Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 132 (2024)

Super-critical Trade-offs in Resolution over Parities Via Lifting

Arkadev Chattopadhyay
Tata Institute of Fundamental Research, Mumbai, India

Pavel Dvořák
Charles University, Prague, Czech Republic

Abstract

Razborov [24] exhibited the following surprisingly strong trade-off phenomenon in propositional proof complexity: for a parameter $k = k(n)$, there exists k-CNF formulas over n variables, having resolution refutations of $O(k)$ width, but every tree-like refutation of width $n^{1-\epsilon}/k$ needs size $\exp\left(n^{\Omega(k)}\right)$. We extend this result to tree-like Resolution over parities, commonly denoted by Res($\oplus$), with parameters essentially unchanged.

To obtain our result, we extend the lifting theorem of Chattopadhyay, Mande, Sanyal and Sherif [11] to handle tree-like affine DAGs. We introduce additional ideas from linear algebra to handle *forget* nodes along long paths.

2012 ACM Subject Classification Theory of computation $\rightarrow$ Proof complexity

Keywords and phrases Proof complexity, Lifting, Resolution over parities

Funding *Arkadev Chattopadhyay*: Funded by the Department of Atomic Energy, Government of India, under project no. RTI4001, and a Google India Research Award.
Pavel Dvořák: Major portion of work done as a visiting fellow at TIFR. Supported by Czech Science Foundation GAČR grant #22-14872O.

1 Introduction

Understanding trade-offs among complexity measures in a computational model is a well known interesting theme, with many published results (for example, time-space trade-offs [15, 16, 26], rounds-communication trade-offs [21, 10, 2] and space-size trade-offs in propositional proof complexity [6, 3, 5]). Typically, in these trade-offs, one showed that in various models of computation, simultaneous optimization of two complexity measures, like space and time, or rounds and total communication, or space and width (in refuting CNF formulas) is not always possible. In particular trying to optimize one complexity measure, necessarily leads to a huge blow-up in the other measure. For instance, in Yao's 2-party model of communication, the Greater-Than function can be computed in 1 round. It can also be computed using randomized protocols of communication cost $O(\log n)$. But every $O(1)$-round protocol, requires $\Omega(n)$ communication cost. On the other hand, every function has a protocol of cost $O(n)$. In all of the trade-off results cited above, the general story was that trying to optimize the use of one resource, led to the cost with respect to to the other resource shooting up to the cost needed by a naive/generic algorithm.

In 2016, Razborov [24] exhibited formulas for which very different and extreme kind of trade-offs hold in the propositional proof system of resolution. Although these unsatisfiable k-CNF formulas on n variables have refutations of $O(k)$ width, every one of their tree-like refutation of width less than $n^{1-\epsilon}/k$ has size $\exp(n^{\Omega(k)})$. That is, despite the fact that every n-variable formula has a generic tree-like refutation of size 2^n, these exhibited formulas that do have refutation of small width require super-critical tree-like refutation size whenever width is mildly restricted. Moreover, the super-critical size is in fact exponentially larger than the generic upper bound. Razborov remarked that such a phenomenon seemed extremely

ISSN 1433-8092

rare in the known body of tradeoff results in the computational complexity literature. In concluding his work, he urged finding more instances of such trade-offs. In response to that, follow-up works have appeared. They can be classified into two types. Ones which continue to focus on resolution and, others on more powerful proof systems. Examples of the former include work by Berkholz and Nordstrom [8], who showed super-critical trade-offs between width and space. A recent work of Berkholz, Lichter and Vinall-Smeeth [7] proves super-critical trade-offs for narrow resolution width and narrow tree-like size for refuting the isomorphism of two graphs.

The second type of work answers Razborov's call by finding such trade-offs in stronger proof systems. This includes the recent work of Fleming, Pitassi and Robere [14] who first showed that the argument of Razborov extends to general resolution DAGs. They then use it along with appropriate lifting theorems to prove trade-offs between size and depth for DAG like Resolution, Res(k), and cutting planes. In very recent progress in the area, De Rezende, Fleming, Jannet, Nordström, and Pang [12], and Göös, Maystre, Risse, Sokolov [18] independently showed super-critical trade-offs not only for various proof systems but also the first super-critical depth-size tradeoffs for monotone circuits. Our work also falls in this second type as we study tree-like resolution over parities, that generalizes tree-like resolution.

We exhibit super-critical trade-offs for width and tree-like size/depth in the style of Razborov for resolution over parities, denoted by Res($\oplus$). This system, introduced by Itsykson and Sokolov [19, 20], is one of the simplest generalizations of resolution for which obtaining super-polynomial lower bounds on size of refutations is a current, well known, challenge. Very recent works (see [13, 9]) managed to obtain exponential lower bounds on the size of *regular proofs* in this system.

Our work here will concern tree-like Res($\oplus$) proofs. Lower bounds for them were obtained by Itsykson and Sokolov [20] themselves. More recently, two independent works, one by Beame and Koroth [4] and the other by Chattopadhyay, Mande, Sanyal and Sherif [11], proved lifting theorems that yielded a systematic way of lifting tree-like resolution width complexity to strong lower bounds on size of tree-like Res($\oplus$) proofs for formulas lifted with constant-size gadgets. In this paper, we extend the lifting theorem by Chattopadhyay et al. [11] in the following manner. Their result was applicable to parity decision trees (duals of tree-like Res($\oplus$) proofs) that only had usual nodes where the algorithm queried (correspondingly the proof resolved on) an $\mathbb{F}_2$ linear form. We call such nodes as query nodes. On the other hand, we want to deal here with width-bounded proofs that could be much deeper than n, the total number of variables of the formula. This would correspond to parity decision trees where the height is much larger than n, and therefore, necessarily there are nodes that *forget.* The affine space corresponding to such a forget node u is strictly contained in the affine space corresponding to u's only child node v. Alternatively, in the bottom-up view of the corresponding proof, the linear clause at v is strictly weakened to get the linear clause at u. Dealing with such nodes, so that the width of the (ordinary) clauses in the extracted resolution proof never exceed the corresponding width of the linear clauses, is the main technical contribution of this work. Thus, we establish a depth-to-size lifting result from tree-like Res($\oplus$) of arbitrary depth to tree-like resolution, which also preserves the width of the refutation.

▶ **Theorem 1.** *Let $\Phi \circ g$ be a lift of a contradiction Φ by an appropriate gadget $g : \{0,1\}^\ell \to \{0,1\}$. Suppose there is a tree-like Res($\oplus$) refutation for $\Phi \circ g$ with size s and width w. Then, there is a tree-like resolution refutation for Φ with depth at most $\log s$ and width at most w.*

▶ Remark 2. We point out the precise difference between our Theorem 1 and the earlier lifitng theorem of Chattopadhyay et al [11]. The earlier theorem, given a tree-like refutation of $\Phi \circ g$ in Res($\oplus$) of size s and width w, would have extracted a tree-like refutation of Φ in ordinary resolution of depth $\log s$, with no guarantees on the width of this refutation. In fact, the width could get as large as the depth of the extracted refutation, i.e. $\log s$. In super-critical trade-offs, which is our chief interest here, the width w of the given Res($\oplus$) refutation of $\Phi \circ g$ could be exponentially smaller than $\log s$. This renders the earlier lifting theorem unusable for demonstrating such trade-offs.

Applying Theorem 1 to the trade-off by Razborov [24], we immediately obtain an analogous trade-off in the Res($\oplus$) proof system.

▶ **Theorem 3.** *Let $k = k(n) \geq 12$ be any parameter and let $\varepsilon > 0$ be an arbitrary constant. Then, there exists a k-CNF contradiction τ over n variables such that there is a resolution refutation for τ with width at most $O(k)$, but for every tree-like Res($\oplus$) refutation Π for τ with $w(\Pi) \leq n^{1-\varepsilon}/k$, we have the bound $|\Pi| \geq \exp(n^{\Omega(k)})$.*

The contradiction τ from the previous theorem is a lift of the contradiction τ' constructed by Razborov [24] by an appropriate gadget $g : \{0,1\}^\ell \to \{0,1\}$ with a constant size. A caveat of τ' (as Razborov also noted) is that the number of clauses of τ' is $n^{\Theta(k)}$. Naturally, this caveat is inherited by our contradiction τ. This issue was addressed in very recent work of de Rezende et al. [12]. They provided a contradiction χ such that the size of its tree-like refutation with bounded width is super-exponential in not just in the number of variables but also in the size of the formula. However, the bound on width of the tree-like resolution for which super-critical size is needed is much more strict than in Razborov's result. They showed the super-critical trade-off only for tree-like refutation of width smaller than $2w$, where w is the width of the resolution refutation of χ. Since our gadget has size 3, we can guarantee only resolution refutation with width at most $3w$ for the lifted formula $\chi \circ g$. Thus, we can not lift their super-critical trade-off to Res($\oplus$) as it is extremely sensitive to width. Razborov's result [24], on the other hand, is more robust making it possible to be lifted by our Theorem 1.

If one were to construct another formula $\tilde{\tau}$ improving state-of-the-art in supercritical trade-off between width and size of tree-like resolution that is not so sensitive to the width of the refutations, this improvement could be used with our simulation theorem (Theorem 1) and be lifted to tree-like Res($\oplus$).

Relation to Other Recent Works

The Res($\oplus$) proof system has been an active area of research. Recently, Efremenko, Garlík, and Itsykson [13] showed that the binary pigeonhole principle formula requires an exponential-size refutation within the so-called bottom-regular Res($\oplus$). The bottom-regular Res($\oplus$) is a fragment of Res($\oplus$) that contains both tree-like Res($\oplus$) and regular resolution proof systems. Furthermore, Bhattacharya, Chattopadhyay, and Dvořák [9] showed that bottom-regular Res($\oplus$) can not polynomially simulate even ordinary, but DAG-like resolution. This separation was very recently improved quantitatively by Alekseev and Itsykson [1].

Furthermore, Alekseev and Itsykson [1] established a width-to-width lifting from resolution to Res($\oplus$). They proved this in a contra-positive way – if there is no resolution refutation of a contradiction Φ with width w, then there is no width-w Res($\oplus$) refutation of a lift of Φ by an appropriate gadget g. They utilized a game interpretation of resolution and Res($\oplus$) to prove their lifting theorem. While their proof is quite short, it is unclear whether their technique can be adapted to prove the depth-to-size lifting theorem as we need in order to show the trade-off in Res($\oplus$) (our Theorem 3). In particular, their theorem seems incomparable to the depth-to-size lifting of Chattopadhyay et al. [11]. On the other hand, since any refutation can be expanded into a tree-like refutation (with a possible exponential blow-up in size), our lifting theorem (Theorem 1) immediately implies the width-to-width lifting theorem of Alekseev and Itsykson (however our proof seems more involved). Hence, our Theorem 1 effectively contains a common generalization of the width-to-width lifting of Alekseev and Itsykson [1] and depth-to-size lifting of Chattopadhyay et al. [11]. Moreover, we use a completely different technique than Alekseev and Itsykson [1]. Specifically, we establish our lifting theorem directly by constructing a tree-like resolution refutation for a contradiction Φ simulating a tree-like Res($\oplus$) refutation for $\Phi \circ g$. To achieve this, we use some ideas from linear algebra that, to our knowledge, have not been previously utilized in the context of lifting.

Overview of Our Ideas

Overall, the ideas behind the proof of Theorem 1 are inspired by the work of Chattopadhyay et al. [11]. However, they did not consider Res($\oplus$) refutation with a limited width. Thus, they only needed to process query nodes to prove their lifting theorem. In contrast, our setting also involves forget nodes, where a linear equation from the span of previously queried equations is forgotten. It turns out that processing forget nodes is non-trivial. In particular, an affine space can be viewed in two ways: the first is by the (linear) space of constraints that could be thought of as the dual view. The primal view is that of the set of vectors lying in space represented by a basis of the underlying vector space and a shift vector. Previously, in [11], the dual view was very effectively used for depth-to-size lifting in the absence of forget nodes. This is because a query node naturally adds a constraint to the dual space. On the other hand, a forget node increases the dimension of the affine space. This new space is not conveniently representable with respect to the basis maintained for the dual space of constraints just before 'forgetting' happens. Here it seems the primal view is more helpful as any basis of a space A_1 can be extended to a basis of a space A_2 whenever $A_1 \subseteq A_2$. The main tool we use is a a characterization, via Theorem 10, of the constraint space of A_2 in terms of the constraint space of A_1, where $A_1 \subseteq A_2 \subseteq \mathbb{F}_2^n$. The proof of this turns out to be simple[1]. With more ideas, including a new notion of *strongly stifled* gadgets that extends the earlier notion of stifling introduced by [11], Section 7 yields the process of dealing with forget nodes.

2 Tree-like Proofs and Decision Trees

A proof in a propositional proof system starts from a set of clauses Φ, called axioms, that is purportedly unsatisfiable. It generates a proof by deriving the empty clause from the axioms, using inference rules. The main inference rule in the standard resolution, called the resolution rule, derives a clause $A \vee B$ from clauses $A \vee x$ and $B \vee \neg x$ (i.e., we resolve the variable x). If we can derive the empty clause from the original set Φ, then it proves that the set Φ is unsatisfiable.

Resolution over parities (Res($\oplus$)) is a generalization of the standard resolution, using linear clauses (disjunction of linear equations in $\mathbb{F}_2$) to express lines of a proof. It consists of two rules:

Resolution Rule: From linear clauses $A \vee (\ell = 0)$ and $B \vee (\ell = 1)$ derive a linear clause $A \vee B$.

[1] Our original proof was complicated. The much simpler proof we present here has been pointed out to us by an anonymous referee.

Weakening Rule: From a linear clause A derive a linear clause B that is semantically implied by A (i.e., any assignment satisfying A also satisfies B).

The length $|\Pi|$ of a resolution (or Res($\oplus$)) refutation Π of a formula Φ is the number of applications of the rules above in order to refute the formula Φ. The width $w(\Pi)$ of a resolution (or Res($\oplus$)) refutation Π is the maximum width of any (linear) clause that is used in the resolution proof. A (linear) resolution proof is *tree-like* if the resolution rule is applied in a tree-like fashion. The depth $d(\Pi)$ of the tree-like proof Π is the depth of the underlying tree (i.e., the length of the longest path from the root to a leaf).

We can replace the general resolution rule with a canonical one:

Canonical Resolution Rule: From linear clauses $C \vee (\ell = 0)$ and $C \vee (\ell = 1)$ derive a linear clause C.

Using the canonical resolution rule instead of the general one will not make the proof system substantially weaker. If we want to apply the resolution rule on the clauses $A \vee (\ell = 0)$ and $B \vee (\ell = 1)$, we can apply the weakening rule to both of them to get linear clauses $A \vee B \vee (\ell = 0)$ and $A \vee B \vee (\ell = 1)$ and then apply the canonical resolution rule to derive the clause $A \vee B$. Thus, from a tree-like Res($\oplus$) refutation Π of a contradiction Φ, we can derive an equivalent tree-like Res($\oplus$) refutation Π' that uses only canonical resolution rule (and the weakening rule) with $|\Pi'| \leq 3 \cdot |\Pi|$, $d(\Pi') \leq 2 \cdot d(\Pi)$, and $w(\Pi') \leq w(\Pi) + 1$ as for each application of the resolution rule in Π we add 2 applications of the weakening rule in Π', and the width might increase by 1 as we introduce clause $A \vee B \vee (\ell = 0)$ and $A \vee B \vee (\ell = 1)$ but only the clause $A \vee B$ was present in Π.

It is known that a tree-like resolution (or Res($\oplus$)) proof, for an unsatisfiable set of clauses Φ, corresponds to a (parity) decision tree for a search problem *Search*(Φ) that is defined as follows. For a given assignment α of the n variables of Φ, one needs to find a clause in Φ that is not satisfied by α (at least one exists as the set Φ is unsatisfiable). The correspondence holds even for general (not only tree-like) proofs (see for example Garg et al. [17], who credit it to earlier work of Razborov [23] that was simplified by Pudlák [22] and Sokolov[25]), but in this paper, we are interested only in tree-like proofs.

Let $R \subseteq \{0,1\}^m \times O$, where O is a set of possible outputs. A forgetting parity decision tree (FPDT) computing R is a tree $\mathcal{T}$ such that each node has at most two children and the following conditions hold:

- Each node v is associated with an affine space $A_v \subseteq \mathbb{F}_2^m$.
- For every node v with two children u and w is called a *query nodes*. There is a linear query f_v such that $A_u = \{x \in A_v | \langle f_v, x\rangle = 0\}$ and $A_w = \{x \in A_v | \langle f_v, x\rangle = 1\}$, or vice versa. We say that f_v is the *query* at v.
- Every node v with exactly one child u is called a *forget node*. It holds that $A_v \subseteq A_u$.
- Each leaf ℓ is labeled by $o_\ell \in O$ such that for all $x \in A_\ell$, it holds that $(x, o_\ell) \in R$.
- For the root r, $A_r = \mathbb{F}_2^m$.

The size $|\mathcal{T}|$ of an FPDT $\mathcal{T}$ is the number of nodes of $\mathcal{T}$ and the width $w(\mathcal{T})$ of FPDT $\mathcal{T}$ is the largest integer w such that there exists an affine space of co-dimension at least w associated with some node of $\mathcal{T}$. The depth of $\mathcal{T}$ is denoted $d(\mathcal{T})$. Note that there are no forget nodes in a standard parity decision tree. Thus, for such trees, the width is exactly the depth of the tree. It no longer holds for this model, because we may "forget" some linear queries that have been made earlier.

A forgetting decision tree (FDT) is defined similarly to FPDT but instead of affine spaces, cubes are associated with each node. Consequently, the width $w(\mathcal{T})$ of FDT is the maximum number w such that there exists a cube of width at least w associated with some node of $\mathcal{T}$ and queries of single variables replace the linear queries at nodes.

The correspondence between an F(P)DT's and tree-like resolution (or Res($\oplus$)) proofs using only canonical resolution rule is the following: We represent a (linear) resolution proof as a tree where nodes are associated with (linear) clauses. The leaves are associated with clauses of Φ and the root is associated with the empty clause. Each node with two children corresponds to an application of the canonical resolution rule and each node with exactly one child corresponds to an application of the weakening rule. To get an F(P)DT for *Search*(Φ) we just negate the clauses that are associated with the nodes. Thus, each node is associated with a cube (or an affine space in the case of Res($\oplus$)/FPDT). Moreover, a cube C_v (or an affine space A_v) associated with the node v of an F(P)DT $\mathcal{T}$ contains exactly the falsifying assignments of the (linear) clause that is associated with the corresponding node in the tree-like refutation Π that corresponds to $\mathcal{T}$. It is clear that the width and the depth of such decision tree $\mathcal{T}$ are exactly the same as the width and the depth of the corresponding tree-like refutation Π and the length $|\Pi|$ equals the number of inner nodes of $\mathcal{T}$ (as the inner nodes of $\mathcal{T}$ correspond to the applications of the resolution rule).

We say an FPDT $\mathcal{T}$ is *canonical* if for each forget node v of $\mathcal{T}$ and its only child u, it holds that $\text{co-dim}(A_v) = \text{co-dim}(A_u) + 1$. We say an FPDT $\mathcal{T}$ is *succinct* if the parent of each forget node is a query node. Note that any FPDT can be transformed into an equivalent canonical (or succinct) FPDT by expanding forget nodes into paths of forget nodes (or contracting paths of forget nodes to single vertices).

Consider an FPDT $\mathcal{T}$ and its succinct form $\bar{\mathcal{T}}$. Note that the number of query nodes of $\bar{\mathcal{T}}$ and $\mathcal{T}$ is the same, and analogously the number of query nodes on a root-leaf path in $\bar{\mathcal{T}}$ equals the number of query nodes of the corresponding path in $\mathcal{T}$. Thus, for an FPDT $\mathcal{T}$ we define query size $|\mathcal{T}|_{\mathsf{q}}$ and query depth $d_{\mathsf{q}}(\mathcal{T})$ to be the number of query nodes of $\mathcal{T}$ and the maximum number of query nodes on a root-leaf path of $\mathcal{T}$.

▶ **Observation 4.** *Let Π be a Res($\oplus$) refutation of a contradiction Φ. Then, there is a canonical FPDT $\mathcal{T}$ computing Search(Φ) with $|\mathcal{T}|_{\mathsf{q}} \le |\Pi|$ and $w(\mathcal{T}) \le w(\Pi) + 1$.*

Proof. As discussed above, first we modify Π to a Res($\oplus$) refutation Π' that uses only canonical resolution rule (and weakening rule). By this modification, we have $w(\Pi') \le w(\Pi)+1$ and the number of applications of the resolution rule in Π is exactly the number of applications of the canonical resolution rule in Π'. From Π', we derive an FPDT $\mathcal{T}'$ computing *Search*(Φ) that we finally modify to an equivalent canonical FPDT $\mathcal{T}$. The width of Π' and $\mathcal{T}'$ is the same, and the query size of $\mathcal{T}$ is exactly the number of applications of the canonical resolution rule in Π'. The modification of $\mathcal{T}'$ to the canonical FPDT $\mathcal{T}$ does not change the width and the query size of the tree. Thus, we have $w(\mathcal{T}) \le w(\Pi) + 1$ and $|\mathcal{T}|_{\mathsf{q}} \le |\Pi|$. ◀

▶ **Observation 5.** *Let $\mathcal{T}$ be a succinct FPDT computing Search(Φ) and Π be the tree-like Res($\oplus$) refutation of Φ corresponding to $\mathcal{T}$. Then, $|\Pi| \le 3 \cdot 2^{d_{\mathsf{q}}(\mathcal{T})}$.*

Proof. As mentioned above, the length of Π equals the number of inner nodes of $\mathcal{T}$. The tree $\mathcal{T}$ has at most $2^{d_{\mathsf{q}}(\mathcal{T})} - 1$ query nodes. Since $\mathcal{T}$ is succinct, the number of forget nodes is at most twice the number of query nodes as each query node might have at most two forget nodes as its children. Further, any child of a forget node is not a forget node. Thus, $|\Pi| \le 3 \cdot 2^{d_{\mathsf{q}}(\mathcal{T})}$. ◀

3 Lifting of Relations and Formulas

Let $g : \{0,1\}^\ell \to \{0,1\}$ be a boolean function. For a relation $R \subseteq \{0,1\}^n \times O$ we define its lift $R \circ g \subseteq \{0,1\}^{n\ell} \times O$ as

$$R \circ g = \big\{(y, o) \in \{0,1\}^{n\ell} \times O \mid \big(\overrightarrow{g}(y), o\big) \in R\big\},$$

where $\overrightarrow{g}(y^1_1, \dots, y^1_\ell, \dots, y^n_1, \dots, y^n_\ell) = \big(g(y^1_1, \dots, y^1_\ell), \dots, g(y^n_1, \dots, y^n_\ell)\big)$.

For CNF Φ over n variables $\{x_1, \dots, x_n\}$, let $\Phi \circ g$ be the following lift of Φ over the variables $\{y^i_j \mid i \in [n], j \in [\ell]\}$. For any clause D of Φ, let $\mathrm{Vars}(D)$ be the set of variables of D, and let $\eta_D \in \{0,1\}^{\mathrm{Vars}(D)}$ be the only falsifying assignment of D. Then,

$$D \circ g = \left\{ \bigvee_{x_i \in \mathrm{Vars}(D), j \in [\ell]} y^i_j \neq \kappa^i_j \;\middle|\; \kappa \in \overrightarrow{g}^{-1}(\eta_D) \right\},$$

where $y^i_j \neq \kappa^i_j$ is the following literal:

$$y^i_j \neq \kappa^i_j = \begin{cases} y^i_j & \text{if } \kappa^i_j = 0, \\ \neg y^i_j & \text{if } \kappa^i_j = 1. \end{cases}$$

Now, the clauses of $\Phi \circ g$ are $\{D \circ g \mid D \text{ clause of } \Phi\}$.

▶ **Observation 6.** *For a clause D, an assignment δ of $\mathrm{Vars}(D \circ g)$ falsifies $D \circ g$ if and only if $\overrightarrow{g}(\delta) = \eta_D$, i.e., $\overrightarrow{g}(\delta)$ falsifies D.*

4 Notation

We use the following notation. For a vector u we use both u_i and $u[i]$ to denote the i-th entry of u, similarly for a matrix M we use $M_{i,j}$ and $M[i;j]$ to denote the entry in the i-th row and j-th column. For an ordered set of indices D we denote by $u[D]$ the subvector of u given by D, i.e., $u[D] = (u_i)_{i \in D}$. For an abbreviation, we use $u[-i]$ to denote the vector u without the i-th entry, i.e., $u[-i] = (u_1, \dots, u_{i-1}, u_{i+1}, \dots, u_n)$.

5 Stifling

In this section, we extend the notion of stifling introduced by Chattopadhyay et al. [11]. Let $g : \{0,1\}^\ell \to \{0,1\}$ be a Boolean function. For $i \in [\ell]$ and $b \in \{0,1\}$, we say a partial assignment $\delta \in \{0,1,*\}^\ell$ is an (i,b)-stifling pattern for g if $\delta_j = *$ if and only if $j = i$, and for any $\gamma \in \{0,1\}^\ell$ such that $\gamma[-i] = \delta[-i]$, we have $g(\gamma) = b$. In words, δ assigns a value to all but the i-th bit and when we extend δ to a full assignment γ, it holds that $g(\gamma) = b$ no matter how we set the value of the i-th bit.

▶ **Definition 7.** *A Boolean function $g : \{0,1\}^\ell \to \{0,1\}$ is* strongly stifled *if there is a collection $P := \left\{\delta^{i,b} | i \in [\ell], b \in \{0,1\}\right\}$ where each $\delta^{i,b}$ is an (i,b)-stifling pattern for g and*

$$\forall\ i \in [\ell], b \in \{0,1\}, \text{and } \emptyset \neq D \subseteq [\ell] \setminus \{i\}$$
$$\exists\ j \in D \text{ such that } \delta^{j,b}\big[D \setminus \{j\}\big] = \delta^{i,b}\big[D \setminus \{j\}\big].$$

The collection P is called a converting collection of stifling patterns *of g.*

Chattopadhyay et al. [11] defined a stifled function (namely 1-stifled) as a function $g : \{0,1\}^\ell \to \{0,1\}$ such that for each $i \in [\ell]$ and $b \in \{0,1\}$ there is an (i,b)-stifling pattern for g. In this work, we require not only the existence of the stifling patterns but a stronger property that we can convert the stifling patterns to each other. More formally, consider an (i,b)-stifling pattern $\delta^{i,b}$ from the collection P (from the definition above). Let an adversary give us a set of coordinates $D \subseteq [\ell] \setminus \{i\}$. Then, we are able to pick a coordinate $j \in D$ such that the stifling pattern $\delta^{j,b}$ is equal to $\delta^{i,b}$ on all coordinates in $D \setminus \{j\}$.

By a simple verification we can show that indexing of two bits $\mathsf{IND}_1 : \{0,1\}^3 \to \{0,1\}$ and majority of 3 bits $\mathsf{MAJ}_3 : \{0,1\}^3 \to \{0,1\}$ are strongly stifled functions, where $\mathsf{IND}_1(a, d_0, d_1) = d_a$ and $\mathsf{MAJ}_3(x) = 1$ if and only if $\big|\{i \in [3] | x_i = 1\}\big| \geq 2$.

▶ **Observation 8.** *The functions* IND_1 *and* MAJ_3 *are strongly stifled.*

Further, the strongly stifled notion is actually stronger than the original stifled notion, because the inner product function of 2-bit vectors $\mathsf{IP}_2 : \{0,1\}^4 \to \{0,1\}$ is stifled [11] but not strongly stifled, where $\mathsf{IP}_2(x_1, x_2, y_1, y_2) = x_1x_2 + y_1y_2 \mod 2$.

▶ **Observation 9.** *The function* IP_2 *is not strongly stifled.*

For more details, see the appendix.

6 Linear Algebraic Tools

Let $A \subseteq \mathbb{F}_2^m$ be an affine space over a field $\mathbb{F}_2$. A *constraint representation* of A is a system of linear equations $(M|z)$ where $M \in \mathbb{F}_2^{q\times m}$ and $z \in \mathbb{F}_2^q$ for some $q \leq m$ such that $A = \{y \mid My = z\}$. The columns of M correspond to the variables of the system $(M|z)$ and rows of M correspond to the constraints. We say a constraint i *contains* a variable a if $M_{i,a} = 1$. A matrix $M \in \mathbb{F}_2^{q\times m}$ is in an *echelon form* if there are q columns $c_1 < c_2 < \cdots < c_q \in [m]$ such that for all $i \in [q]$ it holds that

$$M[i; c_j] = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

Thus, the submatrix of M induced by the columns $c_1, \ldots, c_q$ is the identity matrix $I_q \in \mathbb{F}_2^{q\times q}$. The variables corresponding to the columns $c_1, \ldots, c_q$ are *dependent* variables of the system $(M|z)$ and the remaining variables are *independent*. The c_i-th entry of the i-th row of M is called the *pivot of the i-th row*. We say a constraint representation $(M|z)$ is in an *echelon form* if the matrix M is in an echelon form.

Let $C \in \mathbb{F}_2^{q\times m}$ be a matrix and $t \in \mathbb{F}_2^q$ be a non-zero vector. We define a matrix $C' = \mathsf{Del}(C, t, i) \in \mathbb{F}_2^{q-1\times m}$ where C' arises from C by adding the i-th row to all rows j such that $t_j = 1$ and then deleting the row i. Analogously, we define the Del operation for a constraint representation $(M|z)$ of an affine space $A \subseteq \mathbb{F}_2^m$, where we treat the vector z as the last column of the matrix C. It turns out that the Del operation is the only operation needed to get a constraint representation for a super-space as shown in the following theorem that will be the key technical tool for us to process forget nodes while proving our main lifting theorem.

▶ **Theorem 10.** *Let* $A_1 \subseteq A_2 \subseteq \mathbb{F}_2^m$ *be two affine spaces such that* $\dim(A_2) = \dim(A_1) + 1$. *Let* $(M_1|z_1)$ *be a constraint representation in the echelon form of* A_1 *such that* $M_1 \in \mathbb{F}_2^{q\times m}$. *Then, there is a non-zero vector* $t \in \mathbb{F}_2^q$ *such that the following is true: for every* $i \in [q]$ *with* $t_i = 1$, $\mathsf{Del}\big((M_1|z_1), t, i\big)$ *is a constraint representation of* A_2 *in echelon form.*

We call the vector t given by Theorem 10 as *forgetting vector* because it allows us to forget one constraint in the representation of A_1 to get a representation of A_2. Note that the dimension of t equals the number of equations in the system $(M_1|z_1)$. We say t *contains* a constraint i if $t_i = 1$.

Theorem 10 follows from the following well-known lemma, which we will prove for completeness. Let $V_2 \subseteq V_1 \subseteq \mathbb{F}_2^n$ be two vector spaces such that $\dim(V_2) = \dim(V_1) - 1$.

▶ **Lemma 11.** *For any $u, v \in V_1 \setminus V_2$ holds that $u + v \in V_2$.*

Proof. Since u is in V_1 but not in V_2, we have that $\mathrm{Span}(V_2, u) = V_1$ by the dimensions of V_1 and V_2. Since $v \in V_1 \setminus V_2$ as well, we have that $v = w + u$ for an appropriate vector $w \in V_2$. It follows that $u + v = w \in V_2$. ◀

▶ **Corollary 12.** *Let $v_1, \ldots, v_d$ be a basis of V_1. Let $T = \{i \in [d] \mid v_i \notin V_2\}$. Let $j \in T$, and for $i \in [d] \setminus j$ let*

$$u_i = \begin{cases} v_i + v_j & \text{if } i \in T, \\ v_i & \text{otherwise.} \end{cases}$$

Then, $(u_i)_{i \in [d] \setminus j}$ is a basis of V_2.

Proof. Vectors u_i's are linearly independent as v_i's are linearly independent. By definition and Lemma 11, all u_i's are in V_2. Since $\dim(V_2) = d - 1$, the vectors u_i's has to form a basis of V_2. ◀

Theorem 10 follows from the previous corollary by setting V_1 and V_2 to be the constraints spaces of A_1, and A_2, respectively (i.e., row spaces of matrices $(M_1|z_1)$ and $(M_2|z_2)$). Further, the forgetting vector of Theorem 10 is the characteristic vector of the set T given by Corollary 12.

7 Simulation

In this section, we prove our lifting theorem.

▶ **Theorem 13** (Theorem 1 stated for F(P)DT). *Let $R \subseteq \{0,1\}^n \times O$ be a relation and $\mathcal{T}$ be a canonical FPDT computing $R \circ g$ where $g : \{0,1\}^\ell \to \{0,1\}$ is a strongly stifled gadget. Then, there is an FDT $\mathcal{T}'$ computing R such that $d_{\mathsf{q}}(\mathcal{T}') \le \log |\mathcal{T}|_{\mathsf{q}}$ and $w(\mathcal{T}') \le w(\mathcal{T})$.*

Algorithm

We prove Theorem 13 by simulation. On an input $x \in \{0,1\}^n$, the constructed FDT $\mathcal{T}'$ simulates given FPDT $\mathcal{T}$ on an input $y \in \{0,1\}^m$ (for $m := n\ell$) with $x = \overrightarrow{g}(y)$ by traversing a path from the root r of $\mathcal{T}$ to a leaf. The main loop of the simulation is quite simple. We start in the root r of $\mathcal{T}$ and in each iteration, we process the current node v of $\mathcal{T}$ and pick a new node. Sometimes during the processing of a node of $\mathcal{T}$, we query or forget a bit of x. When we reach a leaf s of $\mathcal{T}$ we just output the value of s. The main loop is summarized in Algorithm 1.

Let v be a current node of $\mathcal{T}$ we have just encountered. We maintain a constraint representation in echelon form $(M_v|z_v)$ of the affine space A_v. We store queried (and not forgotten) bits of x in a partial assignment $\rho_v \in \{0,1,*\}^n$. Let $C(\rho_v) \subseteq \{0,1\}^n$ be a set of all possible extension of ρ_v and $w(\rho_v)$ be a number of fixed bits of ρ_v. Thus, $C(\rho_v)$ is a cube and $w(\rho_v)$ is its width. Our goal is that any $\bar{x} \in C(\rho_w)$ is represented in A_v (i.e., there is $y \in A_v$ such that $\overrightarrow{g}(y) = \bar{x}$), and that $w(\rho_v)$ is at most the co-dimension of A_v.

An input y of $\mathcal{T}$ is divided into n blocks $B_1, \ldots, B_n \subseteq [m]$, each of size ℓ, and each such block corresponds to exactly one entry of x. Formally, $B_i = \{(i-1)\ell + 1, \ldots, i\ell\}$. During the simulation of $\mathcal{T}$, we divide the blocks into two groups – free and fixed. Fixed blocks correspond to the entries of x that were queried and were not forgotten – i.e., the entries fixed by ρ_v. The other blocks are free.

For each fixed bit $i \in [n]$ of ρ_v, we have a unique constraint j_i of $(M_v|z_v)$ such that the pivot of the constraint j_i is in the block B_i. The constraint j_i is called *the primary constraint of* B_i. The dependent variables of primary constraints are called *marked variables.* We will keep an invariant that each marked variable is in a different block, i.e., each fixed block contains a unique marked variable. The other (non-marked) variables of the fixed blocks are called stifling variables.

The constraints that are not primary for any block are called *secondary.* We will keep invariants that the all secondary constraints contain variables only from fixed blocks. All variables of free blocks are called free. Thus, the matrix M_v has the following form (after rearranging columns):

Linear algebra classification:	Dependent		Independent		
$M_v =$	I_{d_1}	0	C_1	F	primary constraints
	0	I_{d_2}	C_2	0	secondary constraints
Simulation classification:	Marked	Stifling		Free	variables
	Fixed			Free	blocks

Let $P := \left\{\delta^{i,b} \mid i \in [\ell], b \in \{0,1\}\right\}$ be a converting collection of stifling patterns of g, given by the assumption. Let $\alpha_v \in \{0,1,*\}^m$ be the following partial assignment:

$$\alpha_v[B_i] = \begin{cases} \delta^{j,x_i} & \text{if } B_i \text{ is a fixed block and } j \text{ is the index of the marked variable of } B_i \\ *^{\ell} & \text{if } B_i \text{ is a free block} \end{cases}$$

We will keep an invariant that if we set all dangerous variables according to the pattern α_v all secondary constraints of $(M_v|z_v)$ will be satisfied. This will help us to ensure that each $\bar{x} \in C(\rho_v)$ is represented in A_v.

At the beginning of our simulation, we are at the root r of $\mathcal{T}$. Since $A_r = \{0,1\}^m$, the matrix M_r is an empty matrix and all variables are free because we have not queried any entry of x yet. Also, the patterns ρ_r and α_r do not contain any fixed bit, i.e. they are equal to $*^n$, or $*^m$, respectively.

Algorithm 1 Simulation

Input: $x \in \{0,1\}^n$ ▷ Input for FDT $\mathcal{T}'$
FPDT $\mathcal{T}$ with the root r computing $R \circ g$

Initialization:
```
v ← r                         ▷ Current node of 𝒯
ρ_r ← *^n                     ▷ Known bits of x
(M_r|z_r) ← ∅                 ▷ Constraint representation of A_r = {0,1}^m
```

Simulation:
```
while v is not a leaf do
    if v is a query node then
        Process Query Node    ▷ Algorithm 2
    else                      ▷ v is a forget node
        Process Forget Node   ▷ Algorithm 3
return the output of v
```

During the simulation, we will maintain the following invariants.

Invariant 1: The system of equations $(M_v|z_v)$ is a constraint representation of A_v in the echelon form.

Invariant 2: All variables of all free blocks are independent.

Invariant 3: For each fixed bit $i \in [n]$ of ρ_v, there is a unique constraint j_i of $(M_v|z_v)$ such that the pivot (i.e., the marked variable) of the constraint j_i is in the block B_i.

The constraint j_i from the previous invariant is called the *primary constraint* of the block B_i. The constraints that are not primary for some block are called *secondary*.

Invariant 4: The partial assignment α_v assigns values to all stifling variables and any extension of α_v to a full assignment satisfies all secondary constraints.

Note that Invariant 4 implies that secondary constraints of $(M_v|z_v)$ contain only variables of fixed blocks. We will show these invariants hold for any node v of $\mathcal{T}$ at the moment, when v is checked whether v is a leaf – i.e., at Line 4 of Algorithm 1. Clearly, the invariants hold for the root r of $\mathcal{T}$. Now, we describe how we process a current node v (depending on whether v is a query node or forget node). We suppose the invariants hold for v. During the processing, we pick an appropriate child u of v and make u the new current node. Subsequently, we argue why the invariants hold for u.

We remark that the query node processing is a careful adaptation of the node processing given by Chattopadhyay et al. [11]. All new machinery (strongly stifled function and obtaining a constraint representation of a super-space by Theorem 10) is used only for the processing of forget nodes.

Query Nodes

When v is a query node, then v introduce a new parity query f_v, and if $\langle f_v, y\rangle = 0$ the computation of $\mathcal{T}$ proceeds to the left child u_0 of v, otherwise to the right child u_1. Our goal is to pick an appropriate child u of v and create the system $(M_u|z_u)$ representing A_u satisfying all our requirements. Let us start with a system $\big(M'|z(b)\big)$, where

$$M' = \begin{pmatrix} M_v \\ f_v \end{pmatrix}, z(b) = \begin{pmatrix} z_v \\ b \end{pmatrix},$$

with b being a parameter equal to 0 or 1. We fix the value of b when we pick the appropriate child u of v as the new node. Surely, the system $\big(M'|z(b)\big)$ represents the space A_{u_b}, however, it might not satisfy our requirements (for example the matrix M' might not be in the echelon form). Note that the matrix M' does not depend on the value of b. We do another pivoting step of the Gaussian elimination to get the system $\big(M'|z(b)\big)$ into the echelon form, i.e.,

1. We zero out all coefficients in f_v corresponding the dependent variables in $(M_v|z_v)$, to get a new constraint $(f'|b')$, where b' is a function of b. We call the new constraint $(f'|b')$ the *reduced form* of the constraint $(f_v|b)$.
2. We pick one of the remaining variables a contained in f' as a new dependent variable, we pick an appropriate child u of v and we set the value of b (and b'), accordingly.
3. We zero out all coefficients corresponding to a in all original constraints from the system $(M_v|z_v)$ to get the new system $(M_u|z_u)$.

It is clear the new system $(M_u|z_u)$ is a constraint representation of A_u (i.e., Invariant 1 will hold for u). The crucial part is to pick a new dependent variable a in Step 2 of the executed Gaussian elimination. Note that the reduced constraint $(f'|b')$ does not contain any marked variable as all marked variables are dependant and, thus, they are zeroed out from $(f|b)$ in Step 1 of the executed pivoting step. There are two cases to consider as follows.

Case 1: The new constraint $(f'|b')$ contains only variables of the fixed blocks. Then, the new constraint becomes a secondary constraint, and the new dependent variable a can be any variable of f'. Since the constraint $(f'|b')$ contains only variables of fixed blocks (but no marked variable), we can assign a value to all variables of $(f'|b')$ given by α_v. Thus, there is $\bar{b} \in \{0,1\}$ such that for any extension y of α_v, it holds that $\langle f', y\rangle = \bar{b}$. Then, we pick the appropriate child u of v, that gives us the right value of b (and b') such that the new constraint $(f'|b')$ holds for any extension of α_v. This ensures that Invariant 4 holds for u.

We did not query any new bit of x in this case. It follows that the partial assignment ρ_v and the set of fixed and free blocks are not changed. The set of primary constraints is unchanged as well. Further, the set of pivots of $(M_v|z_v)$ is not changed by the pivoting step of the Gaussian elimination. Thus, Invariant 3 holds for u. Invariant 2 holds because the constraint $(f'|b')$ does not contain any free variable of $(M_v|z_v)$ and thus the new dependent variable a can not be from a free block.

Case 2: The new constraint $(f'|b')$ contains at least one variable a of a free block B_i. In this case, we can pick the new vertex u as an arbitrary child of v. Let $\mathcal{T}_w$ be a subtree of $\mathcal{T}$ rooted in a node w of $\mathcal{T}$. We compare the query size of subtrees $\mathcal{T}_{u_0}$ and $\mathcal{T}_{u_1}$ and we pick u to be the root of the subtree with the smaller query size, i.e., $|\mathcal{T}_u|_{\mathsf{q}} \leq |\mathcal{T}_w|_{\mathsf{q}}$, where w is the other children of v.

We query x_i and update the partial assignment ρ_v by the value of x_i to get ρ_u. The block B_i becomes a fixed block. The new constraint $(f'|b')$ becomes the primary constraint of B_i and the variable $a \in B_i$ becomes the pivot of $(f'|b')$, i.e., a becomes a marked variable. Since the set of pivots of $(M_v|z_v)$ is not changed, Invariant 3 holds for u. Since the only new dependant variable is $a \in B_i$, Invariant 2 holds as well.

The partial assignment α_u differs only at the block B_i from α_v ($\alpha_v[B_i] = *^\ell$, and $\alpha_u[B_i] = \delta^{j,x_i}$). Since the block B_i was free in $(M_v|z_v)$, no secondary constraint of $(M_v|z_v)$ contains any variable of the block B_i. Thus, no secondary constraints of $(M_v|z_v)$ were changed by the pivoting step in this case. The new constraint $(f'|b')$ is primary. Thus, no secondary constraint of $(M_u|z_u)$ contains any variable of the block B_i as well. Therefore, any extension of α_u still satisfies all secondary constraints of $(M_u|z_u)$ and Invariant 4 holds for α_u.

See Algorithm 2 for a summary of the query node processing.

Forget Nodes

In the case when v is a forget node, the node v has the only child u and $\dim(A_u) = \dim(A_v)+1$. We have the constraint representation $(M_v|z_v)$ of A_v maintained by our simulation for $M_v \in \mathbb{F}_2^{c\times m}$ and $z_v \in \mathbb{F}_2^c$. For processing the forget node, we introduce a classification of stifling variables. The variables of fixed blocks that are contained in the secondary constraints are called dangerous. Note that the marked variables can not be dangerous. The remaining variables of fixed blocks (i.e., non-marked and non-dangerous) are called safe. Thus, with this new classification, the matrix M_v has the following form:

<table>
<tr><td>Linear algebra classification:</td><td>Dependent</td><td colspan="3">Independent</td><td></td><td></td></tr>
<tr><td rowspan="2">$M_v =$</td><td>I_{d_1}</td><td>0</td><td>D</td><td>S</td><td>F</td><td>primary constraints</td></tr>
<tr><td>0</td><td>I_{d_2}</td><td>E</td><td>0</td><td>0</td><td>secondary constraints</td></tr>
<tr><td rowspan="2">Simulation classification:</td><td rowspan="2">Marked</td><td colspan="2">Dangerous</td><td>Safe</td><td rowspan="2">Free</td><td rowspan="2">variables</td></tr>
<tr><td colspan="3">Stifling</td></tr>
<tr><td></td><td colspan="4">Fixed</td><td>Free</td><td>blocks</td></tr>
</table>

Algorithm 2

Process Query Node (v: the current (query) node):

$(f', b') \leftarrow$ reduced form of the constraint (f_v, b) ▷ b' is a parameter that will be set later
if f' does not contain a free variable **then**
  $a \leftarrow$ arbitrary variable of f'
  $u \leftarrow$ the child of v where α_v satisfy the new constraint $(f'|b')$ ▷ $(f'|b')$ is a secondary constraint
  $\rho_u \leftarrow \rho_v$ ▷ The sets of primary constraints, fixed blocks, and marked variables are not changed
else ▷ f' contains a free variables
  $a \leftarrow$ arbitrary free variable in f'
  $u \leftarrow$ a child of v such that $\mathcal{T}_u$ has smaller query size
  $\rho_u \leftarrow \rho_v, \rho_u[i] \leftarrow$ query x_i ▷ B_i is the block of a ▷ $(f'|b')$ is the primary constraint of the newly fixed block B_i, a is a marked variable
Set b' that the constraint $(f'|b')$ is satisfied by all elements of A_u
$(M_u|z_u) \leftarrow$ add the constraint $(f'|b')$ to the system $(M_v|z_v)$ and change it to the echelon form by pivoting a
$v \leftarrow u$ ▷ New current node

Let $t \in \mathbb{F}_2^c$ be a forgetting vector given by an application of Theorem 10 to spaces A_v and A_u. The new system $(M_u|z_u)$ is obtained after applying $\mathsf{Del}\big((M_v|z_v), t, i\big)$ for a right choice of i (the function Del is defined in Section 6). By Theorem 10, the system $(M_u|z_u)$ is a constraint representation of A_u in the echelon form, i.e. Invariant 1 holds for u. Let p be the number of primary constraints in $(M_v|z_v)$, i.e. wlog, the constraints $1, \dots, p$ are primary and the constraints $p+1, \dots, c$ are secondary. We consider two cases.

Case 1: There is an $i \in \{p+1, \dots, c\}$ such that $t_i = 1$. Then, fix one such i and take a system $(M_u|z_u) = \mathsf{Del}\big((M_v|z_v), t, i\big)$. We do not query or forget any bit of x, thus $\rho_u = \rho_v$ and $\alpha_u = \alpha_v$. To create $(M_u|z_u)$, we only added the secondary constraint i to some rows of $(M_v|z_v)$ and then we deleted it. Thus, the set of variables which appear in secondary constraints cannot grow in size and, therefore, secondary constraints are still satisfied by the assignment α_u. Therefore, Invariant 4 holds for α_u.

The set of primary constraints is not changed. The Del operation does not change the set of marked variables as the secondary constraint i does not contain any pivot of the primary constraints. Thus, Invariants 3 holds for u. The set of fixed blocks does not change and there is no new dependant variable. Thus, Invariant 2 holds as well.

Case 2: For all $i \in \{p+1, \dots, c\}$, it holds that $t_i = 0$. Then, we fix some $i \in \{1, \dots, p\}$ such that $t_i = 1$. Note that such an i exists as t is a non-zero vector. Again, let $(M_u|z_u) = \mathsf{Del}\big((M_v|z_v), t, i\big)$. Since t has only zeroes at the coordinates corresponding to the secondary constraints, the secondary constraints are not changed by the Del operation. As the constraint i is deleted and it was a primary one, one marked variable a (the pivot of the constraint i) becomes independent and safe. Let B_j be the block containing the variable a, i.e., the Constraint i of $(M_v|z_v)$ is the primary constraint for B_j. We consider two sub-cases.

Sub-case 2.1: The other variables of B_j are safe in $(M_u|z_u)$ as well, i.e., they are not in any secondary constraint. Thus, the whole block B_j contains only independent and safe variables of $(M_u|z_u)$. We forget the bit x_j and make the block B_j free. The set of other primary constraints (different from i) may change their form, but their pivots are not changed. Hence, Invariant 3 holds for u. There is no new dependant variable. Thus Invariant 2 holds as well.

We get the partial assignment ρ_u by simply setting the variable x_j free. The partial assignment α_u differs from α_v only at the block B_j ($\alpha_u[B_j] = *^\ell$, and $\alpha_v[B_j] = \delta^{j,x_j}$). Further, the secondary constraints of $(M_u|z_u)$ do not contain any variable of the block B_j by the assumption. Thus, Invariant 4 holds for α_u.

Sub-case 2.2: There is a dangerous variable in the block B_j, i.e., there is a secondary constraint of $(M_u|z_u)$ that contains a variable of B_j. In this case, we use the strong stifling property of g. Let $D \subseteq [\ell]$ be the set of indices of all dangerous variables of $(M_u|z_u)$ in B_j. Let j_1 be the index of the variable a in B_j (i.e., the previously marked variable in B_j). Note that $j_1 \notin D$ because the variable a is safe. Thus by definition, there is a $j_2 \in D$ such that $\delta^{j_2,x_j}\big[D \setminus \{j_2\}\big] = \delta^{j_1,x_j}\big[D \setminus \{j_2\}\big]$ (note that $\alpha_v[B_j] = \delta^{j_1,x_j}$). Let a' be the j_2-th variable in the block B_j and k be a secondary constraint that contains a' (such constraint exists by the assumption).

We run again the pivoting step for a', i.e., we zero out all coefficients corresponding to a' in all other constraints of $(M_u|z_u)$ by adding the constraints k to all other constraints containing a'. We denote the final system of constraints as $(M'_u|z'_u)$. Note that $(M'_u|z'_u)$ is still a constraint representation of A_u as it arises from $(M_u|z_u)$ only by row operations.

The constraint k is now the only constraint containing the variable a' and a' becomes a dependent variable. Thus, we make the constraint k a primary constraint for B_j and we mark the variable a'. The primary constraint for B_j was changed from the constraint i of $(M_v|z_v)$ to the constraint k of $(M'_u|z'_u)$ and the marked variable in the block B_j was changed from a to a'. The set of other primary constraints and their pivots were not changed. Thus, Invariant 3 holds for u.

We do not change the assignment ρ_v, thus the sets of free and fixed blocks are the same. The only change in the set of dependent variables was done in the block B_j (that remains a fixed block), thus Invariant 2 holds for u.

The secondary constraints of $(M_v|z_v)$ were not changed by the $\mathsf{Del}\big((M_v|z_v), t, i\big)$ executed at the beginning of this case (as $t_{i'} = 0$ for all secondary constraints i'). Since k is a secondary constraint of $(M_u|z_u)$, the secondary constraints of $(M'_u|z'_u)$ contains only variables of fixed blocks. However, we change the marked variable in the block B_j. Thus, the partial assignment α_u differs from α_v at the block B_j ($\alpha_v = \delta^{j_1,x_i}$, and $\alpha_u = \delta^{j_2,x_i}$, where j_1 and j_2 are indices of a and a' in the block B_j). We need to be sure that α_u still gives a solution to the secondary constraints of $(M'_u|z'_u)$. Note that the secondary constraints of $(M'_u|z'_u)$ might still contain variables from the block B_j.

By pivoting a' and making the constraint k primary, the variable a' is not in any secondary constraint of $(M'_u|z'_u)$. Since k was a secondary constraint of $(M_u|z_u)$, it can not happen that a safe variable in $(M_u|z_u)$ would become a dangerous one in $(M'_u|z'_u)$ (i.e., by the pivoting of a'). In other words, the set of variables of the secondary constraints of $(M'_u|z'_u)$ is a subset of the set of variables of the secondary constraints of $(M_u|z_u)$. Thus, the set $D \setminus \{j_2\}$ still contains all dangerous variables of B_j in $(M'_u|z'_u)$. Since $\alpha_v\big[D \setminus \{j_2\}\big] = \alpha_u\big[D \setminus \{j_2\}\big]$ by the assumption, any extension α_u satisfy all secondary constraints of $(M'_u|z'_u)$ and Invariant 4 holds for α_u.

A summary of the forget node processing is in Algorithm 3.

Proof of Theorem 13

Theorem 13 follows from the following lemma.

Algorithm 3

Process Forget Node (v: the current (forget) node):

```
t ← forgetting vector given by Theorem 10
u ← the only child of v
if t contains a secondary constraint then
    i ← arbitrary secondary constraint in t
    (M_u|z_u) ← Del((M_v|z_v), t, i)                ▷ The constraint i is now removed
    ρ_u = ρ_v
        ▷ The sets of primary constraints, fixed blocks, and marked variables are not changed
else                                                ▷ t contains only primary constraints
    i ← arbitrary primary constraint in t
    (M_u|z_u) ← Del((M_v|z_v), t, i)                ▷ The constraint i is now removed
    a ← the marked variable of i
    B_j ← the block of the variable a     ▷ i is the primary constraint of B_j in (M_v|z_v)
    j_1 ← the index of a in B_j
    if the variables B_j \ {j_1} are safe in (M_u|z_u) then
        forget x_j                                  ▷ B_j is a new free block
    else                           ▷ B_i contains a dangerous variable of (M_u|z_u)
        D ← indices of all dangerous variables of (M_u|z_u) in B_j
        j_2 ∈ D \ {j_1} by Definition 7
        a' ← the j_2-th variable of B_j
        k ← a secondary constraint of (M_u|z_u) containing a'
        (M_u|z_u) ← pivoting a' in (M_u|z_u) by adding the constraint k to other constraints
            ▷ k is the new primary constraint of B_j, a' is the new marked variable of B_j
            ▷ a is a new safe (and thus independent) variable
v ← u
```

▶ **Lemma 14.** *Suppose the simulation is at Line 4 of Algorithm 1, i.e., it checks whether the current node v is a leaf. Then,*

1. $w\big(C(\rho_v)\big) \leq \text{co-dim}(A_v)$.
2. *For any $\bar{x} \in C(\rho^v)$, there is $y \in A_v$ such that $\overrightarrow{g}(y) = x$.*

Proof of Item 1. By Invariant 1, the co-dimension of A_v is exactly the number of equations in the system $(M_v|z_v)$. By Invariant 3, the number of fixed bits by ρ_v is exactly the number of primary constraints in $(M_v|z_v)$. Thus, $w\big(C(\rho_v)\big) \leq \text{co-dim}(A_v)$. ◀

Proof of Item 2. Let $\bar{x} \in C(\rho_v)$. We will find a solution y to the system $(M_v|z_v)$ such that $\overrightarrow{g}(y) = \bar{x}$. Thus, by Invariant 1, $y \in A_v$.

First, we set variables of free blocks. Let B_i be a free block. Thus, by Invariant 2, all variables of B_i are independent. We set the variables of B_i in a way such that the block B_i is mapped to $\bar{x}_i$ by the gadget g.

Now, we set the values of the stifling variables according to α_v. By Invariant 4, all secondary constraints are satisfied by any extension of α_v. Recall that for a fixed block B_i, $\alpha_v[B_i] = \delta^{j,x_i}$ where j is the index of the marked variable of B_i and $\bar{x}_i = \rho_v[i]$. Since δ^{j,x_i} is a (j, x_i)-stifling pattern, it holds that the block B_i will be always mapped to $\bar{x}_i$ by g, no matter how we set the marked variables. Thus, the constructed solution y will be mapped onto $\bar{x}$. By Invariant 3, each primary constraint contains a unique marked variable. Thus, we can set a value to each marked variable a in such a way the primary constraint containing a is satisfied. ◀

Proof of Theorem 13. By Item 1 of Lemma 14, the width of a cube $C(\rho_v)$ in a time of checking whether a vertex v is a leaf is at most co-dimension of A_v. Thus, the width of the constructed FDT $\mathcal{T}'$ is at most the width of $\mathcal{T}$.

Now, we bound the query depth of $\mathcal{T}'$. Consider a root-leaf path P of $\mathcal{T}'$ and let d be the number of queries made on P. Note that any time we query a bit of x (Line 9 of Algorithm 2) we also pick a subtree with a smaller query size (Line 8 of Algorithm 2). Thus, by each query of $\mathcal{T}'$ we halve the query size of $\mathcal{T}$. Thus, $2^d \leq |\mathcal{T}|_{\mathsf{q}}$.

It remains to prove the constructed FDT $\mathcal{T}'$ is correct. Let s be a leaf of $\mathcal{T}$ that is reached during the simulation and $o \in O$ is the output of s. Since $\mathcal{T}$ computes $R \circ g$, it holds that for all $y \in A_s$ we have $(y, o) \in R \circ g$. Note that the processing phase (Lines 5-8 of Algorithm 1) is not executed for any leaf. Thus, the assertion of Lemma 14 holds for the leaf s even at the time of output – Line 9 of Algorithm 1. Therefore at the end of the simulation, for any $\bar{x} \in C(\rho_s)$ there is $y \in A_s$ such that $\overrightarrow{g}(y) = \bar{x}$. Since $(y, o) \in R \circ g$, it holds that $(\bar{x}, o) \in R$ and the constructed FDT $\mathcal{T}'$ indeed outputs a correct answer. ◀

8 Application

Razborov [24] showed the following trade-off between width and size of tree-like resolution.

▶ **Theorem 15** (Theorem 3.1, Razborov [24])**.** *Let $k = k(n) \geq 4$ be any parameter and let $\varepsilon > 0$ be an arbitrary constant. Then, there exists a k-CNF contradiction τ' over n variables such that there is a resolution refutation for τ' with width at most $O(k)$, but for any tree-like resolution refutation Π for τ' with $w(\Pi) \leq n^{1-\varepsilon}/k$, we have the bound $|\Pi| \geq \exp(n^{\Omega(k)})$.*

By our simulation, given by Theorem 13, we can lift the trade-off (given by the previous theorem) to tree-like Res($\oplus$) and prove Theorem 3.

▶ **Theorem 3.** *Let $k = k(n) \geq 12$ be any parameter and let $\varepsilon > 0$ be an arbitrary constant. Then, there exists a k-CNF contradiction τ over n variables such that there is a resolution refutation for τ with width at most $O(k)$, but for every tree-like Res($\oplus$) refutation Π for τ with $w(\Pi) \leq n^{1-\varepsilon}/k$, we have the bound $|\Pi| \geq \exp(n^{\Omega(k)})$.*

Proof. Let $g : \{0,1\}^3 \to \{0,1\}$ be a strongly stifled gadget – such functions exist as observed in Section 5. Let $k' := \lfloor k/3 \rfloor$, and τ' be a k'-CNF contradiction given by Theorem 15. We set $\tau := \tau' \circ g$ that is a k-CNF contradiction. Since there is a resolution refutation for τ' with width at most $O(k')$, then there is a resolution refutation for τ with width at most $O(k)$.

Now, let Π be a tree-like Res($\oplus$) refutation for τ with $w(\Pi) \leq n^{1-\varepsilon}/k$. By Observation 4, let $\mathcal{T}$ be a canonical FPDT corresponding to Π that computes *Search*(τ). Thus, we have $w(\mathcal{T}) \leq w(\Pi) + 1$ and $|\mathcal{T}|_q \leq |\Pi|$. We change $\mathcal{T}$ to compute *Search*(τ') $\circ g$. Let s be a leaf of $\mathcal{T}$ outputting a clause D of $\tau' \circ g$. The clause D has to appear in a set of clauses $D' \circ g$ for a clause D' of τ'. We change the output of s to be the clause D' instead of D. By Observation 6, the tree $\mathcal{T}$ now computes *Search*(τ') $\circ g$.

By Theorem 13, there is FDT $\mathcal{T}'$ computing *Search*(τ') with $d_{\mathsf{q}}(\mathcal{T}') \leq \log |\mathcal{T}|_{\mathsf{q}}$ and $w(\mathcal{T}') \leq w(\mathcal{T})$. Let Π' be the resolution refutation for τ' corresponding to the succinct form of $\mathcal{T}'$. Thus, $w(\Pi') = w(\mathcal{T}')$ and $|\Pi'| \leq 3 \cdot 2^{d_{\mathsf{q}}(\mathcal{T}')}$ (by Observation 5). Since

$$w(\mathcal{T}') \leq w(\mathcal{T}) \leq n^{1-\varepsilon}/k + 1 \leq n^{1-\varepsilon}/k', \tag{1}$$

we have that $|\Pi'| \geq \exp(n^{\Omega(k')})$ by Theorem 15. The last inequality in (1) holds if $k \leq 2n^{1-\varepsilon}$, which holds as we suppose that $1 \leq w(\Pi) \leq n^{1-\varepsilon}/k$. Putting everything together, we have

$$|\Pi| \geq |\mathcal{T}|_{\mathsf{q}} \geq 2^{d_{\mathsf{q}}(\mathcal{T}')} \geq \frac{1}{3} \cdot |\Pi'| \geq \exp(n^{\Omega(k')}) = \exp(n^{\Omega(k)}).$$

◀

Acknowledgment

We would like to thank an anonymous reviewer for pointing out the short and elegant proof of Thoerem 10 that we include here. Our original proof was complicated.

References

1. Yaroslav Alekseev and Dmitry Itsykson. Lifting to regular resolution over parities via games. *Electron. Colloquium Comput. Complex.*, TR24-128, 2024. URL: `https://eccc.weizmann.ac.il/report/2024/128/`, `arXiv:TR24-128`.
2. Sepehr Assadi, Gillat Kol, and Zhijun Zhang. Rounds vs communication tradeoffs for maximal independent sets. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 1193–1204. IEEE, 2022. URL: `https://doi.org/10.1109/FOCS54457.2022.00115`.
3. Paul Beame, Christopher Beck, and Russell Impagliazzo. Time-space tradeoffs in resolution: superpolynomial lower bounds for superlinear space. In Howard J. Karloff and Toniann Pitassi, editors, *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 213–232. ACM, 2012. URL: `https://doi.org/10.1145/2213977.2213999`.
4. Paul Beame and Sajin Koroth. On disperser/lifting properties of the index and inner-product functions. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPIcs*, pages 14:1–14:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. URL: `https://doi.org/10.4230/LIPIcs.ITCS.2023.14`, `doi:10.4230/LIPICS.ITCS.2023.14`.
5. Chris Beck, Jakob Nordström, and Bangsheng Tang. Some trade-off results for polynomial calculus: extended abstract. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 813–822. ACM, 2013. URL: `https://doi.org/10.1145/2488608.2488711`.
6. Eli Ben-Sasson. Size space tradeoffs for resolution. In John H. Reif, editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 457–464. ACM, 2002. URL: `https://doi.org/10.1145/509907.509975`.
7. Christoph Berkholz, Moritz Lichter, and Harry Vinall-Smeeth. Supercritical size-width tree-like resolution trade-offs for graph isomorphism, 2024. URL: `https://arxiv.org/abs/2407.17947`, `arXiv:2407.17947`.
8. Christoph Berkholz and Jakob Nordström. Supercritical space-width trade-offs for resolution. *SIAM J. Comput.*, 49(1):98–118, 2020. `doi:10.1137/16M1109072`.
9. Sreejata Kishor Bhattacharya, Arkadev Chattopadhyay, and Pavel Dvořák. Exponential Separation Between Powers of Regular and General Resolution over Parities. In Rahul Santhanam, editor, *39th Computational Complexity Conference (CCC 2024)*, volume 300 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 23:1–23:32, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: `https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.CCC.2024.23`, `doi:10.4230/LIPIcs.CCC.2024.23`.
10. Mark Braverman and Rotem Oshman. A rounds vs. communication tradeoff for multi-party set disjointness. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 144–155. IEEE Computer Society, 2017. URL: `https://doi.org/10.1109/FOCS.2017.22`.

11 Arkadev Chattopadhyay, Nikhil S. Mande, Swagato Sanyal, and Suhail Sherif. Lifting to parity decision trees via stifling. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPIcs*, pages 33:1–33:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. URL: https://doi.org/10.4230/LIPIcs.ITCS.2023.33, doi:10.4230/LIPICS.ITCS.2023.33.

12 Susanna F. de Rezende, Noah Fleming, Duri Andrea Janett, Jakob Nordström, and Shuo Pang. Truly supercritical trade-offs for resolution, cutting planes, monotone circuits, and weisfeiler-leman, 2024. URL: https://arxiv.org/abs/2411.14267, arXiv:2411.14267.

13 Klim Efremenko, Michal Garlík, and Dmitry Itsykson. Lower bounds for regular resolution over parities. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 640–651. ACM, 2024. doi:10.1145/3618260.3649652.

14 Noah Fleming, Toniann Pitassi, and Robert Robere. Extremely Deep Proofs. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*, volume 215 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 70:1–70:23, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ITCS.2022.70, doi:10.4230/LIPIcs.ITCS.2022.70.

15 Lance Fortnow. Time-space tradeoffs for satisfiability. *J. Comput. Syst. Sci.*, 60(2):337–353, 2000. URL: https://doi.org/10.1006/jcss.1999.1671.

16 Lance Fortnow, Richard J. Lipton, Dieter van Melkebeek, and Anastasios Viglas. Time-space lower bounds for satisfiability. *J. ACM*, 52(6):835–865, 2005. URL: https://doi.org/10.1145/1101821.1101822.

17 Ankit Garg, Mika Göös, Pritish Kamath, and Dmitry Sokolov. Monotone circuit lower bounds from resolution. *Theory Comput.*, 16:1–30, 2020. Preliminary version in STOC 2018. URL: https://doi.org/10.4086/toc.2020.v016a013, doi:10.4086/TOC.2020.V016A013.

18 Mika Göös, Gilbert Maystre, Kilian Risse, and Dmitry Sokolov. Supercritical tradeoffs for monotone circuits, 2024. URL: https://arxiv.org/abs/2411.14268, arXiv:2411.14268.

19 Dmitry Itsykson and Dmitry Sokolov. Lower bounds for splittings by linear combinations. In Erzsébet Csuhaj-Varjú, Martin Dietzfelbinger, and Zoltán Ésik, editors, *Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part II*, volume 8635 of *Lecture Notes in Computer Science*, pages 372–383. Springer, 2014. doi:10.1007/978-3-662-44465-8_32.

20 Dmitry Itsykson and Dmitry Sokolov. Resolution over linear equations modulo two. *Ann. Pure Appl. Log.*, 171(1), 2020. URL: https://doi.org/10.1016/j.apal.2019.102722.

21 Noam Nisan and Avi Wigderson. Rounds in communication complexity revisited. *SIAM J. Comput.*, 22(1):211–219, 1993. URL: https://doi.org/10.1137/0222016.

22 Pavel Pudlák. On extracting computations from propositional proofs (a survey). In Kamal Lodaya and Meena Mahajan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, December 15-18, 2010, Chennai, India*, volume 8 of *LIPIcs*, pages 30–41. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010. URL: https://doi.org/10.4230/LIPIcs.FSTTCS.2010.30.

23 Alexander A. Razborov. Unprovability of lower bounds on circuit size in certain fragments of bounded-arithmetic. *Izvestiya. Math.*, 59(1):205–227, 1995.

24 Alexander A. Razborov. A new kind of tradeoffs in propositional proof complexity. *J. ACM*, 63(2):16:1–16:14, 2016. URL: https://doi.org/10.1145/2858790.

25 Dmitry Sokolov. Dag-like communication and its applications. In Pascal Weil, editor, *Computer Science - Theory and Applications - 12th International Computer Science Symposium in Russia, CSR 2017, Kazan, Russia, June 8-12, 2017, Proceedings*, volume 10304 of *Lecture Notes in Computer Science*, pages 294–307. Springer, 2017. URL: https://doi.org/10.1007/978-3-319-58747-9_26.

26 R. Ryan Williams. Time-space tradeoffs for counting NP solutions modulo integers. *Comput. Complex.*, 17(2):179–219, 2008. URL: https://doi.org/10.1007/s00037-008-0248-y.

A Appendix

In this section, we show that the functions IND_1 and MAJ_3 are strongly stifled and IP_2 is not strongly stifled.

▶ **Observation 8.** *The functions* IND_1 *and* MAJ_3 *are strongly stifled.*

Proof. We present collections of (i, b)-stifling patterns $P(\mathsf{IND}_1)$ and $P(\mathsf{MAJ}_3)$ for IND_1 and MAJ_3, respectively. It is straight-forward to verify that these collections are converting collections of stifling patterns for IND_1 and MAJ_3.

i \ b	0	1
1	$(*, 0, 0)$	$(*, 1, 1)$
2	$(1, *, 0)$	$(1, *, 1)$
3	$(0, 0, *)$	$(0, 1, *)$

Table 1 $P(\mathsf{IND}_1)$

i \ b	0	1
1	$(*, 0, 0)$	$(*, 1, 1)$
2	$(0, *, 0)$	$(1, *, 1)$
3	$(0, 0, *)$	$(1, 1, *)$

Table 2 $P(\mathsf{MAJ}_3)$

◀

▶ **Observation 9.** *The function* IP_2 *is not strongly stifled.*

Proof. The only $(1, 1)$-stifling pattern for $\mathsf{IP}_2 : \{0, 1\}^4 \to \{0, 1\}$ is $\delta^1 := (*, 1, 0, 1)$. Similarly, the only $(2, 1)$- and $(4, 1)$-stifling patterns for IP_2 are $\delta^2 := (1, *, 1, 0)$, and $\delta^4 := (1, 0, 1, *)$, respectively. Now, let $D = \{2, 4\}$. There is no $j \in D$ such that $\delta^1[D \setminus \{j\}] = \delta^j[D \setminus \{j\}]$ as required to be a strongly stifled function. ◀

ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il