report No. 6 (2021)

How Limited Interaction Hinders Real Communication (and What It Means for Proof and Circuit Complexity)*

Susanna F. de Rezende
Institute of Mathematics of the Czech Academy of Sciences

Jakob Nordström
University of Copenhagen and Lund University

Marc Vinyals
Technion

January 18, 2021

Abstract

We obtain the first true size-space trade-offs for the cutting planes proof system, where the upper bounds hold for size and total space for derivations with constant-size coefficients, and the lower bounds apply to length and formula space (i.e., number of inequalities in memory) even for derivations with exponentially large coefficients. These are also the first trade-offs to hold uniformly for resolution, polynomial calculus and cutting planes, thus capturing the main methods of reasoning used in current state-of-the-art SAT solvers.

We prove our results by a reduction to communication lower bounds in a round-efficient version of the real communication model of [Krajíček '98], drawing on and extending techniques in [Raz and McKenzie '99] and [Göös et al. '15]. Such lower bounds are in turn established by a reduction to trade-offs between cost and number of rounds in the game of [Dymond and Tompa '85] played on directed acyclic graphs.

As a by-product of the techniques developed to show these proof complexity trade-off results, we also obtain a separation between monotone-AC^{i-1} and monotone-NC^{i}, and an exponential separation between monotone-AC^{i-1} and monotone-AC^{i}, improving exponentially over the superpolynomial separation in [Raz and McKenzie '99].

1 Introduction

Ever since the discovery of NP-completeness by Cook and Levin in [Coo71, Lev73], the problem of how hard it is to decide satisfiability of formulas in propositional logic has played a leading role in theoretical computer science. Although the conventional wisdom is that SAT should be a very hard problem—to the extent that the *Exponential Time Hypothesis* [IP01] concerning its worst-case complexity is a standard assumption used in many other hardness results—essentially no non-trivial lower bounds on the time complexity of the SAT problem are known.

A less ambitious goal is to ask for lower bounds if not only the running time but also the memory usage of the algorithm is restricted. Yet it took until [For00] to rule out a linear-time, logarithmic-space algorithm for SAT. Later research has shown that refuting unsatisfiable formulas on random-access machines cannot be done non-deterministically in simultaneous time $n^{4^{1/3}}$ and space $n^{o(1)}$ [DvMW11] and SAT cannot be decided deterministically in simultaneous time $n^{1.8}$ and space $n^{o(1)}$ [Wil08]. On Turing machines, no non-deterministic algorithm solving SAT in time T and space s can achieve $T \cdot s = n^2 / \log^3 n$ [San01]. (See [vM07] for a good survey of the area with more details on this kind of results.)

For a problem that is believed to require exponential time, the results listed above might not seem very impressive. Yet they should not necessarily be viewed only as an illustration of the weaknesses of current techniques for proving lower bounds. It is important to realize that the adversary is formidable—applied

*A preliminary version [dRNV16] of this work appeared in FOCS 2016.

ISSN 1433-8092

research in the last 15–20 years has led to the development of amazingly efficient algorithms, so-called *SAT solvers*, that solve many real-world instances with millions of variables, and do so in linear time. Today, practitioners often think of SAT as an *easy problem to reduce to*, rather than a hard problem to reduce from (we refer the reader to [BHvMW09] for more on this fascinating topic).

Virtually the only tool currently available for a rigorous analysis of the performance of such SAT solvers is *proof complexity* [CR79], where one studies the methods of reasoning used by the corresponding algorithms. The transcript of the computations made can be viewed as a formal proof applying the relevant method of reasoning, and proof complexity analyses the resources needed when all computational choices are made optimally (i.e., non-deterministically). Even though this is quite a challenging adversarial setting, proof complexity has nevertheless managed to give tight exponential lower bounds on the worst-case running time for many approaches for SAT used in practice by lower-bounding proof size.

The focus of this paper is on time-space trade-offs in computational models describing current state-of-the-art SAT solvers. This research is partly driven by SAT solver running time and memory usage—in practice, space consumption can be almost as much of a bottleneck as running time—but is also motivated by the fundamental importance of time and space complexity in computational complexity.

1.1 Previous Work on Proof Complexity Trade-offs

In *resolution* [Bla37], which is arguably the most well-studied proof system in proof complexity, the input is an unsatisfiable formula in conjunctive normal form (CNF) and new disjunctive clauses are derived from this formula until an explicit contradiction is reached (in the form of the empty clause without literals). Resolution is also the method of reasoning underlying the currently most successful SAT solving paradigm based on so-called *conflict-driven clause learning (CDCL)* [BS97, MS99, MMZ$^+$01]. The question of time-space trade-offs for resolution was first raised by Ben-Sasson in 2002 (journal version in [Ben09]), who also obtained such trade-offs for the restricted subsystem of tree-like resolution. Size-space trade-offs for general, unrestricted resolution were later shown in [Nor09, BN11, BBI16, BNT13].

In contrast to the trade-off results for random-access and Turing machines reviewed above, in these more limited models of computation one can obtain exponential lower bounds on proof size (corresponding to running time) for proofs in sublinear but polynomial space [Nor09, BN11], and results in [BBI16, BNT13] even exhibit trade-offs where size has to be superpolynomial and space has to be superlinear simultaneously. Another difference is that these results are *true trade-offs* in the sense that it is actually possible to refute the formulas both in small size and small space, only not simultaneously. A third nice feature of the trade-offs are that the upper bounds are on proof size and total space, whereas the (sometimes tightly matching) lower bounds are on *length* and *formula space*, meaning that one only charges one time unit for each derivation step regardless of its complexity, and only one space unit per “formula” (for resolution: per clause) regardless of how large it is. Thus, the upper bounds are algorithmically achievable, while the lower bounds hold in a significantly stronger model.

A stronger proof system than resolution is *polynomial calculus* [CEI96, ABRW02], where the clauses of a formula are translated to multilinear polynomials and calculations inside the ideal generated by these polynomials (basically corresponding to a Gröbner basis computation) establishes unsatisfiability. Among other things, polynomial calculus captures CDCL solvers extended with reasoning about systems of linear equations $\bmod 2$. The first size-space trade-offs for polynomial calculus—which were not true trade-offs in the sense discussed above, however—were obtained in [HN12], and these results were further improved in [BNT13] to true trade-offs essentially matching the results cited above for resolution except for a small loss in parameters.

Another proof system that is also stronger than resolution and that has been the focus of much research is *cutting planes* [CCT87], which formalizes the integer linear programming algorithm in [Gom63, Chv73] and underlies so-called *pseudo-Boolean* SAT solvers. In cutting planes the clauses of a CNF formula are translated to linear inequalities, which are then manipulated to derive a contradiction. Thus, the question of Boolean satisfiability is reduced to the geometry of polytopes over the real numbers. Cutting planes is much more poorly understood than resolution and polynomial calculus, however, and size-space trade-offs have proven elusive. The results in [HN12] apply not only to resolution and polynomial calculus but also to cutting planes, and were improved further in [GP18] to hold for even stronger proof systems, but unfortunately are not true trade-offs in the sense discussed above.

The problem is that what is shown in [HN12, GP18] is only that proofs in small space for certain formulas have to be very large, but it is not established that these formulas can be refuted space-efficiently. In fact, for resolution it can be shown using techniques from [BN08] that such small-space proofs provably do not exist, and for polynomial calculus there is circumstantial evidence for a similar claim. As discussed in Section 3, this turns out to be an inherent limitation of the technique used.

In a recent surprising paper [GPT15], it was shown that cutting planes can refute any formula in *constant* space if we only count the number of lines or formulas. Plugging this result into [HN12, GP18] yields a trade-off of sorts, since "small-space" proofs will always exist, but the catch is that such proofs will have exponentially large coefficients. This means that these trade-offs do not seem very "algorithmically relevant" in the sense that such proofs could hardly be found in practice, and saying that a proof with exponential-size coefficients has "constant space" somehow does not feel quite right.

1.2 Our Proof Complexity Contributions

In this paper we report the first true, algorithmically realizable trade-offs for cutting planes, where the upper bounds hold for proof size and total space and the lower bounds apply to proof length and formula space (i.e., number of inequalities). The trade-offs also hold for resolution and polynomial calculus, making them the first trade-offs that hold for essentially all methods of reasoning used in the most successful SAT solvers to date.[1]

Below, we state two examples of the kind of trade-offs we obtain (referring the reader to Section 2 for the missing formal definitions). In the rest of this section we will focus on cutting planes, since this proof system is the main target of this work. However, all the lower bounds stated also hold for polynomial calculus (and for the strictly weaker proof system resolution), and since all our upper bounds are actually proven in resolution they transfer to both polynomial calculus and cutting planes.

The first result is a "robust trade-off" that holds all the way from polylogarithmic to polynomial space as stated next.

Theorem 1.1 (Informal). *There exists an explicitly constructible family of* 6*-CNF formulas* $\{F_N\}_{N=1}^{\infty}$ *of size* $\Theta(N)$ *such that:*

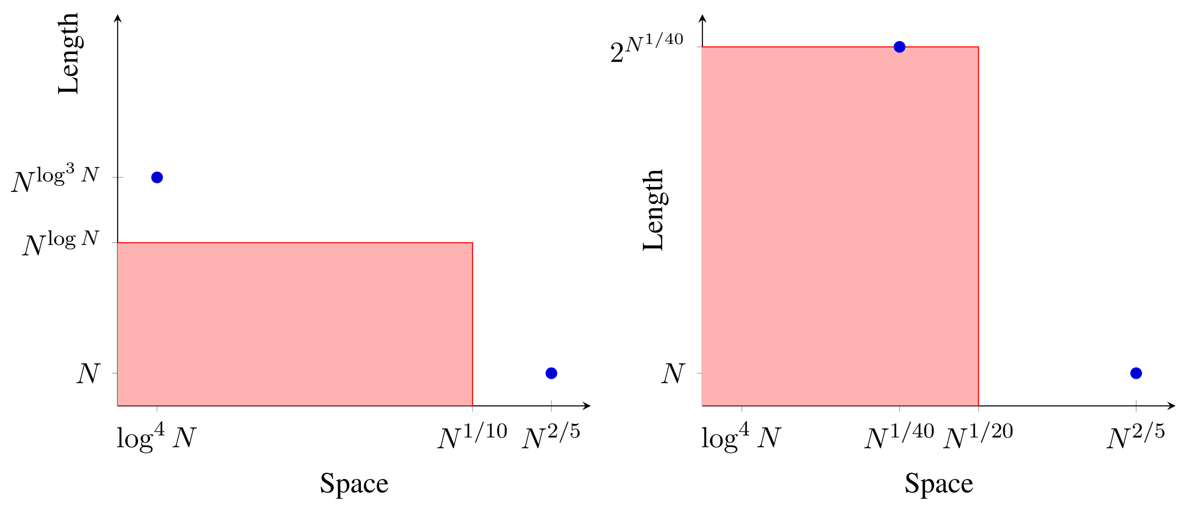
1. F_N *can be refuted by cutting planes with constant-size coefficients in size* $\mathrm{O}(N)$ *and total space* $\mathrm{O}\big(N^{2/5}\big)$.
2. F_N *can be refuted by cutting planes with constant-size coefficients in total space* $\mathrm{O}(\log^4 N)$ *and size* $2^{\mathrm{O}(\log^4 N)}$.
3. *Any cutting planes refutation of* F_N*, even with coefficients of unbounded size, in formula space less than* $N^{1/10-\epsilon}$ *requires length greater than* $2^{\Omega(\log^2 N)}$.

The second trade-off holds over a smaller space range, but causes an exponential and not just superpolynomial blow-up in proof size.

Theorem 1.2 (Informal). *There exists an explicitly constructible family of* 6*-CNF formulas* $\{F_N\}_{N=1}^{\infty}$ *of size* $\Theta(N)$ *such that:*

1. F_N *can be refuted by cutting planes with constant-size coefficients in size* $\mathrm{O}(N)$ *and total space* $\mathrm{O}\big(N^{2/5}\big)$.
2. F_N *can be refuted by cutting planes with constant-size coefficients in total space* $\mathrm{O}\big(N^{1/40}\big)$ *and size* $2^{\mathrm{O}(N^{1/40})}$.

[1]We remark that this ignores the issue of formula *preprocessing techniques*, which are heavily used in most state-of-the-art SAT solvers, and some of which potentially require the full extended Frege proof system for a complete formal description (but can also sometimes cause a provable exponential *loss* in reasoning power). Since in practice SAT solvers fail to solve many of the combinatorial benchmark formulas that are hard for resolution, polynomial calculus, and cutting planes but easy for (even non-extended) Frege, however, and since in addition it is usually not hard to come up with formulas that foil any concrete preprocessing techniques actually used, this seems like a reasonable simplification.



(a) Robust trade-off **(b)** Exponential trade-off

Figure 1: Pictorial illustrations of trade-offs in Theorems 1.1 and 1.2

3. *Any cutting planes refutation of F_N, even with coefficients of unbounded size, in formula space less than $N^{1/20-\epsilon}$ requires length greater than $2^{\Omega(N^{1/40})}$.*

See Figure 1 for an illustration of these results, where blue dots denote provable upper bounds on time-space parameters of cutting planes refutations and the shaded red areas show ranges of parameters that are impossible to achieve.

1.3 Previous Work in Monotone Circuit Complexity

Since this paper also makes contributions to monotone circuit complexity, we next review some relevant background in this area. After superpolynomial lower bounds on the size of monotone circuits computing explicit functions were obtained in [Raz85, And85] (see also [BS90]), the first step towards the natural next goal of establishing a depth hierarchy for monotone circuits was taken in [KW90], proving that connectivity, which is in monotone-NC^2, requires depth $\Omega(\log^2 n)$ for monotone circuits with fan-in 2. This implies a separation between monotone-NC^1 and monotone-NC^2. The same approach was used in [RM99] to prove a separation between monotone-NC^{i-1} and monotone-NC^i for every i. This result can be rephrased as saying that there is a family of Boolean functions $\{f^i\}$ such that f^i can be computed by monotone circuits of depth $\log^i n$, fan-in 2, and polynomial size but cannot be computed by any monotone circuit of depth $\mathrm{o}(\log^i n)$ and fan-in 2.

Going into more details, the function in [RM99] that witnesses the separation between monotone-NC^{i-1} and monotone-NC^i can be computed by a monotone circuit of depth $\log^{i-1} n$, polynomial fan-in, and polynomial size, and therefore the separation is between monotone-NC^{i-1} and monotone-AC^{i-1}. This separation was later refined in [Joh01] to circuits of semi-unbounded fan-in—i.e., with AND-gates of fan-in 2 and OR-gates of unbounded fan-in—leaving the question of a separation between monotone-AC^{i-1} and monotone-NC^i open.

We can also view the separation between monotone-NC^i and monotone-AC^i as a separation between monotone-AC^{i-1} and monotone-AC^i, since monotone-AC^{i-1} is contained in monotone-NC^i; however, this separation only guarantees a superpolynomial circuit size lower bound. Furthermore, the function f^i only depends on $\log^{40i} n$ variables and so it can be computed by a monotone DNF of size $2^{\log^{40i} n}$, i.e., there is a quasipolynomial upper bound.

We remark that it is not possible to prove more than a quasipolynomial separation between monotone-NC^{i-1} and monotone-NC^i in view of the simple fact that circuits in these classes have quasipolynomial size, and hence it only makes sense to talk about superpolynomial versus exponential separations in the monotone-AC hierarchy. It should be noted that exponential separations between monotone circuits of bounded depth were previously known, but only for depth less than logarithmic. It was shown in [KPPY84] that the complete tree of depth k, arity $n^{1/k}$, and size $\Theta(n)$, with alternating levels of AND and OR, requires size $2^{\Omega(n^{1/k}/k)}$ to compute with circuits of depth $k-1$. This result was later reproven in [NW93] using the communication complexity of the pointer jumping function (see also [RY20]).

1.4 Our Monotone Circuit Complexity Contributions

We solve the open problem of [Joh01] mentioned above by separating monotone-AC^{i-1} from monotone-NC^{i}.

Theorem 1.3. *For every $i \in \mathbb{N}$ there is a Boolean function over n variables that can be computed by a monotone circuit of depth $\log^i n$, fan-in 2, and size $\mathrm{O}(n)$, but for which every monotone circuit of depth $\mathrm{o}(\log^i n/(\log\log n)^2)$ requires superpolynomial size.*

This result allows us to complete the picture of the three interwoven monotone circuit depth hierarchies, which looks like

$$\text{monotone-NC}^{i-1} \subsetneq \text{monotone-SAC}^{i-1} \subsetneq \text{monotone-AC}^{i-1} \subsetneq \text{monotone-NC}^{i} \ , \tag{1.1}$$

the first non-inclusion being proven in [RM99], the second in [Joh01], and the third in the present work.

We note that in terms of depth our separation is close to optimal. Indeed, it is not hard to see that if a Boolean function can be computed by a monotone circuit of depth d, fan-in 2 and size s, then it can also be computed by a monotone circuit of depth d/k, unbounded fan-in and size $s \cdot 2^{2^k}$. Therefore, in order to have a superpolynomial separation it must be the case that $k \geq \log\log n + \omega(1)$, and the theorem above holds for any $k = \omega((\log\log n)^2)$.

In addition we establish an exponential separation in the monotone-AC hierarchy. More precisely, for each $i \in \mathbb{N}$ we exhibit a Boolean function f^i that can be computed by monotone circuits of depth $\log^i n$ but such that every monotone circuit of depth at most $\mathrm{O}(\log^{i-1} n)$ requires size $2^{n^{\Omega(1)}}$ (where the hidden constant in the lower bound depends inversely on that in the upper bound).

Theorem 1.4. *For every $i \in \mathbb{N}$ there is a Boolean function over n variables that can be computed by a monotone circuit of depth $\log^i n$, fan-in $n^{4/5}$, and size $\mathrm{O}(n)$, but for which every monotone circuit of depth $q \log^{i-1} n$ requires size $2^{\Omega(n^{1/(10+4\epsilon)q})}$.*

1.5 Discussion of Techniques

Let us now briefly discuss the techniques we use to establish the above results, focusing for concreteness on Theorems 1.1 and 1.2. These theorems are proven by a careful chain of reductions as follows.

1. Our first step is to use the connection made explicit in [HN12], and also used in [GP18], that short and space-efficient proofs for a CNF formula F can be converted to efficient communication protocols for the *falsified clause search problem* for F. Going beyond [HN12, GP18], however, we make the simple but absolutely crucial additional observation that protocols obtained in this way are also round-efficient. Furthermore, in contrast to [HN12, GP18] we do not study randomized communication, but instead focus on the *real communication model* introduced by Krajíček [Kra98] with the purpose of getting a tighter correspondence with cutting planes.

2. We next generalize the communication-to-decision-tree simulation theorem for *composed search problem* in the celebrated paper by Göös et al. [GPW18] to the real communication model, and then extend it further to be able to handle rounds using the *parallel decision trees* introduced by Valiant [Val75]. This part is inspired by [BEGJ00], where the simulation theorem in the precursor [RM99] of [GPW18] was proven for real communication but without taking round efficiency into account.

3. To leverage this machinery we need a base CNF search problem, and just as in [BN11, GP18, HN12, BNT13] (and many other papers) the *pebbling formulas* Peb_G from [BW01] turn out to be handy here, provided that they are defined over appropriately chosen directed acyclic graphs G. These formulas are then *lifted* (corresponding to composition of search problems) as described in [BHP10], though the parameters of the lifting are different (and unfortunately significantly worse than in [HN12]).

4. The following step is the relatively straightforward observation that efficient parallel decision trees for formulas Peb_G yield good strategies in the pebble game of Dymond and Tompa [DT85] played on the underlying graph G. At the same time, this is a somewhat unexpected twist, since in previous papers such as [BN08, BN11, BNT13] size and space lower bounds for pebbling formulas always followed from the *black-white pebble game* [CS76] on G, but we cannot make use of that latter game here.

5. Since we have to use the Dymond–Tompa game rather than the black-white pebble game, as a consequence we also have to use different graphs than in [BN11, HN12, BNT13]—in particular, modifying the construction of graphs with good black-white pebbling trade-offs in [LT82]—and as a concluding step we prove Dymond–Tompa trade-offs for these graphs.

Putting all these pieces together, we obtain a general theorem saying that graphs with Dymond–Tompa trade-offs yield explicit 6-CNF formulas with size-space trade-offs for cutting planes (and polynomial calculus and resolution). Theorem 1.4 follows by a similar chain of reductions.

1.6 Paper Outline

The rest of this paper is organized as follows. In Section 2 we give a more detailed overview of the steps in the proofs of our main theorems, introducing formal definitions of the concepts discussed above as need arises. In Section 3 we translate proofs into communication protocols. The heart of the paper is then in Section 4, where we establish that communication protocols for lifted search problems can be simulated by decision trees for the original search problems. In Section 5 we show how decision trees for our search problem for pebbling formulas can be converted to Dymond–Tompa game strategies for the corresponding graphs, and in Section 6 we show Dymond–Tompa trade-offs. After having established the upper bounds needed for our proof complexity trade-offs in Section 7, we put all the pieces together in Section 8. Section 9 then discusses how we can use the same tools to obtain circuit complexity separations. Finally, we make some concluding remarks in Section 10.

2 Preliminaries and Proof Overview

In this section, we describe which components are needed for our results stated in Section 1 and how they fit together. Our goal is to give an accessible high-level outline of the proofs, but still make clear what are the main technical points in the arguments and also indicate some of the challenges that have to be overcome.

Let us start by reviewing the concepts we need from proof complexity. Throughout this paper all logarithms are to base 2 unless otherwise specified, and we write $[n]$ to denote the set $\{1, 2, \ldots, n\}$.

2.1 Proof Complexity Basics and Cutting Planes

For x a Boolean variable, a *literal over* x is either the variable x itself or its negation, denoted $\overline{x}$. It will also be convenient to use the notation $x^1 = x$ and $x^0 = \overline{x}$. A *clause* $C = a_1 \vee \cdots \vee a_k$ is a disjunction of literals and a *CNF formula* $F = C_1 \wedge \cdots \wedge C_m$ is a conjunction of clauses. We will think of clauses and CNF formulas as sets, so that the ordering is inconsequential and there are no repetitions. A k*-CNF formula* is a CNF formula consisting of clauses containing at most k literals.

We write α, β to denote truth value assignments, i.e., functions to $\{0,1\}$, where we identify 0 with false and 1 with true (thus, x^b is the literal satisfied by setting $x = b$). We have the usual semantics that a clause is true under α, or *satisfied* by α, if at least one literal in it is true, and a CNF formula is satisfied if all clauses in it are satisfied. We write $\perp$ to denote the empty clause without literals, which is false under all truth value assignments.

Following [ABRW02, ET01], we view a proof of unsatisifiability of a CNF formula F, or *refutation* of F, as a non-deterministic computation, with a special read-only input tape from which the clauses of the formula F being refuted (which we refer to as *axioms*) can be downloaded and a working memory where all derivation steps are made. In a *cutting planes (CP)* derivation, memory configurations are sets of linear inequalities $\sum_j a_j x_j \geq c$ with $a_j, c \in \mathbb{Z}$. We translate clauses C to linear inequalities $L(C)$ by identifying the clause $\bigvee_j x_j^{b_j}$ with the inequality $\sum_j (-1)^{1-b_j} x_j \geq 1 - \sum_j (1 - b_j)$. A CP refutation of F is a sequence of configurations $(\mathbb{L}_0, \ldots, \mathbb{L}_\tau)$ such that $\mathbb{L}_0 = \emptyset$, the inequality $0 \geq 1$ occurs in $\mathbb{L}_\tau$, and for $t \in [\tau]$ we obtain $\mathbb{L}_t$ from $\mathbb{L}_{t-1}$ by one of the following rules:

Axiom download $\mathbb{L}_t = \mathbb{L}_{t-1} \cup \{L\}$ for L being either the encoding $L(C)$ of an *axiom clause* $C \in F$ or a *variable axiom* $x_j \geq 0$ or $-x_j \geq -1$ for any variable x_j.

Inference $\mathbb{L}_t = \mathbb{L}_{t-1} \cup \{L\}$ for L inferred by *addition* $\dfrac{\sum_j a_j x_j \geq c \qquad \sum_j b_j x_j \geq d}{\sum_j (a_j + b_j) x_j \geq c + d}$, *multiplication* $\dfrac{\sum_j a_j x_j \geq c}{\sum_j k a_j x_j \geq kc}$, or *division* $\dfrac{\sum_j k a_j x_j \geq c}{\sum_j a_j x_j \geq \lceil c/k \rceil}$ for $k \in \mathbb{N}^+$.

Erasure $\mathbb{L}_t = \mathbb{L}_{t-1} \setminus \{L\}$ for some $L \in \mathbb{L}_{t-1}$.

The *length* of a CP refutation is the number of linear inequalities L appearing in download and inference steps, counted with repetitions. We obtain the *size* of a refutation by also summing the sizes of the coefficients and constant terms in the inequalities, i.e., each inequality $\sum_j a_j x_j \geq c$ contributes $\log|c| + \sum_j \log|a_j|$. The *formula space* of a configuration $\mathbb{L} = \{\sum_j a_{i,j} x_{i,j} \geq c_i \mid i \in [s]\}$ is the number of inequalities s in it, and the *total space* of $\mathbb{L}$ is $\sum_{i \in [s]} \big(\log|c_i| + \sum_j \log|a_{i,j}|\big)$. We obtain the formula space or total space of a refutation by taking the maximum over all configurations in it. Finally, the length, size, formula space, and total space of refuting a formula F is obtained by taking the minimum over all CP refutations of the formula with respect to the corresponding complexity measure.

2.2 Composed Search Problems and Lifted CNF Formulas

Informally speaking, the idea behind *lifting*, or *composition*, is to take a relation over some domain and extend it to tuples from the same domain by combining it with an *indexing* function that determines on which coordinates from the tuples the relation should be evaluated.

Let S be any relation on the Cartesian product $Z \times Q$. We will think of S as a *search problem* with input domain Z and output range Q, where on any input $z \in Z$ the task is to find some $q \in Q$ such that $(z, q) \in S$ (assuming that S is such that there always exists at least one solution). With this interpretation in mind, we write $S(z) = \{q \mid (z, q) \in S\}$. Throughout this paper, we will have $Z = \{0,1\}^m$ for some $m \in \mathbb{N}^+$, so for simplicity we fix Z to be such a domain from now on.

For any $\ell \in \mathbb{N}^+$, we define the *lift of length ℓ of S* to be a new search problem $Lift_\ell(S) \subseteq [\ell]^m \times \{0,1\}^{m \cdot \ell} \times Q$ with input domain $[\ell]^m \times \{0,1\}^{m \cdot \ell}$ and output range Q such that for any $x \in [\ell]^m$, any bit-vector $\{y_{i,j}\}_{i \in [m], j \in [\ell]}$, and any $q \in Q$, it holds that

$$(x, y, q) \in Lift_\ell(S) \quad \text{if and only if} \quad \big((y_{1,x_1}, y_{2,x_2}, \ldots, y_{m,x_m}), q\big) \in S \ . \tag{2.1}$$

In what follows, we will refer to the coordinates of the x-vector as *selector variables* and those of the y-vector as *main variables*, and we will sometimes use the notation

$$\mathsf{Ind}(x_i, y_i) = y_{i,x_i} \tag{2.2}$$

to denote the bit in y_i indexed by x_i. We extend this notation to vectors to write $\mathsf{Ind}(x,y) = y_x = (y_{1,x_1},\ldots,y_{m,x_m})$. Observe that

$$S(\mathsf{Ind}(x,y)) = \{q : (\mathsf{Ind}(x,y),q) \in S\} = \{q : (x,y,q) \in \mathit{Lift}(S)\} = (\mathit{Lift}(S))(x,y) \ . \tag{2.3}$$

As in [HN12, GP18], we obtain our results by studying lifted search problems defined in terms of CNF formulas and proving communication lower bounds for such problems. Syntactically speaking, however, these objects are not themselves CNF formulas, which is what we use to feed to our proof system. Therefore, we need an additional step which translates the lifted search problems back to CNF as follows.

Definition 2.1 (Lifted formula [BHP10]). Given $\ell \in \mathbb{N}^+$ and a CNF formula F over variables $u_1,\ldots,u_n$, the *lift of length* ℓ *of* F, denoted $\mathit{Lift}_\ell(F)$, is the formula over variables $\{x_{i,j}\}_{i\in[n],j\in[\ell]}$ (*selector variables*) and $\{y_{i,j}\}_{i\in[n],j\in[\ell]}$ (*main variables*) containing the following clauses:

- For every $i \in [n]$, an *auxiliary clause*

$$x_{i,1} \lor x_{i,2} \lor \cdots \lor x_{i,\ell} \ . \tag{2.4a}$$

- For every clause $C \in F$, where $C = u_{i_1} \lor \cdots \lor u_{i_s} \lor \overline{u}_{i_{s+1}} \lor \cdots \lor \overline{u}_{i_t}$ for some $i_1,\ldots,i_t \in [n]$, and for every tuple $(j_1,\ldots,j_t) \in [\ell]^t$, a *main clause*

$$\overline{x}_{i_1,j_1} \lor y_{i_1,j_1} \lor \cdots \lor \overline{x}_{i_s,j_s} \lor y_{i_s,j_s} \lor \overline{x}_{i_{s+1},j_{s+1}} \lor \overline{y}_{i_{s+1},j_{s+1}} \lor \cdots \lor \overline{x}_{i_t,j_t} \lor \overline{y}_{i_t,j_t} \ . \tag{2.4b}$$

Intuitively, we can think of the selector variables as encoding the vector $x \in [\ell]^m$ in the lifted search problem (2.1). Since $\overline{x}_{i,j} \lor y_{i,j}$ is equivalent to the implication $x_{i,j} \to y_{i,j}$, we can rewrite (2.4b) as

$$(x_{i_1,j_1} \to y_{i_1,j_1}) \lor \cdots \lor (x_{i_t,j_t} \to \overline{y}_{i_t,j_t}) \ , \tag{2.5}$$

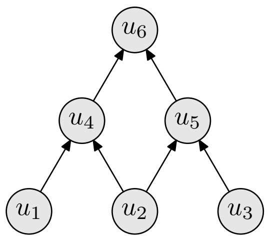
from which we can see that for every clause C the auxiliary clauses encode that there is at least one choice of selector variables $x_{i,j}$ which are all true, and for this choice of selector variables the $y_{i,j}$-variables in the lifted main clause will play the role of the u_i-variables, giving us back the original clause C. It is easily verified that F is unsatisfiable if and only if $H = \mathit{Lift}_\ell(F)$ is unsatisfiable, and that if F is a k-CNF formula with m clauses, then H is a $\max(2k,\ell)$-CNF formula with at most $m\ell^k + n$ clauses. A small technical issue for us compared to [HN12, GP18] is that $\ell \gg k$ will not be constant, but we can convert the wide clauses in (2.4a) to constant width using extension variables, and so we will just ignore this issue in our proof overview.

2.3 Pebbling Contradictions

An important role in many proof complexity trade-off results is played by so-called *pebbling contradictions*. For our purposes it suffices to say that they are defined in terms of directed acyclic graphs (DAGs) G, where for simplicity we assume that all vertices have indegree 0 or 2. We refer to vertices with indegree 0 as *sources* and assume that there is a unique *sink* vertex with outdegree 0. What the pebbling contradiction over G says is that the sources are true and that truth propagates from predecessors to successors, but that the sink is false. The formal definition follows.

Definition 2.2 (Pebbling contradiction [BW01]). Let G be a DAG with sources S and a unique sink t, and with all non-sources having indegree 2. Then the *pebbling contradiction* over G, denoted Peb_G, is the conjunction of the following clauses over variables $\{v \mid v \in V(G)\}$:

- for every source vertex $s \in S$, a unit clause s (*source axioms*),
- For all non-sources w with immediate predecessors u, v, a clause $\overline{u} \lor \overline{v} \lor w$ (*pebbling axioms*),



(a) Pyramid graph Π_2 of height 2.

$$
\begin{aligned}
& u_1 \\
\land\ & u_2 \\
\land\ & u_3 \\
\land\ & (\overline{u}_1 \lor \overline{u}_2 \lor u_4) \\
\land\ & (\overline{u}_2 \lor \overline{u}_3 \lor u_5) \\
\land\ & (\overline{u}_4 \lor \overline{u}_5 \lor u_6) \\
\land\ & \overline{u}_6
\end{aligned}
$$

(b) Pebbling contradiction Peb_{Π_2}.

Figure 2: Example pebbling contradiction for the pyramid of height 2.

- for the sink t, the unit clause $\overline{t}$ (*sink axiom*).

If G has n vertices, the formula Peb_G is an unsatisfiable 3-CNF formula with $n + 1$ clauses over n variables. For an example of a pebbling contradiction, see the CNF formula in Figure 2b defined in terms of the graph in Figure 2a.

To make the connection back to Definition 2.1, in Figure 3 we present the lift of length 2 of the CNF formula in Figure 2b, with the auxiliary clauses at the top of the left column followed by the main clauses one by one, listed for all tuples of selector indices. We will refer to the main clauses in Figure 3 as source axioms, pebbling axioms, and sink axioms, respectively, when they have been obtained by lifting of the correspondingly named axioms in the pebbling contradiction.

2.4 Real Communication and Falsified Clause Search Problems

For our communication complexity results we study a two-player communication model, referring to the players as *Alice* and *Bob* following tradition. We first briefly discuss the basic deterministic model, and then explain how we need to extend it, directing the reader to [KN97] for any omitted standard communication complexity facts.

In the communication problem of solving a relation $S \subseteq X \times Y \times Q$, Alice is given an input $x \in X$, Bob is given an input $y \in Y$, and they are required to find some q such that $(x, y, q) \in S$ while minimizing the communication between them. A communication protocol for S (or that *solves* S) is a binary tree where Alice and Bob start at the root; every internal node v specifies who is going to speak; the value of the spoken bit, which determines which child v_0 or v_1 to move to, is a function of only the node v and the input x if Alice speaks or y if Bob does; and each leaf is labelled by some q such that for all x, y that lead to this leaf $(x, y, q) \in S$. The cost of a protocol is the maximum number of bits communicated on any input, and the number of rounds is the maximum number of alternations between Alice and Bob speaking.

In order to obtain trade-offs for cutting planes, we need to study the more general *real communication* model in [Kra98], where Alice and Bob interact via a referee, and also introduce the concept of rounds in this model. It is convenient to describe the protocol as a (non-binary) tree, where at node v in the protocol tree Alice and Bob send k_v real numbers $\phi_{v,1}(x), \ldots, \phi_{v,k_v}(x)$ and $\psi_{v,1}(y), \ldots, \psi_{v,k_v}(y)$, respectively, to the referee. The referee announces the results of the comparisons $\phi_{v,j}(x) \leq \psi_{v,j}(y)$ for $j \in [k_v]$ as a k_v-bit binary string, after which the players move to the corresponding child of v indexed by the string. The number of rounds r of a protocol is the depth of the tree and the cost c is the total number of comparisons made by the referee for any input. It is easy to see that this model can simulate standard deterministic communication (for instance, if Alice wants to send a message, she sends the complement of that message to the referee and Bob sends a list of the same length with all entries $1/2$) and is in fact strictly stronger (since equality can be solved with just two bits of communication).

$$\begin{aligned}
&(x_{1,1} \lor x_{1,2})\\
\land\ &(x_{2,1} \lor x_{2,2})\\
\land\ &(x_{3,1} \lor x_{3,2})\\
\land\ &(x_{4,1} \lor x_{4,2})\\
\land\ &(x_{5,1} \lor x_{5,2})\\
\land\ &(x_{6,1} \lor x_{6,2})\\
\land\ &(\overline{x}_{1,1} \lor y_{1,1})\\
\land\ &(\overline{x}_{1,2} \lor y_{1,2})\\
\land\ &(\overline{x}_{2,1} \lor y_{2,1})\\
\land\ &(\overline{x}_{2,2} \lor y_{2,2})\\
\land\ &(\overline{x}_{3,1} \lor y_{3,1})\\
\land\ &(\overline{x}_{3,2} \lor y_{3,2})\\
\land\ &(\overline{x}_{1,1} \lor \overline{y}_{1,1} \lor \overline{x}_{2,1} \lor \overline{y}_{2,1} \lor \overline{x}_{4,1} \lor y_{4,1})\\
\land\ &(\overline{x}_{1,1} \lor \overline{y}_{1,1} \lor \overline{x}_{2,1} \lor \overline{y}_{2,1} \lor \overline{x}_{4,2} \lor y_{4,2})\\
\land\ &(\overline{x}_{1,1} \lor \overline{y}_{1,1} \lor \overline{x}_{2,2} \lor \overline{y}_{2,2} \lor \overline{x}_{4,1} \lor y_{4,1})\\
\land\ &(\overline{x}_{1,1} \lor \overline{y}_{1,1} \lor \overline{x}_{2,2} \lor \overline{y}_{2,2} \lor \overline{x}_{4,2} \lor y_{4,2})\\
\land\ &(\overline{x}_{1,2} \lor \overline{y}_{1,2} \lor \overline{x}_{2,1} \lor \overline{y}_{2,1} \lor \overline{x}_{4,1} \lor y_{4,1})\\
\land\ &(\overline{x}_{1,2} \lor \overline{y}_{1,2} \lor \overline{x}_{2,1} \lor \overline{y}_{2,1} \lor \overline{x}_{4,2} \lor y_{4,2})\\
\land\ &(\overline{x}_{1,2} \lor \overline{y}_{1,2} \lor \overline{x}_{2,2} \lor \overline{y}_{2,2} \lor \overline{x}_{4,1} \lor y_{4,1})\\
\land\ &(\overline{x}_{1,2} \lor \overline{y}_{1,2} \lor \overline{x}_{2,2} \lor \overline{y}_{2,2} \lor \overline{x}_{4,2} \lor y_{4,2})
\end{aligned}$$

$$\begin{aligned}
\land\ &(\overline{x}_{2,1} \lor \overline{y}_{2,1} \lor \overline{x}_{3,1} \lor \overline{y}_{3,1} \lor \overline{x}_{5,1} \lor y_{5,1})\\
\land\ &(\overline{x}_{2,1} \lor \overline{y}_{2,1} \lor \overline{x}_{3,1} \lor \overline{y}_{3,1} \lor \overline{x}_{5,2} \lor y_{5,2})\\
\land\ &(\overline{x}_{2,1} \lor \overline{y}_{2,1} \lor \overline{x}_{3,2} \lor \overline{y}_{3,2} \lor \overline{x}_{5,1} \lor y_{5,1})\\
\land\ &(\overline{x}_{2,1} \lor \overline{y}_{2,1} \lor \overline{x}_{3,2} \lor \overline{y}_{3,2} \lor \overline{x}_{5,2} \lor y_{5,2})\\
\land\ &(\overline{x}_{2,2} \lor \overline{y}_{2,2} \lor \overline{x}_{3,1} \lor \overline{y}_{3,1} \lor \overline{x}_{5,1} \lor y_{5,1})\\
\land\ &(\overline{x}_{2,2} \lor \overline{y}_{2,2} \lor \overline{x}_{3,1} \lor \overline{y}_{3,1} \lor \overline{x}_{5,2} \lor y_{5,2})\\
\land\ &(\overline{x}_{2,2} \lor \overline{y}_{2,2} \lor \overline{x}_{3,2} \lor \overline{y}_{3,2} \lor \overline{x}_{5,1} \lor y_{5,1})\\
\land\ &(\overline{x}_{2,2} \lor \overline{y}_{2,2} \lor \overline{x}_{3,2} \lor \overline{y}_{3,2} \lor \overline{x}_{5,2} \lor y_{5,2})\\
\land\ &(\overline{x}_{4,1} \lor \overline{y}_{4,1} \lor \overline{x}_{5,1} \lor \overline{y}_{5,1} \lor \overline{x}_{6,1} \lor y_{6,1})\\
\land\ &(\overline{x}_{4,1} \lor \overline{y}_{4,1} \lor \overline{x}_{5,1} \lor \overline{y}_{5,1} \lor \overline{x}_{6,2} \lor y_{6,2})\\
\land\ &(\overline{x}_{4,1} \lor \overline{y}_{4,1} \lor \overline{x}_{5,2} \lor \overline{y}_{5,2} \lor \overline{x}_{6,1} \lor y_{6,1})\\
\land\ &(\overline{x}_{4,1} \lor \overline{y}_{4,1} \lor \overline{x}_{5,2} \lor \overline{y}_{5,2} \lor \overline{x}_{6,2} \lor y_{6,2})\\
\land\ &(\overline{x}_{4,2} \lor \overline{y}_{4,2} \lor \overline{x}_{5,1} \lor \overline{y}_{5,1} \lor \overline{x}_{6,1} \lor y_{6,1})\\
\land\ &(\overline{x}_{4,2} \lor \overline{y}_{4,2} \lor \overline{x}_{5,1} \lor \overline{y}_{5,1} \lor \overline{x}_{6,2} \lor y_{6,2})\\
\land\ &(\overline{x}_{4,2} \lor \overline{y}_{4,2} \lor \overline{x}_{5,2} \lor \overline{y}_{5,2} \lor \overline{x}_{6,1} \lor y_{6,1})\\
\land\ &(\overline{x}_{4,2} \lor \overline{y}_{4,2} \lor \overline{x}_{5,2} \lor \overline{y}_{5,2} \lor \overline{x}_{6,2} \lor y_{6,2})\\
\land\ &(\overline{x}_{6,1} \lor \overline{y}_{6,1})\\
\land\ &(\overline{x}_{6,2} \lor \overline{y}_{6,2})
\end{aligned}$$

Figure 3: Lifted formula $\mathit{Lift}_2(\mathit{Peb}_{\Pi_2})$ of length 2 obtained from the pebbling contradiction over Π_2.

The communication problem that we are interested in is the *(falsified) clause search problem*. This is the problem of, given an unsatisfiable CNF formula F and a truth value assignment α, finding a clause $C \in F$ falsified by α. We denote this problem by $\mathit{Search}(F)$. In fact, from a communication complexity point of view we will be interested in lifts of this search problem $\mathit{Lift}(\mathit{Search}(F))$, while for our proof complexity trade-offs the perspective is slightly different in that we need to study the CNF formula $\mathit{Lift}(F)$ from Definition 2.1 and relate the hardness of this formula to the communication complexity of the falsified clause search problem $\mathit{Search}(\mathit{Lift}(F))$. Happily, this distinction does not really matter to us, since a good communication protocol for $\mathit{Search}(\mathit{Lift}(F))$ can also be used to solve $\mathit{Lift}(\mathit{Search}(F))$, and hence a lower bound for the latter communication problem applies also to the former, as stated formally in the next observation.

Observation 2.3. *Suppose that F is an unsatisfiable CNF formula. Then any two-player real communication protocol for $\mathit{Search}(\mathit{Lift}_\ell(F))$ where all selector variables $x_{i,j}$ in the same block are given to the same player can be adapted to a protocol for $\mathit{Lift}_\ell(\mathit{Search}(F))$ with the same parameters.*

We refer to, e.g., [HN12] for the easy proof (which is independent of the concrete communication model under consideration). Thanks to this observation, we can freely switch perspectives between $\mathit{Lift}_\ell(\mathit{Search}(F))$ and $\mathit{Search}(\mathit{Lift}_\ell(F))$ when we want to prove lower bounds for the latter problem. The reason that such lower bounds are interesting, in turn, is that if a CNF formula H has a CP refutation in short length and small space, then such a proof can be used to construct a round- and communication-efficient protocol for $\mathit{Search}(H)$.

Lemma 2.4. *If a CNF formula H can be refuted by cutting planes in length L and formula space s, then for any partition of the variables of H between Alice and Bob there is a real communication protocol solving $\mathit{Search}(H)$ in $\lceil \log L \rceil$ rounds with total communication cost at most $s \cdot \lceil \log L \rceil$.*

Sketching the proof very briefly, given a truth value assignment α Alice and Bob can do binary search over the refutation $(\mathbb{L}_0 = \emptyset, \mathbb{L}_1, \ldots, \mathbb{L}_L)$ of H until they find a $t \in [L]$ such that $\mathbb{L}_t$ evaluates to true under α but $\mathbb{L}_{t-1}$ evaluates to false. Then the derivation step at time t must be a download of an axiom $C \in H$ falsified by α. For the details we can reuse the proof from [HN12] verbatim, just adding the one simple but absolutely crucial observation that the protocol obtained in this way is also round-efficient, since all communication needed to evaluate a particular configuration $\mathbb{L}_t$ can be performed in parallel.

It is worth noting that although we state Lemma 2.4 for cutting planes here, there is nothing that really uses the syntactic properties of the cutting planes refutation. Thus, the proof works equally well for resolution, polynomial calculus, or any proof system for which configurations can be evaluated by round-efficient protocols where the communication scales as the space of the configuration.

2.5 Simulations of Protocols by Parallel Decision Trees

A *parallel decision tree* [Val75] for a search problem $S \subseteq \{0,1\}^m \times Q$ is a tree T such that each node v is labelled by a set of variables V_v and has exactly one outgoing edge for each of the $2^{|V_v|}$ possible assignments to these variables, and such that for every $\alpha \in \{0,1\}^m$ the path from the root of T defined by the edges consistent with α ends at a leaf labelled by some $q \in Q$ such that $(\alpha, q) \in S$ (where again the tacit assumption is that S is such that such a solution always exists). We say such a tree *solves* the search problem S. The *number of queries* of T is the maximal sum of set sizes $|V_v|$ along any path in T, and the *depth* of T is the length of a longest path.

Any decision tree T for a search problem S can be simulated by a communication protocol for the lifted problem $\mathit{Lift}(S)$ in a straightforward way, where if T wants to query the ith variable Alice and Bob can communicate to find y_{i,x_i} and then walk in T according to this value. Such a walk will end in a leaf labelled by a q such that $\big((y_{1,x_1}, y_{2,x_2}, \ldots, y_{m,x_m}), q\big) \in S$, i.e., a solution to the lifted search problem, and thus the query complexity of the original search problem provides an upper bound on the communication cost of the lifted problem. If in addition there is a parallel decision tree with small depth, then a protocol simulating such a tree will also be round-efficient. The main technical result of our paper is that simulating such a parallel decision tree is essentially the best any round-efficient protocol can do (provided that the lifting of the search problem is done with appropriate parameters).

Theorem 2.5 (Simulation theorem). *Let S be a relation with domain $\{0,1\}^m$ and let $\ell = m^{3+\epsilon}$ for some constant $\epsilon > 0$. If there is an r-round real communication protocol in cost c that solves $\mathit{Lift}_\ell(S)$, then there is a parallel decision tree in depth r solving S using $\mathrm{O}(c/\log \ell)$ queries.*

We remark that similar simulation theorems have previously been shown for both deterministic communication [RM99, GPW18] and real communication [BEGJ00], but unfortunately they fail to take round efficiency into account. Our proof of Theorem 2.5 follows the approach in these papers to build a decision tree for the original problem that simulates the communication protocol for the lifted problem. In order to obtain an efficient simulation we have to maintain (in an amortized sense) that the decision tree queries a variable only when a noticeable amount of communication has taken place. To prove that the decision tree constructed in this way is correct, we need to show that at the end of the simulation there exists a pair of inputs to Alice and Bob that are compatible both with the transcript and with a lift of the original input. Towards this end, during the simulation we maintain a set of such compatible inputs, which must not be allowed to shrink too fast.

In order for the proof to work we need to be able to handle two kinds of events: *communication events*, where we simulate the players communicating; and *query events*, where the decision tree under construction queries some variable and gets its actual value. Both of these events force us to prune the set of compatible communication inputs. In the first case we want to choose a communication message that removes as few inputs as possible, whereas in the second case we have to restrict the communication inputs to a subset that is compatible with the value returned by the decision tree query. We make sure to query a variable only when the transcript "reveals too much information" about Alice's and Bob's lifted input related to that variable, and thanks to this we can argue that query events do not happen too often and that the amount of communication provides an upper bound on the total number of queries.

Extending these techniques to round-efficient protocols and simulations by parallel decision trees causes significant additional complications, however. Very briefly, one issue is that we cannot let the tree query an individual variable as soon as sufficient information has been "revealed" about it during the simulation, but have to wait until we can issue a whole set of queries corresponding to a single message of the protocol. This makes it challenging to maintain a set of compatible inputs for variables we have not yet been allowed to query. Another issue is that, in contrast to deterministic communication protocols, real protocols do not partition the input domain into combinatorial rectangles. While this is not a big problem for a single comparison by the referee, it becomes more challenging when we want to handle a round consisting of many simultaneous comparisons.

2.6 From Decision Trees to Dymond–Tompa Trade-offs

The *Dymond–Tompa game* [DT85][2] is played in rounds on a DAG G by two players *Pebbler* and *Challenger*. In the first round, Pebbler places pebbles on a non-empty subset of vertices of G including the unique sink t and Challenger picks some vertex in this set. In all subsequent rounds, Pebbler places pebbles on some non-empty subset of vertices not yet containing pebbles, and Challenger either challenges a vertex in this new set (*jumps*) or re-challenges the previously chosen vertex (*stays*). This repeats until at the end of a round Challenger is standing on a vertex with all immediate predecessors pebbled (or on a source, for which the condition vacuously holds), at which point the game ends. Intuitively, Challenger is challenging Pebbler to "catch me if you can" and wants to play for as many rounds as possible, whereas Pebbler wants to "surround" Challenger as quickly as possible. We say that Pebbler *wins the r-round Dymond–Tompa game on G in cost c* if there is a strategy such that Pebbler can always finish the game in at most r rounds placing a total of at most c pebbles regardless of how Challenger plays.

In order to obtain lower bounds on the query complexity of parallel decision trees of bounded depth, we use an adversary argument and describe strategies that give as unhelpful answers as possible for variables queried by the decision trees. If we specialize this to the clause search problem for pebbling contradictions Peb_G, such adversary strategies are equivalent to Challenger strategies in the Dymond–Tompa game on G. For standard binary decision trees and the Dymond–Tompa game with unlimited number of rounds this was proven in [Cha13],[3] and we show that this equivalence extends also to our more general setting where decision trees can issue queries in parallel and we account for the number of rounds in the Dymond–Tompa game.

Lemma 2.6. *If there is a parallel decision tree for $Search(Peb_G)$ in depth r using at most c queries, then Pebbler has a winning strategy in the r-round Dymond–Tompa game on G in cost at most $c + 1$.*

It follows from this lemma that round-cost trade-offs for Dymond–Tompa pebbling implies depth-query trade-offs for parallel decision trees. To conclude the proof of the lower bound in our trade-off results, we need to find a family of graphs for which we can prove lower bounds for the cost of Pebbler strategies in the Dymond–Tompa game with bounded number of rounds. Towards this end, we establish that graphs that satisfy a certain connectivity property possess trade-offs between number of rounds and cost, and then exhibit such graphs. These graphs were inspired by graphs for which black-white pebbling trade-offs were shown in [LT82], but we need to make some modifications in order to obtain Dymond–Tompa trade-offs. Round-cost trade-offs have been previously studied in [KS90] but for a different range of parameters.

Lemma 2.7. *For any $n, r \in \mathbb{N}^+$ there exists an explicitly constructible DAG $G(n, r)$ with $\mathrm{O}(rn \log n)$ vertices such that the cost of the r-round Dymond–Tompa game on $G(n, r)$ is at least $\Omega(n)$.*

[2]We give a slightly different, but essentially equivalent, description of the Dymond–Tompa game that is closer to recent papers such as [Cha13, CLNV15].

[3]This game on decision trees is called the *Raz–McKenzie game* in [Cha13].

The graph $G(n, r)$ is structured in $r + 1$ layers and we obtain the lemma by designing a strategy for Challenger such that as long as Pebbler does not place too many pebbles, Challenger can make sure that in the ith round the challenged pebble is above the ith layer. Hence, the game does not end within r rounds.

2.7 Proofs of Main Theorems

Combining all the components discussed above we can now prove the following trade-off lower bound.

Theorem 2.8. *Let G be a DAG over m vertices such that any winning strategy for Pebbler in the r-round Dymond–Tompa game on G has cost $\Omega(c)$, and let $\epsilon > 0$ and $\ell = m^{3+\epsilon}$. Then $Lift_\ell\big(Peb_G\big)$ is a 6-CNF formula over $\Theta(m^{4+\epsilon})$ variables and $N = \Theta(m^{10+3\epsilon})$ clauses such that any cutting planes refutation of it in formula space less than $\frac{c}{r}\log N$, even with coefficients of unbounded size, requires length at least $2^{\Omega(r)}$.*

Proof. Suppose for the sake of contradiction that there is a cutting planes refutation of $Lift_\ell\big(Peb_G\big)$ in length $2^{o(r)}$ and formula space less than $\frac{c}{r}\log N$. By Lemma 2.4 this implies that there is a real communication protocol that solves $Lift_\ell\big(Search(Peb_G)\big)$ in $o(r)$ rounds and total cost $o(c\log N)$. Using Theorem 2.5 we obtain a parallel decision tree solving $Search(Peb_G)$ using $o(c)$ queries and depth $o(r)$. But if so, by Lemma 2.6 Pebbler wins the $o(r)$-round Dymond–Tompa game on G in cost $o(c)$, which contradicts the assumption of the theorem. □

In order to attain our trade-off results we also need upper bounds on refutations of these formulas. Small-size upper bounds follow by essentially the same approach of lifting black pebbling upper bounds as in [BN11, HN12], although more care is needed since our lifts are of non-constant length. For the small-space refutations, this technique does not work because the space loss due to the large lift length is larger than the upper bound we are aiming for. Luckily, we can instead prove upper bounds in the Dymond–Tompa game with unlimited rounds and then convert them into refutations in small space. Theorems 1.1 and 1.2 then follow from Theorem 2.8 applied to an appropriate family of graphs that exhibit Dymond–Tompa trade-offs as in Lemma 2.7.

The tools we have developed also allow us to prove the monotone circuit separation in Theorem 1.4. The function that witnesses the separation is inspired by the PYRAMID-GEN function of [RM99] adapted to the graphs in Lemma 2.7. Then we translate the Dymond–Tompa trade-off into a lower bound for deterministic communication protocols with few rounds, which we then transform into a lower bound for circuits of small depth via the Karchmer–Wigderson game [KW90].

3 From Proofs to Communication Protocols

As mentioned in the preliminaries, length space trade-offs for a given proof system can be obtained via reduction to the falsified clause search problem. Exactly which communication model to consider for the search problem depends on the proof system. Given a sequential proof system $\mathcal{P}$ and a communication model $\mathcal{M}$, let $c_{\mathcal{P},\mathcal{M}}$ and $r_{\mathcal{P},\mathcal{M}}$ be the maximum cost and the maximum number of rounds, respectively, required to evaluate a line/formula of any configuration.

The idea behind the reduction is to, given a refutation as a sequence of configurations, do a binary search in this sequence in order to find two consecutive configurations such that the first is evaluated to true and the second to false. Since the proof system is sound, this false configuration must be due to an axiom download and this axiom must be falsified. Finally, observing that each line/formula of a configuration can be evaluated in parallel, we get the following lemma.

Lemma 3.1. *Let π be a refutation in a sequential proof system $\mathcal{P}$ of a CNF formula F in length L and formula space s. Then, in any (deterministic) communication model $\mathcal{M}$ and for any partition of the variables of F between Alice and Bob there is a communication protocol that solves $Search(F)$ in $r_{\mathcal{P},\mathcal{M}} \cdot \lceil \log L \rceil$ rounds with total communication cost at most $s \cdot c_{\mathcal{P},\mathcal{M}} \cdot \lceil \log L \rceil$.*

Proof. Suppose Alice and Bob are each given a part of an assignment α to F. Fix a $\mathcal{P}$-refutation $\pi = \{\mathbb{D}_0, \mathbb{D}_1, \ldots, \mathbb{D}_L\}$ of F as in the statement of the lemma. By definition of refutation, it must be the case that $\mathbb{D}_0 = \emptyset$ and $\perp \in \mathbb{D}_L$.

Alice and Bob consider the configuration $\mathbb{D}_{L/2}$ in the refutation at time $L/2$ and with joint efforts (involving some communication, which we will discuss shortly) evaluate the truth value of $\mathbb{D}_{L/2}$ under the assignment α. If $\mathbb{D}_{L/2}$ is true under α, they continue their search in the subderivation $\{\mathbb{D}_{L/2}, \mathbb{D}_1, \ldots, \mathbb{D}_L\}$. If $\mathbb{D}_{L/2}$ is false, the search continues in the first half of the refutation $\{\mathbb{D}_0, \mathbb{D}_1, \ldots, \mathbb{D}_{L/2}\}$ Note that $\mathbb{D}_0 = \emptyset$ is vacuously true under any assignment, and since $\perp \in \mathbb{D}_L$ this last configuration evaluates to false under any assignment. The binary search is carried out so as to maintain that the first configuration in the current subderivation under consideration evaluates to true and the last one evaluates to false under the given assignment α. Hence, after at most $\lceil \log L \rceil$ steps Alice and Bob find a $t \in [L]$ such that $\mathbb{D}_{t-1}$ is true under α but $\mathbb{D}_t$ is false. Since the proof system is sound, the derivation step to get from $\mathbb{D}_{t-1}$ to $\mathbb{D}_t$ cannot have been an inference or erasure, but must be a download of some axiom clause $C \in H$. This clause C must be false under α, and so Alice and Bob give C as their answer.

Now all that remains is to discuss how much communication is needed to evaluate a configuration in the refutation. Every line/formula in the configuration can be evaluated with cost at most $c_{\mathcal{P},\mathcal{M}}$ and in at most $r_{\mathcal{P},\mathcal{M}}$ rounds. Moreover, the $r_{\mathcal{P},\mathcal{M}}$ rounds to evaluate each line can be done in parallel by simply concatenating, at each round i, all the ith messages of the protocol for evaluating each line of the configuration. Since each configuration has at most s lines, it can be evaluated with cost at most $s \cdot c_{\mathcal{P},\mathcal{M}}$ and in at most $r_{\mathcal{P},\mathcal{M}}$ rounds. □

We note that, for randomized communication models (which we do not use in this paper, but are used for example in [HN12, GP18]), the above theorem holds if $c_{P,M}$ and $r_{P,M}$ are defined to be the maximum cost and the maximum number of rounds, respectively, required to evaluate a line/formula of any configuration with high enough probability of success so that the union bound of the probability of error over all the evaluations of configurations is small enough.

Ideally, given a proof system P we want a communication model M such that $r_{P,M}$ and $c_{P,M}$ are constants, or at least a slow growing function. We only consider semantic versions of proof systems, where we specify the format of proof lines and use the fact that derivation rules, whichever they are, are semantically sound (as defined in [ABRW02]).

For example, if P is resolution, where lines are clauses of the form $\bigvee_j x_j^{b_j}$, then Alice and Bob can evaluate a line in two rounds and two bits of communication in the deterministic communication model.

If we consider polynomial calculus over any field $\mathbb{F}$, where lines are polynomials of the form $\sum_m \prod_j a_{m,j} x_j$ but the space measure is the number of monomials, Alice and Bob first check that the assignment is $\{0, 1\}$-valued—and immediately output a falsified axiom otherwise—and then run the binary search protocol, where they can evaluate a monomial in two rounds and two bits of communication in the deterministic communication model.

For cutting planes with bounded coefficients, Alice and Bob can evaluate a line in two rounds and either $\mathrm{O}(\log N)$ bits of communication in the deterministic communication model if the bound is a polynomial in the size of the formula or $\mathrm{O}(\log L)$ bits if the bound is a polynomial in the size of the refutation.

For unrestricted cutting planes, Alice and Bob can evaluate a line in one round and one comparison in the real communication model.

The small but key difference from previous papers [HN12, GP18] is that we explicitly consider rounds. The theorem states that if there exists a refutation in small space and small length, then there is a communication protocol that solves the falsified clause search problem not only with small communication cost, but also in few rounds.

It seems unlikely that the techniques used so far (without rounds) would yield true trade-offs; let us discuss why. For a trade-off of the form $s \cdot \log L \geq x$ to be a true trade-off there must exist a refutation in small space and another one in small length. The formulas for which trade-offs have been proven are lifted versions of pebbling formulas, which have refutations in small (linear) length, but not necessarily small space: the black-white pebbling number is a lower bound for resolution space, as proven in [BN11].

For the pyramid graphs studied in [HN12, GP18], the black-white pebbling number is asymptotically equal to the Dymond–Tompa price, which in turn is an upper bound for the communication complexity, as we argued in Section 2. Therefore, for the resolution proof system, the apparent trade-off is actually $s \cdot \log L = \Omega(s)$, giving only an uninformative length lower bound for the feasible range of space, and so the formula properties are better described by a space lower bound rather than a trade-off.

It seems plausible that the black-white pebbling number is also a lower bound for polynomial calculus space and cutting planes total space, and thus the "trade-offs" between PC-space or CP-total space and length, might also turn out to be unconditional space lower bounds.

Even if we consider other graph families, the best separation between black-white pebbling number and Dymond–Tompa price so far is logarithmic in the size of the graph [CLNV15], which still does not give meaningful results for resolution. It seems more promising to search for trade-offs in graphs where the black-white pebbling number is small but nonetheless have trade-offs in resolution, established by some means other than communication complexity.

To keep the discussion short and focused we only mention that trade-offs have been proven for stacks of superconcentrators [BN11] and Carlson–Savage graphs [Nor12]. Yet in both cases the Dymond–Tompa price is too small to give meaningful trade-offs: in the first case, it is enough to note that the Dymond–Tompa price is at most the depth, and a stack of superconcentrators has small depth; for Carlson–Savage graphs, the proof is similar, but the depth argument is not enough.

To sum up, we showed that the graphs for which the previous techniques yield trade-offs are likely to have unconditional space lower bounds (but we cannot prove it), and that for graph families that may have trade-offs—and indeed we prove that this is the case for a special family of superconcentrators—the previous techniques cannot prove them.

4 From Real Communication to Parallel Decision Trees

We have reached the heart of the reduction. This is by far the most intriguing and also the most difficult part of the paper. The reader that first wishes to have an overview of the whole proof should skip Sections 4.1 and 4.2.

To prove Theorem 2.5 we use the same high-level approach of [RM99, BEGJ00, GPW18]: we build a decision tree that simulates a protocol solving the composed search problem $\mathit{Lift}(S)$ and it only queries a variable when, in a certain sense, the transcript reveals too much information about the index for that variable.

More precisely, we keep two sets of inputs A and B that are compatible with the communication so far and that are large enough. Additionally for A we enforce that for each coordinate i that we have not queried yet, if we fix every other coordinate to some value, there are still many choices for what to set the index x_i to.

To maintain the invariant, we need to handle two kinds of events: communication events where we guess a message and restrict A and B to the new compatible inputs, with some additional cleanup, and query events, where some coordinate i becomes too dependent on other coordinates. Since for each coordinate there are more choices for y_i than Bob can expect to communicate, we will be able to find an input for the players such that $\mathsf{Ind}(x_i, y_i)$ agrees with z_i, and then restrict A and B appropriately.

At the end of the protocol, if we were to query the remaining variables, we would have a pair of inputs (x, y) that are compatible with the transcript, therefore the protocol answers correctly, and such that $\mathsf{Ind}(x_i, y_i) = z_i$ in every coordinate, therefore the answer is also correct for the decision tree.

To argue that we do not make too many queries we keep a density function that measures how many choices we have for Alice's input over not queried coordinates. This function increases on communication, decreases after a query, and is nonnegative, which gives us a bound on the number of queries in terms of communication.

The description up to this point is common to all flavours of the simulation theorem, with or without rounds and with deterministic or real communication. The differences will surface once we try to implement it.

The first challenge we encounter is when exactly should we query a variable. If we do not have any bound on depth, then we can do that as soon as we detect that the invariant is broken and we need to restore it. Since we want to measure the effect of rounds, however, this exact approach will not work for us, because we might need to query a variable mid-message. Indeed, if Alice sends the message x_1x_2, we would first simulate sending some bits of x_1, then query z_1 and restrict the inputs to those such that $\mathsf{Ind}(x_1, y_1) = z_1$, keep simulating sending bits of x_1 and x_2, then query z_2 and restrict the inputs, and finish sending bits. We had to use two rounds of queries in a single round of communication.

It seems natural to delay querying the variables until the end of the message, but now we have another problem. Assume that Bob had sent that the 0-th bit of y_1 is a 0, and that Alice's message is x_1. If we guess that the message is $x_1 = 0$ but after we query z_1 we get that $z_1 = 1$, then the inputs compatible with the communication stop being compatible with the Ind gadget.

Our solution is to indeed delay querying the variables until the end of the message but still restrict the inputs as soon as the invariant breaks, in a way that, after fixing x_1, $\mathsf{Ind}(x_1, y_1)$ can take any value. During the interval of communication between the restriction and the query we keep a set C that acts as a proxy for B over the coordinates that we have not queried (and do not intend to). This is a harder task, but still feasible because only Alice speaks during this time, so we do not need to know B precisely.

The second challenge is to adapt the simulation to a real communication protocol. As opposed to deterministic protocols, the set of compatible inputs does not necessarily form a rectangle. However, as observed by [Joh98], the result of one comparison splits the matrix of inputs in such a way that one quadrant—thus a rectangle—is monochromatic, and [BEGJ00] uses this fact to decide the outcome of each comparison.

Since we want to query variables only at the end of each round we might not know what B is at the time we want to extract a rectangle from a comparison matrix, and unfortunately the proxy does not help. Therefore we need to extract rectangles from the input when we do know B, that is before we start simulating the message. We could easily extract a rectangle of size a $2^{-k} \times 2^{-k}$-fraction of the inputs, where k is the size of the message, but that would be equivalent to simulating the message in one go, which we argued does not work in the deterministic case.

Our solution is to extract a rectangle of size a $1/4 \times 2^{-k \log k}$-fraction of the inputs. Even though this is equivalent to simulating a long Bob message at once, it has the advantage that the equivalent message for Alice is very short, so we can still use the very same techniques to recover the invariants as in the deterministic case.

4.1 Simulation of Deterministic Communication Protocols by Decision Trees

We begin by proving the simulation theorem for the more common deterministic communication model. Throughout this section we assume that the arity of the Ind function is $\ell = m^{3+\epsilon}$ for some small constant $\epsilon > 0$, where m is the size of the input.

Theorem 4.1. *If there is a deterministic communication protocol solving* $\mathit{Lift}(S)$ *using communication* c *and* r *rounds, then there is a parallel decision tree solving* S *using* $\mathrm{O}(c/\log \ell)$ *queries and depth* r*.*

We mention in Section 2 that we have to use a lift of polynomial length as opposed to constant length; this is needed for the simulation theorem to apply. In [GPW18] and [RM99] the lift is of size m^{20}, while in [BEGJ00] the lift is of size m^{14}, and a more careful analysis shows that $m^{4+\epsilon}$ is enough. Using a combinatorial approach to the analysis we can lower the lift length to $m^{3+\epsilon}$, but getting beyond this requires new techniques.

To be able to prove Theorem 4.1 we need to introduce some notation. Let $\gamma = 1/(3+\epsilon)$, δ, λ, μ be numbers strictly between 0 and 1 such that the inequalities

$$\lambda - \gamma > \mu \tag{4.1a}$$

$$\mu + \delta - 1 > \gamma \tag{4.1b}$$

$$\gamma + \delta < 1 \tag{4.1c}$$

hold. Note that a solution exists iff $\gamma < 1/3$. For concreteness, we can think of $\gamma = 1/3 - 2\xi$, $\lambda = 1 - \xi$, $\mu = 2/3$, and $\delta = 2/3$ for some $\xi > 0$.

Let Π be an r-round deterministic communication protocol solving $\mathit{Lift}(S)$ in cost c. Let X^v and Y^v be the sets of inputs to Alice and Bob, respectively, that are compatible with node v of the protocol tree. We are going to keep two sets $A \subseteq [\ell]^m$ and $B \subseteq \{0,1\}^{\ell m}$ of inputs that are compatible with the communication so far, that is $A \subseteq X^v$ and $B \subseteq Y^v$, but that have been "cleaned up".

We are often interested in the coordinates that the decision tree has not yet queried, which we denote $I \subseteq [m]$. We denote vectors and sets of vectors with coordinates in I with a subscript I to indicate their type. For two disjoint sets of coordinates I and J, we denote by $x_I{\cdot}x_J$ the vector with coordinates in $I \cup J$ obtained from the concatenation of x_I and x_J. The projection of a vector x_J to the coordinates in $I \subseteq J$, denoted by $\pi_I(x_J)$, is the vector x_I such that $x_J = x_I{\cdot}x_{J\setminus I}$ for some vector $x_{J\setminus I}$. We generalize this to sets of vectors and denote the projection of a set S_J to the coordinates in $I \subseteq J$ by $\pi_I(S_J) = \{\pi_I(x_J) : x_J \in S_J\}$.

In order to formalize the property of having little information about a coordinate, we define $\mathit{Graph}_i(A_I)$ as the bipartite graph where left vertices $x_{\{i\}}$ are elements of $[\ell]$, right vertices $x_{I\setminus\{i\}}$ are elements of $[\ell]^{|I|-1}$, and there is an edge between two vertices if $x_{I\setminus\{i\}}{\cdot}x_{\{i\}} \in A_I$. Analogously, let $\mathit{Graph}_i(B_I)$ be the bipartite graph where left vertices $y_{\{i\}}$ are elements of $\{0,1\}^\ell$, right vertices $y_{I\setminus\{i\}}$ are elements of $\{0,1\}^{\ell(|I|-1)}$, and there is an edge between two vertices if $y_{I\setminus\{i\}}{\cdot}y_{\{i\}} \in B_I$. Now let $\mathit{MinDeg}_i(A_I)$ and $\mathit{AvgDeg}_i(A_I)$ be the minimum and average right degree of $\mathit{Graph}_i(A_I)$, both taken over the set of vertices of positive degree. We consider that we do not know too much about a coordinate i if $\mathit{AvgDeg}_i(A_I) \geq \ell^\lambda$. Moreover, we say A_I is *thick* if $\mathit{MinDeg}_i(A_I) \geq \ell^\mu$ for all $i \in I$, a property we make sure to keep throughout the simulation. Observe that, since $|A_I|$ is the number of edges in $\mathit{Graph}_i(A_I)$ and $|\pi_{I\setminus\{i\}}(A_I)|$ is the number of right vertices with positive degree, the definition of average degree is equivalent to

$$\mathit{AvgDeg}_i(A_I) = \frac{|A_I|}{|\pi_{I\setminus\{i\}}(A_I)|} \ , \tag{4.2}$$

which is more convenient to work with.

A useful observation is that minimum degree (and therefore thickness) is monotone with respect to projections.

Observation 4.2 ([RM99]). *$\mathit{MinDeg}_j(\pi_{I\setminus\{i\}}(A)) \geq \mathit{MinDeg}_j(\pi_I(A))$ for all $j \in I \setminus \{i\}$.*

Proof. For each index $j \in I \setminus \{i\}$, if there is an edge between $x_{I\setminus\{j\}}$ and $x_{\{j\}}$ in $\mathit{Graph}_j(A_I)$, then there is also an edge between $x_{I\setminus\{i,j\}} = \pi_{I\setminus\{i,j\}}(x_{I\setminus\{j\}})$ and $x_{\{j\}}$ in $\mathit{Graph}_j(\pi_{I\setminus\{i\}}(A_I))$. Formally, we have that the set $N_j(x_{I\setminus\{i,j\}})$ of neighbours of a vertex $x_{I\setminus\{i,j\}}$ in $\mathit{Graph}_j(\pi_{I\setminus\{i\}}(A_I))$ is

$$\begin{aligned} N_j(x_{I\setminus\{i,j\}}) &= \{x_{\{j\}} \in [\ell] : x_{I\setminus\{i,j\}}{\cdot}x_{\{j\}} \in \pi_{I\setminus\{i\}}(A)\} && (4.3)\\ &= \{x_{\{j\}} \in [\ell] : \exists x_{\{i\}} \in [\ell] \text{ s.t. } x_{I\setminus\{i,j\}}{\cdot}x_{\{i\}}{\cdot}x_{\{j\}} \in \pi_I(A)\} && (4.4)\\ &= \bigcup_{x_{\{i\}}\in[\ell]} \{x_{\{j\}} \in [\ell] : x_{I\setminus\{i,j\}}{\cdot}x_{\{i\}}{\cdot}x_{\{j\}} \in \pi_I(A)\} && (4.5)\\ &= \bigcup_{x_{\{i\}}\in[\ell]} N_j(x_{I\setminus\{i,j\}}{\cdot}x_{\{i\}}) \ , && (4.6) \end{aligned}$$

where $N_j(x_{I\setminus\{i,j\}}{\cdot}x_{\{i\}})$ is the set of neighbours of the vertex $x_{I\setminus\{i,j\}}{\cdot}x_{\{i\}}$ in $\mathit{Graph}_j(A_I)$. Therefore, for any $x_{\{i\}}$ such that $x_{I\setminus\{i,j\}}{\cdot}x_{\{i\}}$ has positive degree we have that $|N_j(x_{I\setminus\{i,j\}})| \geq |N_j(x_{I\setminus\{i,j\}}{\cdot}x_{\{i\}})| \geq \mathit{MinDeg}_j(\pi_I(A))$. □

Finally, we define two density loss measures for inputs over non-queried coordinates. For $A_I \subseteq [\ell]^{|I|}$, let $\alpha(A_I) = -\log\frac{|A_I|}{\ell^{|I|}} = |I|\log\ell - \log|A_I|$ measure how many inputs on I coordinates Alice has lost out of the $\ell^{|I|}$ possibles. Analogously, for $B_I \subseteq \{0,1\}^{\ell|I|}$, let $\beta(B_I) = -\log\frac{|B_I|}{2^{\ell|I|}} = |I|\ell - \log|B_I|$. We make sure to keep these measures small, so that at any point there are many inputs compatible with the communication so far.

While the absolute number of choices decreases with projections, density actually increases (and so the loss of density decreases).

Observation 4.3 ([GPW18]). $\alpha(\pi_{I\setminus\{i\}}(A)) = \alpha(\pi_I(A)) - \log \ell + \log AvgDeg_i(\pi_I(A))$

Proof. By definition of α, this is equivalent to $|\pi_{I\setminus\{i\}}(A)| = |\pi_I(A)|/AvgDeg_i(\pi_I(A))$, which follows from the definition of average degree (4.2). □

The decision tree that witnesses Theorem 4.1 is Algorithm 4. In order to describe this algorithm, we have two auxiliary procedures: `prune` and `project`. For now, we state the properties that these procedures satisfy and defer their description and the proof that they indeed satisfy the properties claimed to after the proof of Theorem 4.1. Procedure `prune` is used to restore the thickness of A after Alice speaks and satisfies the following.

Lemma 4.4 (Thickness Lemma [RM99]). *If $AvgDeg_i(\pi_I(A)) \geq \ell^\lambda/4$ for all $i \in I$, then the return value A' of* `prune`(A, I) *satisfies*

1. *$\pi_I(A')$ is thick;*
2. *$\alpha(\pi_I(A')) \leq \alpha(\pi_I(A)) + 1$.*

The auxiliary procedure `project` is used to pick the appropriate U after we make a query to coordinate i in order to recover the density of A with respect to the remaining coordinates. To restrict the inputs of A on one coordinate i to a set $U \subseteq [\ell]$ we write $\rho_{i,U}(A) = \{x \in A : x_i \in U\}$, and similarly to restrict B on coordinate i to a set $V \subseteq \{0,1\}^\ell$, we have $\rho_{i,V}(B) = \{y \in B : y_i \in V\}$.

Note that in the description of Algorithm 4 we employ sets C_I that act as a proxy for $\pi_I(B)$. Moreover, we denote by $V^b(U) = \{w \in \{0,1\}^\ell : \forall j \in U\ w_j = b\}$ the set of b-monochromatic colourings of U and denote by ${C_{I\setminus\{i\}}}^{(b)}(U) = \pi_{I\setminus\{i\}}(\rho_{i,V^b(U)}(C_I))$ the projection to $I \setminus \{i\}$ of the inputs of C_I that in the ith coordinate are b-monochromatic colourings of U. Procedure `project` satisfies the following properties.

Lemma 4.5 (Projection Lemma). *If $\pi_I(A)$ is thick, and $\beta(C_I) \leq 2\ell^\gamma \log^2 \ell$, then the return value U of* `project`(A, C_I, I, i) *satisfies*

1. *$\pi_{I\setminus\{i\}}(\rho_{i,U}(A))$ is thick ;*
2. *$\alpha(\pi_{I\setminus\{i\}}(\rho_{i,U}(A))) \leq \alpha(\pi_I(A)) - \log \ell + \log AvgDeg_i(\pi_I(A))$;*
3. *$\beta({C_{I\setminus\{i\}}}^{(0)}(U) \cap {C_{I\setminus\{i\}}}^{(1)}(U)) \leq \beta(C_I) + 1$.*

The difference between Algorithm 4 and the algorithm in [RM99, GPW18] is that, when we need to restrict the sets A and B, instead of making a query we consider both possible outcomes for Bob. We can delay committing to either outcome until the moment before Bob starts to speak, at which point we make all queries at once. We assume that Q is a queue, that is, its elements are sorted in insertion order. We also assume that $\arg\max$ decides arbitrarily in case of tie, and observe that if $b = \arg\max|\pi_I(A \cap X^{v_b})|$, then $\alpha(\pi_I(A \cap X^{v_b})) \leq \alpha(\pi_I(A)) + 1$.

Lemma 4.6 (Main Lemma). *If Π is a protocol that solves $Lift(S)$ using communication $c < \frac{m}{2}(1 - \lambda)\log \ell$ and r rounds, then* `eval` *solves S using at most $2c/(1-\lambda)\log \ell$ queries and depth r.*

Observe that Theorem 4.1 follows from this lemma, since if $c \geq \frac{m}{2}(1-\lambda)\log \ell$ then a parallel decision tree that queries all variables in depth 1, satisfies the theorem. Before proving Lemma 4.6, let us consider some possible protocols and what Algorithm 4 does in each case.

Consider the trivial protocol where Alice in one round sends all of her input to Bob, who outputs the answer. To be fair, this is a bit too much communication for Lemma 4.6 to apply, but let us look over this fact and try to get an intuition of what happens. While Alice is speaking, the algorithm has to commit to

```
1  let $A = [\ell]^m$, $B = \{0,1\}^{\ell m}$, $I = [m]$, $v$ be the root of $\Pi$
2  while $v$ not a leaf do
3      let $Q = \emptyset$, $C_I = \pi_I(B)$
4      while $v$ is a node where Alice speaks do
5          while $\exists i \in I$ such that $AvgDeg_i(\pi_I(A)) < \ell^\lambda$ do
6              let $U_i$ = project $(A, C_I, I, i)$
7              let $A = \rho_{i,U_i}(A)$, $C_{I\setminus\{i\}} = {C_{I\setminus\{i\}}}^{(0)}(U_i) \cap {C_{I\setminus\{i\}}}^{(1)}(U_i)$,
                   $I = I \setminus \{i\}$, $Q = Q \cup \{i\}$
8          let $b = \arg\max|\pi_I(A \cap X^{v_b})|$
9          let $A = \texttt{prune}(A \cap X^{v_b}, I)$, $v = v_b$
10     query coordinates $Q$ to get string $z_Q$
11     for $i \in Q$ do
12         let $B = \rho_{i,V}(B)$, where $V = V^{z_i}(U_i)$
13     while $v$ is a node where Bob speaks do
14         let $b = \arg\max|\pi_I(B \cap Y^{v_b})|$
15         let $B = B \cap Y^{v_b}$, $v = v_b$
16 return the answer at $v$
```

Figure 4: Procedure eval(Π,z)

the values of her coordinates, and therefore, for all coordinates i, $AvgDeg_i(\pi_I(A))$ eventually becomes too small and i is marked to be queried. After she finishes speaking the algorithm queries all coordinates and restricts Bob's input to be compatible with the queries and with Alice's message, i.e., it only keeps inputs that have the queried value on the corresponding position. Finally, the algorithm answers what Bob would output for any of the compatible inputs. Note that the decision tree in this case is the depth-1 decision tree that queries all coordinates at once and answers accordingly.

Another possible protocol is the one that follows a parallel decision tree T, as explained in Section 2: Alice and Bob communicate to find the value of the coordinates queried on each node and then continue on T according to these values. For this protocol, Algorithm 4 marks to be queried all the coordinates Alice and Bob talk about because the corresponding average degree gets too low. Note that the final decision tree is exactly the same as the one Alice and Bob were following.

Now we consider what happens in a more extreme case. Suppose the protocol is very unbalanced in the sense that Alice's first bit is a 0 if her first coordinate is 42, and otherwise is a 1. In this case, when at line 8 the algorithm chooses a bit for Alice to speak, it will always choose 1 since it is compatible with the most inputs.

Proof of Lemma 4.6. Let R^v be the rectangle of inputs compatible with node v, and let c_v^A (resp. c_v^B) be the number of bits sent by Alice (resp. Bob) up to node v. We show that the following invariants hold throughout the algorithm:

1. $\pi_I(A)$ is thick;
2. $A \times B \subseteq R^v$;
3. $m - |I| \leq 2c_v^A/(1-\lambda)\log\ell$;
4. $\beta(C_I) \leq m - |I| + c_v^B$;

and that the following invariants hold at the beginning of each round:

5. $\beta(\pi_I(B)) \leq m - |I| + c_v^B$;

6. $\mathsf{Ind}(x_i, y_i) = z_i$ for all $(x, y) \in A \times B$ and $i \notin I$.

All six invariants are true at the beginning of the algorithm. We assume invariants 1 through 4 are true up to the current point of the algorithm, and invariants 5 and 6 are true at the beginning of the current round. We then show that, after executing the next line, 1 through 4 still hold and, at the beginning of the next round, 5 and 6 still hold.

To see that invariant 1 holds, we note that the set A is modified only at lines 7 and 9. For line 7 it is enough to argue that the assumptions of Lemma 4.5 hold, since Lemma 4.5 guarantees that $\pi_I(A)$ is thick. This is indeed the case, since we assume invariant 1 holds before this point, and since invariants 3 and 4 together with the assumption of Lemma 4.6 that $c_v^A + c_v^B \leq m \log \ell = \ell^\gamma \log \ell$ imply that

$$\beta(C_I) \leq m - |I| + c_v^B \leq 2c_v^A/(1-\lambda)\log \ell + c_v^B \leq c_v^A + c_v^B \leq \ell^\gamma \log \ell \ . \tag{4.7}$$

Similarly, for line 9 it is enough to argue that the assumption of Lemma 4.4 applies. Indeed it does, since we have that, before line 9,

$$AvgDeg_i(\pi_I(A \cap X^{v_b})) = \frac{|\pi_I(A \cap X^{v_b})|}{|\pi_{I\setminus\{i\}}(A \cap X^{v_b})|} \geq \frac{\frac{1}{2}|\pi_I(A)|}{|\pi_{I\setminus\{i\}}(A)|} = \frac{1}{2} AvgDeg_i(\pi_I(A)) \geq \frac{\ell^\lambda}{2} \quad , \tag{4.8}$$

where for the last inequality we are using the assumption that invariant 1 holds before this line. We note that the conditions for Lemmas 4.4 and 4.5 to apply are more relaxed than what we prove. This is so because we will apply the same lemmas when dealing with real communication and there we will need this extra slack.

For invariant 2, first note that A and B never increase. Moreover, the rectangle of compatible inputs R^v changes when v is modified at lines 9 and 15, and in both cases we restrict A (resp. B) to a subset of X^v (resp. Y^v).

To prove invariant 3, we show that

$$\alpha(\pi_I(A)) \leq 2c_v^A - (m - |I|)(1-\lambda)\log \ell \tag{4.9}$$

holds at all times. The invariant then follows since $\alpha(A_I) \geq 0$ by definition. First note that at the beginning of the algorithm $\alpha(\pi_I(A)) = 0$. Intuitively, we want to show that $\alpha(\pi_I(A))$ increases by at most 2 with every bit sent by Alice and decreases by at least $(1-\lambda)\log \ell$ with every query. More formally, to see why (4.9) is true, we argue that every time I decreases (by 1) at line 7, $\alpha(\pi_I(A))$ decreases by at least $(1-\lambda)\log \ell$, and every time c_v^A increases (by 1) at line 9, $\alpha(\pi_I(A))$ increases by at most 2. This is indeed sufficient since A, I and c_v^A are only modified at these two lines. At line 7, Lemma 4.5 guarantees that $\alpha(\pi_I(A))$ decreases by at least $(1-\lambda)\log \ell$ since the average degree (before updating A and I) was $AvgDeg_i(\pi_I(A)) < \ell^\lambda$. At line 8, b is chosen so that $\alpha(\pi_I(A \cap X^{v_b})) \leq \alpha(\pi_I(A)) + 1$ and thus, by Lemma 4.4, $\alpha(\pi_I(A))$ increases by at most 2 at line 9. Therefore, we conclude that (4.9) holds.

As for invariant 4, first note that C_I is only updated at lines 3 and 7. At line 3, invariant 4 holds because invariant 5 is true at the beginning of a round. At line 7, Lemma 4.5 guarantees that $\beta(C_{I\setminus\{i\}}) = \beta({C_{I\setminus\{i\}}}^{(0)}(U) \cap {C_{I\setminus\{i\}}}^{(1)}(U)) \leq \beta(C_I) + 1$. Note that it is possible to apply Lemma 4.5 as argued before. Therefore, invariant 4 follows.

To prove that invariant 5 holds at the end of a round we distinguish between Alice speaking and Bob speaking. If Bob speaks, B is updated at line 15 and the invariant clearly holds since each bit b that Bob says is chosen such that $\beta(\pi_I(B \cap Y^{v_b}))$ increases by at most 1.

If Alice speaks, B is updated at line 12. Let $Q = \{i_1, \ldots, i_{|Q|}\}$. Let I_0 be the non-queried coordinates at the beginning of the round and $I_\eta = I_{\eta-1} \setminus \{i_\eta\}$, for $\eta \in [|Q|]$. Moreover, let B_0 be the value of B at the beginning of the round and $B_\eta = \rho_{i,V}(B_{\eta-1})$, for $\eta \in [|Q|]$ and $i = i_\eta$. We prove by induction over η that $\pi_{I_\eta}(B_\eta) \supseteq C_{I_\eta}$. Recall that C_{I_η} is a set whose elements are vectors with coordinates in I_η and is not the projection of a set C to I_η. Therefore, the subscript serves not only as a reminder of the form of its elements, but also as an identifier of the set. At the beginning of the round $\pi_{I_0}(B_0) = C_{I_0}$. At each iteration,

$$\pi_{I_\eta}(B_\eta) = \pi_{I_\eta}(\rho_{i,V}(B_{\eta-1})) \supseteq \pi_{I_\eta}(\rho_{i,V}(\pi_{I_{\eta-1}}(B_{\eta-1}))) \tag{4.10}$$

$$\supseteq \pi_{I_\eta}(\rho_{i,V}(C_{I_{\eta-1}})) = C_{I_{\eta-1}}{}^{(z_i)}(U_i) \supseteq C_{I_\eta} \ , \tag{4.11}$$

so at the end of the round, that is, when $I = I_{|Q|}$ and $B = B_{|Q|}$, it holds that $\beta(\pi_I(B)) \leq \beta(C_I) \leq m - |I| + c_v^B$, where this last inequality follows from invariant 4.

For invariant 6, recall that A and B never increase. Furthermore, each time I is modified at line 7, we add to Q the coordinate i for which invariant 6 no longer holds (since the element i was removed from I and it has not been queried yet). Then we restore the invariant before the next iteration by restricting B at line 12. Indeed, if $(x, y) \in A \times B$, then $x_i \in U_i$ and $y_i \in V^{z_i}(U_i)$, so by definition of V it holds that $\mathsf{Ind}(x_i, y_i) = z_i$.

We have finished proving the invariants. From now on we assume that the algorithm ended and reached a leaf v. It is clear that the decision tree has depth r and the total number of queries is at most $2c/(1-\lambda)\log\ell < m$ by invariant 3. It remains to prove correctness, that is, that for any $z \in \{0,1\}^m$, the decision tree answers $\texttt{eval}(\Pi, z) \in S(z)$.

In order to prove this, we show that there exists an input $(x, y) \in R_v$ such that $\mathsf{Ind}(x, y) = z$, which implies $\texttt{eval}(\Pi, z) = \Pi(x, y) \in (\mathit{Lift}(S))(x, y) = S(\mathsf{Ind}(x, y)) = S(z)$. This is done by restricting A and B to A' and B' so that any input in A' and any input in B' are compatible with z and finally arguing that A' and B' are non-empty. For every queried coordinate i, we have already restricted A and B so that for any $x \in A$ and any $y \in B$, $\mathsf{Ind}(x_i, y_i) = z_i$; thus we are left to deal with the non-queried coordinates.

Note that $\mathit{Graph}_i(\pi_{\{i\}}(A))$ has only one vertex on the right with label $\emptyset$, and an edge from this vertex to an element $x_{\{i\}} \in [\ell]$ if $x_{\{i\}} \in \pi_{\{i\}}(A)$. Although $\mathit{Graph}_i(\pi_\emptyset(A))$ is not defined (which is the reason why below we apply Claims 4.7 and 4.8 directly and not Lemma 4.5), the projection $\pi_\emptyset(A)$ is well-defined and it is either the empty set, in which case A is empty, or it is the singleton set containing the empty string, in which case there exists at least one element $x \in A$. The same holds for B, $\pi_\emptyset(B)$ is either empty or it is the singleton set containing the empty string. Therefore, by the definition of α and β, if $\alpha(\pi_\emptyset(A))$ (resp. $\beta(\pi_\emptyset(B))$) is finite, then A (resp. B) is non-empty.

We start with $A' = A$, $B' = B$ and $I' = I$, and therefore we have that $\pi_{I'}(A')$ is thick, $\beta(\pi_{I'}(B')) \leq m - |I| + c_v^B \leq \ell^\gamma \log\ell$ and I' is non-empty since we queried less than m coordinates. While I' is not empty, we choose $i \in I'$ and apply Claims 4.7 and 4.8 to get a set U such that $\pi_{I'\setminus\{i\}}(\rho_{i,U}(A')) = \pi_{I'\setminus\{i\}}(A')$ and $\beta(\pi_{I'\setminus\{i\}}(B')^{(0)}(U) \cap \pi_{I'\setminus\{i\}}(B')^{(1)}(U)) \leq \beta(\pi_I(B')) + 1$. We then set $A' = \rho_{i,U}(A')$ and $B' = \rho_{i,V}(B')$, where $V = V^{z_i}(U)$, and thus, by definition of V, for all $x \in A'$ and $y \in B'$ it holds that $\mathsf{Ind}(y_i, x_i) = z_i$. Finally we set $I' = I' \setminus \{i\}$ and repeat. Note that these claims can indeed be applied while I' is not empty, since the fact that thickness is monotone (Observation 4.2) implies that $\pi_{I'}(A')$ is thick, and since $\beta(\pi_{I'}(B')) \leq c_v^B + (m - |I|) + (|I| - |I'|) \leq c + m < 2\ell^\gamma \log\ell$ because $\beta(\pi_{I'}(B'))$ increases by at most 1 for each $i \in I \setminus I'$.

At the end of this process, the set B' is such that $\beta(\pi_{I'}(B')) \leq 2\ell^\gamma \log\ell < \infty$, and thus B' is non-empty. Moreover, we have that $\pi_\emptyset(A') = \pi_\emptyset(A)$, and since A is non-empty (because $\alpha(\pi_I(A)) \leq 2c_v^A$), $\pi_\emptyset(A')$ is the singleton set containing the empty string and, therefore, A' is non-empty. □

Now we explain the auxiliary procedures and argue that they satisfy Lemmas 4.4 and 4.5. Procedure $\texttt{prune}$ just removes vertices with too small degree. Thus, we only need to argue that it does not remove too many vertices, that is, that $\alpha(\pi_I(A')) \leq \alpha(\pi_I(A)) + 1$.

Proof of Lemma 4.4. Let $A'_I = \pi_I(A')$. A'_I is thick by construction. By definition of α, showing that $\alpha(A'_I) \leq \alpha(\pi_I(A)) + 1$ is equivalent to showing that $|A'_I| \geq |\pi_I(A)|/2$. For each coordinate $i \in I$ we observe that, by the definition of average degree, there are at most $|\pi_I(A)|/\mathit{AvgDeg}_i(\pi_I(A))$ right vertices of positive degree in $\mathit{Graph}_i(\pi_I(A))$, and since $\mathit{Graph}_i(A'_I) \subseteq \mathit{Graph}_i(\pi_I(A))$, there are at most $|\pi_I(A)|/\mathit{AvgDeg}_i(\pi_I(A))$ right vertices of positive degree in $\mathit{Graph}_i(A'_I)$. This is an upper bound on the number of iterations of procedure $\texttt{prune}$ for coordinate i, since every iteration removes a right vertex of positive degree. Since $\mathit{AvgDeg}_i(\pi_I(A)) \geq \ell^\lambda/4$ for all $i \in I$, the total number of iterations

```
1 let $A_I = \pi_I(A)$
2 while $\exists i$ such that $MinDeg_i(A_I) < \ell^\mu$ do
3     let $x_{I\setminus\{i\}} \in V(Graph_i(A_I))$ be a right vertex of degree less than $\ell^\mu$
4     let $A_I = \{x_I \in A_I : \pi_{I\setminus\{i\}}(x_I) \neq x_{I\setminus\{i\}}\}$
5 return $\{x \in A : \pi_I(x) \in A_I\}$
```

Figure 5: Procedure `prune`(A, I)

```
1 let $U \subseteq [\ell]$ be a set such that
2     $\pi_{I\setminus\{i\}}(\rho_{i,U}(A)) = \pi_{I\setminus\{i\}}(A)$
3     $\beta(C_{I\setminus\{i\}}{}^{(0)}(U) \cap C_{I\setminus\{i\}}{}^{(1)}(U)) \geq \beta(C_I) + 1$
4 return $U$
```

Figure 6: Procedure `project`(A, C_I, I, i)

for each coordinate is at most $|\pi_I(A)|/AvgDeg_i(\pi_I(A)) \leq 4|\pi_I(A)|/\ell^\lambda$, which makes a total of at most $4|I||\pi_I(A)|/\ell^\lambda \leq 4\ell^{\gamma-\lambda}|\pi_I(A)|$ iterations overall. Each iteration removes $\deg(X'_{I\setminus\{i\}}) < \ell^\mu$ elements from A'_I, for a total of at most $4\ell^{\gamma+\mu-\lambda}|\pi_I(A)| \leq |\pi_I(A)|/2$ elements removed. The last inequality holds because of assumption (4.1a). Thus, at least $|\pi_I(A)|/2$ elements remain. □

Our Lemma 4.5 is slightly stronger than the projection lemmas in [RM99, GPW18] because it needs to handle both outcomes of the query to z_i, so we care not only about each of $\rho_{i,V^b(U)}(B)$ separately but about $B^{(0)}(U) \cap B^{(1)}(U)$. Nonetheless, essentially the same proof of [RM99] using the probabilistic method works—but not that of [GPW18], where the probabilities are too small for us. Procedure `project`, therefore, just asserts the existence of a return value U with the required properties.

The claims below show that with high probability a set U chosen at random satisfies Lemma 4.5. The probabilities in the claims are with respect to U picked uniformly among all subsets of $[\ell]$ of size ℓ^δ.

Claim 4.7 ([RM99]). If A is thick, then $\pi_{I\setminus\{i\}}(\rho_{i,U}(A)) = \pi_{I\setminus\{i\}}(A)$ with probability $1 - o(1)$.

Claim 4.8. If $\beta(C_I) \leq 2\ell^\gamma \log \ell$, then $\beta(C_{I\setminus\{i\}}{}^{(0)}(U) \cap C_{I\setminus\{i\}}{}^{(1)}(U)) \leq \beta(C_I) + 1$ with probability $1 - o(1)$.

Given these claims we can easily prove Lemma 4.5.

Proof of Lemma 4.5. By union bound there exists a set that satisfies both Claim 4.7 and Claim 4.8. Let U be such a set. Item 3 holds since U satisfies Claim 4.8 and, since U satisfies Claim 4.7, item 1 follows from Observation 4.2 and item 2 from Observation 4.3. □

We now proceed to prove Claim 4.7 and Claim 4.8.

Proof of Claim 4.7. We observe that the equality holds if every right vertex of $Graph_i(\pi_I(A))$ with positive degree has an edge into U. Since $MinDeg_i(\pi_I(A)) \geq \ell^\mu$, the probability that U is contained within the non-neighbours of any right vertex $x_{I\setminus\{i\}}$ is

$$\frac{\binom{\ell-|N_i(x_{I\setminus\{i\}})|}{\ell^\delta}}{\binom{\ell}{\ell^\delta}} \leq \frac{\binom{\ell-\ell^\mu}{\ell^\delta}}{\binom{\ell}{\ell^\delta}} \leq (1-\ell^{\mu-1})^{\ell^\delta} \leq e^{-\ell^{\mu+\delta-1}} \quad . \tag{4.12}$$

By a union bound over all right vertices, the probability of the equality not holding is at most

$$\ell^{|I|-1}e^{-\ell^{\mu+\delta-1}} < e^{|I|\log\ell-\ell^{\mu+\delta-1}} \leq e^{\ell^\gamma\log\ell-\ell^{\mu+\delta-1}} = o(1) \quad . \tag{4.13}$$

The last equality holds because of assumption (4.1b). □

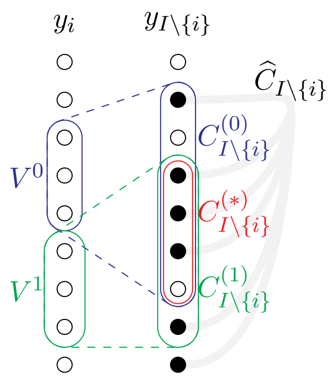

y_i
$y_{I\setminus\{i\}}$
$\widehat{C}_{I\setminus\{i\}}$
V^0
$C^{(0)}_{I\setminus\{i\}}$
$C^{(*)}_{I\setminus\{i\}}$
V^1
$C^{(1)}_{I\setminus\{i\}}$

Figure 7: Sets used in the proof of Claim 4.8

The proof of Claim 4.8 follows [RM99] except that our version of Claim 4.9 stated below is stronger and we apply it twice.

Claim 4.9. Let $W \subseteq \{0,1\}^\ell$ be any set of size at least $2^{-\phi\ell^\gamma} \cdot 2^\ell$, where ϕ is any function of ℓ such that $\log\phi = \mathrm{o}(\log\ell)$. Let U be a uniformly random subset of $[\ell]$ of size ℓ^δ. Then, for any $b \in \{0,1\}$, $\{b\}^{|U|} \in \pi_U(W)$ with probability at least $1 - o(1)$.

For the proof of Claim 4.8 we also use the following easy observation.

Observation 4.10. *Let T be a set and S a set-valued random variable. If $\Pr[s \in S] \geq p$ for every $s \in T$, then $\Pr[|S| \geq q|T|] \geq (p-q)/(1-q)$.*

Proof. Let $x = \Pr[|S| \geq q|T|]$. Then

$$
\begin{aligned}
p &\leq \Pr[s \in S] && (4.14)\\
&= \Pr[s \in S : |S| \geq q|T|] \cdot \Pr[|S| \geq q|T|] + \Pr[s \in S : |S| < q|T|] \cdot \Pr[|S| < q|T|] && (4.15)\\
&\leq 1 \cdot x + q \cdot (1-x) \ , && (4.16)
\end{aligned}
$$

from which the observation follows. □

Proof of Claim 4.8. We begin by observing that ${C_{I\setminus\{i\}}}^{(b)}(U)$, the set of right vertices that can be completed to b over U, is equal to $N_j(V^b(U))$ (see Figure 7). We want to prove that the set $C^{(*)}_{I\setminus\{i\}}(U) = {C_{I\setminus\{i\}}}^{(0)}(U) \cap {C_{I\setminus\{i\}}}^{(1)}(U)$ of right vertices that can be completed to any colour over U is large, namely that $|C^{(*)}_{I\setminus\{i\}}|/2^{\ell(|I|-1)} \geq \psi/2$ where $\psi = |C_I|/2^{\ell|I|} = 2^{-\beta(C_I)}$.

Right vertices of degree larger than $2^\ell\psi/4$ can be completed to any colour with high probability. Indeed, $|N_i(y_{I\setminus\{i\}})| \geq 2^\ell \cdot \psi/4 = 2^\ell \cdot 2^{-\beta(C)}/4 \geq 2^\ell \cdot 2^{-2\ell^\gamma \log^2\ell}/4 \geq 2^\ell \cdot 2^{-3\ell^\gamma \log^2\ell}$, so for each $b \in \{0,1\}$ we can apply Claim 4.9 with $\phi = 3\log^2\ell$ to show that $N_i(y_{I\setminus\{i\}}) \cap V^b(U) \neq \emptyset$ with probability $1 - \mathrm{o}(1)$. Taking a union bound, we can assume that $N_i(y_{I\setminus\{i\}}) \cap V^b(U) \neq \emptyset$ holds for both $b \in \{0,1\}$ except with probability $\mathrm{o}(1)$. In other words, $y_{I\setminus\{i\}} \in C^{(b)}_{I\setminus\{i\}}$ for $b \in \{0,1\}$, so $y_{I\setminus\{i\}} \in C^{(*)}_{I\setminus\{i\}}$.

Let $\widehat{C}_{I\setminus\{i\}} = \{y_{I\setminus\{i\}} : \deg(y_{I\setminus\{i\}}) \geq 2^\ell\psi/4\}$. We have shown that for every $y_{I\setminus\{i\}} \in \widehat{C}_{I\setminus\{i\}}$ it holds that $y_{I\setminus\{i\}} \in C^{(*)}_{I\setminus\{i\}}$ with probability $1 - \mathrm{o}(1)$. By Observation 4.10, with probability $1 - \mathrm{o}(1)$,

$$|C^{(*)}_{I\setminus\{i\}}| \geq 2/3 \cdot |\widehat{C}_{I\setminus\{i\}}| \ . \tag{4.17}$$

In fact, $2/3$ can be chosen to be any arbitrary number strictly smaller than 1, and the statement would still hold.

Therefore it is enough to prove that the set $\widehat{C}_{I\setminus\{i\}}$ of right vertices with large degree is large. Indeed, we have

$$2^{\ell|I|}\psi = |C_I| \le |\widehat{C}_{I\setminus\{i\}}| \cdot 2^\ell + |\{0,1\}^{\ell(|I|-1)} \setminus \widehat{C}_{I\setminus\{i\}}| \cdot 2^\ell \cdot \psi/4 \tag{4.18}$$

$$\le |\widehat{C}_{I\setminus\{i\}}| \cdot 2^\ell + 2^{\ell(|I|-1)} \cdot 2^\ell \cdot \psi/4 = |\widehat{C}_{I\setminus\{i\}}| \cdot 2^\ell + 2^{\ell|I|} \cdot \psi/4 \ , \tag{4.19}$$

from where

$$|\widehat{C}_{I\setminus\{i\}}| \ge 3/4 \cdot \psi \cdot 2^{\ell(|I|-1)} \ . \tag{4.20}$$

and the claim follows by combining equations (4.17) and (4.20). We observe that the $1/4$ in the definition of right vertices with large degree can be any arbitrarily small constant. □

For the proof of Claim 4.9, we think of $W \subseteq \{0,1\}^\ell$ as a set of binary colourings of $[\ell]$ and denote by $\pi_U(W)$ the projection of W to a subset $U \subseteq [\ell]$, i.e. $\pi_U(W) = \{w_U \in \{0,1\}^U : w \in W \text{ for some } w_{[\ell]\setminus U} \in \{0,1\}^{[\ell]\setminus I}\}$. Note that this is the same operation as $\pi_I(A)$, except they apply to different domains.

In the original paper [RM99] the constants are set to $\gamma = 2/20$, $\delta = 5/20$, and the same probabilistic argument works for any choice of constants such that $\gamma + 3\delta/2 < 1$. We present a combinatorial proof that works for any constants such that $\gamma + \delta < 1$, and in particular holds for $\gamma = 1/3 - \xi$ and $\delta = 2/3$, for any $\xi > 0$. We use the following corollary of the Kruskal–Katona Theorem, which we prove later on.

Claim 4.11. Let $\mathcal{U}$ be any family of subsets of $[\ell]$, where every $U \in \mathcal{U}$ is of size u. If $|\mathcal{U}| \ge \sum_{i=0}^{t} \binom{\ell-1-i}{\ell-u-i}$ and if $W \subseteq \{0,1\}^\ell$ is such that $\{b\}^{|U|} \notin \pi_U(W)$ for every $U \in \mathcal{U}$, then it holds that $|W| \le 2^\ell - \sum_{j=0}^{\ell-u}\sum_{i=0}^{t}\binom{\ell-1-i}{\ell-u-i-j}$.

Proof of Claim 4.9. We prove the following equivalent statement: if $\Pr_U[\{b\}^{|U|} \notin \pi_U(W)] \ge q$ (for, say, $q = \phi/\log\ell$), then $|W| \le 2^{\ell-\phi\ell^\gamma}$. We only use the fact that $\gamma + \delta < 1 + \frac{\log\frac{q}{\phi}}{\log\ell}$ (for $q = \phi/\log\ell$ this says that $\gamma + \delta < 1 - \frac{\log\log\ell}{\log\ell}$).

Let $\mathcal{U}$ be the set of all $U \subset [\ell]$ of size ℓ^δ such that $\{b\}^{|U|} \notin \pi_U(W)$. We have that

$$\frac{|\mathcal{U}|}{\binom{\ell}{\ell^\delta}} = \Pr_U[\{b\}^{|U|} \notin \pi_U(W)] \ge q \ . \tag{4.21}$$

Hence we get

$$|\mathcal{U}| \ge q\binom{\ell}{\ell^\delta} = q\frac{\ell}{\ell^\delta}\binom{\ell-1}{\ell^\delta-1} \ge \sum_{i=0}^{q\frac{\ell}{\ell^\delta}}\binom{\ell-1-i}{\ell^\delta-1} = \sum_{i=0}^{q\frac{\ell}{\ell^\delta}}\binom{\ell-1-i}{\ell-\ell^\delta-i} \ . \tag{4.22}$$

We can therefore apply Claim 4.11 with $u = \ell^\delta$ and $t = q\frac{\ell}{\ell^\delta}$ to get

$$|W| \le 2^\ell - \sum_{j=0}^{\ell-\ell^\delta}\sum_{i=0}^{q\frac{\ell}{\ell^\delta}}\binom{\ell-1-i}{\ell-\ell^\delta-i-j} \le 2^{\ell-q\frac{\ell}{\ell^\delta}+1} + 2^{\ell^\delta\log\ell} \le 2^{\ell-\phi\ell^\gamma} \ , \tag{4.23}$$

where the second inequality is a straightforward calculation that we prove in Claim 4.15; and the last inequality follows from $\gamma + \delta < 1 + \frac{\log\frac{q}{\phi}}{\log\ell}$. □

To prove Claim 4.11 we need to introduce some notation from extremal combinatorics. We use the following terminology from [Juk11]. If w is a binary colouring of $[\ell]$ (i.e. a binary vector of length ℓ), we say a *neighbour* of a w is a colouring which can be obtained from w by flipping one of its 1-entries to 0. A *shadow* of a set $A \subseteq \{0,1\}^\ell$ of binary colourings is the set $\partial(A)$ of all its neighbours. A set A is k-*regular* if every colouring in A colours exactly k elements 1. Note that in this case $\partial(A)$ is $(k-1)$-regular. The best possible lower bounds for the size of $\partial(A)$ were obtained independently by Kruskal [Kru63] and Katona [Kat68].

Theorem 4.12 (Kruskal–Katona Theorem). *If $A \subseteq \{0,1\}^\ell$ is k-regular, and if*

$$|A| = \binom{a_k}{k} + \binom{a_{k-1}}{k-1} + \ldots + \binom{a_s}{s}$$

then

$$|\partial(A)| \geq \binom{a_k}{k-1} + \binom{a_{k-1}}{k-2} + \ldots + \binom{a_s}{s-1} \ .$$

Let $\mathcal{P}(S)$ denote the power set of S. In [Fra84] it is proven that there exists an explicit compression function $C : \mathcal{P}(\{0,1\}^\ell) \to \mathcal{P}(\{0,1\}^\ell)$ such that, given a k-regular set $A \subseteq \{0,1\}^\ell$, $C(A)$ is k-regular, $|C(A)| = |A|$ and $|\partial(C(A))|$ matches the lower bound in Theorem 4.12, and, furthermore, the following proposition holds.

Proposition 4.13. $\partial(C(A)) \subseteq C(\partial(A))$.

Although this follows directly from the proposition in Section 2 of [Fra84], the formulation above is from [And87].

We define the *iterated shadow* of a k-regular set $A \subseteq \{0,1\}^\ell$ to be $\partial^{\leq k}(A) = \cup_{j=0}^{k} A_j$, where $A_0 = A$ and $A_j = \partial(A_{j-1})$ for $0 < j \leq k$. The following corollary follows immediately from Theorem 4.12 and Proposition 4.13.

Corollary 4.14. *If $A \subseteq \{0,1\}^\ell$ is k-regular, and if*

$$|A| = \binom{a_k}{k} + \binom{a_{k-1}}{k-1} + \ldots + \binom{a_s}{s}$$

then

$$|\partial^{\leq k}(A)| \geq \sum_{j=0}^{k} \binom{a_k}{k-j} + \binom{a_{k-1}}{k-1-j} + \ldots + \binom{a_s}{s-j} \ .$$

We are now ready to prove Claim 4.11.

Proof of Claim 4.11. Given $\mathcal{U}$, define $W_\mathcal{U}$ to be the largest set of colourings $\{0,1\}^\ell$ such that $\{b\}^{|U|} \notin \pi_U(W)$ for all $U \in \mathcal{U}$. For simplicity, we consider $b = 0$, i.e. $W_\mathcal{U}$ is the set that contains all the colourings of $[\ell]$ that do not colour any $U \in \mathcal{U}$ completely 0.

Let $\mathcal{U}' \subseteq \mathcal{U}$ be a set of size exactly $\sum_{i=0}^{t} \binom{\ell-1-i}{\ell-u-i}$. Obviously, $W_{\mathcal{U}'}$ is at least as large as $W_\mathcal{U}$.

Let $\mathbf{1}_{U^C} \in \{0,1\}^\ell$ be the indicator functions for the complement of a set U, i.e. $\mathbf{1}_{U^C} = 1 - \mathbf{1}_U$. Let A be the set of $\mathbf{1}_{U^C} \in \{0,1\}^\ell$ for $U \in \mathcal{U}'$. Note that A is $(\ell - u)$-regular and that the iterated shadow of A is exactly the set of colourings that are not in $W_{\mathcal{U}'}$. Applying Corollary 4.14 to A, we get

$$|\partial^{\leq k}(A)| \geq \sum_{j=0}^{k}\sum_{i=0}^{t} \binom{\ell-1-i}{k-i-j} = \sum_{j=0}^{\ell-u}\sum_{i=0}^{t} \binom{\ell-1-i}{\ell-u-i-j}.$$

Therefore, $|W_\mathcal{U}| \leq |W_{\mathcal{U}'}| = 2^\ell - |\partial^{\leq k}(A)| \leq 2^\ell - \sum_{j=0}^{\ell-u}\sum_{i=0}^{t} \binom{\ell-1-i}{\ell-u-i-j}$. ☐

For completeness we include the calculations needed in Claim 4.9.

Claim 4.15. It holds that

$$\sum_{j=0}^{\ell-u}\sum_{i=0}^{t} \binom{\ell-1-i}{\ell-u-i-j} \geq 2^\ell - 2^{\ell-t+1} + 2^{u\log \ell} \ .$$

Proof. The claim follows from the following sequence of elementary calculations

$$\begin{aligned}
&\sum_{j=0}^{\ell-u}\sum_{i=0}^{t}\binom{\ell-1-i}{\ell-u-i-j} \\
&=\sum_{j=0}^{\ell-u}\sum_{i=0}^{t}\binom{\ell-1-i}{\ell-1-j} && (*)\\
&=\sum_{j=0}^{\ell-u}\left(\sum_{i=0}^{\ell-1}\binom{i}{\ell-1-j}-\sum_{i=0}^{\ell-t}\binom{i}{\ell-1-j}\right) \\
&=\sum_{j=0}^{\ell-u}\left(\binom{\ell}{\ell-j}-\binom{\ell-t+1}{\ell-j}\right) && (**)\\
&=\sum_{j=0}^{\ell-u}\binom{\ell}{j}-\sum_{j=0}^{\ell-u-t+1}\binom{\ell-t+1}{j} && (*)\\
&=2^{\ell}-\sum_{j=\ell-u+1}^{\ell}\binom{\ell}{j}-2^{\ell-t+1}+\sum_{j=\ell-u-t+2}^{\ell-t+1}\binom{\ell-t+1}{j} \\
&\geq 2^{\ell}-2^{\ell-t+1}+\sum_{j=0}^{u-1}\binom{\ell}{j} \\
&\geq 2^{\ell}-2^{\ell-t+1}+\ell^{u}=2^{\ell}-2^{\ell-t+1}+2^{u\log\ell}\quad ,
\end{aligned}$$

where the equalities in $(*)$ follow from renaming of variables and the fact that $\binom{n}{k}=\binom{n}{n-k}$; and the equality in $(**)$ follows from $\sum_{i=0}^{n-1}\binom{i}{k-1}=\binom{n}{k}$. ☐

4.2 Simulation of Real Communication Protocols by Decision Trees

In this section we show how to adapt the simulation theorem to real communication.

Theorem 4.16. *If there is a real communication protocol solving $Lift(S)$ using communication c and r rounds, then there is a parallel decision tree solving S using $\mathrm{O}(c/\log\ell)$ queries and depth r.*

The proof follows the same strategy as in the deterministic case, that is, we are going to construct a decision tree by simulating a real communication protocol and only querying the coordinates where the communication protocol would have too much information on x_i.

The major difference in analyzing real communication protocols as opposed to deterministic ones is that the set of compatible inputs is not a rectangle, but a monotone set as defined next.

Definition 4.17. A Boolean matrix M is *monotone* if $M_{i_1 j_1} \leq M_{i_2 j_2}$ for all pairs of entries such that $i_1 \leq i_2$ and $j_1 \leq j_2$. A set is monotone if it is the monochromatic set of some monotone matrix.

Recall that each communication step is a comparison $\phi(x) \leq \psi(y)$ and that we restrict our attention to inputs in a set $A \times B$. We lay out the results of the comparison in the matrix $(\llbracket\phi(x) \leq \psi(y)\rrbracket)_{x,y}$ indexed by $x \in A$, $y \in B$, with rows sorted decreasingly according to ϕ and columns increasingly according to ψ. Note that we use the Iverson bracket notation

$$\llbracket Z\rrbracket = \begin{cases} 1 & \text{if the Boolean expression } Z \text{ is true;} \\ 0 & \text{otherwise.} \end{cases} \tag{4.24}$$

The comparison matrix $(\llbracket\phi(x) \leq \psi(y)\rrbracket)_{x,y}$ is monotone: if $x_1 \leq x_2$ and $y_1 \leq y_2$ according to our order, that is $\phi(x_1) \geq \phi(x_2)$ and $\psi(y_1) \leq \psi(y_2)$, and if $M_{x_1,y_1} = 1$, then $\phi(x_2) \leq \phi(x_1) \leq \psi(y_1) \leq \psi(y_2)$, and thus $M_{x_2,y_2} = 1$.

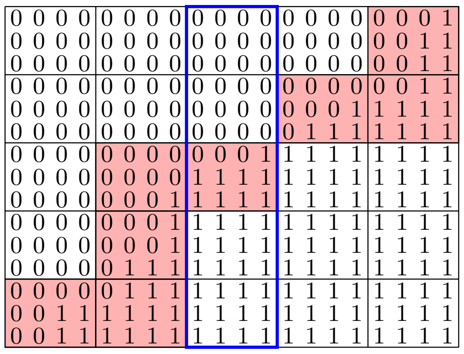


Figure 8: Monotone matrix partitioned in 5×5 blocks; the 3rd block-column has 4 monochromatic blocks.

The fact that the set of compatible inputs is not a rectangle can be circumvented, since as observed in [Joh98] in every monotone Boolean matrix there exists one quadrant—hence a large rectangle—that is monochromatic. Therefore, when we want to choose the outcome of a comparison, it is possible to restrict the set of compatible inputs to a quadrant as done in [BEGJ00].

This simple solution, however, is not enough for us. Since we want to query variables only at the end of each round of $k = k_v$ comparisons and since after using procedure `project` we no longer know what B is, we need to restrict the inputs to rectangles beforehand. If we did so by picking a monochromatic quadrant for each comparison, the size of the set A would decrease by a factor 2^k, and potentially so would its average degree. The problem is that the thickness lemma only works with a constant decrease in average degree. We could tweak the thickness lemma to handle up to a polynomial decrease, corresponding to revealing a constant fraction of information about an index, but not an exponential decrease.

Our solution to avoid A shrinking too much is to partition the matrix into $(k+1) \times (k+1)$ blocks of size $|A|/(k+1) \times |B|/(k+1)$ and then restrict Bob's input to one of the $(k+1)$ block-columns, so that Alice's input forms a rectangle in k out of the $k+1$ block-rows (see Figure 8). Formally, we have the following lemma.

Lemma 4.18. *Let M be a monotone matrix partitioned into $(k+1) \times (k+1)$ blocks. There is a block-column such that k of its blocks are monochromatic.*

Proof. Consider a non-monochromatic block $M_{i,j} = M_{[i,i+|A|/(k+1)) \times [j,j+|B|/(k+1))}$. Since its bottom-right corner has value 1, all blocks $\{M_{i',j'} \mid i' > i \text{ and } j' > j\}$ strictly below and to the right of $M_{i,j}$ are 1-blocks.

Now consider all non-monochromatic blocks sorted in a sequence left-to-right, ties broken down-to-up, and fix two consecutive blocks $M_{i,j}$ and $M_{i',j'}$ in this sequence. If $j' = j$ then $i' < i$ by the choice of ordering; otherwise $j' > j$ and, since all blocks with $i' > i$ and $j' > j$ are 1-blocks, it must hold that $i' \leq i$. This implies that the quantity $j' - i'$ is strictly increasing and, since it ranges between $-k$ and k, that there are at most $2k+1$ non-monochromatic blocks overall. By the pigeonhole principle, at least one of the column-blocks contains at most one non-monochromatic block. $\square$

We take advantage of this lemma with the following construction. Given two sets $A \subseteq [\ell]^m$ and $B \subseteq \{0,1\}^{\ell m}$, we define the b-monochromatic part of A with respect to B as $A[b,B]_{\phi,\psi} = \{x \in A : \forall y \in B,\ [\![\phi(x) \leq \psi(y)]\!] = b\}$.

In order to measure the size of $\pi_I(A)$ directly from A we would like each element in $\pi_I(A)$ to have a unique completion to A. Since that is not true in general, we use in place of A a new set $\sigma_I(A)$ that has that property. We define $\sigma_I(A) = \{x \in A : \forall x' \in A \text{ if } \pi_I(x') = \pi_I(x) \text{ then } x < x'\}$, where the order is, say, the lexicographic order. In other words, each element of $\sigma_I(A)$ is the minimum among all elements of A that share the same I coordinates. Observe that $\pi_I(A) = \pi_I(\sigma_I(A))$. We define $\sigma_I(B)$ analogously.

We have all the ingredients to explain the simulation procedure `eval`. Note that for each node v, we are considering the comparisons done at v twice: once at line 4 to extract rectangles from monotone sets

```
1  let $A = [\ell]^m$, $B = \{0,1\}^{\ell m}$, $I = [m]$, $v$ be the root of $\Pi$, $s = \emptyset$ be a string
2  while $v$ is not a leaf do
3      let $A' = \sigma_I(A)$, $B = \sigma_I(B)$
4      foreach comparison $\phi_{v,j}$ vs $\psi_{v,j}$ at $v$ do
5          let $B = \arg\max_{B' \subset B : |B'| = |B|/(k_v+1)} |A'[0,B']_{\phi_{v,j},\psi_{v,j}} \cup A'[1,B']_{\phi_{v,j},\psi_{v,j}}|$
6          let $A' = A'[0,B]_{\phi_{v,j},\psi_{v,j}} \cup A'[1,B]_{\phi_{v,j},\psi_{v,j}}$
7      let $A = \texttt{prune}(A', I)$
8      let $Q = \emptyset$, $C_I = \pi_I(B)$
9      foreach comparison $\phi_{v,j}$ vs $\psi_{v,j}$ at $v$ do
10         while $\exists i \in I$ such that $AvgDeg_i(\pi_I(A)) < \ell^\lambda$ do
11             let $U_i$ = project $(A, C_I, I, i)$
12             let $A = \rho_{i,U_i}(A)$, $C_{I\setminus\{i\}} = {C_{I\setminus\{i\}}}^{(0)}(U_i) \cap {C_{I\setminus\{i\}}}^{(1)}(U_i)$,
                   $I = I \setminus \{i\}$, $Q = Q \cup \{i\}$
13         let $b = \arg\max_b |\pi_I(A[b,B]_{\phi_{v,j},\psi_{v,j}})|$
14         let $A = \texttt{prune}(A[b,B]_{\phi_{v,j},\psi_{v,j}}, I)$, $s = sb$
15     query coordinates $Q$ to get string $z_Q$
16     for $i \in Q$ do
17         let $B = \rho_{i,V}(B)$, where $V = V^{z_i}(U_i)$
18     let $v = v_s$, $s = \emptyset$
19 return the answer at $v$
```

Figure 9: Procedure eval(Π,z)

of inputs, and then again at line 9, which is similar to how we handled Alice speaking in the deterministic case. The operation sb at line 14 simply appends the bit b to the string s. The simulation theorem (Theorem 4.16) follows from the next lemma.

Lemma 4.19 (Main Lemma). *If Π is a real protocol that solves Lift(S) using communication $c < \frac{m}{2}(1-\lambda)\log \ell$ and r rounds, then* eval *solves S using $5c/(1-\lambda)\log \ell$ queries and depth r.*

Proof. The proof is very similar to the proof of Lemma 4.6. Let R^v be the set (not necessarily a rectangle) of inputs compatible with node v. Let c_v (resp. r_v) be the amount of communication (resp. rounds) up to node v, and recall that k_v is the number of comparisons at node v. We show that the following invariants hold throughout the algorithm:

1. $\pi_I(A)$ is thick;
2. $A \times B \subseteq R^v$;
3. $m - |I| \leq (2(c_v + |s|) + 3r_v)/(1-\lambda)\log \ell$;
4. $\beta(C_I) \leq c_v \log(c_v + 1) + k_v \log(k_v + 1) + m - |I|$;

and that the following invariants hold at the beginning of each round:

5. $\beta(\pi_I(B)) \leq c_v \log(c_v + 1) + m - |I|$;
6. $\mathsf{Ind}(x_i, y_i) = z_i$ for all $(x,y) \in A \times B$ and $i \notin I$.

All six invariants are true at the beginning of the algorithm. As before, we assume invariants 1 through 4 are true up to the current point of the algorithm, and invariants 5 and 6 are true at the beginning of the current round. We then show that, after executing the next line, 1 through 4 still hold and, at the beginning of the next round, 5 and 6 still hold.

The main difference from the proof of Lemma 4.6 is in proving that invariant 1 holds. This is because in Algorithm 9 we modify A not only at lines 12 and 14 (analogous to lines 7 and 9 in Algorithm 4), but also at line 7. For line 12, it is enough to argue that the assumption of Lemma 4.5 applies since Lemma 4.5 guarantees that $\pi_I(A)$ is thick. This is indeed the case, since we assume invariant 1 holds before this point and since invariants 4 and 3 together with the assumption of Lemma 4.19 that $c_v+k_v \leq \frac{m}{2}\log\ell = \frac{\ell^\gamma}{2}\log\ell$ imply that

$$\beta(C_I) \leq (c_v+k_v)\cdot\log(c_v+1) + \frac{2(c_v+|s|)+3r_v}{(1-\lambda)\log\ell} \leq (c_v+k_v)(2\log m+5) \leq 2\ell^\gamma\log^2\ell \ . \tag{4.25}$$

For lines 7 and 14, it is enough to argue that the assumption of Lemma 4.4 applies in each case. For line 14 the argument is the same as that in the proof of Lemma 4.6: by the choice of b we have that, before line 14,

$$|\pi_I(A[b,B]_{\phi_{v,j},\psi_{v,j}})| \geq \frac{1}{2}|\pi_I(A)| \tag{4.26}$$

and therefore,

$$AvgDeg_i(\pi_I(A[b,B]_{\phi_{v,j},\psi_{v,j}})) \geq \frac{1}{2}AvgDeg_i(\pi_I(A)) \geq \frac{\ell^\lambda}{2} \ , \tag{4.27}$$

where we again use the assumption that invariant 1 holds before modifying A.

We now prove that the assumption of Lemma 4.4 holds at line 7. Denote by A'' the set A' at line 3. We begin by observing that $AvgDeg_i(\pi_I(A'')) \geq \ell^\lambda$, since, at line 3, $\pi_I(A'')=\pi_I(A)$ by definition of $\sigma_I(A)$ and $\pi_I(A)$ is thick by invariant 1. It is enough to argue that, at line 7,

$$\pi_I(A') \geq \frac{1}{4}\pi_I(A'') \tag{4.28}$$

since this would imply that

$$AvgDeg_i(\pi_I(A')) = \frac{|\pi_I(A')|}{|\pi_{I\setminus\{i\}}(A')|} \geq \frac{\frac{1}{4}|\pi_I(A'')|}{|\pi_{I\setminus\{i\}}(A'')|} = \frac{1}{4}AvgDeg_i(\pi_I(A'')) \geq \frac{\ell^\lambda}{4} \ . \tag{4.29}$$

In order to do so, note that at line 6, the size of A' decreases by at most a $1-1/(k_v+1)$ factor. Indeed, if we divide the comparison matrix $([\![\phi(x)\leq\psi(y)]\!])_{x,y}$ into $(k_v+1)\times(k_v+1)$ blocks of size $|A'|/(k_v+1)\times|B|/(k_v+1)$, by Lemma 4.18 at least one of the column-blocks contains k_v monochromatic blocks, i.e., a $1-1/(k_v+1)$ fraction is monochromatic. Since we have at most k_v comparisons, the size of A' at line 7 is at least a $(1-1/(k_v+1))^{k_v} \geq 1/4$ fraction of the size of A''. By definition of $\sigma_I(A)$, there is a bijection between A'' and $\pi_I(A'')$ and this bijection is maintained until line 7, so $|A''|=|\pi_I(A'')|$ and $|A'|=|\pi_I(A')|$ and, therefore, (4.28) indeed holds.

For invariant 2, first note that A and B never increase. Moreover, the set of compatible inputs R^v only changes when v is modified at line 18, and, before this, A is restricted at line 14 so that for each comparison $\phi_{v,j}$ vs $\psi_{v,j}$ it holds that $[\![\phi_{v,j}(x)\leq\psi_{v,j}(y)]\!]=b$ for every $x\in A$ and $y\in B$; in other words, the restriction guarantees that $A\times B\subseteq R^{v_s}$, therefore invariant 2 holds. Note that we can only restrict A in this manner because we had already restricted B at line 5.

To see that invariant 3 holds we show that

$$\alpha(\pi_I(A)) \leq (2(c_v+|s|)+3r_v) - (m-|I|)(1-\lambda)\log\ell \ , \tag{4.30}$$

and the invariant follows by the nonnegativity of α. The RHS of (4.30) only decreases when I is updated in line 12 after a call to `project`, which in turn, by Lemma 4.5, decreases $\alpha(\pi_I(A))$ by at least $(1-\lambda)\log\ell$ (since, before updating A and I, $AvgDeg_i(A_I) < \ell^\lambda$). Meanwhile, $\alpha(\pi_I(A))$ only increases at line 14, i.e., once per comparison, and at line 7, i.e., once per round: by Lemma 4.4 and recalling (4.26) and (4.28), at line 14 the increase is of at most 2, which $2(c_v+|s|)$ accounts for; and at line 7 of at most 3, which $3r_v$ accounts for.

To prove invariant 4, first note that C_I is updated only at lines 8 and 12. To show the invariant holds after line 8, we use the assumption that item 5 holds at the beginning of the round. Note that, since

$\pi_I(B) = \pi_I(\sigma_I(B))$, item 5 still holds after line 3. Now, after the k_v executions of line 5, B decreases in size by a factor of $(k_v+1)^{k_v}$. Since, by definition of $\sigma_I(B)$, there is a bijection between B and $\pi_I(B)$, it holds that $\pi_I(B)$ also decreases by a factor of $(k_v+1)^{k_v}$, so $\beta(\pi_I(B))$ increases by $k_v \cdot \log(k_v+1)$. Therefore, at line 8, $\beta(C_I) = \beta(\pi_I(B)) \le c_v \log(c_v+1) + k_v \log(k_v+1) + m - |I|$ and invariant 4 holds. At line 12, invariant 4 holds since Lemma 4.5 guarantees that $\beta(C_{I\setminus\{i\}}) \le \beta(C_I)+1$.

As for invariant 5, by the exact same argument as in the proof of of Lemma 4.6 we conclude that at the end of a round $\beta(\pi_I(B)) \le \beta(C_I)$. Invariant 4 implies $\beta(C_I) \le c_v \log(c_v+1) + k_v \log(k_v+1) + m - |I|$ and, since c_v is updated to $c_v + k_v$ when v is updated at line 18, invariant 5 holds at the beginning of the next round.

For invariant 6, recall that A and B never increase. Moreover, each time I is modified at line 12, we add the coordinate i for which invariant 6 possibly breaks to Q. Then we restore the invariant before the next iteration by restricting B at line 17. Indeed, if $(x,y) \in A \times B$, then $x_i \in U$ and $y_i \in V^{z_i}(U)$, so by definition of $V^{z_i}(U)$ it holds that $\mathsf{Ind}(x_i, y_i) = z_i$.

It is clear that the decision tree has depth r and, by invariant 3, that the total number of queries is at most $5c/(1-\lambda)\log \ell$. The proof of correctness is identical to that of Lemma 4.6. □

5 From Parallel Decision Trees to Dymond–Tompa Games

In this section we prove that the adversary argument on a parallel decision tree for the falsified search problem of a pebbling contradiction gives a Pebbler strategy for the Dymond–Tompa game.

It is more convenient to work with the Dymond–Tompa game when there is a challenged pebble at all times. Therefore in this and the following section we use an alternative but equivalent definition. Initially the unique sink has a pebble and it is challenged, and then the game starts without a special first round. The number of rounds is the number of actual rounds, not counting the setup, and the cost is the total number of pebbles, including the initial pebble on the sink.

Lemma 2.6 (Restated). *If there is a parallel decision tree for $Search(Peb_G)$ in depth r using at most c queries, then Pebbler has a winning strategy in the r-round Dymond–Tompa game on G in cost at most $c+1$.*

We prove that, in fact, the parallel decision tree complexity of the falsified clause search problem of a pebbling contradiction is equivalent to the Dymond–Tompa game on the graph with an extra sink on top. Formally, we define $\widehat{G}$ as a graph with vertices $V(G) \cup \{t'\}$ and edges $E(G) \cup \{(t,t')\}$, where t is the unique sink of G. Clearly the game on $\widehat{G}$ needs as many pebbles as G, and one more pebble is enough, so Lemma 2.6 follows from Lemma 5.1.

Lemma 5.1. *There is a parallel decision tree for $Search(Peb_G)$ in depth r using c queries if and only if Pebbler has a winning strategy in the r-round Dymond–Tompa game on $\widehat{G}$ in cost $c+1$.*

Proof. Assume there is a parallel decision tree for $Search(Peb_G)$ in depth r using c queries. We construct a strategy for Pebbler in r rounds and $c+1$ pebbles. We say that a vertex u reaches a vertex v if there is a (possibly empty) path from u to v where all intermediate vertices are not queried. We keep these invariants.

1. The challenged pebble in the Dymond–Tompa game is false.
2. A false vertex is reachable from another false vertex if and only if it is not challenged in the Dymond–Tompa game.
3. In the subtree of the challenged pebble a vertex has a pebble if and only if it has been queried.

When it is Pebbler's turn, Pebbler looks at the decision tree and places pebbles in those vertices being queried that can reach the challenged pebble. After Challenger's turn, Pebbler follows the branch in the decision tree in which the challenged pebble is false and other vertices are false if they are reachable from a false vertex or true otherwise.

Dymond–Tompa moves are valid and the invariants are kept. When we reach a leaf in the decision tree we made at most c queries in r rounds by assumption, therefore Pebbler also used at most c pebbles on vertices of G plus one pebble on the extra sink and r rounds. It remains to show that the Dymond–Tompa game also ended. The decision tree points to a falsified clause, which is not the sink axiom because the sink is always false. Therefore we have a false vertex whose predecessors are true. By item 2, that false pebble is challenged, and by item 3 all of its predecessors have pebbles, therefore the Dymond–Tompa game also ended.

To prove the opposite direction, assume there is a Pebbler strategy in r rounds and $c+1$ pebbles. We construct a parallel decision tree for $\mathit{Search}\left(\mathit{Peb}_G\right)$ in depth r using c queries by keeping the same three invariants as before.

We look at the strategy for Pebbler and add a node to the decision tree that queries the variables corresponding to vertices being pebbled that can reach the challenged pebble. Each branch can be viewed as a true-false colouring of the queried vertices. For each branch, we simulate a Challenger move. We consider the set of new vertices coloured false and that are not reachable by any false vertex other than itself. If this set is empty, then Challenger stays. Otherwise Challenger jumps to any of these vertices.

Dymond–Tompa moves are valid and the invariants are kept. When the Dymond–Tompa game ends, Pebbler has used at most $c+1$ pebbles in r rounds by assumption, one of which outside G, therefore the decision tree also made at most c queries in r rounds. It remains to show that we can label the leaves of the decision tree in such a way that the assignment induced by the decision tree falsifies a clause. At the end of the Dymond–Tompa game, all of the predecessors of the challenged pebble have pebbles. By item 1 the challenged vertex is false, and by item 3 its predecessors are queried. By item 2, its predecessors are true, therefore we can label the leaf with the clause claiming that if the predecessors of the challenged vertex are true then the challenged vertex is true. ☐

6 Dymond–Tompa Trade-offs

In this section we prove upper and lower bounds for the Dymond–Tompa game on graphs of a given family. The lower bounds are the final missing piece in order to get length-space trade-offs for cutting plane proofs, and the upper bounds will be used to obtain space-efficient proofs, as explained in Section 7.

Our goal is to prove the following lemma.

Lemma 6.1. *For any $n, d \in \mathbb{N}^+$ such that n is a power of 2, there exists an explicitly constructible DAG $G(n,d)$ of depth d with $\mathrm{O}(dn)$ vertices and indegree at most 2 such that:*

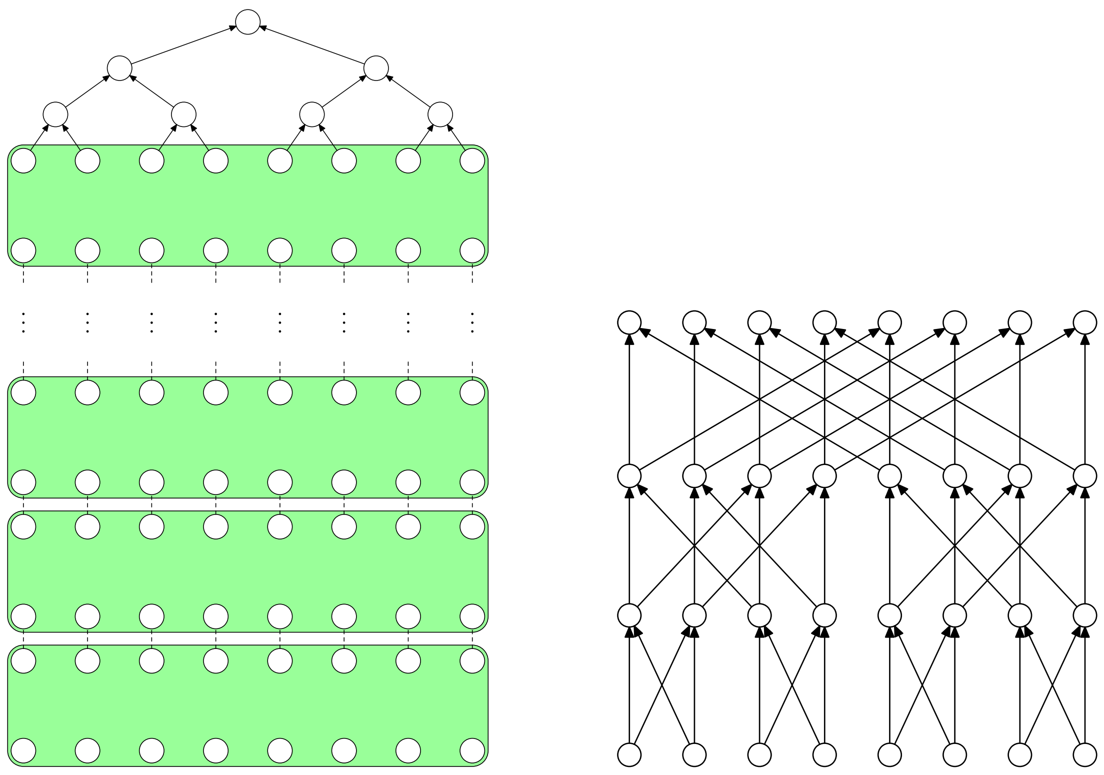
1. *for any $r \leq d$, the cost of an r-round DT game is at most $\min\{r2(2^{\lceil d/r\rceil}-1), rn\lceil d^{1/r}-1\rceil\}$;*
2. *for any $r \leq d$, the cost of an r-round DT game is at least $\min\{\frac{r2^{d/r}}{8}, \frac{n}{8}\}$.*

We first define a family of graphs for which we will prove the lemma.

Definition 6.2 (Butterfly graph). A *k-dimensional butterfly graph G* is a DAG with vertices labelled by pairs (w,i) for $0 \leq w \leq 2^k-1$ and $0 \leq i \leq k$, and with edges from vertex (w,i) to $(w',i+1)$ if the binary representations of w and w' are equal except for possibly in the $(i+1)$st most significant bit. Note that G has $(2^k+1)k$ vertices, has 2^k sources and 2^k sinks, and that all vertices that are not sources have indegree two.

Moreover, if H is a graph with n sinks and n sources, we say a graph is a stack of s Hs, if it consists of s copies of H such that sources on level i are identified with sinks on level $i+1$ for $i \in \{1,\ldots,s-1\}$.

For any $n, d \in \mathbb{N}$ such that n is a power of 2, the graph $G(n,d)$ we will consider for the Dymond–Tompa game consists of a (possibly fractional) stack of butterfly graphs of dimension $\log n$, with an attached binary tree on top such that the depth of this graph is exactly d (see Figure 10a). Note that if d is



(a) Stacks of graphs with binary tree on top (dashed lines represent vertex identification)

(b) 3-dimensional butterfly graphs

Figure 10: Stack of butterflies

a multiple of $\log n$, then this graph has exactly $d/\log n$ blocks (the 1st block is a binary tree). Moreover, if $d \le \log n$, then $G(n,d)$ is just a binary tree of depth d. Observe that, if $d \ge \log n$, $G(n,d)$ has $(d-\log n)n + 2n - 1$ vertices.

6.1 Upper Bounds for the Cost of the Dymond–Tompa Game on Butterfly Graphs

To prove item 1 of Lemma 6.1 we describe two Pebbler strategies, one that is efficient when many rounds are available and that we use as part of building a space-efficient refutation, and one that is efficient when we only have few rounds and that we include for completeness.

The first strategy is to chase the challenged pebble down the graph.

Lemma 6.3. *Every graph of depth d and indegree 2 has an r-round Pebbler strategy of price at most $r2(2^{\lceil d/r\rceil}-1))$.*

Proof. Let $k = \lceil d/r \rceil$. At each round Pebbler pebbles all the ancestors at distance at most k of the challenged vertex.

The strategy lasts for at most r rounds: if Challenger stays or jumps to a vertex at distance less than k the game ends immediately, hence we can assume that Challenger jumps to a vertex at distance exactly k from the challenged vertex. After r rounds the new challenged vertex would be at distance $rk \ge d$ from the root, hence either it does not exist or it is a source, and in both cases the game ends.

The number of pebbles used in each round is at most the number of vertices in a binary tree of depth k, except for the root, that is $2^{k+1}-2$. □

Corollary 6.4. *Every graph of depth d and indegree 2 has a d-round Pebbler strategy of price at most $2d$.*

The second strategy is essentially a binary (or rather $\lceil d^{1/r}\rceil$-ary) search.

Lemma 6.5. *There is an r-round Pebbler strategy for the graph $G(n,d)$ of price at most $rn\lceil d^{1/r}-1\rceil$.*

Proof. Throughout the game Pebbler keeps in mind a subgraph H, isomorphic to a stack of butterflies, with the invariant that the root of H is challenged and the predecessors (in G) of the sources of H, if they exist, are pebbled. Initially $H = G$.

Let $k = \lceil d^{1/r} \rceil$. Each round Pebbler chooses $k-1$ equally-spaced layers in H (up to rounding) and places pebbles on every vertex in these layers. This divides the graph into k parts, where part 0 spans the challenged pebble up to the layer before the first pebbled layer, and part i spans the i-th pebbled layer up to the layer before the $i+1$-st pebbled layer. If Challenger stays then Pebbler updates H to be the 0-th part, while if Challenger jumps to a vertex on the i-th pebbled layer then Pebbler chooses the i-th part restricted to ancestors of the challenged vertex.

After r rounds, H has depth $\lceil d/k^r \rceil = 1$, hence the challenged pebble is its only vertex. Since by construction all predecessors of sources of H are pebbled, the game ends. Because $G(n,d)$ has at most n vertices per row, the total cost of the strategy is $rn(k-1)$. □

We can view the strategy of Lemma 6.3 as an example of a more general meta-strategy to reduce the number of rounds of another strategy. We can divide the number of rounds by ℓ by looking ℓ levels deep into the strategy tree at each round, then playing the union of all possible placed pebbles at once.

If we start with the strategy from Corollary 6.4 where we just pebble the two predecessors of the challenged pebble, applying this meta-strategy results in pebbling the $2^{\ell+1} - 2$ vertices at distance ℓ, as in Lemma 6.3.

If we apply the round-reducing meta-strategy to a binary search, then we obtain a 2^ℓ-ary search, which coincides with the strategy of Lemma 6.5 when k is a power of 2. Note that since many branches of the strategy tree overlap—because the strategy depends on which layer the challenged pebble is in, but not on which column—the number of pebbles per round goes from n to only $2^\ell n$ instead of the worst case of n^ℓ.

6.2 Lower Bounds for the Cost of the Dymond–Tompa Game on Butterfly Graphs

Now we would like to show that the strategies described in the previous subsection are essentially the best Pebbler can do. As a warm up, and to give some intuition on the strategy, we prove a special case of Lemma 6.1. In order to keep the proof simple, we use the alternative definition of the Dymond–Tompa game and consider a stack of butterflies with an extra vertex on top, $\widehat{G}$, as defined in Section 5.

Lemma 6.6. *For any $n, r \in \mathbb{N}^+$ such that n is a power of 2, there exists an explicitly constructible DAG $\widehat{G}(n, r\log n)$ of depth $r\log n + 1$ with $\mathrm{O}(nr\log n)$ vertices and indegree at most 2 such that for any $r \leq d$, the cost of an r-round DT game is at least $\frac{n}{4}$.*

This lemma holds not only for stacks of butterfly graphs, but also for stacks of other kinds of graphs as long as they have a strong version of the grate property [Val77].

Definition 6.7. A graph with n sources and sinks is an *α-uniform grate* if after removing α vertices, there still are at least $n/2+1$ sources, each of which can reach $n/2+1$ sinks.

Throughout the Dymond–Tompa game, we say a vertex t is reachable from s if there is a path from s to t with no pebbles neither on internal vertices of the path nor on the vertex s (but t may be pebbled). We say a sink at level ℓ is *good* if it is unpebbled and is reachable by at least $n/2+1$ sources at level ℓ. Furthermore, we say a source s is disconnected from a sink t if there is no completely (including end points) unpebbled path from s to t, and we consider the number of source-sink disconnections in a graph as the number of pairs (t, s) such that s is disconnected from t.

Observation 6.8. *Butterfly graphs are $(n/4-1)$-uniform grates.*

Proof. If there are less than $n/2+1$ good sink-vertices, then the number of source-sink disconnections is at least $n/2 \cdot n/2$ (at least $n/2$ non-good sinks are not reached by at least $n/2$ sources). Note that any vertex in a butterfly graph is in exactly n distinct source-sink paths. So if α is the number of vertices removed, then there are at most αn source-sink disconnections. This implies that $\alpha \geq n/4$. □

We can now proceed to the proof of the warm-up lemma.

Proof of Lemma 6.6. We give a strategy for Challenger in the Dymond–Tompa game over the graph $\widehat{G}(n, r \log n)$ defined above so that, assuming Pebbler has at most $n/4-1$ pebbles, the game will not end within r rounds.

At a high level, Challenger's strategy will be to keep in mind, before every round, a good sink that can reach the challenged vertex; more precisely, before round $\ell+1$ Challenger will have a good sink at level ℓ in mind, say t_ℓ. After the Pebbler places pebbles on the graph, Challenger chooses a good sink at level $\ell+1$ that is reachable from t_ℓ and decides to have that in mind. He will then check if there are any new pebbles that are causing the challenged vertex to be unreachable from t_ℓ, and if so, challenges one that is reachable from t_ℓ.

The proof goes as follows. We maintain the invariant that before round ℓ, Challenger's chosen vertex t_ℓ is a good sink at level ℓ and reaches the challenged vertex. Before the first rounds, the challenged vertex is the sink of $\widehat{G}$ (the vertex that is not in G) and Challenger's chosen vertex is the original sink of G, t_1, which clearly is a good sink at level 1 (it is unpebbled and reachable from all sources at level 1) and reaches the challenged vertex.

Suppose that the invariant was true until round ℓ, i.e. suppose that before Pebbler's $(\ell-1)$st move, Challenger's chosen vertex $t_{\ell-1}$ is a good sink at level $\ell-1$ and reaches the challenged vertex. At round $\ell-1$, Pebbler places some pebbles. Since Pebbler has at most $n/4-1$ pebbles in total, we can conclude by the uniform grate property that there are at least $n/2+1$ good sinks at level ℓ. Since $t_{\ell-1}$ was a good sink before round $\ell-1$, there must be a good sink at level ℓ, say t_ℓ, which was reachable from $t_{\ell-1}$ before round $\ell-1$. Challenger decides t_ℓ will be the next chosen vertex. Since before round $\ell-1$, t_ℓ reached $t_{\ell-1}$ and $t_{\ell-1}$ reached the challenged vertex, the only possible pebbles that are disconnecting t_ℓ from the challenged vertex are the newly put pebbles. If there are no such blocking pebbles, i.e., if t_ℓ reaches the challenged vertex, Challenger stays. If there are newly put pebbles that block all paths from t_ℓ to the challenged vertex, Challenger challenges one that is reachable from t_ℓ. Thus, before round ℓ, Challenger's chosen vertex t_ℓ is a good sink at level ℓ and reaches the challenged vertex, and the invariant is maintained.

We conclude that before round $(r+1)$, Challenger's chosen vertex t_{r+1} is an unpebbled vertex global source (which would have been a good sink at level $r+1$, if such a level had existed) that reaches the challenged vertex, and hence the game has not ended. □

Now to prove the lower bound in Lemma 6.1 in its full generality, we must allow any number of rounds (at most the depth) and still get a good bound on the cost of the game. We again describe a strategy for Challenger; the difference is that Challenger cannot afford to jump $\log n$ rows every round. Intuitively, we do not think of the graph as a stack of blocks, but as a continuous stream such that any consecutive $\log n$ rows is isomorphic to a butterfly graph.

Note that given any vertex $v \in V(G(n,d))$ at distance d' from the set of sources, the subgraph induced by all vertices that reach v is isomorphic to $G(n,d')$. We therefore refer to the top binary tree of the subgraph $G(n,d')$ as the tree induced by the vertices that reach v and are at distance at most $\log n$ from v.

We give a more general definition of a good vertex and define a partially good vertex. Let T be a complete directed binary tree rooted at v. We say v (or T) is good if v can be reached by strictly more than half of the leaves. If T has n leaves this is equivalent to requiring that, for any $h' \leq \log n$, v can be reached by strictly more than $2^{h'}/2$ vertices at distance h' from v. Given a vertex $u \in V(T)$ at distance h from the leaves, we say u (or the subtree of T rooted at u) is T-partially good if, for any $h' \leq h$, u can be reached by strictly more than $2^{h'}/2$ vertices at distance h' from u. When T is clear from the context, we just say u is partially good.

We are now ready to prove the lower bound.

Proof of Lemma 6.1, item 2. We actually prove something stronger: we allow Pebbler to place some pebbles before the game begins, provided that the top binary tree remains good. We charge only for the pebbles placed outside the binary tree. Challenger is not allowed to challenge any pebble that was placed in this initial stage. We denote this game DT*.

Formally, we prove the following claim. Given a graph G and a challenged vertex on this graph, if there is a vertex v in G that reaches the challenged vertex and that is the sink of a graph $G(n,d)$, then the cost of the r-round DT* game on G is at least $\min\{\frac{\gamma 2^{d/\gamma}}{8}, \frac{n}{8}\}$, where $\gamma = \min\{d, r\}$.

We prove this claim by induction on γ. For $\gamma = 1$, either $d > r = 1$ or $d = 1$. If $d > r = 1$, $G(n,d)$ consists of at least a binary tree of depth $d' = \min\{d, \log n\}$ with $2^{d'+1} - 1$ vertices and such that the sink reaches the challenged vertex. Since after Pebbler places the initial pebbles the binary tree must be a good tree, at least half of the tree reaches the challenged vertex (actually, strictly more than half of the pebbles in every row must reach the challenged vertex, which makes a total of at least $2^{d'} + d'$ vertices that reach the challenged vertex). Clearly Pebbler must pebble all the vertices that reach the challenged vertex in order to finish the game in one round, therefore the cost is more than $\min\{\frac{2^d}{8}, \frac{n}{8}\}$. If $d = 1$, then clearly at least 1 pebble is needed and $1 \geq 1/4 = \gamma 2^{d/\gamma}/8$, so the base case holds.

Now suppose $\gamma \geq 2$ and that Pebbler has placed some initial pebbles on the graph, but maintaining the top binary tree good. Pebbler then starts the first round by placing some pebbles. Let $x \geq 1$ be the number of pebbles Pebbler placed in the top binary tree in the first round (note we are not counting the initial pebbles placed before the game began). If $d \leq \lceil \log 4x \rceil$ (i.e., if the graph is shallow or if Pebbler placed too many pebbles), the claim holds since this implies $x \geq \frac{2^d}{8}$ and clearly $\frac{2^d}{8} \geq \frac{2^{d/\gamma + \log \gamma}}{8} = \frac{\gamma 2^{d/\gamma}}{8}$, for any γ and d that satisfy $2 \leq \gamma \leq d$. We thus assume $d > \lceil \log 4x \rceil$.

Note that the row that is at distance $\lceil \log 4x \rceil$ from the root of the top binary tree has exactly $y = 2^{\lceil \log 4x \rceil} \geq 4x$ vertices. Before the first round, at least $y/2$ of these vertices were partially good (with respect to the top binary tree). Since $x \leq y/4$ pebbles were placed, at least $y/4$ of these partially good trees were untouched at this round. We will show that, provided that there are less than $n/8$ pebbles in the graph, then at least one of these partially good trees is totally good.

Fix a set of $y/4$ partially good trees that were untouched at this round. If $d \leq \log n$, then the partially good trees are all totally good. If $d > \log n$, we consider the set S of all the (pebbled or unpebbled) vertices at distance $\log n$ from the root of the top binary tree that are in one of these $y/4$ partially good trees. Since these trees are disjoint there are at least $y/4 \cdot (n/y) = n/4$ such vertices. Consider the block consisting of vertices at distance at most $\log y$ from S. Note that the number of source-sinks paths in this block is at least $n/4 \cdot y$ and any vertex in this block is in at most y such paths. Therefore, if there are less than $n/8$ pebbles, then there are more than $ny/8$ unpebbled source-sink paths in this block. This means that at least one of the $y/4$ partially good tree has more than $n/2$ unpebbled source-sink paths in this block, which implies that it is a totally good tree.

Let v be the root of this totally good tree. We know that v reached the challenged vertex before this rounds. This implies that if v no longer reaches the challenged vertex then there are newly placed pebbles blocking a path between v and the challenged vertex. If this is the case, Challenger challenges a newly placed pebble that is in such a path and that is closest to v (i.e., is reachable from v). The graph induced by all vertices that reach v satisfies the invariants and has depth $d - \log y \geq d - \log 8x$. We observe that, the x pebbles we account for at this round were placed on the binary tree, and therefore are not counted again when applying the induction hypothesis.

In order to apply the induction hypothesis to the $(r-1)$-round DT* game on the subgraph induced by all vertices that reach v, we consider two cases, depending on whether the number of rounds left is greater or less than the depth of the remaining subgraph. If the number of rounds left is at most the depth of the remaining subgraph, that is, if $r - 1 \leq d - \log y$ (which implies $r \leq d$ and $\gamma = r$), then by the induction hypothesis we have that the cost of the $(r-1)$-round DT* game on the remaining subgraph is at least $\min\{\frac{(r-1)2^{(d-\log 8x)/(r-1)}}{8}, \frac{n}{8}\}$. Therefore, it suffices to show that

$$(r-1)2^{(d-\log 8x)/(r-1)} + 8x \geq r 2^{d/r} \quad .$$

Let $a = \frac{d}{r} - \frac{d-\log 8x}{r-1}$, so that $8x = 2^{d/r+a(r-1)}$. Rewriting the equation above we have

$$(r-1)2^{(d-\log 8x)/(r-1)} + 8x = (r-1)2^{d/r-a} + 2^{d/r+a(r-1)} = r2^{d/r}\left(\frac{(r-1)2^{-a}}{r} + \frac{2^{a(r-1)}}{r}\right) .$$

Note that, for any $r \geq 1$, $(r-1)2^{-a} + 2^{a(r-1)} \geq r$. Indeed, for any fixed $r \geq 1$, a straightforward calculation shows that the real function $f(a) = (r-1)2^{-a} + 2^{a(r-1)} - r$ is minimized at $a = 0$ and $f(0) = 0$.

We are left with the case in which the number of rounds remaining is larger than the depth of the subgraph, that is, $r - 1 > d - \log y$. We first argue that if $r < d/2$, then the claim follows even without applying the induction hypothesis. This is because the number of pebbles placed in this one round has to be large in order to reduce the depth of the remaining subgraph by so much. Indeed, if $r < d/2$ then $\log y > d/2 + 1$ and thus $8x \geq y > 2^{d/2+1}$. Moreover, $\gamma = \min\{r, d\} = r < d/2$ and since $d/\gamma + \log\gamma$ is a monotone decreasing function for $\gamma \in [2, d/2]$, we have that $8x > 2^{d/2+1} \geq 2^{d/\gamma+\log\gamma}$ and the claim follows.

We can therefore assume that $d/2 \leq r$. The intuition as to why the claim is true in this case is that what we need to prove is not so strong. Indeed, note that $d/2 \leq r$ implies that $d/2 \leq \gamma = \min\{r, d\} \leq d$, and thus $2^{d/\gamma+\log\gamma} \leq 2d$, so it is enough to show that at least $2d/8$ pebbles are needed. Applying the induction hypothesis on the subgraph induced by all vertices that reach v we have that the cost of this subgraph is at least $\min\{\frac{2(d-\log y)}{8}, \frac{n}{8}\}$. The claim follows by noting that $8x + 2(d - \log y) \geq 2d + 8x - 2\log 8x \geq 2d$, where the last inequality holds since $x \geq 1$. This concludes the proof. □

To conclude this section, we prove that Lemma 6.6 also applies to stacks of a more general class of previously studied graphs.

Definition 6.9 (Grate). An (α, β)-*grate* is a DAG such that after removing any α vertices, at least β pairs of a source and a sink are connected.

It is straightforward to verify that an $(\alpha, n^2 - (n/2 + 1)^2)$-grate is an α-uniform grate—in fact, this is what we noted in the proof of Observation 6.8—hence Lemma 6.6 holds for stacks of grates. In the converse direction, an α-uniform grate is an $(\alpha, (n/2 + 1)^2)$-grate. By Proposition 6.2 in [Val77], any $(\Omega(n), \Omega(n^{1+\epsilon}))$-grate of logarithmic depth needs to have superlinear size, so butterfly graphs are close to optimal.

Other than butterfly graphs, an example of $\Omega(n)$-uniform grates are supergrates [KS90]. Supergrates are of linear size, but they are too deep for the strategy of Lemma 6.3 to give meaningful upper bounds. Another example are connector graphs, as we proceed to show. Connector graphs can be shallow but require $\Omega(n \log n)$ edges [PV76].

Definition 6.10 (Connector). An n-*connector* is a DAG with n sources S and n sinks T, and that satisfies the following property: for any subsets $S' \subseteq S$ of sources and $T' \subseteq T$ of sinks of size $|S'| = |T'|$ and for any specification M of which source in S' should be connected to which sink in T' (one-to-one correspondence), it holds that there are $|S'|$ vertex-disjoint paths between S' and T' satisfying M.

Proposition 6.11. *An n-connector is an $(\alpha, n^2 - \alpha n)$-grate.*

Proof. We prove that an n-connector satisfies the following two properties:

1. any source can reach any sink;
2. the removal of any set of α vertices causes at most αn source-sink disconnections, i.e., the sum over all sinks v of the number of sources that cannot reach v is at most αn.

Let G be a n-connector. Obviously G satisfies property 1. Let A be any set of vertices in G. Let $\alpha = |A|$. We will show that the removal of A causes at most αn source-sink disconnections, thus concluding that G also satisfies property 2. Let G' be the graph that results from G after the removal of A.

Let $H = ((S, T), E)$ be a bipartite graph, where S correspond to the sources in G and T to the sinks, and there is an edge (s, t) if source s does not reach sink t in G'. Let q be the size of a maximum matching in H. This implies that H has a vertex cover of size q (Kőnig's theorem). Since every vertex in H has degree at most n, we conclude that H has at most qn edges, which means that A caused at most qn source-sink disconnections.

Suppose $q > \alpha$, and let $M = \{(s_1, t_1), (s_2, t_2), \ldots, (s_{\alpha+1}, t_{\alpha+1})\}$ be a matching of size $\alpha + 1$ in H. Given the set $S' = \{s_1, s_2, \ldots, s_{\alpha+1}\}$ of sources, the set $T' = \{t_1, t_2, \ldots, t_{\alpha+1}\}$ of sinks and M as the specification of which source should be connected to which sink, we have that in G there are $\alpha+1$ disjoint paths connecting S' to T' according to M. But this is a contradiction, since all paths must contain a vertex in A. Therefore, we conclude that $q \leq \alpha$ and A caused at most $qn \leq \alpha n$ source-sink disconnections. □

Interestingly, butterfly graphs and connectors also relate in that connecting two k-dimensional butterfly graphs in a certain back-to-back fashion gives a 2^k-connector. A description of this construction and a proof of this fact can be found, e.g., in [Nor20].

7 Upper Bounds for Size and Space

We build two separate refutations, one with small size in Lemma 7.3 of Section 7.1, and another with small space in Lemma 7.7 of Section 7.2. Both refutations are more convenient to describe in terms of the weaker *resolution* proof system. A resolution configuration $\mathbb{C}$ is a set of disjunctive clauses. A resolution refutation of a CNF formula F is a sequence of configurations $\mathbb{C}_0, \ldots, \mathbb{C}_\tau$ such that $\mathbb{C}_0 = \emptyset$, the empty clause $\bot \in \mathbb{C}_\tau$, and for $t \in [\tau]$ we obtain $\mathbb{C}_t$ from $\mathbb{C}_{t-1}$ by one of the following steps:

Axiom download $\mathbb{C}_t = \mathbb{C}_{t-1} \cup \{C\}$, for $C \in F$.

Inference $\mathbb{C}_t = \mathbb{C}_{t-1} \cup \{C \vee D\}$, where $C \vee D$ is inferred by the resolution rule $\dfrac{C \vee x \qquad D \vee \overline{x}}{C \vee D}$.

Erasure $\mathbb{C}_t = \mathbb{C}_{t-1} \setminus \{C\}$, for some $C \in \mathbb{C}_{t-1}$.

The length of a refutation is the number of axiom downloads and inferences. The line space of a configuration is the number of clauses, and the total space is the number of literals. The (line/total) space of a refutation is the maximum over all configurations.

It is easy to see that cutting planes can simulate the resolution rule using at most w additions and one division, where w is the width of the shortest clause being resolved, and therefore a resolution refutation of length L, width w and space s gives a cutting planes refutation of length $\mathrm{O}(wL)$, size $\mathrm{O}(w^2L)$ and space $s + 1$ where the largest coefficient is 2. We note that the refutation that we construct in Lemma 7.2 is of constant width, so cutting planes can simulate it with constant overhead, and that in Lemma 7.7 is not, but this only gives a polynomial factor overhead in length.

7.1 Upper Bound for Size

To build a refutation with small size we simulate a black pebbling in resolution and then lift that refutation, as done in for instance [BN11].

The *black pebble game* is played by a single player on a DAG. The allowed moves are to place a pebble on a vertex if its predecessors have pebbles and to remove a pebble from any vertex. A pebbling is a sequence of moves that begin with the empty graph and end with a pebble on the sink. The number of moves of a pebbling is called the time, and the maximum number of pebbles on the graph at the same time the space. An excellent survey of pebbling up to ca 1980 is [Pip80], and some more recent developments are covered in the upcoming survey [Nor20].

Lemma 7.1. *If there is a black pebbling for an indegree 2 graph G in space s and time τ, then there is a resolution refutation of Peb_G of length $\mathrm{O}(\tau)$, width 3, and total space $\mathrm{O}(s)$.*

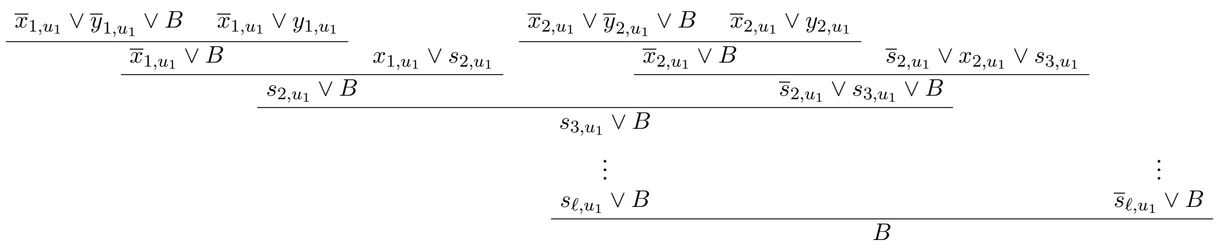

$\overline{x}_{1,u_1} \lor \overline{y}_{1,u_1} \lor B$
$\overline{x}_{1,u_1} \lor y_{1,u_1}$
$\overline{x}_{1,u_1} \lor B$
$x_{1,u_1} \lor s_{2,u_1}$
$s_{2,u_1} \lor B$
$\overline{x}_{2,u_1} \lor \overline{y}_{2,u_1} \lor B$
$\overline{x}_{2,u_1} \lor y_{2,u_1}$
$\overline{x}_{2,u_1} \lor B$
$\overline{s}_{2,u_1} \lor x_{2,u_1} \lor s_{3,u_1}$
$\overline{s}_{2,u_1} \lor s_{3,u_1} \lor B$
$s_{3,u_1} \lor B$
$s_{\ell,u_1} \lor B$
$\overline{s}_{\ell,u_1} \lor B$
B

Figure 11: Simulation of a pebbling step

Proof. We build a refutation π of Peb_G by keeping in memory the unit clause v for every vertex v that has a pebble. This is trivial for sources because these clauses are already axioms. For a vertex v with predecessors u_1 and u_2, when we place a pebble over v its predecessors have pebbles, therefore the clauses u_1 and u_2 are in memory. We download the axiom $\overline{u_1} \lor \overline{u_2} \lor v$, resolve it with u_1 and u_2 to obtain the clause v, and then delete intermediate clauses. $\square$

We can use a generic procedure to transform any refutation into a refutation for the corresponding lifted formula (see Lemma 4.3 in the ECCC version of [BN11]). However, we obtain better upper bounds if we take the structure of the refutation into account.

Lemma 7.2. *Let G be a graph of indegree 2 with a black pebbling in space s and time τ. Then there is a cutting planes refutation of $Lift_\ell(Peb_G)$ in size $O(\tau \cdot \ell^3)$ and total space $O(s \cdot \ell)$.*

Proof. Let π be the resolution refutation of Peb_G given by Lemma 7.1. We build a resolution refutation π' of $Lift(F)$ by deriving, for each unit clause v, the ℓ clauses $Lift(v)$. This is trivial in the axiom download and erasure cases, and we are left with inference. The only inference steps we need to deal with are of the form

$$\dfrac{\dfrac{\overline{u_1} \lor \overline{u_2} \lor v \qquad u_1}{\overline{u_2} \lor v} \qquad u_2}{v} \tag{7.1}$$

and we handle both inference steps at once.

Recall that for a lifted formula to have constant width we have to split the wide auxiliary clauses (2.4a), introducing extension variables, but we were not explicit about how to do that. We split the clause $\bigvee_{a=1}^{\ell} x_{a,u}$ into a clause $x_{1,u} \lor s_{2,u}$, $\ell-2$ clauses of the form $\overline{s}_{a,u} \lor x_{a+1,u} \lor s_{a,u}$, and a clause $\overline{s}_{\ell,u} \lor x_{\ell,u}$.

First we fix a clause $C \in Lift(v)$ that we want to derive. Then we fix a clause $B \in Lift(\overline{u_2} \lor v)$ that contains C as a subclause. We can derive B by resolving the clauses $\overline{x}_{a,u_1} \lor \overline{y}_{a,u_1} \lor B$, which are actual axioms of $Lift(Peb_G)$, first with $\overline{x}_{a,u_1} \lor y_{a,u_1}$, which are in memory by hypothesis, and then with the axioms $\overline{s}_{a,u_1} \lor x_{a,u_1} \lor s_{a+1,u_1}$ that result of breaking $\bigvee_{a=1}^{\ell} x_{a,u_1}$ into clauses of constant width. See Figure 11 for details. Such a derivation requires $O(\ell)$ steps and constant space.

We repeat this procedure for all of the ℓ clauses in $B \in Lift(\overline{u_2} \lor v)$ that contain C as a subclause, using at most $O(\ell^2)$ steps and space $\ell + O(1)$. Now we have all the clauses required to derive C by repeating the above procedure with the clauses $Lift(\overline{u_2}) \lor C$ that we just derived, the clauses $\overline{x}_{a,u_1} \lor y_{a,u_1}$, which are also in memory by hypothesis, and the axioms $\overline{s}_{a,u_1} \lor x_{a,u_1} \lor s_{a+1,u_1}$. Such a derivation requires an additional $O(\ell)$ steps and constant additional space, for a total of $O(\ell^2)$ steps and space $\ell + O(1)$. Finally we repeat the whole procedure ℓ times, once for each clause $C \in Lift(v)$, for a total of $O(\ell^3)$ steps and space $2\ell + O(1)$.

Observe that all clauses are of constant width, so the size and total space are also, respectively, $O(\ell^3)$ and $O(\ell)$, and furthermore we can simulate the resolution proof in cutting planes with constant overhead. $\square$

If we only care about optimizing size, then a strategy that places pebbles in topological order and never removes a pebble is a valid pebbling of any graph of order m in time m and space m, which by

Lemma 7.2 gives a short refutation of the lifted pebbling formula in size $\mathrm{O}(m\ell^3)$ and space $\mathrm{O}(m\ell)$, as stated next.

Lemma 7.3. *Let G be a graph of order m and indegree 2. For any $\ell \geq m^3$ there is a cutting planes refutation of $\mathit{Lift}_\ell(\mathit{Peb}_G)$ in size $\mathrm{O}(N)$ and total space $\mathrm{O}(N^{2/5})$, where $N = \Theta(m\ell^3)$ is the size of $\mathit{Lift}_\ell(\mathit{Peb}_G)$.*

7.2 Upper Bound for Space

In terms of space, even the most space-efficient pebbling strategy would give a refutation in space $\mathrm{O}(\ell)$, which is too weak. Therefore to obtain good space upper bounds we go through the Dymond–Tompa game and search depth instead of black pebbling.

The search depth of a formula F is the minimum number of queries of a decision tree for the search problem of F. As observed in [LNNW95, BIW04], a search tree for the falsified clause search problem is equivalent to a tree-like resolution refutation. We can construct a refutation essentially by replacing each internal node labelled with a variable x in the search tree with the result of resolving its two children over the variable x. It is straightforward to check that this is indeed a valid resolution refutation.

Lemma 7.4 ([ET01]). *If a CNF formula has search depth d, then it has a resolution refutation of length 2^d, width d, and space d simultaneously.*

Proof. Consider the refutation tree T equivalent to a minimal depth search tree. Traversing the refutation tree in depth-first order it is straightforward to reconstruct a tree-like refutation of length $|V(T)| \leq 2^d$, width d, and space d. □

The following lemma follows from Lemma 5.1 and was first proved in [Cha13].

Lemma 7.5 ([Cha13]). *If there is a Dymond–Tompa pebbling strategy for a graph G in space s, then the formula Peb_G has search depth s.*

If we lay out the extension variables so that their indices form an ordered binary tree and attach two nodes labelled $x_{a,u}$ and $x_{a+1,u}$ to each leaf $s_{a,u}$ we get a search tree that finds a selector variable set to true by any assignment that respects auxiliary clauses. We can use this tree to build search trees for a lifted formula.

Lemma 7.6. *Given a CNF formula F of search depth d, the lifted formula $\mathit{Lift}_\ell(F)$ has search depth $d \log \ell$.*

Proof. Given a decision tree T_1 for the falsified clause search problem on F of depth d and a decision tree T_2 that finds a selector variable set to true of depth $\log \ell$, we build a decision tree T_3 for the falsified clause search problem on $\mathit{Lift}_\ell(F)$ of depth $d \log \ell$ by composing the trees as follows.

First we modify T_2. We reinterpret the leaves as queries to selector variables $x_{a,u}$, and we attach two new nodes to every selector variable query. We label the 0-leaf of $x_{a,u}$ with the falsified clause $\overline{s}_{a,u} \vee x_{a,u} \vee s_{a+1,u}$, and we label the 1-node with the main variable $y_{a,u}$. We add two unlabelled leaves to the $y_{a,u}$ node: a 0-leaf and a 1-leaf.

Then, starting at the root of T_1, we apply the following recursive procedure. If the root is an inner vertex labelled with a variable u, then we add a copy of T_2 that queries variables corresponding to u. To each unlabelled 0-leaf we attach the result of this procedure on the 0-subtree of T_1, and to each unlabelled 1-leaf we attach the result of this procedure on the 1-subtree.

Finally, for each leaf of T_3 that we did not label yet, there is a corresponding leaf in T_1 labelled with a clause C. C is falsified by the assignment α induced by the branch leading to C. By construction, the assignment β induced by the branch in T_3 respects auxiliary clauses and, for every variable $u \in \mathit{Vars}(C)$ it sets $x_{a,u} = 1$ and $y_{a,u} = \alpha(u)$ for some $a \in [\ell]$. Therefore we can label the leaf of T_3 with the main clause $\bigvee_{u \in \mathit{Vars}(C)} \overline{x}_{a,u} \vee y_{a,u}^{1-\alpha(u)}$. □

The space upper bound follows immediately from Lemmas 7.5, 7.6, and 7.4.

Lemma 7.7. *Let G be a graph of order m and indegree 2 with Dymond–Tompa price s. For any $\ell \geq m^3$ there is a cutting planes refutation of $Lift_\ell(Peb_G)$ of size $2^{\mathrm{O}(s \log N)}$ and space $\mathrm{O}(s \log N)$, where $N = \Theta(m\ell^3)$ is the size of $Lift_\ell(Peb_G)$.*

8 Putting the Pieces Together

By the technical result proved in Section 2, Theorem 2.8, we know that if G is a graph over m vertices such that the r-round Dymond–Tompa game on G costs $\Omega(c)$, then for $\ell = m^{3+\epsilon}$, $Lift_\ell(Peb_G)$ is a 6-CNF formula over $\Theta(m^{4+\epsilon})$ variables and $N = \Theta(m^{10+3\epsilon})$ clauses such that for any CP refutation of $Lift_\ell(Peb_G)$ even with coefficients of unbounded size in formula space less than $\frac{c}{r} \log N$ requires length greater than $2^{\Omega(r)}$. This fact together with the lower and upper bounds proven in Sections 6 and 7 yield the following theorem.

Theorem 8.1. *There is an explicitly constructible two-parameter family of unsatisfiable 6-CNF formulas $F(n, d)$, for $n, d \in \mathbb{N}^+$, of size $N = \Theta((dn)^{10+\epsilon})$ such that:*

1. *$F(n, d)$ can be refuted by CP with small coefficients in size $\mathrm{O}(N)$ and total space $\mathrm{O}(N^{2/5})$.*
2. *$F(n, d)$ can be refuted by CP with small coefficients in total space $\mathrm{O}(d \log N)$ and size $2^{\mathrm{O}(d \log N)}$.*
3. *For any $r \leq d$, any CP refutation even with coefficients of unbounded size of $F(n, d)$ in formula space less than $\min\{\frac{2^{d/r} \log N}{8}, \frac{n \log N}{8r}\}$ requires length greater than $2^{\Omega(r)}$.*

Proof. Let G be a stack of depth d of butterfly graphs of dimension $\log n$ which has a total of $\Theta(dn)$ vertices. Let $F(n, d) = Lift_\ell(Peb_G)$.

Item 1 follows directly from Lemma 7.3. Item 2 follows from combining Corollary 6.4 with Lemma 7.7.

By Lemma 6.1 we get that for any $r \leq d$, the r-round Dymond–Tompa game played on G has cost at least at least $\min\{\frac{r2^{d/r}}{8}, \frac{n}{8}\}$. Thus, by Theorem 2.8, we get item 3. ☐

Choosing the right values for d and r in Theorem 8.1, we get the following to corollaries. These are generalizations of Theorems 1.1 and 1.2.

Corollary 8.2. *For any positive constant K, there exists a family of 6-CNF formulas $\{F_N\}_{N=1}^{\infty}$ of size $\Theta(N)$ such that:*

1. *F_N can be refuted by CP with small coefficients in size $\mathrm{O}(N)$ and total space $\mathrm{O}(N^{2/5})$.*
2. *F_N can be refuted by CP with small coefficients in total space $\mathrm{O}(\log^{K+2} N)$ and size $2^{\mathrm{O}(\log^{K+2} N)}$.*
3. *Any CP refutation even with coefficients of unbounded size of F_N in formula space less than $N^{1/10-\epsilon}$ requires length greater than $2^{\Omega(\log^K N)}$, for any constant $\epsilon > 0$.*

Proof. The proof follows from setting $d = \log^{K+1} n$ and $r = d/\log n$ in Theorem 8.1 and for every N, choosing n to be a power of 2 such that N is at most a factor off from $(nd)^{10+\epsilon}$.

We note that $\log N = \Theta(\log n)$, so 2 holds. Moreover, $N = \mathrm{o}\big((nd)^{10+2\epsilon}\big)$, thus $N^{1/10-\epsilon} = \mathrm{o}\big((nd)^{1-9\epsilon}\big)$ and

$$(nd)^{1-9\epsilon} \leq \frac{nd}{8 \log^{2K} n} \leq \frac{n}{8r} \log N,$$

and therefore 3 also holds. ☐

Corollary 8.3. *For any positive constant K, there exists a family of 6-CNF formulas $\{F_N\}_{N=1}^{\infty}$ of size $\Theta(N)$ such that:*

1. *F_N can be refuted by CP with small coefficients in size $\mathrm{O}(N)$ and total space $\mathrm{O}(N^{2/5})$.*
2. *F_N can be refuted by CP with small coefficients in total space $\mathrm{O}\big(N^{\frac{1}{10(K+1)}}\big)$ and size $2^{\mathrm{O}\left(N^{\frac{1}{10(K+1)}}\right)}$.*
3. *Any CP refutation even with coefficients of unbounded size of F_N in formula space less than $N^{\frac{K-1}{10(K+1)}-\epsilon}$ requires length greater than $2^{\Omega\left(N^{\frac{1}{10(K+1)}}\right)}$, for any constant $\epsilon > 0$.*

Proof. The proof follows from setting $d = n^{1/K}\log n$ and $r = d/\log n$ in Theorem 8.1 and for every N, choosing n to be a power of 2 such that N is at most a factor off from $(nd)^{10+\epsilon}$.

We note that $N^{\frac{1}{10(K+1)}} = \Theta\big((nd)^{\frac{10+\epsilon}{10(K+1)}}\big)$ and $(nd)^{\frac{10+\epsilon}{10(K+1)}} > d\log N$, hence 2 holds. Moreover, $N = \mathrm{o}\big((nd)^{10+2\epsilon}\big)$, thus $N^{\frac{K-1}{10(K+1)}-\epsilon} = \mathrm{o}\big((nd)^{\frac{K-1}{K+1}-9\epsilon}\big)$ and

$$(nd)^{\frac{K-1}{K+1}-9\epsilon} = n^{\frac{K-1}{K}} \cdot (\log n)^{\frac{K-1}{K+1}} \cdot (nd)^{-9\epsilon} < \frac{n^{\frac{K-1}{K}}}{8}\log N = \frac{n}{8r}\log N,$$

and, therefore 3 also holds. □

9 Exponential Separation of the Monotone AC Hierarchy

Unsurprisingly, we follow the same approach as [RM99]. Our function is a restriction of the GEN function, except that instead of restricting the valid instances to pyramid graphs, which are unconditionally hard, we restrict the valid instances to the graphs from Section 6 that exhibit round-space trade-offs. We then use our round-aware simulation theorem to lift the trade-off to communication complexity and the Karchmer–Wigderson game to translate it to a trade-off for monotone circuits.

Definition 9.1. The *Karchmer–Wigderson game* [KW90] is the following communication problem: given a monotone function f, Alice gets an input x such that $f(x) = 1$ and Bob gets an input y such that $f(y) = 0$. Their task is to find a coordinate i such that $x_i = 1$ and $y_i = 0$

Theorem 9.2 ([KW90]). *If there is a monotone circuit for f of fan-in 2^c and depth r, then there is a protocol for the Karchmer–Wigderson game with cost rc and r rounds.*

Proof. The proof is a simple induction on the depth of the circuit. If the circuit has no gates, the players just return the index of the output variable. Otherwise, if the output gate is an OR-gate, i.e., $f = \bigvee g_i$, then it must be the case that $g_i(y) = 0$ for all i and that there exists an index i such that $g_i(x) = 1$. Alice sends such an index i to Bob, which costs at most c, and we apply the induction hypothesis on the function g_i and a circuit of smaller depth. If the output gate is an AND-gate, Bob acts analogously. □

Informally, the G-GEN function computes whether a subset of the lifted pebbling formula on a graph G is unsatisfiable given the indicator vector of such subset. It can be further generalized to CSP-SAT as in [GP18], but we give a definition specialized to pebbling formulas that already suggests a circuit to compute it.

Definition 9.3. Given a graph G of indegree 2 and $\ell \in \mathbb{N}$, the G-GEN Boolean function is defined as follows. There is a variable (v, a) for every source $v \in V(G)$ and index $a \in [\ell]$. There is a variable $(v \vee \overline{u}_1 \vee \overline{u}_2, a, b, c)$ for every non-source vertex $v \in V(G)$ with predecessors u_1 and u_2 and triple $(a, b, c) \in [\ell]^3$. There is a variable $(\overline{t}, a)$ for every index $a \in [\ell]$. A pair (v, a) is reachable if v is a source and (v, a) is 1, or, if v has predecessors u_1 and u_2, if there exist $(b, c) \in [\ell]^2$ such that $(v \vee \overline{u}_1 \vee \overline{u}_2, a, b, c)$ is 1, (u_1, b) is reachable, and (u_2, c) is reachable. The value of G-GEN is 1 if there exists some index $a \in [\ell]$ such that (t, a) is reachable and $(\overline{t}, a)$ is 1.

Lemma 9.4. *There is a monotone circuit that computes G-GEN in depth $2d$, fan-in ℓ^2, and size $\mathrm{O}(m\ell^3)$, where d is the depth of G.*

Proof. The circuit computes, for every $v \in V(G)$ and every $a \in [\ell]$, whether the pairs (v, a) is reachable. This is done in topological order. For a source v, the pair (v, a) is a variable. For a non-source v with predecessors u_1 and u_2, the circuit has for each pair $(b, c) \in [\ell]^2$ an AND-gate of fan-in 3 that receives as inputs the gate that computes (u_1, b), the gate that computes (u_2, c), and the variable $(v \vee \overline{u}_1 \vee \overline{u}_2, a, b, c)$. These ℓ^2 gates are the inputs of an OR-gate that indeed computes whether (v, a) is reachable.

Finally, the output gate of the circuit is an OR-gate of fan-in ℓ that receives as input, for every $a \in [\ell]$, an AND-gate of fan-in 2 with inputs the gate that computes (t, a) and the variable $(\overline{t}, a)$. □

It remains to give a reduction from the Karchmer–Wigderson game on G-GEN to the communication game for which we proved lower bounds, and this follows straightforwardly from the corresponding reduction in [RM99].

Lemma 9.5. *If there is a deterministic communication protocol for the Karchmer–Wigderson game on* G-GEN *with cost c and r rounds, then there is a deterministic communication protocol for* $\mathit{Lift}(\mathit{Search}(\mathit{Peb}_G))$ *with cost c and r rounds.*

Proof. Let (x, y) be an input to $\mathit{Lift}(\mathit{Search}(\mathit{Peb}_G))$, that is, x is a vector of indices and y is a vector of binary strings such that $\mathsf{Ind}(x_v, y_v)$ is an assignment to a variable v of Peb_G.

Alice builds her input for the Karchmer–Wigderson game on G-GEN as follows. For every source $v \in V(G)$, Alice sets the variable (v, x_v) to 1, and for every non-source vertex $v \in V(G)$ with predecessors u_1 and u_2, Alice sets the variable $(v \vee \overline{u}_1 \vee \overline{u}_2, x_v, x_{u_1}, x_{u_2})$ to 1. Finally, Alice sets $(\overline{t}, x_t)$ to 1 and all the remaining variables to 0. Note that the value of G-GEN on this input is 1.

Bob builds his input as follows. For every source $v \in V(G)$, Bob sets the variable (v, a) to $\mathsf{Ind}(a, y_v)$, and for every non-source vertex $v \in V(G)$, Bob sets the variable $(v \vee \overline{u}_1 \vee \overline{u}_2, a, b, c)$ to 0 if $\mathsf{Ind}(a, y_v) = 0$, $\mathsf{Ind}(b, y_{u_1}) = 1$, and $\mathsf{Ind}(c, y_{u_2}) = 1$, and to 1 otherwise. Bob sets $(\overline{t}, a)$ to $1 - \mathsf{Ind}(a, y_t)$. Observe that, in Bob's input, if a pair (v, a) is reachable, then $\mathsf{Ind}(a, y_v) = 1$. For the sink, this means that if (t, a) is reachable then $(\overline{t}, a) = 1 - \mathsf{Ind}(a, y_t) = 0$ and therefore the value of G-GEN on Bob's input is 0.

Both players then simulate the protocol for the Karchmer–Wigderson game and find a variable that is set to 1 on Alice's input and to 0 on Bob's. We analyse the different possibilities for such a variable and show that in all cases they identify a falsified clause of Peb_G. If it is a variable (v, x_v) (for some source v), then since Bob sets this variable to 0 it must be that $\mathsf{Ind}(x_v, y_v) = 0$, and therefore the unit clause v is falsified. If it is a variable $(v \vee \overline{u}_1 \vee \overline{u}_2, x_v, x_{u_1}, x_{u_2})$, then $\mathsf{Ind}(x_v, y_v) = 0$, $\mathsf{Ind}(x_{u_1}, y_{u_1}) = 1$, and $\mathsf{Ind}(x_{u_2}, y_{u_2}) = 1$, and thus the axiom $v \vee \overline{u}_1 \vee \overline{u}_2$ is falsified. If it is $(\overline{t}, x_v)$, then $\mathsf{Ind}(x_v, y_v) = 1$, so axiom $\overline{t}$ is falsified. □

We have all the ingredients to prove the two main theorems of this section.

Theorem 1.3 (Restated). *For every $i \in \mathbb{N}$ there is a Boolean function over n variables that can be computed by a monotone circuit of depth* $\log^i n$*, fan-in 2, and size* $\mathrm{O}(n)$*, but for which every monotone circuit of depth* $\mathrm{o}(\log^i n/(\log\log n)^2)$ *requires superpolynomial size.*

Proof. Let $G = G(\log^{2i} n, \log^i n/40i \log\log n)$ be the graph given by Lemma 6.1, that is a stack of butterflies with $w = \log^{2i} n$ sources and depth $d = \log^i n/40i \log\log n$, hence size $m < \log^{3i} n$. Consider the Boolean function G-GEN for $\ell = m^{3+\epsilon} < \log^{10i} n$.

By Lemma 9.4 there is a monotone circuit of depth $2d$, fan-in ℓ^2, and size $\mathrm{O}(m\ell^3)$ that computes G-GEN, which we can expand into a circuit of depth $4d \log \ell < \log^i n$, fan-in 2, and size $\mathrm{O}(m\ell^4) = \mathrm{O}(\log^{43i} n)$.

The lower bound follows by combining the links in the chain of the reductions. Let C be a circuit of fan-in 2^c and depth $r \leq d/\log\log n = \mathrm{O}(\log^i n/(\log\log n)^2)$ that computes G-GEN. By Theorem 9.2 there is a protocol for the Karchmer–Wigderson game with cost rc and r rounds. This implies, by Lemma 9.5, that there is a deterministic communication protocol for $\mathit{Lift}(\mathit{Search}(\mathit{Peb}_G))$ with cost rc and r rounds. By Theorem 4.1 (the simulation theorem), there is a parallel decision tree solving $\mathit{Search}(\mathit{Peb}_G)$ using $\mathrm{O}(rc/\log \ell)$ queries and depth r and by Lemma 2.6 Pebbler has a winning strategy in the r-round Dymond–Tompa game on G in cost at most $\mathrm{O}(rc/\log \ell)$. Finally, by Lemma 6.1, we have that $rc/\log \ell = \Omega\big(\min\{r2^{d/r}, w\}\big) \geq \Omega\big(r \log n\big)$ and therefore the circuit C requires size at least

$$2^c = 2^{\Omega(\log n \cdot \log \ell)} = n^{\Omega(\log \log n)} , \tag{9.1}$$

which is superpolynomial. □

Note that the function G-GEN above only depends on a polylogarithmic number of variables, with the remaining being padding, so that we can use a small enough gadget to keep the function in NC^i. This implies that we can only obtain a superpolynomial lower bound. We show next that if we are willing to forfeit the function belonging in NC^i then we can achieve an exponential lower bound.

Theorem 1.4 (Restated). *For every $i \in \mathbb{N}$ there is a Boolean function over n variables that can be computed by a monotone circuit of depth $\log^i n$, fan-in $n^{4/5}$, and size $\mathrm{O}(n)$, but for which every monotone circuit of depth $q \log^{i-1} n$ requires size $2^{\Omega(n^{1/(10+4\epsilon)q})}$.*

Proof. Let $G = G(n^{1/(10+3\epsilon)}/(\log^i n), \log^i n/(10+3\epsilon))$ be the graph given by Lemma 6.1, that is a stack of butterflies with $w = n^{1/(10+3\epsilon)}/(\log^i n)$ sources and depth $d = \log^i n/(10+3\epsilon)$, hence size $m \leq n^{1/(10+3\epsilon)}$. Consider the function G-GEN for $\ell = m^{3+\epsilon} \leq n^{(3+\epsilon)/(10+3\epsilon)}$. Note that the number of variables of G-GEN is indeed at most $m\ell^3 \leq n$.

By Lemma 9.4 there is a monotone circuit of depth $2d \leq \log^i n$, fan-in $\ell^2 \leq n^{4/5}$, and size $\mathrm{O}(n)$ that computes G-GEN. As for the lower bound, by the exact same chain of reductions described in the proof of Theorem 1.3 above, that is, by combining Theorem 9.2, Lemma 9.5, Theorem 4.1, Lemma 2.6, and Lemma 6.1, we conclude that if C is a circuit of fan-in 2^c and depth $r = q \log^{i-1} n$ that computes G-GEN, then $rc/\log \ell = \Omega\big(\min\{r2^{d/r}, w\}\big) = \Omega\big(\min\{rn^{1/(10+3\epsilon)q}, n^{1/(10+3\epsilon)}/\log^i n\}\big) \geq \Omega\big(rn^{1/(10+4\epsilon)q}\big)$, where the inequality holds for n large enough. Therefore C requires size at least

$$2^c = 2^{\Omega\left(n^{1/(10+4\epsilon)q} \log \ell\right)} \geq 2^{\Omega\left(n^{1/(10+4\epsilon)q}\right)} , \tag{9.2}$$

as we wanted to show. □

A simulation theorem with a smaller gadget would allow us to obtain a stronger separation between monotone-AC^{i-1} and monotone-NC^i, as we remark in the next section.

10 Concluding Remarks

In this paper we report the first true size-space trade-offs for cutting planes, exhibiting CNF formulas which have small-size and small-space proofs with constant-size coefficients but for which any short proofs must use a lot of memory, even when using exponentially large coefficients and even when we measure just the number of lines (i.e., inequalities) rather than total size. Furthermore, these results also hold for resolution and polynomial calculus, and are thus the first trade-offs to uniformly capture the proof systems underlying the currently best SAT solvers.

The main technical component in our proof is a reduction to communication complexity as in [HN12, GP18], but with the crucial difference that we reduce to round-efficient protocols in the real communication model of [Kra98]. Extending the techniques in [RM99, GPW18, BEGJ00] to this more general setting, and combining them with new trade-off results for Dymond–Tompa pebbling [DT85], yields our results. Using the same approach we are also able to obtain a separation between monotone-AC^{i-1} and monotone-NC^i, and an exponential separation between monotone-AC^{i-1} and monotone-AC^i, improving on the superpolynomial separation in [RM99].

An interesting challenge would be to extend our reduction to stronger communication models such as two-party randomized or multi-party real communication, which would yield trade-offs for stronger proof systems. A recent result in this direction is [GLM+16], but unfortunately it seems hard to incorporate round-efficiency in this framework.

Another question concerns the size of the lifting gadget we need to construct formulas exhibiting trade-offs. Our gadget has a CNF representation of more than cubic size, which incurs a substantial loss in the results. The (CNF) gadget size has been subsequently improved to almost quadratic [CKLM19] and almost linear [LMZ20, LMM$^+$20], but it would be nice to construct constant-size gadgets, which could lead to tighter trade-off results.

A further consequence of having a constant-size gadget is that we would obtain an exponential separation between $\mathsf{monotone\text{-}AC}^{i-1}$ and $\mathsf{monotone\text{-}NC}^{i}$, combining the best parts of our two results. Moreover, we could also construct a function witnessing a superpolynomial separation between $\mathsf{monotone\text{-}AC}^{i-1}$ and $\mathsf{monotone\text{-}NC}^{i}$ that is also computable by superpolynomial $\mathsf{monotone\text{-}NC}^{i-1}$ circuits.

Many proof complexity trade-offs have been obtained by reducing to the *black-white pebble game* [CS76], but in this paper we use the Dymond–Tompa game. It would be desirable to obtain a better understanding of the role of these games and what kind of trade-offs can be obtained from them.

Finally, from a proof complexity perspective we have very few examples of formula families that exhibit size-space trade-offs. Apart from the pebbling formulas studied in this work, the only natural examples[4] are the Tseitin contradictions over long, narrow grids in [BBI16, BNT13]. It would be interesting to prove size-space trade-offs for the latter formulas also in cutting planes, or to find other formulas with size-space trade-offs for this or other proof systems.

Acknowledgements

The authors wish to acknowledge Mladen Mikša, who participated in the initial stages of this work and has kept contributing helpful remarks throughout the project, Massimo Lauria, with whom we have had many fruitful discussions on time-space trade-offs and other topics in proof complexity, and Arkadev Chattopadhyay for suggesting us to look into the $\mathsf{monotone\text{-}AC}^{i-1}$ vs $\mathsf{monotone\text{-}NC}^{i}$ problem as well as general discussions on communication and circuit complexity. We want to thank Siu Man Chan for introducing us to the wonderful world of Dymond–Tompa pebbling. Different subsets of the authors are grateful for detailed and very helpful discussions on communication complexity with Joshua Brody, Prahladh Harsha, Johan Håstad, Troy Lee, Jaikumar Radakrishnan, and Anup Rao. Finally, we are thankful to Dieter van Melkebeek and Ryan Williams for help with references for general SAT time-space trade-offs.

The authors performed this work while at KTH Royal Institute of Technology and were funded by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013) / ERC grant agreement no. 279611 as well as by the Swedish Research Council grant 621-2012-5645. The first author has also received support from the Knut and Alice Wallenberg grant KAW 2018.0371, the second author from the Swedish Research Council grant 2016-00782 and from the Independent Research Fund Denmark grant 9040-00389B, and the third author from the European Union's Horizon 2020 research and innovation programme under grant agreement № 802020.

References

[ABRW02] Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Space complexity in propositional calculus. *SIAM Journal on Computing*, 31(4):1184–1211, April 2002. Preliminary version in *STOC '00*.

[And85] A. E. Andreev. On a method for obtaining lower bounds for the complexity of individual monotone functions. *Soviet Mathematics Doklady*, 31(3):530–534, 1985. English translation of a paper in *Doklady Akademii Nauk SSSR*.

[And87] Ian Anderson. *Combinatorics of Finite Sets*. Oxford University Press, 1987.

[4] Ignoring trade-offs obtained in [Nor09] by gluing together disjoint copies of unrelated formulas.

[BBI16] Paul Beame, Chris Beck, and Russell Impagliazzo. Time-space tradeoffs in resolution: Superpolynomial lower bounds for superlinear space. *SIAM Journal on Computing*, 45(4):1612–1645, August 2016. Preliminary version in *STOC '12*.

[BEGJ00] María 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. Preliminary version in *FOCS '98*.

[Ben09] Eli Ben-Sasson. Size-space tradeoffs for resolution. *SIAM Journal on Computing*, 38(6):2511–2525, May 2009. Preliminary version in *STOC '02*.

[BHP10] Paul Beame, Trinh Huynh, and Toniann Pitassi. Hardness amplification in proof complexity. In *Proceedings of the 42nd Annual ACM Symposium on Theory of Computing (STOC '10)*, pages 87–96, June 2010.

[BHvMW09] Armin Biere, Marijn J. H. Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, February 2009.

[BIW04] Eli Ben-Sasson, Russell Impagliazzo, and Avi Wigderson. Near optimal separation of tree-like and general resolution. *Combinatorica*, 24(4):585–603, September 2004.

[Bla37] Archie Blake. *Canonical Expressions in Boolean Algebra*. PhD thesis, University of Chicago, 1937.

[BN08] Eli Ben-Sasson and Jakob Nordström. Short proofs may be spacious: An optimal separation of space and length in resolution. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS '08)*, pages 709–718, October 2008.

[BN11] Eli Ben-Sasson and Jakob Nordström. Understanding space in proof complexity: Separations and trade-offs via substitutions. In *Proceedings of the 2nd Symposium on Innovations in Computer Science (ICS '11)*, pages 401–416, January 2011.

[BNT13] Chris Beck, Jakob Nordström, and Bangsheng Tang. Some trade-off results for polynomial calculus. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC '13)*, pages 813–822, May 2013.

[BS90] Ravi B. Boppana and Michael Sipser. The complexity of finite functions. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science. Volume A: Algorithms and Complexity*, pages 757–804. MIT Press, 1990.

[BS97] Roberto J. Bayardo Jr. and Robert Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI '97)*, pages 203–208, July 1997.

[BW01] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow—resolution made simple. *Journal of the ACM*, 48(2):149–169, March 2001. Preliminary version in *STOC '99*.

[CCT87] William Cook, Collette Rene Coullard, and György Turán. On the complexity of cutting-plane proofs. *Discrete Applied Mathematics*, 18(1):25–38, November 1987.

[CEI96] Matthew Clegg, Jeffery Edmonds, and Russell Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC '96)*, pages 174–183, May 1996.

[Cha13] Siu Man Chan. Just a pebble game. In *Proceedings of the 28th Annual IEEE Conference on Computational Complexity (CCC '13)*, pages 133–143, June 2013.

[Chv73] Vašek Chvátal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, 4(1):305–337, 1973.

[CKLM19] Arkadev Chattopadhyay, Michal Koucký, Bruno Loff, and Sagnik Mukhopadhyay. Simulation theorems via pseudo-random properties. *Computational Complexity*, 28(4):617–659, 2019.

[CLNV15] Siu Man Chan, Massimo Lauria, Jakob Nordström, and Marc Vinyals. Hardness of approximation in PSPACE and separation results for pebble games (Extended abstract). In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS '15)*, pages 466–485, October 2015.

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC '71)*, pages 151–158, May 1971.

[CR79] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44(1):36–50, March 1979. Preliminary version in *STOC '74*.

[CS76] Stephen A. Cook and Ravi Sethi. Storage requirements for deterministic polynomial time recognizable languages. *Journal of Computer and System Sciences*, 13(1):25–37, 1976. Preliminary version in *STOC '74*.

[dRNV16] Susanna F. 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 Annual IEEE Symposium on Foundations of Computer Science (FOCS '16)*, pages 295–304, October 2016.

[DT85] Patrick W. Dymond and Martin Tompa. Speedups of deterministic machines by synchronous parallel machines. *Journal of Computer and System Sciences*, 30(2):149–161, April 1985. Preliminary version in *STOC '83*.

[DvMW11] Scott Diehl, Dieter van Melkebeek, and Ryan Williams. An improved time-space lower bound for tautologies. *Journal of Combinatorial Optimization*, 22(3):325–338, October 2011. Preliminary version in *COCOON '09*.

[ET01] Juan Luis Esteban and Jacobo Torán. Space bounds for resolution. *Information and Computation*, 171(1):84–97, 2001. Preliminary versions of these results appeared in *STACS '99* and *CSL '99*.

[For00] Lance Fortnow. Time-space tradeoffs for satisfiability. *Journal of Computer and System Sciences*, 60(2):337–353, April 2000. Preliminary version in *CCC '97*.

[Fra84] Péter Frankl. A new short proof for the Kruskal–Katona theorem. *Discrete Mathematics*, 48(2):327–329, February 1984.

[GLM$^+$16] Mika Göös, Shachar Lovett, Raghu Meka, Thomas Watson, and David Zuckerman. Rectangles are nonnegative juntas. *SIAM Journal on Computing*, 45(5):1835–1869, October 2016. Preliminary version in *STOC '15*.

[Gom63] Ralph E. Gomory. An algorithm for integer solutions of linear programs. In R.L. Graves and P. Wolfe, editors, *Recent Advances in Mathematical Programming*, pages 269–302. McGraw-Hill, New York, 1963.

[GP18] Mika Göös and Toniann Pitassi. Communication lower bounds via critical block sensitivity. *SIAM Journal on Computing*, 47(5):1778–1806, October 2018. Preliminary version in *STOC '14*.

[GPT15] Nicola Galesi, Pavel Pudlák, and Neil Thapen. The space complexity of cutting planes refutations. In *Proceedings of the 30th Annual Computational Complexity Conference (CCC '15)*, volume 33 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 433–447, June 2015.

[GPW18] Mika Göös, Toniann Pitassi, and Thomas Watson. Deterministic communication vs. partition number. *SIAM Journal on Computing*, 47(6):2435–2450, 2018. Preliminary version in *FOCS '15*.

[HN12] Trinh Huynh and Jakob Nordström. On the virtue of succinct proofs: Amplifying communication complexity hardness to time-space trade-offs in proof complexity (Extended abstract). In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC '12)*, pages 233–248, May 2012.

[IP01] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, March 2001. Preliminary version in *CCC '99*.

[Joh98] Jan Johannsen. Lower bounds for monotone real circuit depth and formula size and tree-like cutting planes. *Information Processing Letters*, 67(1):37–41, July 1998.

[Joh01] Jan Johannsen. Depth lower bounds for monotone semi-unbounded fan-in circuits. *RAIRO-Theoretical Informatics and Applications*, 35(3):277–286, 2001.

[Juk11] Stasys Jukna. *Extremal Combinatorics with Applications in Computer Science*. Springer, 2nd edition, 2011.

[Kat68] Gyula O. H. Katona. A theorem of finite sets. In *Theory of Graphs*, pages 187–207. Akadémiai Kiadó, 1968.

[KN97] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[KPPY84] Maria Klawe, Wolfgang J. Paul, Nicholas Pippenger, and Mihalis Yannakakis. On monotone formulae with restricted depth. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing (STOC '84)*, pages 480–487, 1984.

[Kra98] Jan Krajíček. Interpolation by a game. *Mathematical Logic Quarterly*, 44(4):450–458, 1998.

[Kru63] Joseph B. Kruskal. The number of simplices in a complex. In Richard Bellman, editor, *Mathematical Optimization Techniques*, pages 251–278. University of California Press, 1963.

[KS90] Bala Kalyanasundaram and Georg Schnitger. Rounds versus time for the two person pebble game. *Information and Computation*, 88(1):1–17, September 1990. Preliminary version in *STACS '89*.

[KW90] Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *SIAM Journal on Discrete Mathematics*, 3(2):255–265, 1990. Preliminary version in *STOC '88*.

[Lev73] Leonid A. Levin. Universal sequential search problems. *Problemy peredachi informatsii*, 9(3):115–116, 1973. In Russian. Available at http://mi.mathnet.ru/ppi914.

[LMM$^+$20] Shachar Lovett, Raghu Meka, Ian Mertz, Toniann Pitassi, and Jiapeng Zhang. Lifting with sunflowers. Technical Report TR20-111, Electronic Colloquium on Computational Complexity (ECCC), November 2020.

[LMZ20] Shachar Lovett, Raghu Meka, and Jiapeng Zhang. Improved lifting theorems via robust sunflowers. Technical Report TR20-48, Electronic Colloquium on Computational Complexity (ECCC), April 2020.

[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. Preliminary version in *FOCS '91*.

[LT82] Thomas Lengauer and Robert Endre Tarjan. Asymptotically tight bounds on time-space trade-offs in a pebble game. *Journal of the ACM*, 29(4):1087–1130, October 1982. Preliminary version in *STOC '79*.

[MMZ$^+$01] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th Design Automation Conference (DAC '01)*, pages 530–535, June 2001.

[MS99] João P. Marques-Silva and Karem A. Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, May 1999. Preliminary version in *ICCAD '96*.

[Nor09] Jakob Nordström. A simplified way of proving trade-off results for resolution. *Information Processing Letters*, 109(18):1030–1035, August 2009.

[Nor12] Jakob Nordström. On the relative strength of pebbling and resolution. *ACM Transactions on Computational Logic*, 13(2):16:1–16:43, April 2012. Preliminary version in *CCC '10*.

[Nor20] Jakob Nordström. New wine into old wineskins: A survey of some pebbling classics with supplemental results. Manuscript in preparation. To appear in *Foundations and Trends in Theoretical Computer Science*. Current draft version available at http://www.csc.kth.se/~jakobn/research/, 2020.

[NW93] Noam Nisan and Avi Wigderson. Rounds in communication complexity revisited. *SIAM Journal on Computing*, 22(1):211–219, February 1993. Preliminary version in *STOC '91*.

[Pip80] Nicholas Pippenger. Pebbling. Technical Report RC8258, IBM Watson Research Center, 1980. In *Proceedings of the 5th IBM Symposium on Mathematical Foundations of Computer Science*.

[PV76] Nicholas Pippenger and Leslie G. Valiant. Shifting graphs and their applications. *Journal of the ACM*, 23(3):423–432, July 1976.

[Raz85] Alexander A. Razborov. Lower bounds for the monotone complexity of some Boolean functions. *Soviet Mathematics Doklady*, 31(2):354–357, 1985. English translation of a paper in *Doklady Akademii Nauk SSSR*.

[RM99] Ran Raz and Pierre McKenzie. Separation of the monotone NC hierarchy. *Combinatorica*, 19(3):403–435, March 1999. Preliminary version in *FOCS '97*.

[RY20] Anup Rao and Amir Yehudayoff. *Communication Complexity: and Applications*. Cambridge University Press, 2020.

[San01] Rahul Santhanam. Lower bounds on the complexity of recognizing SAT by Turing machines. *Information Processing Letters*, 79(5):243–247, September 2001.

[Val75] Leslie G. Valiant. Parallelism in comparison problems. *SIAM Journal on Computing*, 4(3):348–355, March 1975.

[Val77] Leslie G. Valiant. Graph-theoretic arguments in low-level complexity. In *Proceedings of the 6th International Symposium on Mathematical Foundations of Computer Science (MFCS '77)*, volume 53 of *Lecture Notes in Computer Science*, pages 162–176. Springer, September 1977.

[vM07] Dieter van Melkebeek. A survey of lower bounds for satisfiability and related problems. *Foundations and Trends in Theoretical Computer Science*, 2(3):197–303, October 2007.

[Wil08] Ryan Williams. Time-space tradeoffs for counting NP solutions modulo integers. *Computational Complexity*, 17(2):179–219, May 2008. Preliminary version in *CCC '07*.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il