Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 202 (2024)

Lifting to Randomized Parity Decision Trees

Farzan Byramji [*]   Russell Impagliazzo [†]

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.

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 [RM97, GPW18, GKPW19, GPW20, CFKMP21, LMMPZ22]. These have several applications and have led to the resolution of some long-standing problems [RM97, GP18, GGKS18, CKLM19, PR18]. 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 [IS20]. Lifting theorems for deterministic parity decision trees using constant size gadgets

[*] University of California, San Diego. fbyramji@ucsd.edu. Supported by NSF Award AF: Medium 2212136.

[†] University of California, San Diego. rimpagliazzo@ucsd.edu. Supported by NSF Award AF: Medium 2212136.

ISSN 1433-8092

were recently proved by Chattopadhyay, Mande, Sanyal and Sherif [CMSS23] and independently by Beame and Koroth [BK23] 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($\oplus$) [EGI24, BCD24, AI25]. (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 [CMSS23] 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 1.1. 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 Appendix A for details). In their proof of an exponential separation between regular Res($\oplus$) and general Resolution, Bhattacharya, Chattopadhyay and Dvořák [BCD24] required a randomized PDT lifting theorem and so they used a lifting theorem for randomized communication complexity [CFKMP21]. 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 $\exp(\tilde{\Omega}(\sqrt[3]{n}))$ [EGI24] 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 [AFS24] (also [DM23]) 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 1.2. For any function $f : \{0,1\}^n \to \{0,1\}$, any 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 1.1 and 1.2 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 [CMSS23, BK23] and the simulation-based proof of the deterministic DT size lifting theorem [AFS24], 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 [EH89]. 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 [Dah24]. The corresponding question in the deterministic case has a positive answer, as shown by Dahiya and Mahajan [DM23].[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 [Sav02, Tal13, Mon14], 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 [GLSS23] and even for the composition of partial functions [BB20]. There is a long line of work [ABK16, BK18, AGJK+18, GJPW18, GLSS23, BDGH+20, BB20, BBGM22, CKMP+23, San24] 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 1.2 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 [BK18, GLSS23, BBGM22] can be adapted to give analogous composition theorems for randomized DT rank (see Appendix B).

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. [CMSS23] 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 [CMSS23] 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 1.3. For all functions $f : \{0,1\}^n \to \{0,1\}, g : \{0,1\}^m \to \{0,1\}$,

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

[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 [AFS24]. 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 Section 6.

This is inspired by discussion at the end of the talk [She23], 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 $\mathsf{D}^{*\text{-dt}}(g_*) > k$ 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 $\mathsf{D}^{*\text{-dt}}(g_*) = m$, which shows that this bound can sometimes be better.

We also prove a randomized analogue of Theorem 1.3.

Theorem 1.4. For all functions $f : \{0,1\}^n \to \{0,1\}, g : \{0,1\}^m \to \{0,1\}$,

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

Here LR^* is the natural $*$ analogue of the linearized complexity measure introduced in [BBGM22] 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 $\mathsf{RRank}^{\oplus\text{-dt}}(f \circ g) \leq O(\mathsf{R}^{\mathsf{dt}}(f)\mathsf{RRank}_{\epsilon}^{\oplus\text{-dt}}(g))$ 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 $O(\mathsf{R}^{\mathsf{dt}}(f)\mathsf{RRank}^{\oplus\text{-dt}}(g))$.

Corollary 1.5. For $m \geq 2$, for all functions $f : \{0,1\}^n \to \{0,1\}$,

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

While Theorems 1.3 and 1.4 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. However, the description of the query model for affine subspaces requires some setup, so we defer the precise definitions and statements to Section 5.

Theorems 1.3 and 1.4 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 [AFS24] completely classified gadgets based on what gadgets allow polynomial lifting, $\log \mathsf{DSize}^{\oplus\text{-dt}}(f \circ g) = \Omega_g(\mathsf{D}^{\mathsf{dt}}(f)^{\epsilon})$ (for some $\epsilon > 0$). However, this does not answer the question of which gadgets allow linear lifting, $\log \mathsf{DSize}^{\oplus\text{-dt}}(f \circ g) = \Omega_g(\mathsf{D}^{\mathsf{dt}}(f))$. We observe by considering a mild generalization of stifling that for any gadget g whose minimum parity certificate complexity is at least 2, $\mathsf{DRank}^{\oplus\text{-dt}}(f \circ g) \geq \mathsf{D}^{\mathsf{dt}}(f)$. This can be seen as the natural

[2] Strictly speaking, Theorem 1.4 does not seem to directly imply Theorem 1.1 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 1.1 (see Remark 4.10).

parity analogue of the statement that if g has minimum certificate complexity at least 2, then for all f, $\mathsf{DRank}^{\mathsf{dt}}(f \circ g) \geq \mathsf{D}^{\mathsf{dt}}(f)$ [AFS24, DM23].

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 1.6. 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 [CMSS23]. 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 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 [BCD24]. 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 [CMSS23] 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, 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 1.3. The simulation in [CMSS23] 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 [PS25] 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 [PS25] 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. [BGGMY25] 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 [JKS10].

These direct sum results are incomparable to those of [BGGMY25]. 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 [San24]. Similarly, in the deterministic case, the NAND tree also shows that deterministic $*$-depth can sometimes be quadratically larger than (parity) certificate complexity.

Organization. In Section 2, we state definitions for the query complexity measures used. In Section 3, we prove Theorems 1.1 and 1.2. In Section 4, we prove Theorems 1.3 and 1.4. In Section 5, we prove strengthenings of results in Section 4 by considering decision trees for affine subspaces and some observations comparing stifling-related measures. In Section 6, we prove Proposition 1.6 and give a counterexample to a stronger version of a conjecture of Alekseev, Filmus and Smal [AFS24]. In Appendix A, we discuss applications to subsystems of $\mathrm{Res}(\oplus)$. In Appendix B, we adapt known composition theorems to block decision trees, which generalize all query models considered here, other than PDTs. These composition theorems are used in Sections 4 and 5.

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 [BdW02] 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 [CMP22] but the idea also appears implicitly in prior work. For such a marked tree T, we will use $\mathrm{cost}(T, x)$ 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 $\mathrm{cost}(\mathcal{T}, x) = \mathbb{E}_{T\sim\mathcal{T}}[\mathrm{cost}(T, x)]$. Define $\mathrm{cost}(\mathcal{T}) = \max_x \mathrm{cost}(\mathcal{T}, x)$. Similarly $\overline{\mathrm{rank}}(\mathcal{T}) = \mathbb{E}_{T\sim\mathcal{T}}[\mathrm{rank}(T)]$ and $\overline{\mathrm{size}}(\mathcal{T}) = \mathbb{E}_{T\sim\mathcal{T}}[\mathrm{size}(T)]$.

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, $\mathrm{rank}(T) \leq \log \mathrm{size}(T)$. This directly implies for a randomized decision tree $\mathcal{T}$, $\mathrm{rank}(\mathcal{T}) \leq \log \mathrm{size}(\mathcal{T})$ 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 $\mathbb{E}_{T\sim\mathcal{T}}[\mathrm{rank}(T)] \leq \mathbb{E}_{T\sim\mathcal{T}}[\log \mathrm{size}(T)] \leq \log \mathbb{E}_{T\sim\mathcal{T}}[\mathrm{size}(T)]$, which in our notation is $\overline{\mathrm{rank}}(\mathcal{T}) \leq \log \overline{\mathrm{size}}(\mathcal{T})$. 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 [PI00] 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 [DM23] 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 [IS20].

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 $C_i = *$. Abusing notation, 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 $\mathcal{R}$, $\mathsf{C}_{\mathsf{min}}(\mathcal{R})$, 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 $\{0,1\}^n$, $S = \{\langle \alpha_1, x\rangle = b_1, \langle \alpha_2, x\rangle = b_2, \dots, \langle \alpha_k, x\rangle = b_k\}$ 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 $\mathcal{R}$, $\mathsf{C}^{\oplus}_{\mathsf{min}}(\mathcal{R})$, 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 [AFS24] showed that resistant gadgets suffice for lifting decision tree depth to size by generalizing Urquhart's argument for the XOR gadget [Urq11]. A function g is said to be k-resistant if $\mathsf{C}_{\mathsf{min}}(f) \geq k+1$.

Theorem 3.1 ([AFS24, DM23])**.** 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 [DM23] 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 3.2. 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 1.2.

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

Definition 3.3 (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 3.4. 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 [AFS24]. However, it is analyzed differently for which presenting it as a simulation is more convenient.

Proposition 3.5. 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 $\overline{\mathrm{rank}}(\mathcal{T})/p$.

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 $\mathrm{rank}(T)/p$ 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 $i \in [\mathrm{rank}(T)]$, 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 $i \in [\mathrm{rank}(T)]$ by what was argued above.

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

Theorem 3.2 now follows from combining Observation 3.4 and Proposition 3.5.

3.2 Lifting to randomized parity decision tree rank

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

Definition 3.6 (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 [BCD24].

Definition 3.7 (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 3.8. 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 [CMSS23, BK23] by keeping a list of parity queries made during the simulation.

Proposition 3.9. 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 $\text{rank}(T)/p$ 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

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

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 3.10. As pointed out by a reviewer, we can alternatively get a bound on the number of queries made during the simulation in Proposition 3.9 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. Moreover, 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 3.8 and Proposition 3.9, we get the following lifting theorem for randomized PDTs.

Theorem 3.11. *Suppose $g : \{0,1\}^m \to \{0,1,*\}$ is k-stifled. For any relation $\mathcal{R} \subseteq \{0,1\}^n \times \mathcal{O}$,*

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

Remark 3.12. The factor m loss in the lower bound in Theorem 3.11 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 $|y| \le k$, 1 if $|y| \ge m - k$ and $*$ otherwise. Note that k here denotes the ends rather than the gap.

When $kn \le m/4$ and $\epsilon = 1/4$, the lower bound from Theorem 3.11 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 $nk/m \le 1/4$ by the union bound.

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

In this section, we prove the lifting theorems using $*$-depth, Theorems 1.3 and 1.4.

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

The proof of the lifting theorem for deterministic PDT size [CMSS23] 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 $y \in \{0,1,*\}^n$,

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

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 $\mathsf{D}^{*\text{-dt}}(\mathcal{R})$ 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 $\mathsf{R}_\epsilon^{*\text{-dt}}(\mathcal{R})$ and $\overline{\mathsf{R}_\epsilon^{*\text{-dt}}}(\mathcal{R})$ 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 $\mathsf{D}^{*\text{-dt}}(\mathcal{R}_*)$, 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 Lemma B.3.

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 4.1 (implicit in [CMSS23])**.** 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, [CMSS23] 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 4.2. 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 [Sav02, Tal13, Mon14]. For completeness, we sketch a proof of a more general composition theorem, Proposition B.2, in Appendix B which implies Lemma 4.2.

Combining Lemmas 4.1 and 4.2, we obtain the following.

Theorem 4.3. For any relation $\mathcal{R} \subseteq \{0,1\}^n \times \mathcal{O}$ and any function $g : \{0,1\}^m \to \{0,1,*\}$,

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

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 $\text{UDISJ}_{2m} : (\{0,1\}^2)^m \to \{0,1,*\}$ is the partial function such that $\text{UDISJ}(x_1, x_2, \ldots, x_m) = \vee_{i \in [m]}(x_{i1} \wedge x_{i2})$ 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.

4.2 Reduction to randomized $*$-depth

The following lemma is the randomized analogue of Lemma 4.1.

Lemma 4.4. 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 $2\,\overline{\text{rank}}(\mathcal{T})$. 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

[3] This strategy is essentially from discussion after the talk [She23], but we have been unable to recognize who suggested it.

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 4.5 (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 4.5. 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 4.4 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 4.4 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, we adapt the composition theorem of [BBGM22] to prove a composition theorem for a general class of decision trees, Theorem B.4. This composition theorem provides the best dependence on the inner function (see Theorem B.8) up to some loss by a constant multiplicative factor and an additive constant.

Lemma 4.6 (following [BBGM22])**.** For any relation $\mathcal{R} \subseteq \{0,1\}^n \times \mathcal{O}$, any function $g : \{0,1,*\}^m \to \{0,1,*\}$,

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

Combining Lemmas 4.4 and 4.6, we get the following.

Theorem 4.7. 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 4.7 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 1.5.

Corollary 4.8. 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 $\text{IND}_{2m+2^{2m}}$ contains UDISJ_{2m} as a subfunction. By Theorem 4.7, it is sufficient to show that $\mathsf{LR}^*(\text{UDISJ}_*) \geq \Omega(m)$. Instead of showing this directly, we will show that the simpler quantity sabotage $*$-complexity of $(\text{UDISJ}_{2m})_*$ is $\Omega(m)$.

Sabotage complexity was first defined by Ben-David and Kothari [BK18] for ordinary decision trees and here we consider its natural $*$-analogue. Sabotage complexity $\mathsf{R}^*_{\text{sab}}(g)$ 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 $\mathsf{R}^*_{\text{sab}}(g) = \max_\mu \min_T \mathbb{E}_{(x,y)\in\mu}[\text{sep}^*_T(x,y)]$ [GLSS23, 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 $\text{sep}^*_T(x,y)$ 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 [BK18] is straightforward to adapt to $*$-decision trees, $\overline{\mathsf{R}^{*\text{-dt}}_\epsilon}(\mathcal{R} \circ g) \geq \overline{\mathsf{R}^{\text{dt}}_\epsilon}(\mathcal{R})\mathsf{R}^*_{\text{sab}}(g)$, so we omit it.

Lemma 4.9. $\mathsf{R}^*_{\text{sab}}((\text{UDISJ}_{2m})_*) \geq \frac{m-1}{4}$.

Proof. For brevity, let h_m denote $(\text{UDISJ}_{2m})_*$. 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 $z_1 \ldots z_{m-1}$ is exactly what we would get if we performed the above procedure for $m-1$. This will let us use induction.

Let $l(m) = \min_T \mathbb{E}_{\mu_m}[\text{sep}^*_T(x,y)]$. 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 4.8. 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 $\mathcal{R} \circ \text{UDISJ}_{2m}$ also implies the same lower bound for $\mathcal{R} \circ \text{IND}_{2m+2^{2m}}$. So we only need to show the lower bound for $\mathcal{R} \circ \text{UDISJ}_{2m}$. By combining $\overline{\mathsf{R}^{*\text{-dt}}_\epsilon}(\mathcal{R} \circ g) \geq \overline{\mathsf{R}^{\text{dt}}_\epsilon}(\mathcal{R})\mathsf{R}^*_{\text{sab}}(g)$ and Lemma 4.9, we get $\overline{\mathsf{R}^{*\text{-dt}}_\epsilon}(\mathcal{R} \circ (\text{UDISJ}_{2m})_*) = \Omega(\overline{\mathsf{R}^{\text{dt}}_\epsilon}(\mathcal{R})m)$. Now using this with Lemma 4.4, we get $\overline{\mathsf{RRank}^{\oplus\text{-dt}}_\epsilon}(\mathcal{R} \circ g) = \Omega(\overline{\mathsf{R}^{\text{dt}}_\epsilon}(\mathcal{R})m)$. □

Remark 4.10. The simulation using stifling gadgets, Theorem 3.11, can be understood as using the fact that for a k-stifled function g, $\mathsf{R}^*_{\mathsf{sab}}(g_*) \geq k/m$. 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.

5 More on parity decision trees for composed problems

5.1 Parity stifling and the parity Blocker-Certifier game

In this subsection, we start by giving a different way in which the lifting theorem of [CMSS23] can be slightly improved. In the original description of the simulation [CMSS23], a parity query is only localized to a block $i \in [n]$ instead of a single variable. Such a block is said to be marked by this parity query. It is only when a block has been marked k times that we consider which variables in this block can be chosen as pivots for the queries marking the block, and set the remaining according to a certificate using the k-stifled property of the gadget.

We observe that it is not necessary to only consider certificates which fix bits. Instead it suffices to have any parity certificate which is completely independent of the linear forms coming from these marked queries obtained by projecting onto the block i. We say that two linear subspaces A and B of $\mathbb{F}_2^m$ are independent if $A \cap B = \{0^m\}$. For two collections C and D of linearly independent linear forms, we say that C and D are independent if their respective spans are independent. In other words, there is no non-zero linear form which can be expressed both as a linear combination of forms from C and a linear combination of forms from D.

Definition 5.1. A function $g : \{0,1\}^m \to \{0,1,*\}$ is k-parity stifled if for every k-dimensional subspace $V \subseteq \mathbb{F}_2^n$ (viewed as part of the dual space, i.e. a collection of parities) and every $b \in \{0,1\}$, there is a domain b-parity certificate C such that C and V are independent. Here by C and V being independent, we actually mean that the linear forms involved in the constraints of C are independent of V.

If g is 1-parity stifled, then we sometimes drop the 1 and simply say that g is parity stifled. Note that a k-stifled function is also k-parity stifled by using the idea of localizing a basis for any k-dimensional subspace V to a set of k indices in $[n]$ and considering a certificate which only fixes the remaining $n-k$ bits. On the other hand, there exist 1-parity stifled functions which do not contain any subfunctions (when only restricting bits) which are 1-stifled. An example of such a function is $g(x_1, x_2, x_3) = x_1 \oplus (x_2 \wedge x_3)$.

We now argue that being parity stifled is also sufficient for lifting to PDTs by making a small change to the original proof of [CMSS23] for stifled gadgets. Since we will prove something stronger later, we only sketch the changes to the proof of [CMSS23] required for the following proposition.

Proposition 5.2. Let $g : \{0,1\}^m \to \{0,1,*\}$ be k-parity stifled. Then for all relations $\mathcal{R}$,

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

Proof sketch. After a block z_i has been marked k-times, a query is made to x_i. Since g is k-parity stifled, there exists a domain parity certificate which only fixes parities that do not lie in the span of the projections of the parities which marked z_i. The linear equations coming from this parity certificate are now added to the list of equations and used for simplifying any future parity queries. Since the added parities only depend on a particular block and are independent of the projections onto that block of the corresponding marked parities, the final matrix of coefficients corresponding to the collection of equations can be written as a block-triangular matrix where each block on the diagonal has linearly independent rows. This implies that there exists a solution to this system of equations. □

We now combine ideas of parity stifling and the Blocker-Certifier game to unify the lower bounds on PDT rank of composed functions discussed so far. Let $\mathcal{V}_m$ denote the collection of all affine subspaces of $\mathbb{F}_2^m$. We will define a model of decision trees for computing functions of affine subspaces of $\mathbb{F}_2^m$. Such a decision tree makes parity queries to learn whether these parities are free ($*$) or fixed to some $b \in \{0,1\}$. To properly define such a decision tree, we need the notion of a partial subspace.

Definition 5.3. A partial subspace V of $\mathbb{F}_2^m$ is defined by a pair (C, B) where C is a collection of linear equations (or constraints), B is a collection of linear forms and B and C are independent. We interpret V as being the collection of affine subspaces U such that U is consistent with C (every input in U satisfies C) and every $v \in \operatorname{span}(B)$ is free in U. We do not distinguish between different representations of the same partial subspace obtained by performing some invertible transformation on C or B.

We will assume that C and B do not contain any redundant constraints or linear forms, so the involved linear forms are always linearly independent.

Definition 5.4. A $(\oplus, *)$-decision tree is a decision tree for affine subspaces on $\mathbb{F}_2^m$ where each query is a parity and the set of inputs reaching a node is defined by some partial subspace (C, B). We will require that the parity query v at a node with the partial subspace (C, B) must be independent of the linear forms appearing in C and B, i.e. it cannot be expressed as a linear combination of the linear forms (defining the constraints) in C and the forms in B. Such a node has exactly $2 \cdot 2^{|B|} + 1$ children. One child corresponds to v being free, i.e. the partial subspace $(C, B \cup \{v\})$. The edge between the nodes (C, B) and $(C, B \cup \{v\})$ is labeled by $*$. For every $S \subseteq B$ and $b \in \{0,1\}$, we have the partial subspace $(C \cup \{v + \sum_{w \in S} w = b\}, B)$. The corresponding edge is labeled by the new constraint $v + \sum_{w \in S} w = b$.

Observe that these $2^{|B|+1} + 1$ partial subspaces indeed form a partition of (C, B). Also note that since each query increases the dimension of $C \cup B$, m queries always suffice to fully determine any affine subspace.

To prove a relation similar to Lemma 4.1, we first generalize the notion of a $(\oplus, *)$-decision tree to also apply to inputs in $\mathcal{V}_m^n$. Such a decision tree makes queries like an $(\oplus, *)$-decision tree on $\mathcal{V}_m$ but each query is contained in a single block. So each node now corresponds to an n-fold product of partial subspaces of $\mathcal{V}_m$ and each query only affects one of these n partial subspaces.

The $*$-depth of an $(\oplus, *)$-decision tree T is the maximum number of $*$'s on any root-to-leaf path in T. The $*$-depth of a relation $\mathcal{R} \subseteq \mathcal{V}_m^n \times \mathcal{O}$, denoted $\mathsf{D}^{\oplus,*\text{-dt}}(\mathcal{R})$ is the minimum $*$-depth of a $(\oplus, *)$-decision tree computing $\mathcal{R}$.

We now define for any relation $\mathcal{R}$ with a natural block structure on the input bits, a related problem whose $*$-depth will give a lower bound on the PDT rank of $\mathcal{R}$. For simplicity, we will assume below that these blocks have the same size m, but this is not required for the proof. Different partitions into blocks will lead to different relations but we will suppress the dependence on the partition, since the statements hold for all partitions (unless noted otherwise). Let $\mathcal{R} \subseteq (\{0,1\}^m)^n \times \mathcal{O}$. Define $\mathcal{R}_{\oplus,*} \subseteq \mathcal{V}_m^n \times \mathcal{O}$ as follows. For any input $U = (U_1, U_2, \ldots, U_n) \in \mathcal{V}_m^n$, we have $(U, o) \in \mathcal{R}_{\oplus,*}$ iff there is some $y \in (\{0,1\}^m)^n$ such that $(y, o) \in \mathcal{R}$ and y is consistent with U in the sense that for each $i \in [n]$, y_i is in U_i. Again $\mathsf{D}^{\oplus,*\text{-dt}}(\mathcal{R}_{\oplus,*})$ may be captured by a game, which is the parity analogue of the Blocker-Certifier game. In this game, Certifier picks some parities which are to be fixed to 0 or 1 and Blocker picks a parity contained in a block (independent from previously picked parities) to be left free in each round. The game ends when Blocker can announce an $o \in \mathcal{O}$ such that every $z \in \mathcal{V}_m^n$ consistent with the current partial subspace contains some $x \in (\{0,1\}^m)^n$ such that $(x, o) \in \mathcal{R}$. Define $\mathsf{BCval}^{\oplus}(\mathcal{R})$ to be the maximum score a parity Certifier strategy can guarantee in the parity Blocker-Certifier game for relation $\mathcal{R}$. We again have $\mathsf{BCval}^{\oplus}(\mathcal{R}) = \mathsf{D}^{\oplus,*\text{-}\mathsf{dt}}(\mathcal{R}_{\oplus,*})$ (see Lemma B.3 for a proof).

Proposition 5.5. Let $\mathcal{R} \subseteq (\{0,1\}^m)^n \times \mathcal{O}$. Then

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

This can be proved by lifting a Certifier strategy in the parity Blocker-Certifier game to a parity Delayer strategy for $\mathcal{R}$, but we phrase it as a simulation below.

Proof. Let T be a parity decision tree computing $\mathcal{R}$. We will give a $(\oplus, *)$-decision tree T' for $\mathcal{R}_{\oplus,*}$ of $*$-depth at most $\text{rank}(T)$.

To perform the simulation, for each $i \in [n]$, we will keep a set P_i of linear equations which correspond to the free parities B_i in the i^{th} partial subspace of the current node in the $(\oplus, *)$-decision tree. For each $v \in B_i$, there will be some equation $v = w + b$ in P_i where $b \in \mathbb{F}_2$ and w is a parity on $(\mathbb{F}_2^m)^n$ which only depends on variables from blocks $j > i$. Each such equation in P_i will be equivalent to a constraint coming from parity queries made in T combined with the constraints in C_i ($i \in [n]$), where C_i is the collection of fixed parities of the i^{th} partial subspace.

In the beginning, each P_i is empty. Suppose we are at a node in T with the parity query $v = \langle \alpha, z \rangle + b$. Clean it up in the following manner. For each $i \in [n]$,

- If $v|_i$ lies in the span of the linear forms in B_i and C_i, let $T_1 \subseteq B_i$ and $T_2 \subseteq C_i$ such that $v|_i = \sum_{w \in T_1} w + \sum_{w \in T_2} w$. Let $T_1' \subseteq P_i$ be the collection of equations corresponding to the forms in T_1. Update v by substituting $v|_i$ using the equations in T_1' and T_2. (Note that now $v|_i = 0$ and for $j < i$, $v|_j$ continue to be 0.)
- Otherwise, exit the loop.

At this point, if $v \in \mathbb{F}_2$, we simply move to the child corresponding to v without making any queries.

Otherwise, consider the smallest i such that some variable of z_i still appears in v. We make a query $v|_i$ in T'. If the response is $*$ (so that in the updated partial subspace, we have $v|_i \in B_i$), then in T, we move to the marked child, corresponding to say $c \in \mathbb{F}_2$ and add the equation $v|_i = (v - v|_i) + c$ to P_i.

Now suppose the response is $v|_i + v' = b$ where v' lies in the span of B_i. Then we apply the clean-up procedure again (with the updated C_i) and continue as above. Note that this process must terminate since with each clean-up phase at the same query, we are clearing away one block.

Once we are at a leaf of T, we give the same output in T'. It is easy to see that the cost of T' is at most $\text{rank}(T)$ since each time we see a $*$ in T', we move to the marked child in T. Let us verify that T' correctly solves $\mathcal{R}_{\oplus,*}$. Suppose the input to T' is $U = U_1 U_2 \dots U_n$. We claim that there is a string $y \in (\{0,1\}^m)^n$ consistent with U which is also present in the leaf of T which is reached by performing the above simulation with input U. To see this, we fix each block y_i in reverse order ($i = n, n-1, \dots, 1$) according to the equations in P_i and C_i. Some such y_i exists since B_i and C_i are independent, the projections of P_i onto block i give exactly B_i and each equation in P_i only depends on blocks $j \geq i$. Since U_i belongs to the partial subspace (C_i, B_i), we see that y_i is indeed consistent with U_i. Also note that y reaches the leaf reached during the simulation since we always ensure that every parity constraint on the root-to-leaf path is implied by the equations in $\bigcup_{i=1}^n P_i \cup C_i$. □

We may now combine this with a composition statement for $(\oplus, *)$-decision trees which follows from the general composition theorem for block decision trees, Proposition B.2, to get the following theorem.

Theorem 5.6. For any relation $\mathcal{R} \subseteq \{0,1\}^n \times \mathcal{O}$, any function $g : \{0,1\}^m \to \{0,1,*\}$,

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

By $\mathsf{BCval}^{\oplus}(g)$, we refer to the game when considering the trivial partition where all variables are contained in just one block.

It is easy to see that if g is k-parity stifled, then $\mathsf{BCval}^{\oplus}(g) \geq k + 1$ since any $(\oplus, *)$-decision tree for $g_{\oplus,*}$ must make see at least $k + 1$ $*$'s on the input which corresponds to the whole space $\mathbb{F}_2^m$. We now verify that the parity Blocker-Certifier game (on the partition into one block) is at least as hard as the Blocker-Certifier game (which can be thought of as the parity Blocker-Certifier game on the partition into singletons). This shows that Theorem 5.6 also implies Theorem 4.3.

Lemma 5.7. For any relation $\mathcal{R} \subseteq \{0,1\}^n \times \mathcal{O}$,

$$\mathsf{BCval}^{\oplus}(\mathcal{R}) \geq \mathsf{BCval}(\mathcal{R}).$$

Proof. Suppose T is a $(\oplus, *)$-decision tree solving $\mathcal{R}_{\oplus,*}$. We will give a decision tree T' for $\mathcal{R}_*$ whose $*$-depth is at most that of T. The simulation is similar to the one in Lemma 4.1. We will use $M \subseteq [n]$ to denote the set of all coordinates in z which have been revealed to be $*$. During the simulation, we will follow the root-to-leaf path in T which is taken by the subcube provided as input to T'. So we will ensure that in the current partial subspace (C, B), all constraints in C are implied by the fixed coordinates in the input $z \in \{0,1,*\}^n$ and that M gives a collection of pivots for the linear forms in B. This would imply correctness of our simulation. To see why each non-zero linear form in the span of B is free, simply note that every $v \in \operatorname{span}(B)$ must contain some variable in M since M is a collection of pivots. Since M gives pivots for B, we also have $|M| = |B|$.

Suppose the current query in T is $v = \sum_{i \in S} z_i$. We first clean up v by adding a suitable linear combination of linear forms from B so that v does not depend on any of the coordinates in M. This can be done because of our invariant that M gives pivots for B. Now if v is already determined because of the previous queries, we move to the appropriate child in T without making any queries in T'. If there is some variable z_i appearing in v which hasn't already been queried, query it. If it is $*$, we add i to M and move to the $*$-child in T and repeat this process. Note that we still have that M is a collection of pivots for B. If z_i is revealed to be some $b \in B$, we continue querying any other unfixed variables in v. If all variables in v have been fixed in z, then we move to the appropriate child in T.

Once we are at a leaf of T, we give the same output in T'. The bound on the $*$-depth follows by our invariant $|M| = |B|$. □

5.2 Relations and separations between measures related to stifling

In this subsection, we try to understand whether the stifling-related measures considered in the previous subsection can be separated or if they coincide in some cases.

We first give a contrived example showing that parity stifling does not imply stifling, even if we are allowed to restrict some of the bits.

Proposition 5.8. For $n = 2m + 1$, consider the function $g : \{0,1\}^n \to \{0,1\}$ defined by $g(x) = \mathrm{MAJ}(y)$ where $y \in \{0,1\}^n$ is defined by $y_i = \oplus_{j=1}^{i} x_i$. The function g is m-parity stifled, but no subfunction of g is stifled (which is equivalent to saying that $\mathsf{BCval}(g) \leq 1$).

Proof. The function g is m-parity stifled since it is essentially MAJ after a change of basis and MAJ is m-stifled. In more detail, for a set of m linearly independent forms, we consider their coordinate representation in the basis defined by y. Now find a set of pivots and fix the $m+1$ non-pivot y_i's appropriately to get a parity certificate for $g(x) = \mathrm{MAJ}(y)$.

We now argue that no restriction of g is stifled. Observe that any subfunction h of g is a threshold function, on say l bits with threshold k, of the form $\mathrm{Th}_{l,k}(b_1 \oplus z_1, b_2 \oplus z_2, \ldots, b_l \oplus z_l)$ where each $b_i \in \{0,1\}$ and $z_i = \oplus_{j=1}^{i} x_{m_j}$ for some free indices $1 \leq m_1 < m_2 < \ldots m_l \leq n$.

Without loss of generality, suppose $k \leq l/2$. We will show that there is no 0-certificate of h which leaves x_{m_1} free. Consider any fixing of the other free bits $x_{m_i}, i \geq 2$. For $1 \leq i \leq l$, define $c_i = \oplus_{j=2}^{i} x_{m_i}$. Now set $x_{m_1} = 1 \oplus \mathrm{MAJ}(b_1 \oplus c_1, b_2 \oplus c_2, \ldots, b_l \oplus c_l)$. For this x, we get $b \oplus z$ such that at least $l/2$ positions are 1. So this input is a 1-input for h. □

As discussed in the proof, even though the function is not stifled, it is stifled after performing a change of basis. Formally, there is a stifled function g and an invertible $n \times n$ matrix A over $\mathbb{F}_2$ such that $f(x) = g(Ax)$. Let us call such a function $\exists$-stifled. In other words, f is $\exists$-stifled if there is a basis $v_1, v_2, \ldots, v_n$ of $\mathbb{F}_2^n$ such that for every $i \in [n]$, $b \in \{0,1\}$, there is a domain parity certificate only setting $v_i (i \neq j)$ which fixes the function value to b.

We now note that for total functions, the notions of parity stifling and $\exists$-stifling (qualitatively) coincide. Before proving it, it will be useful to make some observations about parity stifled functions. For $b \in \{0,1\}$, say that function f is parity stifled with respect to b if for every $v \in \mathbb{F}_2^n$, there is a domain parity certificate $\langle v_1, x\rangle = b_1, \langle v_2, x\rangle = b_2, \ldots, \langle v_{n-1}, x\rangle = b_{n-1}$ fixing the value of f to b, where $v, v_1, v_2, \ldots, v_{n-1}$ form a basis of $\mathbb{F}_2^n$.

Observation 5.9. If $f : \{0,1\}^n \to \{0,1,*\}$ is a partial function which is parity stifled with respect to b, then there exist $n+1$ b-inputs $x_0, x_1, \ldots, x_n$ such that $x_0 + x_1, x_0 + x_2, \ldots, x_0 + x_n$ is a basis for $\mathbb{F}_2^n$.

Proof. Since f is parity stifled with respect to b, there is no affine subspace of codimension 1 fixing the output to $1-b$. This implies that the affine span of the b-inputs has dimension n and so there must be $n+1$ inputs satisfying the conditions in the statement of the observation. □

Observation 5.10. Suppose $x_0, x_1, \ldots, x_n$ are b-inputs of f and $x_0 + x_1, x_0 + x_2, \ldots, x_0 + x_n$ are linearly independent. Then f is stifled w.r.t b in some basis.

Proof. Let $v_i = x_0 + x_i$ for all $i \in [n]$. Define $w_i (i \in [n])$ such that they satisfy

$$\langle w_i, v_j\rangle = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{otherwise.} \end{cases}$$

In other words, w_i's are the rows of the inverse of the matrix which has v_i's as the columns.

Then f is stifled w.r.t b in the basis $w_1, w_2, \ldots, w_n$. Indeed for any $i \in [n]$, the affine subspace defined by the equations $\langle w_j, x\rangle = \langle w_j, x_0\rangle$ $(j \neq i)$ only contains the inputs x_0 and x_i which are b-inputs. □

Call a matrix $M \in \{0,1\}^{n \times m}$ good, if for every $i \in [n]$, there exist columns j_1 and j_2 which differ only on the i^{th} coordinate. Now for a function f, let the matrix M_b be the matrix with all b-inputs as columns (their order is irrelevant). Then f is stifled w.r.t b if M_b is good. More generally, f is stifled w.r.t b in some basis if there is some invertible $n \times n$ matrix A such that AM_b is good. Also, f is $\exists$-stifled if there is some invertible A such that both AM_0 and AM_1 are good.

Proposition 5.11. Let $f : \{0,1\}^n \to \{0,1\}$ be a total function. Then f is parity stifled if and only if f is $\exists$-stifled.

Proof. One direction is clear: if f is stifled in some basis, it is also parity stifled.

For the other direction, we consider two cases depending on $|f^{-1}(0)|$.

- $|f^{-1}(0)| = |f^{-1}(1)| = 2^{n-1}$

  We will show that for every basis A, if f is not stifled w.r.t. 0 in the basis A, it is not stifled w.r.t. 1 in the same basis A. Using this, we get that if AM_1 is good (such an A exists since f is parity stifled), then so is AM_0.

  Suppose f is not stifled w.r.t. 0 in the basis A. This means that for some $i \in [n]$, all columns in $AM_0|_{-i}$ (the matrix AM_0 with row i removed) are distinct. Since there are exactly 2^{n-1} columns, this means that AM_0 contains exactly one input from each pair $\{x, x^{\oplus i}\}$, where $x^{\oplus i}$ denotes x with the i^{th} bit flipped. All other inputs must appear as columns of AM_1. So AM_1 also contains exactly one input from each pair $\{x, x^{\oplus i}\}$ and thus all columns in $AM_1|_{-i}$ are distinct.

- $|f^{-1}(0)| \neq |f^{-1}(1)|$

  Without loss of generality, assume $|f^{-1}(0)| < |f^{-1}(1)|$.

  Observe that any matrix $n \times m$ matrix M with $m > 2^{n-1}$ is good since $|\{0,1\}^{n-1}| = 2^{n-1}$. This implies that since M_1 has more than 2^{n-1} columns, for every $n \times n$ matrix A, the matrix AM_1 is good.

  Now, we only need to show that f is stifled w.r.t 0 in some basis. This follows from combining Observations 5.9 and 5.10. □

We will show next that the above proposition fails for partial functions. The following lemmas will be useful.

Lemma 5.12. Let M be a good $n \times (n+1)$ matrix whose first column is 0^n. In the hypercube $\{0,1\}^n$, consider the subgraph G induced by the vertices given by the columns of M. Then G is a tree containing an edge in each direction $i \in [n]$.

Proof. We first argue that G is connected which follows from the following claim.

Claim 5.13. For any good matrix N of dimensions $n \times m$, we must have $m \geq n+1$. Moreover if $m = n+1$, then the subgraph contains a spanning tree in which edges in all n directions appear.

Proof of claim. Consider the subgraph H corresponding to N. By removing edges, we will remove all cycles in H while ensuring that for each $i \in [n]$, there is an edge in direction i. This can be done since any cycle must contain, for each $i \in [n]$, an even number of edges in direction i.

So now we have a forest F which is a subgraph of H and which contains an edge in direction i for each $i \in [n]$. Thus the number of vertices in F (and so in H) is at least $n+1$. Moreover if $m = n+1$, F is a tree. □

By the above claim, G is connected. Consider a spanning tree T of G containing edges in all directions. We need to show that $G = T$. Towards a contradiction, suppose there is an edge e in G but not in T. The graph $T \cup e$ contains a cycle. This cycle must contain, for each $i \in [n]$, an even number of edges in direction i but this is not possible since T only contains one edge in each direction. □

Lemma 5.14. The number of $n \times (n+1)$ good matrices whose first column is 0^n is exactly $(n+1)^{n-1}n!$.

Proof. By the previous lemma, every $n \times (n+1)$ good matrix corresponds to a tree with $n+1$ vertices in the hypercube (along with an ordering on the vertices). Such a tree can be described by a labelled tree T on vertices $\{0,1,2,\ldots,n\}$, along with a distinct edge label $i \in [n]$ for each edge of T. The corresponding tree in $\{0,1\}^n$ is given by mapping vertex 0 of T to 0^n and then using the edge labels of T to find the other vertices.

Thus the number of trees which send vertex 0 to 0^n is exactly the number of labelled trees on $\{0,1,\ldots,n\}$ multiplied by the number of ways of giving edge labels. By Cayley's formula, this is $(n+1)^{n-1}n!$. □

We are now in a position to prove the existence of partial functions which are parity stifled but not $\exists$-stifled.

Proposition 5.15. For $n \geq 8$, there exists a partial function $f : \{0,1\}^n \to \{0,1,*\}$ which is parity stifled but not $\exists$-stifled.

Proof. The function f is defined randomly in the following way. Let $w_1, w_2, \ldots, w_n$ be linearly independent vectors in $\mathbb{F}_2^n$ picked uniformly at random.

$$f(x) = \begin{cases} 0, & \text{if } |x| \leq 1 \\ 1, & \text{if } x \in \{1^n, 1^n + w_1, 1^n + w_2, \ldots, 1^n + w_n\} \\ *, & \text{otherwise} \end{cases}$$

For the above function to be well-defined, we need that none of $1^n + w_1, 1^n + w_2, \ldots, 1^n + w_n$ have weight at most 1. The probability that there is some i such that $|1^n + w_i| \leq 1$ is at most $\frac{n(n+1)}{2^n-1}$ by the union bound.

Any function f defined as above is parity stifled by Observation 5.10. We will show that f is $\exists$-stifled with low probability. For any fixed basis A such that f is stifled w.r.t. 0 in the basis A, we will estimate the probability that f is also stifled w.r.t. 1 in the basis A, and then use the union bound over all good bases for 0.

Let A be an invertible matrix such that f is stifled w.r.t 0 in the basis corresponding to A. This means that AM_0 is good where M_0 consists of 0^n followed by the identity matrix. But this is the same as $[0^n|A]$ being good. To estimate the probability that AM_1 is good, we first consider the shifted matrix $M_1' = [0|w_1|w_2|\ldots|w_n]$ obtained by adding 1^n to each column of M_1. Note that AM_1 is good iff AM_1' is good. By Lemma 5.14, AM_1' is good if it is one of $(n+1)^{n-1}n!$ matrices. So the probability that AM_1 is good is at most $\frac{(n+1)^{n-1}n!}{(2^n-1)(2^n-2)\ldots(2^n-2^{n-1})}$.

We will now use the union bound over all possible A. Note that since changing the order of the basis vectors does not affect whether AM_1 is good, it suffices to consider such A up to permuting the vectors. By Lemma 5.14, there are $(n+1)^{n-1}$ such good bases (disregarding order).

$$\begin{aligned} &\Pr[f \text{ is stifled in some basis}] \\ &= \Pr[\exists \text{ good basis } A \text{ such that } f \text{ is stifled w.r.t.1 in basis } A] \\ &\leq \sum_{\text{good } A} \Pr[f \text{ is stifled w.r.t.1 in basis } A] \\ &\leq (n+1)^{n-1} \cdot \frac{(n+1)^{n-1}n!}{(2^n-1)(2^n-2)\ldots(2^n-2^{n-1})}. \end{aligned}$$

Combining this with the probability that the function f is not well-defined, we get that except with probability $\frac{(n+1)n}{2^n-1} + \frac{(n+1)^{2(n-1)}n!}{(2^n-1)(2^n-2)\ldots(2^n-2^{n-1})}$, f satisfies the properties we want. This quantity is strictly less than 1 for $n \geq 8$. □

5.3 Reduction to randomized $(\oplus, *)$-decision trees

We now prove the randomized analogue of Proposition 5.5. We only need to combine the ideas used in that proof with the ideas used in the reduction to randomized $*$-depth.

Proposition 5.16. For any relation $\mathcal{R} \subseteq (\{0,1\}^m)^n \times \mathcal{O}$,

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

Proof. Let $\mathcal{T}$ be a randomized parity decision tree computing $\mathcal{R}$ to error ϵ. We will give a randomized $(\oplus, *)$-decision tree $\mathcal{T}'$ for $\mathcal{R}_{\oplus,*}$ of expected $*$-depth at most $2\,\overline{\mathrm{rank}}(\mathcal{T})$.

As before, we will run $\mathcal{T}$ on the distribution obtained by picking a uniformly random vector in each of the n subspaces of the input to $\mathcal{T}'$. This will imply correctness since for each string $z \in (\{0,1\}^m)^n$, $\mathcal{T}$ is correct with probability at least $1-\epsilon$.

Let T be a deterministic PDT in the support of $\mathcal{T}$. Our invariants will be the same as in the deterministic $(\oplus, *)$-decision tree simulation. Let us recall them. For each $i \in [n]$, there is a set P_i of linear equations which correspond to the free parities B_i in the i^{th} partial subspace of the current node in the $(\oplus, *)$-decision tree. For each $v \in B_i$, there will be some equation $v = w + b$ in P_i where $b \in \mathbb{F}_2$ and w is a parity on $(\mathbb{F}_2^m)^n$ which only depends on variables from blocks $j > i$. Each such equation in P_i will be equivalent to a constraint coming from parity queries made in T combined with the constraints in C_i $(i \in [n])$, where C_i is the collection of fixed parities of the i^{th} partial subspace.

In the beginning, each P_i is empty. Suppose we are at a node in T with the parity query $v = \langle \alpha, z \rangle + b$. We clean up the parity query v as in the deterministic case. Now, if v is fixed to some $b \in \mathbb{F}_2$, we simply move to the child corresponding to b without making any queries.

Otherwise, consider the smallest i such that some variable of z_i still appears in v. We make a query $v|_i$ in T'. If the response is $*$ (so that in the updated partial subspace, we have $v|_i \in B_i$), then in T, we move to a child uniformly at random. Suppose this child corresponds to setting the parity to $c \in \mathbb{F}_2$. Then we add the equation $v|_i = (v - v|_i) + c$ to P_i.

Now suppose the response is $v|_i + v' = b$ where v' lies in the span of B_i. Then we apply the clean-up procedure again (with the updated C_i) and continue as above.

Once we are at a leaf of T, we give the same output in T'. The expected cost of T' is at most $2\,\mathrm{rank}(T)$ since each time we see a $*$ in T', we move to the marked child in T with probability 1/2. By an argument similar to the one for Lemma 4.5, it can be shown that the above simulation reaches each node of T with the same probability as an input obtained by picking a vector in each of the n affine subspaces uniformly at random. This implies correctness as explained in the beginning of the proof. □

By combining this with the randomized composition theorem for block decision trees, Theorem B.4, we get the following.

Theorem 5.17. For all relations $\mathcal{R} \subseteq \{0,1\}^n \times \mathcal{O}$, all functions $g : \{0,1\}^m \to \{0,1,*\}$,

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

6 Classification of gadgets which allow lifting

In this section, we try to better understand which gadgets allow lifting to parity decision trees. This question was studied by Alekseev, Filmus and Smal [AFS24] in the deterministic case who gave a classification of gadgets which allow lifting from DT depth to depth and size in PDTs when composing with a total function. They show that gadgets which affine project to AND and OR allow lifting. However, this lifting theorem for PDTs actually lifts from certificate complexity and to get lifting from DT depth, they use the known relation $\mathsf{D}^{\mathsf{dt}}(f) \leq \mathsf{C}(f)^2$ because of which the exponent in the lower bound is only 1/2 instead of 1. It is unclear if these gadgets allow linear lifting, $\log \mathsf{DSize}^{\oplus\text{-}\mathsf{dt}}(f \circ g) \geq \Omega_g(\mathsf{D}^{\mathsf{dt}}(f))$ for all f, rather than just polynomial.

This situation also occurs for ordinary decision trees, though there the class of gadgets for which we do not know if linear lifting is possible is relatively small. By the results of [AFS24, DM23], we know that for any g satisfying $\mathsf{DRank}^{\mathsf{dt}}(g) \geq 2$, we have $\mathsf{DRank}^{\mathsf{dt}}(\mathcal{R} \circ g) \geq \mathsf{D}^{\mathsf{dt}}(\mathcal{R})$. The condition $\mathsf{DRank}^{\mathsf{dt}}(g) \geq 2$ is equivalent to there being some subfunction h of g with $\mathsf{C}_{\mathsf{min}}(h) \geq 2$. In this sense, minimum certificate complexity gives an explanation for linear lifting for all gadgets where this is known for ordinary decision trees.

For parity decision trees, it is perhaps less immediate if the analogous quantity minimum parity certificate complexity being at least 2 is sufficient for lifting. (Since our focus in this section is only on understanding when linear lifting is possible, we will mostly drop the word linear from here on.) However, we note that the observations in Section 5.2, actually show that being parity stifled is equivalent to not having a parity certificate of codimension 1. Indeed, Observation 5.9 uses the fact that a parity stifled function cannot have a parity certificate of size 1 and this is the only implication of being parity stifled which is used there. By combining with Observation 5.10, we get the following claim.

Claim 6.1. Let $g : \{0,1\}^m \to \{0,1,*\}$ be a partial function. The function g is parity stifled if and only if $\mathsf{C}^{\oplus}_{\mathsf{min}}(g) \geq 2$.

In particular, $\mathsf{C}^{\oplus}_{\mathsf{min}}(g) \geq 2$ is indeed a sufficient condition for lifting to PDT rank, since parity stifling is sufficient for lifting (Proposition 5.2). Moreover, using the above claim and Proposition 5.11, for any total function $f : \{0,1\}^m \to \{0,1\}$, we get that f satisfies $\mathsf{C}^{\oplus}_{\mathsf{min}}(f) \geq 2$ if and only if f is stifled in some basis.

Similar to the case of ordinary decision trees, we can further relax the condition $\mathsf{C}^{\oplus}_{\mathsf{min}}(g) \geq 2$ to $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(g) \geq 2$. If $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(g) \geq 2$, then there is some affine subspace of $\mathbb{F}_2^m$ on which the restricted function h satisfies $\mathsf{C}^{\oplus}_{\mathsf{min}}(h) \geq 2$ and g must therefore allow lifting to PDT rank.

So the only remaining case is when $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(g) = 1$, i.e. when g can be computed by a decision list of parities. We show that for any *total* g satisfying $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(g) = 1$ which is not a parity, there is some function $g' : \{0,1\}^{m'} \to \{0,1\}$ such that for every relation $\mathcal{R}$, $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(\mathcal{R} \circ g) = \mathsf{DRank}^{\mathsf{dt}}(\mathcal{R} \circ g')$. Therefore, to understand which gadgets allow lifting to deterministic PDT rank, it is enough to understand which gadgets allow lifting to DT rank. We start by showing this statement for the OddMaxBit gadget and then show how every gadget is essentially equivalent to OddMaxBit for some input length.

To simplify notation and avoid confusion about the definition, we work with a variant of OddMaxBit obtained by reversing the order of the bits. The function EvenMinBit $\mathrm{EMB}_m : \{0,1\}^m \to \{0,1\}$ is defined in the following way. For $x \in \{0,1\}^m$, $\mathrm{EMB}_m(x) = 1$ if the index of the first 1 in x is even. More formally, if $i \in [n+1]$ is the unique number such that $x_i = 1$ and $x_j = 0$ for all $1 \leq j < i$, then $\mathrm{EMB}(x)$ is 1 if and only if i is even. By convention, we always have $x_{n+1} = 1$, so that $\mathrm{EMB}_m(0^m)$ is 1 if $m+1$ is even.

Lemma 6.2. For all relations $\mathcal{R} \subseteq \{0,1\}^n \times \mathcal{O}$ and $m \geq 2$, the following hold:

- $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(\mathcal{R} \circ \mathrm{EMB}_m) = \mathsf{DRank}^{\mathsf{dt}}(\mathcal{R} \circ \mathrm{EMB}_m)$.
- There exists a marked decision tree T computing $\mathcal{R} \circ \mathrm{EMB}_m$ of rank $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(\mathcal{R} \circ \mathrm{EMB}_m)$ with the following properties. Within each block $i \in [n]$, for all $j \in [m]$, if $x_{i,j}$ is queried, then $x_{i,j'}, j' < j$ must have been queried earlier. The marked edge at a query $x_{i,j}$ is $x_{i,j} = 1$ if $j < m$ and is $x_{i,j} = 0$ if $j = m$.

Proof. For the first point, we will prove the following chain of inequalities

$$\mathsf{DRank}^{\mathsf{dt}}(\mathcal{R} \circ \mathrm{EMB}_m) \geq \mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(\mathcal{R} \circ \mathrm{EMB}_m) \geq \mathsf{D}^{*\text{-}\mathsf{dt}}(\mathcal{R} \circ (\mathrm{EMB}_m)_*) \geq \mathsf{DRank}^{\mathsf{dt}}(\mathcal{R} \circ \mathrm{EMB}_m).$$

The inequality $\mathsf{DRank}^{\mathsf{dt}}(\mathcal{R} \circ \mathrm{EMB}_m) \geq \mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(\mathcal{R} \circ \mathrm{EMB}_m)$ is obvious. The inequality $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(\mathcal{R} \circ \mathrm{EMB}_m) \geq \mathsf{D}^{*\text{-}\mathsf{dt}}(\mathcal{R} \circ (\mathrm{EMB}_m)_*)$ follows from Lemma 4.1.

So we only need to show $\mathsf{D}^{*\text{-}\mathsf{dt}}(\mathcal{R} \circ (\mathrm{EMB}_m)_*) \geq \mathsf{DRank}^{\mathsf{dt}}(\mathcal{R} \circ \mathrm{EMB}_m)$. For brevity, let $h_m = (\mathrm{EMB}_m)_*$. Suppose we have a decision tree T solving $\mathcal{R} \circ h_m$. We will give a decision tree for $\mathcal{R} \circ \mathrm{EMB}_m$ whose rank is at most the $*$-depth of T. We will use z_{ij} to denote the inputs of $\mathcal{R} \circ h_m$ (queried in T) and x_{ij} to denote the inputs of $\mathcal{R} \circ \mathrm{EMB}_m$.

With each possible input $x \in \{0,1\}^m$ to a block (except $0^{m-1}1$), we will associate two certificates z_1 and z_2 of EMB_m such that $h_m(z_1) = h_m(z_2) = \mathrm{EMB}_m(x)$. The certificates associated with x will depend only on the index of the first 1 in x. For now, consider only the case where the first index j is at most $m-1$. (The case of the input being 0^m will be handled in the same way as any input where $j = m-1$.) Additionally, we will ensure that the certificates z_1, z_2 have 0s in all positions before j and for each $k \geq j$ at least one of z_1, z_2 will have $*$ at index j. The latter condition will be used to go down a $*$-edge while remaining consistent with some certificate which gives the same input for that block.

We now define the associated certificates. For $j \in [m-1]$, let $C_{j,1} = 0^{j-1}1*^{m-j}$ and $C_{j,2} = 0^{j-1}*01^{m-1-j}$. These are the certificates associated with any x whose first 1 is at index $j \in [m-1]$. The string 0^m (in which the first 1 occurs at index $m+1$) is associated with the certificates $C_{m-1,1}$ and $C_{m-1,2}$. The indices $m-1$ and $m+1$ are always treated in the same way. The properties described above can be easily verified for these certificates. The string $0^{m-1}1$ is associated with the certificate $0^{m-1}1$.

During the simulation, we will maintain the following invariants. For any block $i \in [n]$, we will always make queries in order, i.e. if $x_{i,j}$ is queried at some point, then $x_{i,j'}$ for $j' < j$ must have been queried at some point earlier. After seeing a 1 in the block x_i, we fix z_i completely to one of the associated certificates. For $j \leq m-2$, if the first j bits of x_i are 0, then the first j bits of z_i are also 0 (note that this property holds for the associated certificates).

We now describe the simulation. Suppose we are at a node querying $z_{i,j}$ in the tree T. If the block z_i has already been fixed, we move to the appropriate child in T without making any queries in T'. Otherwise suppose $x_{i,j'}$ is the last queried bit in x_i (if x_i has not been queried at all, then $j' = 0$). Starting from $x_{i,j'+1}$, query bits of x_i one by one until we see a 1 or we have queried $x_{i,j-1}$ which is 0. Here we mark the outgoing edge where a 1 is seen for all x_{i,j_1} $(j'+1 \leq j_1 < j)$. Note that we only see at most one marked edge when making these queries.

Suppose we see a 1 when making these queries. Let j'' be the index of the 1. Then we set $z_i = C_{j'',1}$ which ensures that $z_{ij} = *$. So we may move to the $*$-child in T and continue the simulation from that node. Note that we crossed one marked edge in T' when simulating the query in T in this case.

Now suppose $x_{i,j_1} = 0$ for all $j_1 < j$. There are there are two cases depending on whether $j \leq m-2$.

- If $j \leq m-2$, query $x_{i,j}$ in T' where the edge $x_{i,j} = 1$ is marked. If $x_{i,j} = 1$, set $z_i = C_{j,2}$ so that $z_{i,j} = *$. We now move to the $*$-child in T. If $x_{i,j} = 0$, then set $z_{i,j} = 0$ and move accordingly in T. Note that in each case, the number of marked edges crossed in T' is equal to the number of $*$'s seen in T.

- If $j \geq m-1$, we query the remaining bits of x_i until we either see a 1 or both $x_{i,m-1} = x_{i,m} = 0$. The marked edges here are $x_{i,m-1} = 1$ (this query has not been made previously only if $j = m-1$) and $x_{i,m} = 0$. Since the first $m-2$ bits of x_i were fixed to 0 earlier, focus on the last two bits of x_i. If they are 00 or $1\cdot$, set z_i to one of $C_{m-1,1}$ or $C_{m-1,2}$ to ensure that $z_{i,j} = *$. Note that in this case, we have crossed exactly one marked edge in T' and moved to the $*$-child in T. If the last bits are 01, set $z_i = 0^{m-1}1$. In this case, we do not incur any cost in either tree.

Once we reach a leaf of T, we give the same output in T'.

We now need to verify that T' is correct. Let x be an input to $\mathcal{R} \circ \mathrm{EMB}_m$. Consider the partially fixed z created when performing the simulation on input x. Any input which is consistent with this partially fixed z reaches the same leaf in T. So we need to just find some $z' \in (\{0,1,*\}^m)^n$ which is consistent with the partially fixed z and $\mathrm{EMB}_m^n(x) = h_m^n(z)$. Any block z_i of z which is completely fixed satisfies $h_m(z_i) = g(x_i)$ by the properties of the defined certificates. For any block z_i which is not fixed, note that the only fixed positions are among the first $m-2$ which are set to 0. So the remaining can be set according to one of the certificates associated with x_i to get z_i'. This gives z_i' as desired.

For the cost of T', we have argued above that we only go to the $*$-child in T iff we cross a marked edge in T' in each of the possible cases that could arise during the simulation. This shows that the rank of T' is at most the $*$-depth of T.

For the second point, note that the tree T' obtained in the simulation above has the desired properties. □

We will say that two functions g and h are equivalent if for all relations $\mathcal{R}$, we have $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(\mathcal{R} \circ g) = \mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(\mathcal{R} \circ h)$.

Lemma 6.3. For every total function $g : \{0,1\}^m \to \{0,1\}$ which is not a parity and satisfies $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(g) = 1$, there exists some function $h \in \{\mathrm{EMB}_k, \neg\mathrm{EMB}_k \mid k \geq 2\}$ such that for all relations $\mathcal{R}$,

$$\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(\mathcal{R} \circ g) = \mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(\mathcal{R} \circ h).$$

Proof. Let T be a PDT of rank 1 computing g. We will assume that the last two leaves (the two leaves with the same parent) have different labels. Since T is a decision list of parities, we may list the parities in T in the order in which they are queried, $v_1, v_2, \ldots, v_d$. We may assume that all these linear forms are linearly independent. If this is not the case, we can simplify the tree by removing a linear form v_i which can be written as a linear combination of the linear forms queried earlier $v_j, j < i$, since the value of the query v_i is already determined by the previous queries. Extend this set of linear forms to get a basis $v_1, v_2, \ldots, v_m$ of $\mathbb{F}_2^m$. Let A be the invertible matrix in $\mathbb{F}_2^{m \times m}$ whose rows are $v_1, v_2, \ldots, v_m$. Define $g' : \{0,1\}^m \to \{0,1\}$ by $g'(x) = g(A^{-1}x)$. Since PDT rank does not change under a change of basis, we have for all relations $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(\mathcal{R} \circ g) = \mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(\mathcal{R} \circ g')$. Also note that $d \geq 2$ since g is not a parity.

Now g' can be computed by the decision list T' obtained by replacing each linear form v_i in T by the bit query x_i for $i \in [d]$, while leaving the leaves unchanged. Since g' is total, we may identify T' and g' from now on.

By negating the inputs to g', we can assume that in T', at each query node x_i ($i < d$), the leaf corresponds to $x_i = 1$. Additionally, by negating x_d if needed, we can assume that the label of the leaf at $x_{d-1} = 1$ is not equal to the leaf label at $x_d = 1$. Note that negating the input bits does not change $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(\mathcal{R} \circ g')$. Finally, we will assume that the first leaf label (at $x_1 = 1$) is 0. Under these assumptions, we will show that g' is equivalent to EMB_k for some k.

To make g' satisfy this assumption about the first leaf label, we may need to negate g. In general, it can be the case that $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(\mathcal{R} \circ g) \neq \mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(\mathcal{R} \circ \neg g)$. However, in the case that the first leaf label is 1 instead of 0, we get that $\neg g$ is equivalent to EMB_k for some k which implies that g is equivalent to $\neg\mathrm{EMB}_k$.

We will now show that g' is equivalent to EMB_k for some k under the above assumptions on T'. Under these assumptions, we can succinctly represent T' by an ordered partition of d, $[s_1, s_2, \ldots, s_{k-1}, s_k]$, where $\sum_{i\in[k]} s_i = d$ and $s_k = 1$. We interpret such an ordered partition in the following way. The number s_1 indicates that the first s_1 leaves are all labeled 0, the next s_2 leaves are all labeled 1, and so on. Since $s_k = 1$ and $d \geq 2$, we have $k \geq 2$. Let $t_0 = 0$ and for $i \in [k]$, $t_i = \sum_{j=1}^{i} s_j$. Observe that such a function can be expressed as $g'(x) = \mathrm{EMB}_k(z_1, z_2, \ldots, z_k)$ where for each $i \in [k]$, $z_i = \bigvee_{j=t_{i-1}+1}^{t_i} x_j$. Note that z_k is simply x_d. This implies that g' contains EMB_k as a subfunction by restricting some of the inputs to 0 so that each z_i becomes just one input. So for every relation $\mathcal{R}$, $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(\mathcal{R} \circ g') \geq \mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(\mathcal{R} \circ \mathrm{EMB}_k)$. For the other direction, we will give a simulation argument.

By Lemma 6.2, there is a decision tree T for $\mathcal{R} \circ \mathrm{EMB}_k$ whose rank is $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(\mathcal{R} \circ \mathrm{EMB}_k)$ and for which the marked edge at a query $y_{i,j}$ is $y_{i,j} = 1$ when $j < k$. For each query $y_{i,j}$, we will compute $z_{i,j}$ (as defined above), the corresponding OR of inputs in the block x_i. The OR function has a simple marked decision tree where at each node, we mark the edge where the query evaluates to 1. Note that for this tree, we see exactly one marked edge if the output is 1 and no marked edges if the output is 0. So we can simply use this marked tree (for suitable input lengths) to compute $z_{i,j}$ for $j < k$ since the query $y_{i,j}$ has cost 1 when $y_{i,j} = 1$ and cost 0 otherwise. At a node $y_{i,k}$ in T, we only need to query $x_{i,d}$ and mark the edge $x_{i,d} = 0$ since $y_{i,k} = 0$ is marked in T. Clearly this simulation correctly computes $\mathcal{R} \circ g'$ since we have essentially substituted each $y_{i,j}$ in T by the appropriate OR of variables in x_i. The cost is also preserved by construction. □

We can now combine the above lemmas and the discussion following Claim 6.1 to get the following.

Proposition 6.4. 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 any relation $\mathcal{R}$, $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(\mathcal{R} \circ g) \geq \mathsf{D}^{\mathsf{dt}}(\mathcal{R})$.
- $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(g) = 1$ and there exists $h \in \{\mathrm{EMB}_k, \neg\mathrm{EMB}_k \mid k \geq 2\}$ such that for all relations $\mathcal{R}$, $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(\mathcal{R} \circ g) = \mathsf{DRank}^{\mathsf{dt}}(\mathcal{R} \circ h)$.

Remark 6.5. The simulation in Lemma 6.2 can be modified to show that $\mathsf{D}^{*\text{-}\mathsf{dt}}(\mathcal{R} \circ (\mathrm{EMB}_m)_*) \geq \mathsf{D}^{1\text{-}\mathsf{dt}}(\mathcal{R} \circ \mathrm{EMB}_{m-1})$. To see this, it will be useful to note that

$$\mathrm{EMB}_m(x_1, x_2, \ldots, x_m) = \mathrm{EMB}_{m-1}(x_1, x_2, \ldots, x_{m-2}, x_{m-1} \vee \neg x_m).$$

The only change in the simulation occurs when $j \geq m-1$ in which case the corresponding query in x_i is just $x_{i,m-1}$. If $x_{i,m-1} = 1$, we set $z_{i,j} = *$ and ensure that $z_{i,m-1} \vee \neg z_{i,m} = 1$ by setting the other input in z_i appropriately.

It is also easy to show that $\mathsf{D}^{\text{1-dt}}(\mathcal{R} \circ \mathrm{EMB}_{m-1}) \geq \mathsf{DRank}^{\mathsf{dt}}(\mathcal{R} \circ \mathrm{EMB}_m)$ by using the above relation between EMB_m and EMB_{m-1}. Given a decision tree for $\mathcal{R} \circ \mathrm{EMB}_{m-1}$ where the 1-edges are marked, we only need to replace a query to $x_{i,m-1}$ by a rank 1 decision tree computing $x_{m-1} \vee \neg x_m$. Together these show that we also have $\mathsf{DRank}^{\mathsf{dt}}(\mathcal{R} \circ \mathrm{EMB}_m) = \mathsf{D}^{\text{1-dt}}(\mathcal{R} \circ \mathrm{EMB}_{m-1})$.

A corollary of this observation is that DT rank for the composition with OR_2 corresponds exactly to 1-depth : for any relation $\mathcal{R}$, we have $\mathsf{DRank}^{\mathsf{dt}}(\mathcal{R} \circ \mathrm{OR}_2) = \mathsf{D}^{\text{1-dt}}(\mathcal{R})$. The fact that OR_2 lifts 1-depth to DT rank was implicitly used in [BIW04] to give a separation between tree-like Resolution size and Resolution size. Since Lemma 6.2 implies $\mathsf{DRank}^{\oplus\text{-dt}}(\mathcal{R} \circ \mathrm{OR}_2) = \mathsf{DRank}^{\mathsf{dt}}(\mathcal{R} \circ \mathrm{OR}_2)$, we also get that OR_2 lifts 1-depth to PDT rank. We discuss implications of this for tree-like Res($\oplus$) in Appendix A.3.

So far our discussion in this section has only been about deterministic PDTs, but the proofs above can be modified using ideas from earlier sections to give analogous statements for randomized PDTs. Since the ideas stay mostly they some, we only sketch the proofs of these, focusing on the differences.

Proposition 6.6. Suppose $g : \{0,1\}^m \to \{0,1,*\}$ satisfies $\mathsf{C}^{\oplus}_{\mathsf{min}}(g) \geq 2$. Then for all relations $\mathcal{R} \subseteq \{0,1\}^n \times \mathcal{O}$,

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

Proof sketch. By Observations 5.9 and 5.10, there exist bases B_0 and B_1 such that g is parity stifled w.r.t b in basis B_b for $b \in \{0,1\}$. As in Observation 3.8, for $b \in \{0,1\}$, we create a distribution μ_b by picking a random $i \in [m]$, setting the parity $v_i \in B_b$ uniformly and fixing the other parities to give a domain parity b-certificate of g.

The simulation is essentially the same as the one in the proof of Proposition 3.9. The main change is that when simulating a parity w involving some $z_{i,j}$, after querying x_i which is revealed to be, say, b, we express $w|_i$ in the basis B_b to get, say, $w|_i = \sum_{j \in S} v_j$ for some $S \subseteq [m]$ where v_j's are parities from B . Now sample block z_i according to μ_b in the following way. Pick a random $i \in [m]$ and fix all parities in B_b other than v_i according to the domain certificate. If $i \in S$, then we move to a random child. Otherwise we also sample a random bit $c \in \mathbb{F}_2$ which we assign to v_i.

After doing this, we simplify the PDT as before. Since we have now fixed parities instead of individual bits, at each parity query in the PDT, we first express the projection onto block i as a linear combination of the parities in B_b and then substitute accordingly. (Alternatively we could have kept track of the fixed parities and substituted them when required as in [CMSS23] and other proofs in earlier sections.)

The analysis of the expected cost stays the same since we still move to each child with probability at least $1/(2m)$ when making a query. □

Lemma 6.7. For $m \geq 2$, the following hold:

- For all relations $\mathcal{R} \subseteq \{0,1\}^n \times \mathcal{O}$, real $\beta > 0$, $\mathsf{RRank}^{\oplus\text{-dt}}_{\epsilon}(\mathcal{R} \circ \mathrm{EMB}_m) = \Omega(\beta \mathsf{RRank}^{\mathsf{dt}}_{\epsilon+\beta}(\mathcal{R} \circ \mathrm{EMB}_m))$.
- For all functions, $f : \{0,1\}^n \to \mathcal{O}$, there exists a marked randomized decision tree $\mathcal{T}$ computing $f \circ g$ to constant error of rank $O(\mathsf{RRank}^{\oplus\text{-dt}}(f \circ g))$ with the following properties. Within each block $i \in [n]$, for all $j \in [m]$, if $x_{i,j}$ is queried, then $x_{i,j'}, j' < j$ must have been queried earlier. The marked edge at a query $x_{i,j}$ is $x_{i,j} = 1$ if $j < m$ and is $x_{i,j} = 0$ if $j = m$.

Proof sketch. As usual, we will prove relations between the expected cost measures and the worst-case analogues will follow from standard arguments.

$$\overline{\mathsf{RRank}^{\mathsf{dt}}_{\epsilon}}(\mathcal{R}\circ \mathrm{EMB}_m) \geq \overline{\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}_{\epsilon}}(\mathcal{R}\circ \mathrm{EMB}_m) \geq \frac{1}{2}\overline{\mathsf{R}^{*\text{-}\mathsf{dt}}_{\epsilon}}(\mathcal{R}\circ (\mathrm{EMB}_m)_*),$$
$$\overline{\mathsf{R}^{*\text{-}\mathsf{dt}}_{\epsilon}}(\mathcal{R}\circ (\mathrm{EMB}_m)_*) \geq \frac{1}{2}\overline{\mathsf{R}^{1\text{-}\mathsf{dt}}_{\epsilon}}(\mathcal{R}\circ \mathrm{EMB}_{m-1}) \geq \frac{1}{2}\overline{\mathsf{RRank}^{\mathsf{dt}}_{\epsilon}}'(\mathcal{R}\circ \mathrm{EMB}_m)$$

In the last inequality, $\overline{\mathsf{RRank}^{\mathsf{dt}}_{\epsilon}}'(\mathcal{R})$ is $\min_{\mathcal{T}}\max_x \operatorname{cost}(\mathcal{T},x) = \min_{\mathcal{T}}\max_x \mathbb{E}_{T\sim\mathcal{T}}[\operatorname{cost}(T,x)]$, where $\mathcal{T}$ varies over marked randomized decision trees computing $\mathcal{R}$ to error ϵ, x varies over the domain of $\mathcal{R}$ and $\operatorname{cost}(T,x)$ is the number of marked edges seen when running T on input x. This could be smaller than the usual definition of expected rank. By truncating the tree after $\overline{\mathsf{RRank}^{\mathsf{dt}}_{\epsilon}}'(\mathcal{R})/\beta$ marked edges have been seen, we get a decision tree whose worst case rank is bigger by a factor $1/\beta$ and the error incurred increases by an additive β by using Markov's inequality.

If we are composing with a function f and ϵ,β are constants, then by repeating a constant number of times we can bring the error back down to any constant. This will be used to go from a decision tree with constant error probability whose expected cost on each input is at most $O(\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}(f\circ g))$ to a randomized decision tree whose worst case rank is $O(\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}(f\circ g))$.

We now prove the inequalities stated above. The first inequality is obvious and the second follows from Lemma 4.4.

The first inequality in the next line follows by a simulation similar to the one in Lemma 6.2 modified in Remark 6.5. The main change now is that with each input to EMB_{m-1}, we associate some distribution on the certificates defined in the proof of Lemma 6.2. In all cases, except when the input is 0^{m-1}, the associated distribution will be uniform over the two associated certificates. The simulation stays the same except that when we see a 1 in x_i, instead of picking z_i to be an associated certificate greedily, we sample it according to the distribution. Since the distribution is uniform, with probability $1/2$, we still go down the $*$-edge in the tree T. So we cross a marked edge in T after seeing two 1's in x in expectation. This gives the desired bound. This will also imply the second statement in the lemma by combining with the last inequality.

For the last inequality, we do the same thing as described in Remark 6.5, replacing a query $x_{i,m-1}$ by a marked decision tree computing $x_{i,m-1}\vee\neg x_{i,m}$. □

Remark 6.8. Similar to the deterministic case, the above simulation shows that $\overline{\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}_{\epsilon}}(\mathcal{R}\circ\vee_2)\geq\Omega(\overline{\mathsf{R}^{1\text{-}\mathsf{dt}}_{\epsilon}}(\mathcal{R}))$. Let $f:\{0,1\}^n\to\{0,1\}$ be a total function. Fractional monotone block sensitivity of f is a lower bound on $\overline{\mathsf{R}^{0\text{-}\mathsf{dt}}}(f)$ (see, for instance, [CDMRS23]) and therefore also a lower bound on $\overline{\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}}(f\circ\wedge_2)$.

Now consider the 3-bit gadget $g(x,y,z)=x\vee(y\wedge z)$ which embeds both $\vee$ and $\wedge$. By considering the above fractional monotone block sensitivity lower bound for all 2^n possible shifts of f, we get that $\overline{\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}}(f\circ g)\geq\Omega(\mathsf{fbs}(f))$ where $\mathsf{fbs}(f)$ denotes the fractional block sensitivity of f. Using the relation $\mathsf{R}^{\mathsf{dt}}(f)\leq O(\mathsf{fbs}(f)^3)$ for total functions, we get $\overline{\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}}(f\circ g)\geq\Omega(\mathsf{R}^{\mathsf{dt}}(f)^{1/3})$. This shows that in the randomized setting also all gadgets other than indicators of subcubes allow polynomial lifting to ordinary decision tree rank.

Lemma 6.9. For every total function $g:\{0,1\}^m\to\{0,1\}$ which is not a parity and satisfies $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(g)=1$, there exists some function $h\in\{\mathrm{EMB}_k,\neg\mathrm{EMB}_k \mid k\geq 2\}$ such that for all functions $f:\{0,1\}^n\to\mathcal{O}$,

$$\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}(f\circ g)=\Theta(\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}(f\circ h)).$$

Proof sketch. Note that our assumption is still that g can be computed by a deterministic parity decision list. This lemma essentially follows from the same arguments used in proving Lemma 6.3. For an $h \in \{\mathrm{EMB}_k, \neg\mathrm{EMB}_k \mid k \geq 2\}$ which is equivalent to g with respect to deterministic PDTs as in Lemma 6.3, it is clear that $\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}(f \circ g) = \Omega(\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}(f \circ h))$ since h is a subfunction of g after a change of basis.

For the other direction, we use the second point of Lemma 6.7 and apply the modifications described in the proof of Lemma 6.3 to each deterministic tree in the support of a randomized decision tree. This works since all our modifications were based on operations which do not incur any error, like applying an invertible linear transformation, negating inputs and computing OR with a rank 1 deterministic decision tree . □

Combining Proposition 6.6 and Lemmas 6.7 and 6.9 gives the following.

Proposition 6.10. 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 any relation $\mathcal{R}$, $\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}(\mathcal{R} \circ g) \geq \Omega_m(\mathsf{R}^{\mathsf{dt}}_{4/9}(\mathcal{R}))$.
- $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(g) = 1$ and there exists $h \in \{\mathrm{EMB}_k, \neg\mathrm{EMB}_k \mid k \geq 2\}$ such that
  - for all relations $\mathcal{R}$, $\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}(\mathcal{R} \circ g) = \Omega(\mathsf{RRank}^{\mathsf{dt}}_{4/9}(\mathcal{R} \circ h))$.
  - for all functions f, $\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}(f \circ g) = \Theta(\mathsf{RRank}^{\mathsf{dt}}(f \circ h))$.

Remark 6.11. In the proof of Lemma 6.7, we saw that $\overline{\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}}(\mathcal{R} \circ \mathrm{EMB}_2) \geq \Omega(\overline{\mathsf{R}^{1\text{-}\mathsf{dt}}}(\mathcal{R} \circ \mathrm{EMB}_1))$. Since $\mathrm{EMB}_2(x,y) = \neg x \wedge y$ and $\mathrm{EMB}_1(x) = \neg x$, this is equivalent to $\overline{\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}}(\mathcal{R} \circ \mathrm{AND}_2) \geq \Omega(\overline{\mathsf{R}^{0\text{-}\mathsf{dt}}}(\mathcal{R}))$. Using this we can give another proof of $\overline{\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}}(\mathcal{R} \circ \mathrm{UDISJ}_{2m}) \geq \Omega(\overline{\mathsf{R}^{\mathsf{dt}}}(\mathcal{R})m)$ for all relations $\mathcal{R}$ and $m \geq 2$.

We view UDISJ_{2m} as the composed partial function $\mathrm{PrOR}_m \circ \mathrm{AND}_2$ where the promise OR function PrOR computes the OR function under the promise that the input has Hamming weight at most 1. It is known that $\overline{\mathsf{R}^{0\text{-}\mathsf{dt}}}(\mathcal{R} \circ \mathrm{PrOR}_m) \geq \Omega(\overline{\mathsf{R}^{\mathsf{dt}}}(\mathcal{R})m)$ [ABK16, Theorem 5]. Theorem 5 in [ABK16] deals with ordinary depth instead of 0-depth and is stated only for the total function OR, but the proof there can be readily modified to work for 0-depth and only relies on inputs to OR of Hamming weight at most 1. We now combine this with the observation above that AND_2 lifts 0-depth to PDT rank to get

$$\overline{\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}}(\mathcal{R} \circ \mathrm{UDISJ}_{2m}) = \overline{\mathsf{RRank}^{\oplus\text{-}\mathsf{dt}}}(\mathcal{R} \circ \mathrm{PrOR}_m \circ \mathrm{AND}_2) \geq \Omega(\overline{\mathsf{R}^{0\text{-}\mathsf{dt}}}(\mathcal{R} \circ \mathrm{PrOR}_m)) \geq \Omega(\overline{\mathsf{R}^{\mathsf{dt}}}(\mathcal{R})m).$$

We now consider the question of when EMB_k allows lifting to PDTs since this is essentially the only remaining case. For $k \leq 2$, Alekseev, Filmus and Smal [AFS24] observed that EMB_k does not lift DT depth to DT size since for these k, EMB_k is just a conjunction of literals and thus the AND_n function shows that EMB_k does not allow depth to size lifting. As discussed in the beginning of this section, for $k \geq 3$, they showed that EMB_k lifts certificate complexity to DT size. They conjectured that EMB_k should also lift DT depth to DT size (perhaps with an exponent less than 1) even for relations when $k \geq 3$.

While we are not able to prove or disprove this, we observe that the following stronger version of the conjecture is false. Specifically, we may ask if these gadgets allow lifting from DT depth to DT rank. This may seem plausible considering that all proofs discussed above also work for rank. Since $\log \mathsf{DSize}^{\mathsf{dt}}(\mathcal{R}) \geq \mathsf{DRank}^{\mathsf{dt}}(\mathcal{R})$, this would have implied the above conjecture.

We observe that for all $k \geq 3$, EMB_k fails to lift DT depth to DT rank for some partial function. By Lemma 6.3, this shows that all gadgets g with $\mathsf{DRank}^{\mathsf{dt}}(g) = 1$ fail to lift DT depth to DT rank. The partial function $f : \{0,1\}^n \to \{0,1,*\}$ we consider is a promise version of the EvenMinBit function where there is a unique occurrence of the pattern 01.

$$f(x) = \begin{cases} 1, & \text{if } x = 0^{i-1}1^{n-i+1} \text{ and } i \equiv 0 \pmod 2 \\ 0, & \text{if } x = 0^{i-1}1^{n-i+1} \text{ and } i \equiv 1 \pmod 2 \\ * & \text{otherwise.} \end{cases}$$

Proposition 6.12. Let $k \geq 3$. For the function $f : \{0,1\}^n \to \{0,1,*\}$ defined above, we have $\mathsf{D}^{\mathsf{dt}}(f) = \lceil \log(n+1) \rceil$ (and even $\mathsf{R}^{\mathsf{dt}}(f) = \Omega(\log n)$) but $\mathsf{DRank}^{\mathsf{dt}}(f \circ \mathrm{EMB}_k) \leq k-1$.

This does not give a counterexample to the original conjecture since $\mathsf{DSize}^{\mathsf{dt}}(f \circ \mathrm{EMB}_k) \geq \mathsf{DSize}^{\mathsf{dt}}(f) \geq n+1$. But a stronger separation between $\mathsf{D}^{\mathsf{dt}}(f)$ and $\mathsf{DRank}^{\mathsf{dt}}(f \circ \mathrm{EMB}_k)$, say of the form $\mathsf{DRank}^{\mathsf{dt}}(f \circ \mathrm{EMB}_k) \leq O_k(1)$ and $\mathsf{D}^{\mathsf{dt}}(f) \geq \omega(\log n)$, would be enough to refute it, since $\log \mathsf{DSize}^{\mathsf{dt}}(f \circ \mathrm{EMB}_k) \leq \mathsf{DRank}^{\mathsf{dt}}(f \circ \mathrm{EMB}_k) O_k(\log n)$. We also do not know of a total function f which provides a separation between $\mathsf{D}^{\mathsf{dt}}(f)$ and $\mathsf{DRank}^{\mathsf{dt}}(f \circ \mathrm{EMB}_k)$, but such a separation can at best be quadratic because of the relation between $\mathsf{C}(f)$ and $\mathsf{D}^{\mathsf{dt}}(f)$.

Proof of Proposition 6.12. It is well known that $\mathsf{D}^{\mathsf{dt}}(f) = \lceil \log(n+1) \rceil$ and $\mathsf{R}^{\mathsf{dt}}(f) = \Omega(\log n)$. We prove the lower bounds for completeness. Suppose T is a deterministic decision tree computing f. For any $i \in [n]$, x_i must be queried somewhere in the tree, since otherwise T cannot distinguish between $0^i 1^{n-i}$ and $0^{i-1}1^{n-i+1}$. This implies that $\mathsf{DSize}^{\mathsf{dt}}(f) \geq n+1$, using which we get $\mathsf{D}^{\mathsf{dt}}(f) \geq \log \mathsf{DSize}^{\mathsf{dt}}(f) \geq \log(n+1)$. A similar argument works in the randomized case. In any randomized tree $\mathcal{T}$ for f, each x_i must be queried with probability at least $1/3$ if $\mathcal{T}$ computes f to error $1/3$. So the expected size of $\mathcal{T}$ is at least $\Omega(n)$ which implies $\mathsf{R}^{\mathsf{dt}}(f) = \Omega(\log n)$ as desired.

We now prove the upper bound $\mathsf{DRank}^{\mathsf{dt}}(f \circ \mathrm{EMB}_k) \leq k-1$. By Remark 6.5, it suffices to show that $\mathsf{D}^{\text{1-}\mathsf{dt}}(f \circ \mathrm{EMB}_l) \leq l$ for $l \geq 2$. Our upper bound will hold for the more general induction principle $\mathcal{R} \subseteq \{0,1\}^n \times [n+1]$ defined as follows. For any $x \in \{0,1\}^n$, $i \in [n+1]$, we have $(x,i) \in \mathcal{R}$ if and only if $x_{i-1} = 0$ and $x_i = 1$, where by convention, we always have $x_0 = 0$ and $x_{n+1} = 1$. Since f is the same as $\mathcal{R}$ restricted to its critical inputs, it suffices to show that $\mathsf{D}^{\text{1-}\mathsf{dt}}(\mathcal{R} \circ \mathrm{EMB}_l) \leq l$.

Our goal is to find an $i \in [n+1]$ such that $\mathrm{EMB}_l(x_{i-1}) = 0$ and $\mathrm{EMB}_l(x_i) = 1$. We will use s and t to indicate the ends of the current search space. We will always have $g(x_{s-1}) = 0$ and $g(x_t) = 1$. In the beginning, $s = 1$ and $t = n+1$. Starting from $j = t-1$ down to $j = s$, query $x_{j,1}$ one by one until we see a 1 or all $x_{j,1}$ were revealed to be 0. In the former case, suppose $x_{j,1} = 1$ and $x_{j',1} = 0$ for all $j < j' \leq n$. Then $g(x_j) = 0$ and so we may set $s = j+1$. Otherwise if no 1 was seen, the interval stays $[1, n+1]$ as before. Note that in either case at most one 1 was seen.

If $s = t$, then output s (at this point, this can only happen if $j = n$ above). If the interval $[s,t]$ has size at least 2, we now query $x_{j,2}$ starting from $j = s$ up to $j = t-1$. Again we stop as soon as we see a 1. If a 1 was seen, then $g(x_j) = 1$ and we set $t = j$. We continue in this manner going back and forth until $s = t$ at which point we can output s because of our invariant. Additionally, if we are in the last iteration making queries to $x_{i,l}$ for $i \in [s, t-1]$ (in some order) as soon as we see a 1 or if all $x_{i,l} = 0$ for $i \in [s, t-1]$, we can give an output. Note that in each iteration, we see at most one 1. Since there are at most l iterations, this shows $\mathsf{D}^{\text{1-}\mathsf{dt}}(\mathcal{R} \circ \mathrm{EMB}_l) \leq l$. □

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.

References

[ABK16] 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*. 2016, pp. 863–876.

[AFS24] Yaroslav Alekseev, Yuval Filmus, and Alexander Smal. “Lifting Dichotomies”. In: *39th Computational Complexity Conference (CCC 2024)*. Ed. by Rahul Santhanam. Vol. 300. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024, 9:1–9:18. ISBN: 978-3-95977-331-7. DOI: 10.4230/LIPIcs.CCC.2024.9. URL: https://drops.dagstuhl.de/entities/

[AGJK+18] 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 fuer Informatik. 2018.

[AI25] 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. Prague, Czechia: Association for Computing Machinery, 2025, pp. 584–595. ISBN: 9798400715105. DOI: 10.1145/3717823.3718150. URL: https://doi.org/10.1145/3717823.3718150.

[BB20] 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)*. IEEE. 2020, pp. 403–411.

[BBGM22] 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)*. IEEE. 2022, pp. 624–635.

[BCD24] Sreejata Kishor Bhattacharya, Arkadev Chattopadhyay, and Pavel Dvořák. “Exponential Separation Between Powers of Regular and General Resolution over Parities”. In: *39th Computational Complexity Conference (CCC 2024)*. Ed. by Rahul Santhanam. Vol. 300. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024, 23:1–23:32. ISBN: 978-3-95977-331-7. DOI: 10.4230/LIPIcs.CCC.2024.23. URL: https://drops.dagstuh

[BDGH+20] 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.

[BdW02] Harry Buhrman and Ronald de Wolf. “Complexity measures and decision tree complexity: a survey”. In: *Theoretical Computer Science* 288.1 (2002), pp. 21–43.

[BGGMY25] 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: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025.

[BIW04] Eli Ben-Sasson, Russell Impagliazzo, and Avi Wigderson. “Near optimal separation of tree-like and general resolution”. In: *Combinatorica* 24.4 (2004), pp. 585–603.

[BK18] Shalev Ben-David and Robin Kothari. “Randomized Query Complexity of Sabotaged and Composed Functions”. In: *Theory of Computing* 14.5 (2018), pp. 1–27. DOI: 10.4086/toc.2018.v014a005. URL: https://theoryofcomputing.org/articles/v014a005.

[BK23] Paul Beame and Sajin Koroth. “On Disperser/Lifting Properties of the Index and Inner-Product Functions”. In: *14th Innovations in Theoretical Computer Science Conference (ITCS 2023)*. Ed. by Yael Tauman Kalai. Vol. 251. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, 14:1–14:17. ISBN: 978-3-95977-263-1. DOI: 10.4230/LIPIcs.ITCS.2023.14. URL: https://drops.dagstuhl.de/entities/document/10.423

[CDMRS23] Arkadev Chattopadhyay, Yogesh Dahiya, Nikhil S Mande, Jaikumar Radhakrishnan, and Swagato Sanyal. “Randomized versus deterministic decision tree size”. In: *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*. 2023, pp. 867–880.

[CFKMP21] Arkadev Chattopadhyay, Yuval Filmus, Sajin Koroth, Or Meir, and Toniann Pitassi. “Query-to-communication lifting using low-discrepancy gadgets”. In: *SIAM Journal on Computing* 50.1 (2021), pp. 171–210.

[CKLM19] Arkadev Chattopadhyay, Michal Kouckỳ, Bruno Loff, and Sagnik Mukhopadhyay. “Simulation theorems via pseudo-random properties”. In: *computational complexity* 28 (2019), pp. 617–659.

[CKMP+23] 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.

[CMP22] 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.

[CMSS23] 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.

[Dah24] Yogesh Dahiya. “Exploring Size Complexity and Randomness in the Query Model”. HBNI, 2024. URL: https://www.imsc.res.in/xmlui/handle/123456789/881.

[DM23] Yogesh Dahiya and Meena Mahajan. “On (simple) decision tree rank”. In: *Theoretical Computer Science* 978 (2023), p. 114177.

[EGI24] Klim Efremenko, Michal Garlik, and Dmitry Itsykson. “Lower Bounds for Regular Resolution over Parities”. In: *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*. 2024, pp. 640–651.

[EH89] Andrzej Ehrenfeucht and David Haussler. “Learning decision trees from random examples”. In: *Information and Computation* 82.3 (1989), pp. 231–246.

[GGKS18] 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*. 2018, pp. 902–911.

[GJPW18] Mika Göös, TS Jayram, Toniann Pitassi, and Thomas Watson. "Randomized communication versus partition number". In: *ACM Transactions on Computation Theory (TOCT)* 10.1 (2018), pp. 1–20.

[GKPW19] Mika Göös, Pritish Kamath, Toniann Pitassi, and Thomas Watson. "Query-to-communication lifting for P NP". In: *computational complexity* 28 (2019), pp. 113–144.

[GLSS23] Dmytro Gavinsky, Troy Lee, Miklos Santha, and Swagato Sanyal. "Optimal Composition Theorem for Randomized Query Complexity". In: *Theory of Computing* 19.1 (2023), pp. 1–35.

[GOR24] Svyatoslav Gryaznov, Sergei Ovcharov, and Artur Riazanov. "Resolution Over Linear Equations: Combinatorial Games for Tree-like Size and Space". In: *ACM Transactions on Computation Theory* 16.3 (2024), pp. 1–15.

[GP14] Mika Göös and Toniann Pitassi. "Communication lower bounds via critical block sensitivity". In: *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*. 2014, pp. 847–856.

[GP18] Mika Göös and Toniann Pitassi. "Communication Lower Bounds via Critical Block Sensitivity". In: *SIAM Journal on Computing* 47.5 (2018), pp. 1778–1806. DOI: 10.1137/16M1082007. URL: https://doi.org/10.1137/16M1082007.

[GPW18] Mika Göös, Toniann Pitassi, and Thomas Watson. "Deterministic communication vs. partition number". In: *SIAM Journal on Computing* 47.6 (2018), pp. 2435–2450.

[GPW20] Mika Göös, Toniann Pitassi, and Thomas Watson. "Query-to-communication lifting for BPP". In: *SIAM Journal on Computing* 49.4 (2020), FOCS17–441.

[IS20] Dmitry Itsykson and Dmitry Sokolov. "Resolution over linear equations modulo two". In: *Annals of Pure and Applied Logic* 171.1 (2020), p. 102722.

[JKS10] Rahul Jain, Hartmut Klauck, and Miklos Santha. "Optimal direct sum results for deterministic and randomized decision tree complexity". In: *Information Processing Letters* 110.20 (2010), pp. 893–897.

[LMMPZ22] Shachar Lovett, Raghu Meka, Ian Mertz, Toniann Pitassi, and Jiapeng Zhang. "Lifting with Sunflowers". In: *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*. Ed. by Mark Braverman. Vol. 215. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 104:1–104:24. ISBN: 978-3-95977-217-4. DOI: 10.4230/LIPIcs.ITCS.2 URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ITCS.2022.104.

[Mon14] Ashley Montanaro. "A composition theorem for decision tree complexity". In: *Chicago Journal of Theoretical Computer Science* 2014.6 (2014). DOI: 10.4086/cjtcs.2014.006.

[PI00] 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*. 2000, pp. 128–136.

[PR18] 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*. 2018, pp. 1207–1219.

[PS25] 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.

[RM97] Ran Raz and Pierre McKenzie. "Separation of the monotone NC hierarchy". In: *Proceedings 38th Annual Symposium on Foundations of Computer Science*. IEEE. 1997, pp. 234–243.

[San24] 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.

[Sav02] Petr Savický. *On determinism versus unambiquous nondeterminism for decision trees*. Tech. rep. TR02-009. Electronic Colloquium on Computational Complexity (ECCC), 2002. URL: https://eccc.weizmann.ac.il//report/2002/009/.

[She23] Suhail Sherif. *Lifting to Parity Decision Trees via Stifling (with applications to proof complexity)*. Proof Complexity and Meta-Mathematics Workshop, Simons Institute. Mar. 20, 2023. URL: https://www.youtube.com/watch?v=PeZVs6WUf-4 (visited on 11/13/2024).

[Tal13] Avishay Tal. "Properties and Applications of Boolean Function Composition". In: *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science (ITCS)*. 2013, pp. 441–454. DOI: 10.1145/2422436.2422485.

[Urq11] Alasdair Urquhart. "The depth of resolution proofs". In: *Studia Logica* 99 (2011), pp. 349–364.

A Proof complexity applications

In this section, we give simple proofs of some known lower bounds for tree-like Res($\oplus$), and some improvements to prior results on regular Res($\oplus$). We assume some familiarity with the proof system Resolution over parities and its subsystems. The reader may refer to the introduction of [AI25] and references therein.

For the lower bounds on tree-like Res($\oplus$) below, let us note what $\mathcal{R}^{\phi}_{*}$ looks like for a false clause search problem $\mathcal{R}^{\phi} \subseteq \{0,1\}^n \times [m]$ coming from a CNF formula ϕ on n variables and m clauses. The goal in solving $\mathcal{R}^{\phi}_{*}$ is to output an index $i \in [m]$ such that each literal appearing in the i^{th} clause is set to 0 or $*$ by the partial assignment (where we think of $\neg x$ as being $*$ if x is $*$).

A.1 Pigeonhole principle in tree-like Res($\oplus$)

Pigeonhole principle Itsykson and Sokolov [IS20] showed that the pigeonhole principle PHP^m_n with m pigeons and n holes ($m > n$) requires $2^{\Omega(n)}$ size tree-like Res($\oplus$) proofs. For this, they gave a parity Delayer strategy for the corresponding false clause search problem which scored $n/2$ points. A parity Delayer strategy scoring $n-1$ points is described in [GOR24, Lemma 3.3] where it is attributed to M. Garlík.

Here we give a simple Certifier strategy in the Blocker-Certifier game for PHP^m_n scoring n points. By Lemma 4.1, this implies that the rank of any PDT solving the false clause search problem associated with PHP^m_n is at least n (and this is tight since the usual decision tree solving PHP^{n+1}_n has rank n). We will use $x_{i,j}$, $i \in [m], j \in [n]$ to denote the variables of PHP^m_n where $x_{i,j}$ being 1 indicates that the pigeon i is sent to hole j.

In the beginning, Certifier fixes no variables. Suppose Blocker sets $x_{i,j} = *$. Then Certifier sets $x_{k,j} = 0$ for all $k \in [m] \setminus \{i\}$. Certifier uses the same strategy in each round to ensure that for every hole j, if there is some $i \in [m]$ such that $x_{i,j} = *$, then all the other pigeons do not fly to hole j ($x_{k,j} = 0$ for $k \neq i$). At any point, for any hole j, either all the variables corresponding to hole j

are set or all of them are unset. The number of holes whose variables have been fixed is precisely the number of rounds.

We claim that the game cannot end before n rounds. Indeed at any point, for any hole clause $\neg x_{i,j} \vee \neg x_{k,j}$ either both $x_{i,j}$ and $x_{k,j}$ are unset or at least one of them is set to 0. Also, if fewer than n rounds have been played, for any pigeon clause $\vee_{j\in[m]} x_{i,j}$, there must be some $j \in [m]$ such that $x_{i,j}$ is unset. So Certifier can always score n points.

Onto pigeonhole principle By considering the parity Blocker-Certifier game, we can show that the above lower bound also holds for the onto pigeonhole principle onto-PHP^m_n. In addition to the clauses of PHP^m_n, the onto pigeonhole principle contains clauses expressing that each hole receives at least one pigeon $\vee_{i\in[m]} x_{i,j}$ for all $j \in [n]$.

For the parity Blocker-Certifier game on onto-PHP^m_n, the variables are split into blocks according to the holes; for each $j \in [n]$, the variables $x_{i,j}, i \in [m]$ form a block. Certifier's strategy tries to ensure that exactly one pigeon flies into each hole. In the beginning, Certifier sets $\sum_{i\in[m]} x_{i,j} = 1$ for all $j \in [n]$. Now suppose Blocker picks the parity $\sum_{i\in S} x_{i,j}$ for some $S \subseteq [m], j \in [n]$. Note that since this parity must be linearly independent of any previously assigned parities, we must have $S \neq [m]$. So there exist $i_1 \in S$ and $i_2 \in [m] \setminus S$. Certifier now sets $x_{i,j} = 0$ for all $i \in [m] \setminus \{i_1, i_2\}$. Note that these parities are indeed linearly independent of the parities $\{\sum_{i\in[m]} x_{i,j}, \sum_{i\in S} x_{i,j}\}$ and thus together with them form a basis. So in the future, Blocker's moves must be on other blocks. The only solutions to these parity constraints are where exactly one of $x_{i_1,j}$ or $x_{i_2,j}$ is set to 1 and all other $x_{i,j}$ are 0. This ensures that none of the hole clauses or onto clauses are falsified.

As in the Certifier strategy for the (basic) pigeonhole principle, as long as Blocker has made fewer than n moves, no pigeon clause, say for pigeon i, is falsified, since there is at least one hole, say $j \in [n]$, where Blocker has not made a move and so Certifier may set $x_{i,j} = 1$. So the game cannot end before n rounds.

Functional pigeonhole principle Garlík and Kołodziejczyk showed that lower bounds for tree-like Res(⊕) can be obtained from polynomial calculus degree lower bounds, which implies that the functional pigeonhole principle requires $2^{\Omega(n)}$ size tree-like Res(⊕) proofs.

We will show that the rank of any PDT solving the false clause search problem associated with FPHP^m_n $(m > n)$ is at least $\lceil \frac{n+1}{2} \rceil$. The blocks in the parity Blocker-Certifier game will now be defined according to the pigeons. Similar to the strategy for onto-PHP, Certifier will try to ensure that whenever Blocker makes a move, the corresponding pigeon flies to exactly one hole. In doing so, we will also need to take care that no hole receives more than one pigeon. This latter step is where we lose a factor of 2.

We will show the bound by induction on n. For $n = 1$, the statement is clear. For $n = 2$, Certifier starts by setting for each pigeon i, $\sum_{j\in[n]} x_{i,j} = 1$. Since $n = 2$, this ensures that each pigeon flies into exactly one hole. So the only clauses which could be falsified are the hole clauses. Observe that the game cannot end after the first round since Blocker's move only affects one pigeon and a hole clause involves two pigeons. So Certifier scores at least 2 points.

Now consider $n \geq 3$. As before, Certifier starts by setting for each pigeon i, $\sum_{j\in[n]} x_{i,j} = 1$. Suppose Blocker picks a parity $\sum_{j\in S} x_{i,j}$ for some $i \in [m], S \subseteq [n]$. Again since this parity must be linearly independent of previous parities, S cannot be the empty set or $[n]$. So there exist $j_1 \in S, j_2 \notin S$. Certifier sets $x_{i,j} = 0$ for $j \in [n] \setminus \{j_1, j_2\}$. Additionally he sets $x_{i',j_1} = x_{i',j_2} = 0$ for all $i' \in [m] \setminus \{i\}$. Note that this effectively reduces the problem to FPHP on pigeons $[m] \setminus \{i\}$ and holes $[n] \setminus \{j_1, j_2\}$. Indeed pigeon i is effectively forced to go to exactly one of two holes and no other pigeon flies to these two holes. Now by induction, the game goes on for at least $\lceil \frac{(n-2)+1}{2} \rceil$

additional rounds. So Certifier scores at least $\lceil \frac{n+1}{2} \rceil$ points totally.

Bit pigeonhole principle For the bit pigeonhole principle BPHP_n^m where $n = 2^k$ for some positive integer k and $m > n$, Efremenko, Garlík and Itsykson [EGI24] gave a parity Delayer strategy scoring $n/4$ points. Underlying their Delayer strategy is a Certifier strategy, though their proof uses this Certifier strategy in a more delayed manner, compared to the Delayer strategy that would be obtained by directly combining the Certifier strategy described next and the proof of Lemma 4.1.

The formula is on the variables $x_{i,j}$ $(i \in [m], j \in [k])$. For each $i \in [m]$, x_i (viewed as a k-bit integer) encodes which hole in $[n]$ pigeon i flies to. We now explain the Certifier strategy used in their proof. Suppose Blocker sets $x_{i,j} = *$. Then Certifier fixes all bits in x_i (other than $x_{i,j}$) so that the pigeon i can only go to one of two holes. Certifier will ensure that these two holes are different from any holes that have previously been assigned to some pigeon. This can be done as long as the number of rounds (excluding the current round), say t, satisfies $2t < n/2$. This is because in each round we use up 2 new holes and there are $n/2$ pairs of holes where a pair contains two holes differing only on the j^{th} bit. So this property can be fulfilled at least for the first $n/4$ rounds. The game cannot end earlier as we explain now. Any clause which involves some pigeon that has not been queried at all cannot be falsified. Additionally, a clause involving two pigeons who have been assigned holes is already satisfied because of the Certifier strategy. So Certifier scores at least $n/4$ points.

A.2 Ordering principle in tree-like Res($\oplus$)

A Delayer strategy scoring $n - 2$ points for the ordering principle Ord_n was shown in [GOR24]. The ordering principle encodes the fact that every finite total order has a least element. We have variables $x_{i,j}$ $(i, j \in [n], i \neq j)$ where $x_{i,j} = 1$ encodes that $i \prec j$ in the linear order. The clauses of Ord_n encode the conditions of being a total order and for each $i \in [n]$, that i is not the least element.

We cannot directly use the Blocker-Certifier game to prove a lower bound for Ord_n since Blocker has a simple strategy to end the game in 2 rounds. Blocker picks any unset variable $x_{i,j}$ and sets it to $*$. In the next round, if $x_{j,i}$ is unset, then Blocker sets $x_{j,i} = *$. Otherwise, Certifier must have fixed $x_{j,i}$ in one of the two rounds. In either case, the game has ended since one of the clauses $x_{i,j} \vee x_{j,i}$ or $\neg x_{i,j} \vee \neg x_{j,i}$ can be falsified by Blocker. So any Certifier strategy can only ensure at most 2 points.

However, we can give a better Certifier strategy if we first move to an affine restriction such that Blocker can no longer use this trivial strategy which falsifies one of the totality or anti-symmetry conditions. Specifically, for all $i \neq j$, we set $x_{i,j} + x_{j,i} = 1$. Under this restriction, we may think of the game as being played only over the variables $x_{i,j}$ $(1 \leq i < j \leq n)$. (We could view this as a special case of the parity Blocker-Certifier game where we have a block for each pair $i \neq j$ containing only the variables $x_{i,j}$ and $x_{j,i}$.)

We can now give a Certifier strategy scoring $n/2$ points which recovers the lower bound from [GOR24] up to a constant factor. At a high level, Certifier ensures that whenever Blocker sets $x_{i,j} = *$, in the underlying order, i and j must be the largest among the elements which have not been touched by Blocker so far. Since Blocker can affect only two elements in a round, the game must last at least $n/2$ rounds. We now explain the strategy in more detail. We will use $S \subseteq [n]$ to denote the current set of possible least elements. Effectively the game at any stage has been reduced to the game on just the elements in S. Certifier will always ensure that the only unset variables $x_{i,j}$ satisfy $i, j \in S$ and any transitivity constraint involving some element outside S is satisfied.

Initially $S = [n]$ and Certifier does not fix any variables in the first round. Suppose in the $(i-1)^{th}$ round (where $i < \lceil n/2 \rceil$) Blocker sets $x_{i,j} = *$ where both $i, j \in S$. Then update $S := S \setminus \{i, j\}$. For each $k \in S$, Certifier sets variables so that $k \prec i$ and $k \prec j$. This ensures that any transitivity constraints involving i or j and some element(s) from S are satisfied. By our invariant, this implies that all transitivity constraints involving i or j are satisfied. Additionally, the clauses expressing that i and j are not the least element are also satisfied. Moreover, since $i < \lceil n/2 \rceil$, S still contains at least 3 elements. So no matter which pair i', j' in S Blocker picks in the i^{th} round, the game cannot end since every clause which can potentially still be falsified also contains some $k' \in S \setminus \{i', j'\}$. Thus, the game must last at least $\lceil n/2 \rceil$ rounds. (In the $\lceil n/2 \rceil^{th}$ round, to avoid falsifying a clause, Certifier simply skips his move.)

A.3 Separations between tree-like Res($\oplus$) and (regular) resolution

Itsykson and Sokolov [IS20] used the lifting theorem for versatile gadgets [GP14] to give a formula which has a regular resolution proof of size $O(n)$ but requires tree-like Res($\oplus$) proofs of $2^{\Omega(\sqrt{n})}$. Using the lifting theorem for deterministic PDTs [CMSS23, BK23], one can lift the nearly optimal separation between tree-like resolution and regular resolution [BIW04] to get a nearly optimal separation between tree-like Res($\oplus$) and regular resolution.

We observe that the formula of Ben-Sasson, Impagliazzo and Wigderson [BIW04] already provides the desired separation. Their formula is an OR-lift of a pebbling formula. Their proof can be understood as implicitly combining the fact that OR lifts 1-depth to DT rank and that the 1-depth of the search problem associated with a pebbling formula is at least the pebbling number of the underlying graph up to an additive constant. Finally, to get their $2^{\Omega\left(\frac{n}{\log n}\right)}$ lower bound they use a graph whose pebbling number is $\Omega\left(\frac{n}{\log n}\right)$. It follows from Remark 6.5 that $\mathsf{DRank}^{\oplus\text{-}\mathsf{dt}}(\mathcal{R} \circ \vee_2) = \mathsf{DRank}^{\mathsf{dt}}(\mathcal{R} \circ \vee_2) = \mathsf{D}^{1\text{-}\mathsf{dt}}(\mathcal{R})$ for any relation $\mathcal{R}$. In particular, taking $\mathcal{R}$ to be the search problem associated with the pebbling formula on a graph with pebbling number $\Omega(n/\log n)$, the same formula also requires $2^{\Omega\left(\frac{n}{\log n}\right)}$ size tree-like Res($\oplus$) proofs.

A.4 Separation between regular Res($\oplus$) and ordinary resolution

Bhattacharya, Chattopadhyay and Dvořák [BCD24] gave an exponential separation between regular Res($\oplus$) and general Resolution. Specifically they gave a family of formulas on M variables, which has Resolution proofs of size poly(M) but requires bottom-regular proofs of size $\exp(\Omega(M^{1/12}/\log^{13/12} M))$. Later Alekseev and Itsykson [AI25] gave a different family of formulas on M variables which have poly(M) size Resolution proofs but require regular Res($\oplus$) proofs of size $\exp(\Omega(M^{1/2}))$. These lower bounds are very strong, but the upper bounds are fairly large polynomials.

The upper bound in [BCD24] is a large polynomial since they use an existing randomized query-to-communication lifting theorem which requires a large gadget. By using Theorem 1.1 instead, we can replace the large gadget with a constant size gadget like IP_4. This improves the upper bound to $O(M^2)$ which is linear in the total number of clauses. The lower bound slightly improves to $\exp(\Omega(M^{1/12}))$.

We briefly sketch the changes needed for the lower bound. The lower bound uses a distributional version of Theorem 1.1. This is standard and easy to obtain using the simulation in the proof, but we sketch it for completeness. Let g be a constant size stifling gadget and (μ_0, μ_1) be distributions showing that g is p-affine balanced (uniform distributions would suffice for constant size g) for some constant $p > 0$. Suppose a deterministic parity decision tree T of worst-case depth d computes a relation $\mathcal{R} \circ g$ on a distribution $\eta \circ (\mu_0, \mu_1)$ to some error ϵ, where η is some distribution on the inputs of $\mathcal{R}$. Then the simulation applied to T gives a randomized decision tree whose expected depth on any input is $O(d)$ and computes $\mathcal{R}$ to error ϵ with respect to the distribution η and the randomness of the tree. By terminating this randomized decision tree we can ensure that its worst case depth is $O(d)$ and the error on η is at most, say, $\epsilon + 1/10$. By averaging, there must be a deterministic decision tree whose depth is $O(d)$ and computes $\mathcal{R}$ to error $\epsilon + 1/10$ on distribution η.

Also note that though the distributions above are not necessarily uniform on $g^{-1}(0)$ and $g^{-1}(1)$, Lemma 20 in [BCD24] continues to hold for these distributions since the only property needed in the proof is that these distributions are balanced.

A.5 Bit pigeonhole principle in regular Res($\oplus$)

Efremenko, Garlík and Itsykson [EGI24] showed that the bit pigeonhole principle BPHP_n^{n+1} requires $2^{\Omega\left(\frac{\sqrt[3]{n}}{\log n}\right)}$ size regular Res($\oplus$) proofs. Their analysis can be further sharpened to get a $2^{\Omega\left(\frac{\sqrt{n}}{\log n}\right)}$ lower bound. This improvement comes from analyzing the random walk in Lemmas 6.1 and 6.2 [EGI24] (full version) slightly differently.

Informally, at the t^{th} step of the random walk, the size of the closure can increase by t which they use to give a bound of $O(t^2/n)$ on the probability that the assignment is no longer injective on the new closure, conditioned on being injective on the previous closure. Then by the union bound, they conclude that the probability that a random assignment is not injective on the closure of the clause reached after taking t steps is at most $O(t^3/n)$. The slack in the inequality seems to come from the fact that even though the size of the closure can increase by more than 1 at a step, this cannot be true at all steps since the size of the closure is bounded by the total number of steps taken so far. So one may hope that a more careful analysis should give an upper bound of $O(t^2/n)$ on the probability of not being injective on the closure instead of $O(t^3/n)$.

While we do not know how to directly work with closure to prove the better bound, we can give the desired $O(t^2/n)$ upper bound on the probability of the random walk not being 'good', for a notion of good which is different from the assignment being injective on the closure. Even though our definition of being good is different from the one in [EGI24], we do rely on arguments used in proving their Lemma 6.2. For the rest of the argument of [EGI24] to go through, at the end of the random walk, we need to switch back from this notion of good to being locally consistent. This can be done as long as the number of steps t is not too large by using some other ideas in [EGI24].

The analysis below uses ideas similar to those for the randomized PDT simulation, Proposition 3.9 where at a parity query, we reveal all bits of a block other than one which appears in that parity. For BPHP_n^{n+1}, each block x_i indicates the hole pigeon i flies to. For such a block where all but one of the bits have been fixed, we think of the pigeon as being sent to both possible holes. Let ρ be a partial assignment of the variables $x_{i,j}$ such that for each $i \in [n]$, either $\rho_i = *^k$ or exactly one position in ρ_i is $*$ and the rest are fixed to 0/1. Let $\tilde{\rho} : [n+1] \to \binom{[n]}{2} \cup \{*\}$ be the partial function expressing which holes a pigeon flies to under ρ. Each pigeon i such that $\rho_i = *^k$ is not in the domain of $\tilde{\rho}$ while every other pigeon is mapped to the set of the two holes it has been assigned. Say that $\tilde{\rho}$ is 2-injective if there is no hole in $[n]$ such that two pigeons are mapped to it under $\tilde{\rho}$.

The notion of goodness will rely on a simulation of a PDT on the uniform distribution. To describe it, we view all the variables $x_{i,j}$ ($i \in [n+1], j \in [k]$ where $n = 2^k$) as being ordered lexicographically according to the indices. This is to allow picking the first variable occurring in a parity. Let T be a deterministic PDT on the variables $x_{i,j}$. Let $\sigma \in (\{0,1\}^k)^{n+1}$ be a total assignment. For any $r \geq 0$, we associate a partial assignment ρ_r, a set of marked pigeons $M_r \subseteq [n+1]$ and a (marked) system Φ_r of linear equations with σ in the following way. Informally, ρ_r, M_r and Φ_r capture the information we will reveal in the first r steps of the random walk to analyze the desired probability. We will always have $|M_r| \leq r$ and for each pigeon $i \in [n+1]$, if $i \in M$, ρ_i is fixed according to σ except for one bit remaining $*$, and otherwise $\rho_i = *^k$. Additionally, the variables which are free in ρ but belong to the marked pigeons form pivots of the system Φ_r. We say that these variables are marked. The assignment σ satisfies Φ_i.

ρ_r, M_r and Φ_r are defined inductively in the following way. For $i = 0$, $\rho_0 = *^{k(n+1)}, M_i$ and Φ_0 are empty. Now suppose $i > 0$. If the leaf reached by σ has depth at most $i-1$ (the root is at depth 0), set $M_i = M_{i-1}, \rho_i = \rho_{i-1}, \Phi_i = \Phi_{i-1}$.

Otherwise consider the node in T at depth $i-1$ which is reached by σ. Let $\sum_{(p,j)\in S} x_{p,j}$ (for some $S \subseteq [n+1] \times [k]$) be the parity at that node. We simplify this parity so that it only depends on pigeons outside M_{i-1} by substituting according to Φ_{i-1} and ρ_{i-1}. Suppose the cleaned-up parity is $b + \sum_{(p,j)\in S'} x_{p,j}$ for some $b \in \mathbb{F}_2, S' \subseteq [n+1] \times [k]$. If $S' = \emptyset$, set $M_i = M_{i-1}, \rho_i = \rho_{i-1}, \Phi_i = \Phi_{i-1}$. Suppose $S' \neq \emptyset$. Let (p', j') be the first variable appearing in S'. Then $M_i = M_{i-1} \cup \{p'\}$ and $\Phi_i = \Phi_{i-1} \cup \{\sum_{(p,j)\in S'} x_{p,j} = \sum_{(p,j)\in S'} \sigma_{p,j}\}$. Let ρ_i be the same as ρ_{i-1}, except that for $j \in [k]\backslash\{j'\}$, $\rho_{i',j} = \sigma_{i',j}$. Note that all the properties of M_i, ρ_i and Φ_i stated above are satisfied.

We can now describe when an assignment is *good*. Say that a total assignment σ is r-good for $r \geq 0$ if the partial assignment ρ_r is 2-injective. Note that since ρ_{r+1} extends ρ_r, if ρ_{r+1} is 2-injective, then ρ_r must also be 2-injective. We will show the following.

Lemma A.1. Let $t \geq 0$ be an integer. For a total assignment σ in $(\{0,1\}^k)^{n+1}$ picked uniformly at random,

$$\Pr_\sigma[\sigma \text{ is not } t\text{-good}] \leq \frac{2t^2}{n}.$$

Our proof will follow the same high level structure of the proofs of Lemmas 6.1 and 6.2 in [EGI24].

Proof. We will show that for any $r \geq 1$,

$$\Pr_\sigma[\sigma \text{ is not } r\text{-good}|\sigma \text{ is } (r-1)\text{-good}] \leq \frac{4(r-1)}{n}.$$

This immediately implies the statement of the lemma as shown below.

$$\begin{aligned}
\Pr_\sigma[\sigma \text{ is not } t\text{-good}] &= \sum_{r=1}^{t} \Pr[\sigma \text{ is not } r\text{-good, but } \sigma \text{ is } (r-1)\text{-good}] \\
&= \sum_{r=1}^{t} \Pr[\sigma \text{ is not } r\text{-good}|\sigma \text{ is } (r-1)\text{-good}] \cdot \Pr[\sigma \text{ is } (r-1)\text{-good}] \\
&\leq \sum_{r=1}^{t} \Pr[\sigma \text{ is not } r\text{-good}|\sigma \text{ is } (r-1)\text{-good}] \\
&\leq \sum_{r=1}^{t} \frac{4(r-1)}{n} \leq \frac{2t^2}{n}.
\end{aligned}$$

So we only need to show $\Pr_\sigma[\sigma \text{ is not } r\text{-good}|\sigma \text{ is } (r-1)\text{-good}] \leq \frac{4(r-1)}{n}$. For this, we will condition on any possible $\rho'_{r-1}, M'_{r-1}, \Phi'_{r-1}$ such that ρ'_{r-1} is 2-injective. (We use $'$ to distinguish these fixed objects from the random variables $\rho_{r-1}, M_{r-1}, \Phi_{r-1}$ associated with σ.) By the definitions of these objects, note that $\rho'_{r-1}, M'_{r-1}, \Phi'_{r-1}$ uniquely determine the node at depth $r-1$ (or possibly a leaf at depth at most $r-1$) reached by any assignment σ which is consistent with ρ'_{r-1} and satisfies Φ'_{r-1}.

We wish to show that $\Pr[\sigma \text{ is not } r\text{-good} | \rho_{r-1} = \rho'_{r-1}, M_{r-1} = M'_{r-1}, \Phi_{r-1} = \Phi'_{r-1}] \leq \frac{4(r-1)}{n}$. If ρ'_{r-1}, Φ'_{r-1} are such that any assignment consistent with them reaches a leaf at depth $r-1$ or less, then this probability is 0 since in this case $\rho_r = \rho_{r-1} = \rho'_{r-1}$ which is 2-injective. So suppose the node in T at depth $r-1$ reached by assignments consistent with ρ'_{r-1} and satisfying Φ'_{r-1} is an internal node. Clean up this parity according to ρ'_{r-1} and Φ'_{r-1} to get a parity $b + \sum_{(i,j)\in S} x_{i,j}$ which does not contain variables involving pigeons from M'_{r-1}. If $S = \emptyset$, then again the probability is 0 since $\rho_r = \rho_{r-1}$.

So the only non-trivial case is when $S \neq \emptyset$. Let (i,j) be the first variable in S. Note that the distribution of σ on all blocks except those in M'_{r-1}, after conditioning on σ being consistent with ρ'_{r-1} and satisfying Φ'_{r-1}, is still uniform. This follows from the fact that we started with the uniform distribution, the marked variables form pivots of Φ'_{r-1} and the only fixed variables in ρ'_{r-1} are the other variables in the marked blocks. So we only need to estimate the probability that a uniform assignment to the variables $\{x_{i,j'} | j' \in [k], j' \neq j\}$ sends pigeon i to a hole already containing some pigeon according to ρ'_{i-1}. Since $|M'_{r-1}| \leq r-1$, at most $2(r-1)$ holes are occupied already. So the probability that ρ_r is not 2-injective is the probability that pigeon i gets mapped to a pair containing one of these at most $2(r-1)$ holes. This probability is at most $\frac{2(r-1)}{n/2}$ since the distribution is uniform.

This proves $\Pr[\sigma \text{ is not } r\text{-good} | \rho_{r-1} = \rho'_{r-1}, M_{r-1} = M'_{r-1}, \Phi_{r-1} = \Phi'_{r-1}] \leq \frac{4(r-1)}{n}$ for all $\rho'_{r-1}, M'_{r-1}, \Phi'_{r-1}$ where ρ'_{r-1} is 2-injective. In particular,

$$\begin{aligned}\Pr_{\sigma}[\sigma \text{ is not } r\text{-good} | \sigma \text{ is } (r-1)\text{-good}] &\leq \max \Pr[\sigma \text{ is not } r\text{-good} | \rho_{r-1} = \rho'_{r-1}, M_{r-1} = M'_{r-1}, \Phi_{r-1} = \Phi'_{r-1}] \\ &\leq \frac{4(r-1)}{n}\end{aligned}$$

where the maximum is over all $\rho'_{r-1}, M'_{r-1}, \Phi'_{r-1}$ such that ρ'_{r-1} is 2-injective. This finishes the proof. □

Next, we need to show that being good is sufficient for local consistency.

Lemma A.2. Let ρ be a 2-injective partial assignment, M the set of marked pigeons in ρ and Φ a marked linear system whose pivots are precisely the variables in the blocks lying in M which are not fixed in ρ. Let Φ' be a linear system which is implied by $\rho \cup \Phi$ where we view ρ as a collection of $|M|(k-1)$ linear equations each fixing a variable of a marked pigeon (which is not a pivot of Φ). Suppose the rank of Φ' is at most $n/2$. Then Φ' is locally consistent.

This lemma uses ideas from Lemma 7.1 in [EGI24].

Proof. Since the system $\rho \cup \Phi$ implies Φ', it is sufficient to find a solution of $\rho \cup \Phi$ which is injective on $\mathrm{Cl}(\Phi')$. Any such solution σ must agree with ρ and, in particular, is already determined for the pigeons in $M \cap \mathrm{Cl}(\Phi')$ except for the pivots of Φ. Since ρ is 2-injective, no matter what values these pivots take, if we can find distinct holes for the pigeons in $\mathrm{Cl}(\Phi') \setminus M$ which also differ from any of the possible holes assigned to pigeons in $M \cap \mathrm{Cl}(\Phi')$, we will be done.

We can find such holes if $|\operatorname{Cl}(\Phi') \setminus M| \le n - 2|\operatorname{Cl}(\Phi') \cap M|$. This holds since

$$\begin{aligned}|\operatorname{Cl}(\Phi') \setminus M| + 2|\operatorname{Cl}(\Phi') \cap M| &\le 2(|\operatorname{Cl}(\Phi') \setminus M| + |\operatorname{Cl}(\Phi') \cap M|)\\ &\le 2|\operatorname{Cl}(\Phi')|\\ &\le 2\operatorname{rank}(\Phi') \le n.\end{aligned}$$

We assign any distinct holes to the pigeons in $\operatorname{Cl}(\Phi') \setminus M$ different from the holes assigned to pigeons in $M \cap \operatorname{Cl}(\Phi')$. To get a total assignment, we set all bits that have not already been set (except for the pivots of Φ) arbitrarily and extend this to a solution of Φ by setting the pivots appropriately. □

We now sketch the remaining changes needed to get a $2^{\Omega\left(\frac{\sqrt{n}}{\log n}\right)}$ lower bound on the size of regular Res(⊕) proofs of BPHP_n^{n+1}. Let $t := \lfloor \sqrt{n}/2 \rfloor$. Consider a regular Res(⊕) refutation graph and the associated linear branching program solving the search problem on BPHP_n^{n+1}. Let σ be a random assignment and take t steps on the branching program according to σ. If we expand this initial depth t segment of the branching program to a PDT, Lemma A.1 tells us that σ is t-good except with probability at most $2t^2/n \le 1/2$. Let Φ' be the linear clause in the refutation graph after taking t steps. If σ is t-good, then by Lemma A.2, Φ' must be locally consistent since we always ensure that ρ_t and Φ_t (as used in the definition of t-good) together imply the t equations along the edges of the branching program (starting from the source), which in turn imply the system Φ'. So with probability at least $1/2$, the linear clause reached by a random assignment is locally consistent.

The rest of the argument stays the same. The rank of Φ' is at least $\Omega\left(\frac{\sqrt{n}}{\log n}\right)$ by Lemma 7.2 in [EGI24]. This implies that the size of such a proof must be at least $2^{\Omega\left(\frac{\sqrt{n}}{\log n}\right)}$.

B Composition for block decision trees

In this section, we adapt known composition theorems which work for ordinary decision trees to get composition theorems for more general decision trees when the queries come from a set of allowed queries and each query must be contained in a block.

Let M be a finite set. A query on M is described by a tuple $(S, (S_i, c_i)_{i \in [k]})$ where $S \subseteq M$, S_i's form a partition of S and each $c_i \in \{0, 1\}$. Here S is the collection of possible inputs after having made some queries and each S_i is a possible response to the query. The integer c_i is the cost on getting response S_i. We call S the head of the query. We will only allow non-trivial queries where the partition has at least two parts and each part is non-empty. A query set Q on M is a collection of queries. A query set may contain multiple queries with the same underlying partition but with different costs associated to the parts.

A Q-decision tree on M with outputs in $\mathcal{O}$ is a decision tree whose internal nodes are labeled by queries from Q and leaves are labeled by pairs (S, o) where $S \subseteq M$ and $o \in \mathcal{O}$. The labels of the nodes must be consistent in the following natural sense. An internal node v labeled by a query $(S, (S_i, c_i)_{i \in [k]})$ has exactly k children w_i ($i \in [k]$), where w_i is labeled with a query with head S_i. We think of the edge between v and w_i as being marked if $c_i = 1$. In this case, we say that w_i is a marked node. The head of the query at the root node must be M as all inputs are possible at the root.

We will call a Q-decision tree a Q-tree for short. We will assume below that Q has the following properties:

1. Q is complete in the following sense: any subset S_i, which appears as a response in Q and has $|S_i| \geq 2$, must also appear as the head of some query in Q. This would guarantee that any function on M can be computed by some Q-tree.

2. Q is closed under restrictions. Let $S' \subseteq M$ which can appear as the response of some query in Q and $|S'| \geq 2$. Let $q = (S, (S_i, c_i)_{i \in [k]})$ be a query in Q where $S' \subseteq S$. Then one of the following holds:

    - $S' \subseteq S_i$ for some $i \in [k]$, or,
    - Q contains a query $q' = (S', (U_j, d_j)_{j \in [l]})$ such that for each $j \in [l]$, there exists $i \in [k]$ with $U_j \subseteq S_i$ and $d_j \leq c_i$.

The first condition above ensures that no matter what queries have been made previously there is a sequence of future queries which will allow us to uniquely identify the input. The second condition ensures that for every query q with head S and every subset $S' \subseteq S$ which could occur, there is a query q' on S' which at least gives as much information as q and is not more expensive.

The cost of a Q-tree T for M on input $x \in M$, denoted $\mathrm{cost}(T, x)$, is defined as the number of marked edges on the unique root-to-leaf path in T which is consistent with x. The deterministic Q-query complexity of a relation $\mathcal{R} \subseteq M \times \mathcal{O}$, denoted $\mathsf{D}^Q(\mathcal{R})$, is the minimum cost of a Q-tree computing $\mathcal{R}$. For a randomized Q-tree $\mathcal{T}$, cost is defined in the following manner. For an input x, $\mathrm{cost}(\mathcal{T}, x) = \mathbb{E}_{T \sim \mathcal{T}}[\mathrm{cost}(T, x)]$. We define expected randomized Q-query complexity of $\mathcal{R}$ by $\overline{\mathsf{R}}^Q_\epsilon(\mathcal{R}) = \min_{\mathcal{T}} \max_x \mathrm{cost}(\mathcal{T}, x)$ where $\mathcal{T}$ varies over all randomized Q-trees solving $\mathcal{R}$ with error at most ϵ.

The next claim will be useful in later sections. Informally, it says that a search problem can only become easier when restricted to a subset of inputs.

Lemma B.1. *Let $S \subseteq S' \subseteq M$ where both S and S' can be obtained by some sequence of queries. Let T be a Q-tree on the set S' which computes a relation $\mathcal{R} \subseteq S' \times \mathcal{O}$. Then there exists a Q-tree T' computing $\mathcal{R}$ restricted to S such that for all $x \in S$, $\mathrm{cost}(T', x) \leq \mathrm{cost}(T, x)$.*

Proof. We will show this by induction on $|S|$. If $|S| = 1$, this is clear since we can take T' to be a leaf.

Suppose $|S| \geq 2$. As long as S is completely contained in a response to the current query in T, move to that child. If we reach a leaf while doing this, we can let T' be a leaf with the same output. Otherwise we reach a query q such that S is split non-trivially by q. Suppose the subtrees at this node are $T_1, T_2, \ldots, T_l$. Since Q is closed under restrictions, there is some query q' which refines q on S. The tree T' will have q' at the root. For each response A to q', we can use a tree T'_i obtained by using the induction hypothesis on the tree T_j (whose head is, say, S_j) with the sets $S_j \cap S$ and $S_j \cap S'$, where T_j corresponds to the response of q containing A.

Let $x \in S$. Suppose x goes to the subtree T_j of T after the first query. Suppose the subtree in T' reached by x is T'_i. By induction, $\mathrm{cost}(T'_i, x) \leq \mathrm{cost}(T_j, x)$. Additionally by the choice of the query at the root, the cost of the response in T' at the root on x is at most the cost of the response in T. So $\mathrm{cost}(T', x) \leq \mathrm{cost}(T, x)$. □

We now extend the above definitions to relations $\mathcal{R} \subseteq M^n \times \mathcal{O}$. In a Q-decision tree for M^n, each internal node v is labeled by a tuple (S, i, q) where $S = S_1 \times S_2 \times \cdots \times S_n \subseteq M^n$, $i \in [n]$ and $q = (S_i, (S'_j, c_j)_{j \in [k]}) \in Q$. Here S is the set of inputs reaching v and q is the query made at that node into block i. Such a node v has exactly k children, w_j, $j \in [k]$, where w_i is labeled with the set $S_1 \times S_2 \ldots S_{i-1} \times S'_j \times S_{i+1} \times \ldots S_n$. Informally, the tuple (S, i, q) corresponds to making the query q in the i^{th} block when the current set of possible inputs is S. We define marked edges and marked nodes as before. $\mathsf{D}^Q(\mathcal{R})$ and $\mathsf{R}^Q(\mathcal{R})$ are defined similarly for relations $\mathcal{R} \subseteq M^n \times \mathcal{O}$. For an input $x \in M^n$, we will continue to say the i^{th} block of x to refer to x_i.

Before stating the composition theorems, let us state how the above definition captures the query models considered in this work.

- Ordinary decision trees: Here each node corresponds to a subcube of $\{0,1\}^n$ and each query partitions the current subcube into two subcubes based on a coordinate. Depending on how the costs are defined, we can get ordinary depth, 0-depth, 1-depth or rank.
- $*$-depth: Similar to ordinary decision trees, here each node corresponds to a subcube in $\{0,1,*\}^n$ and the cost is 1 only when the coordinate is fixed to $*$
- $(\oplus, *)$-decision trees: The definition of $(\oplus, *)$-decision trees explicitly works with a partial subspace as the current set of inputs and the partition is induced by how a parity interacts with the current partial subspace.

B.1 Composition for deterministic block decision trees

The following proposition has the same proof as the usual composition theorem for deterministic decision trees, so we only provide a sketch.

Proposition B.2 (essentially [Sav02, Tal13, Mon14])**.** For any relation $\mathcal{R} \subseteq \{0,1\}^n \subseteq \mathcal{O}$, any function $g : M \to \{0,1,*\}$,

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

Proof sketch. Consider an (ordinary) Adversary strategy for $\mathcal{R}$ and a Q-Adversary strategy for g. The strategy for g ensures that the value of g is undetermined as long as the cost so far is less than $\mathsf{D}^Q(g)$. We now combine these to give a Q-Adversary strategy for $\mathcal{R} \circ g$. We have n separate Q-adversaries, one for each block. On a query to the i^{th} block, if the block has already been fixed earlier, respond accordingly. Otherwise, respond according to the Adversary strategy for g on that block. If at this point, the total cost in the block has reached $\mathsf{D}^Q(g) - 1$, fix x_i so that $g(x_i)$ is set according to the Adversary strategy for $\mathcal{R}$. The number of fixed blocks at the end must be at least $\mathsf{D}^{\mathsf{dt}}(\mathcal{R})$ and each such block contributes $\mathsf{D}^Q(g) - 1$ to the total cost. □

We next show an equivalence between the Querier-Adversary game and a generalization of the Blocker-Certifier game. Let Q be a query set such that for every query $(S, (S_i, c_i)_{i \in [k]})$ in Q, there is a unique $i \in [k]$ such that $c_i = 1$ (and for all $j \neq i$, $c_j = 0$). From now on we will assume that in all queries $(S, (S_i, c_i)_{i \in [k]})$, $c_1 = 1$ and $c_j = 0$ for $j \geq 2$. For lack of a better name, we also call this game the Blocker-Certifier game. The Blocker-Certifier game for the relation $\mathcal{R} \subseteq M \times \mathcal{O}$ with query set Q is played in rounds. The game is played on a set $S \subseteq M$ which changes with each move. In the beginning $S = M$. In a round,

1. Certifier updates S to be some subset $S' \subseteq S$ such that there is a sequence of queries $q_1, q_2, \ldots, q_k$ and responses $S \supset S'_1 \supset S'_2 \cdots \supset S'_k = S'$ such that S is the head of q_1 and for each $i \in [k]$, S'_i is a response for the query q_i with cost 0. We also allow Certifier to leave S unchanged.
2. Blocker picks a query $(S, (S_i, c_i)_{i \in [k]})$ and updates S to be S_1.

The game ends when there is some $o \in \mathcal{O}$, such that $(x, o) \in \mathcal{R}$ for each $x \in S$. Certifier's score is the number of rounds played.

Lemma B.3. Suppose Q satisfies the above property that every query has unique response of cost 1. Let $\mathcal{R} \subseteq M \times \mathcal{O}$. The following are equivalent:

- Adversary has a strategy to score k points in the Querier-Adversary game on $\mathcal{R}$ with query set Q.
- Certifier has a strategy to score k points in the Blocker-Certifier game on $\mathcal{R}$ with query set Q.

Proof. Suppose we have an Adversary strategy scoring k points. Certifier copies Adversary's strategy in the following way. Suppose the current set is S in the Blocker-Certifier game (in the beginning $S = M$). Consider all the queries $q_1, q_2, \ldots, q_k$ whose head is S. If there exists a query q_i such that Adversary responds to it with a cost 0 set S', then Certifier's sequence would start with q_i and S'. Certifier repeats this with S' to (possibly) get another query q' (with head S') such that Adversary responds to q' with S'' which has cost 0. Certifier adds q' and S'' to his current sequence. He continues in this way until he reaches a set such that for all queries on it, Adversary responds with the cost 1 response. When this happens, Certifier ends his turn.

Here Certifier can be thought of as playing the role of Querier but only making queries to which the responses have cost 0. Therefore the score that Adversary can score on the obtained set remains unchanged. Since Certifier makes sure that no matter what query Blocker makes, the corresponding response of Adversary would score 1, Certifier achieves the same score as Adversary.

For the other direction, we will show that if Querier has a strategy which allows Adversary to score at most k points, then Blocker also has a strategy to force Certifier to score at most k points. Querier's strategy can be described as a Q-tree T for $\mathcal{R}$ of cost k. Blocker will essentially simulate this decision tree. To do this, we will need to use Lemma B.1 which follows from the assumption that Q is closed under restrictions.

We now explain Blocker's strategy. Suppose after Certifier makes his first move, the current set is S. By Lemma B.1, there is a tree T' on S computing $\mathcal{R}$ whose cost is no more than the cost of T. Blocker picks the query at the root of T' as his move in the game. This updates S to some $S' \subset S$. We may now move to the subtree T'' of T' corresponding to S'. In doing so, the cost of the tree has decreased by 1. By using this strategy, in each round, the cost of the tree decreases by at least 1. So in at most k rounds, the tree is reduced to a leaf at which point the game ends. □

For simplicity, we only gave the proof for relations $\mathcal{R} \subseteq M \times \mathcal{O}$, but it can easily be modified to show the equivalence for relations on M^n instead of just M.

B.2 Composition for randomized block decision trees

In this subsection, we adapt the proof of Ben-David, Blais, Göös and Maystre [BBGM22] to give a randomized composition theorem for Q-decision trees using linearized complexity. Then we prove $\overline{\mathsf{R}}^{Q}_{\epsilon}(\mathcal{R} \circ g) \geq \Omega\left(\overline{\mathsf{R}}^{\mathsf{dt}}_{\epsilon}(\mathcal{R})\sqrt{\mathsf{R}^{Q}(g)}\right) - n$ by considering a variant of conflict complexity [GLSS23].

B.2.1 Linearized Q-complexity

For a partial function $g : M \to \{0, 1, *\}$, the linearized Q-complexity of g is defined as

$$\mathsf{LR}^{Q}(g) = \inf_{\mathcal{T}} \max_{x} \frac{\mathrm{cost}(\mathcal{T}, x)}{\mathrm{bias}(\mathcal{T}, x)}$$

where $\mathcal{T}$ varies over randomized Q-decision trees and x varies over inputs in the domain of g.

Theorem B.4. For any relation $\mathcal{R} \subseteq \{0,1\}^n \times \mathcal{O}$, partial function $g : M \to \{0,1,*\}$, query set Q on M,

$$\overline{\mathsf{R}^Q_\epsilon}(\mathcal{R} \circ g) \geq \overline{\mathsf{R}^{\mathsf{dt}}_\epsilon}(\mathcal{R}) \frac{\mathsf{LR}^Q(g) - 4}{6}.$$

To prove this, we wish to use a minimax theorem for ratios [BB20, Theorem 2.18] to get that $\mathsf{LR}^Q(g) = \sup_\mu \min_T \frac{\mathrm{cost}(\mathcal{T},\mu)}{\mathrm{bias}(T,\mu)}$ where μ varies over distributions on M and T varies over deterministic Q-trees. However some conditions required to directly use the minimax theorem may not hold for the query set Q. Specifically, one of the required conditions is that for any tree, the cost should either be positive for all distributions or 0 for all distributions. This condition fails, for instance, when the query set corresponds to 1-depth or rank.

So we will instead work with an approximate version of LR^Q which is always positive. For any query set Q and real $\alpha > 0$, define $\mathrm{cost}_\alpha(\mathcal{T}, x) = \alpha + \mathrm{cost}(\mathcal{T}, x)$. Now define $\mathsf{LR}^Q_\alpha(g) = \inf_\mathcal{T} \max_x \frac{\mathrm{cost}_\alpha(\mathcal{T},x)}{\mathrm{bias}(\mathcal{T},x)}$. Note that we always have $\mathrm{cost}_\alpha(\mathcal{T}, x) \geq \alpha > 0$. In addition, since Q is closed, there is some decision tree which computes g exactly and has finite cost. So we may use the minimax theorem [BB20, Theorem 2.18] to show that

$$\mathsf{LR}^Q_\alpha(g) = \sup_\mu \min_T \frac{\mathrm{cost}_\alpha(T,\mu)}{\mathrm{bias}(T,\mu)}.$$

We rewrite the right hand side above in terms of the underlying distributions μ_0 and μ_1 over 0-inputs and 1-inputs of g respectively. The distribution μ must place equal weight on μ_0 and μ_1 for otherwise some decision tree which makes no queries solves g on this distribution with positive bias. Using this, we get

$$\mathsf{LR}^Q_\alpha(g) = \sup_{\mu_0,\mu_1} \min_T \frac{\mathrm{cost}_\alpha(T,\mu_0) + \mathrm{cost}_\alpha(T,\mu_1)}{2\mathrm{TV}(\mathrm{tran}(T,\mu_0), \mathrm{tran}(T,\mu_1))}$$

by reasoning exactly as in [BBGM22] (note that this does not depend at all on the query set). Here μ_0 and μ_1 vary over distributions on 0-inputs and 1-inputs of g respectively and T varies over Q-trees whose leaves are not labeled.

The next lemma will be used to write the above expression with $\min\{\mathrm{cost}_\alpha(T,\mu_0), \mathrm{cost}_\alpha(T,\mu_1)\}$ in the numerator instead of the sum $\mathrm{cost}_\alpha(T,\mu_0) + \mathrm{cost}_\alpha(T,\mu_1)$. Compared to the corresponding statement in [BBGM22, Lemma 7], we lose an additional additive constant because some query responses may have cost 0.

Lemma B.5 (essentially Lemma 7 in [BBGM22])**.** Let μ_0, μ_1 be distributions on M. Then

$$\min_T \frac{\max\{\mathrm{cost}_\alpha(T,\mu_0), \mathrm{cost}_\alpha(T,\mu_1)\}}{\mathrm{TV}(\mathrm{tran}(T,\mu_0), \mathrm{tran}(T,\mu_1))} \leq 6 \min_T \frac{\min\{\mathrm{cost}_\alpha(T,\mu_0), \mathrm{cost}_\alpha(T,\mu_1)\}}{\mathrm{TV}(\mathrm{tran}(T,\mu_0), \mathrm{tran}(T,\mu_1))} + 4.$$

Proof. Let T be a Q-tree achieving the minimum on the right hand side. Without loss of generality, assume that $\mathrm{cost}_\alpha(T,\mu_0) \leq \mathrm{cost}_\alpha(T,\mu_1)$. We will modify T to get a Q-tree T' whose cost is not much more than $\mathrm{cost}(T,\mu_0)$ and for which the total variation distance does not grow by much. T' is obtained by converting each marked node v in T' satisfying $\Pr_{\mu_1}[v] > 3\Pr_{\mu_0}[v]$ into a leaf. Here $\Pr_\mu[v]$ denotes the probability that an input drawn from μ reaches the node v. Note that $\mathrm{cost}(T',\mu_0) \leq \mathrm{cost}(T,\mu_0)$ since we have only truncated the tree.

To estimate $\mathrm{cost}(T',\mu_1)$, consider the following indicator random variables. For each marked node v in T', let X_v denote the indicator variable for reaching node v. Then $\mathrm{cost}(T',\mu_1) = \mathbb{E}[\sum_v X_v]$ where the sum is over only marked nodes in T'. We now group the nodes according to whether they were converted to leaves or not. Let V denote the leaves of T' and let C denote the set of nodes where a cutoff occurs. Then

$$\mathrm{cost}(T',\mu_1) = \sum_{v\notin C} \Pr_{\mu_1}[v] + \sum_{v\in C} \Pr_{\mu_1}[v] \leq 3 \sum_{v\notin C} \Pr_{\mu_0}[v] + \Pr_{\mu_1}[C] \leq 3\,\mathrm{cost}(T,\mu_0) + \Pr_{\mu_1}[C].$$

The first inequality follows from the fact that a cutoff does not occur only when $\Pr_{\mu_1}[v] \leq 3\Pr_{\mu_0}[v]$ and that since the nodes where a cutoff occurs are leaves in T', they form a partition of the event C. This implies

$$\mathrm{cost}_\alpha(T',\mu_1) \leq \alpha + 3\,\mathrm{cost}(T,\mu_0) + \Pr_{\mu_1}[C] \leq 3(\alpha + \mathrm{cost}(T,\mu_0)) + \Pr_{\mu_1}[C] \leq 3\,\mathrm{cost}_\alpha(T,\mu_0) + \Pr_{\mu_1}[C].$$

The calculation for the total variation distance stays the same as in [BBGM22].

$$\begin{aligned}
\mathrm{TV}(\mathrm{tran}(T,\mu_0),\mathrm{tran}(T,\mu_1)) &\leq \frac{1}{2}\sum_{v\in V\setminus C} |\Pr_{\mu_0}[v] - \Pr_{\mu_1}[v]| + \frac{1}{2}\sum_{v\in C}(\Pr_{\mu_0}[v] + \Pr_{\mu_1}[v]) \\
&\leq \frac{1}{2}\sum_{v\in V\setminus C} |\Pr_{\mu_0}[v] - \Pr_{\mu_1}[v]| + \frac{\Pr_{\mu_0}[C] + \Pr_{\mu_1}[C]}{2} \\
\mathrm{TV}(\mathrm{tran}(T',\mu_0),\mathrm{tran}(T',\mu_1)) &\geq \frac{1}{2}\sum_{v\in V\setminus C} |\Pr_{\mu_0}[v] - \Pr_{\mu_1}[v]| + \frac{1}{4}\sum_{v\in C}(\Pr_{\mu_0}[v] + \Pr_{\mu_1}[v]) \\
&\geq \frac{1}{2}\sum_{v\in V\setminus C} |\Pr_{\mu_0}[v] - \Pr_{\mu_1}[v]| + \frac{\Pr_{\mu_0}[C] + \Pr_{\mu_1}[C]}{4} \geq \frac{\Pr_{\mu_1}[C]}{4}
\end{aligned}$$

These imply the stated bound. □

Lemma B.5 implies that

$$\sup_{\mu_0,\mu_1}\min_T \frac{\min\{\mathrm{cost}_\alpha(T,\mu_0),\mathrm{cost}_\alpha(T,\mu_1)\}}{\mathrm{TV}(\mathrm{tran}(T,\mu_0),\mathrm{tran}(T,\mu_1))} \leq \mathsf{LR}^Q_\alpha(g) \leq 6\sup_{\mu_0,\mu_1}\min_T \frac{\min\{\mathrm{cost}_\alpha(T,\mu_0),\mathrm{cost}_\alpha(T,\mu_1)\}}{\mathrm{TV}(\mathrm{tran}(T,\mu_0),\mathrm{tran}(T,\mu_1))} + 4.$$

Let $\mathsf{dLR}^Q_\alpha(g)$ denote $\sup_{\mu_0,\mu_1}\min_T \frac{\min\{\mathrm{cost}_\alpha(T,\mu_0),\mathrm{cost}_\alpha(T,\mu_1)\}}{\mathrm{TV}(\mathrm{tran}(T,\mu_0),\mathrm{tran}(T,\mu_1))}$. The following lemma captures the main simulation underlying Theorem B.4.

Lemma B.6. For any relation $\mathcal{R} \subseteq \{0,1\}^n \times \mathcal{O}$, partial function $g : M \to \{0,1,*\}$ and real $\alpha > 0$,

$$\overline{\mathsf{R}^Q_\epsilon}(\mathcal{R}\circ g) \geq \overline{\mathsf{R}^{\mathsf{dt}}_\epsilon}(\mathcal{R})\ \mathsf{dLR}^Q_\alpha(g) - \alpha n.$$

To prove this, we will need the analogue of the decision tree simulator from [BBGM22]. A decision tree simulator is a randomized algorithm which is given two input distributions μ_0 and μ_1 on M, an unknown bit $b \in \{0,1\}$ and a stream of queries coming from some Q-tree T. The goal of the algorithm is to respond to these Q-queries according to the distribution μ_b while querying b with low probability. [BBGM22] show that when the inputs come from some set Σ^m and each query reveals a coordinate, this can be done while querying b with only probability $\mathrm{TV}(\mathrm{tran}(T,\mu_0),\mathrm{tran}(T,\mu_1))$ which is optimal.

Proposition B.7 ([BBGM22])**.** There exists a decision tree simulator that simulates the queries of a Q-tree T on μ_b while querying b with only probability $\mathrm{TV}(\mathrm{tran}(T,\mu_0),\mathrm{tran}(T,\mu_1))$.

Their simulator can be straightforwardly modified to get a simulator for Q-trees. We still describe it in detail since it will also be useful later.

Algorithm 1: Q-tree simulator

```
for all $S \subseteq M$ do
    $\mu_{\min}(S) \leftarrow \min\{\mu_0(S), \mu_1(S)\}$;
$S \leftarrow M$;
$b \leftarrow *$;
while more queries remain do
    Let $(S, (S_i)_{i \in [k]})$ be the next query ;        // Costs can be ignored for the simulation.
    $u \leftarrow \sum_{i \in [k]} \mu_{\min}(S_i)$;
    if $b = *$ then
        With probability $1 - u/\mu_{\min}(S)$, query the value of $b$;
    if $b = *$ then
        $S \leftarrow S_i$ for a random $i \in [k]$, where each $i$ is picked with probability $\mu_{\min}(S_i)/u$ ;
    else
        $S \leftarrow S_i$ for a random $i \in [k]$, where each $i$ is picked with probability $\frac{\mu_b(S_i) - \mu_{\min}(S_i)}{\mu_b(S) - u}$ ;
```

Proof. The simulator is described in Algorithm 1.

We now verify that for any node v in a Q-tree T, on running the simulator on T,

1. v is reached with probability $\mu_b(v)$, where we identify v with the subset of M reaching v (which is the same as the head of the query at v if v is an internal node)
2. v is reached without querying b with probability $\mu_{\min}(v)$.

We prove this by induction on the actual depth of v in T. The statements clearly hold at the root.

Suppose v has parent w. Let $(S, (S_i)_{i \in [k]})$ be the query at w and suppose S_j is the response corresponding to node v.

By induction, we know that w is reached without querying b with probability $\mu_{\min}(w) = \mu_{\min}(S)$. Conditioned on reaching w without querying b, the probability that b is queried when processing query q is $1 - u/\mu_{\min}(S)$ where $u = \sum_{i \in [k]} \mu_{\min}(S_i)$. Conditioned on not having queried b when processing the query, the probability that the simulator picks S_j is $\mu_{\min}(S_j)/u$. So the probability that we reach v without querying b is

$$\mu_{\min}(S) \cdot \frac{u}{\mu_{\min}(S)} \cdot \frac{\mu_{\min}(S_j)}{u} = \mu_{\min}(S_j).$$

The probability that we reach v after querying b (either before reaching w or when processing the query at w) is

$$(\mu_b(S) - \mu_{\min}(S)) \cdot \frac{\mu_b(S_j) - \mu_{\min}(S_j)}{\mu_b(S) - u} + \mu_{\min}(S) \cdot \frac{\mu_{\min}(S) - u}{\mu_{\min}(S)} \cdot \frac{\mu_b(S_j) - \mu_{\min}(S_j)}{\mu_b(S) - u} = \mu_b(S_j) - \mu_{\min}(S_j).$$

So the total probability of reaching node v is $\mu_{\min}(S_j) + (\mu_b(S_j) - \mu_{\min}(S_j)) = \mu_b(S_j)$ as desired.

The final step is to note that the probability that we query b during the simulation is $1 - \sum_{v \in L} \mu_{\min}(v) = \mathrm{TV}(\mathrm{tran}(T, \mu_0), \mathrm{tran}(T, \mu_1))$ where L is the set of leaves of T. □

We now prove Lemma B.6.

Proof of Lemma B.6. Let β be a real satisfying $0 < \beta < \alpha$. Let μ_0, μ_1 be the distributions such that for all trees $\mathcal{T}$, $\frac{\min\{\mathrm{cost}_\alpha(T,\mu_0),\mathrm{cost}_\alpha(T,\mu_1)\}}{\mathrm{TV}(\mathrm{tran}(T,\mu_0),\mathrm{tran}(T,\mu_1))} \geq \mathsf{dLR}^Q_\alpha(g) - \beta > 0$. Let $\mathcal{T}$ be a randomized Q-tree for $\mathcal{R} \circ g$ with error ϵ. We will construct a randomized decision tree $\mathcal{T}'$ of expected cost at most $\frac{\overline{\mathsf{R}^Q_\epsilon}(\mathcal{R}\circ g)+\alpha n}{\mathsf{dLR}^Q_\alpha(g)-\beta}$. On input $x \in \{0,1\}^n$, $\mathcal{T}'$ will essentially simulate $\mathcal{T}$ on the distribution μ^x obtained by sampling for each $i \in [n]$, block z_i according to μ_{x_i}. This will ensure correctness since for each input y in M^n, $\mathcal{T}'$ is incorrect on y with probability at most ϵ and therefore simulating $\mathcal{T}$ on any distribution which is supported only on inputs y such that $g^n(y) = x$ will only make an error with probability at most ϵ.

We now describe the simulation. It suffices to show how to simulate a deterministic Q-tree T. For each $i \in [n]$, we will run a separate instance of the decision tree simulator from Proposition B.7 corresponding to bit x_i. At a query (S, i, q) in T, we run the i^{th} simulator with query q. If the simulator queries its unknown bit b, we query x_i and use this to continue the simulation. Based on the response of the simulator, we move to the corresponding child in T. We continue doing this until we reach a leaf of T and give the same output.

We now estimate the cost on input x. Let p_i denote the probability that the i^{th} bit x_i is queried by $\mathcal{T}'$. Let c_i denote the expected cost of queries made in the i^{th} block when running T on $x \circ (\mu_0, \mu_1)$. To relate c_i and p_i, consider the randomized tree $\mathcal{T}_i$ which on input $z \in M$ runs T on the following distribution. The i^{th} block is fixed to z and for all $j \neq i$, the j^{th} block is sampled according to the distribution μ_{x_j}. Note that the only actual queries made by this tree to z correspond to queries in T made to the i^{th} block. Running the decision tree simulator on $\mathcal{T}_i$ with distributions μ_0 and μ_1 and the hidden bit being x_i corresponds exactly to the simulation above of T when restricted to the i^{th} block. Therefore p_i is the same as the probability that the decision tree simulator queries x_i. Also c_i is the expected cost of running $\mathcal{T}_i$ on μ_{x_i}.

We need to show that $c_i + \alpha \geq p_i(\mathsf{dLR}^Q_\alpha(g) - \beta)$. This would imply the desired bound since $\sum_{i=1}^n c_i \leq \overline{\mathsf{R}^Q_\epsilon}(\mathcal{R} \circ g)$ and $\sum_{i=1}^n p_i$ is the expected cost of $\mathcal{T}'$ on x.

Now $p_i = \mathbb{E}_{S\sim\mathcal{T}_i}[\mathrm{TV}(\mathrm{tran}(S,\mu_0), \mathrm{tran}(S,\mu_1))]$ by Proposition B.7 and $c_i = \mathbb{E}_{S\sim\mathcal{T}_i}[\mathrm{cost}(S,\mu_{x_i})]$. Since p_i and c_i come from the same probability distribution on Q-trees, it suffices to show for any S in the support of $\mathcal{T}_i$, $\mathrm{cost}(S,\mu_{x_i}) + \alpha \geq \mathrm{TV}(\mathrm{tran}(S,\mu_0), \mathrm{tran}(S,\mu_1)) \cdot (\mathsf{dLR}^Q_\alpha(g) - \beta)$. For this, observe that the left hand side is just $\mathrm{cost}_\alpha(S,\mu_{x_i})$ which we can bound from below by $\min\{\mathrm{cost}_\alpha(S,\mu_0), \mathrm{cost}_\alpha(S,\mu_1)\} \geq \mathrm{TV}(\mathrm{tran}(S,\mu_0), \mathrm{tran}(S,\mu_1)) \cdot (\mathsf{dLR}^Q_\alpha(g) - \beta)$. This gives $\overline{\mathsf{R}^Q_\epsilon}(\mathcal{R} \circ g) \geq \overline{\mathsf{R}^{\mathsf{dt}}_\epsilon}(\mathcal{R})(\mathsf{dLR}^Q_\alpha(g) - \beta) - \alpha n$.

Since this holds for all $\beta > 0$, we have $\overline{\mathsf{R}^Q_\epsilon}(\mathcal{R} \circ g) \geq \overline{\mathsf{R}^{\mathsf{dt}}_\epsilon}(\mathcal{R})\ \mathsf{dLR}^Q_\alpha(g) - \alpha n$. □

We now finish the proof of Theorem B.4.

Proof of Theorem B.4. By Lemma B.6, we have for every $\alpha > 0$,

$$\overline{\mathsf{R}^Q_\epsilon}(\mathcal{R} \circ g) \geq \overline{\mathsf{R}^{\mathsf{dt}}_\epsilon}(\mathcal{R})\ \mathsf{dLR}^Q_\alpha(g) - \alpha n.$$

By combining with Lemma B.5, we get

$$\overline{\mathsf{R}^Q_\epsilon}(\mathcal{R} \circ g) \geq \frac{1}{6}\overline{\mathsf{R}^{\mathsf{dt}}_\epsilon}(\mathcal{R})(\mathsf{LR}^Q_\alpha(g) - 4) - \alpha n.$$

We will now take the limit of the right hand side as $\alpha \to 0^+$. We will show that $\lim_{\alpha\to 0^+} \mathsf{LR}^Q_\alpha(g) = \mathsf{LR}^Q(g)$. First, note that we always have $\mathsf{LR}^Q(g) \leq \mathsf{LR}^Q_\alpha(g)$. We now show that $\mathsf{LR}^Q_\alpha(g)$ can be made arbitrarily close to $\mathsf{LR}^Q(g)$ by picking α to be small enough. Let $\epsilon > 0$. Let $\mathcal{T}$ be a randomized

Q-tree such that $\max_x \frac{\mathrm{cost}(\mathcal{T},x)+\alpha}{\mathrm{bias}(\mathcal{T},x)} \le \mathsf{LR}^Q(g)+\epsilon/2$. Then we have

$$\begin{aligned}\mathsf{LR}^Q_\alpha(g) \le \max_x \frac{\mathrm{cost}(\mathcal{T},x)+\alpha}{\mathrm{bias}(\mathcal{T},x)} &\le \max_x \frac{\mathrm{cost}(\mathcal{T},x)}{\mathrm{bias}(\mathcal{T},x)} + \max_x \frac{\alpha}{\mathrm{bias}(\mathcal{T},x)}\\ &\le \mathsf{LR}^Q(g) + \frac{\epsilon}{2} + \alpha \max_x \frac{1}{\mathrm{bias}(\mathcal{T},x)}.\end{aligned}$$

The last term in the sum can be made smaller than $\epsilon/2$ by choosing α appropriately. □

We next show that LR^Q gives an essentially inner-optimal composition theorem for randomized Q-trees.

Theorem B.8. Let P be a complexity measure for partial functions $g : M \to \{0,1,*\}$ satisfying $\mathsf{R}^Q(f \circ g) \ge \Omega(\mathsf{R}^{\mathsf{dt}}(f)P(g))$ for all partial functions $f : \{0,1\}^n \to \{0,1,*\}$, $g : M \to \{0,1,*\}$. Then for all $g : M \to \{0,1,*\}$, $\mathsf{LR}^Q(g) = \Omega(P(g))$.

Note that even though the above statement gives $\mathsf{LR}^Q(g) = \Omega(P(g))$, the composition theorem for LR^Q Theorem B.4 may fail to be meaningful for some g for which $P(g) > 0$ because of the additive constant loss. It would be interesting to know if this is necessary.

To prove Theorem B.8, again we will need to work with LR^Q_α ($\alpha > 0$). The following lemma shows that $\mathsf{LR}^Q_\alpha(g)$ can be used to upper bound the randomized Q-complexity of $\mathrm{ApproxIND}_k \circ g$ when k is large enough in terms of $m := |M|$ and α. Recall that since our query set is complete, there is always a deterministic Q-tree of cost at most m solving any g. The partial function $\mathrm{ApproxIND}_k : \{0,1\}^k \times \{0,1,2\}^{2^k} \to \{0,1,*\}$ is defined as follows. On input (x,y), we are promised that whenever $z \in \{0,1\}^k$ is at Hamming distance at most $k/2 - 2\sqrt{k \log k}$, we have $y_z = y_x \in \{0,1\}$ and for all other z, we have $y_z = 2$. By using 2 bits to represent each of $\{0,1,2\}$, we obtain a function with binary alphabet while the complexity stays the same up to a constant factor.

Lemma B.9 (Lemma 15 in [BBGM22])**.** For all $\alpha > 0$, all partial functions $g : M \to \{0,1,*\}$, all $k \in \mathbb{N}$ such that $\frac{k}{\log k} \ge \left(\frac{36(m+\alpha+1)}{\alpha}\right)^2$,

$$\mathsf{R}^Q(\mathrm{ApproxIND}_k \circ g) = O(\sqrt{k \log k} \cdot \mathsf{LR}^Q_\alpha(g))$$

where the implicit constant does not depend on α.

Proof. Let β be a real such that $0 < \beta < 1$. Let $\mathcal{T}$ be a randomized Q-tree for which on all inputs x, $\frac{\mathrm{cost}_\alpha(\mathcal{T},x)}{\mathrm{bias}(\mathcal{T},x)} \le \mathsf{LR}^Q_\alpha(g) + \beta$. We will give a randomized decision tree for $\mathrm{ApproxIND}_k \circ g$ which has worst-case cost $O(\sqrt{k \log k} \cdot (\mathsf{LR}^Q_\alpha(g) + \beta))$ on every input.

The tree $\mathcal{T}'$ for $\mathrm{ApproxIND}_k \circ g$ is obtained in the following way. For each $i \in [n]$, run $\mathcal{T}$ on x_i, the i^{th} block of the address part of the input (independently of the previous runs) and let X_i be the random variable for the α-cost (cost $+\alpha$) for block x_i. If $\sum_{j \le i} X_j$ has exceeded $36\sqrt{k \log k}(\mathsf{LR}^Q_\alpha(g) + \beta)$, stop. Since $\mathrm{cost}_\alpha(T,x) \ge \alpha$ for all T and x and $\mathsf{LR}^Q_\alpha(g) \le m + \alpha$, we necessarily have $\sum_{j \le k} X_j > 36\sqrt{k \log k}(\mathsf{LR}^Q_\alpha(g) + \beta)$. Let l be the index of the last block queried. For $i \in [l]$, let u_i denote the output given by (the current run of) T' on x_i. Let $z = u_1 u_2 \ldots u_l v_{l+1} v_{l+2} \ldots v_k$ where each v_i is a bit chosen uniformly and independently at random. The algorithm's output is obtained by solving the instance of g at index z.

We now estimate the cost of the algorithm. The total cost of all iterations except for the last is at most $36\sqrt{k \log k}(\mathsf{LR}^Q_\alpha(g) + \beta)$. The cost in the last iteration while computing the address is at most m. Solving the instance of g at index z also costs at most m. By the choice of k, we have $m = O(\sqrt{k \log k} \cdot \mathsf{LR}^Q_\alpha(g))$ and so the overall cost is also at most $O(\sqrt{k \log k} \cdot (\mathsf{LR}^Q_\alpha(g) + \beta))$.

Let us argue correctness now. Let Y_i be a Bernoulli random variable indicating whether $u_i = g(x_i)$. Here Y_i is correlated with X_i according to the decision tree T. By the definition of LR^Q_α, we have $\mathbb{E}[Y_i] \geq \frac{1}{2} + \frac{\mathbb{E}[\mathrm{cost}_\alpha(T,x_i)]}{2(\mathsf{LR}^Q_\alpha(g)+\beta)}$. Note that the number of correct output bits is exactly $S := \sum_{1\leq i\leq l} Y_i + \sum_{l<i\leq k} Z_i$ where Z_i is a uniform random bit.

We will show that $\Pr[S \geq \frac{k}{2} + 2\sqrt{k\log k}] \geq \frac{2}{3}$. Let l_0 be the smallest integer such that $\sum_{i=1}^{l_0} \mathbb{E}[X_i] > 6\sqrt{k\log k}(\mathsf{LR}^Q_\alpha(g) + \beta)$. Consider the random variable $S' = \sum_{1\leq i\leq \max\{l,l_0\}} Y_i + \sum_{\max\{l,l_0\}<i\leq k} Z_i$. Since $S \neq S'$ implies $l < l_0$, we have by Markov's inequality

$$\Pr[S \neq S'] \leq \Pr[\sum_{i=1}^{l_0-1} X_i > 36\sqrt{k\log k}\ (\mathsf{LR}^Q_\alpha(g)+\beta)] \leq \frac{\sum_{i=1}^{l_0-1}\mathbb{E}[X_i]}{36\sqrt{k\log k}(\mathsf{LR}^Q_\alpha(g)+\beta)} \leq \frac{1}{6}.$$

Also note that we can couple Z_i and Y_i for each i so that $Y_i \geq Z_i$ since $\mathbb{E}[Y_i] \geq 1/2$. This implies that $S' \geq \sum_{i=1}^{l_0} Y_i + \sum_{i=l_0+1}^{k} Z_i$. So it suffices to show that $\sum_{i=1}^{l_0} Y_i + \sum_{i=l_0+1}^{k} Z_i \geq \frac{k}{2} + 2\sqrt{\log k}$ with probability at least $1 - \frac{1}{9}$. By the choice of l_0, we have

$$\sum_{i=0}^{l_0}\mathbb{E}[Y_i] \geq \sum_{i=0}^{l_0}\left(\frac{1}{2} + \frac{\mathbb{E}[\mathrm{cost}_\alpha(T,x_i)]}{2(\mathsf{LR}^Q_\alpha(g)+\beta)}\right) > \frac{l_0}{2} + 3\sqrt{k\log k}.$$

So by the Chernoff bound, we have $\sum_{i=1}^{l_0} Y_i + \sum_{i=l_0+1}^{k} Z_i \geq \frac{k}{2} + 2\sqrt{k\log k}$ except with probability at most $1/9$.

So with probability at least $2/3$, z agrees with $g^k(x)$ on at least $k/2 + 2\sqrt{k\log k}$ bits. The last step of the algorithm incurs no error. □

Proof of Theorem B.8. By the assumption on P, we have for any k, $\mathsf{R}^Q(\mathrm{ApproxIND}_k \circ g) \geq \Omega(\sqrt{k\log k}P(g))$. By Lemma B.9, for any $\alpha > 0$, if k is large enough, we have $\mathsf{R}^Q(\mathrm{ApproxIND}_k \circ g) \leq O(\sqrt{k\log k}\mathsf{LR}^Q_\alpha(g))$. Combining these gives $\mathsf{LR}^Q_\alpha(g) = \Omega(P(g))$. Since this holds for all $\alpha > 0$, we get $\mathsf{LR}^Q(g) = \lim_{\alpha\to 0^+}\mathsf{LR}^Q_\alpha(g) = \Omega(P(g))$. □

B.2.2 $\hat{\chi}^Q$, a variant of conflict complexity

We will now prove $\overline{\mathsf{R}^Q_\epsilon}(\mathcal{R}\circ g) \geq \Omega(\overline{\mathsf{R}^{\mathsf{dt}}_\epsilon}\sqrt{\mathsf{R}^Q(g)}) - n$. To do this, we consider a complexity measure $\hat{\chi}^Q$, which is a variant of conflict complexity [GLSS23]. The proofs of the composition theorem using $\hat{\chi}^Q$ and the bound $\mathsf{R}^Q(g) \leq O(\hat{\chi}^Q(g)^2)$ closely follow the proofs involving conflict complexity. For readers familiar with conflict complexity, we start by explaining the differences.

The natural Q-analogue of conflict complexity would use the following generalization of the simulation in [GLSS23]. Suppose we have a deterministic Q-tree T on M and wish to simulate the action of some distribution μ_b where μ_0 and μ_1 are some distributions on M and b is some unknown bit. We wish to simulate T without querying b before having seen many marked edges in T during the simulation (in expectation). At a node v in T with children $v_1, v_2, \ldots, v_k$, we move to the child v_i with probability $\min\{\Pr_{x\sim\mu_0}[x \in v_i | x \in v], \Pr_{x\sim\mu_1}[x \in v_i | x \in v]\}$. With the remaining probability, $1 - \sum_{i\in[k]}\min\{\Pr_{x\sim\mu_0}[x \in v_i|x \in v], \Pr_{x\sim\mu_1}[x \in v_i|x \in v]\}$, we query b and move to v_i with probability $\Pr_{x\sim\mu_b}[x \in v_i|x \in v]$. Note that $1 - \sum_{i\in[k]}\min\{\Pr_{x\sim\mu_0}[x \in v_i|x \in v], \Pr_{x\sim\mu_1}[x \in v_i|x \in v]\}$ is just the total variation distance between the distributions induced on the children of v by the distributions $\mu_0|v$ and $\mu_1|v$. We may now define Q-conflict complexity of two distributions μ_0, μ_1 to be the minimum number of marked edges seen by the above simulation before stopping to query b for some tree T separating μ_0 and μ_1. Using this simulation, one can prove a composition theorem for this measure, say Q-conflict complexity χ^Q, $\overline{\mathsf{R}^Q_\epsilon}(\mathcal{R}\circ g) \geq \overline{\mathsf{R}^{\mathsf{dt}}_\epsilon}\chi^Q(g)$.

It is not very clear, however, how to modify the proof in [GLSS23] to show that $\mathsf{R}^{\mathsf{dt}}(g) \leq O(\chi^Q(g)^2+1)$. As a first attempt, since we only include marked edges in the cost, it seems natural to only keep track of marked nodes instead of all nodes as in [GLSS23, Claim 6.2]. Informally, we may think of compressing any unmarked edges which don't end at leaves. We may now try to lower bound for any marked node v, the mutual information between $g(x)$ and the next marked node reached after reaching v. This can be shown to be related to the total variation distance between the distributions induced on the children of v in the compressed tree by $\mu_0|v$ and $\mu_1|v$. We would now like to show that this quantity is lower bounded by the probability that the random process queries b after it reaches v and before crossing any more marked edges. The issue now is that this probability can be larger than the total variation distance described above since the [GLSS23] simulation is only locally optimal rather than over the whole tree. We get around this by using the optimal decision tree simulator [BBGM22] to simulate the moves from one marked node to another. Since the probability that the simulator queries b is exactly equal to the TV distance of the involved distributions, this lets us finish the proof. Of course, since we changed the simulation, we also need to modify the composition theorem. Here we are only able to show $\overline{\mathsf{R}_\epsilon^Q}(\mathcal{R}\circ g) \geq \overline{\mathsf{R}_\epsilon^{\mathsf{dt}}}(\mathcal{R})\hat{\chi}^Q(g) - n$ instead of $\overline{\mathsf{R}_\epsilon^Q}(\mathcal{R}\circ g) \geq \overline{\mathsf{R}_\epsilon^{\mathsf{dt}}}(\mathcal{R})\hat{\chi}^Q(g) - O(\overline{\mathsf{R}_\epsilon^{\mathsf{dt}}}(\mathcal{R}))$.

We now define $\hat{\chi}^Q$ formally. We will use notation similar to [GLSS23]. Let T be a Q-tree. For a marked node v in T, we use $\pi(v)$ to denote the unique closest marked node on the path from the root to v. (Note that $\pi(v)$ is often not the actual parent of v.) By convention, we think of the root as being marked. For a node v, we use $d_T(v)$ to indicate the number of marked edges on the path from the root to v.

Let $g : M \to \{0,1,*\}$ be a partial function. Fix probability distributions μ_0, μ_1 over 0-inputs and 1-inputs of g respectively. We use $\mu_0|v$ to denote the distribution μ_0 after conditioning on reaching the vertex v and similarly use $\mu_1|v$. Let v be a marked node. We use T_v to denote the subtree of T rooted at v where every v to leaf path in T is truncated at the first marked node after v. T_v includes any leaf l of T such that the path from v to l does not include any marked edges. For $b \in \{0,1\}$, we use $\sigma_b(T,v)$ to denote the distribution on the leaves of T_v induced by the distribution $\mu_b|v$. Note that σ_b depends on μ_b but we suppress this dependence for ease of notation.

Let T be a Q-tree computing g. Define for each marked node v in T,

$$R(v) = \begin{cases} 1 & \text{if } v \text{ is the root,} \\ R(\pi(v)) \cdot \min\{\Pr_{\mu_0|\pi(v)}[x \in v], \Pr_{\mu_1|\pi(v)}[x \in v]\} & \text{otherwise.} \end{cases}$$

Also define $\Delta(v) = \mathrm{TV}(\sigma_0(T,v), \sigma_1(T,v))$. Similar to the simulation in [GLSS23], these quantities are related to a random walk done on the tree T. The difference here is that we always stay at marked nodes. (Strictly speaking, this means that the random walk is not really a walk on the tree, but we will continue calling it so for simplicity.) At a marked node v, we move to an immediate marked descendant w with probability $\min\{\Pr_{\mu_0|v}[x \in w], \Pr_{\mu_1|v}[x \in w]\}$. With the remaining probability $\Delta(v) = \mathrm{TV}(\sigma_0(T,v), \sigma_1(T,v))$, we stop the random walk at v.

Define $\hat{\chi}^Q(T,(\mu_0,\mu_1)) = (\sum_v d_T(v)\Delta(v)R(v))+1$ where the sum is only over marked nodes in T. (The $+1$ term is not necessary but will be convenient later for the upper bound $\mathsf{R}^Q(g) \leq O(\hat{\chi}^Q(g)^2)$.) Define

$$\hat{\chi}^Q(g) = \max_{\mu_0,\mu_1} \min_T \hat{\chi}^Q(T,(\mu_0,\mu_1)).$$

where μ_0 and μ_1 vary over distributions on the 0-inputs and the 1-inputs of g respectively, T varies over deterministic Q-trees solving g.

Before we describe the modified query process, we explain how to convert the above random walk which jumps from marked node to marked node into an actual random walk on the tree.

To do this, we will use the online decision tree simulator of [BBGM22] (Algorithm 1) to simulate one big step of the above random walk. In more detail, at any marked node v, we will initialize the simulator with the distributions $\mu_0|v$ and $\mu_1|v$. We simulate the following queries using the simulator until we reach a marked node or the simulator queries the bit b. In the latter case, we stop the random walk and in the former case, we reset the distributions in the simulator according to the new marked node. Note that this does indeed give the right distribution.

The simulation using $\hat{\chi}^Q(g)$ is described in Algorithm 2. We now informally describe what it is doing. In lines 1-7, we set up for each block some variables required for the simulation. These variables roughly are the ones required by the decision tree simulator of Proposition B.7. Here (μ_0^i, μ_1^i) indicate the distributions being currently simulated in block i. The counter cnt_i indicates the number of marked edges in block i crossed before making a query to x_i. $S = S_1 \times S_2 \times \cdots \times S_n$ indicates the set of inputs reaching the current node.

Lines 11 to 24 are essentially the steps from the decision tree simulator (Algorithm 1) applied to block j. If the cost of the current response is 1 (line 25), then we end the current run of the simulator. If x_j was queried during the current run (indicated by $\mathrm{query}_j^{\mathrm{curr}} = 1$) , we set $\mathrm{query}^{\mathrm{prev}} \leftarrow 1$ (line 27) so that in the future we can naively simulate the query. Otherwise, we reset the distributions of the simulator for block i to the conditional distributions at the new vertex (line 30). Lines 32-34 perform the naive simulation where we directly move to the appropriate child according to the conditional distribution at the current node.

We next verify that the simulation reaches each node v of T with the correct probability.

Lemma B.10. Let v be a node of T. Let $x \in \{0,1\}^n$. Let $A(v)$ denote the event that Algorithm 2 reaches v during the simulation on input x and distributions (μ_0, μ_1). Let $B(v)$ denote the event that we reach v when running T on a random input from the distribution μ^x. Then $\Pr[A(v)] = \Pr[B(v)]$.

Proof. Let $S = S_1 \times S_2 \times \cdots \times S_n$ be the set of inputs reaching v. Since $\mu^x = \prod_{i \in [n]} \mu_{x_i}$ is a product distribution, $\Pr[B(v)] = \prod_{i\in[n]} \Pr_{z\sim\mu_{x_i}}[z \in S_i]$.

Let $v_0, v_1, \ldots, v_k = v$ be the nodes on the root to v path in T. We may write $\Pr[A(v)] = \prod_{i\in[k]} \Pr[A(v_i)|A(v_{i-1})]$. In any iteration, the probabilities used by the simulation only depend on the block being queried (and is independent of the rest) and the set of consistent inputs also changes only in that block. This means that we can group terms in the product $\prod_{i\in[k]} \Pr[A(v_i)|A(v_{i-1})]$ according to which block is being queried. For each block $i \in [n]$, if we can show that the corresponding product of terms is $\Pr_{z\sim\mu_{x_i}}[z \in S_i]$, we will be done.

So fix any $i \in [n]$. Suppose the queries into block i (ignoring the costs) to reach the node v are

$$(M, (U_l^1)_{l\in[m_1]}), (U_1^1, (U_l^2)_{l\in[m_2]}), \ldots, (U_1^{s-1}, (U_l^s)_{l\in[m_s]})$$

where s is the total number of queries into block i before reaching node v, and we assume, without loss of generality, that the response to query $(U_1^{j-1}, (U_l^j)_{l\in m_j})$ which is consistent with S_i is U_1^j. In particular, $U_1^s = S_i$.

We will show by induction on the number of cost-1 responses in block i, that the probability of getting to the set S_i is exactly $\Pr_{z\sim\mu_{x_i}}[z \in S_i]$. This is clearly true if v is the root.

Now suppose $U_1^{s'}$ is the last cost-1 response among $M, U_1^1, U_1^2, \ldots, U_1^{s-1}$. Conditioned on having queried x_i before reaching $U_1^{s'}$, the probability that we reach $U_1^s = S_i$ during the simulation is

$$\prod_{j=s'+1}^{s} \Pr_{z\sim\mu_{x_i}}[z \in U_1^j | z \in U_1^{j-1}] = \Pr_{z\sim\mu_{x_i}}[z \in U_1^s | z \in U_1^{s'}].$$

Algorithm 2: $\hat{\chi}^Q$ simulator for a Q-tree T on M^n

```
for $i \in [n]$ do
    query$_i^{\text{prev}} \leftarrow 0$ ; // Indicates if $x_i$ was queried before the last cost 1 query response in block $i$
    query$_i^{\text{curr}} \leftarrow 0$ ; // Indicates if $x_i$ was queried after the last cost 1 query response in block $i$
    $x_i \leftarrow *$;
    $(\mu_0^i, \mu_1^i) \leftarrow (\mu_0, \mu_1)$ ; // Current distributions to be simulated
    cnt$_i \leftarrow 0$ ; // Counts cost 1 responses in block $i$
    $S_i \leftarrow M$ ; // Current set of inputs in block $i$
// $S$ denotes $S_1 \times S_2 \times \cdots \times S_n$
$v \leftarrow$ root of $T$ ; // Current node in $T$
while $v$ is not a leaf of $T$ do
    Let $(S, j, q)$ be the query at $v$ where $q = (S_j, (U_i, c_i)_{i \in [k]})$ ;
    if query$_j^{\text{prev}} = 0$ then
        $u \leftarrow \sum_{i \in [k]} \min\{\mu_0^j(U_i), \mu_1^j(U_i)\}$;
        $p_{\min} \leftarrow \min\{\mu_0^j(S_j), \mu_1^j(S_j)\}$;
        if query$_j^{\text{curr}} = 0$ then
            $b \leftarrow$ Bernoulli$(1 - u/p_{\min})$;
            if $b = 1$ then
                Query $x_i$;
                query$_j^{\text{curr}} = 1$;
        if query$_j^{\text{curr}} = 0$ then
            Pick $i \in [k]$ at random where $i$ has probability $\min\{\mu_0^j(U_i), \mu_1^j(U_i)\}/u$;
        else
            Pick $i \in [k]$ at random where $i$ has probability $\frac{\mu_{x_i}^j(U_i) - \min\{\mu_0^j(U_i), \mu_1^j(U_i)\}}{p_{\min} - u}$;
        $S_j \leftarrow U_i$;
        $v \leftarrow v_i$ where $v_i$ is the child of $v$ corresponding to response $U_i$;
        if $c_i = 1$ then
            if query$_j^{\text{curr}} = 1$ then
                query$_j^{\text{prev}} \leftarrow 1$;
            else
                cnt$_j \leftarrow$ cnt$_j + 1$;
                $(\mu_0^j, \mu_1^j) \leftarrow (\mu_0|S_j, \mu_1|S_j)$ ; // Update distributions to current conditional distributions
    else
        Pick $i \in [k]$ randomly, where $i$ is picked with probability $\Pr_{z \sim \mu_{x_i}}[z \in U_i | z \in S^j]$ ;
        $S_j \leftarrow U_i$;
        $v \leftarrow v_i$ where $v_i$ is the child of $v$ corresponding to response $U_i$;
```

On the other hand, if we reach $U_1^{s'}$ without having queried x_i earlier, we run the decision tree simulator with distributions $(\mu_0|U_1^{s'}, \mu_1|U_1^{s'})$. This implies that conditioned on reaching $U_1^{s'}$ without having queried x_i, the probability that we reach U_1^s is $\Pr_{z\sim\mu_{x_i}|U_1^{s'}}[z \in U_1^s] = \Pr_{z\sim\mu_{x_i}}[z \in U_1^s | z \in U_1^{s'}]$.

By induction, we know that the total probability with which we reach $U_1^{s'}$ during the simulation is $\Pr_{z\sim\mu_{x_i}}[z \in U_!^{s'}]$. So we have

$$\begin{aligned}
&\Pr[U_1^s \text{ reached during simulation}] \\
&= \Pr[U_1^s \text{ reached during simulation} | U_1^{s'} \text{ reached without querying } x_i] \cdot \Pr[U_1^{s'} \text{ reached without querying } x_i] \\
&\quad + \Pr[U_1^s \text{ reached during simulation} | U_1^{s'} \text{ reached after querying } x_i] \cdot \Pr[U_1^{s'} \text{ reached after querying } x_i] \\
&= \Pr_{z\sim\mu_{x_i}}[z \in U_1^s | z \in U_1^{s'}] (\Pr[U_1^{s'} \text{ reached without querying } x_i] + \Pr[U_1^{s'} \text{ reached after querying } x_i]) \\
&= \Pr_{z\sim\mu_{x_i}}[z \in U_1^s | z \in U_1^{s'}] \cdot \Pr[U_1^{s'} \text{ reached}] \\
&= \Pr_{z\sim\mu_{x_i}}[z \in U_1^s | z \in U_1^{s'}] \cdot \Pr_{z\sim\mu_{x_i}}[z \in U_1^{s'}] = \Pr_{z\sim\mu_{x_i}}[z \in U_1^s]. \qquad \square
\end{aligned}$$

For $i \in [n]$, define $D_{x,i}(T, (\mu_0, \mu_1)) = \mathbb{E}[\text{cnt}_i]$, the expected number of marked queries made into block i before the simulator (Algorithm 2) queries x_i when run on input x with distributions (μ_0, μ_1). Define $D_x(T, (\mu_0, \mu_1)) = \sum_{i\in[n]} D_x(T, (\mu_0, \mu_1))$. Note that if $n = 1$, for any $x \in \{0, 1\}$, $D_x(T, (\mu_0, \mu_1)) = \hat{\chi}^Q(T, (\mu_0, \mu_1)) - 1$.

An analogue of the direct sum theorem, [GLSS23, Theorem 5.2] can be proved in the same way.

Proposition B.11. *Let μ_0 and μ_1 be distributions on $g^{-1}(0)$ and $g^{-1}(1)$ respectively. Let T be a Q-tree on M^n which solves g on each block. Then for any $x \in \{0, 1\}^n$,*

$$D_x(T, (\mu_0, \mu_1)) \geq n(\min_S \hat{\chi}^Q(S, (\mu_0, \mu_1)) - 1)$$

where S varies over Q-trees on M solving g.

The -1 in the lower bound accounts for the $+1$ added separately in the definition of $\hat{\chi}^Q$.

Using the above direct sum result, we get a composition theorem with $\hat{\chi}^Q$ by following the proof of [GLSS23, Theorem 5.1].

Theorem B.12. *For any relation $\mathcal{R} \subseteq M^n \times \mathcal{O}$ and partial function $g : M \to \{0, 1, *\}$,*

$$\overline{\mathsf{R}_\epsilon^Q}(\mathcal{R} \circ g) \geq \overline{\mathsf{R}_\epsilon^{\mathsf{dt}}}(\mathcal{R}) \hat{\chi}^Q(g) - n.$$

Proof. Let μ_0, μ_1 be the distributions on 0-inputs and 1-inputs of g achieving the maximum in the definition of $\hat{\chi}^Q(g)$, so that for all deterministic Q-trees T solving g, $\hat{\chi}^Q(g) \leq \hat{\chi}^Q(T, (\mu_0, \mu_1))$. Let $\mathcal{T}$ be a randomized Q-tree computing $\mathcal{R} \circ g$ to error ϵ. We will give a randomized decision tree $\mathcal{T}'$ computing $\mathcal{R}$ to error ϵ. $\mathcal{T}'$ is defined by picking sampling a Q-tree T according to $\mathcal{T}$ and then running Algorithm 2 on T. Finally, when a leaf is reached in the simulation, the same output is given in $\mathcal{T}'$.

By Lemma B.10, during the simulation each leaf is reached with the same probability as an input $z \sim \mu^x$, where x is the input to $\mathcal{T}'$. This implies that $\Pr[\mathcal{T}'(x) \notin \mathcal{R}(x)] = \Pr_{z\sim\mu^x}[\mathcal{T}(z) \notin (\mathcal{R} \circ g)(z)] = \mathbb{E}_z[\Pr[\mathcal{T}(z) \notin (\mathcal{R} \circ g)(z)]] \leq \epsilon$. This shows correctness of $\mathcal{T}'$.

We now bound the number of queries made by $\mathcal{T}'$ on any input. We will modify $\mathcal{T}$ to get a randomized Q-tree $\mathcal{W}$ which always computes g correctly (no error) on each block and for which we can relate the total number of marked queries in $\mathcal{W}$ with the expected cost of $\mathcal{T}$ and $\mathcal{T}'$. The

Algorithm 3: $\mathcal{W}$

```
1 Run $\mathcal{T}$ on $z \in M^n$.
2 Suppose $\mathcal{T}$ terminates at leaf with inputs $S = S_1 \times S_2 \times \ldots S_n$.
  // $S_i$ always refers to the current set of possible inputs in block $i$
3 for $i \in [n]$ do
4     while the last response obtained in block $i$ had cost 0 and $g$ is not constant on $S_i$ do
          // If the last query made by $\mathcal{T}$ into block $i$ has cost 1, this loop is
             not entered
5         Make any query $q$ into block $i$.
6     if $\mu_0(S_i) > 0$ and $\mu_1(S_i) > 0$ then
7         Run $H$ on $z_i$ where $H$ is a $Q$-tree on $M$ satisfying $\hat{\chi}^Q(H, (\mu_0|S_i, \mu_1|S_i)) \leq \hat{\chi}^Q(g)$.
8     else
          // The queries made in this case do not affect the cost of the
             simulation
9         Run any $Q$-tree on $z_i$ which solves $g$ restricted to the set $S_i$.
          // Possibly $g$ is constant on $S_i$ in which case no new queries are made.
```

tree $\mathcal{W}$ is described in the Algorithm 3. Inside the while loop on lines 4-5, we keep making queries until we get a cost 1 response. (Possibly the response on the last query into block i made by $\mathcal{T}$ already had cost 1, in which case this while loop is skipped.) In line 6, since $\mu_0(S_i) > 0$ and $\mu_1(S_i) > 0$, the distributions $\mu_0|S_i$ and $\mu_1|S_i$ are well-defined and so there exists a tree H satisfying $\hat{\chi}^Q(H, (\mu_0|S_i, \mu_1|S_i)) \leq \hat{\chi}^Q(g)$ by the definition of $\hat{\chi}^Q(g)$. Lines 8-9 are not crucial to the argument, but are only included because the definition of $\hat{\chi}^Q$ requires g to be computed correctly on all inputs (even those not in the support of either distribution). In lines 7 and 9, we use Lemma B.1 to find a Q-tree whose root has inputs S_i.

We now bound $D_x(\mathcal{W}, (\mu_0, \mu_1)) := \mathbb{E}_{W \sim \mathcal{W}}[D_x(W, (\mu_0, \mu_1))]$ in terms of $\hat{\chi}^Q(g)$, $\text{cost}(\mathcal{T}', x)$ and the maximum cost of running $\mathcal{T}$ on any input. By averaging, there is a deterministic Q-tree W in the support of $\mathcal{W}$ for which $D_x(W, (\mu_0, \mu_1)) \leq D_x(\mathcal{W}, (\mu_0, \mu_1))$. By Proposition B.11, $D_x(W, (\mu_0, \mu_1)) \geq n(\hat{\chi}^Q(g) - 1)$ where we used that μ_0, μ_1 are distributions achieving the maximum in the definition of $\hat{\chi}^Q(g)$.

We will now show that $D_x(\mathcal{W}, (\mu_0, \mu_1)) \leq d + (n - \text{cost}(\mathcal{T}', x))\hat{\chi}^Q(g)$ where $d = \max_z \text{cost}(\mathcal{T}, z)$. For $i \in [n]$, let E_i denote the event that x_i is queried when running $\mathcal{T}'$ on x. Let X_i be the random variable counting the number of cost 1 responses seen in block i when running Algorithm 2 on $\mathcal{T}$. (So $D_{x,i}(\mathcal{T}, (\mu_0, \mu_1)) = \mathbb{E}_{T \sim \mathcal{T}}[D_{x,i}(T, (\mu_0, \mu_1))] = \mathbb{E}[X_i]$.) This corresponds to the contribution to $D_x(\mathcal{W}, (\mu_0, \mu_1))$ coming from line 1 of Algorithm 3. Let Y_i be the random variable counting the number of cost 1 responses seen when running line 7 of Algorithm 3 in the iteration for block i before x_i is queried. (So if x_i had already been queried before reaching line 7, then $Y_i = 0$.) Let I_i be the indicator random variable for seeing a cost 1 response in line 5 of Algorithm 3 during the iteration for block i before having queried x_i.

We may now write

$$D_x(\mathcal{W}, (\mu_0, \mu_1)) = \mathbb{E}[\sum_{i \in [n]} (X_i + I_i + Y_i)] = \mathbb{E}[\sum_{i \in n} X_i] + \sum_{i \in [n]} \mathbb{E}[I_i] + \sum_{i \in [n]} \mathbb{E}[Y_i].$$

We will bound each of these terms separately. To see $\mathbb{E}[\sum_{i \in n} X_i] \leq d$, we will use that $\sum_{i \in [n]} X_i$ is at most the total number of marked edges seen on the path from the root to leaf of $\mathcal{T}$. By Lemma

B.10, we know that each leaf of $\mathcal{T}$ is reached with the same probability as a random $z \sim \mu^x$ would reach it. This implies that $\mathbb{E}[\sum_{i\in[n]} X_i] \leq \mathbb{E}_{z\sim\mu^x}[\mathrm{cost}(\mathcal{T}, z)] \leq d$.

Note that $I_i = 1$ implies that x_i has not been queried in $\mathcal{T}'$ (E_i does not occur) and similarly $Y_i > 0$ implies E_i does not occur. So $\mathbb{E}[I_i] \leq \Pr[\neg E_i] = 1 - \Pr[E_i]$. To bound $\mathbb{E}[Y_i]$, we first condition on reaching a node v in $\mathcal{W}$ corresponding to line 7 in the iteration for block i, and not having queried x_i earlier. Let us denote this event by F_v. By the choice of H in line 7, we know that $\mathbb{E}[Y_i|F_v] \leq \hat{\chi}^Q(g) - 1$. This gives

$$\mathbb{E}[Y_i] = \sum_v \Pr[F_v] \cdot \mathbb{E}[Y_i|F_v] \leq \sum_v \Pr[F_v](\hat{\chi}^Q(g) - 1) \leq (1 - \Pr[E_i])(\hat{\chi}^Q(g) - 1),$$

where v varies over the nodes corresponding to line 7 in Algorithm 2 for block i. Combining these, we have $D_x(\mathcal{W}, (\mu_0, \mu_1)) \leq d + \sum_{i\in[n]}(1 - \Pr[E_i])\hat{\chi}^Q(g) = d + (n - \mathrm{cost}(\mathcal{T}', x))\hat{\chi}^Q(g)$ as desired.

By using the two bounds on $D_x(\mathcal{W}, (\mu_0, \mu_1))$, we get

$$n(\hat{\chi}^Q(g) - 1) \leq d + (n - \mathrm{cost}(\mathcal{T}', x))\hat{\chi}^Q(g)$$

which on rearranging gives $d \geq \mathrm{cost}(\mathcal{T}', x)\hat{\chi}^Q(g) - n$. □

We next show that $\mathsf{R}^{\mathsf{dt}}_\epsilon(g) \leq O(\hat{\chi}^Q(g)^2)$ for some fixed constant $\epsilon < 1/2$ by describing the changes required to the proof of [GLSS23, Theorem 6.1]. By Lemma B.1, we can still amplify the success probability by repeating a randomized decision tree a constant number of times.

Theorem B.13. There exists a constant $\epsilon < 1/2$ such that for all $g : M \to \{0, 1, *\}$, $\mathsf{R}^Q_\epsilon(g) \leq O(\hat{\chi}^Q(g)^2)$.

Proof sketch. Let $d := \lceil \hat{\chi}^Q(g) \rceil$. Since $\hat{\chi}^Q(g) \geq 1$, we have $d \leq 2\hat{\chi}^Q(g)$. We will show that for each distribution μ supported only on $g^{-1}(0) \cup g^{-1}(1)$, there exists a deterministic Q-tree whose worst case cost is bounded by $O(d^2)$ and which makes an error with probability at most ϵ on the distribution μ for some fixed $\epsilon < 1/2$. Define $\mu_0 = \mu|g^{-1}(0)$ and $\mu_1 = \mu|g^{-1}(1)$. Consider a Q-tree T' which achieves the minimum in the definition of $\hat{\chi}^Q(g)$ on the distributions μ_0, μ_1.

Obtain the tree T from T' by making any marked node whose marked depth is $D := 10d^2$ a leaf. A leaf v in T is labeled by $b \in \{0, 1\}$ such that $\Pr_{x\sim\mu}[x \in v \text{ and } g(x) = b] \geq \Pr_{x\sim\mu}[x \in v \text{ and } g(x) = 1 - b]$.

Since we have already ensured that $\mathrm{cost}(T, x)$ on any $x \in M$ is at most $10d^2$, we only need to verify correctness. Let E be the event that x does not reach a marked node v where $\Pr_{x\sim\mu}[g(x) = 0|x \in v] \leq 1/3$ or $\Pr_{x\sim\mu}[g(x) = 1|x \in v] \leq 1/3$.

The case where E happens with not too large probability ($\Pr[E] < 3/4$) can be handled in exactly the same way as in [GLSS23]. Once a vertex v is reached such that $\Pr[g(x) = 0 \mid x \in v] \leq 1/3$ or $\Pr[g(x) = 1 \mid x \in v] \leq 1/3$, the contribution to the error of the inputs that reach v cannot be more than $1/3$. This follows from an application of Jensen's inequality and the triangle inequality.

We now explain the changes required for the other case $\Pr[E] \geq 3/4$. In this case, we will bound the mutual information between the transcript of T on x and $g(x)$. Since the transcript of T is determined by the leaf v reached, we will treat the two interchangeably. Note that since T' solves g, if E occurs, a leaf is not reached. For $i \in \{0\} \cup [10d^2]$, let v_i denote the i^{th} marked node reached on an input $x \sim \mu$. Recall that the root is considered a marked node and so v_0 is always the root of T. It will also be convenient to count all leaves as marked nodes. So if x reaches a leaf before crossing i marked edges, then v_i is the leaf reached by x. For $i \in [D]$, let S_i denote the set of all inputs that reach v_i (so if v_i is an internal node, S_i is the head of the query at v_i). We will show that the mutual information $I(S_1, S_2, \ldots, S_D; g(x))$ is at least some positive constant under the assumption $\Pr[E] \geq 3/4$.

We need the following claim which relates the information between S_i and $g(x)$ after having reached v in terms of $\Delta(v)$ following [GLSS23, Claim 6.4].

Claim B.14. Let v be a vertex in T which lies in the support of v_{i-1} and is not a leaf. Then

$$I(S_i; g(x) \mid x \in v) \geq 8(\Pr_{x\sim\mu}[g(x) = 0 \mid x \in v] \cdot \Pr_{x\sim\mu}[g(x) = 1 \mid x \in v] \cdot \Delta(v))^2.$$

Proof of claim. Let $s_1, s_2, \ldots, s_k \subset M$ be all the sets in the support of S_i after conditioning on $x \in v$. Note that these actually form a partition of the set of inputs reaching v, corresponding to the immediate marked descendants of v. Since in all quantities being considered, we condition on v, we suppress this conditioning in the rest of the proof.

For two random variables X and Y, we use $X \times Y$ to denote their independent coupling. By Pinsker's inequality,

$$I(S_i; g(x)) = D_{\mathrm{KL}}((S_i, g(x))||S_i \times g(x)) \geq 2\mathrm{TV}((S_i, g(x)), S_i \times g(x))^2.$$

We now show that the right hand side can be rewritten as the quantity we want. Fix any $b \in \{0, 1\}$, $j \in [k]$. We have

$$\Pr[(g(x), S_i) = (b, s_j)] = \Pr[g(x) = b] \Pr[S_i = s_j \mid g(x) = b].$$

Also

$$\begin{aligned}\Pr[(g(x), S_i) = (b, s_j)] = \Pr[g(x) = b](&\Pr[g(x) = b] \Pr[S_i = s_j \mid g(x) = b] \\ &+ \Pr[g(x) = 1 - b] \Pr[S_i = s_j \mid g(x) = 1 - b]).\end{aligned}$$

Combining

$$\begin{aligned}|\Pr[(g(x), S_i) = (b, s_j)] - &\Pr[(g(x), S_i) = (b, s_j)]| \\ &= \Pr[g(x) = 0] \Pr[g(x) = 1]| \Pr_{x\sim\mu_0}[S_i \in s_j] - \Pr_{x\sim\mu_1}[S_i \in s_j]|.\end{aligned}$$

Adding over all $b \in \{0, 1\}$ and $j \in [k]$, we get

$$\begin{aligned}\mathrm{TV}((S_i, g(x)), S_i \times g(x)) &= 2 \Pr[g(x) = 0] \Pr[g(x) = 1] \cdot \mathrm{TV}(\sigma_0(T, v), \sigma_1(T, v)) \\ &= 2 \Pr[g(x) = 0] \Pr[g(x) = 1] \cdot \Delta(v).\end{aligned}$$

Plugging this into $I(S_i; g(x)) \geq 2\mathrm{TV}((S_i, g(x)), S_i \times g(x))^2$ gives the statement of the claim. □

The rest of the proof stays exactly the same as in [GLSS23]. □