Lifting to Randomized Parity Decision Trees

Farzan Byramji
University of California, San Diego, CA, USA

Russell Impagliazzo
University of California, San Diego, CA, USA

Abstract

We prove a lifting theorem from randomized decision tree depth to randomized parity decision tree (PDT) size. We use the same property of the gadget, stifling, which was introduced by Chattopadhyay, Mande, Sanyal and Sherif [ITCS 23] to prove a lifting theorem for deterministic PDTs. Moreover, even the milder condition that the gadget has minimum parity certificate complexity at least 2 suffices for lifting to randomized PDT size.

To improve the dependence on the gadget g in the lower bounds for composed functions, we consider a related problem g_* whose inputs are certificates of g. It is implicit in the work of Chattopadhyay et al. that for any function f, lower bounds for the $*$-depth of f_* give lower bounds for the PDT size of f. We make this connection explicit in the deterministic case and show that it also holds for randomized PDTs. We then combine this with composition theorems for $*$-depth, which follow by adapting known composition theorems for decision trees. As a corollary, we get tight lifting theorems when the gadget is Indexing, Inner Product or Disjointness.

2012 ACM Subject Classification Theory of computation → Oracles and decision trees

Keywords and phrases Parity decision trees, composition

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2025.55

Category RANDOM

Related Version *Full Version*: https://eccc.weizmann.ac.il/report/2024/202/

Funding *Farzan Byramji*: Supported by NSF Award AF: Medium 2212136.
Russell Impagliazzo: Supported by NSF Award AF: Medium 2212136.

Acknowledgements FB thanks Sreejata Kishor Bhattacharya, Eric Blais, Zachary Chase, Arkadev Chattopadhyay, Jyun-Jie Liao, Shachar Lovett, Jackson Morris and Anthony Ostuni for discussions and feedback at various stages of this work. The authors would like to thank the anonymous reviewers for helpful comments.

1 Introduction

Lifting theorems provide a way to convert lower bounds for a function f in a weak model of computation to lower bounds in a stronger model of computation by composing with a function g, typically called a *gadget*. Given functions $f : \{0,1\}^n \to \{0,1\}$ and $g : \{0,1\}^m \to \{0,1\}$, their composition $f \circ g : (\{0,1\}^m)^n \to \{0,1\}$ is defined by

$$(f \circ g)(x_1, x_2, \ldots, x_n) = f(g(x_1), g(x_2), \ldots, g(x_n)).$$

Typically such lifting theorems show that for certain choices of the gadget g, the two-party communication complexity of $f \circ g$ in some model is lower bounded by the complexity of f in a related query model [36, 27, 25, 28, 14, 31]. These have several applications and have led to the resolution of some long-standing problems [36, 26, 22, 15, 33]. An important challenge in the area is to decrease the gadget size to a constant. Current proofs require the gadget size to be logarithmic in the input length of the outer function.

As a stepping stone towards query-to-communication lifting theorems with improved gadget size, we may consider the problem of lifting to models which lie between communication protocols and decision trees. One such natural model is that of parity decision trees (PDTs). A parity decision tree is a decision tree where each node queries a parity $\sum_{i \in S} x_i$ for some $S \subseteq [n]$ and the sum is over $\mathbb{F}_2$. While being interesting on its own, another motivation for proving PDT lower bounds comes from proof complexity. The minimum size of a refutation of an unsatisfiable CNF formula ϕ in the proof system tree-like Resolution over parities (Res($\oplus$)) is (essentially) equal to the minimum size of a deterministic parity decision tree solving the related false clause search problem for ϕ [29]. Lifting theorems for deterministic parity decision trees using constant size gadgets were recently proved by Chattopadhyay, Mande, Sanyal and Sherif [16] and independently by Beame and Koroth [6] which gave a direct way to transform tree-like Resolution lower bounds to tree-like Res($\oplus$) lower bounds.

As a next step, we may ask for lifting theorems for randomized parity decision trees. While lower bounds for randomized PDTs do not seem to directly imply lower bounds for stronger proof systems, lower bound techniques against randomized PDTs (along with several other ideas) have been recently used to prove lower bounds against certain subsystems of (dag-like) Res($\oplus$) [20, 11, 3]. (More precisely, they use distributional lower bounds against deterministic PDTs.)

In this work, we prove a lifting theorem from randomized decision tree (DT) depth to randomized parity decision tree (PDT) size with constant size gadgets. For a function f, we use $\mathsf{D}^{\mathsf{dt}}(f)$ to denote the deterministic DT depth of f and $\mathsf{R}^{\mathsf{dt}}(f)$ to denote the 1/3-error randomized DT depth of f. Similarly, we use $\mathsf{DSize}^{\mathsf{dt}}(f)$ and $\mathsf{RSize}^{\mathsf{dt}}(f)$ to denote the corresponding size measures. We use $\oplus$ in the superscript to denote the analogous PDT measures. For example, $\mathsf{RSize}^{\oplus\text{-}\mathsf{dt}}(f)$ denotes the minimum size of any randomized PDT computing f to error 1/3.

To prove the lifting theorem for randomized PDTs, we use the same property of the gadget, stifling, which was introduced by [16] to prove their lifting theorem for deterministic PDTs. A function $g : \{0,1\}^m \to \{0,1\}$ is said to be k-stifled if for all $S \subseteq [m]$ of size k and $b \in \{0,1\}$, there is some way to fix all bits other than those in S to force the function g to output b. A function g which is 1-stifled is also simply called stifled. Some examples of stifled functions include Inner Product, Indexing and Majority.

▶ **Theorem 1.** For any function $f : \{0,1\}^n \to \{0,1\}$, any stifled function $g : \{0,1\}^m \to \{0,1\}$,

$$\log \mathsf{RSize}^{\oplus\text{-}\mathsf{dt}}(f \circ g) \geq \Omega_m(\mathsf{R}^{\mathsf{dt}}(f))$$

where the implicit constant depends on the gadget size m.

Our results also hold for relations f (like most other lifting theorems) and partial g, but we mostly focus on total functions in this section for simplicity.

Let us mention two applications of the above lifting theorem and its underlying ideas to regular Res($\oplus$) (see the full version for details). In their proof of an exponential separation between regular Res($\oplus$) and general Resolution, Bhattacharya, Chattopadhyay and Dvořák [11] required a randomized PDT lifting theorem and so they used a lifting theorem for randomized communication complexity [14]. This lifting theorem requires a logarithmic size gadget with a fairly large multiplicative constant which implies that the number of clauses in the resulting formula (and thereby the upper bound) is a large polynomial. By instead using the above lifting theorem for randomized PDTs with a constant size gadget, the lifted formula has the same number of clauses as the base formula (up to a constant factor) thereby improving the separation. We also show that ideas similar to those in the simulation can be used to improve the known lower bound for the bit pigeonhole principle in regular Res($\oplus$) from $\exp(\tilde{\Omega}(\sqrt[3]{n}))$ [20] to $\exp(\tilde{\Omega}(\sqrt{n}))$.

As warm-up to proving the lifting theorem for randomized PDTs, we also consider the simpler question of which gadgets allow lifting from randomized DT depth to randomized DT size, that also does not seem to have been considered before. In the deterministic case, Alekseev, Filmus and Smal [2] (also [19]) showed that a resistant gadget allows lifting DT depth to DT size, where a gadget is resistant if fixing a single bit of the input cannot fix the value of the function. We observe that their ideas can also be used to prove the analogous statement for randomized decision trees.

▶ **Theorem 2.** For any function $f : \{0,1\}^n \to \{0,1\}$, resistant function $g : \{0,1\}^m \to \{0,1\}$,

$$\log \mathsf{RSize}^{\mathsf{dt}}(f \circ g) \geq \Omega_m(\mathsf{R}^{\mathsf{dt}}(f)).$$

Theorems 1 and 2 are in fact slightly stronger than stated since they lift to *rank*, a measure which lower bounds log size. This is also true of the deterministic PDT lifting theorem [16, 6] and the simulation-based proof of the deterministic DT size lifting theorem [2], though they do not explicitly mention it. PDT rank also lower bounds depth in subspace decision trees. A subspace decision tree is a decision tree where each internal node can query the indicator of an affine subspace.

Let us recall the definition of rank, introduced by Ehrenfeucht and Haussler [21]. It will be convenient to work with the following alternative way of looking at rank. Consider decision trees where at each node, one of the two outgoing edges is marked, and define the cost of such a marked decision tree to be the maximum number of marked edges along any root-to-leaf path. The rank of a function g, $\mathsf{DRank}^{\mathsf{dt}}(g)$, is the minimum cost of such a marked decision tree computing g. Compared to size, rank is closer in spirit to depth, which better motivates some of the ideas discussed later.

The general question of whether randomized DT size satisfies a composition theorem 'Does $\log \mathsf{RSize}^{\mathsf{dt}}(f \circ g) = \Omega(\mathsf{R}^{\mathsf{dt}}(f) \log \mathsf{RSize}^{\mathsf{dt}}(g))$ hold for all f and g up to polylog factors?' was recently asked by Dahiya [18]. The corresponding question in the deterministic case has a positive answer, as shown by Dahiya and Mahajan [19].[1] They actually show that deterministic DT rank satisfies a composition theorem which implies the composition theorem for size.

The composition question is interesting even in the most basic setting of decision tree depth. While a composition theorem for deterministic depth has been known for long [38, 40, 32], the case of randomized depth is more subtle. It is still unknown whether we have $\mathsf{R}^{\mathsf{dt}}(f \circ g) = \Omega(\mathsf{R}^{\mathsf{dt}}(f)\mathsf{R}^{\mathsf{dt}}(g))$ for all total functions f and g. In fact, it is known that the statement is false in its most general form for the composition of a relation and a partial function [23] and even for the composition of partial functions [7]. There is a long line of work [1, 9, 4, 24, 23, 5, 7, 8, 13, 37] studying this question and proving lower bounds on $\mathsf{R}^{\mathsf{dt}}(f \circ g)$ of the form $\Omega(\mathsf{R}^{\mathsf{dt}}(f)M(g))$ or $\Omega(M(f)\mathsf{R}^{\mathsf{dt}}(g))$ where $M(\cdot)$ is some complexity measure.

Theorem 2 can be used to show that a composition theorem for rank implies one for depth, or, taking the contrapositive, counterexamples for depth also provide counterexamples for rank. Still, some of these composition theorems [9, 23, 8] can be adapted to give analogous composition theorems for randomized DT rank (see Appendix B in the full version).

[1] Here we require $\mathsf{DSize}^{\mathsf{dt}}(g) > m + 1$, where m is the input length for g. Considering $f = \mathrm{AND}_n$ and $g = \mathrm{AND}_m$ shows that some such condition is necessary [2]. But this still leaves open the possibility that there are other functions with $\mathsf{DSize}^{\mathsf{dt}}(g) \leq m + 1$ which allow lifting. We discuss this in more detail in the full version.

Motivated by the work on the composition question for ordinary decision trees, we try to better understand the dependence on the inner function in the lower bounds on PDT rank for composed functions. [16] proved that for any k-stifled g, $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(f \circ g) \geq k\mathsf{D}^{\mathsf{dt}}(f)$ and the dependence on stifling in this lower bound cannot be improved since for some functions g such as Indexing, g is k-stifled and $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(g) = k + 1$.

We observe that the ideas in [16] also work with an adaptive version of stifling. To state this notion precisely, we consider the following task. Let $g : \{0,1\}^m \to \{0,1\}$ be a Boolean function. Given query access to a certificate $z \in \{0,1,*\}^m$ of g, recognize whether z is a 0-certificate or a 1-certificate. Let g_* denote this problem. Now instead of counting all queries in the cost, only include the queries which evaluate to $*$. The minimum number of $*$'s required to solve g_* is denoted by $\mathsf{D}^{*\text{-}\mathsf{dt}}(g_*)$. Using this notion, we show the following.

▶ **Theorem 3.** For all functions $f : \{0,1\}^n \to \{0,1\}, g : \{0,1\}^m \to \{0,1\}$,

$$\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(f \circ g) \geq \mathsf{D}^{\mathsf{dt}}(f)(\mathsf{D}^{*\text{-}\mathsf{dt}}(g_*) - 1).$$

This is inspired by discussion at the end of the talk [39], where it is shown that for the Inner Product gadget we can get linear dependence on the gadget size even though Inner Product is not 2-stifled. Observe that if g is k-stifled, then $\mathsf{D}^{*\text{-}\mathsf{dt}}(g_*) > k$ since the first k queries made by an algorithm could all be $*$ and it has still not determined whether z is a 0-certificate or a 1-certificate. Thus, the above lower bound is always at least as good as the one obtained from stifling. When g is Inner Product or Disjointness on $2m$ bits, g is not 2-stifled but $\mathsf{D}^{*\text{-}\mathsf{dt}}(g_*) = m$, which shows that this bound can sometimes be better.

We also prove a randomized analogue of Theorem 3.

▶ **Theorem 4.** For all functions $f : \{0,1\}^n \to \{0,1\}, g : \{0,1\}^m \to \{0,1\}$,

$$\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}(f \circ g) \geq \Omega(\mathsf{R}^{\mathsf{dt}}(f)(\mathsf{LR}^*(g_*) - O(1))).$$

Here LR^* is the natural $*$ analogue of the linearized complexity measure introduced in [8] where an inner composition theorem $\mathsf{R}^{\mathsf{dt}}(f \circ g) = \Omega(\mathsf{R}^{\mathsf{dt}}(f)\mathsf{LR}(g))$ was proved. For a function $h : \{0,1,*\}^m \to \{0,1,*\}$,

$$\mathsf{LR}^*(h) = \inf_{\mathcal{T}} \max_x \frac{\mathrm{cost}^*(\mathcal{T}, x)}{\mathrm{bias}(\mathcal{T}, x)},$$

where $\mathcal{T}$ varies over randomized decision trees on $\{0,1,*\}^n$, x varies over inputs in the domain of h, $\mathrm{cost}^*(\mathcal{T}, x)$ denotes the expected number of $*$'s seen when running $\mathcal{T}$ on x and $\mathrm{bias}(\mathcal{T}, x) = \max\{0, 2\Pr[\mathcal{T}(x) = h(x)] - 1\}$.

Using this, we can show that when the inner function is Inner Product, Disjointness or Indexing, the upper bound $\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}(f \circ g) \leq O(\mathsf{R}^{\mathsf{dt}}(f)\mathsf{RRank}_{\epsilon}^{\oplus\text{-}\mathsf{dt}}(g))$ for $\epsilon \ll 1/\mathsf{R}^{\mathsf{dt}}(f)$ is the best one can do. For these inner functions, the deterministic PDT rank is equal to the randomized PDT rank (up to constants) so the upper bound is simply $O(\mathsf{R}^{\mathsf{dt}}(f)\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}(g))$.

▶ **Corollary 5.** For $m \geq 2$, for all functions $f : \{0,1\}^n \to \{0,1\}$,

$$\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}(f \circ \mathrm{DISJ}_{2m}) = \Theta(\mathsf{R}^{\mathsf{dt}}(f)\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}(\mathrm{DISJ}_{2m})),$$
$$\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}(f \circ \mathrm{IP}_{2m}) = \Theta(\mathsf{R}^{\mathsf{dt}}(f)\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}(\mathrm{IP}_{2m})),$$
$$\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}(f \circ \mathrm{IND}_{m+2^m}) = \Theta(\mathsf{R}^{\mathsf{dt}}(f)\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}(\mathrm{IND}_{m+2^m})).$$

While Theorems 3 and 4 do imply the lifting theorems mentioned earlier[2], they still work in the standard basis. It is natural to consider analogues of the above measures which work with parity certificates instead of ordinary certificates, and indeed we can prove analogues of the above results using such measures. These lifting theorems can be found in the full version.

Theorems 3 and 4 can be viewed as improving the quantitative dependence on the gadget in the lifting theorems obtained via stifling. We can also consider the qualitative question of whether a more general property than stifling allows lifting to parity decision trees. In this direction, Alekseev, Filmus and Smal [2] completely classified gadgets based on what gadgets allow polynomial lifting, $\log \mathsf{DSize}^{\oplus\text{-}\mathsf{dt}}(f \circ g) = \Omega_g(\mathsf{D}^{\mathsf{dt}}(f)^{\epsilon})$ (for some $\epsilon > 0$). However, this does not answer the question of which gadgets allow linear lifting, $\log \mathsf{DSize}^{\oplus\text{-}\mathsf{dt}}(f \circ g) = \Omega_g(\mathsf{D}^{\mathsf{dt}}(f))$. We observe by considering a mild generalization of stifling that for any gadget g whose minimum parity certificate complexity is at least 2, $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(f \circ g) \geq \mathsf{D}^{\mathsf{dt}}(f)$. This can be seen as the natural parity analogue of the statement that if g has minimum certificate complexity at least 2, then for all f, $\mathsf{DRank}^{\mathsf{dt}}(f \circ g) \geq \mathsf{D}^{\mathsf{dt}}(f)$ [2, 19].

The similarities go further. The class of gadgets g which are not already captured by the above condition (possibly after first moving to a subfunction) are the ones which satisfy $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(g) = 1$. If g is a total function which cannot be computed by a single parity query and satisfies $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(g) = 1$, we show that there is some gadget h such that understanding whether g allows lifting to PDT rank is equivalent to understanding whether h allows lifting to DT rank. More precisely, we have the following.

▶ **Proposition 6.** Let $g : \{0,1\}^m \to \{0,1\}$ be a total function which is not a parity. Then one of the following holds:

- $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(g) \geq 2$ and for all functions f, we have $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(f \circ g) \geq \mathsf{D}^{\mathsf{dt}}(f)$ and $\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}(f \circ g) \geq \Omega_m(\mathsf{R}^{\mathsf{dt}}(f))$.
- $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(g) = 1$ and there exists some $h : \{0,1\}^k \to \{0,1\}$ such that for all functions f, we have $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(f \circ g) = \mathsf{DRank}^{\mathsf{dt}}(f \circ h)$ and $\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}(f \circ g) = \Theta(\mathsf{RRank}^{\mathsf{dt}}(f \circ h))$.

1.1 Techniques

The simulation for proving the lifting theorem for randomized PDTs with stifled gadgets builds on the ideas of [16]. Let us start by recalling their main idea for simulating a parity query efficiently. On an input x for f, we will simulate a parity decision tree T for $f \circ g$. For concreteness, suppose the parity at the root of T is $z_{11} + z_{21} + z_{22} + z_{32}$. While this parity depends on multiple blocks, we would like to simulate it while making just one query to x. To do this, we localize the parity query to a single bit, say z_{11}, which is informally thought of as being responsible for the value of the whole parity. Specifically, we view $z_{11} + z_{21} + z_{22} + z_{32} = b$ for some $b \in \mathbb{F}_2$ as fixing $z_{11} = z_{21} + z_{22} + z_{32} + b$ where we think of z_{21}, z_{22}, z_{32} as still being free. At this point, we no longer have control of z_{11} but since g is stifled, we can fix the remaining bits in block z_1 to force $g(z_1) = x_1$. So we only need to make one actual query to simulate a parity query.

Now suppose we wish to simulate a randomized PDT. To ensure correctness, following the usual simulation framework for randomized communication protocols and decision trees, we now require suitable hard distributions μ_0 and μ_1 on the 0-inputs and 1-inputs respectively of the gadget g. On an input x for f, we simulate a parity decision tree T for $f \circ g$ on the

[2] Strictly speaking, Theorem 4 does not seem to directly imply Theorem 1 because of the additive constant loss in LR^*. However, one can use other measures for which an inner composition theorem holds, like sabotage complexity, in place of LR^* to recover Theorem 1.

distribution $\mu^x := \mu_{x_1} \times \mu_{x_2} \times \cdots \times \mu_{x_n}$. Continuing with the above example, suppose we wish to simulate the parity $z_{11} + z_{21} + z_{22} + z_{32}$ on the distribution μ^x where x is unknown. In general, simulating this parity by only querying one bit of x seems hard but consider the following very special case. On querying x_1, suppose we find that the corresponding distribution μ of block z_1 is such that z_{11} is a uniform random bit which is independent of all the other bits, i.e. $\mu = \mathcal{U}_1 \times \mu'$ for some distribution μ' on $\{0,1\}^{m-1}$. Then irrespective of the distributions of z_2 and z_3, we know that this parity is equally likely to be 0 or 1 and so we can just move to a uniform random child. The remaining bits in z_1 are set according to μ'. So at least in this special case, we could simulate the parity with just one query.

To actually use the above observation in our simulation, we rely on the following idea of Bhattacharya, Chattopadhyay and Dvořák [11]. If g is a stifling gadget of constant size m, the uniform distributions on $g^{-1}(0)$ and $g^{-1}(1)$ have the following useful property. For any $i \in [m]$, with constant probability, the bits other than i form a certificate of g in which case the i^{th} bit is uniformly distributed. Using this property, we can now simulate a parity query across multiple blocks by a constant number of queries in expectation since each time we make an actual query, with good probability, we will be able to simulate the current parity without making any more queries.

Next we briefly describe the ideas behind the PDT simulation theorems with improved dependence on the gadget. We focus here on the deterministic case; the randomized case follows a similar proof outline. For the deterministic PDT lifting theorem, we proceed in two steps. First, we observe that the proof in [16] implicitly uses a general reduction showing that for all functions, $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(f) \geq \mathsf{D}^{*\text{-}\mathsf{dt}}(f_*)$. In particular, f does not need to be of composed form. This lower bound even works for relations $\mathcal{R}$, once we suitably define $\mathcal{R}_*$. In Appendix A of the full version, we use this lemma and other ideas in this work to give simple proofs of some known lower bounds for tree-like Res($\oplus$).

With the lower bound $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(f \circ g) \geq \mathsf{D}^{*\text{-}\mathsf{dt}}((f \circ g)_*)$ in hand, the next step is to simply note that $\mathsf{D}^{*\text{-}\mathsf{dt}}$ satisfies a composition theorem $\mathsf{D}^{*\text{-}\mathsf{dt}}(f \circ g_*) \geq \mathsf{D}^{\mathsf{dt}}(f)(\mathsf{D}^{*\text{-}\mathsf{dt}}(g_*) - 1)$ analogous to usual decision tree complexity. Combining these gives Theorem 3. The simulation in [16] can be understood as instead using the relation $\mathsf{D}^{*\text{-}\mathsf{dt}}(f \circ g_*) \geq k\mathsf{D}^{\mathsf{dt}}(f)$ whenever g is k-stifled, which can be seen as some analogue of $\mathsf{D}^{\mathsf{dt}}(f \circ g) \geq \mathsf{D}^{\mathsf{dt}}(f)\mathsf{C}(g)$.

1.2 Related work

In independent work, Podolskii and Shekhovtsov [34] have also proved a lifting theorem for randomized PDTs. In fact, they lift to the stronger model of semistructured communication protocols where one party is restricted to sending parities and the other can send arbitrary messages. The gadgets they allow are certain generalizations of Indexing where each index points to a distinct parity and their lifting theorem has the right dependence on the gadget size for such gadgets. This class naturally captures gadgets like Indexing and Inner Product, and by considering suitable reductions, their lifting theorem also applies to other gadgets. Some of the underlying ideas for simulating parities in their work are similar to ours, though it seems that our techniques do not directly imply their result for PDTs with the correct dependence on the gadget size and vice-versa.

Since we focus on the simpler model of PDTs, our proof of the lifting theorem using stifling gadgets is quite short, and we also provide a refined classification of gadgets for when lifting to PDTs is possible. It would be interesting to give a simulation unifying the lower bounds from [34] and our lower bounds for composed problems via $*$-depth. We suspect that by combining techniques useful for composition in ordinary decision trees, with techniques used in query-to-communication lifting, it may be possible to find broader classes of gadgets which allow lifting to semi-structured protocols.

Besselman et al. [10] have recently obtained direct sum theorems for randomized PDT depth. They show that a direct sum theorem holds when the lower bound is proved via discrepancy or against product distributions. The direct sum question can be seen as the special case of composition where the outer function is the identity function. Our results using $*$-depth (and its parity generalization) also give direct sum results for PDTs, where direct sum theorems hold for $*$-depth by adapting the proofs for ordinary decision trees [30].

These direct sum results are incomparable to those of [10]. In one direction, for the Majority function, we have $\mathsf{R}^{*\text{-}\mathsf{dt}}(\mathrm{MAJ}_*) = O(\sqrt{n})$ while any PDT solving MAJ on the uniform distribution requires depth $\Omega(n)$. On the other hand, there are functions for which randomized $*$-depth is polynomially larger than the PDT depth on product distributions. Such functions can be obtained by lifting such a separation for ordinary decision trees using a stifled gadget. The NAND tree provides such a separation for ordinary decision trees [37]. Similarly, in the deterministic case, the NAND tree also shows that deterministic $*$-depth can sometimes be quadratically larger than (parity) certificate complexity.

1.3 Organization

In Section 2, we state definitions for the query complexity measures used. In Section 3, we prove Theorems 1 and 2. In Section 4, we prove Theorems 3 and 4.

2 Preliminaries

For a positive integer n, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. All logs are to the base 2. We use $|x|$ to denote the Hamming weight of a string $x \in \{0,1\}^n$. Let $\mathcal{R} \subseteq \{0,1\}^n \times \mathcal{O}$ be a relation. Let $g : M \to \{0, 1, *\}$ be a partial function on some domain M (typically $M = \{0,1\}^m$). Then the composed relation $\mathcal{R} \circ g \subseteq M^n \times \mathcal{O}$ is defined as follows. If for $x \in M^n$, there is some $i \in [n]$ such that $g(x_i) = *$, then $(x, o) \in \mathcal{R} \circ g$ for all $o \in \mathcal{O}$ (in this case, we think of x as lying outside the domain). Otherwise $(x, o) \in \mathcal{R} \circ g$ if and only if $(g^n(x), o) \in \mathcal{R}$. For a relation $\mathcal{R}$ and input $x \in \{0,1\}^n$, we sometimes use $\mathcal{R}(x) := \{o \in \mathcal{O} | (x, o) \in \mathcal{O}\}$ to denote the legal outputs on x. A partial function $g : \{0,1\}^m \to \{0, 1, *\}$ is also sometimes interpreted as the relation where x is related to $g(x)$ if x is in the domain of g and otherwise x is related to all possible outputs $\{0, 1\}$.

We use standard $\Omega(\cdot), O(\cdot)$ notation in most places to only represent universal constants and when required, we will explicitly note which parameters need to be large for the inequalities to hold. Additionally, if the constant depends on some parameter or function, this will be indicated by a subscript. We now define the query complexity measures used in this work. Refer to [12] for a survey about some of these measures.

Decision trees. A deterministic decision tree T on $\{0,1\}^n$ is a binary rooted tree with leaves being labeled from some set $\mathcal{O}$ and internal nodes labeled by $i \in [n]$ each with two outgoing edges labeled 0 and 1. On an input $x \in \{0,1\}^n$, starting at the root, we repeatedly follow the edge according to the value of x_i where i is the label of the current node, until we reach a leaf. The label of the leaf is the output of T on x, which we denote by $T(x)$.

The cost of T on x, $\mathrm{depth}(T, x)$ (or sometimes $\mathrm{cost}(T, x)$), is the number of queries made by T on x. The depth of T is defined by $\mathrm{depth}(T) = \max_{x \in \{0,1\}^n} \mathrm{depth}(T, x)$. The size of T, denoted $\mathrm{size}(T)$, is the number of leaves of T.

The rank of T is defined in the following way. Consider markings of the edges of T such that one of the outgoing edges from each internal node is marked. For such a marked tree, the cost associated with this marking is the maximum number of marked edges on any

root-to-leaf path. The rank of a tree T is the minimum cost of any marking of T. This definition of rank is equivalent to the more common bottom-up definition and the equivalence is proved in [17] but the idea also appears implicitly in prior work. For such a marked tree T, we will use $\text{cost}(T, x)$ to denote the number of marked edges on the root-to-leaf path taken by x.

For a relation $\mathcal{R} \subseteq \{0,1\}^n \times \mathcal{O}$, a decision tree T is said to compute $\mathcal{R}$ if for all $x \in \{0,1\}^n$, $(x, T(x)) \in \mathcal{R}$. Define $\mathsf{D}^{\mathsf{dt}}(\mathcal{R})$ to be the minimum depth of a deterministic decision tree computing $\mathcal{R}$. Define $\mathsf{DSize}^{\mathsf{dt}}(\mathcal{R})$ and $\mathsf{DRank}^{\mathsf{dt}}(\mathcal{R})$ to be the minimum size and rank respectively of a decision tree computing $\mathcal{R}$.

A randomized decision tree $\mathcal{T}$ is a probability distribution over deterministic decision trees. We will use the same notation as in the deterministic case to denote the worst-case depth, size or rank of a randomized decision tree. We also use the corresponding expected measures described next. On input $x \in \{0,1\}^m$, we define $\text{cost}(\mathcal{T}, x) = \mathbb{E}_{T \sim \mathcal{T}}[\text{cost}(T, x)]$. Define $\text{cost}(\mathcal{T}) = \max_x \text{cost}(\mathcal{T}, x)$. Similarly $\overline{\text{rank}}(\mathcal{T}) = \mathbb{E}_{T \sim \mathcal{T}}[\text{rank}(T)]$ and $\overline{\text{size}}(\mathcal{T}) = \mathbb{E}_{T \sim \mathcal{T}}[\text{size}(T)]$.

For a relation $\mathcal{R}$, a randomized decision tree $\mathcal{T}$ is said to compute $\mathcal{R}$ to error ϵ if for all $x \in \{0,1\}^n$, $\Pr_{T \sim \mathcal{T}}[(x, T(x)) \in \mathcal{R}] \geq 1 - \epsilon$. We use $\mathsf{R}^{\mathsf{dt}}_\epsilon(\mathcal{R}), \mathsf{RSize}^{\mathsf{dt}}_\epsilon(\mathcal{R}), \mathsf{RRank}^{\mathsf{dt}}_\epsilon(\mathcal{R})$ to denote the worst case analogues of $\mathsf{D}^{\mathsf{dt}}(\mathcal{R}), \mathsf{DSize}^{\mathsf{dt}}(\mathcal{R}), \mathsf{DRank}^{\mathsf{dt}}(\mathcal{R})$ for randomized decision trees that compute $\mathcal{R}$ to error ϵ. The corresponding expected measures are denoted by $\overline{\mathsf{R}^{\mathsf{dt}}_\epsilon}(\mathcal{R}), \overline{\mathsf{RSize}^{\mathsf{dt}}_\epsilon}(\mathcal{R}), \overline{\mathsf{RRank}^{\mathsf{dt}}_\epsilon}(\mathcal{R})$. We omit the subscript when dealing with error $\epsilon = 1/3$.

For any decision tree T, $\text{rank}(T) \leq \log \text{size}(T)$. This directly implies for a randomized decision tree $\mathcal{T}$, $\text{rank}(\mathcal{T}) \leq \log \text{size}(\mathcal{T})$ by applying the above relation to each tree in the support of $\mathcal{T}$. To get the corresponding relation for the expected complexity measures, we use Jensen's inequality to get $\mathbb{E}_{T \sim \mathcal{T}}[\text{rank}(T)] \leq \mathbb{E}_{T \sim \mathcal{T}}[\log \text{size}(T)] \leq \log \mathbb{E}_{T \sim \mathcal{T}}[\text{size}(T)]$, which in our notation is $\overline{\text{rank}}(\mathcal{T}) \leq \log \overline{\text{size}}(\mathcal{T})$. This inequality does not depend on what queries are allowed and also holds for other kinds of decision trees (like parity decision trees).

A parity decision tree (PDT) T is like a decision tree but the internal nodes can now query parities. We will denote a parity in different ways, $\langle \alpha, x \rangle$ for some $\alpha \in \mathbb{F}_2^n$ or as $\sum_{i \in S} x_i$ where α is the indicator vector for S. The notation for parity decision trees is similar to that for (ordinary) decision trees. We will use $\oplus$ in the superscript to denote the parity analogue of an ordinary query complexity measure. For example, $\overline{\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}}(\mathcal{R})$ denotes the minimum expected rank of a parity decision tree computing $\mathcal{R}$ to error $1/3$. When dealing with a parity v on inputs with an underlying block structure $(\{0,1\}^m)^n$, for $i \in [n]$, we use $v|_i$ to denote the projection of v onto the i^{th} block.

We now define 0-depth and 1-depth. The 0-depth of an ordinary decision tree T is the maximum number of edges labeled 0 on any root-to-leaf path in T. We use $\mathsf{D}^{0\text{-}\mathsf{dt}}(\mathcal{R})$ to denote the minimum 0-depth of a deterministic decision tree for $\mathcal{R}$, and similar notation with 0 in the superscript for other 0-query complexity measures. The measures related to 1-depth are defined similarly.

We next mention two standard techniques for proving lower bounds on deterministic decision tree depth and rank. To prove lower bounds on $\mathsf{D}^{\mathsf{dt}}(\mathcal{R})$ for a relation $\mathcal{R}$, it suffices to give an Adversary strategy in the Querier-Adversary game for the relation $\mathcal{R}$. In this game, Adversary has a hidden string x and Querier's goal is to find some o related to x according to $\mathcal{R}$ while making as few queries as possible. This technique is complete in the sense that if $\mathsf{D}^{\mathsf{dt}}(\mathcal{R}) = d$, then there is an Adversary strategy scoring d points. This game also works for other deterministic query complexity measures by changing the kinds of queries Querier is allowed to make.

A similar game can be used to characterize the rank of a relation. In the Prover-Delayer game [35] for relation $\mathcal{R}$, similar to the Querier-Adversary game, Prover makes queries to a hidden string x and Delayer responds by revealing the corresponding bits of x, except for the following change. Delayer may instead choose to respond with $*$, which is interpreted as Prover getting to decide how to fix the queried bit. Delayer gets to know what bit Prover picks in this case. The game continues in this manner until Prover can correctly output an o which is related to x. Delayer's score is the number of $*$'s announced during the game. The maximum score guaranteed by a Delayer strategy for $\mathcal{R}$ is equal to the rank of $\mathcal{R}$ (see [19] for a proof).

The Prover-Delayer game can be equivalently described in a way closer to the Querier-Adversary game in the following way. Now instead of just picking some x_i to query, Querier also picks a bit $b \in \{0,1\}$ and Adversary gets a point only if the announced value is equal to b. The best score achievable by an Adversary strategy in this game is equal to the best score of a Delayer strategy. This equivalent view can be seen as the natural game corresponding to the description of rank using marked decision trees.

By changing the allowed queries, the Prover-Delayer game can capture rank in other query models. For instance, rank in PDTs is captured by the parity Prover-Delayer game [29].

Certificate complexity. A partial assignment C is a string in $\{0,1,*\}^n$. Say that C is consistent with $x \in \{0,1\}^n$ if for all $i \in [n]$, $C_i = x_i$ or $C_i = *$. We will sometimes interpret a partial assignment as the subcube it defines, $\{x \in \{0,1\}^n \,|\, x \text{ is consistent with } C\}$. So we write $x \in C$ to express that x is consistent with C.

C is said to be a certificate for a relation $\mathcal{R} \subseteq \{0,1\}^n \times \mathcal{O}$ if there exists some $o \in \mathcal{O}$, such that for all $x \in C$, $(x,o) \in \mathcal{R}$. For a partial function $g : \{0,1\}^m \to \{0,1,*\}$ and $b \in B$, a partial assignment C is said to be a b-certificate if for all $x \in C$, $(x,b) \in g$ (interpreting g as a relation). In other words, we require that for all $x \in C$, $g(x) \in \{b,*\}$. The size of a certificate C is the number of non-$*$'s in it. The certificate complexity of a relation $\mathcal{R}$ at $x \in \{0,1\}^n$, denoted $\mathsf{C}(\mathcal{R},x)$, is defined as the minimum size of a certificate which is consistent with x. The certificate complexity of $\mathcal{R}$, $\mathsf{C}(\mathcal{R})$ is the maximum certificate complexity of any input. The minimum certificate complexity of $\mathcal{R}$, $\mathsf{C}_{\min}(\mathcal{R})$, is the minimum size of a certificate for $\mathcal{R}$.

When working with a partial function g, we will sometimes only be interested in certificates whose corresponding subcubes are completely contained in the domain of g. We will call these *domain* certificates. We will drop the word domain when it is clear from context or when working with total functions.

A parity certificate for $\mathcal{R} \subseteq \{0,1\}^n \times \mathcal{O}$ is given by a collection of $\mathbb{F}_2$ linear equations on $\{0,1\}^n$, $S = \{\langle \alpha_1, x\rangle = b_1, \langle \alpha_2, x\rangle = b_2, \ldots, \langle \alpha_k, x\rangle = b_k\}$ such that there exists $o \in \mathcal{O}$ for which for all $x \in \mathbb{F}_2^n$ satisfying the equations in S, we have $(x,o) \in \mathcal{R}$. The size of a parity certificate is the number of equations in it. We may always assume that the linear forms involved in a parity certificate are linearly independent, since we can remove redundant equations without changing the defined affine subspace. The minimum parity certificate complexity of $\mathcal{R}$, $\mathsf{C}^{\oplus}_{\min}(\mathcal{R})$, is the minimum size of a parity certificate for $\mathcal{R}$. A domain parity certificate of a partial function g is a parity certificate for g whose corresponding affine subspace is completely contained in the domain of g.

3 Lifting theorems for randomized rank

In this section, we describe simple simulation theorems which lift randomized decision tree depth to rank in randomized decision trees and randomized parity decision trees.

3.1 Lifting to randomized ordinary decision tree rank

Alekseev, Filmus and Smal [2] showed that resistant gadgets suffice for lifting decision tree depth to size by generalizing Urquhart's argument for the XOR gadget [41]. A function g is said to be k-resistant if $\mathsf{C}_{\mathsf{min}}(f) \geq k+1$.

▶ **Theorem 7** ([2, 19])**.** For any k-resistant function g and any relation $\mathcal{R}$,

$$\log \mathsf{DSize}^{\mathsf{dt}}(\mathcal{R} \circ g) \geq \mathsf{DRank}^{\mathsf{dt}}(\mathcal{R} \circ g) \geq k\mathsf{D}^{\mathsf{dt}}(\mathcal{R}).$$

This also follows from results of Dahiya and Mahajan [19] who show more generally that $\mathsf{DRank}^{\mathsf{dt}}(\mathcal{R} \circ g) \geq (\mathsf{DRank}^{\mathsf{dt}}(g) - 1)\mathsf{D}^{\mathsf{dt}}(\mathcal{R})$ and $\mathsf{DRank}^{\mathsf{dt}}(g) \geq \mathsf{C}_{\mathsf{min}}(g)$.

We prove the following for randomized decision trees.

▶ **Theorem 8.** Suppose $g : \{0,1\}^m \to \{0,1,*\}$ is k-resistant. For any relation $\mathcal{R} \subseteq \{0,1\}^n \times \mathcal{O}$,

$$\log \overline{\mathsf{RSize}^{\mathsf{dt}}_{\epsilon}}(\mathcal{R} \circ g) \geq \overline{\mathsf{RRank}^{\mathsf{dt}}_{\epsilon}}(\mathcal{R} \circ g) \geq \frac{k}{2m}\overline{\mathsf{R}^{\mathsf{dt}}_{\epsilon}}(\mathcal{R}).$$

By standard arguments, we get lifting in the worst case if we incur an additive loss in the error. This additive loss can be removed by standard amplification when the outer relation is a function and the error is a constant to get Theorem 2.

For the simulation, it will be convenient to work with an equivalent distributional description of resistant functions.

▶ **Definition 9** (balanced function)**.** A function $g : \{0,1\}^m \to \{0,1,*\}$ is p-balanced for $p \in (0,1/2]$ if for every $b \in \{0,1\}$, there is a distribution μ_b supported on $g^{-1}(b)$ such that for each $i \in [m]$, for each $c \in \{0,1\}$, $\Pr_{x\sim\mu_b}[x_i = c] \geq p$. A function is balanced if it is p-balanced for some $p \in (0,1/2]$.

Note that a balanced function is necessarily resistant. It is also easy to see that being resistant is a sufficient condition for being balanced, as shown below.

▶ **Observation 10.** *If $g : \{0,1\}^m \to \{0,1\}$ is k-resistant, then it is $k/(2m)$-balanced.*

Proof. For $b \in \{0,1\}$, the distribution μ_b witnessing that g is $k/(2m)$-balanced is defined in the following way. Select a subset S of $[m]$ of size k uniformly at random. For each $i \in S$, pick x_i uniformly at random (independently of other bits). Finally set all remaining bits so that the resulting string is a b-input for g. This last step can be performed by the assumption that g is k-resistant. Each $i \in [m]$ is included in S with probability k/m and conditioned on being included in the first step, it is fixed to $c \in \{0,1\}$ with probability $1/2$. ◀

We now show that any balanced gadget can be used for lifting to randomized decision tree rank. Using the distributions coming from the above observation, the simulation below is equivalent to applying a suitable random projection as in the second proof of the depth to size lifting theorem in [2]. However, it is analyzed differently for which presenting it as a simulation is more convenient.

▶ **Proposition 11.** Let $g : \{0,1\}^m \to \{0,1,*\}$ be p-balanced for some $p \in (0,1/2]$. Then for all relations $\mathcal{R} \subseteq \{0,1\}^n \times \mathcal{O}$,

$$\overline{\mathsf{RRank}^{\mathsf{dt}}_{\epsilon}}(\mathcal{R} \circ g) \geq p\overline{\mathsf{R}^{\mathsf{dt}}_{\epsilon}}(\mathcal{R}).$$

Proof. We will show how to simulate a randomized decision tree $\mathcal{T}$ computing $\mathcal{R} \circ g$ with error probability ϵ by a randomized decision tree $\mathcal{T}'$ to compute $\mathcal{R}$ with the same error probability. The expected depth of $\mathcal{T}'$ on any input will be at most $\overline{\text{rank}}(\mathcal{T})/p$.

Let μ_0 and μ_1 be the distributions on $g^{-1}(0)$ and $g^{-1}(1)$ respectively showing that g is p-balanced. For each $x \in \{0,1\}^n$, let μ^x be the distribution on $(\{0,1\}^m)^n$ defined by independently sampling for each $i \in [n]$, the block z_i from μ_{x_i}. The simulation essentially samples $z \sim \mu^x$, where x is the input on which we wish to compute $\mathcal{R}$, and executes $\mathcal{T}$ on z. Since x is unknown, the individual blocks of z are sampled as they are queried by the decision tree $\mathcal{T}$. Since $\mathcal{T}$ computes $\mathcal{R} \circ g$ correctly for each z with probability $1 - \epsilon$, the probability that $\mathcal{T}'$ outputs incorrectly on input x is at most $\mathbb{E}_{z \sim \mu^x}[\Pr[(z, \mathcal{T}(z)) \notin \mathcal{R} \circ g]] \leq \epsilon$ where the inner probability is over the randomness of $\mathcal{T}$. So $\mathcal{T}'$ computes $\mathcal{R}$ to error ϵ.

We now describe $\mathcal{T}'$ in more detail. Since a randomized decision tree $\mathcal{T}$ is a distribution over deterministic decision trees T, it is enough to show how to simulate a deterministic decision tree T while making at most $\text{rank}(T)/p$ queries in expectation. Suppose T queries $z_{i,j}$ at the root. Then we query x_i and sample $z_i \sim \mu_{x_i}$. Now that z_i is known, we move to the appropriate child. In the future, if T makes any queries to bits of z_i, we move according to the already sampled z_i. We repeat this process until we reach a leaf at which point we output the label of the leaf reached. Note that this procedure also generates $T(z)$ where $z \sim \mu^x$ since $\mu^x = \prod_{i \in [n]} \mu_{x_i}$.

To estimate the number of queries made to x, we will show next that starting at any node in T during the simulation, the number of queries made until we cross a marked edge is at most $1/p$ in expectation. At any node, one of the outgoing edges is marked. By the assumption on μ_0, μ_1, whenever a query is made, we follow the marked edge with probability at least p. Thus, it takes at most $1/p$ queries in expectation to cross a marked edge. (During the simulation, we may reach a node where we directly move to the unmarked child with probability 1 because the bit there had already been sampled earlier, but in this case no new query is made at that node.)

To get the total number of queries made in expectation when simulating T, define for each $i \in [\text{rank}(T)]$, the random variable X_i counting the number of queries made between crossing the $(i-1)^{th}$ marked edge and crossing the i^{th} marked edge during the simulation. If we have reached a leaf of T without crossing i edges, then $X_i = 0$. Then we have $\mathbb{E}[X_i] \leq 1/p$ for all $i \in [\text{rank}(T)]$ by what was argued above.

Now the total number of queries made is $\sum_{i=1}^{\text{rank}(T)} X_i$ since we must reach a leaf in T after crossing at most $\text{rank}(T)$ many marked edges. By linearity of expectation $\mathbb{E}[\sum_{i=1}^{\text{rank}(T)} X_i] = \sum_{i=1}^{\text{rank}(T)} \mathbb{E}[X_i] \leq \text{rank}(T)/p$. The total number of queries made when simulating $\mathcal{T}$ is at most $\mathbb{E}_{T \sim \mathcal{T}}[\text{rank}(T)/p] = \overline{\text{rank}}(\mathcal{T})/p$. ◀

Theorem 8 now follows from combining Observation 10 and Proposition 11.

3.2 Lifting to randomized parity decision tree rank

We now prove that stifled gadgets allow lifting to randomized parity decision tree rank.

▶ **Definition 12** (stifled functions)**.** A function $g : \{0,1\}^m \to \{0,1,*\}$ is k-stifled if for every subset S of $[n]$ of size k and each $b \in \{0,1\}$, there is a domain b-certificate $C \in \{0,1,*\}^n$ of g which leaves S free, i.e. $C_i = *$ for all $i \in S$.

Similar to the case of ordinary decision trees in the previous subsection, it will be convenient to work with an equivalent property arising from certain distributions on the 0-inputs and 1-inputs of the gadget. The following definition is a slight generalization of balanced functions considered in [11].

▶ **Definition 13** (affine balanced functions)**.** A function $g : \{0,1\}^m \to \{0,1,*\}$ is p-affine balanced if for each $b \in \{0,1\}$, there is a distribution μ_b supported on $g^{-1}(b)$ such that for each $i \in [m]$, there exist distributions A_b^i on $\{0,1\}^m$ and B_b^i on $\{0,1\}^{m-1}$ such that μ_b can be written as the mixture $\mu_b = (1-2p)A_b^i + (2p)B_b^i \times U_1$. Here U_1 is a uniform random bit independent of B^i and we think of U_1 as the bit z_i and B_b^i as assigning bits in $z_{[m]\setminus\{i\}}$.

The above definition says we can sample z from μ_b in the following way:
- With probability $1-2p$, sample $z \sim A_b^i$.
- With probability $2p$, set z_i uniformly at random, and independently $z_{[m]\setminus\{i\}} \sim B_b^i$.

▶ **Observation 14.** *If $g : \{0,1\}^m \to \{0,1,*\}$ is k-stifled, then it is $k/2m$-affine balanced.*

Proof. The distribution μ_b witnessing that g is $k/(2m)$-affine balanced is defined in the following way. Select a subset S of $[m]$ of size k uniformly at random. Fix the bits outside S to a domain b-certificate for g (which can be done by the assumption that g is k-stifled). Finally for each $i \in S$, pick x_i independently and uniformly at random. Each $i \in [m]$ is included in S with probability k/m and conditioned on being included in the first step, it is fixed to each $c \in \{0,1\}$ with probability $1/2$ independent of the other bits. ◀

We now prove the lifting theorem for randomized PDTs. In the proof below, we do not give a truly online simulation but after each query, we simplify the PDT being simulated. This is primarily done to make it easy to verify correctness and analyze the number of queries made. We could alternatively have given a simulation closer to [16, 6] by keeping a list of parity queries made during the simulation.

▶ **Proposition 15.** Let $g : \{0,1\}^m \to \{0,1,*\}$ be a p-affine balanced function. For any relation $\mathcal{R} \subseteq \{0,1\}^n \times \mathcal{O}$,

$$\overline{\mathsf{RRank}_\epsilon^{\oplus\text{-}\mathsf{dt}}}(\mathcal{R} \circ g) \geq p\overline{\mathsf{R}_\epsilon^{\mathsf{dt}}}(\mathcal{R}).$$

Proof. Let $\mathcal{T}$ be a randomized parity decision tree computing $\mathcal{R} \circ g$. For the induction below, it will be convenient to allow each parity query to also involve a constant term. Let μ_0 and μ_1 be distributions on $g^{-1}(0)$ and $g^{-1}(1)$ respectively showing that g is p-affine balanced. For each $x \in \{0,1\}^n$, define μ^x to be the distribution on $(\{0,1\}^m)^n$ defined by independently sampling for each $i \in [n]$, the block z_i from μ_{x_i}. We will define a randomized decision tree $\mathcal{T}'$ computing $\mathcal{R}$ by simulating $\mathcal{T}$ on the distribution μ^x. The correctness of $\mathcal{T}'$ follows from the correctness of $\mathcal{T}$.

$\mathcal{T}'$ is defined in the following way. First sample a deterministic parity decision tree T from $\mathcal{T}$. We simulate it by a randomized decision tree in the following way. Suppose the query $\sum_{(i',j')\in S} z_{i',j'} + c$ at the root of T involves a variable $z_{i,j}$. Query the variable x_i and suppose it was $b \in \{0,1\}$. We will now set the variables in the block z_i according to the distribution μ_b in the following way. Recall that μ_b can be written as a mixture $(1-2p)A_b^j + 2pB_b^j \times U_1$ where A_b^j is a distribution on strings in $g^{-1}(b)$ and B_b^j is a distribution on domain b-certificates that leave the j^{th} bit free. We set z_i as follows:

1. With probability $1-2p$, set z_i according to A_b^j
2. With the remaining probability $2p$, set bits $z_{i,j'} (j' \neq j)$ according to the distribution B_j and independently "set" $\sum_{(i',j')\in S} z_{i',j'} = c$ for a uniform random bit c.

In the second case, even though the parity may depend on variables from blocks other than i, we may informally think of it equivalently as fixing $z_{i,j} = \sum_{(i',j')\in S\setminus\{(i,j)\}} z_{i',j'} + c$. Note that irrespective of the distribution of blocks other than i, it is indeed true that the above parity is equally likely to be 0 or 1 if we are the second case since $z_{i,j}$ is a uniform random bit (even

after conditioning on all the other blocks) which appears in the parity. Additionally, note that after conditioning on $z_{i,j} = \sum_{(i',j') \in S \setminus \{(i,j)\}} z_{i',j'} + c$ and the other $z_{i,j'}$, the distribution on all blocks other than i is still $\prod_{j \in [n] \setminus \{i\}} \mu_{x_j}$ since block i is independent of the rest.

Once we have set z_i as above (where possibly $z_{i,j}$ is a linear form depending on other blocks) we substitute them in the tree T and simplify appropriately. Specifically if any query node becomes a constant we remove it and directly attach the appropriate child to its parent. In particular, if we are in the second case, then the query at the root is set to a random c and we move to that child. Note that this simplification preserves the action of the tree T on the distribution μ^x when conditioned on the revealed z_i.

Since the distribution on the other blocks stays $\prod_{j \in [n] \setminus \{i\}} \mu_{x_j}$, we can now repeat this process with the query at the new root (which may be the same as the previous one if we were in case one, with all variables from block z_i removed). This is done until T has become just a leaf and we give the same output in $\mathcal{T}'$.

We now analyze the expected number of queries made by $\mathcal{T}'$ in simulating T and show that it is at most $\mathrm{rank}(T)/p$ on any input x. We will show by induction that a PDT of rank at most r which only depends on variables from at most l blocks is simulated using at most $Q(r,l) \coloneqq r/p$ queries in expectation.

Since a PDT with rank 0 or which does not depend on any blocks is just a leaf, the statement holds whenever $r = 0$ or $l = 0$. Suppose the statement holds for all pairs (r', l') with $l' < l$. Let T be a PDT of rank at most r and depending on at most l blocks. Suppose x_i is the first variable queried by the above simulation because of some $z_{i,j}$ appearing in the query at the root.

Consider what happens after we simplify the tree T based on the sampled z_i. In all cases, the rank of the resulting tree, say T_1 is at most r since the rank cannot increase by removing parts of the tree, and the number of blocks on which the tree depends has decreased by 1. Since case 2 while sampling z_i happens with probability $2p$, with probability at least p we go down the marked edge and the rank of T_1 is at most $r - 1$. Thus, the expected number of queries made in simulating T is at most

$$\begin{aligned} &1 + (1-p)\, Q(r, l-1) + p Q(r-1, l-1) \\ &\leq 1 + (1-p)\,\frac{r}{p} + p\frac{r-1}{p} = \frac{r}{p} \qquad \text{(by induction)} \end{aligned}$$

Since the simulation of a randomized PDT $\mathcal{T}$ corresponds to a distribution over trees simulating the deterministic PDTs T, we get that on any input x, the expected number of queries made is at most $\mathbb{E}_{T \sim \mathcal{T}}[\mathrm{rank}(T)/p]$ which is $\overline{\mathsf{RRank}}_{\epsilon}^{\oplus\text{-}\mathsf{dt}}(\mathcal{R} \circ g)/p$ if we take an optimal RPDT $\mathcal{T}$ for $\mathcal{R} \circ g$. ◀

▶ Remark 16. As pointed out by a reviewer, we can alternatively get a bound on the number of queries made during the simulation in Proposition 15 in the following way. Each time an actual query is made, with probability at least p, we go down the marked edge in the PDT being simulated, thereby contributing to the number of marked edges crossed. This implies that the expected number of marked edges crossed in the PDT when simulating it on the distribution μ^x is at least p times the expected number of queries made which gives what we wanted. This argument also shows that instead of considering expected rank of the PDT we could have considered the expected cost on the worst-case input.

Combining Observation 14 and Proposition 15, we get the following lifting theorem for randomized PDTs.

▶ **Theorem 17.** Suppose $g : \{0,1\}^m \to \{0,1,*\}$ is k-stifled. For any relation $\mathcal{R} \subseteq \{0,1\}^n \times \mathcal{O}$,

$$\log \overline{\mathsf{RSize}}_{\epsilon}^{\oplus\text{-}\mathsf{dt}}(\mathcal{R} \circ g) \geq \overline{\mathsf{RRank}}_{\epsilon}^{\oplus\text{-}\mathsf{dt}}(\mathcal{R} \circ g) \geq \frac{k}{2m} \overline{\mathsf{R}}_{\epsilon}^{\mathsf{dt}}(\mathcal{R}).$$

▶ Remark 18. The factor m loss in the lower bound in Theorem 17 is necessary at least when we allow g to be a partial function as the following example shows. We will take the outer function to be parity $\oplus_n$ and the inner function to be approximate majority $\mathrm{ApproxMAJ}_{m,k}$ which is defined as follows: $\mathrm{ApproxMAJ}(y)$ is 0 if $|y| \leq k$, 1 if $|y| \geq m - k$ and $*$ otherwise. Note that k here denotes the ends rather than the gap.

When $kn \leq m/4$ and $\epsilon = 1/4$, the lower bound from Theorem 17 is just a constant. For this regime of parameters, there is a PDT computing $\oplus_n \circ \mathrm{ApproxMAJ}_{m,k}$ which makes 1 parity query. For each block $i \in [n]$, pick $j_i \in [m]$ uniformly. For each $i \in [n]$, except with probability at most k/m, we have $x_{i,j_i} = \mathrm{ApproxMAJ}_{m,k}(x_i)$. The PDT simply outputs the parity of $(x_{1,j_1}, x_{2,j_2}, \ldots, x_{n,j_n})$. The error probability is at most $nk/m \leq 1/4$ by the union bound.

4 Parity decision tree lower bounds via $*$-depth

In this section, we prove the lifting theorems using $*$-depth, Theorems 3 and 4.

4.1 Reduction to deterministic $*$-depth and the Blocker-Certifier game

The proof of the lifting theorem for deterministic PDT size [16] implicitly contains a claim which reduces the task of proving lower bounds on PDT rank to the simpler task of proving lower bounds in a certain query model where one can only query one coordinate at a time but the input is a partial assignment instead of a binary string. This reduction works for all relations and, in particular, does not need the problem to be of composed form.

To describe the reduction, we need some definitions. Let $\mathcal{R} \subseteq \{0,1\}^n \times \mathcal{O}$ be a relation. Define the relation $\mathcal{R}_* \subseteq \{0,1,*\}^n \times \mathcal{O}$ as follows. For every $y \in \{0,1,*\}^n$,

$$\mathcal{R}_*(y) = \bigcup_{x \in \{0,1\}^n :\, x \in y} \mathcal{R}(x).$$

In words, $o \in \mathcal{O}$ is a correct output on y if there is some x consistent with y for which o is a correct output according to relation $\mathcal{R}$.

For a Boolean function $f : \{0,1\}^n \to \{0,1,*\}$, f_* has the following simple description. The input to the partial function $f_* : \{0,1,*\}^n \to \{0,1,*\}$ is promised to be a domain certificate of f and the goal is to output whether it is a 0-certificate or a 1-certificate. Recall that for a partial function f, for $\rho \in \{0,1,*\}^n$ to be a domain certificate, we require that the subcube corresponding to ρ is completely contained in the domain of f.

Similar to 0-depth and 1-depth for ordinary decision trees, we may define $*$-depth for decision trees on $\{0,1,*\}^n$. For a relation $\mathcal{R} \subseteq \{0,1,*\}^n \times \mathcal{O}$, let $\mathsf{D}^{*\text{-}\mathsf{dt}}(\mathcal{R})$ be the smallest number of $*$'s any deterministic decision tree (which is only allowed to query an index at a time) computing $\mathcal{R}$ must see in the worst case. Similarly let $\mathsf{R}_{\epsilon}^{*\text{-}\mathsf{dt}}(\mathcal{R})$ and $\overline{\mathsf{R}}_{\epsilon}^{*\text{-}\mathsf{dt}}(\mathcal{R})$ be the analogous randomized query complexity measures when computing $\mathcal{R}$ to error ϵ.

Since we mainly care about the $*$-depth of relations of the form $\mathcal{R}_*$, we now introduce a game capturing $\mathsf{D}^{*\text{-}\mathsf{dt}}(\mathcal{R}_*)$, called the Blocker-Certifier game. This is essentially obtained by specializing the usual Querier-Adversary game corresponding to decision tree depth to our setting. However, since the score only depends on the number of $*$'s, we may allow the Adversary to fix positions to 0 or 1 before they are queried (similar to some Delayer

strategies in the Prover-Delayer game) and then Querier only picks a coordinate to be fixed to $*$. In the game below, Blocker's role is similar to that of Querier (or Prover) and Certifier corresponds to an Adversary (or Delayer).

Let $\overline{\mathcal{R}}$ denote $(\{0,1\}^n \times \mathcal{O}) \setminus \mathcal{R}$, the complement of $\mathcal{R}$. The Blocker-Certifier game for $\mathcal{R}$ is played on a string $s \in \{0,1,*,\dagger\}^n$ which is initially $\dagger^n$. The game is played in rounds. In a round,

1. Certifier picks a subset $S \subseteq \{i \in [n] \mid s_i = \dagger\}$ (possibly empty) and for each $i \in S$, sets $s_i = b_i$ for some $b_i \in \{0,1\}$.
2. Blocker picks an $i \in [n]$ such that $s_i = \dagger$ and sets $s_i = *$.

The game ends when we have the following situation. There is some $o \in \mathcal{O}$, such that for every way of fixing the remaining $\dagger$'s in s to bits $\{0,1\}$, there is a way to fix $*$'s in s to bits such that the resulting string x satisfies $(x,o) \in \mathcal{R}$. In other words, the game has not ended if for every $o \in \mathcal{O}$, Certifier can fix the remaining $\dagger$'s to bits to get an o-certificate for $\overline{\mathcal{R}}$. Certifier's score is the number of $*$'s in s at the end of the game. The Blocker-Certifier value $\mathsf{BCval}(\mathcal{R})$ is the maximum score guaranteed by a Certifier strategy for the Blocker-Certifier game on $\mathcal{R}$. The equivalence between the Blocker-Certifier game and the usual Querier-Adversary game in this setting (or equivalently the definition of $*$-depth) is proved in the full version.

We can now relate PDT rank for a relation $\mathcal{R}$ and the $*$-depth of $\mathcal{R}_*$. In the proof below, we use a Certifier strategy in the Blocker-Certifier game to give a parity Delayer strategy, but the argument can also be expressed as a simulation.

▶ **Lemma 19** (implicit in [16]). For any relation $\mathcal{R}$,

$$\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(\mathcal{R}) \geq \mathsf{BCval}(\mathcal{R}) = \mathsf{D}^{*\text{-}\mathsf{dt}}(\mathcal{R}_*).$$

Proof. Suppose we have a Certifier strategy for the Blocker-Certifier game on $\mathcal{R}$ scoring r points. We will use this to give a Delayer strategy for the parity Prover-Delayer game on $\mathcal{R}$ achieving the same score.

Delayer essentially imitates the Certifier strategy by localizing the parity queries of Prover to the input z so that they may be treated as positions that have been touched by Blocker in the Blocker-Certifier game. Delayer will have fixed some bits in z to 0 or 1 while some positions would have been marked (denoted $*$) based on the queries made by Prover. Each such marked position corresponds to a linear equation $z_i = \sum_{i' \in S} z_{i'}$ coming from a parity query, where z_i is marked and none of the positions in S were marked at the time of the query.

We now explain this in detail. We will use x to denote the string in the Blocker-Certifier game. L will denote a collection of linear equations as explained above, which starts off empty. In the beginning, Delayer fixes bits in z exactly according to the move made by Certifier at the start of the Blocker-Certifier game. On a parity query $\sum_{i' \in S} z_{i'}$, Delayer first simplifies this parity query according to previously fixed bits to get a parity $b + \sum_{i' \in S'} z_{i'}$ where $b \in \mathbb{F}_2$ and all variables in S' are still free. If $S' = \emptyset$, then Delayer simply responds with b. Otherwise arbitrarily mark some $i \in S'$ and respond with $*$. Suppose Prover responds with $c \in \mathbb{F}_2$. Then the equation $z_i = b + c + \sum_{i' \in S' : i' \neq i} z_{i'}$ is added to L.

Next in the Blocker-Certifier game, Blocker sets x_i to $*$ to which Certifier responds by (possibly) fixing some other bits of x. As before, Delayer fixes the variables in z in the same way as x. Later queries of the Prover are handled in essentially the same way as before, but the simplification now also has to remove any marked variables appearing the query by substituting suitable parities using the appropriate equations in L.

We claim the Prover-Delayer game cannot end unless the corresponding Blocker-Certifier game is over. Suppose less than r variables have been marked so far. For every $o \in \mathcal{O}$, we will create an input z consistent with all the parity queries made so far such that $(z,o) \notin \mathcal{R}$.

Since fewer than r variables are set to $*$ in the Blocker-Certifier game, there is a way to fix the remaining free variables in x such that for all inputs y consistent with the obtained partial assignment x, $(y, o) \notin \mathcal{R}$. Now consider the string z obtained by fixing all the marked bits according to the equations in L. Since the marked variables are the pivots of these equations, such an extension indeed exists. By construction, this satisfies all the parity queries made so far and is consistent with the partial assignment x. Thus Delayer can always score at least r points. ◀

To get a lifting theorem for PDT rank, the above lemma can now be combined with a lower bound on the Blocker-Certifier value for composed problems. First, note that $\mathsf{D}^{*\text{-}\mathsf{dt}}((\mathcal{R} \circ g)_*) \geq \mathsf{D}^{*\text{-}\mathsf{dt}}(\mathcal{R} \circ g_*)$ since in the problem $\mathcal{R} \circ g_*$, we are only required to be correct when each block lies in the domain of g_*, i.e. is a domain certificate of g. To get the lifting theorem for any k-stifled gadget g, [16] use that $\mathsf{D}^{*\text{-}\mathsf{dt}}(\mathcal{R} \circ g_*) \geq k\mathsf{D}^{\mathsf{dt}}(\mathcal{R})$. This inequality is in the same spirit as $\mathsf{D}^{\mathsf{dt}}(\mathcal{R} \circ g) \geq \mathsf{D}^{\mathsf{dt}}(\mathcal{R})\mathsf{C}(g)$ or $\mathsf{DRank}^{\mathsf{dt}}(\mathcal{R} \circ g) \geq \mathsf{D}^{\mathsf{dt}}(\mathcal{R})(\mathsf{C}_{\mathsf{min}}(g) - 1)$. Similar to the case of decision tree depth or rank for composed problems, we can get an essentially tight lower bound on the $*$-depth of composed problems.

▶ **Lemma 20.** For any relation $\mathcal{R} \subseteq \{0,1\}^n \times \mathcal{O}$ and any function $g : \{0,1,*\}^m \to \{0,1,*\}$,

$$\mathsf{D}^{*\text{-}\mathsf{dt}}(\mathcal{R} \circ g) \geq \mathsf{D}^{\mathsf{dt}}(\mathcal{R})(\mathsf{D}^{*\text{-}\mathsf{dt}}(g) - 1).$$

This is proved in the same way as the usual composition theorem for deterministic (ordinary) decision tree depth [38, 40, 32]. For completeness, we sketch a proof of a more general composition theorem in Appendix B of the full version which implies Lemma 20.

Combining Lemmas 19 and 20, we obtain the following.

▶ **Theorem 21.** For any relation $\mathcal{R} \subseteq \{0,1\}^n \times \mathcal{O}$ and any function $g : \{0,1\}^m \to \{0,1,*\}$,

$$\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(\mathcal{R} \circ g) \geq \mathsf{D}^{\mathsf{dt}}(\mathcal{R})(\mathsf{D}^{*\text{-}\mathsf{dt}}(g_*) - 1).$$

The unique disjointness function UDISJ_{2m} is an example of a function for which the Blocker-Certifier value is much larger than how stifled it is. Recall that $\mathrm{UDISJ}_{2m} : (\{0,1\}^2)^m \to \{0,1,*\}$ is the partial function such that $\mathrm{UDISJ}(x_1, x_2, \ldots, x_m) = \vee_{i \in [m]}(x_{i1} \wedge x_{i2})$ where we are promised that there is at most one $i \in [m]$ such that $x_{i1} \wedge x_{i2} = 1$. This is a subfunction of both inner product and disjointness. It is easy to see that UDISJ is not 2-stifled, since there is no 0-certificate which leaves both x_{11} and x_{12} free.

On the other hand, there is a simple Certifier strategy in the Blocker-Certifier game[3] for UDISJ_{2m} which achieves score m. Initially Certifier does not fix any bits. Suppose Blocker sets $x_{i1} = *$ (the case $x_{i2} = *$ is analogous). Then Certifier responds by setting $x_{i2} = 0$ to ensure that $x_{i1} \wedge x_{i2} = 0$. Certifier follows this strategy until the end of the game ensuring that for each i, either both x_{i1}, x_{i2} are unset or at least one of them is fixed to 0. We claim that the game cannot end before m rounds. Indeed if fewer than m rounds have taken place, then there is some $i \in [m]$ such that $x_{i1} = x_{i2} = \dagger$. Moreover Certifier's strategy ensures that wherever this is not the case, we have $x_{i1} = 0$ or $x_{i2} = 0$. Therefore, for any $b \in \{0,1\}$, by setting $x_{i1} = x_{i2} = b$ and all other unset bits to 0, we obtain a domain b-certificate for UDISJ which is consistent with all the moves made so far.

[3] This strategy is essentially from discussion after the talk [39], but we have been unable to recognize who suggested it.

4.2 Reduction to randomized $*$-depth

The following lemma is the randomized analogue of Lemma 19.

▶ **Lemma 22.** For any relation $\mathcal{R} \subseteq \{0,1\}^n \times \mathcal{O}$,

$$\overline{\mathsf{RRank}}_\epsilon^{\oplus\text{-dt}}(\mathcal{R}) \geq \frac{1}{2}\overline{\mathsf{R}}_\epsilon^{*\text{-dt}}(\mathcal{R}_*).$$

Proof. Let $\mathcal{T}$ be a randomized PDT computing $\mathcal{R}$ to error ϵ. We will give a randomized decision tree $\mathcal{T}'$ for computing $\mathcal{R}_*$ with expected $*$-depth at most $2\,\overline{\mathrm{rank}}(\mathcal{T})$. To do this, on any input $z \in \{0,1,*\}^n$, we will simulate $\mathcal{T}$ on the distribution μ_z which is the uniform distribution over all strings in the subcube defined by z.

By the definition of $\mathcal{R}_*$, $\mathcal{T}'$ makes an error only when $\mathcal{T}$ makes an error on the input in μ_z being simulated. Therefore,

$$\Pr[\mathcal{T}' \text{ makes an error on input } z] = \mathbb{E}_{x\sim\mu_z}[\Pr_{T\sim\mathcal{T}}[T \text{ makes an error on } x]] \leq \epsilon.$$

We now describe how $\mathcal{T}'$ simulates $\mathcal{T}$. First sample a deterministic PDT $T \sim \mathcal{T}$. The tree will keep track of a list L of linear equations $x_i = \langle \alpha_i, x\rangle$, one for each x_i that has already been queried. The linear form on the right hand side of any such linear equation does not depend on any of the variables that have previously been queried. From this description, it is clear that these equations are linearly independent. Moreover, for each such i, if $z_i \in \{0,1\}$, the corresponding equation in L is exactly $x_i = z_i$. We will additionally maintain the invariant that the system L is equivalent to the system defined by all the parities from the root to the current node when combined with equations $x_i = z_i$ for all z_i which have already been queried and are not $*$.

Starting at the root of T, the tree performs the following steps until a leaf is reached in T.

1. Let $\sum_{i\in S'} x_i$ (for some $S' \subseteq [n]$) be the query at the current node v of T. Iteratively perform substitutions using the equations in L until the query has been simplified to $c + \sum_{i\in S} x_i$ which does not contain any variables that have already been queried and $c \in \mathbb{F}_2$. Set $U = \emptyset$ which will later store which $i \in S$ have already been queried.
2. Repeat the following
   - Pick any $i \in S \setminus U$ and query z_i. If $z_i = *$, go to step 3(a). Otherwise add i to U, and $x_i = z_i$ to L.

   until all $x_i, i \in S$ have been queried. When this happens, go to step 3(b).
3. **a.** ($z_i = *$) Pick $b \in \{0,1\}$ uniformly at random. Move to the child of v corresponding to $c + \sum_{j\in S} x_j = b$. Add to L the equation $x_i = c + b + \sum_{j\in U} z_j + \sum_{j\in S\setminus(U\cup\{i\})} x_j$.
   b. (none of z_i, $i \in S$ is $*$) Since all z_i's in the parity have been determined, move to the appropriate child $c + \sum_{j\in S} x_i = c + \sum_{j\in S} z_i$.

The output is the same as the label of the leaf reached in T.

Lemma 23 (proved later) shows that each leaf of T is reached with the correct probability according to the distribution μ_z. As argued earlier, this implies the correctness of the tree.

We now analyze the expected $*$-depth of the above randomized decision tree simulating T. Note that in each round, the tree sees one $*$ if we reach step 3(a) and otherwise no $*$'s. We will keep track of the number of marked edges seen as a measure of progress. In step 3(b), the number of $*$'s seen has not changed this round. On the other hand, in step 3(a), since we move to a random child, with probability at least $1/2$ we move down the marked edge in T. Thus, in expectation, after 2 $*$'s, we move down a marked edge in T. By linearity of expectation, after at most $2\,\mathrm{rank}(T)$ $*$-queries in expectation, a leaf is reached since the maximum number of marked edges on any root to leaf path is $\mathrm{rank}(T)$. ◀

▶ **Lemma 23.** Let T be a deterministic PDT on $\{0,1\}^n$. Let $z \in \{0,1,*\}^n$. Let $W_z(v)$ be the event that node v of T is visited by the randomized procedure described in the proof of Lemma 22 when run on input z. Let $V_z(v)$ be the event that for a random $x \in \mu_z$, running T on x reaches v. Then for every $z \in \{0,1,*\}^n, v \in T$, we have $\Pr[W_z(v)] = \Pr[V_z(v)]$.

Proof. Fix $z \in \{0,1,*\}^n$. The proof is by induction on the depth of v. The statement holds when v is the root since in this case, $\Pr[W_z(v)] = \Pr[V_z(v)] = 1$.

Now suppose v has depth at least 1. Let w be its parent. If $\Pr[W_z(w)] = 0$, then by induction $\Pr[V_z(w)] = 0$ and, therefore, $\Pr[V_z(v)] = \Pr[W_z(v)]$. Hence, we may assume that $\Pr[W_z(w)] = \Pr[V_z(w)] > 0$. We can write $\Pr[V_z(v)] = \Pr[V_z(w)] \Pr[V_z(v) \mid V_z(w)]$ and $\Pr[W_z(v)] = \Pr[W_z(w)] \Pr[W_z(v) \mid W_z(w)]$. So it suffices to prove that $\Pr[V_z(v) \mid V_z(w)] = \Pr[W_z(v) \mid W_z(w)]$.

Let L_w be the list L of equations at the begin of the round where the current node is w during the execution of the decision tree simulation with z as the input string. Let Q_w be the set of all z_i that were queried before reaching w and which are not $*$. Let $\langle \alpha, x \rangle = \sum_{j \in S'} x_j$ be the parity query at w and let b be such that v is the child corresponding to $\sum_{j \in S} x_j = b$. So $\Pr[V_z(v) \mid V_z(w)]$ is the probability that for a random $x \sim \mu_z$, $\sum_{j \in S'} x_j = b$ conditioned on all the equations from the root to w being satisfied. Note that since $x_i = z_i$ whenever $z_i \in \{0,1\}$, we may additionally condition on any subset of these fixed x_i's being the corresponding z_i's. By the invariants, the system of equations L_w is equivalent to the system containing equations describing the parities from the root to w as well as the queries made to z_i so far. Therefore, by abuse of notation, we may express $\Pr[V_z(v) \mid V_z(w)]$ as $\Pr[\sum_{j \in S'} x_j = b \mid L_w]$ where we view L_w as the event that all equations in L_w hold. Moreover under L_w, the parity $\sum_{j \in S'} x_j$ is equal to $c + \sum_{j \in S} x_j$ for some S as in step 1 of the round. So we have $\Pr[V_z(v) \mid V_z(w)] = \Pr_{x \sim \mu_z}[c + \sum_{j \in S} x_j = b \mid L_w]$.

Now we only need to verify that when simulating the query at node w we go to v with the correct probability $\Pr_{x \sim \mu_z}[c + \sum_{j \in S} x_j = b \mid L_w]$. There are two cases to consider:

1. For all $i \in S$, we have $z_i \in \{0,1\}$. In this case, all z_i are queried and step 3(b) is executed in the round starting at w. So we move to the correct child with probability $1 = \Pr_{x \sim \mu_z}[c + \sum_{j \in S} x_j = c + \sum_{j \in S} z_j \mid L_w]$.
2. Step 3(a) is executed in the round starting at w. In this case, there is some $i \in S$ such that $z_i = *$. Since $c + \sum_{j \in S} x_j$ is independent of L_w by construction, $\Pr_{x \sim \mu_z}[c + \sum_{j \in S} x_j = b \mid L_w] = 1/2$.

This finishes the proof. ◀

We now combine Lemma 22 with a composition theorem for randomized $*$-depth. Recall that a tight composition theorem does not hold in general for ordinary decision trees when composing a relation with a partial function and so we cannot have such a statement for $*$-depth (by lifting with, say, $(\mathrm{MAJ}_3)_*$). However, we can still adapt known randomized composition theorems to the setting of $*$-depth. In Appendix B of the full version, we adapt the composition theorem of [8] to prove a composition theorem for a general class of decision trees. This composition theorem provides the best dependence on the inner function up to some loss by a constant multiplicative factor and an additive constant.

▶ **Lemma 24** (following [8])**.** For any relation $\mathcal{R} \subseteq \{0,1\}^n \times \mathcal{O}$, any function $g : \{0,1,*\}^m \to \{0,1,*\}$,

$$\overline{\mathsf{R}^{*\text{-}\mathsf{dt}}_\epsilon}(\mathcal{R} \circ g) \geq \Omega(\overline{\mathsf{R}^{\mathsf{dt}}_\epsilon}(\mathcal{R})(\mathsf{LR}^*(g) - O(1))).$$

Combining Lemmas 22 and 24, we get the following.

▶ **Theorem 25.** For any relation $\mathcal{R} \subseteq \{0,1\}^n \times \mathcal{O}$, any function $g : \{0,1\}^n \to \{0,1,*\}$,

$$\overline{\mathsf{RRank}_\epsilon^{\oplus\text{-}\mathsf{dt}}}(\mathcal{R} \circ g) \geq \Omega(\overline{\mathsf{R}_\epsilon^{\mathsf{dt}}}(\mathcal{R})(\mathsf{LR}^*(g_*) - O(1))).$$

Using Theorem 25 or some other related composition theorem, we can show that, for instance, when the inner function is UDISJ or IND, then the obvious upper bound on PDT rank of $\mathcal{R} \circ g$ is optimal. Note that both IP_{2m} and DISJ_{2m} are total functions extending UDISJ_{2m}. So the lower bound for UDISJ also implies the lower bounds for inner product and disjointness in Corollary 5.

▶ **Corollary 26.** For any relation $\mathcal{R} \subseteq \{0,1\}^n \times \mathcal{O}$, for any $m \geq 2$,

$$\overline{\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}}(\mathcal{R} \circ \mathrm{UDISJ}_{2m}) = \Theta(\overline{\mathsf{R}^{\mathsf{dt}}}(\mathcal{R})m),$$
$$\overline{\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}}(\mathcal{R} \circ \mathrm{IND}_{m+2^m}) = \Theta(\overline{\mathsf{R}^{\mathsf{dt}}}(\mathcal{R})m).$$

Proving the lower bound for UDISJ will suffice to prove it also for IND since $\mathrm{IND}_{2m+2^{2m}}$ contains UDISJ_{2m} as a subfunction. By Theorem 25, it is sufficient to prove $\mathsf{LR}^*(\mathrm{UDISJ}_*) \geq \Omega(m)$. Instead of showing this directly, we will show that the simpler quantity sabotage $*$-complexity of $(\mathrm{UDISJ}_{2m})_*$ is $\Omega(m)$.

Sabotage complexity was first defined by Ben-David and Kothari [9] for ordinary decision trees and here we consider its natural $*$-analogue. Sabotage complexity $\mathsf{R}^*_{\mathrm{sab}}(g)$ is the expected zero-error query complexity of the following task. Let $g : \{0,1,*\}^m \to \{0,1,*\}$ be a partial function. Given a string $z \in \{0,1,*,\dagger\}^m$, where we interpret $\dagger$ as representing that the coordinate is free, find a $\dagger$ in z under the promise that z is consistent with some 0-input x and some 1-input y. It can alternatively be characterized as $\mathsf{R}^*_{\mathrm{sab}}(g) = \max_\mu \min_T \mathbb{E}_{(x,y)\in\mu}[\mathrm{sep}^*_T(x,y)]$ [23, Theorem B.1], where μ varies over distributions on pairs in $g^{-1}(0) \times g^{-1}(1)$, T varies over deterministic decision trees solving g and $\mathrm{sep}^*_T(x,y)$ denotes the number of marked edges ($*$-queries) on the path from the root to the node v where x and y separate. More precisely, v is the unique node in T such that both x and y reach v but they disagree on the query made at node v.

The composition theorem using sabotage complexity [9] is straightforward to adapt to $*$-decision trees, $\overline{\mathsf{R}_\epsilon^{*\text{-}\mathsf{dt}}}(\mathcal{R} \circ g) \geq \overline{\mathsf{R}_\epsilon^{\mathsf{dt}}}(\mathcal{R})\mathsf{R}^*_{\mathrm{sab}}(g)$, so we omit it.

▶ **Lemma 27.** $\mathsf{R}^*_{\mathrm{sab}}((\mathrm{UDISJ}_{2m})_*) \geq \frac{m-1}{4}$.

Proof. For brevity, let h_m denote $(\mathrm{UDISJ}_{2m})_*$. The hard distribution μ_m is generated as follows. First sample $z \in (\{0,1,*,\dagger\}^2)^m$ in the following way. Pick $i \in [m]$ uniformly. Set z_i to $(1,\dagger)$ or $(\dagger,1)$ uniformly. For each $j \neq i$, independently set z_i to $(0,*)$ or $(*,0)$ uniformly. Finally, obtain x by replacing the $\dagger$ in z by 0 and y by replacing the $\dagger$ by 1. Observe that after conditioning on $i \neq m$, the distribution on $z_1 \dots z_{m-1}$ is exactly what we would get if we performed the above procedure for $m-1$. This will let us use induction.

Let $l(m) = \min_T \mathbb{E}_{\mu_m}[\mathrm{sep}^*_T(x,y)]$. We will show that $l(m) \geq \frac{m-1}{4}$ by induction. The base case $m = 1$ is clear. For the induction step, we start by making some simplifying assumptions about T since we only care about the cost of T on the distribution μ. Since our distribution is invariant under permuting blocks and permuting bits within a block, we may assume that the query at the root in T is w_{m1} (we use w to denote the queries in T to avoid confusion with x, y, z). In the subtree where $w_{m1} = 0$, for any query to w_{m2}, we remove it and directly attach its parent to the subtree where $w_{m2} = *$. We do the same with the roles of 0 and $*$ interchanged. Note that this does not affect correctness of T on μ since in any pair (x,y) in the support of μ, if $x_{m1} = y_{m1} = 0$, then also $x_{m2} = y_{m2} = *$, and similarly the other way around. Also the cost of T does not increase by performing this simplification.

By the observation above, since the distribution μ conditioned on $i \neq m$ is identical to μ_{m-1}, the subtrees where $x_{m1} = 0$ and $x_{m1} = *$ give trees solving the separation task on the distribution μ_{m-1}. Therefore, we have the recurrence,

$$l(m) \geq \frac{m-1}{m}\left(\frac{1}{2} + l(m-1)\right),$$

which by induction gives $l(m) \geq \frac{m-1}{4}$. ◀

Proof of Corollary 26. The upper bounds follow from simulating a randomized decision tree T for $\mathcal{R}$ by using a deterministic tree for the inner function at each node of T.

For the lower bounds, as stated earlier, a lower bound for $\mathcal{R} \circ \mathrm{UDISJ}_{2m}$ also implies the same lower bound for $\mathcal{R} \circ \mathrm{IND}_{2m+2^{2m}}$. So we only need to show the lower bound for $\mathcal{R} \circ \mathrm{UDISJ}_{2m}$. By combining $\overline{\mathsf{R}^{*\text{-}\mathsf{dt}}_{\epsilon}}(\mathcal{R} \circ g) \geq \overline{\mathsf{R}^{\mathsf{dt}}_{\epsilon}}(\mathcal{R})\mathsf{R}^{*}_{\mathrm{sab}}(g)$ and Lemma 27, we get $\overline{\mathsf{R}^{*\text{-}\mathsf{dt}}_{\epsilon}}(\mathcal{R} \circ (\mathrm{UDISJ}_{2m})_{*}) = \Omega(\overline{\mathsf{R}^{\mathsf{dt}}_{\epsilon}}(\mathcal{R})m)$. Now using this with Lemma 22, we get $\overline{\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}_{\epsilon}}(\mathcal{R} \circ g) = \Omega(\overline{\mathsf{R}^{\mathsf{dt}}_{\epsilon}}(\mathcal{R})m)$. ◀

▶ Remark 28. The simulation using stifling gadgets, Theorem 17, can be understood as using the fact that for a k-stifled function g, $\mathsf{R}^{*}_{\mathrm{sab}}(g_{*}) \geq k/m$. Indeed, if we couple the distributions of certificates underlying μ_0 and μ_1 used in that proof according to the set of coordinates that are fixed, any decision tree correctly computing g_{*} must see a $*$ on the first query with probability k/m.

References

1 Scott Aaronson, Shalev Ben-David, and Robin Kothari. Separations in query complexity using cheat sheets. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 863–876, 2016. doi:10.1145/2897518.2897644.

2 Yaroslav Alekseev, Yuval Filmus, and Alexander Smal. Lifting Dichotomies. In Rahul Santhanam, editor, *39th Computational Complexity Conference (CCC 2024)*, volume 300 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 9:1–9:18, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CCC.2024.9.

3 Yaroslav Alekseev and Dmitry Itsykson. Lifting to bounded-depth and regular resolutions over parities via games. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*, STOC '25, pages 584–595, New York, NY, USA, 2025. Association for Computing Machinery. doi:10.1145/3717823.3718150.

4 Anurag Anshu, Dmitry Gavinsky, Rahul Jain, Srijita Kundu, Troy Lee, Priyanka Mukhopadhyay, Miklos Santha, and Swagato Sanyal. A composition theorem for randomized query complexity. In *37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2017)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.FSTTCS.2017.10.

5 Andrew Bassilakis, Andrew Drucker, Mika Göös, Lunjia Hu, Weiyun Ma, and Li-Yang Tan. The power of many samples in query complexity. In *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.9.

6 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)*, volume 251 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:17, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ITCS.2023.14.

7 Shalev Ben-David and Eric Blais. A new minimax theorem for randomized algorithms. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 403–411. IEEE, 2020.

8 Shalev Ben-David, Eric Blais, Mika Göös, and Gilbert Maystre. Randomised composition and small-bias minimax. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 624–635. IEEE, 2022. doi:10.1109/FOCS54457.2022.00065.

9 Shalev Ben-David and Robin Kothari. Randomized query complexity of sabotaged and composed functions. *Theory of Computing*, 14(5):1–27, 2018. doi:10.4086/toc.2018.v014a005.

10 Tyler Besselman, Mika Göös, Siyao Guo, Gilbert Maystre, and Weiqiang Yuan. Direct sums for parity decision trees. In *40th Computational Complexity Conference (CCC 2025)*, Leibniz International Proceedings in Informatics (LIPIcs), Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CCC.2025.16.

11 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. doi:10.4230/LIPIcs.CCC.2024.23.

12 Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288(1):21–43, 2002. doi:10.1016/S0304-3975(01)00144-X.

13 Sourav Chakraborty, Chandrima Kayal, Rajat Mittal, Manaswi Paraashar, Swagato Sanyal, and Nitin Saurabh. On the composition of randomized query complexity and approximate degree. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.APPROX/RANDOM.2023.63.

14 Arkadev Chattopadhyay, Yuval Filmus, Sajin Koroth, Or Meir, and Toniann Pitassi. Query-to-communication lifting using low-discrepancy gadgets. *SIAM Journal on Computing*, 50(1):171–210, 2021. doi:10.1137/19M1310153.

15 Arkadev Chattopadhyay, Michal Kouckỳ, Bruno Loff, and Sagnik Mukhopadhyay. Simulation theorems via pseudo-random properties. *computational complexity*, 28:617–659, 2019. doi:10.1007/S00037-019-00190-7.

16 Arkadev Chattopadhyay, Nikhil S Mande, Swagato Sanyal, and Suhail Sherif. Lifting to parity decision trees via stifling. In *14th Innovations in Theoretical Computer Science Conference (ITCS 2023)*. Schloss Dagstuhl – Leibniz Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ITCS.2023.33.

17 Arjan Cornelissen, Nikhil S Mande, and Subhasree Patro. Improved quantum query upper bounds based on classical decision trees. In *42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2022)*. Schloss Dagstuhl – Leibniz Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.FSTTCS.2022.15.

18 Yogesh Dahiya. *Exploring Size Complexity and Randomness in the Query Model*. PhD thesis, HBNI, 2024. URL: https://www.imsc.res.in/xmlui/handle/123456789/881.

19 Yogesh Dahiya and Meena Mahajan. On (simple) decision tree rank. *Theoretical Computer Science*, 978:114177, 2023. doi:10.1016/J.TCS.2023.114177.

20 Klim Efremenko, Michal Garlík, and Dmitry Itsykson. Lower bounds for regular resolution over parities. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 640–651, 2024. doi:10.1145/3618260.3649652.

21 Andrzej Ehrenfeucht and David Haussler. Learning decision trees from random examples. *Information and Computation*, 82(3):231–246, 1989. doi:10.1016/0890-5401(89)90001-1.

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

23 Dmytro Gavinsky, Troy Lee, Miklos Santha, and Swagato Sanyal. Optimal composition theorem for randomized query complexity. *Theory of Computing*, 19(1):1–35, 2023. doi:10.4086/TOC.2023.V019A009.

24 Mika Göös, TS Jayram, Toniann Pitassi, and Thomas Watson. Randomized communication versus partition number. *ACM Transactions on Computation Theory (TOCT)*, 10(1):1–20, 2018. doi:10.1145/3170711.

25 Mika Göös, Pritish Kamath, Toniann Pitassi, and Thomas Watson. Query-to-communication lifting for p np. *computational complexity*, 28:113–144, 2019. doi:10.1007/S00037-018-0175-5.

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

27 Mika Goos, Toniann Pitassi, and Thomas Watson. Deterministic communication vs. partition number. *SIAM Journal on Computing*, 47(6):2435–2450, 2018. doi:10.1137/16M1059369.

28 Mika Göös, Toniann Pitassi, and Thomas Watson. Query-to-communication lifting for bpp. *SIAM Journal on Computing*, 49(4):FOCS17–441, 2020. doi:10.1137/17M115339X.

29 Dmitry Itsykson and Dmitry Sokolov. Resolution over linear equations modulo two. *Annals of Pure and Applied Logic*, 171(1):102722, 2020. doi:10.1016/J.APAL.2019.102722.

30 Rahul Jain, Hartmut Klauck, and Miklos Santha. Optimal direct sum results for deterministic and randomized decision tree complexity. *Information Processing Letters*, 110(20):893–897, 2010. doi:10.1016/J.IPL.2010.07.020.

31 Shachar Lovett, Raghu Meka, Ian Mertz, Toniann Pitassi, and Jiapeng Zhang. Lifting with Sunflowers. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*, volume 215 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 104:1–104:24, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ITCS.2022.104.

32 Ashley Montanaro. A composition theorem for decision tree complexity. *Chicago Journal of Theoretical Computer Science*, 2014(6), 2014. doi:10.4086/cjtcs.2014.006.

33 Toniann Pitassi and Robert Robere. Lifting nullstellensatz to monotone span programs over any field. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1207–1219, 2018. doi:10.1145/3188745.3188914.

34 Vladimir Podolskii and Alexander Shekhovtsov. Randomized lifting to semi-structured communication complexity via linear diversity. In *16th Innovations in Theoretical Computer Science Conference (ITCS)*, LIPIcs. Schloss Dagstuhl, 2025. doi:10.4230/LIPIcs.ITCS.2025.78.

35 Pavel Pudlák and Russell Impagliazzo. A lower bound for dll algorithms for k-sat (preliminary version). In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 128–136, 2000. URL: http://dl.acm.org/citation.cfm?id=338219.338244.

36 Ran Raz and Pierre McKenzie. Separation of the monotone nc hierarchy. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 234–243. IEEE, 1997. doi:10.1109/SFCS.1997.646112.

37 Swagato Sanyal. Randomized query composition and product distributions. In *41st International Symposium on Theoretical Aspects of Computer Science (STACS 2024)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.STACS.2024.56.

38 Petr Savický. On determinism versus unambiquous nondeterminism for decision trees. Technical Report TR02-009, Electronic Colloquium on Computational Complexity (ECCC), 2002. URL: https://eccc.weizmann.ac.il//report/2002/009/.

39 Suhail Sherif. Lifting to parity decision trees via stifling (with applications to proof complexity). URL: https://www.youtube.com/watch?v=PeZVs6WUf-4.

40 Avishay Tal. Properties and applications of boolean function composition. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 441–454, 2013. doi:10.1145/2422436.2422485.

41 Alasdair Urquhart. The depth of resolution proofs. *Studia Logica*, 99:349–364, 2011. doi:10.1007/S11225-011-9356-9.