Electronic Colloquium on Computational Complexity, 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 [22] 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 → 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 [14, 15, 24], rounds-communication trade-offs [19, 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 wrt to the other resource shooting up to the cost needed by a naive/generic algorithm.

In 2016, Razborov [22] 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 narrow 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 narrow refutation 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 rare in

ISSN 1433-8092

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 very recent work of Berkholz, Lichter and Vinall-Smeeth [7] proves super-critical trade-offs for resolution width and tree-like size for refuting the isomorphism of two graphs[1].

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 [13] 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. Our work also falls in this second type of trade-offs.

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 [17, 18], 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 [12, 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 [18] 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 $\Phi$ 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 $\Phi$ 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

[1] This work appears to have improved Razborov's work in the following manner. Their size lower bound is exponential in the size of the formula whereas this was not so for Razborov.

of $\Phi \circ g$ in Res($\oplus$) of size $s$ and width $w$, would have extracted a tree-like refutation of $\Phi$ 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 [22], 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 $\tau$ over $n$ variables such that there is a resolution refutation for $\tau$ with width at most $O(k)$, but for every tree-like Res($\oplus$) refutation $\Pi$ for $\tau$ with $w(\Pi) \leq n^{1-\varepsilon}/k$, we have the bound $|\Pi| \geq \exp(n^{\Omega(k)})$.*

The contradiction $\tau$ from the previous theorem is a lift of the contradiction $\tau'$ constructed by Razborov [22] by an appropriate gadget $g : \{0,1\}^{\ell} \to \{0,1\}$ with a constant size. A caveat of $\tau'$ (as Razborov also noted) is that the number of clauses of $\tau'$ is $n^{\Theta(k)}$. Naturally, this caveat is inherited by our contradiction $\tau$. However, since Theorem 3 is proved via a lifting theorem using a gadget of constant size, if one were to construct a formula $\tilde{\tau}$ with significantly fewer clauses while yielding a similar trade-off for resolution, then Theorem 3 would be immediately improved – the number of clauses of $\tilde{\tau} \circ g$ would also be reduced.

### Relation to Other Recent Works

The Res($\oplus$) proof system has been an active area of research. Recently, Efremenko, Garlík, and Itsykson [12] 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 $\Phi$ with width $w$, then there is no width-$w$ Res($\oplus$) refutation of a lift of $\Phi$ 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 $\Phi$ 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. The linear algebraic machinery we develop to process forget nodes appears novel and is our main technical contribution. 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 wrt 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 $A_1 \subseteq A_2$. Our technical centre-piece is to combine the primal and dual views to arrive at a characterization, via Theorem 13, of the constraint space of $A_2$ in terms of the constraint space of $A_1$, where $A_1 \subseteq A_2 \subseteq \mathbb{F}_2^n$. 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.

We feel that this machinery should find use in tackling other open problems at the interface of linear algebra and computational complexity.

## 2 Tree-like Proofs and Decision Trees

A proof in a propositional proof system starts from a set of clauses $\Phi$, 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 $\Phi$ then it proves the set $\Phi$ 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 $A \vee B$.

**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 $\Pi$ of a formula $\Phi$ is the number of applications of the rules above in order to refute the formula $\Phi$. The width $w(\Pi)$ of a resolution (or Res($\oplus$)) refutation $\Pi$ 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 of the tree-like proof $\Pi$ is the depth of the underlying tree (i.e., the length of the longest path from the root to a leaf).

It is known that a tree-like resolution (or Res($\oplus$)) proof, for an unsatisfiable set of clauses $\Phi$, corresponds to a (parity) decision tree for a search problem $Search(\Phi)$ that is defined as follows. For a given assignment $\alpha$ of the $n$ variables of $\Phi$, one needs to find a clause in $\Phi$ that

is not satisfied by $\alpha$ (at least one exists as the set $\Phi$ is unsatisfiable). The correspondence holds even for general (not only tree-like) proofs (see for example Garg et al. [16], who credit it to earlier work of Razborov [21] that was simplified by Pudlák [20] and Sokolov[23]), 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 $w$, it holds that $A_v = A_u \cup A_w$. Such nodes are called *query nodes*.
- Every node $v$ with exactly one child $u$ is called a *forget node*. It holds that $A_v \subseteq A_u$.
- Each leaf $\ell$ 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 $d$ such that there exists an affine space of co-dimension at least $d$ associated with some node of $\mathcal{T}$. 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.

By properties of affine spaces, it holds that for every query node $v$ with children $u$ and $w$, 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 $v$.

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

The correspondence between a F(P)DT's and tree-like resolution (or Res($\oplus$)) proofs 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 $\Phi$ and the root is associated with the empty clause. Each node with two children corresponds to an application of the 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*($\Phi$) 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). Therefore, from a tree-like resolution (or Res($\oplus$)) refutation $\Pi$ for $\Phi$ we can get an F(P)DT $\mathcal{T}$ for *Search*($\Phi$) and vice versa. 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 Res($\oplus$) refutation $\Pi$ 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 Res($\oplus$) rules).

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 co-dim$(A_v)$ = 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 $\mathcal{T}$ be a succinct FPDT and $\Pi$ be the tree-like Res($\oplus$) refutation corresponding to $\mathcal{T}$. Then, $|\Pi| \leq 3 \cdot 2^{d_q(\mathcal{T})}$.*

**Proof.** As mentioned above, the length of $\Pi$ equals the number of inner nodes of $\mathcal{T}$. The tree $\mathcal{T}$ has at most $2^{d_q(\mathcal{T})} - 1$ 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, $|\Pi| \leq 3 \cdot 2^{d_q(\mathcal{T})}$. ◀

## 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

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

where $\overrightarrow{g}(y_1^1, \ldots, y_\ell^1, \ldots, y_1^n, \ldots, y_\ell^n) = \left(g(y_1^1, \ldots, y_\ell^1), \ldots, g(y_1^n, \ldots, y_\ell^n)\right)$.

For CNF $\Phi$ over $n$ variables $\{x_1, \ldots, x_n\}$, let $\Phi \circ g$ be the following lift of $\Phi$ over the variables $\{y_j^i \mid i \in [n], j \in [\ell]\}$. For any clause $D$ of $\Phi$, 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,

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

where $y_j^i \neq \kappa_j^i$ is the following literal:

$$y_j^i \neq \kappa_j^i = \begin{cases} y_j^i & \text{if } \kappa_j^i = 0, \\ \neg y_j^i & \text{if } \kappa_j^i = 1. \end{cases}$$

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

▶ **Observation 5.** *For a clause $D$, an assignment $\delta$ of $Vars(D \circ g)$ falsifies $D \circ g$ if and only if $\overrightarrow{g}(\delta) = \eta_D$, i.e., $\overrightarrow{g}(\delta)$ falsifies $D$.*

## 4 Notation

Let $S$ be a set. We present a notation that we use for matrices and vectors with entries from the set[2] $S$. For a matrix $M \in S^{q \times m}$ and $j \in [m]$, $M[*; j]$ is a vector in $S^q$ corresponding to the $j$-th column of $M$. For a sequence of coordinates $r_1, \ldots, r_k \in [q]$, we denote $M[r_1, \ldots, r_k; j]$ the projection of the $j$-th column of $M$ onto the coordinates $r_1, \ldots, r_k$, i.e., $M[r_1, \ldots, r_k; j] = \left(M[r_1; j], \ldots, M[r_k; j]\right) \in S^k$. Analogously, we use this notation also for rows and vectors, e.g., $M[i; *] \in S^m$ is the vector corresponding to the $i$-th row of $M$, and for a vector $u \in S^q$, $u[r_1, \ldots, r_k]$ is the projection of $u$ onto the coordinates $r_1, \ldots, r_k$, etc. For a projection on all but one coordinate $i \in [q]$, we use the notation $M[-i; j]$ or $u[-i]$ for brevity. I.e., $M[-i; j] := \left(M[1; j], \ldots, M[i-1; j], M[i+1; j], \ldots, M[q; j]\right) \in S^{q-1}$ is the $j$-th column of $M$ without the $i$-th entry, and similarly $u[-i] := \left(u[1], \ldots, u[i-1], u[i+1], \ldots, u[q]\right) \in S^{q-1}$ is the vector $u$ without the $i$-th entry. For indexing single entries of vectors or matrices we also use subscripts – i.e. $v_i := v[i]$ and $M_{i,j} := M[i; j]$.

[2] In this paper, the set $S$ will be either the field $\mathbb{F}_2$ or $\{0, 1, *\}$.

## 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, $\delta$ assigns a value to all but the $i$-th bit and when we extend $\delta$ to a full assignment $\gamma$, it holds that $g(\gamma) = b$ no matter how we set the value of the $i$-th bit.

▶ **Definition 6.** *A Boolean function $g : \{0,1\}^\ell \to \{0,1\}$ is* strongly stifled *if there is a collection $P := \{\delta^{i,b} | i \in [\ell], b \in \{0,1\}\}$ where each $\delta^{i,b}$ is an $(i,b)$-stifling pattern for $g$ and*

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

*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 7.** *The functions $\mathsf{IND}_1$ and $\mathsf{MAJ}_3$ 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 8.** *The function $\mathsf{IP}_2$ 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.

It turns out that transforming a constraint representation into a basis representation and back is useful for processing of the forget nodes. Hence, we exhibit several properties of a basis representation of affine spaces. Each affine space $A \subseteq \mathbb{F}_2^m$ is a shift of a vector space $V \subseteq \mathbb{F}_2^m$ by a shift vector $s \in \mathbb{F}_2^m$, i.e. $A = V + s = \{v + s \mid v \in V\}$. Consider a basis of $V$ of dimension $d$ and arrange the basis vectors as columns of a matrix $B \in \mathbb{F}_2^{m \times d}$. Thus, the vector space $V$ is the column space of $B$ that we denote $\mathcal{C}(B)$. We say $[B, s]$ is a *basis representation* of the affine space $A$.

▶ **Observation 9.** *Let $[B, s]$ be a basis representation of an affine space $A \subseteq \mathbb{F}_2^m$. Let $b \in \mathbb{F}_2^m$ be a column in $B$ and $B'$ be a matrix arising from $B$ by adding the vector $b$ to a different column of $B$. Let $s' := s + b$. Then, $[B', s]$ and $[B, s']$ are basis representations of $A$.*

A matrix $B \in \mathbb{F}_2^{m \times d}$, where $m \geq d$, is in a *canonical form* if there are $d$ rows $r_1 < r_2 < \cdots < r_d \in [m]$ such that for all $i \in [d]$ it holds that

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

Thus, analogous to the echelon form, the submatrix of $B$ induced by the rows $r_1, \ldots, r_d$ is the identity matrix $I_d \in \mathbb{F}_2^{d \times d}$.

One can transform a constraint representation in an echelon form into a basis representation in a canonical form and vice versa as follows. Let $M \in \mathbb{F}_2^{q \times m}$ be a matrix in an echelon form with columns $c_1 < \cdots < c_q \in [m]$ inducing the identity matrix $I_q$. Let $W \in \mathbb{F}_2^{q \times m-q}$ be the submatrix of $M$ after removing the columns $c_1, \ldots, c_q$. We construct a matrix $B \in \mathbb{F}_2^{m \times m-q}$ in the following way. The $c_i$-th row of $B$ is the $i$-th row of $W$. Thus, we have set $q$ rows of $B$ and let $r_1 < \cdots < r_{m-q} \in [m]$ be the rows of $B$ that have not been set yet. We set the $r_i$-th row of $B$ to be the canonical vector $e_i \in \mathbb{F}_2^{m-q}$, where $e_i = (0, \ldots, \underset{i}{1}, \ldots, 0)$.

Thus, the rows $r_1, \ldots, r_{m-q}$ of $B$ induce the identity matrix $I_{m-q}$ and the matrix $B$ is in a canonical form. If $c_i = i$ for all $i \in [q]$, then $r_j = q + j$ and the matrices $M$ and $B$ look as follows:

$$M = (I_q \; W), \text{ and } B = \begin{pmatrix} W \\ I_{m-q} \end{pmatrix}.$$

Note that the sequence of columns $c_1, \ldots, c_q$ of $M$ inducing the identity matrix does not have to be unique. Thus, we say $B$ is the *canonical transform of $M$ over the columns $c_1, \ldots, c_q$*. Further, by a reverse procedure, we can create the matrix $M$ again from the matrix $B$ and then the matrix $M$ is the *echelon transform of $B$ over the rows $r_1, \ldots, r_{m-q}$*. i.e., the matrix $W$ is a submatrix of $B$ induced by the rows $c_1 < \cdots < c_q$, $r_j$-th column of $M$ is the $j$-th column of $W$ and $c_i$-th column of $M$ is the canonical vector $e_i \in \mathbb{F}_2^q$.

▶ **Lemma 10.** *Let $M \in \mathbb{F}_2^{q \times m}$ be a matrix in an echelon form with columns $c_1, \ldots, c_q$ inducing the identity matrix $I_q$. Let $B \in \mathbb{F}_2^{m \times m-q}$ be the canonical transform of $M$ over $c_1, \ldots, c_q$. Then, the rows of $M$ are orthogonal (under the standard dot product over $\mathbb{F}_2$) to the columns of $B$.*

**Proof.** Let $W$ be the submatrix of $M$ after removing the columns $c_1, \ldots, c_q$. Let $u := M[i; *] \in \mathbb{F}_2^m$, and $v := B[*; j] \in \mathbb{F}_2^m$. Then,

$$\langle u, v \rangle = \sum_{k \in [m]} u_k v_k = \sum_{k \in \{c_1, \ldots, c_q\}} u_k v_k + \sum_{k \in [m] \setminus \{c_1, \ldots, c_q\}} u_k v_k. \tag{1}$$

The projection $u[c_1, \ldots, c_q]$ equals to $e_i \in \mathbb{F}_2^q$ as it corresponds to the $i$-th row of the identity matrix $I_q$. Thus, $u_{c_\ell} = 1$ if and only if $\ell = i$ and we have that $\sum_{k \in \{c_1, \ldots, c_k\}} u_k v_k = v_{c_i}$. Further, it holds $B[c_i; *] = W[i; *]$ by the construction. Therefore, $v_{c_i} = W_{i,j}$ as $v$ is the $j$-th column of $B$.

Let $r_1 < r_2 < \cdots < r_{m-q} \in [m]$ be the rows of $B$ different from $c_1, \ldots, c_q$. Recall that the submatrix of $B$ induced by the rows $r_1, \ldots, r_{m-q}$ is the identity matrix $I_{m-q}$. Thus, the projection $v[r_1, \ldots, r_{m-q}]$ corresponds to the $j$-th column of the identity matrix $I_{m-q}$ and $v_{r_\ell} = 1$ if and only if $\ell = j$. Therefore, $\sum_{k \in [m] \setminus \{c_1, \ldots, c_q\}} u_k v_k = u_{r_j}$. The $r_j$-th column of $M$ is the $j$-th column of $W$ as $r_j \notin \{c_1, \ldots, c_q\}$. Thus, $u_{r_j} = W_{i,j}$. After plugging it into (1), we have $\langle u, v \rangle = W_{i,j} + W_{i,j} = 0$ in $\mathbb{F}_2$. ◀

Now, we show how to get a basis representation from a constraint representation. Let $z \in \mathbb{F}_2^q$ for $q \leq m$ and $c_1 < \cdots < c_q \in [m]$. We pad $z$ with zeroes to get a vector in $\mathbb{F}_2^m$ in the following way:

$$\mathsf{Pad}\big(z, m, \{c_1, \ldots, c_q\}\big)_j = \begin{cases} z_i & \text{if } j = c_i, \\ 0 & \text{otherwise.} \end{cases}$$

In words, to create the vector $s := \mathsf{Pad}\big(z, m, \{c_1, \ldots, c_q\}\big)$ we put the value $z_i$ on the $c_i$-th entry of $s$ and zeroes on all entries different from $c_1, \ldots, c_q$.

▶ **Lemma 11.** *Let $z \in \mathbb{F}_2^q$ be a vector, $M \in \mathbb{F}_2^{q \times m}$ be a matrix in an echelon form with columns $c_1 < \cdots < c_q \in [m]$ inducing the identity matrix $I_q$. Let $(M|z)$ be a constraint representation of an affine space $A \subseteq \mathbb{F}_2^m$. Let $B \in \mathbb{F}_2^{m \times q}$ be the canonical transform of $M$ over the columns $c_1, \ldots, c_q$, and $s := \mathsf{Pad}\big(z, m, \{c_1, \ldots, c_q\}\big) \in \mathbb{F}_2^m$. Then, $[B, s]$ is a basis representation of the affine space $A$.*

**Proof.** Let $A' = \mathcal{C}(B) + s$, i.e., $[B, s]$ is a basis representation of $A'$. We will show that $A' = A$. Let $u \in A'$, thus $u = v + s$ for a vector $v \in \mathcal{C}(B)$. By Lemma 10 and linearity of the inner product, we have that $Mv = 0$. By the construction, the vector $s$ contains only 0's on the coordinates different from $c_1, \ldots, c_q$ and the projection $s[c_1, \ldots, c_q]$ equals the vector $z$. Since the submatrix of $M$ induced by the columns $c_1, \ldots, c_q$ is the identity matrix $I_q$, we have $Ms = z$. We conclude that $Mu = Mv + Ms = Ms = z$. Thus, $A' \subseteq A$. Since the rows of $M$ and columns of $B$ are linearly independent, we have that $\dim(A) = \dim(A') = m - q$. Therefore, $A' = A$. ◀

Let $C \in \mathbb{F}_2^{q \times m}$ be a matrix and $t \in \mathbb{F}_2^m$ 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 $\mathsf{Del}$ 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 the $\mathsf{Del}$ operation is the only operation needed to get a constraint representation for a superspace as shown later in Theorem 13. Before we prove Theorem 13, we show how the $\mathsf{Del}$ operation changes the columns of the input matrix.

▶ **Lemma 12.** *Let $C \in \mathbb{F}_2^{q \times m}$ be a matrix in an echelon form with columns $c_1, \ldots, c_q \in [m]$ inducing the identity matrix $I_q$. Let $C' := \mathsf{Del}(C, t, i)$ for a non-zero vector $t \in \mathbb{F}_2^q$ with $t_i = 1$. Then,*

1. *The columns $c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_q$ of $C'$ induce the identity matrix $I_{q-1}$.*
2. *The $c_i$-th column of $C'$ equals to the vector $t$ without the $i$-th entry, i.e., $C'[*; c_i] = t[-i]$.*
3. *Let $j \in [m] \setminus \{c_1, \ldots, c_q\}$ and let $v_j \in \mathbb{F}_2^q$ be the following vector.*

$$v_j := \begin{cases} C[*, j] + t & \text{if } C_{i,j} = 1 \\ C[*, j] & \text{otherwise} \end{cases}$$

*Then, $C'[*; j] = v_j[-i]$.*

**Proof.** Let $D$ be a matrix arising from $C$ by adding the $i$-th row to the rows $j$ such that $t_j = 1$. Thus, the matrix $C'$ arises from $D$ by removing the $i$-th row.

The columns $c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_q$ are not modified by the adding of the $i$-th row during the $\mathsf{Del}$ operation as the $i$-th entry of all these columns is 0. I.e., the column $D[*; c_j]$ (for $j \neq i$) equals to the canonical vector $e_j \in \mathbb{F}_2^q$. Thus after removing the $i$-th row, the columns $c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_q$ of $C'$ indeed induce the identity matrix $I_{q-1}$.

The column $C[*; c_i]$ equals to the canonical vector $e_i \in \mathbb{F}_2^q$. After adding the $i$-th row to the other rows of $C$ according to the vector $t$, it holds that $D[*; c_i] = t$ as $t_i = 1$ and thus, $C'[*; c_i] = D[-i; c_i] = t[-i]$.

Let $j \in [m] \setminus \{c_1, \ldots, c_q\}$. If $C_{i,j} = 0$, then the $j$-th column of $C$ is not modified by the adding of $i$-th row during the $\mathsf{Del}$ operation, i.e., $D[*; j] = C[*; j] = v_j$. On the other hand, if $C_{i,j} = 1$, then for any $k \in [q]$ holds that

$$D_{k,j} = \begin{cases} C_{k,j} + C_{i,j} & \text{if } t_k = 1, \\ C_{k,j} & \text{otherwise.} \end{cases}$$

It follows that $D[*, j] = C[*, j] + t = v_j$ if $C_{i,j} = 1$. In both cases, we have $C'[*, j] = v_j[-i]$. ◀

We are now ready to state below the main linear algebraic ingredient needed for our work.

▶ **Theorem 13.** *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.*

**Proof of Theorem 13.** Let $c_1 < \cdots < c_q$ be columns of $M$ inducing the identity matrix $I_q$. Let $B_1 \in \mathbb{F}_2^{m \times m-q}$ be the canonical transform of $M_1$ over $c_1, \ldots, c_q$, and let $s_1 := \mathsf{Pad}\big(z_1, m, \{c_1, \ldots, c_q\}\big)$. By Lemma 11, we have that $A_1 = \mathcal{C}(B_1) + s_1$. Since $A_1 \subseteq A_2$ and $\dim(A_2) = \dim(A_1) + 1$, there is a vector $v$ such that $B_2' := (B_1\ v)$ is a basis of $A_2$, i.e., $A_2 = \mathcal{C}(B_2') + s_1$ where the matrix $B_2' \in \mathbb{F}_2^{m \times m-q+1}$ arises by appending the vector $v$ as the last column to the matrix $B_1$.

Let $r_1 < \cdots < r_{m-q}$ be the rows of $B_1$ different from $c_1, \ldots, c_q$. By the construction, the rows $r_1, \ldots, r_{m-q}$ of $B_1$ induce the identity matrix $I_{m-q}$. Thus, we can assume that $v[r_1, \ldots, r_{m-q}] = (0, \ldots, 0)$. Otherwise, we can zero out those entries by adding the appropriate columns of $B_1$ to $v$. By Observation 9, this operation does not change the affine space that is represented by $[B_2', s_1]$. Note that $s_1[r_1, \ldots, r_{m-q}] = (0, \ldots, 0)$ as well by the construction.

We set the vector $t \in \mathbb{F}_2^q$ as $t := v[c_1, \ldots, c_q]$. The vector $t$ is indeed non-zero as the columns of $B_2'$ are linearly independent and we suppose that the vector $v$ contains only zeroes on all coordinates different from $c_1, \ldots, c_q$. We will show that the vector $t$ is indeed the

sought vector, i.e., for any $i \in [q]$ with $t_i = 1$ the system of equations $\mathsf{Del}\big((M|z), t, i)\big)$ is a constraint representation of $A_2$.

To get a constraint representation for $A_2$, we change $B_2'$ to a canonical form first. Let $i \in [q]$ such that $t_i = 1$. We create a matrix $B_2 \in \mathbb{F}_2^{m \times m-q+1}$ in two steps.

1. Add $v$ (the last column of $B_2'$) to each column $j$ such that the $c_i$-th entry of the column $j$ is 1 (i.e., $B_2'[c_i, j] = 1$).
2. Let $k-1$ be the number of $r_j$'s smaller then $c_i$, i.e., we have $r_1 < \cdots < r_{k-1} < c_i < r_k < \cdots < r_{m-q}$ (recall that $r_j$'s are different from $c_\ell$'s). Move the last column (the vector $v$) to be the $k$-th column of $B_2$ (the remaining columns move right by 1 position).

Formally, we have $B_2[*; k] := v$ and for $j \neq k$ :

$$B_2[*;j] := \begin{cases} B_2'[*;j] + v & \text{if } j < k \text{ and } B_2'[c_i, j] = 1, \\ B_2'[*;j] & \text{if } j < k \text{ and } B_2'[c_i, j] = 0, \\ B_2'[*;j-1] + v & \text{if } j > k \text{ and } B_2'[c_i, j-1] = 1, \\ B_2'[*;j-1] & \text{if } j > k \text{ and } B_2'[c_i, j-1] = 0. \end{cases}$$

Now, the $k$-th column of $B_2$ is the only column of $B_2$ that has 1 on the $c_i$-th entry. Further, $B_2[r_1, \ldots, r_{m-q}; k] = (0, \ldots, 0)$ as it corresponds to the last column of $B_2'$ (i.e., to the vector $v$). Let $r_1', \ldots, r_{m-q+1}'$ be the following sequence:

$$r_j' := \begin{cases} r_j & \text{if } j < k, \\ c_i & \text{if } j = k, \\ r_{j-1} & \text{if } j > k. \end{cases}$$

I.e., we insert the value $c_i$ into the $k$-th position of the sequence $r_1, \ldots, r_{m-q}$. Then, the rows $r_1' < \cdots < r_{m-q+1}'$ of $B_2$ induce the identity matrix $I_{m-q+1}$. Therefore, $B_2$ is in a canonical form. Let $s_2 \in \mathbb{F}_2^m$ be defined as

$$s_2 := \begin{cases} s_1 + v & \text{if } s_1[c_i] = 1, \\ s_1 & \text{otherwise.} \end{cases}$$

Note that the vector $s_2[r_1', \ldots, r_{m-q+1}'] = (0, \ldots, 0)$ as $v[c_i] = t_i$ and we assume that $t_i = 1$. Since changing the order of columns of a matrix does not change the columns space, $[B_2, s_2]$ is a basis representation of $A_2$ by Observation 9.

Now, let $M_2 \in \mathbb{F}_2^{q-1 \times m}$ be the echelon transform of $B_2$ over the rows $r_1', \ldots, r_{m-q+1}'$. Let $c_1', \ldots, c_{q-1}'$ be the following sequence.

$$c_j' := \begin{cases} c_j & \text{if } j < i, \\ c_{j+1} & \text{if } i \leq j. \end{cases}$$

i.e, we remove $c_i$ from the sequence $c_1, \ldots, c_q$ to get $c_1', \ldots, c_{q-1}'$. Let $z_2 \in \mathbb{F}_2^{q-1}$ be the projection of $s_2$ to the coordinates $c_1', \ldots, c_{q-1}'$ (i.e., we remove the entries $r_1', \ldots, r_{m-q+1}'$). Note that $s_2 = \mathsf{Pad}\big(z_2, m, \{c_1', \ldots, c_{q-1}'\}\big)$. Since $B_2$ is the canonical transform of $M_2$ over $c_1', \ldots, c_{q-1}'$, the system $(M_2|z_2)$ is a constraint representation of $A_2$ by Lemma 11.

Let $(M_2'|z_2') := \mathsf{Del}\big((M_1|z_1), t, i\big)$. It remains to prove that $(M_2|z_2) = (M_2'|z_2')$. By the construction, the columns $c_1', \ldots, c_{q-1}'$ of $M_2$ induce the identity matrix $I_{q-1}$. By Item 1 of Lemma 12, the columns $c_1', \ldots, c_{q-1}'$ of $M_2'$ induce the identity matrix $I_{q-1}$ as well as $c_1', \ldots, c_{q-1}' = c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_q$ by definition.

Now, consider the $r'_j$-th column of $M_2$. By the construction, we have that $M_2[*; r'_j] = B_2[c'_1, \ldots, c'_{q-1}; j]$. First, suppose that $j = k$ (i.e., $r'_j = c_i$). We have $B_2[*; k] = v$ and $t = v[c_1, \ldots, c_q]$. Thus, $M_2[*; r'_k] = v[c'_1, \ldots, c'_{q-1}] = t[-i]$ and by Item 2 of Lemma 12, the $r'_k$-th column of $M_2$ and $M'_2$ are same.

Now, suppose that $j \in [m - q + 1], j \neq k$. Let $j' := j$ if $j < k$, and $j' := j - 1$ if $j > k$. The $j$-th column of $B_2$ arises from the $j'$-th column by adding the vector $v$ if $B_1[c_i; j'] = 1$ (otherwise $B_2[*; j] = B_1[*; j']$). Note that $B_1[c'_1, \ldots, c'_q; j'] = M_1[-i; r_{j'}]$. When we put everything together we get the following.

$$M_2[*; r'_j] = B_2[c'_1, \ldots, c'_{q-1}; j] = \\ = \begin{cases} B_1[c'_1, \ldots, c'_{q-1}; j'] + v[c'_1, \ldots, c'_{q-1}] = M_1[-i; r_{j'}] + t[-i] & \text{if } B_1[c_i; j'] = 1 \\ B_1[c'_1, \ldots, c'_{q-1}; j'] = M_1[-i; r_{j'}] & \text{otherwise} \end{cases}$$

Since $r_{j'} = r'_j$ for $j \neq k$, we have that $M_2[*, r'_j] = M'_2[*, r'_j]$ by Item 3 of Lemma 12.

To show that $z_2 = z'_2$ we can use the same argument. By the construction, $z_2 = s_2[c'_1, \ldots, c'_q]$ and

$$s_2[c'_1, \ldots, c'_q] = \begin{cases} s_1[c'_1, \ldots, c'_q] + v[c'_1, \ldots, c'_{q-1}] = z_1[-i] + t[-i] & \text{if } s_1[c_i] = 1, \\ s_1[c'_1, \ldots, c'_q] = z_1[-i] & \text{otherwise.} \end{cases}$$

Thus again, we have $z_2 = z'_2$ by Item 3 of Lemma 12. Therefore, we show that $(M_2|z_2) = (M'_2|z'_2)$. ◀

We call the vector $t$ given by Theorem 13 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 $t_i = 1$.

## 7 Simulation

In this section, we prove our lifting theorem.

▶ **Theorem 14** (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 $d_{\mathsf{q}}(\mathcal{T}') \leq \log |\mathcal{T}|_{\mathsf{q}}$ and $w(\mathcal{T}') \leq w(\mathcal{T})$.*

### Algorithm

We prove Theorem 14 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 $\rho_v$ and $w(\rho_v)$ be a number of fixed bits of $\rho_v$. 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 $\ell$, 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 $\rho_v$. The other blocks are free.

For each fixed bit $i \in [n]$ of $\rho_v$, 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 constraints that are not primary for any block are called *secondary*. 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 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. All variables of free blocks are called free. Thus, the matrix $M_v$ has the following form (after rearranging columns):

<table>
<tr><td>Linear algebra classification:</td><td colspan="3">Dependent</td><td colspan="2">Independent</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>

Let $P := \{\delta^{i,b} \mid i \in [\ell], b \in \{0,1\}\}$ 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 $\alpha_v$ all secondary constraints of $(M_v|z_v)$ will be satisfied. This will help us to ensure that each $\bar{x} \in C_v$ 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 $\rho_r$ and $\alpha_r$ do not contain any fixed bit, i.e. they are equal to $*^n$, or $*^m$, respectively.

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:** For each fixed bit $i \in [n]$ of $\rho_v$, there is a unique primary 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 $B_i$.

**Invariant 3:** All variables of all free blocks are independent.

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

**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:**

1: $v \leftarrow r$ ▷ Current node of $\mathcal{T}$
2: $\rho_r \leftarrow *^n$ ▷ Known bits of $x$
3: $(M_r|z_r) \leftarrow \emptyset$ ▷ Constraint representation of $A_r = \{0,1\}^m$

**Simulation:**

4: **while** $v$ is not a leaf **do**
5: **if** $v$ is a query node **then**
6: **Process Query Node** ▷ Algorithm 2
7: **else** ▷ $v$ is a forget node
8: **Process Forget Node** ▷ Algorithm 3
9: **return** the output of $v$

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 13) 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 $\alpha_v$. Thus, there is $\bar{b} \in \{0,1\}$ such that for any extension $y$ of $\alpha_v$, 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 $\alpha_v$. 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 $\rho_v$ 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 2 holds for $u$. Invariant 3 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 $\rho_v$ by the value of $x_i$ to get $\rho_u$. 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 2 holds for $u$. Since the only new dependant variable is $a \in B_i$, Invariant 3 holds as well.

The partial assignment $\alpha_u$ differs only at the block $B_i$ from $\alpha_v$ ($\alpha_v[B_i] = *^\ell$, 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 $\alpha_u$ still satisfies all secondary constraints of $(M_u|z_u)$ and Invariant 4 holds for $\alpha_u$.

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 $z_v \in \mathbb{F}_2^c$. Let $t \in \mathbb{F}_2^c$ be a forgetting vector given by an application of Theorem 13 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 $\mathsf{Del}$ is defined in Section 6). By Theorem 13, 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

**Algorithm 2**

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

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

constraints $1, \ldots, p$ are primary and the constraints $p+1, \ldots, c$ are secondary. We consider two cases.

**Case 1:** There is an $i \in \{p+1, \ldots, c\}$ 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 $\alpha_u$. Therefore, Invariant 4 holds for $\alpha_u$.

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

**Case 2:** For all $i \in \{p+1, \ldots, c\}$, it holds that $t_i = 0$. Then, we fix some $i \in \{1, \ldots, p\}$ 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 $\mathsf{Del}$ 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 2 holds for $u$. There is no new dependant variable. Thus Invariant 3 holds as well.

We get the partial assignment $\rho_u$ by simply setting the variable $x_j$ free. The partial

assignment $\alpha_u$ differs from $\alpha_v$ 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 $\alpha_u$.

**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 2 holds for $u$.

We do not change the assignment $\rho_v$, 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 3 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 $\alpha_u$ differs from $\alpha_v$ 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 $\alpha_u$ 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 $\alpha_u$ satisfy all secondary constraints of $(M'_u|z'_u)$ and Invariant 4 holds for $\alpha_u$.

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

## Proof of Theorem 14

Theorem 14 follows from the following lemma.

▶ **Lemma 15.** *Suppose the simulation is at Line 4 of Algorithm 1, i.e., it checks whether the current node $v$ is a leaf. Then,*

1. $w\big(C(\rho_v)\big) \leq$ *co-dim*$(A_v)$.

**Algorithm 3**

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

$t \leftarrow$ forgetting vector given by Theorem 13
$u \leftarrow$ the only child of $v$
**if** $t$ contains a secondary constraint **then**
    $i \leftarrow$ arbitrary secondary constraint in $t$
    $(M_u|z_u) \leftarrow \mathsf{Del}\big((M_v|z_v), t, i\big)$ ▷ The constraint $i$ is now removed
    $\rho_u = \rho_v$
        ▷ The sets of primary constraints, fixed blocks, and marked variables are not changed
**else** ▷ $t$ contains only primary constraints
    $i \leftarrow$ arbitrary primary constraint in $t$
    $(M_u|z_u) \leftarrow \mathsf{Del}\big((M_v|z_v), t, i\big)$ ▷ The constraint $i$ is now removed
    $a \leftarrow$ the marked variable of $i$
    $B_j \leftarrow$ the block of the variable $a$ ▷ $i$ is the primary constraint of $B_j$ in $(M_v|z_v)$
    $j_1 \leftarrow$ the index of $a$ in $B_j$
    **if** the variables $B_j \setminus \{j_1\}$ 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 $(M_u|z_u)$
        $D \leftarrow$ indices of all dangerous variables of $(M_u|z_u)$ in $B_j$
        $j_2 \in D \setminus \{j_1\}$ by Definition 6
        $a' \leftarrow$ the $j_2$-th variable of $B_j$
        $k \leftarrow$ a secondary constraint of $(M_u|z_u)$ containing $a'$
        $(M_u|z_u) \leftarrow$ 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
$v \leftarrow u$

**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 Invariants 2, the number of fixed bits by $\rho_v$ is at most the number of 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 3, 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 $\alpha_v$. By Invariants 4, all secondary constraints are satisfied by any extension of $\alpha_v$. 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 $\delta^{j,x_i}$ 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 2, 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 14.** By Item 1 of Lemma 15, 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 15 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 [22] showed the following trade-off between the width and size of the tree-like resolution.

▶ **Theorem 16** (Theorem 3.1, Razborov [22]). *Let $k = k(n) \geq 4$ be any parameter and let $\varepsilon > 0$ be an arbitrary constant. Then, there exists a $k$-CNF contradiction $\tau'$ over $n$ variables such that there is a resolution refutation for $\tau'$ with width at most $O(k)$, but for any tree-like resolution refutation $\Pi$ for $\tau'$ with $w(\Pi) \leq n^{1-\varepsilon}/k$, we have the bound $|\Pi| \geq \exp(n^{\Omega(k)})$.*

By our simulation, given by Theorem 14, 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 $\tau$ over $n$ variables such that there is a resolution refutation for $\tau$ with width at most $O(k)$, but for every tree-like Res($\oplus$) refutation $\Pi$ for $\tau$ 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 $\tau'$ be a $k'$-CNF contradiction given by Theorem 16. We set $\tau := \tau' \circ g$ that is a $k$-CNF contradiction. Since there is a resolution refutation for $\tau'$ with width at most $O(k')$, then there is a resolution refutation for $\tau$ with width at most $O(k)$.

Now, let $\Pi$ be a tree-like Res($\oplus$) refutation for $\tau$ with $w(\Pi) \leq n^{1-\varepsilon}/k$. Let $\mathcal{T}$ be a canonical FPDT corresponding to $\Pi$ that computes *Search*$(\tau)$. Thus, we have $w(\mathcal{T}) = w(\Pi)$ and $|\mathcal{T}|_q \leq |\Pi|$. We change $\mathcal{T}$ to compute *Search*$(\tau') \circ g$. 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 $\tau'$. We change the output of $s$ to be the clause $D'$ instead of $D$. By Observation 5, the tree $\mathcal{T}$ now computes *Search*$(\tau') \circ g$.

By Theorem 14, there is FDT $\mathcal{T}'$ computing *Search*$(\tau')$ with $d_{\mathsf{q}}(\mathcal{T}') \leq \log |\mathcal{T}|_{\mathsf{q}}$ and $w(\mathcal{T}') \leq w(\mathcal{T})$. Let $\Pi'$ be the resolution refutation for $\tau'$ 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 4). Since $w(\mathcal{T}') \leq w(\mathcal{T}) \leq n^{1-\varepsilon}/k \leq n^{1-\varepsilon}/k'$, we have that $|\Pi'| \geq \exp(n^{\Omega(k')})$ by Theorem 16. Putting everything together, we have

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

◀

## 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 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.

13 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.

14 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.

15 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.

16 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.

17 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.

18 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.

19 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.

20 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.

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

22 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.

23 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.

24 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 $\mathsf{IND}_1$ and $\mathsf{MAJ}_3$ are strongly stifled and $\mathsf{IP}_2$ is not strongly stifled.

▶ **Observation 7.** *The functions* $\mathsf{IND}_1$ *and* $\mathsf{MAJ}_3$ *are strongly stifled.*

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

| $i$ \ $b$ | 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 8.** *The function* $\mathsf{IP}_2$ *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 $\mathsf{IP}_2$ 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 $\delta^1\big[D \setminus \{j\}\big] = \delta^j\big[D \setminus \{j\}\big]$ as required to be a strongly stifled function. ◀

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