Electronic Colloquium on Computational Complexity, Report No. 139 (2024)

Distinguishing, Predicting, and Certifying: On the Long Reach of Partial Notions of Pseudorandomness

Jiatu Li*
MIT
jiatuli@mit.edu

Edward Pyne†
MIT
epyne@mit.edu

Roei Tell
University of Toronto
roei@cs.toronto.edu

September 11, 2024

Abstract

This paper revisits the study of two classical technical tools in theoretical computer science: Yao's transformation of distinguishers to next-bit predictors (FOCS 1982), and the "reconstruction paradigm" in pseudorandomness (e.g., as in Nisan and Wigderson, JCSS 1994). Recent works of Pyne, Raz, and Zhan (FOCS 2023) and Doron, Pyne, and Tell (STOC 2024) showed that both of these tools can be *derandomized* in the specific context of *read-once branching programs* (ROBPs), but left open the question of derandomizing them in more general settings.

Our main contributions give appealing evidence that derandomization of the two tools is possible in general settings, show surprisingly strong consequences of such derandomization, and reveal several new settings where such derandomization is unconditionally possible for algorithms stronger than ROBPs (with useful consequences). Specifically:

- We show that derandomizing these tools is equivalent to general derandomization. Specifically, we show that derandomizing distinguish-to-predict transformations is equivalent to prBPP = prP, and that derandomized reconstruction procedures (in a more general sense that we introduce) is equivalent to prBPP = prZPP. These statements hold even when scaled down to weak circuit classes and to algorithms that run in super-polynomial time.
- Our main technical contributions are unconditional constructions of derandomized versions of Yao's transformation (or reductions of this task to other problems) for classes and for algorithms beyond ROBPs. Consequently, we deduce new results: A significant relaxation of the hypotheses required to derandomize the isolation lemma for logspace algorithms and deduce that NL = UL; and proofs that derandomization necessitates targeted PRGs in catalytic logspace (unconditionally) and in logspace (conditionally).

In addition, we introduce a natural subclass of prZPP that has been implicitly studied in recent works (Korten FOCS 2021, CCC 2022): The class of problems reducible to a problem called "Lossy Code". We provide a structural characterization for this class in terms of derandomized reconstruction procedures, and show that this characterization is robust to several natural variations.

Lastly, we present alternative proofs for classical results in the theory of pseudorandomness (such as two-sided derandomization reducing to one-sided), relying on the notion of deterministically transforming distinguishers to predictors as the main technical tool.

*Supported by MIT Akamai Presidential Fellowship and NSF grant CCF-2127597.
†Supported by a Jane Street Graduate Research Fellowship.

ISSN 1433-8092

Contents

1 Introduction

This paper revisits the study of two classical technical tools in theoretical computer science: Yao's transformation of distinguishers to next-bit predictors [Yao82], and the "reconstruction paradigm", both of which will be explained next.

More than four decades ago, Yao introduced a very simple probabilistic transformation of any distinguisher for a distribution into a **next-bit predictor** for the same distribution; that is:

Definition 1.1 (distinguisher)**.** We say that $C\colon \{0,1\}^n \to \{0,1\}$ is an ε-distinguisher for a distribution $\mathbf{D}$ over $\{0,1\}^n$ if $\left|\mathbb{E}[C(\mathbf{U}_n)] - \mathbb{E}[C(\mathbf{D})]\right| \geq \varepsilon$, where $\mathbf{U}_n$ is the uniform distribution.

Definition 1.2 (next-bit predictor)**.** For $i \in [n]$, we say that $P\colon \{0,1\}^{i-1} \to \{0,1\}$ is a δ-next-bit-predictor for a distribution $\mathbf{D}$ over $\{0,1\}^n$ if $\Pr_{x\leftarrow\mathbf{D}}[P(x_{<i}) = x_i] \geq \frac{1}{2} + \delta$.

Lemma 1.3 (Yao's next-bit-predictor)**.** *For any $C\colon \{0,1\}^n \to \{0,1\}$ and distribution $\mathbf{D}$ over $\{0,1\}^n$, if C is an ε-distinguisher for $\mathbf{D}$, then there exists $i \in [n]$ and $\sigma_1, \sigma_2 \in \{0,1\}$ such that with noticeable probability over $z \in \{0,1\}^{n-i+1}$ it holds that $P(x_{<i}) = C(x_{<i} \circ \sigma_1 \circ z) \oplus \sigma_2$ is a $(1/O(n))$-next-bit-predictor for $\mathbf{D}$.*

The simplicity and generality of this transformation have made it an invaluable tool, most prominently in cryptography and in pseudorandomness (expositions appear in standard textbooks, e.g. [Gol08; AB09; Gol01]). The canonical example for its use is in analyses of pseudorandom generator constructions: Assuming that the output distribution $\mathbf{D}$ of a generator does not "fool" C (i.e., C is a distinguisher for $\mathbf{D}$), one obtains a next-bit-predictor for $\mathbf{D}$, and the argument uses the latter to contradict the results or assumptions on which the generator is based.

Another ubiquitous technical tool is the **"reconstruction paradigm"**, which appeared explicitly in the works of Nisan and Wigderson [Nis91; NW94] and can be traced back to prior works (e.g., to [Sip88]). Loosely speaking, this is a general way of constructing algorithms from "hard strings" (e.g., truth-tables of hard functions, or incompressible strings). One designs an algorithm that transforms an input string f into the desired object $\mathcal{O}_f$, and this algorithm is coupled with a **reconstruction procedure**, which supports the following claim: For any string f, if $\mathcal{O}_f$ does not have the desired properties, then f is "not hard". Indeed, the reconstruction procedure outputs an "easy" representation of f, such as an efficient algorithm or a small circuit.

Our main focus in this context is trying to construct a distribution $\mathcal{O}_f$ that is pseudorandom for a class of efficient procedures. The canonical example is the "hardness versus randomness" paradigm, introduced in [Nis91; NW94]: In this context, the string f represents the truth-table of a hard function, the algorithm transforms f into a (hopefully pseudorandom) multiset $\mathcal{O}_f$, and the reconstruction procedure shows that if $\mathcal{O}_f$ is not pseudorandom, then the function represented by f can be computed efficiently (e.g., by a small circuit).[1] Closely related examples exist in numerous areas, such as extractor theory [Tre01], expanders [TSUZ07], and error-correcting codes [STV01].[2]

[1] More recent versions of the "hardness vs randomness" paradigm, following [Gol11b; CT21a], work in an instance-wise fashion: Given input x, they compute $f = g(x)$ for some function g, and the reconstruction procedure shows that if $\mathcal{O}_f$ is not pseudorandom, then g is easy to compute on the specific input x. See, e.g., [CT23a] for more details.

[2] Many standard reconstruction procedures (e.g., [NW94; TSZS06; SU05; Uma03]) use Yao's next-bit-predictor lemma as a key step. This can be viewed as *reducing* reconstruction to constructing a next-bit-predictor. We further explain this point in Section 1.4 and Section 7.

1.1 Recent Progress: Derandomizing Yao's Transformation and Reconstruction Arguments for Read-Once Branching Programs

The computational complexity of these two technical tools is crucial. This is because the tools are used in analyses that contradict an initial assumption or result, and the higher the complexity of the tools, the stronger the assumption or result that we need. As an illustrative example, suppose that we want to use Yao's transformation to show that a distribution $\mathbf{D}$ is pseudorandom for a class $\mathcal{C}$. In the analysis, we assume towards a contradiction that $\mathbf{D}$ is not pseudorandom for some $C \in \mathcal{C}$, and use Lemma 1.3 to obtain a predictor P; the rest of the argument, which proceeds to contradict an initial assumption or result, will carry on the overhead of transforming C to P.

In essentially all classical applications we are aware of, both tools are modeled as probabilistic procedures, or worse, as non-uniform procedures (which are stronger than probabilistic algorithms). However, very recent works showed that in the *specific context of read-once branching programs*, we can do better. To be more concrete, let us recall a definition of Doron, Pyne, and Tell [DPT24]:

Definition 1.4 (D2P, simplified; see Definition 3.7)**.** An algorithm A is a distinguish to predict (D2P) transformation for a class $\mathcal{C}$ if A gets as input a description of a circuit $C\colon \{0,1\}^n \to \{0,1\}$ from $\mathcal{C}$, and prints a list of circuits $P_1, ..., P_m\colon \{0,1\}^* \to \{0,1\}$ such that for *every* distribution $\mathbf{D}$ over $\{0,1\}^n$ the following holds. If C is an ε-distinguisher for $\mathbf{D}$, then there is an $i \in [m]$ such that P_i is an $(\varepsilon/O(n))$-predictor for $\mathbf{D}$.

Some choices in Definition 1.4 may seem arbitrary at this point (e.g., we could also define a non-black-box transformation that takes the distribution $\mathbf{D}$ as part of its input, or require the transformation to work only for certain classes of distributions, or consider more general parameter regimes). Nevertheless, these choices will be justified by showing algorithms that satisfy Definition 1.4 as well as matching lower bounds (the lower bounds appear in Appendix B).

Indeed, Yao's transformation (i.e., Lemma 1.3) can be thought of as a probabilistic D2P transformation for general circuits. The works of Pyne, Raz, and Zhan [PRZ23] and [DPT24], building on [Nis94; CH22; GRZ23], showed that there is a *deterministic logspace D2P* algorithm for the class of *read-once branching programs* (ROBPs). While it is not surprising that their algorithm runs in logarithmic space (since Lemma 1.3 already yields a probabilistic logspace D2P for ROBPs), the crucial novel point is that the D2P transformation algorithm can be made deterministic.

As a consequence of their D2P algorithm, they deduced the existence of a *derandomized reconstruction procedure* for the classical Nisan-Wigderson PRG [NW94] when the distinguisher is an ROBP.[3] This resulted in what they called **"certified derandomization"**: A deterministic logspace algorithm that gets as input a truth-table f and an ROBP C, and either confirms that the PRG instantiated with f is pseudorandom for C, or prints a small circuit whose truth-table is f.

Beyond the ROBP setting, however, the notions of deterministic D2P transformation and certified derandomization have not been studied in detail. For general circuits, it is not a-priori clear whether to expect impossibility results (on the one hand) or easy constructions (on the other hand). In fact, it is not even a-priori clear that *non-explicit* deterministic D2P transformations exist for general circuits, since the algorithm in Definition 1.4 is required to work for *all* distributions $\mathbf{D}$.

[3]The notion of "derandomized" here means that the procedure uses only $O(\log(|f|))$ random coins, where f is the truth-table of the function on which the generator is based. Indeed, the reconstruction procedure is deterministic, but its complexity may be higher than that of computing f to begin with (where the point is that it outputs a small circuit for f).

1.2 Our contributions: A bird's eye

In this work we give appealing evidence that derandomization of the two tools is possible, show surprisingly strong consequences of such derandomization, and reveal several new settings where such derandomization is unconditionally possible for algorithms stronger than ROBPs (and has useful consequences). Specifically:

- We show that derandomizing these tools is equivalent to general derandomization. In particular, we show that derandomized D2P is equivalent to **prBPP** = **prP** and that certified derandomization (in a more general sense that we introduce) is equivalent to **prBPP** = **prZPP**. These results appear in Sections 1.3 and 1.4.

- Our main technical contributions are unconditional constructions of derandomized D2P transformations (or reductions of this task to other problems) for classes and for algorithms beyond ROBPs. Consequently, we deduce new results: A significant relaxation of the hypotheses required to derandomize the isolation lemma for logspace algorithms and deduce that **NL** = **UL**; and proofs that derandomization necessitates targeted PRGs in catalytic logspace (unconditionally) and in logspace (conditionally). These contributions appear in Sections 1.3.1 and 1.3.2.

In addition, we introduce a natural subclass of **prZPP** that has been implicitly studied in recent works on the range avoidance problem [Kor21; Kor22; ILW23]: The class of problems reducible to a problem called LossyCode (see Problem 1.16). We provide a structural characterization for this class using the notion of certified derandomization, and show that this characterization is robust to several natural variations (see Section 1.4).

As a last contribution, we present alternative proofs of two classical results in the theory of pseudorandomness: The reduction of derandomization of **prBPP** to derandomization of **prRP** [Sip83; Lau83; ACR98; ACR+99; GVW11; GZ11; CH22], and the fact that **MA** $\subseteq$ $\mathbf{S_2P}$ [RS98]. Our proofs are technically simple and appealing, and rely on D2P transformations as a main ingredient. See Appendix A.

1.3 Derandomized D2P Transformations and Their Consequences

Should we expect D2P transformations to exist, and should we expect to explicitly construct them any time soon? A recent result of Korten [Kor22, Corollary 41] implies the following statement: If there is a deterministic D2P transformation for general circuits, then **BPP** $\subseteq$ **NP**. Moreover, it was implicitly proved by Goldreich [Gol11b, Appendix A] (following ideas in [GW00]) that D2P transformation exists with respect to a fixed universal distribution, assuming **prBPP** = **prP**. Both works, however, do not settle the existence of this transformation, even in the non-explicit setting.

The first result, which motivates the result of our work, asserts that a derandomized D2P algorithm for general circuits follows from general derandomization (i.e., from **prBPP** = **prP**), and is in fact *equivalent* to it.

Theorem 1.5 (D2P $\iff$ derandomization)**.** *The following are equivalent:*

1. **prBPP** = **prP**.

2. *There exists a deterministic polynomial-time D2P algorithm for general circuits.*

Moreover, there unconditionally exists a polynomial-sized family of non-uniform circuits for D2P of general circuits.

The surprisingly simple proof of Theorem 1.5 combines an idea of Goldreich and Wigderson [GW00] with the recent "instance-wise" approach to derandomization (following [Gol11b; CT21a]).

The equivalence in Theorem 1.5 has a positive aspect and a discouraging one: The result means that D2P exists under the widely believed conjecture $\mathbf{prBPP} = \mathbf{prP}$, but it also means that constructing a D2P algorithm requires proving this conjecture. We focus on the positive aspect.

A natural challenge: Derandomizing D2P beyond ROBPs. Motivated by Theorem 1.5, we consider the following challenge:

Open Problem 1.6. Unconditionally construct deterministic D2P transformations (or deterministic reductions of D2P to other tasks) for algorithms beyond the ROBP setting, and leverage these constructions to make progress on long-standing questions.

Our main technical contributions are two solutions to Open Problem 1.6: We construct new D2P transformations, going beyond the ROBP setting, and leverage them to make progress on the following two long-standing questions: making nondeterministic logspace unambiguous (Section 1.3.1), and reducing targeted PRGs to derandomization (Section 1.3.2). We view these positive results as suggesting that more positive answers to Open Problem 1.6 may be found.

1.3.1 The Isolation Lemma And Unambiguous Logspace

The isolation lemma of Mulmuley, Valiant, and Vazirani [VV86; MVV87] (see also [CRS95]) gives a randomized procedure to reduce a search problem with many solutions to one with a single valid solution. This procedure has found many uses in algorithms and complexity; among the well-known examples are [Tod91; BDCG+92]. We focus on its application in reducing nondeterminism to *unambiguous* nondeterminism, where each "yes" instance has exactly one valid witness (as in [Val76]).

In the general case, there is evidence that derandomizing the isolation lemma for this purpose is impossible (since it was shown in [DKM+13] to be equivalent to $\mathbf{NP} \subseteq \mathbf{P}/\operatorname{poly}$), and even derandomizing restricted versions of it implies circuit lower bounds [AM08]. Part of the difficulty is that it is not clear how to identify a good candidate (i.e., an instance that has exactly one satisfying solution) in an unambiguous way, so even strong PRGs are not known to imply $\mathbf{NP} = \mathbf{UP}$.

In the bounded-space setting, however, there is evidence that we *can* make nondeterminism unambiguous. In contrast to recognizing circuits with a unique satisfying assignment, recognizing if a graph has unique shortest paths can be done in $\mathbf{UL}$ [RA00; GW96]. Leveraging this fact, Reinhardt and Allender [RA00] showed that to prove $\mathbf{NL} = \mathbf{UL}$, it suffices to construct weight functions that induce *unique shortest paths* in $\mathbf{UL}$:

Problem 1.7 (path isolation)**.** Construct in $\mathbf{UL}$ a set of weight functions $\{w_1, \ldots, w_{n^c}\}$ with $w_i : E \to [n^{10}]$ such that for every graph $G = (V, E)$ on n vertices, there is some i such that the weighted graph (G, w_i) has *unique shortest paths (USPs).*

Allender et al. [ARZ99] showed that such a construction is possible assuming strong circuit lower bounds: in particular, hardness of $\mathbf{SPACE}[n]$ for general circuits of size $2^{\varepsilon n}$. Subsequently, there has been extensive work designing space-efficient unambiguous verifiers for various problems. In particular, placing connectivity for restricted families of directed graphs in $\mathbf{UL}$ [BTV09; KV10; GST19], reducing the complexity of connectivity for general graphs [MP19; KT16; Hoz19].

The reason prior conditional results of [GW96; ARZ99] required strong circuit lower bounds is precisely because they apply the generic probabilistic D2P transformation of Yao to the distinguisher $T_G(w)$ that checks if (G, w) has unique shortest paths.

Our results. Motivated by this observation, we unconditionally construct a deterministic logspace D2P transformation for this particular distinguisher T_G:

Theorem 1.8 (Informal, see Theorem 6.8)**.** *There is a logspace-computable D2P transformation for the class of distinguishers $\{T_G\}_G$, where G is a directed graph and $T_G(w) = \mathbb{I}[w \text{ induces USPs in } G]$.*

We leverage this transformation to significantly weaken the assumptions needed in previous work [ARZ99] to deduce that non-deterministic bounded-space computation can be made unambiguous. Specifically, instead of strong circuit lower bounds, we show that lower bounds for *uniform and deterministic* (or near-deterministic) algorithms suffice.

We show two results, one for a "scaled-up" parameter setting, and one for the logspace setting. For context, recall that [ARZ99] deduced that $\mathsf{NL} = \mathsf{UL}$ from hardness of $\mathsf{SPACE}[n]$ for general non-uniform circuits of size $2^{\varepsilon n}$ (for some $\varepsilon > 0$).

Our first result deduces a scaled-up version of their conclusion from a lower bound for *uniform and deterministic procedures*; specifically, from a lower bound for circuits that are printable by a nondeterministic logspace algorithm, that moreover prints a circuit on only one guess sequence.[4] Specifically:

Theorem 1.9 (Informal, see Theorem 6.4)**.** *Suppose there exists $\varepsilon > 0$ such that $\mathsf{SPACE}[n]$ is hard for UL-uniform circuits of size $2^{\varepsilon n}$. Then $\mathsf{NSPACE}[O(n)] = \mathsf{USPACE}[O(n)]$.*

We stress that the hypotheses in Theorem 1.9 (and Assumption 1.10) are considerably weaker than hypotheses that are typically required for hardness-vs-randomness results. In particular, the latter rely on lower bounds either for non-uniform circuits (as in, say, [NW94; IW97]) or for probabilistic algorithms (as in, say, [CT21a; CTW23]).[5]

In fact, our technical result is stronger, and shows that the conclusion of Theorem 1.9 follows from a weaker hypothesis; namely, from hardness of $\mathsf{USPACE}[n]$ against (deterministic, uniform) TC^0 circuits with low-space oracles (see Theorem 6.4).

For our second result, to deduce that $\mathsf{NL} = \mathsf{UL}$ (i.e., a scaled-down conclusion as in [ARZ99]), we will assume hardness of functions in NC^1 for uniform algorithms that use only polylog(n) random coins. Specifically:

Assumption 1.10 (hardness in NC^1 for uniform near-deterministic algorithms)**.** For every $c \in \mathbb{N}$, there exists $C \in \mathbb{N}$ and a family of functions $\{f : \{0,1\}^n \to \{0,1\}^n\}_{n\in\mathbb{N}}$ computable by logspace-uniform NC^1-circuits of size n^C, such that there is no time n^c algorithm using polylog(n) many coins that on infinitely many x prints $f(x)$ with probability at least $2/3$.

Theorem 1.11. *Suppose that Assumption 1.10 holds. Then $\mathsf{NL} = \mathsf{UL}$.*

The technical contribution underlying the foregoing results is two-fold: Constructing the D2P transformation stated in Theorem 1.8, and leveraging it via new "hardness-vs-randomness" tradeoffs to obtain Theorems 1.9 and 1.11. The latter contribution further develops very recent work on targeted pseudorandom generators with randomness-efficient reconstruction procedures [PRZ23; DPT24]. In particular, in the proof of Theorem 1.9 we show that such reconstruction procedures can be made to satisfy $\mathsf{UL}\cap\mathsf{coUL}$ uniformity, and in the proof Theorem 1.11, we construct a version of the Chen-Tell targeted HSG [CT21a] that works with a hard function in NC^1, is computable in logspace, and has a derandomized reconstruction. See Section 2 for details.

[4] The uniformity requirement is significantly weaker than the standard requirement that the circuit is printable in polynomial time (i.e., P-uniformity), let alone from models such as NTIME-uniformity (e.g., as in [SW13; CRT+20]).

[5] One recent exception is the work of Doron, Pyne, and Tell [DPT24] on derandomizing BPL, and another exception is the study of pseudorandomness for deterministic observers by Goldreich and Wigderson [GW00]. We build on both works, and in particular we extend the results of [DPT24] the specific setting of ROBPs to more general settings. See Section 2 for further details.

1.3.2 Derandomization Requires Targeted Generators in CL and in L

Assuming that we can solve CAPP for a class $\mathcal{C}$ of circuits,[6] can we also output a a distribution $\mathbf{D}$ that fools a given $C \in \mathcal{C}$? In particular, do **BPP**-*search* problems (a-la [Gol11b]) reduce to the *decision* problem CAPP? This question was first posed by Goldreich [Gol11b; Gol11c], who phrased it as the question of whether derandomization requires *targeted PRGs*.

Goldreich [Gol11c; Gol11b] proved such a result for **prBPP** (i.e., when the CAPP algorithm is a general probabilistic algorithm), and posed the open question of obtaining analogous results for classes such as **AM** and **L**. For recent progress see, e.g., [HU22; MS23a; MS23b; PR23].

We *resolve this question for the catalytic logspace (***CL***) model* of Buhrman et al. [BCK+14], and weaken the assumptions required to resolve this question for **L**.

Derandomization in CL requires targeted PRGs. In the catalytic logspace model, we are given $O(\log n)$ bits of standard workspace, and a *catalytic* tape $\mathbf{w}$ of length n^c, which functions as follows. The tape $\mathbf{w}$ is initialized to an arbitrary value, and we may edit it during the computation, but must exactly reset the tape to the original configuration at the end. The work of [BCK+14] proved that logspace-uniform $\mathbf{TC}^1$ is contained in **CL**, so in particular **NL** $\subseteq$ **CL**. Since this intriguing result there has been extensive work on the model [BKL+18; GJS+19; DGJ+12; CM20; CM23; DPT24; Pyn24; CLM+24] (see the survey of Mertz [Mer23] for an excellent exposition).

Despite extensive recent interest, many basic structural questions remain open. In particular, prior to this work it was not known whether solving **BPP**-search problems reduces to CAPP in **CL**. As mentioned above, we resolve this question in the affirmative:

Theorem 1.12 (informal, see Theorem 5.4)**.** *Suppose that there is a* **CL***-computable* CAPP *algorithm for a* **CL***-evaluable class of circuits $\mathcal{C}$. Then:*

1. *There is a* **CL***-computable D2P transformation for $\mathcal{C}$ circuits.*
2. *There is a* **CL** *algorithm that, given $C \in \mathcal{C}$, outputs a distribution* $\mathbf{D}$ *that* $(1/3)$*-fools C.*

Our proof proceeds in two steps: We first reduce the task of producing $\mathbf{D}$ to D2P in **CL**, and then reduce D2P to CAPP in **CL**. The first and main step combines the "compress or random" approach in catalytic computation (which tries to use the catalytic tape as a hard truth-table; see, e.g., [DPT24] and [Mer23, Section 3.2.1]) with ideas from the proof of Theorem 1.5. The second step shows that the reduction of D2P to CAPP from Theorem 1.5 can be implemented in **CL**. Crucially, our proof relies on the fact that our reductions of D2P to CAPP are "instance-wise", in the sense that a CAPP algorithm for a fixed circuit C yields a deterministic D2P transformation for C specifically. Details appear in Section 2.3.

In addition, combining the ideas in the proofs of Theorems 1.8 and 1.12 with the main result of [BCK+14], we show that we can derandomize the path isolation lemma in **CL** (Theorem 6.13). Note that this is the first **CL** algorithm for a natural problem that combines *both* main algorithmic techniques for **CL**, namely the algebraic computation approach [BCK+14; CM23] and the compress-or-random approach [Pyn24; DPT24].

Derandomization in L requires targeted PRGs, under weak assumptions. Finally, we show that derandomization in logspace indeed implies targeted logspace PRGs, assuming lower bounds against uniform algorithms that use only polylog(n) random coins:

[6] Recall that CAPP is the promise problem whose "yes" instances are circuits C such that $\Pr_r[C(r) = 1] \geq 2/3$ and whose "no" instances are circuits C such that $\Pr_r[C(r) = 1] \leq 1/3$. Also recall that CAPP is complete for **prBPP** (Definition 3.5).

Theorem 1.13 (informal, see Theorem 5.7)**.** *Suppose that Assumption 1.10 holds. Let $\mathcal{C}$ be an arbitrary circuit class that is evaluable in* L*, and suppose there is a logspace* CAPP *algorithm for $\mathcal{C}$. Then, there is a logspace algorithm that, given $C \in \mathcal{C}$, outputs a distribution D that $(1/3)$-fools C.*

The conditional statement in Theorem 1.13 is the first result indicating that the answer to the open problem is affirmative (i.e., that logspace derandomization necessitates logspace targeted PRGs) without relying on assumptions that are sufficiently strong to immediately yield (by themselves) logspace targeted PRGs. Further details appear in Sections 2.3 and 5.

1.4 Certified Derandomization and the Class LOSSY

We now turn our attention to a notion of *derandomized reconstruction procedures*. Specifically, we focus on what was coined by Pyne, Raz, and Zhan [PRZ23] as certified derandomization: Given a circuit $C : \{0,1\}^n \to \{0,1\}$ and a truth-table $f \in \{0,1\}^{\mathrm{poly}(n)}$, either estimate the acceptance probability of C to an additive error of $1/6$ (i.e. solve CAPP for C), or construct a small circuit for f. (Indeed, we think of such an algorithm as executing a derandomized version of the classical reconstruction procedure a-la [NW94]: When the algorithm is unable to use f to obtain a pseudorandom distribution for C, it deterministically finds a small circuit for f.)

Context: Certified derandomization, derandomized D2P, and reconstruction procedures. Certified derandomization is a natural notion in and of itself, which was constructed for ROBPs in [PRZ23] and which can be constructed for more general classes (see next).

An additional motivation for studying certified derandomization arises from understanding reconstruction procedures in the "hardness vs randomness" paradigm. Classical reconstruction procedures (e.g., in [NW94; TSZS06; SU05; Uma03]) use D2P as a key technical step. However, it is not clear if this approach (i.e., designing reconstruction procedures that go through D2P) is necessary, or if there are approaches that avoid D2P altogether.[7]

To be more concrete, we know that certified derandomization deterministically reduces to constructing a D2P transformation: Either using the derandomized reconstruction procedure for the Nisan-Widgerson PRG of [PRZ23], or using our equivalence between derandomizing D2P and $\mathbf{prBPP} = \mathbf{prP}$ (the latter trivially implies certified derandomization, as we can simply ignore the truth-table f provided and solve CAPP). Nevertheless, the characterization of certified derandomization stated below implies that *reducing certified derandomization to D2P may be an overkill*, as the latter is equivalent to $\mathbf{prBPP} = \mathbf{prP}$ (by Theorem 1.5) whereas the former is only equivalent to $\mathbf{prBPP} = \mathbf{prZPP}$ (see Theorem 1.15).

A general notion of certified derandomization, and $\mathbf{prZPP} = \mathbf{prBPP}$. The certified derandomization algorithm of [PRZ23] uses strings f that are truth-tables with high circuit complexity (i.e., if f has high circuit complexity, then the algorithm estimates the acceptance probability the given circuit). One can think of the set of truth-tables with high circuit complexity as a dense property of strings (i.e., inspired by Razborov and Rudich [RR97]), where this property is in $\mathbf{coNP}$. It is not, however, clear why we should use this specific property for the purpose of certified derandomization, rather than any other property in $\mathbf{coNP}$.

Accordingly, we define a general notion of certified derandomization using *an arbitrary dense property* $\mathcal{P} \in \mathbf{coNP}$: The certified derandomization algorithm is given C and a string τ, and is

[7] A closely related question, focusing on the *hybrid argument*, was studied by Fefferman *et al.* [FSU+13] motivated by avoiding the $1/n$ advantage loss.

required to either solve CAPP for C (which it should be able to do whenever $\tau \in \mathcal{P}$), or provide a witness w that $\tau \notin \mathcal{P}$ (see Definition 7.1).

We prove that certified derandomization in this more general sense is *equivalent* to **prBPP** = **prZPP**; that is, **prBPP** = **prZPP** if and only if there exists a dense property $\mathcal{P} \in$ **coNP** and a certified derandomization algorithm using $\mathcal{P}$ (see Theorem 7.4). As explained above, this result yields a conceptual separation between certified derandomization and D2P.

Definition 1.14 (certified derandomization)**.** For $\ell = \ell(n) = 2^{o(n)}$, let $\mathcal{P} = \{\mathcal{P}_n \subseteq \{0,1\}^\ell\}_{n\in\mathbb{N}} \in$ **coNP** such that $\mathcal{P} \cap \{0,1\}^n \neq \varnothing$ for every $n \in \mathbb{N}$. Let V be a **coNP** verifier of $\mathcal{P}$. An algorithm A is a certified derandomization algorithm using $\mathcal{P}$ (with respect to the verifier V) if for every linear-size circuit $C : \{0,1\}^n \to \{0,1\}$ and every $\tau \in \{0,1\}^\ell$,

- If $\tau \in \mathcal{P}$ then $A(C,\tau)$ solves CAPP on C.
- If $\tau \notin \mathcal{P}$ then either $A(C,\tau)$ solves CAPP on C, or $A(C,\tau)$ prints w such that $V(\tau,w) = 0$.

Theorem 1.15 (certified derandomization $\iff$ **prBPP** = **prZPP**; see Theorem 7.4)**.** *The following statements are equivalent.*

- **prBPP** = **prZPP**.
- *There is a deterministic polynomial-time certified derandomization algorithm using a dense property* $\mathcal{P} = \{\mathcal{P}_n \subseteq \{0,1\}^\ell\}_{n\in\mathbb{N}} \in$ **coNP**, *where* $\ell = \ell(n) = \mathrm{poly}(n)$.

The proof of Theorem 1.15 is elementary, and it appears in Section 7.2 (where we show a more general equivalence; see Theorem 7.4.

The class LOSSY**.** It turns out, however, that the more restricted notion of certified derandomization with hard truth-tables (as in [PRZ23]) is interesting in and of itself. A recent work of Korten [Kor22] introduced a search problem called LossyCode, which admits a randomized polynomial-time zero-error algorithm, and asked what is the set of problems reducible to LossyCode.

Problem 1.16 (LossyCode)**.** Given a pair of circuits $C : \{0,1\}^n \to \{0,1\}^m$, $D : \{0,1\}^m \to \{0,1\}^n$, where $m < n$, find a string $x \in \{0,1\}^n$ such that $D(C(x)) \neq x$.

We define the subclass **LOSSY** $\subseteq$ **ZPP** as the class of languages reducible to LossyCode in deterministic polynomial time.[8] This is an interesting class, with one motivation coming from proof complexity: Loosely speaking, if a statement of the form "$\forall x\ \exists y\ \varphi(x,y)$" (where φ is a quantifier-free formula) can be proved in the bounded theory $\mathbf{APC}_1$ (see [Jeř04; Jeř07] for the definition and related discussion), then the corresponding search problem can be solved in **FLOSSY** (i.e., in the functional version of **LOSSY**; see Appendix D for more details).

We provide additional motivation for studying **LOSSY**, by showing that this class has an interesting structural characterization, which relies on the notion of certified derandomization with hard truth tables. Specifically, we prove the following:

Theorem 1.17 (informal, see Theorem 7.7)**.** *The following statements are equivalent.*

(1) **prBPP** = **prLOSSY**.

(2) *There is a deterministic polynomial-time certified derandomization algorithm using hard truth tables.*

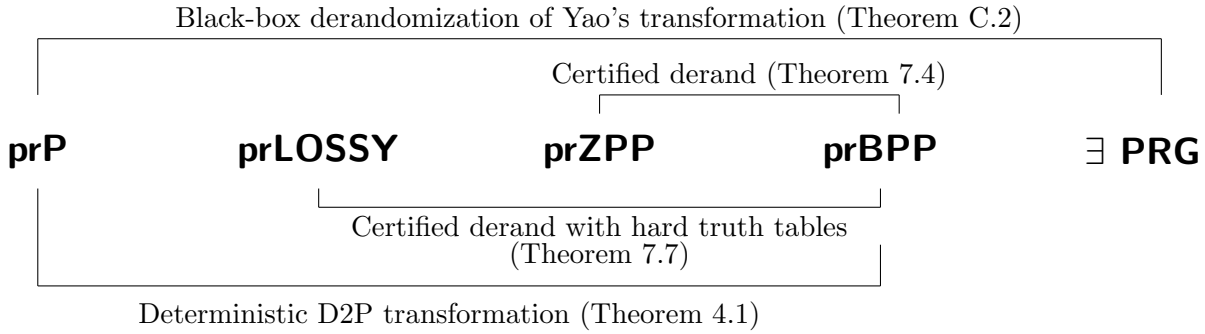

Figure 1: Characterizations of various types of derandomization of **prBPP**. Note that black-box derandomization of Yao's transformation refers to the problem of finding a suitable suffix z in Lemma 1.3 that works for every (fixed-size) distributions and every (fixed-size) circuits (see Appendix C for more details); another notion for "non-black-box" derandomization of Yao's transformation is also discussed in Appendix C.2.

In fact, we further extend Theorem 1.17, by showing that the two statements are also equivalent to the existence of a deterministic polynomial-time certified derandomization algorithm using any property defined by an efficient *Range Avoidance* problem.[9] These equivalences significantly strengthen results from [Kor22]; see Section 7.2 and Section 7.3 for further details.

For a visual illustration of the connections between the notions presented throughout the introduction, see Figure 1. The illustration also mentions a notion of derandomization of Yao's transformation (i.e., deterministically finding a good suffix z such that the conclusion of Lemma 1.3 holds), which will be discussed in Appendix C.

2 Overview of Proofs

In Section 2.1 we describe the equivalence between D2P and derandomization (i.e., the proof of Theorem 1.5). Our main technical contributions are described in Sections 2.2 and 2.3:

- In Section 2.2 we explain our construction of a deterministic logspace D2P algorithm for unique shortest paths (i.e., Theorem 1.8) and how it allows deducing $\mathbf{NL} = \mathbf{UL}$ and $\mathbf{NSPACE}[n] = \mathbf{USPACE}[n]$ from weaker assumptions (i.e., Theorems 1.9 and 1.11).
- In Section 2.3 we explain our reduction of targeted PRGs to CAPP in catalytic logspace (i.e., Theorem 1.12) and the analogous conditional reduction in $\mathbf{L}$ (i.e., Theorem 1.13).

Finally, in Section 2.4 we describe the equivalences between certified derandomization and zero-error derandomization.

[8] It turns out that the definition of **LOSSY** is robust with respect to the type of the reduction. Specifically, we prove in Lemma 7.9 that any language that is Cook-type reducible to LossyCode (i.e. the reduction could call the LossyCode oracle multiple times adaptively) is also Karp-type reducible to LossyCode (i.e. the reduction calls the LossyCode oracle only once).

[9] That is, any property of strings that are outside the range of an efficiently computable function $g : \{0,1\}^n \to \{0,1\}^m$ such that $m > n$ (i.e. a string $y \in \{0,1\}^m$ such that $g^{-1}(y) = \varnothing$); see Section 7 for details. For more details about the range avoidance problem, see [Kor21; RSW22; GLW22; GGN+23; CHL+23; ILW23; CHR24; Li24; CL24].

2.1 D2P is Equivalent to Derandomization

Our goal is to show that deterministic D2P implies $\mathsf{prBPP} = \mathsf{prP}$, and vice versa. At a high level, we build on ideas from a sequence of works by Goldreich and Wigderson [GW00; Gol11c; Gol11b], who examined what they called "deterministic observers". We prove the equivalence by combining their ideas with the "instance-wise" approach to derandomization (i.e., derandomization that uses targeted PRGs instead of classical PRGs), following Goldreich [Gol11b] and Chen and Tell [CT21a].

2.1.1 D2P implies derandomization

Goldreich and Wigderson [GW00] constructed a distribution ensemble $\mathbf{D} = \{\mathbf{D}_n\}_{n\in\mathbb{N}}$ that is *unpredictable* by uniform deterministic Turing machines. In a gist, on input length n they consider the first (say) n machines, and using a diagonalization-style approach, they build $\mathbf{D}_n$ bit-by-bit. In each iteration the distribution $\mathbf{D}_n^{(i)}$ consists of i-bit strings, and they search a pseudorandom sample space to find an extension of the strings in $\mathbf{D}_n^{(i)}$ such that none of the n machines can predict the new distribution $\mathbf{D}_n^{(i+1)}$.[10]

At a high level, if deterministic D2P is possible, then any efficient distinguisher yields a deterministic predictor for $\mathbf{D}_n$. Since $\mathbf{D}_n$ is unpredictable by such machines (by its construction), we intuitively expect to deduce that $\mathbf{D}_n$ is also pseudorandom.

The only issue is that (in contrast to [GW00]) we are trying to obtain worst-case derandomization. That is, in our setting the machine M that tries to distinguish $\mathbf{D}_n$ from uniform also has access to a (worst-case) input $x \in \{0,1\}^n$. Hence, instead of trying to construct a distribution that is unpredictable by such procedures, we simply adapt the approach to yield *non-black-box* derandomization. Specifically, given input $x \in \{0,1\}^n$, we build $\mathbf{D}_x$ that is unpredictable by any *efficient machine that also gets access to* x (using the same diagonalization-style approach). Assuming that deterministic D2P is possible, $\mathbf{D}_x$ is indistinguishable from uniform by efficient machines that get access to x, and in particular by $M(x,\cdot)$. Indeed, this construction is a targeted PRG, mapping x to a multiset $\mathbf{D}_x$ that is pseudorandom for distinguishers of the form $M(x,\cdot)$.[11]

This simple argument is versatile, and yields several interesting corollaries, for instance an equivalence between D2P and superfast derandomization under OWFs. We present the technical details as well as extensions and corollaries in Section 4.

2.1.2 Derandomization implies D2P

To discuss the other direction, recall that by Yao's Lemma, for every circuit $C : \{0,1\}^n \to \{0,1\}$ and distribution $\mathbf{D}_n$ that C distinguishes from uniform, there is an index $i \in [n]$ and $\sigma \in \{0,1\}^2$ such that

$$\mathop{\mathbb{E}}_{z}\left[\Pr_{x\leftarrow\mathbf{D}_n}[x_i = C(x_{<i}\circ\sigma_1\circ z)\oplus\sigma_2]\right] \geq 1/2 + 1/\operatorname{poly}(n). \tag{2.1}$$

A natural strategy is to try and find z that approximately achieves this expectation. For example, this is indeed the strategy undertaken in all uses of Yao's transformation in the "hardness vs randomness" paradigm (see, e.g., [NW94; STV01; CT21a; DPT24], for a collection of arguments

[10] They show that even a pairwise-independent distribution suffices for this purpose, while we apply stronger tools to obtain a stronger equivalence, see Remark 4.3.

[11] The "missing observations" in [GW00] seems to be two-fold: First, clearly defining the notion of D2P and asking about its implications; and secondly, considering non-black-box derandomization by targeted PRGs, which is a notion that was introduced in a follow-up work [Gol11b] and extensively studied only recently (following [CT21a]).

that all rely on finding such a z). Unfortunately, we show in Theorem C.2 that explicitly constructing a good family of these strings is equivalent to the explicit construction of *hitting sets* (and thus circuit lower bounds), and hence we are unlikely to deduce such a result from $\mathbf{prBPP} = \mathbf{prP}$.

Intuitively, the key difficulty is that testing whether z maintains the advantage in Eq. (2.1) for every distribution $\mathbf{D}_n$ over $\{0,1\}^n$ for which C is a distinguisher seems to require doubly exponential time (since there can be doubly exponentially many such distributions). Thus, it is unclear how to apply a CAPP algorithm to a polynomial-sized circuit to find such a z.

Our key observation is as follows. Instead of trying to use the CAPP algorithm to find a good z that will be hard-wired into a predictor P_z, we will construct a predictor that *uses the* CAPP *algorithm to predict the next bit*. This way, instead of considering all possible distributions over inputs $x_{<i}$ to the predictor, we *fix an input* $x_{<i}$ that is explicitly given to the CAPP algorithm, and the CAPP algorithm only needs to consider the uniform distribution over suffixes z to approximate the LHS of Eq. (2.1) on this given prefix. (This approach is technically reminiscent of a proof in [Gol11b, Appendix A], although the settings and notions are different.)

In more detail, fixing $i \in [n]$ and $\sigma \in \{0,1\}$, let us first construct a "non-Boolean predictor" $\tilde{P}_{i,\sigma}$. For simplicity of presentation, let us assume that $\sigma = 00$, and denote $\tilde{P}_i = \tilde{P}_{i,\sigma}$. Given input $x_{<i}$, the predictor estimates the value $\mathbb{E}_z[C(x_{<i} \circ z)]$, up to a polynomially small error.[12] By Eq. (2.1) and linearity of expectation, for some i, σ (again, assume $\sigma = 0$) we have

$$\begin{aligned} 1/2 + 1/\operatorname{poly}(n) &\leq \mathop{\mathbb{E}}_{x}\left[\mathbf{1}[x_i = 1] \cdot \Pr_z[C(x_{<i} \circ z) = 1] + \mathbf{1}[x_i = 0] \cdot \Pr_z[C(x_{<i} \circ z) = 0]\right] \\ &\approx_{1/\operatorname{poly}(n)} \mathop{\mathbb{E}}_{x}\left[\mathbf{1}[x_i = 1] \cdot \tilde{P}_i(x_{<i}) + \mathbf{1}[x_i = 0] \cdot (1 - \tilde{P}_i(x_{<i}))\right] \ ; \end{aligned} \tag{2.2}$$

in other words, $\tilde{P}$ computes a real value whose correlation with the event "$x_i = 1$" is non-trivial.

This almost finishes the construction, since now we just need to convert $\tilde{P}$ into a Boolean predictor. This can be done in a generic way: An elementary argument shows that for any real-valued function $\tilde{P}_i$ with correlation as in Eq. (2.2) there is a threshold $\tau \in \{i \cdot \varepsilon\}_{i=1,\ldots,1/\varepsilon}$ (where $\varepsilon = 1/\operatorname{poly}(n)$) such that the Boolean function $P_{i,\tau}(x_{<i}) = \mathbf{1}[\tilde{P}_i(x_{<i}) \leq \tau]$ has correlation $1/2 + 1/\operatorname{poly}(n)$ with the event "$x_i = 1$" (i.e., $\Pr_{x_{<i}}[P_{i,\tau}(x_{<i}) = x_i] \geq 1/2 + 1/\operatorname{poly}(n)$).[13]

Thus, our D2P algorithm outputs the collection $\{P_{i,\sigma,\tau}\}$. This collection is indeed of polynomial size, and for every distribution $\mathbf{D}_n$ for which Eq. (2.1) holds (in particular, for every distribution for which C is a distinguisher), the collection has a predictor for $\mathbf{D}_n$.

Remark 2.1. Indeed, this direction is a-priori far less obvious than the first one. Note that the argument is "instance-wise": Solving CAPP for (prefixes of) a certain circuit C yields a deterministic D2P transformation for C specifically. We will crucially use this property in Section 2.3.

2.2 D2P for Unique Shortest Paths, and Derandomizing the Path Isolation Lemma

We now explain how to construct a specific deterministic D2P transformation and use it to deduce that $\mathbf{NL} = \mathbf{UL}$ (or $\mathbf{NSPACE}[n] = \mathbf{USPACE}[n]$) from weak hardness assumptions (i.e., for deterministic uniform algorithms).

Our argument has two parts. We first construct a deterministic logspace-computable D2P for a distinguisher $T = T_G$ that decides whether a weight assignment induces unique shortest paths in

[12] That is, the predictor constructs the circuit $D(z) = C(x_{<i} \circ z)$, and outputs the real value obtained by applying the CAPP algorithm to D (with sufficiently small error $1/\operatorname{poly}(n)$).

[13] More generally, for any two random variables $\mathbf{x} \in [0,1]$ and $\mathbf{y} \in [-1,1]$ such that $\mathbb{E}[\mathbf{x} \cdot \mathbf{y}] \geq \delta$, there is $\tau \in [1/\varepsilon]$ such that $\mathbb{E}[\mathbf{1}_{\mathbf{x} \leq \varepsilon \cdot \tau} \cdot \mathbf{y}] \geq \delta - \varepsilon$ (see Fact 4.8).

the graph G. The reason for focusing on this specific T is that finding such a weight assignment suffices to deduce that $\mathsf{UL} = \mathsf{NL}$ (see [RA00; GW96]). The second part of our argument "lifts" this D2P to a proof that $\mathsf{UL} = \mathsf{NL}$, under weak assumptions.

A deterministic logspace D2P for unique shortest paths. Consider the function T_G that takes in a weight assignment $w : E \to [n^{10}]$ and accepts if the weighted graph (G, w) has *unique shortest paths*.[14] Note that T_G does not seem to be computable by an ROBP (since checking unique shortest paths between every pair of vertices seemingly requires reading edge weights many times over), and thus the previously known D2P transformation for ROBPs does not suffice.

In order to obtain our D2P transformation, we first make the distinguisher *stricter*. We place a fixed ordering $e_1, \ldots, e_m$ on the edges of $G = (V, E)$, define $G_i = (V, E_i = \{e_1, \ldots, e_i\})$, and let w_i be the restriction of $w : E \to [n^{10}]$ to E_i. Then we define:

$$T_G(w) = \bigwedge_{i \in [m]} \mathbb{I}\left[w_i \text{ induces USPs in } G_i\right],$$

i.e. every prefix of the weight function w likewise induces unique shortest paths. A random weight assignment still satisfies this stricter condition with high probability (see Lemma 6.9).

Recall that in Section 2.1.2 we showed a construction of D2P that uses a CAPP algorithm; this can be viewed as a reduction of D2P to CAPP. As mentioned in Remark 2.1, this reduction is instance-wise, in the sense that solving CAPP for a specific circuit yields D2P for that circuit. We now use the same instance-wise reduction of D2P to CAPP, while performing the hybrid argument of Yao (that yields Eq. (2.1)) in the same order as T_G reads the edge weights. We deduce that constructing a D2P transform for T_G reduces to solving the following problem:

$$\text{Given an arbitrary partial assignment } w_i : E_i \to [n^{10}], \text{ estimate } \mathop{\mathbb{E}}_{z}[T_G(w_i \circ z)].$$

The key point is that with the stricter distinguisher and the choice of hybrid order, we obtain a *polarization* effect. In particular, one of the following holds:

- The partial assignment has *already* failed to induce USPs in a subgraph G_j for $j \leq i$. In this case, $T_G(w_i \circ z) = 0$ for every z.
- The partial assignment has not already failed to induce USPs in a subgraph. In this case, we show that *almost all suffixes* z will successfully induce USPs, no matter the current prefix.

Due to this polarizing effect, we can estimate $\rho = \mathbb{E}_z[T(w_i \circ z)]$ by determining if w_i has already failed to induce USPs (in which case $\rho = 0$) or not (in which case $\rho \approx 1$). Allender and Reinhardt [RA00] constructed a $\mathsf{UL} \cap \mathsf{coUL}$ algorithm for this task (i.e., testing if a fixed assignment induces USPs), and thus we obtain a deterministic logspace D2P transformation where the predictors it outputs are $\mathsf{UL} \cap \mathsf{coUL}$ algorithms.

From D2P to disambiguation of non-deterministic logspace. Using the D2P above, we now deduce that $\mathsf{NL} = \mathsf{UL}$ (and $\mathsf{NSPACE}[n] = \mathsf{USPACE}[n]$) from hardness for uniform algorithms that are either deterministic or use only polylog(n) coins. The idea, following Pyne, Raz, and Zhan [PRZ23], is to use a (targeted) pseudorandom generator with a *near-deterministic reconstruction procedure* conditioned on a deterministic D2P for the relevant distinguisher.

[14] A simple argument shows that for every graph G, $\mathbb{E}[T_G(\mathbf{U})] \geq 1 - n^{-8}$.

In our case, we let the distinguisher be the test T_G that accepts if the weight set induces unique shortest paths. Given our deterministic D2P for this distinguisher, such a generator transforms hardness for near-deterministic procedures into a set of pseudorandom strings for the distinguisher (which, in our setting, will include a weight assignment that induces unique shortest paths on the input graph). The original generator of [PRZ23] was based on circuit lower bounds, and later on Doron, Pyne, and Tell [DPT24] (building on the framework of [CT21a]) constructed a targeted PRG based on lower bounds for deterministic uniform procedures.

Using these works as our starting point, we will need to construct yet another version of the targeted PRG of Chen and Tell [CT21a] (following [CRT22; CTW23; CLO+23; DPT24]).[15] We build a logspace-computable targeted PRG that is based on a hard function in logspace-uniform $\mathbf{NC}^1$, where the hardness is for uniform algorithms that use only polylogarithmically many random coins (and that have access to a deterministic D2P for the relevant distinguisher); for the full statement see Theorem 5.6. Since the technical details are quite involved (and are not the conceptual focus of the current paper), let us focus mostly on two key differences, postponing the full details to Appendix E.

- As in all previous works, our targeted PRG encodes the computation of the hard function as a sequence of polynomials. Previous works did so relying either on a hard function in $\mathbf{TC}^0$; or on a preprocessing step that incurs a polylog(n) depth blowup (using an idea from [Gol18]), which would prohibit evaluating the generator in logspace. To resolve this, we preprocess the circuit in a more careful way, which still incurs a polylog(n) depth blowup but nevertheless allows us to evaluate the resulting polynomials in logspace (see Claim E.3).

- The conclusion in [DPT24] was average-case derandomization, whereas we are interested in worst-case derandomization. The reason for their weaker conclusion is that their reconstruction algorithm was a logspace-uniform circuit, where the logspace machine constructing the circuit had higher space complexity than that of the hard function; this prohibits making the assumptions necessary to conclude worst-case derandomization.[16] To resolve this, we replace parts of their argument as follows. Instead of modeling the reconstruction as a logspace-uniform circuit, we model it as a probabilistic machine; and then, following [PRZ23], we show how to significantly reduce the randomness complexity of this machine, relying on a combination of derandomized D2P transformation with standard sampler-based techniques.

2.3 Derandomization Requires Targeted PRGs in Catalytic Logspace and in Logspace

Next, we focus on the question of whether solving $\mathbf{BPP}$-search problems in a certain class (specifically, in $\mathbf{CL}$ or in $\mathbf{L}$) reduces to solving the decision problem CAPP in that class. Equivalently, we ask whether derandomization in the class requires targeted PRGs. A straightforward search-to-decision reduction in [Gol11b] establishes this for $\mathbf{P}$, but it is highly space-inefficient, and thus unsuitable

[15] Each previous version has shortcomings making it unsuitable for the current purpose. Specifically, the targeted PRGs of [CT21a; CLO+23] are not evaluable in logspace (even if the hard function is computable in constant depth), and their reconstruction is probabilistic. The targeted PRG in [CTW23] is logspace-computable, but it is based on hardness in $\mathbf{TC}^0$, and its reconstruction is still probabilistic. The targeted PRG in [DPT24] is also based on hardness in $\mathbf{TC}^0$ rather than in $\mathbf{NC}^1$, and does not yield worst-case derandomization (as we explain below).

[16] Specifically, to deduce worst-case derandomization in their framework (following [CT21a]), we need to assume hardness on almost all inputs. If the hard function f is computable in space $c \cdot \log(n)$, and the machine in the reconstruction uses $C \cdot \log(n)$ space for some $C > c$, then the machine can hard-wire values of f (e.g., $f(1^n)$) into the circuit that it prints.

for **CL** and for **L**, where its existence is an open problem [PR23] (the reduction from [Gol11b] also fails for **AM**, due to other reasons; see [Gol11b; MS23a; MS23b]).

For concreteness, throughout this section let us assume that all circuits are in some fixed circuit class $\mathcal{C}$ that can be evaluated in logspace (e.g., $\mathcal{C} = \mathbf{NC}^1$).

Catalytic logspace. Recall the setting: We are given a circuit C, we can solve CAPP for C, and we want to construct a distribution $\mathbf{D}$ that is pseudorandom for C.[17]

Our result combines (a modification of) the result of [DPT24] reducing producing a targeted PRG for C to constructing a D2P transformation for C, with our instance-wise reduction from constructing a D2P transformation to solving CAPP. In more detail, we first modify the reduction of [DPT24], which uses the "compress or random" paradigm. They think of the catalytic tape $\mathbf{w}$ as a *hard truth table*, and instantiate a version of the Nisan-Wigderson generator with this truth table. Letting the generator be $\mathsf{NW}^{\mathbf{w}}$ (and note that it has seed length $O(\log n)$), either the generator is pseudorandom for C (in which case we can let $\mathbf{D}$ be its output set, and halt without modifying the tape), or the D2P transformation can be used to compress the tape $\mathbf{w}$, freeing up polynomially many bits on the tape. This enables us to use our (space inefficient) reduction from D2P to producing a targeted PRG (whereas their result used a time-efficient brute force derandomization).

The only missing piece is a D2P transformation in **CL** (assuming a CAPP algorithm in **CL**). Indeed, to obtain such an algorithm, we show that our reduction from D2P to CAPP can be implemented in catalytic logspace.

Remark 2.2. By combining this search-to-decision reduction with the D2P transformation for the path isolation lemma (and the main result of [BCK+14], which implies that the algorithm of [RA00] can be implemented in **CL**), we unconditionally obtain a **CL** algorithm that, given a graph G, outputs a weight assignment w such that (G, w) has unique shortest paths. This constitutes the first result for **CL** that is proved by *combining* the algebraic computation perspective (to evaluate the D2P transformation) with the compress-or-random perspective (to reduce search to D2P).

Logspace machines. Finally, let us briefly explain how to obtain our *conditional* result that derandomization of a class $\mathcal{C}$ in **L** necessitates targeted PRGs for $\mathcal{C}$ in **L** (again, we suggest thinking of $\mathcal{C} = \mathbf{NC}^1$ for concreteness). The main technical tool is the new version of the targeted PRG of [CT21a], which was described in Section 2.2.

Recall that our hardness assumption is a function $f : \{0,1\}^n \to \{0,1\}^n$ computable by n^C size $\mathbf{NC}^1$ circuits, that is hard for n^c time algorithms that use only $\mathrm{polylog}(n)$ many random coins, for $c < C$. Assuming that we have a CAPP algorithm for $\mathcal{C}$-circuits in **L**, we use the reduction of D2P to CAPP (from Section 2.1) to obtain a deterministic D2P for $\mathcal{C}$-circuits in **L**. The key observation is that this reduction is computable in logspace, and thus the D2P for $\mathcal{C}$ is a deterministic *logspace* algorithm (and hence is computable time n^c for some c).[18] Using this D2P, we instantiate our targeted PRG with this hard function. Supposing the generator is not pseudorandom for C, we can obtain a predictor for the generator using our deterministic logspace D2P transformation, and then compute the function quickly using only $\mathrm{polylog}(n)$ random coins and n^c time, contradicting the assumption (see Theorem 5.7 for further details).

[17] A mistaken intuition is that since **BPL** $\subseteq$ **CL**, we do not need a derandomization hypothesis. However, crucially, the derandomization hypothesis only applies to *decision* problems. Moreover, **BPL** $\subseteq$ **CL** only means that CAPP for ROBPs is in **CL**, and here we are concerned with richer circuit classes.

[18] Similarly to [DPT24], our targeted PRG construction uses D2P both when computing the generator and when computing the reconstruction (see the proof of Theorem 5.6 for details).

2.4 Certified Derandomization Using Hard Truth-Tables, and the Class LOSSY

Recall that certified derandomization using a property $\mathcal{P}$ refers to a deterministic algorithm that gets a linear-size circuit $C : \{0,1\}^n \to \{0,1\}$ and a string $\tau \in \{0,1\}^{\ell(n)}$, and either estimates the acceptance probability of C or provides a witness that $\tau \notin \mathcal{P}$ (see Definition 7.1). In Section 7 we prove that such certified derandomization is equivalent to **prBPP** = **prZPP** (see Theorem 1.15 and Theorem 7.4 for the formal statement).

We now focus on Theorem 1.17 that shows an equivalence between a restricted type of certified derandomization – namely, when $\mathcal{P}$ is the property of truth-tables that do not have small circuits (e.g., truth-tables of length 2^ℓ without circuits of size $2^{.01 \cdot \ell}$) – and a simulation of **prBPP** in the class **LOSSY** of problems reducible to LossyCode. Recall that in LossyCode (see Problem 1.16), we are given a pair of circuits $C : \{0,1\}^n \to \{0,1\}^m$ ("compression") and $D : \{0,1\}^m \to \{0,1\}^n$ ("decompression"), where $m < n$, and we want to find a string $x \in \{0,1\}^n$ such that $D(C(x)) \neq x$.

We first explain the direction ($\Rightarrow$), which is easier. If there is a certified derandomization algorithm A using hard truth tables, one can reduce an instance $C : \{0,1\}^n \to \{0,1\}$ of CAPP to the following instance of LossyCode: The compression circuit C' takes a candidate hard truth table τ, simulates the certified derandomization algorithm $A(C, \tau)$, and outputs a small circuit for τ if $A(C, \tau)$ fails to estimate $\mathbb{E}[C(\mathbf{U}_n)]$; the decompression circuit D' takes the description of a circuit and outputs its truth table. By definition, any solution τ to the LossyCode instance (C', D') is a truth-table such that A correctly estimates $\mathbb{E}[C(\mathbf{U}_n)]$.[19]

The other direction relies on an idea from [Kor21], which was implicit in earlier results in bounded arithmetic [Tha02; Jeř04; Jeř07] and cryptography [GGM86]. Korten [Kor21] proved that LossyCode can be efficiently reduced to the problem of finding truth tables of functions that are not computable by small circuits (for an explanation, see [ILW23, Appendix C]). The key observation leading to our results is that the reduction of LossyCode to finding hard truth-table can be thought of as a certified reduction: It either solves the LossyCode instances, or produces a certificate that the truth-table is not hard (in the form of a small circuit).[20]

Assume that **prBPP** = **prLOSSY**. Then, there is a polynomial-time algorithm M for CAPP with a LossyCode oracle. Our certified derandomization algorithm works as follows: Given a circuit C and a supposedly hard truth table τ, it simulates the algorithm $M(x)$, and attempts to answer the LossyCode oracle calls using the truth table τ and the certified reduction from LossyCode to finding hard truth tables. The certified reduction either solves the LossyCode oracle calls, in which case we can keep simulating $M(x)$, or prints a small circuit for the truth table τ. Therefore, our algorithm either prints a small circuit for τ, or successfully simulates $M(x)$; in the latter case, it will solve $\mathsf{CAPP}(C)$ by the correctness of $M(x)$.

3 Preliminaries

Throughout the paper we fix a machine model (e.g., the RAM model), except for places where it is explicitly mentioned otherwise (i.e., except for the models of logspace machines and catalytic logspace machines). We mention this point because in several places we are concerned with fine-grained time bounds. For simplicity, we assume throughout the paper that parameters defining algorithms are time constructive.

[19] Indeed, this implication does not require that that A will use hard truth-tables, and a more general notion of a "range avoidance" property suffices; see Theorem 7.7 for details.

[20] This was also observed (although phrased in a different context) in the literature of bounded arithmetic, see, e.g., [Tha02, Lemma 3.7].

3.1 Standard Notation

Distributions. Throughout the paper, we will usually denote distributions by boldface. In particular, we use $\mathbf{U}_n$ to denote the uniform distribution over $\{0,1\}^n$, where n may be omitted if it is implicit in the context. Generally, we let $\mathbf{U}_S$ be the uniform distribution over the set S.

We will usually think of distributions as being uniform over a multiset. Accordingly, we may abuse the notation to identify a multiset and the uniform distribution over the multiset. (For instance, we may say "given a distribution $\mathbf{D}$ to an algorithm" meaning that it is given a multiset and we are referring to the uniform distribution over the multiset.) Similarly, we define the size of a distribution as the size of the corresponding multiset.

Predictors and prefixes. For $x \in \{0,1\}^n$, we define $x_{<i} \triangleq x_{1,\ldots,i-1}$, $x_{\leq i} \triangleq x_{1,\ldots,i}$, $x_{>i} \triangleq x_{i+1,\ldots,n}$, and $x_{\geq i} \triangleq x_{i,\ldots,n}$. We write $\mathcal{P} : \{0,1\}^{\leq n} \to \{0,1\}$ to denote a class of functions where for every $P \in \mathcal{P}$, we have $P : \{0,1\}^i \to \{0,1\}$ for $i \leq n$.

Definition 3.1 (correlation)**.** For vectors $x, y \in \{0,1\}^n$, we define the relative correlation between x and y as

$$\mathrm{Cor}(x,y) \triangleq \frac{1}{2n}\sum_{i=1}^{n}(-1)^{x_i \oplus y_i} \in \left(-\frac{1}{2}, \frac{1}{2}\right).$$

Note that

$$\mathrm{Cor}(x,y) = \frac{1}{2}\left(\Pr_{i \leftarrow \mathbf{U}_n}[x_i = y_i] - \Pr_{i \leftarrow \mathbf{U}_n}[x_i \neq y_i]\right) = \Pr_{i \leftarrow \mathbf{U}_n}[x_i = y_i] - \frac{1}{2}.$$

Definition 3.2 (prediction advantage)**.** For a distribution $\mathbf{D}$ over $\{0,1\}^n$ and a predictor $P : \{0,1\}^{i-1} \to \{0,1\}$ for the i-th bit of $\mathbf{D}$, where $i \leq n$, we define its advantage as

$$\mathrm{adv}_{\mathbf{D}}(P) \triangleq \Pr_{x \leftarrow \mathbf{D}}[P(x_{<i}) = x_i] - \frac{1}{2}.$$

Fact 3.3. Let $\mathbf{D} = \{x^1, x^2, \ldots, x^m\}$ be a multiset over $\{0,1\}^n$. Fix $i \in [n]$, and let $v^i \in \{0,1\}^m$ be defined as $v^i_j \triangleq x^j_i$, and $p^i \in \{0,1\}^m$ be defined as $p^i_j \triangleq P(x^j_{<i})$. Then $\mathrm{adv}_{\mathbf{D}}(P) = \mathrm{Cor}(v^i, p^i)$.

Definition 3.4 (predictor)**.** We say that P is a α-predictor for $\mathbf{D}$ if $\mathrm{adv}_{\mathbf{D}}(P) \geq \alpha$.

3.2 Time bounds, Circuit classes, and Notions in Pseudorandomness

Throughout the paper, we assume that all time bounds are time constructive, and all space bounds are space constructive.

Promise problems. A promise problem Π is defined by a pair of disjoint sets $\Pi = (\Pi^{\mathsf{YES}}, \Pi^{\mathsf{NO}})$, where $\Pi^{\mathsf{YES}} \subseteq \{0,1\}^*$ is the set of YES-instances and $\Pi^{\mathsf{NO}} \subseteq \{0,1\}^*$ is the set of NO-instances. A string is said to be an instance in the promise (of Π) if it is in $\Pi^{\mathsf{YES}} \cup \Pi^{\mathsf{NO}}$. An algorithm is said to decide Π if it accepts every YES-instance and rejects every NO-instance.

Circuit classes. For a circuit C (from some circuit family $\mathcal{C}$), the size of C, denoted $\mathrm{size}(C)$, is defined as the number of gates in C.

We say that a circuit class $\mathcal{C}$ is typical if it can be evaluated in polynomial time, and it is closed under negation, Boolean AND, and variable projection. That is, for all $\mathcal{C}$ circuits $C(x), D(x)$ of size s and every $b \in \{0,1\}$, there are $\mathcal{C}$ circuits of size s computing

$$\neg C(x) \quad C(x_1, \ldots, x_{i-1}, x_i \oplus b, x_{i+1}, \ldots, x_n) \quad C(x_1, \ldots, x_{i-1}, b, x_{i+1}, \ldots, x_n),$$

and a circuit of size at most $\mathrm{poly}(s)$ computing $C(x) \wedge D(x)$. Moreover, these circuits can be generated in polynomial time given the description of C (and D).

For a circuit class $\mathcal{C}$, we use $\mathcal{C}_n$ to denote the set of circuits with n input gates.

Let $\mathcal{C}$ and $\mathcal{C}'$ be two circuit classes, $\mathcal{C} \circ \mathcal{C}'$ is a circuit class consisting of circuits with a top $\mathcal{C}$ circuit whose input gates are connected to $\mathcal{C}'$ circuits.

Distinguishers and hitting-sets. For a distribution $\mathbf{D}$ over $\{0,1\}^n$ and a circuit $C : \{0,1\}^n \to \{0,1\}$, we say that $\mathbf{D}$ ε-fools C if and only if $|\mathbb{E}[C(\mathbf{D})] - \mathbb{E}[C(\mathbf{U}_n)]| \leq \varepsilon$.

For a class of circuits $\mathcal{C}$, we say that a family of subsets $\{H_n \subseteq \{0,1\}^n\}_{n\in\mathbb{N}}$ is a ε-hitting set for $\mathcal{C}$ if for every $C \in \mathcal{C}$ on n inputs with $\mathbb{E}[C(\mathbf{U})] \geq \varepsilon$, there is $y \in H_n$ such that $C(y) = 1$. We say that a family of sets $\{S_n \subseteq \{0,1\}^n\}_{n\in\mathbb{N}}$ is explicit if there is a deterministic $\mathrm{poly}(|S_n|)$ time algorithm that on input 1^n outputs S_n. (We will typically consider explicit families of hitting-sets.)

Definition 3.5. We define the CAPP problem as the promise problem where Π^{YES} (resp. Π^{NO}) is the set of linear-size circuits $C : \{0,1\}^* \to \{0,1\}$ with $\mathbb{E}[C(\mathbf{U}_n)] \geq 2/3$ (resp. $\mathbb{E}[C(\mathbf{U}_n)] \leq 1/3$).

Note that CAPP is known to be **prBPP**-hard, i.e., every problem in **prBPP** can be (deterministically) polynomial-time reduced to CAPP; in particular, **prBPP** = **prP** if and only if CAPP $\in$ **prP** (see [Gol08, Exercise 6.14]). We further introduce the prefix-CAPP problem, which is a special case of CAPP that appears as an intermediate problem in our proofs (see Section 4 and Section 5).

Definition 3.6. We say a machine E is an ε-prefix-CAPP (PCAPP) algorithm for $C : \{0,1\}^n \to \{0,1\}$ if for every $x \in \{0,1\}^{\leq n}$ we have

$$\left| E(C, x) - \mathop{\mathbb{E}}_{r}[C(x \circ r)] \right| \leq \varepsilon.$$

Observe that if a circuit class is closed under restrictions, PCAPP reduces to CAPP.

3.3 D2P Transformations and Yao's Lemma

We formally define D2P transformations, both for fixed and arbitrary size.

Definition 3.7. For a circuit $C : \{0,1\}^n \to \{0,1\}$, we say a collection of circuits $\mathcal{P} : \{0,1\}^{<n} \to \{0,1\}$ is an (m, α, δ)-distinguish to predict (D2P) transformation for C if the following holds. For every distribution $\mathbf{D}$ of size at most m that does not δ-fool C, there is $P \in \mathcal{P}$ such that P is an α-predictor for $\mathbf{D}$. We will also refer to this notion as a δ-distinguish to α-predict transformation against m-size distributions.

When $m = \infty$ (i.e., if this is an (m, α, δ)-D2P for C for every m), we may implicitly drop the size parameter; and we may also drop δ if $\delta = 1/3$.

We recall the statement of Yao's lemma that we will actually use:

Lemma 3.8 ([Yao82])**.** *Let $C : \{0,1\}^n \to \{0,1\}$ be an arbitrary function and $\mathbf{D}$ be an arbitrary distribution that does not ε-fool C. Then there is $i \in [n]$ and $\sigma \in \{0,1\}^2$ such that, letting $P_{z,\sigma,i}$ be defined as $P_{z,\sigma,i}(x_{<i}) = C(x_{<i} \circ \sigma_1 \circ z) \oplus \sigma_2$, we have*

$$\mathop{\mathbb{E}}_{z \leftarrow \mathbf{U}_{n-i}}[\mathrm{adv}_{\mathbf{D}}(P_{z,\sigma,i})] \geq \frac{\varepsilon}{n}.$$

Moreover (by the reverse Markov's inequality), on at least a $2\varepsilon/(3n-\varepsilon)$ fraction of $z \in \{0,1\}^{n-i}$, we have $\mathrm{adv}_{\mathbf{D}}(P_{z,\sigma,i}) \geq \varepsilon/3n$.

3.4 Concentration Bounds

We recall the following two standard concentration bounds.

Theorem 3.9 (Chernoff bound, see [Vad12] Theorem 2.21)**.** *Let $X_1, \ldots, X_n$ be independent random variables taking values in $\{0,1\}$. Let $X \triangleq X_1 + \cdots + X_n$ and $\mu \triangleq \mathbb{E}[X]$. For any $\delta > 0$, $\Pr[|X - \mu| \geq \delta\mu] \leq 2\exp(-\delta^2\mu/4)$.*

Theorem 3.10 (Hoeffding's inequality [Hoe63])**.** *Let $X_1, \ldots, X_n$ be independent random variables such that $X_i \in [a_i, b_i]$, where $a_i, b_i \in \mathbb{R}$ for every $i \in [n]$. Let $X \triangleq X_1 + \cdots + X_n$ and $\mu \triangleq \mathbb{E}[X]$. Then for $t \geq 0$:*

$$\Pr\left[|X - \mu| \geq t\right] \leq 2\exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right).$$

4 Distinguish to Predict is Equivalent to Derandomization

We will prove Theorem 1.5 that distinguish-to-predict transforms and derandomization of **prBPP** are equivalent. We formally state this result (recall that $\delta = 1/3$ is dropped for D2P by default):

Theorem 4.1. *The following are equivalent:*

(1) $\mathbf{prBPP} = \mathbf{prP}$.

(2) There is $\alpha = n^{-\Omega(1)}$ and a deterministic $\mathrm{poly}(n)$-time $(1/3)$-distinguish to α-predict transformation against m-size distributions for general circuits, where $m \geq n/\alpha^2$.

(3) There is a deterministic $\mathrm{poly}(n)$-time $(1/3)$-distinguish to $(1/10n)$-predict transformation for general circuits.

In addition to the informal statement in the introduction (see Theorem 1.5), Theorem 4.1 further shows that a deterministic D2P transformation that only works for distributions of fixed size (i.e. (3)) suffices to imply $\mathbf{prBPP} = \mathbf{prP}$. Note that (3) $\Rightarrow$ (2) is trivial, so it suffices to show that (2) $\Rightarrow$ (1) and (1) $\Rightarrow$ (3).

The equivalence of Theorem 4.1 is not completely general (e.g., it does not scale down to the setting of logspace algorithms or **CL**, as mentioned after Theorem 1.12[21]), but it nevertheless applies to many weak circuit classes. For example, we show that even constructing non-trivial algorithms for D2P (i.e., just barely improving on the brute-force algorithm) of classes of constant-depth circuits implies non-trivial CAPP algorithms (and lower bounds) for these circuit classes (see Appendix B.2).

Furthermore, as mentioned in Section 2.1, D2P algorithms for a *fixed circuit* imply a polynomial-time derandomization of that specific circuit. (We crucially use this "instance-wise" connection when proving the results in Section 5; see that section for details.)

[21] This is because the derandomization algorithm that we obtain from D2P is space-inefficient.

4.1 Distinguish to Predict Implies Derandomization

Our derandomization algorithm is instance-wise, in that it is given an arbitrary circuit C and a (valid) D2P transformation for C, and will succeed for the specific circuit C.

Lemma 4.2. *There is a deterministic algorithm $\mathcal{D}$ that, given a δ-distinguish to α-predict transformation $\mathcal{P} = (P_1, \ldots, P_t)$ against m-size distributions for $C : \{0,1\}^n \to \{0,1\}$, where $m \geq n/\alpha^2$ and $t \leq 2^{n/5}$, works as follows.*

- *$\mathcal{D}$ produces a distribution $\mathbf{D}$ of size n/α^2 such that for every $i \in [t]$, $\mathrm{adv}_{\mathbf{D}}(P_i) < \alpha$. In particular, by the security of the D2P transformation we have that $\mathbf{D}$ δ-fools C.*
- *$\mathcal{D}$ runs in time $\tilde{O}(Sn/\alpha^2) + \mathrm{poly}(tn/\alpha)$, where $S = \sum_i \mathrm{size}(P_i)$.*

Our proof follows the approach of Goldreich and Wigderson [GW00], where we iteratively construct $\mathbf{D}$ by extending strings, and at each step we choose the next one-bit suffix to ensure no predictor obtains good correlation. As in [GW00], we reduce finding a good suffix to constructing a vector with a low correlation with t fixed vectors. As the dimension of the vector is large enough, a random vector achieves this whp, so it suffices to derandomize this process.

Remark 4.3. Goldreich and Wigderson [GW00] derandomize finding such a vector using a pairwise independent sample space. Diagonalizing against m predictors in this fashion results in distribution of size $O(m^2)$, which suffices to prove (3) $\Rightarrow$ (1) in Theorem 4.1. To prove a stronger implication (2) $\Rightarrow$ (1), i.e., a collection of m predictors secure against distributions of size $m^{\varepsilon} \ll m$ still implies derandomization, we need to use a different approach via a result of Sivkumar [Siv02] that takes the advantage of an explicit access to the predictors.

We recall the result of Sivkumar [Siv02], which allows fooling a set of read-once branching programs that are explicitly given.

Theorem 4.4 ([Siv02])**.** *There is a polynomial time algorithm that, given t read-once branching programs $B_1, \ldots, B_t : \{0,1\}^m \to \{0,1\}$ where $\mathbb{E}[B_i(\mathbf{U}_m)] \geq 1 - 1/t^2$ for every $i \in [t]$, outputs $z \in \{0,1\}^m$ such that $B_i(z) = 1$ for every $i \in [t]$.*

We show that producing such a vector can itself be reduced to this problem.

Lemma 4.5. *There is a deterministic polynomial time algorithm that, given $\alpha > 0$ and $(v_1, \ldots, v_t) \in \{0,1\}^m$ where $\log(t) \leq m\alpha^2/5$, outputs $z \in \{0,1\}^m$ such that for every $i \in [t]$ it holds that $\mathrm{Cor}(z, v_i) \in (-\alpha, \alpha)$.*

Proof. For every $i \in [t]$, let $B_i : \{0,1\}^m \to \{0,1\}$ be defined as $B_i(z) = \mathbb{I}[\mathrm{Cor}(v_i, z) \in (-\alpha, \alpha)]$. Observe that B_i can be computed by a read-once branching program of width $\mathrm{poly}(m)$. By Hoeffding's inequality (see theorem 3.10), we can see that

$$\begin{aligned}\mathbb{E}[B_i(\mathbf{U}_m)] &= \Pr_{z \leftarrow \mathbf{U}_m}[\mathbb{I}[\mathrm{Cor}(v_i, z) \in (-\alpha, \alpha)] = 1] \\ &\geq 1 - 2\exp(-2m\alpha^2) \geq 1 - 1/t^2.\end{aligned}$$

Then we can apply Theorem 4.4 and obtain a mutually satisfying vector $z \in \{0,1\}^m$. The running time is $\mathrm{poly}(tm)$ as claimed. □

We can then prove the result:

Proof of Lemma 4.2. Without loss of generality let $m = n/\alpha^2$ and set m to be the target size of $\mathbf{D}$ (and note that the D2P transformation is secure against distributions of this size by definition). We construct $\mathbf{D}$ bit-by-bit, iterating $i = 1, \ldots, n$. In stage i, let

$$\mathbf{D}^i = (x_1^i, \ldots, x_m^i), \quad \text{where } x_j^i \in \{0,1\}^{i-1}$$

(where $\mathbf{D}^1$ is simply m copies of the empty string). Next let $(Q_1, \ldots, Q_r) = \mathcal{P}^i$ be the subset of predictors in $\mathcal{P}$ that read the first $i-1$ bits and attempt to predict the i-th bit (and note that $r \le t$), and let $(v^1, \ldots, v^r)$ be the predictions made by each predictor on the current distribution, i.e.

$$(v^l)_j = Q_l(x_j^i). \tag{1}$$

We call the algorithm of Lemma 4.5 with $(v^1, \ldots, v^r)$ and with the parameter α. Observe that we have $\log(r) \le \log(t) \le m/5\alpha^2$ by assumption on the size of t and the D2P parameter m, so we satisfy the preconditions. Thus we obtain in time $\mathrm{poly}(tm)$ a vector $z \in \{0,1\}^m$. Then letting $\mathbf{D}_{i+1}$ be the distribution

$$D_{i+1} = (x_1^i \circ z_1, \ldots, x_m^i \circ z_m).$$

Observe that by the property of z, we have for every $j \in [r]$,

$$(-\alpha, \alpha) \ni \mathrm{Cor}(z, v^j) = \mathrm{adv}_{\mathbf{D}^{i+1}}(Q_l), \tag{Fact 3.3}$$

i.e. every $P \in \mathcal{P}^i$ does not predict $\mathbf{D}^{i+1}$ with advantage α. Since fixing future bits does not change the advantage of previous predictors, $\mathbf{D}^{i+1}$ remains unpredictable to $\mathcal{P}^j$ for $j \le i$, and so by induction after n steps we are done.

It remains to verify the time bound. To construct the vectors v^l defined in (1), we need to run each predictor $P_1, \ldots, P_t$ exactly m times, which takes time $\tilde{O}(Sn/\alpha^2)$. The algorithm of Lemma 4.5 runs in time $\mathrm{poly}(tm) = \mathrm{poly}(tn/\alpha)$ and is called for n times. The overall running time is then bounded as claimed. □

4.2 Derandomization Implies Distinguish to Predict

We now state the lemma showing that D2P transformation is implied by derandomization. Recall that δ-PCAPP is the problem where we are given input $C : \{0,1\}^n \to \{0,1\}$ and $x \in \{0,1\}^k$, and we want to solve CAPP on the circuit C with the first k bits fixed to be x (see Definition 3.6). Note that δ-PCAPP reduces to CAPP for any circuit class close under restrictions.

Lemma 4.6 (derandomization implies D2P)**.** *Fix $\delta > 0$ and an arbitrary circuit $C : \{0,1\}^n \to \{0,1\}$ and function E that is a $(\delta/3n)$-PCAPP algorithm for C. Let $\alpha = \delta/3n$ and $A = \lceil 1/\alpha \rceil$. Then the following is a δ-distinguish to α-predict transformation for C:*

$$\mathcal{P} \stackrel{def}{=} \{P_{\tau,\sigma,i} : i \in [n], \sigma \in \{0,1\}^2, \tau \in \{0, \ldots, A\}\}$$

where

$$P_{\tau,\sigma,i}(x_{<i}) = \mathbb{I}\left[\frac{\tau}{A} \le E(C, x_{<i} \circ \sigma_1)\right] \oplus \sigma_2.$$

Note that the number of predictors is at most $n \cdot 4 \cdot (3n/\delta + 1) = O(n^2/\delta)$.

Proof. In words, $P_{\tau,\sigma,i}$ uses the prefix-CAPP algorithm to estimate $p_{x_{<i} \circ \sigma_1} = \mathbb{E}_r[C(x_{<i} \circ \sigma_1 \circ r)]$, checks if this value is greater than the threshold τ/A, and returns $\neg\sigma_2$ or σ_2 accordingly.

Security of the transformation. Next, we claim that $\mathcal{P} = \{P_{\tau,\sigma,i}\}$ is a valid δ-distinguish to α-predict transformation. To show this, fix a distribution $\mathbf{D}$ over $\{0,1\}^n$ (of arbitrary size) such that

$$|\,\mathbb{E}[C(\mathbf{D})] - \mathbb{E}[C(\mathbf{U}_n)]| > \delta.$$

By Lemma 3.8, there exist $i \in [n]$ and $\sigma \in \{0,1\}^2$ such that the following holds:

$$\mathop{\mathbb{E}}_{z \leftarrow \mathbf{U}_{n-i}} [\mathrm{adv}_{\mathbf{D}}(P_{z,\sigma,i})] > \frac{\delta}{n}.$$

(recall that $P_{z,\sigma,i}$ is defined as $P_{z,\sigma,i}(x) = C(x_{<i} \circ \sigma_1 \circ z) \oplus \sigma_2$). For the remainder of the proof, fix this choice of i and σ.

Next, we claim that the expected advantage of $P_{\tau,\sigma,i}$ over τ is approximately the same as the expected advantage of the randomzied Yao predictor with this same i and σ. Formally:

Claim 4.7. *We have that*

$$\left| \mathop{\mathbb{E}}_{z \leftarrow \mathbf{U}_{n-i}} [\mathrm{adv}_{\mathbf{D}}(P_{z,\sigma,i})] - \mathop{\mathbb{E}}_{\tau \leftarrow \mathbf{U}_{\{0..A\}}} [\mathrm{adv}_{\mathbf{D}}(P_{\tau,\sigma,i})] \right| \leq \frac{2\delta}{3n}.$$

Intuitively, using a random suffix to approximate the expectation is equivalent to computing the expectation over a random suffix, then randomly rounding. As expectation is linear, such a rounding can be coupled for all prefixes.

Proof of Claim 4.7. Notice that

$$\begin{aligned}
\mathop{\mathbb{E}}_{z \leftarrow \mathbf{U}_{n-i}} [\mathrm{adv}_{\mathbf{D}}(P_{z,\sigma,i})] &= \mathop{\mathbb{E}}_{z} \left[\mathop{\mathbb{E}}_{x \leftarrow \mathbf{D}} [\mathbb{I}\,[C(x_{<i} \circ \sigma_1 \circ z) \oplus \sigma_2 = x_i]] - \frac{1}{2} \right] \\
&= \mathop{\mathbb{E}}_{x \leftarrow \mathbf{D}} \left[\mathop{\mathbb{E}}_{z} [\mathbb{I}\,[C(x_{<i} \circ \sigma_1 \circ z) \oplus \sigma_2 = x_i]] \right] - \frac{1}{2} \\
&= \mathop{\mathbb{E}}_{x \leftarrow \mathbf{D}} \left[\mathop{\mathbb{E}}_{z} [C(x_{<i} \circ \sigma_1 \circ z)] \cdot \mathbb{I}\,[\sigma_2 \neq x_i] + (1 - \mathop{\mathbb{E}}_{z} [C(x_{<i} \circ \sigma_1 \circ z)]) \mathbb{I}\,[\sigma_2 = x_i] \right] - \frac{1}{2} \\
&= \mathop{\mathbb{E}}_{x \leftarrow \mathbf{D}} \left[\mathop{\mathbb{E}}_{z} [C(x_{<i} \circ \sigma_1 \circ z)] \cdot a_x + b_x \right] - \frac{1}{2}
\end{aligned}$$

where

$$a_x = (-1)^{\mathbb{I}[\sigma_2 = x_i]} \in \{\pm 1\}, \qquad b_x = \mathbb{I}\,[\sigma_2 = x_i] \in \{0,1\}$$

are constants that depend on x and σ_2. We then compute the expected advantage of the second distribution.

$$\begin{aligned}
\mathop{\mathbb{E}}_{\tau} [\mathrm{adv}_{\mathbf{D}}(P_{\tau,\sigma,i})] &= \mathop{\mathbb{E}}_{\tau} \left[\mathop{\mathbb{E}}_{x \leftarrow \mathbf{D}} \left[\mathbb{I}\left[\mathbb{I}\left[\frac{\tau}{A} \leq E(C, x_{<i} \circ \sigma_1) \right] \oplus \sigma_2 = x_i \right] - \frac{1}{2} \right] \right] \\
&= \mathop{\mathbb{E}}_{x \leftarrow \mathbf{D}} \left[\mathop{\mathbb{E}}_{\tau} \left[\mathbb{I}\left[\frac{\tau}{A} \leq E(C, x_{<i} \circ \sigma_1) \right] \right] \cdot a_x + b_x \right] - \frac{1}{2}
\end{aligned}$$

Finally, note that for every $x \in \{0,1\}^n$, we have that

$$\begin{aligned}
&\left| \mathop{\mathbb{E}}_{z} [C(x_{<i} \circ \sigma_1 \circ z)] - \mathop{\mathbb{E}}_{\tau} \left[\mathbb{I}\left[\frac{\tau}{A} \leq E(C, x_{<i} \circ \sigma_1) \right] \right] \right| \\
&\quad \leq \left| \mathop{\mathbb{E}}_{z} [C(x_{<i} \circ \sigma_1 \circ z)] - E(C, x_{<i} \circ \sigma_1) \right| + \left| E(C, x_{<i} \circ \sigma_1) - \mathop{\mathbb{E}}_{\tau} \left[\mathbb{I}\left[\frac{\tau}{A} \leq E(C, x_{<i} \circ \sigma_1) \right] \right] \right| \\
&\quad \leq \left| \mathop{\mathbb{E}}_{z} [C(x_{<i} \circ \sigma_1 \circ z)] - E(C, x_{<i} \circ \sigma_1) \right| + \frac{1}{A} \qquad \text{(discretization precision of } \tau\text{)} \\
&\quad \leq \frac{2}{A}
\end{aligned}$$

where the last inequality holds from the guarantee that E is a $(1/A)$-PCAPP algorithm. Thus,

$$\begin{aligned}
&\left|\mathop{\mathbb{E}}_{z}\left[\mathrm{adv}_{\mathbf{D}}(P_{z,\sigma,i})\right] - \mathop{\mathbb{E}}_{\tau}[\mathrm{adv}_{\mathbf{D}}(P_{\tau,\sigma,i})]\right| \\
&\quad= \left|\mathop{\mathbb{E}}_{x\leftarrow\mathbf{D}}\left[\mathop{\mathbb{E}}_{z}[C(x_{<i}\circ\sigma_1\circ z)]\cdot a_x - \mathop{\mathbb{E}}_{\tau}\left[\mathbb{I}\left[\frac{\tau}{A}\leq E(C, x_{<i}\circ\sigma_1)\right]\right]\cdot a_x\right]\right| \\
&\quad\leq \max_{x\in\mathbf{D}}\left\{|a_x|\cdot\left|\mathop{\mathbb{E}}_{z}[C(x_{<i}\circ\sigma_1\circ z)] - \mathop{\mathbb{E}}_{\tau}\left[\mathbb{I}\left[\frac{\tau}{A}\leq E(C, x_{<i}\circ\sigma_1)\right]\right]\right|\right\} \\
&\quad\leq \frac{2}{A} = \frac{2\delta}{3n}.
\end{aligned}$$

□

Therefore, we have that $\mathbb{E}_\tau[\mathrm{adv}_{\mathbf{D}}(P_{\tau,\sigma,i})] \geq \delta/n - 2/A \geq \alpha$. As at least one element in a distribution achieves its expectation, we have that there exists τ such that $P_{\tau,\sigma,i}$ predicts $\mathbf{D}$ with advantage α, i.e. $\mathcal{P}$ is a valid D2P transformation for C. □

The reader might have noticed that the structure of the proof above is not identical to the one outlined in Section 2.1. However, both proofs are essentially the same technically. For completeness, we now include the only missing claim from the description that was presented in Section 2.1:

Fact 4.8. Let $\mathbf{x}$ and $\mathbf{y}$ be random variables taking values in $[0,1]$ and $[-1,1]$, respectively, such that $\mathbb{E}[\mathbf{x}\cdot\mathbf{y}]\geq\delta$. Then, there exists $\tau\in[1/\varepsilon]$ such that $\mathbb{E}[\mathbf{1}_{\mathbf{x}\leq\varepsilon\cdot\tau}\cdot\mathbf{y}]\geq\delta-\varepsilon$.

Proof. Let $\ell_1 = 0$ and $h_1 = \varepsilon$ and $I_1 = [\ell_1, h_1]$. For every $\tau\in\{2,...,1/\varepsilon\}$, let $\ell_\tau = (\tau-1)\cdot\varepsilon$, let $h_\tau = \tau\cdot\varepsilon$, and let $I_\tau = (\ell_\tau, h_\tau]$. Then, since $\mathbb{E}[\mathbf{x}\cdot\mathbf{y}]\geq\delta$, we have that

$$\begin{aligned}
\delta-\varepsilon &\geq \sum_{\tau\in[1/\varepsilon]}\mathbb{E}[\mathbf{1}_{\mathbf{x}\in I_\tau}\cdot h_\tau\cdot\mathbf{y}] \\
&= \mathop{\mathbb{E}}_{\tau\in[1/\varepsilon]}[\tau\cdot\mathbb{E}[\mathbf{1}_{\mathbf{x}\in I_\tau}\cdot\mathbf{y}]] \\
&= \mathop{\mathbb{E}}_{\tau\in[1/\varepsilon]}[\mathbb{E}[\mathbf{1}_{\mathbf{x}\leq h_\tau}\cdot\mathbf{y}]] \quad ,
\end{aligned}$$

and hence there exists $\tau\in[1/\varepsilon]$ such that $\mathbb{E}[\mathbf{1}_{\mathbf{x}\leq h_\tau}\cdot\mathbf{y}]\geq\delta-\varepsilon$. □

4.3 Putting it all Together

We now recall the main equivalence and complete the proof.

Theorem 4.1. *The following are equivalent:*

(1) $\mathsf{prBPP} = \mathsf{prP}$.

(2) There is $\alpha = n^{-\Omega(1)}$ *and a deterministic* $\mathrm{poly}(n)$*-time* $(1/3)$*-distinguish to* α*-predict transformation against* m*-size distributions for general circuits, where* $m \geq n/\alpha^2$.

(3) There is a deterministic $\mathrm{poly}(n)$*-time* $(1/3)$*-distinguish to* $(1/10n)$*-predict transformation for general circuits.*

Proof. (1) ⇒ (3). As $\mathsf{prBPP} = \mathsf{P}$, there is a deterministic polynomial-time machine $\mathcal{D}$ that, given a circuit $C:\{0,1\}^n\to\{0,1\}$ of size n, outputs ρ such that $|\rho-\mathbb{E}[C]|\leq 1/10n$. Then by transforming an input (C,x) to the circuit $D(r) = C(x_<\circ r)$ and applying the CAPP algorithm to this circuit (of size $O(n)$), there is an explicit $(1/10n)$-PCAPP algorithm for general circuits of linear size.

Then applying Lemma 4.6, we deduce that there is a deterministic D2P transformation for general circuits of linear size with the claimed parameters.

(3) $\Rightarrow$ (2) Trivial.

(2) $\Rightarrow$ (1). Recall that CAPP is $\mathbf{prBPP}$-complete, so it suffices to distinguish whether $\mathbb{E}[C(\mathbf{U}_n)] \geq 0.9$ or $\mathbb{E}[C(\mathbf{U}_n)] \leq 0.1$ for $C : \{0,1\}^n \to \{0,1\}$ of size n.

Given such a circuit, we use our algorithm to construct a $(1/3)$-distinguish to α-predict transform against m-size distributions for C, denoted $\mathcal{P} = (P_1, \ldots, P_t)$, where $m \geq n/\alpha^2$ and $\alpha = n^{-\Omega(1)}$. As this transformation runs in polynomial time by assumption, we have $t = \mathrm{poly}(n)$. Then we call the algorithm of Lemma 4.2 on $\mathcal{P}$ with parameter values α and m and observe that we satisfy the precondition. Once we obtain the distribution $\mathbf{D}$ in polynomial time, by the postcondition we have that $\mathbf{D}$ $(1/3)$-fools C, and hence we can use it to estimate the expectation to the required accuracy. □

Remark 4.9. As mentioned in Section 1.4, we also study more restricted notions of derandomizing Yao's transformation in Appendix C. Instead of generating a family of candidate predictors as in a D2P transformation, we require a deterministic algorithm to generate a good suffix z for the predictors $\{P_{z,\sigma,i}\}$ in Yao's lemma (see Lemma 3.8), in two settings.

The first setting is *black-box* derandomization of Yao's transformation, where the deterministic algorithm generates a suffix z that works for *all* distributions of fixed size (see Appendix C.1). And the second setting is *non-black-box* derandomization, where the algorithm generates a suffix z for a *given* distribution (see Appendix C.2). We prove that the former notion is equivalent to the existence of hitting sets (which are, in turn, equivalent to circuit lower bounds), whereas the latter notion is closely related to the collapse $\mathbf{prBPP} = \mathbf{prLOSSY}$, where $\mathbf{prLOSSY}$ is a natural subclass of $\mathbf{prZPP}$ that will be introduced in Section 7 (see Remark 7.8).

5 Targeted PRGs in Logspace and Catalytic Logspace

In this section, we prove our result that derandomization unconditionally implies targeted PRGs in $\mathbf{CL}$ (Section 5.1) and apply this to unconditionally derandomize the isolation lemma for graphs in $\mathbf{CL}$, and also prove a conditional analogous statement for $\mathbf{L}$ whose assumptions are weaker than ones previously known to imply that (Section 5.2).

5.1 Targeted PRGs in Catalytic Logspace

Before stating the result, we first recall the formal definition of catalytic computation:

Definition 5.1 (Catalytic Turing Machine [BCK+14])**.** A Turing machine $\mathcal{M}$ is a catalytic machine using time $T(n)$, workspace $S(n)$, and catalytic space $W(n)$ if it has a work tape, a read-only input tape, a write-only output tape, and a read/write catalytic tape $\mathbf{w} \in \{0,1\}^{W(n)}$. We require that for every input x with $|x| = n$ and every $\mathbf{w}$, $\mathcal{M}^{\mathbf{w}}(x)$ (where the oracle notation denotes the catalytic tape) halts in time at most $T(n)$, using at most $S(n)$ cells on the worktape. Moreover, the final configuration of $\mathbf{w}$ must be equal to its initial configuration, for every x and $\mathbf{w}$.

Definition 5.2. Let $\mathbf{CTISP}\,[T(n), S(n), W(n)]$ be the set of languages recognized by catalytic machines that use $O(S(n))$ workspace and $O(W(n))$ catalytic space on inputs of size n, and run in time $O(T(n))$ in the worst case. Let $\mathbf{CL} = \cup_c \mathbf{CTISP}\left[2^{n^c}, c\log n, n^c\right]$ be catalytic logspace, and let $\mathbf{CLP} = \cup_c \mathbf{CTISP}\,[n^c, c\log n, n^c]$ be catalytic logspace where the worst-case runtime is bounded by a polynomial.

Following [Pyn24], we define a notion of search problems in **CL**. We note that the solution output by the search algorithm can differ based on the initial catalytic tape.

Definition 5.3 ((total) search-**CL**)**.** Fix a total $R \subseteq \{0,1\}^* \times \{0,1\}^*$.[22] We say the search problem defined by f is computable in **CL** if there is a catalytic machine $\mathcal{M}$ using workspace $O(\log n)$ and catalytic space $\mathrm{poly}(n)$ such that for every catalytic tape $\mathbf{w}$ and $x \in \{0,1\}^*$, $\mathcal{M}^{\mathbf{w}}(x)$ writes some y such that $(x,y) \in R$ to the write-only output tape, and resets $\mathbf{w}$ to its original configuration.

We now formally state our result that derandomization requires targeted PRGs in **CL**:

Theorem 5.4. *For an arbitrary class of circuits $\mathcal{C}$ that is evaluable in* **CL** *(resp.* **CLP***), suppose there is a* **CL***-computable (resp.* **CLP***) $(1/10n)$-*PCAPP *algorithm for $\mathcal{C}$ circuits. Then there is a* **CL** *(resp.* **CLP***) algorithm that, given $C \in \mathcal{C}_n$, outputs a distribution $\mathbf{D}$ over $\{0,1\}^n$ that $(1/3)$-fools C.*

We note two features (shared with Theorem 4.1): we only require a prefix-CAPP algorithm, and only one with a certain fixed polynomial accuracy; we use both of these features in Theorem 6.13.

Proof overview. We give a more detailed proof overview, following the outline given in Section 2.3. To prove Theorem 5.4, we combine two techniques: the construction of D2P from CAPP of Lemma 4.6, and the compress-or-random framework for **CL** [Pyn24; DPT24], in particular the use of the catalytic tape as a hard truth table by Doron, Pyne, and Tell [DPT24].

Recall that the task is to reduce producing a targeted PRG to solving CAPP in **CL**. At a high-level, given a circuit C, we produce a D2P transformation for C from the CAPP algorithm and use this transformation to test if a candidate pseudorandom set (constructed using the catalytic tape) fools C. If the candidate pseudorandom set fails to fool C, we use the D2P transformation to produce a predictor, which, together with the reconstructive procedure for the pseudorandom set, compresses the catalytic tape and thus allows us to run a space-inefficient derandomization algorithm. We stress that such a D2P transformation is useful because the reconstructive procedure is deterministic only if it is given a predictor rather than a distinguisher.

For the latter step, we use the result of [DPT24] with minor modification, which we describe below. Following the proof of [DPT24, Theorem 1.5], we treat (a section of) the catalytic tape $\mathbf{w}$ as a hard truth table and apply the Nisan-Wigderson generator [NW94; IW97] with this truth table, instantiated with the locally-encodable code of [DPT24] (rather than the usual [STV01]). Denoting the generator as $\mathsf{NW}^{\mathbf{w}}$, on an input C, one of two events occur:

1. **Case I: "Random".** If $\mathsf{NW}^{\mathbf{w}}(\mathbf{U})$ is unpredictable by every candidate predictor in the D2P transformation for C, we conclude that $\mathsf{NW}^{\mathbf{w}}(\cdot)$ is a correct targeted PRG and simply output the distribution $\mathsf{NW}^{\mathbf{w}}(\mathbf{U})$.[23]

2. **Case II: "Compress".** Otherwise, the works of [NW94; IW97] established that if $\mathsf{NW}^{\mathbf{w}}$ does not $(1/3)$-fool C, there is a compressed representation for $\mathbf{w}$ with oracle access to C. The work of [DPT24] established that one can deterministically replace $\mathbf{w}$ with such a compressed representation in place, assuming access to a D2P transformation for C. Specifically, in this case they identify a subinterval of $\mathbf{w}$ of size $\mathrm{poly}(n)$ that can be safely erased, and then compute the exact acceptance probability of C by an exponential-time, polynomial-space brute-force enumeration using the (erased) subinterval as a work tape.

[22] The meaning of "total" here is that for every $x \in \{0,1\}^*$ there is $y \in \{0,1\}^*$ such that $(x,y) \in R$.

[23] In the proof of [DPT24, Theorem 1.5], they output $\mathbb{E}[C(\mathsf{NW}^{\mathbf{w}}(\mathbf{U}))]$ as their goal is to derandomize C instead of producing a targeted PRG.

We modify this component of their algorithm: After erasing the subinterval, we run the polynomial-time space-inefficient algorithm of Lemma 4.2 using the subinterval as a work tape to produce a targeted PRG.

We now state our improvement to [DPT24, Theorem 1.5], reflecting the change above.

Theorem 5.5. *Suppose a circuit class $\mathcal{C}$ satisfies the following.*

1. *There is a* **CL** *(resp.* **CLP***) algorithm that, given $C \in \mathcal{C}$ and $r \in \{0,1\}^n$, outputs $C(r)$.*
2. *There is a* **CL***-computable (resp.* **CLP***-computable) $(\infty, \alpha = 1/10n)$-distinguish-to-predict transformation for $C \in \mathcal{C}$.*

Then there is a **CL** *(resp.* **CLP***) algorithm that, given $C \in \mathcal{C}$, outputs a distribution D that $(1/3)$-fools C.*

We highlight the improvements of Theorem 5.5 upon [DPT24, Theorem 1.5]. First, we obtain a distribution that fools the circuit rather than only outputting the (approximated) acceptance probability. Second, we ensure that the final algorithm is time efficient if the evaluation algorithm and the D2P transformation are both time efficient, which is not true for [DPT24, Theorem 1.5] due to the time-inefficient brute-force search in the "compress" case.

We can then prove the result.

Theorem 5.4. *For an arbitrary class of circuits $\mathcal{C}$ that is evaluable in* **CL** *(resp.* **CLP***), suppose there is a* **CL***-computable (resp.* **CLP***) $(1/10n)$-*PCAPP *algorithm for $\mathcal{C}$ circuits. Then there is a* **CL** *(resp.* **CLP***) algorithm that, given $C \in \mathcal{C}_n$, outputs a distribution $\mathbf{D}$ over $\{0,1\}^n$ that $(1/3)$-fools C.*

Proof. We will only prove the case for **CLP**-evaluable $\mathcal{C}$ with a $\mathcal{C}$-PCAPP algorithm in **CLP**. The result for **CL** follows the same argument.

Our catalytic machine works as follows. We think of the catalytic tape as comprised of blocks $\mathbf{w}_1, \mathbf{w}_2$, both of size $\mathrm{poly}(n)$. Let $\mathcal{E}$ be the **CLP** machine that computes PCAPP, and initialize it with catalytic tape $\mathbf{w}_2$. Given $C \in \mathcal{C}$, consider the D2P transform

$$\left(P_1, \ldots, P_{O(n^2)}\right) \leftarrow C$$

obtained from applying Lemma 4.6 to C, where the PCAPP oracle is implemented with calls to $\mathcal{E}$. Note that each element of the D2P transform can be described using $O(\log n)$ bits (i.e. by specifying τ, σ, i), as long as we do not edit $\mathbf{w}_2$, and thus we can evaluate each predictor in **CLP**. (We need $\mathbf{w}_2$ to remain as-is since it is used by $\mathcal{E}$.)

Next, we instantiate the algorithm of Theorem 5.5 with $\mathbf{w}_1$, where we give that algorithm oracle access to the D2P transformation. Every time the algorithm queries for $P_{\sigma,\tau,i}(x)$, we use $\mathcal{E}$ to evaluate the PCAPP oracle on the prefix $x_<$ and return the value. By the correctness of that algorithm, it returns a distribution $\mathbf{D}$ that $(1/3)$-fools C and resets $\mathbf{w}_1$. While that machine executes, it may call $\mathcal{E}$,[24] which will itself reset $\mathbf{w}_2$ to the original configuration after every call. □

[24] As we are fine with paying additively for both machines' catalytic space, we do not have to deal with calling $\mathcal{E}$ with different initial catalytic tapes (in which case our definition of catalytic search algorithms would permit it to return different solutions), as would occur if we used the same tape for both machines (as was the case in the more challenging setting in the work of [Pyn24]).

5.2 Targeted PRGs in Logspace

We now state our result for constructing targeted PRGs in logspace. To do so, we formally state our targeted hitting set (the proof of correctness of which follows in Appendix E).

Theorem 5.6. *There exists a universal constant $c > 1$ such that the following holds. Let $f : \{0,1\}^n \to \{0,1\}^n$ be computable by logspace-uniform NC^1 circuits of depth $d(n) = O(\log T)$ and size $T(n) \leq 2^{d(n)}$. Let $\delta > 0$ and let $M(n)$ be such that $c \log T(n) \leq M(n) \leq T(n)^{\delta/c}$. Then there is a deterministic algorithm G_f and a probabilistic algorithm R that satisfy the following. For every $x \in \{0,1\}^n$:*

- ***Generator.*** *The generator G_f gets input x, runs in space $O(\log T)$, and outputs $d' = \text{polylog}(T)$ lists of M-bit strings $L_1, \ldots, L_{d'}$. Moreover, each list can be computed in space $O(\log T)$.*
- ***Reconstruction.*** *When R gets input x and oracle access to $(P_1, \ldots, P_{M^3}) : \{0,1\}^{\leq M} \to \{0,1\}$ such that for every $i \in [d']$, there is $j \in [M^3]$ such that*

$$\text{adv}_{L_i}(P_j) \geq \frac{1}{M^2}$$

then R outputs $f(x)$ with probability at least $1 - 1/M$. The procedure R runs in time $T^\delta \cdot n^c$ and uses at most $\text{polylog}(T)$ random coins.

We now leverage Theorem 5.6 to (conditionally) construct a logspace targeted PRG for any class for which we have a CAPP algorithm, and which is logspace evaluable (i.e. the function $(C, x) \to C(x)$ can be computed in logspace).

Theorem 5.7 (reconstructive targeted somewhere-PRG)**.** *Suppose that Assumption 1.10 holds. Then, for every class $\mathcal{C}$ of fixed-polynomial size circuits[25] that is evaluable in logspace and has a polynomial time $(1/10n)$-PCAPP algorithm, there is a logspace algorithm that, on input $C \in \mathcal{C}_n$ with $\mathbb{E}[C] \geq 1/2$, outputs x such that $C(x) = 1$. Moreover, if the PCAPP algorithm can be computed in logspace, the algorithm on every C outputs a distribution $\mathbf{D}$ that $(1/3)$-fools C.*

Proof. Suppose there is a PCAPP algorithm for $\mathcal{C}$ circuits running in time $n = m^b$ on circuits $C : \{0,1\}^m \to \{0,1\}$ (recall that such a b exists per the assumption that the PCAPP algorithm in in polynomial time and the circuits on m bits are of fixed polynomial size) of size at most n.

Denote the function family of Assumption 1.10 as $\{f_n\}_{n \in \mathbb{N}}$, and recall that it can be computed by circuits of size $T(n) = n^A$ and cannot be computed by n^a-time algorithms using $\text{polylog}(n)$ random coins, where $a \geq 3c + 1$ and c is the universal constant of Theorem 5.6. We instantiate the targeted somewhere-PRG of Theorem 5.6 with f_n and parameter values

$$\delta = (a/3A), \qquad m = M(n) \leq T^{\delta/c}.$$

Then our algorithm works as follows.

The generator. We are given $C : \{0,1\}^m \to \{0,1\}$ of size at most n. If we have a PCAPP algorithm computable in L, we estimate $\rho \approx \mathbb{E}[C(\mathbf{U}_m)]$ to error $1/10$. We then enumerate over the lists $L_1, \ldots, L_{d'} \subseteq \{0,1\}^m$ output by $G_f(C)$ and for the first i where

$$\left| \mathop{\mathbb{E}}_{x \leftarrow L_i} [C(x)] - \rho \right| \leq 1/10 \tag{2}$$

[25] That is, every $C \in \mathcal{C}$ on n inputs has size at most n^b, for a fixed constant b.

we output L_i. Indeed, this list (1/3)-fools C, as required. If we do not have an accessible CAPP algorithm, we simply output the first element of any list that hits C. The fact that both procedures are computable in space $O(\log n)$ follows from Theorem 5.6.

Proof of Correctness. We claim that for all but finitely many input circuits C, at least one of the lists satisfies Equation (2).

Suppose that there is a circuit C where Equation (2) does not hold for every $i \in [d']$. This implies that L_i does not (1/3)-fool C for every i. We show how to compute $f_n(C)$ in time n^a with polylog(n) random coins, contradicting the hardness of f. To do this, let

$$\left(P_1, \ldots, P_{O(m^2)}\right) \leftarrow C$$

be the predictors produced by the transformation of Lemma 4.6 applied with the PCAPP algorithm. Note that each predictor can be evaluated in time $\widetilde{O}(n)$, and the number of predictors is at most $m^3 \leq n^3$ (for sufficiently large n). Then we run the reconstruction algorithm of Theorem 5.6, where every time R queries P_i, we answer in time $\widetilde{O}(n)$. By that theorem, with probability at least $1-1/n$ we compute $f_n(C)$, and we use polylog(n) random coins and run in time $\widetilde{O}(n)(n^A)^\delta n^c \leq n^a$ by our choice of a.

Thus, if Equation (2) held for infinitely many C, there would be a time n^a algorithm which computes $f(C)$ on infinitely many C, contradicting the hardness hypothesis. □

6 Distinguish to Predict for the Path Isolation Lemma

In this section we prove Theorem 1.9 and Theorem 1.11. To formally state the results, we first formally define the relevant classes. Let us start by defining unambiguous nondeterministic small-space.

Definition 6.1 (unambiguous nondeterministic small-space)**.** A language L is in unambiguous space $S(n)$, denoted $\mathbf{USPACE}[S(n)]$, if there is a nondeterministic space-S machine $\mathcal{M}(x, g)$ such that for every $x \in L$, there is exactly one witness g such that $\mathcal{M}(x, g)$ accepts, and for every $x \notin L$ and every witness g, $\mathcal{M}(x, g)$ rejects. We let $\mathbf{UL} = \cup_c \mathbf{USPACE}[c \cdot \log n]$.

As in the time-bounded setting, it is immediate that $\mathbf{USPACE}[S] \subseteq \mathbf{NSPACE}[S]$, as we simply remove the requirement that the witness is unique.

Next, we introduce a notion of uniform circuits, where the uniform machine printing the circuit is an unambiguous logspace algorithm.

Definition 6.2 (unambiguous-logspace-uniformity)**.** A family of circuits $\{C_n : \{0,1\}^n \to \{0,1\}\}_{n\in\mathbb{N}}$ of size $S(n)$ is said to be:

- Unambiguous-logspace-uniform (ULU) if there is a nondeterministic machine $\mathcal{M}$ that, on input $0^n 1^{S(n)} \langle i \rangle$ where $i \in [S(n)]$ (and $\langle i \rangle$ is its binary representation), runs in logspace and satisfies the following: For every i there is exactly one guess sequence on which $\mathcal{M}$ outputs the i-th bit of the description of C_n, and otherwise it outputs $\perp$.[26]
- Unambiguous-logspace-evaluable (ULE) if there is a nondeterministic machine $\mathcal{M}'$ that, on input (C_n, x) where $x \in \{0,1\}^n$, runs in logspace and satisfies the following: There is exactly one guess sequence on which $\mathcal{M}'$ outputs $C_n(x)$, and otherwise it outputs $\perp$.

[26] We note that an equivalent definition is that there is a nondeterministic logspace machine that prints a circuit to a write-only output tape, and on exactly one guess sequence the machine writes the entire circuit without writing a special abort symbol $\perp$.

We remark two things. First, **UL**-uniform circuits are **P**-uniform. Second, the languages decided by circuits that are ULU and ULE can be decided in $\mathbf{UL} \cap \mathbf{coUL}$ (and in particular are in **P**).

Proposition 6.3. *Let $\{C_n\}_{n\in\mathbb{N}}$ be a family of circuits of size $S(n) \geq n$ that is ULU and ULE. Then for the language defined on inputs of length n as $x \in L_n \iff C_n(x) = 1$, we have that $L \in \mathbf{USPACE}[O(\log S)] \cap \mathbf{coUSPACE}[O(\log S)]$.*

Proof. On input x, we run the machine $\mathcal{M}'$ for ULE on input C_n, x. Every time that $\mathcal{M}'$ queries its input tape for the ith bit of C_n , we call the machine $\mathcal{M}(0^n, 1^{S(n)}\langle i\rangle)$. If $\mathcal{M}$ outputs $\perp$, we also output $\perp$, and otherwise we continue the execution. Observe that every time $\mathcal{M}'$ reads a bit of C_n, there is one sequence of guesses for $\mathcal{M}$ that does not cause the computation to abort. Thus, for every x there is a unique guess sequence where we output a value, and this value is $L(x)$ (i.e., 1 when $x \in L$ and 0 when $x \notin L$). By the standard composition of space-bounded algorithms, this algorithm can be implemented in space $O(\log S)$. □

Given these definitions, we can now formally state Theorem 1.9, and for convenience let us also restate Theorem 1.11:

Theorem 6.4. *There is a universal constant d such that the following holds. Suppose there exists $\varepsilon > 0$ such that $\mathbf{USPACE}[n] \cap \mathbf{coUSPACE}[n]$ is hard for unambiguous-logspace-uniform $(\mathbf{TC}^0)^{\mathbf{USPACE}[\varepsilon n]\cap\mathbf{coUSPACE}[\varepsilon n]}$ circuits[27] of depth d and size $2^{\varepsilon n}$ that are unambiguous-logspace-evaluable. Then $\mathbf{NSPACE}[O(n)] = \mathbf{USPACE}[O(n)]$.*

Theorem 1.11. *Suppose that Assumption 1.10 holds. Then $\mathbf{NL} = \mathbf{UL}$.*

High-level proof overview. Let us first recall the techniques of Reinhardt and Allender [RA00], who prove that nondeterminism can be made unambiguous in the presence of polynomially many bits of advice. Recall that the canonical **NL**-complete problem calls for deciding if there exists an $s \to t$ path in a directed graph G. The following definition refers to weighing the edges in the graph, and expecting a unique shortest path:

Definition 6.5 (unique shortest paths)**.** Given a directed graph $G = (V, E)$, we say a weight function $w : E \to \mathbb{R}$ induces unique shortest paths (USPs) if for every pair of vertices (s, t), if (s, t) is connected in G, there is exactly one shortest $s \to t$ path under the weight function w.

The work of [RA00] shows that if we consider (weighted) graphs with *unique* shortest paths, we can decide the **NL**-complete problem of st-connectivity unambiguously:

Theorem 6.6 (low-space nondeterminism becomes unambiguous with unique shortest paths [RA00])**.** *There is a nondeterministic logspace machine $\mathcal{U}$ that acts as follows. Given as input a graph $G = (V, E)$, nodes $s, t \in V$, and a weight function $w : E \to [n^{10}]$, the machine makes nondeterministic guesses and outputs a value in $\{\mathtt{BAD}, \mathtt{CONN}, \perp\}$ such that:*

- *If w does* not *induce unique shortest paths in G, there is exactly one guess sequence on which $\mathcal{U}$ outputs* BAD*, and on all other sequences it outputs $\perp$.*

[27] Formally, the oracle gate of the $\mathbf{TC}^0$ circuit decides a language $L \in \mathbf{USPACE}[\varepsilon n] \cap \mathbf{coUSPACE}[\varepsilon n]$, where n is the input length to the circuit instead of the fan-in $m = m(n)$ of the oracle gate. The description of a $(\mathbf{TC}^0)^{\mathbf{USPACE}[\varepsilon n]\cap\mathbf{coUSPACE}[\varepsilon n]}$ circuit is defined as the concatenation of an oracle $\mathbf{TC}^0$ circuit and an unambiguous nondeterministic machine M that runs in space εn (on input length m) and decides L (and the size of the description is the size of the circuit plus the description of M).

- *If w does induce unique shortest paths in G and there exists an $s \to t$ path in G, there is exactly one guess sequence on which $\mathcal{U}$ outputs* CONN*, and on all other sequences it outputs $\perp$.*
- *If there does not exist an $s \to t$ path, on all guess sequences $\mathcal{U}$ outputs $\perp$*

Thus, making nondeterminism unambiguous reduces to, given a graph G, constructing a sequence of weight functions $(w^{(1)}, \ldots, w^{(t)})$ such that $w^{(i)}$ induces unique shortest paths in G for some $i \in [t]$. Finally, it is observed in [RA00] that a random set of $\mathrm{poly}(n)$ weightings satisfies this property, and hence (by wiring in a good PRG as advice), they obtained that $\mathbf{NL/poly} = \mathbf{UL/poly}$.

We will use a targeted PRG constructed from a uniform hardness assumption to produce these weight assignments. Our goal will be to show that, in the case that the targeted PRG is bad, we can (in unambiguous logspace) obtain a *predictor* for the PRG, which will enable us to apply the deterministic and space-efficient reconstruction procedure of [DPT24], contradicting the hardness assumption.

The crucial part is the D2P transformation for the relevant distinguisher, which tests whether or not a given weight assignment induces unique shortest paths in G. To do so, we use the fact that obtaining a D2P transformation for any function T reduces to the task of solving prefix-CAPP, i.e. estimating $\mathbb{E}_r[T(x_< \circ r)]$ up to error $1/m$, where $m = O(n^2 \log n)$ is the length of the hybrid argument (see Lemma 4.2). As explained in Section 2.2, by modifying the distinguisher we get a "polarization" property, and obtain a $(1/m)$-PCAPP algorithm for this distinguisher. We note that our algorithm does *not* solve CAPP for the general problem of arbitrary (i.e. not prefix-) restrictions of T, nor can it produce estimates with arbitrarily good error.

We first define the distinguisher we wish to fool, in terms of the graph G. We represent the weight function as a vector

$$w = (w_1, \ldots, w_m) \in [n^{10}]^m$$

where each w_i is represented by a vector of $10 \cdot \log(n)$ bits.[28]

Definition 6.7. For a directed graph $G = (V, E)$ with $n = |V|, m = |E|$, let $T = T_G : [n^{10}]^m \to \{0,1\}$ be the function defined as follows. Fixing an arbitrary ordering of the edges $(e_1, \ldots, e_m)$, let $E_i \subseteq E$ be the set containing the first i edges, let $G_i = (V, E_i)$, and let $w_{\le i}$ be the restriction of w to these edges. Then define

$$T^i(w_{\le i}) \triangleq \mathbb{I}\left[w_{\le i} \text{ induces USPs in } G_i\right], \qquad T(w) \triangleq \bigwedge_{i \in [m]} T^i(w_{\le i}).$$

As our main technical result, we show that there is a deterministic logspace D2P transformation for this distinguisher, where the circuits are ULE. Note that here we think of T as a Boolean function that takes as input $m \cdot 10 \cdot \log n$ bits.

Theorem 6.8. *There is a deterministic logspace algorithm $\mathcal{T}$ that works as follows. Given a directed graph G with n vertices and m edges, let $\ell = 10 \log n$ and let $T : \{0,1\}^{m \cdot \ell} \to \{0,1\}$ be as in Definition 6.7. Then $\mathcal{T}(G)$ outputs a $(1/2)$-distinguish to $(1/10m\ell)$-predict transformation $(P_1, \ldots, P_t)$ for T, where $t = \widetilde{O}(n^4)$. Moreover, there is a $\mathbf{USPACE}[O(\log n)] \cap \mathbf{coUSPACE}[O(\log n)]$ machine $\mathcal{P}$ that on input (G, i, x), unambiguously computes $P_i(x)$.*

The rest of the section is organized as follows. In Section 6.1 we prove Theorem 6.8. In Section 6.2 we apply this result to construct unique shortest path weightings in $\mathbf{CLP}$. In Section 6.3 we prove Theorem 6.4, and in Section 6.4 we prove Theorem 1.11.

[28] We assume n is a power of two for simplicity of presentation.

6.1 Constructing the D2P Transformation

Recall that by Lemma 4.6, to construct a D2P transformation for T in which each predictor is computable in unambiguous logspace, it suffices to create an (unambiguous logspace) algorithm that, on input $w_< = (w_1, \ldots, w_i)$, estimates $\mathbb{E}_r[T(w_< \circ r)]$ up to error $1/10m\ell$ (for now we ignore the minor complication of receiving the input bit by bit). To do so, we show that that for *every prefix* $w_<$, the expectation of T over suffixes r polarizes, in the following sense (for now we think of T as taking m inputs in $[n^{10}]$):

Lemma 6.9. *For every partial weight assignment $w_<$, either:*

1. *There exists a (possibly non-strict) prefix $w'_<$ of $w_<$ such that $w'_<$ does not induce USPs in the relevant prefix graph (and hence $T(w_< \circ r) = 0$ for every r).*
2. *We have $\mathbb{E}_r[T(w_< \circ r)] \geq 1 - n^{-6}$.*

Proof. Observe that to establish the claim, it suffices to prove that for every partial assignment $(w_1, \ldots, w_{i-1})$ that **does** induce USPs in G_{i-1}[29], when choosing w_i uniformly at random over $[n^{10}]$, the probability that $(w_1, \ldots, w_{i-1}, w_i)$ fails to induce USPs in G_i is at most n^{-8}. Similar bounds have been used in [RA00; GW96], but we carefully spell out the argument that we obtain polarization in all cases.

Let $d_{uv}^{G_j}$ be the length of shortest paths from u to v in G with respect to the weight assignment $w_{\leq j}$. Suppose that after setting w_i to a value w, there are non-distinct shortest paths in G_i. Let the mutual start and end vertices of an arbitrary violation be u and v respectively, and let p_1 and p_2 be two non-equal shortest paths from u to v. It is easy to see that if neither path traverses the edge e_i, G_{i-1} also did not have unique shortest paths (as both would be present in G_{i-1} and removing e_i cannot shorten any other path), contradicting the assumption. Furthermore, if *both* paths traverse $e_i = (a, b)$, then either $u \to a$ or $b \to v$ had non-unique shortest paths in G_{i-1}, again a contradiction. Thus, it must be the case that (wlog) p_1 contains e_i and p_2 does not, and moreover p_1 contains as subpaths shortest $u \to a$ and $b \to v$ paths in G_{i-1} (as otherwise it would contradict the minimality of p_1). Finally, note that p_2 is the shortest $u \to v$ path in G_{i-1}. But then we have

$$w + d_{ua}^{G_i} + d_{bv}^{G_i} = d_{uv}^{G_i} = d_{uv}^{G_{i-1}} \implies w = d_{uv}^{G_{i-1}} - d_{va}^{G_i} - d_{vb}^{G_i}$$

as the two paths have the same lengths by assumption. Moreover, we know that $d_{ua}^{G_i} = d_{ua}^{G_{i-1}}$ and $d_{bv}^{G_i} = d_{bv}^{G_{i-1}}$. Thus, there is exactly one choice $w \in [n^{10}]$ for w_i that causes a failure on (u, v). As there are at most n^2 pairs of vertices on which to cause a failure (and at most n^2 edges for which we assign a random weight), the bound follows. □

Using Lemma 6.9, we now create an unambiguous logspace algorithm which solves (n^{-6})-PCAPP for T, which is smaller than the required threshold for Lemma 4.6.

Lemma 6.10. *There is a nondeterministic logspace machine $\mathcal{M}(G, w_<) \to \{0, 1, \perp\}$ that, given arbitrary $G = (V, E)$ and a partial weight assignment $w_< \in [n^{10}]^i$ as input, satisfies the following:*

1. *There is exactly one guess sequence that causes $\mathcal{M}$ to not output $\perp$.*

[29] Recall that G_k denotes the subgraph of G consisting of all vertices and the first k edges (with respect to an arbitrary fixed ordering).

2. On this guess sequence, $\mathcal{M}(G, w_<)$ outputs

$$b \triangleq \bigwedge_{j \le i} T^j(w_{\le j}).$$

and this value satisfies $|b - \mathbb{E}_r[T(w_< \circ r)]| \le n^{-6}$.

Proof. The machine $\mathcal{M}$ iterates over $j = 1, \ldots, i$, and for each j calls the nondeterministic machine of Theorem 6.6 on $(G_j, w_{\le j}, s = u, t = v)$ where (u, v) are the endpoints of edge e_1. Note that if $w_{\le j}$ does not induce USPs in G_i, there is exactly one guess sequence that outputs `BAD` and all others output $\perp$ (which we return if so), and if $w_{\le j}$ does induce USPs in G_j, u is obviously connected to v and so there is exactly one guess sequence that outputs `CONN` and all others output $\perp$ (which we return if so), so we can unambiguously decide if $w_{\le j}$ induces unique shortest paths. At the first j where the machine returns `BAD`, return 0, and if this does not occur, return 1. By Lemma 6.9, this value b satisfies $|b - \mathbb{E}_r[T(w_< \circ r)]| \le n^{-6}$. □

We can then prove the correctness of the D2P transformation. We first recall the precise statement:

Theorem 6.8. *There is a deterministic logspace algorithm $\mathcal{T}$ that works as follows. Given a directed graph G with n vertices and m edges, let $\ell = 10 \log n$ and let $T : \{0,1\}^{m \cdot \ell} \to \{0,1\}$ be as in Definition 6.7. Then $\mathcal{T}(G)$ outputs a $(1/2)$-distinguish to $(1/10m\ell)$-predict transformation $(P_1, \ldots, P_t)$ for T, where $t = \widetilde{O}(n^4)$. Moreover, there is a $\mathbf{USPACE}[O(\log n)] \cap \mathbf{coUSPACE}[O(\log n)]$ machine $\mathcal{P}$ that on input (G, i, x), unambiguously computes $P_i(x)$.*

Proof. We construct a PCAPP machine for T as follows. Represent the input bits as $w_1, \ldots, w_m$ where $w_i \in \{0,1\}^\ell$. On input a prefix $x = (w_1, \ldots, w_{i-1}, y)$, let $y \in \{0,1\}^d$ correspond to the bits of x that fall into the final block. Then:

1. Enumerate over $s \in \{0,1\}^{\ell - d}$.
2. Run the machine $\mathcal{M}$ of Lemma 6.10 on input $w_< = (x \circ s) = (w_1, \ldots, w_{i-1}, y \circ s)$ and store the average of the values returned by these calls.
3. If all calls return a value, return the average, and otherwise return $\perp$.

The fact that this algorithm can be computed in $\mathbf{UL} \cap \mathbf{coUL}$ follows from Lemma 6.10 and the fact that $\ell = O(\log n)$. We argue correctness:

Claim 6.11. *The algorithm above is a $(1/100m\ell)$-PCAPP algorithm for T.*

Proof. For an arbitrary $x = (w_1, \ldots, w_{i-1}, y)$, note that

$$\begin{aligned}\left| \mathop{\mathbb{E}}_{s,r \leftarrow \mathbf{U}}[T(x \circ s \circ r)] - \mathop{\mathbb{E}}_{s}[\mathcal{M}(G, x \circ s)] \right| &\le \max_s \left| \mathop{\mathbb{E}}_{r \leftarrow \mathbf{U}}[T((x \circ s) \circ r)] - \mathcal{M}(G, x \circ s) \right| \\ &\le n^{-6} \le \frac{1}{100m\ell}\end{aligned}$$

where the last line follows from Lemma 6.10 (and the notation $\mathcal{M}(G, x \circ s)$ represents the (unambiguous) output of $\mathcal{M}$ on valid nondeterministic guesses). □

Then applying Lemma 4.6 with $\delta = 1/2$, we have that the transformation exists, and the number of predictors is $t = 12(m\ell)^2 = \widetilde{O}(n^4)$. Moreover, note that the circuits printed Lemma 4.6 can clearly be constructed in logspace, by printing the code for $\mathcal{M}$.

Evaluability of the predictor circuits. Given i, σ, τ, to evaluate $P_{\tau,\sigma,i}(x)$ we call the PCAPP machine on $x_{<i} \circ \sigma_1$ (which can be evaluated unambiguously in the claimed space bound), then compare to the threshold τ and output accordingly. □

6.2 Unique Shortest Paths in Catalytic Logspace

As an immediate corollary of Theorem 6.8, we show that we can derandomize the path isolation lemma in **CLP**. Our result is a simple combination of the D2P transformation for the relevant distinguisher, with the search to decision reduction of Theorem 5.4. Recall that Theorem 6.8 refers to the test $T = T_G$ defined in Definition 6.7, and asserts a D2P transform for T; then:

Fact 6.12. The predictors produced by the D2P transformation of Theorem 6.8 can be evaluated in **CLP**.

Proof. Theorem 6.8 asserts that the function mapping (G, i, x) to $P_i(x)$ (where $P_1, ..., P_t$ are the predictors in the D2P transform $\mathcal{T}(G)$ of T) is computable in unambiguous logspace $\mathbf{USPACE}[O(\log n)] \cap \mathbf{coUSPACE}[O(\log n)]$. The claim follows since the latter class is contained in **CLP** [BCK+14].[30] □

Then the result follows by Theorem 5.5:

Theorem 6.13. *There is a* **CLP** *algorithm that, given a directed graph $G = (V, E)$, outputs a weighting $w : E \to [n^{10}]$ such that all shortest paths in (G, w) are unique.*

Proof. Given the graph G, recall from Theorem 6.8 that there is a logspace-computable D2P transform for the distinguisher T_G. We then apply the result of Theorem 5.5, where we provide oracle access to this transformation, and evaluate all calls to the oracle using Fact 6.12. Finally, note that we obtain a distribution $\mathbf{D}$ that fools the test T_G, and that $\mathbb{E}[T_G(\mathbf{U})] = 1 - o(1)$, so in particular there is $w \in \mathbf{D}$ such that $T_G(w) = 1$, i.e., w induces shortest paths in G. We simply test one-by-one the elements of $\mathbf{D}$ (using the argument of Fact 6.12), and return the first such good weight assignment. □

We make two remarks. First, this result is incomparable to the result of [MP19], which gives a $\mathbf{UTISP}[n^c, O(\log^{3/2} n)]$ algorithm for the same problem - our result does not use unambiguous nondeterminism, and uses only $O(\log n)$ workspace, but requires a catalytic tape of length $\mathrm{poly}(n)$. Second, this result represents the first problem known to be solvable in **CL** whose proof uses *both* main algorithmic techniques known for **CL** (i.e., the algebraic computation approach [BCK+14; CM23] and the compress-or-random approach [Pyn24; DPT24; CLM+24]); all known results so far relied only on one of the two.

6.3 Making Nondeterministic Linear Space Unambiguous

In this section we prove Theorem 6.4. To do so, let us recall the standard reduction from nondeterministic machines to deciding $s \to t$ connectivity.

Theorem 6.14 (st-connectivity is complete for nondeterministic space, Theorem 4.16 [AB09])**.** *For $S \geq \log n$, let $L_0 \in \mathbf{NSPACE}[S(n)]$ be decided by a nondeterministic machine $\mathcal{N}$ using space $S(n)$. For $x \in \{0,1\}^n$, let G_x be the (directed) graph on $2^{S(n)}$ vertices*[31] *corresponding to the transitions*

[30] It is proved in [BCK+14] that (uniform) $\mathbf{TC}^1 \subseteq \mathbf{CL}$. By inspection, the algorithm has polynomial runtime for every initial catalytic tape.

[31] Technically the number is slightly larger, as we must track the head positions and FSM configuration, but we ignore this minor technicality for simplicity.

made by the machine, and let s and t be the start and accept configurations, respectively. Then, there is a deterministic space $O(S(n))$ machine that gets input x and outputs (G_x, s, t), and we have that:

$$x \in L_0 \iff s, t \text{ are connected in } G_x.$$

Next, we recall the PRG of [DPT24] with deterministic reconstruction:

Theorem 6.15 ([DPT24] Theorem 7.4)**.** *There are universal constants $d, c_{\mathsf{NW}} > 1$ such that for every sufficiently small constant $\varepsilon_{\mathsf{NW}} > 0$ the following holds. There is an algorithm NW computing*

$$\mathsf{NW}^f \colon \{0,1\}^{(c_{\mathsf{NW}}/\varepsilon_{\mathsf{NW}})\cdot \log N} \to \{0,1\}^{M=N^{\varepsilon_{\mathsf{NW}}}}$$

such that for any $f \in \{0,1\}^N$ we have the following.

1. ***Efficiency.*** *On input s and $i \in [M]$, $\mathsf{NW}^f(s)_i$ can be computed in space $(c_{\mathsf{NW}}/\varepsilon_{\mathsf{NW}}) \cdot \log N$.*
2. ***Reconstruction.*** *There is a deterministic space $O(\log N)$ algorithm R that, given oracle access to f and oracle access to a $(\frac{1}{M^2})$-predictor P for NW^f, prints a oracle circuit C of depth d and size $M^{c_{\mathsf{NW}}}$ that has majority gates, makes non-adaptive queries, and satisfies the following: $C^P(x) = f_x$.*

We can now restate Theorem 6.4 and prove it. Our result closely follows the strategy of [DPT24, Theorem 6.16], except that we must be careful to achieve the desired uniformity condition (using the D2P from the previous section).

Theorem 6.4. *There is a universal constant d such that the following holds. Suppose there exists $\varepsilon > 0$ such that $\mathbf{USPACE}[n] \cap \mathbf{coUSPACE}[n]$ is hard for unambiguous-logspace-uniform $(\mathbf{TC}^0)^{\mathbf{USPACE}[\varepsilon n]\cap\mathbf{coUSPACE}[\varepsilon n]}$ circuits[27] of depth d and size $2^{\varepsilon n}$ that are unambiguous-logspace-evaluable. Then $\mathbf{NSPACE}[O(n)] = \mathbf{USPACE}[O(n)]$.*

Proof. We first define the relevant nondeterministic machines and languages:

- Let $L_0 \in \mathbf{NSPACE}[O(n)]$ be decided by a nondeterministic machine $\mathcal{N}$ using space $c_0 \cdot n$. For $x \in \{0,1\}^n$, let $(G_x = (V, E), u, v)$ be the graph $s \to t$ connectivity instance produced by the reduction of Theorem 6.14 with $S = c_0 \cdot n$, where $|V| = 2^{c_0 n}$ and $|E| \le 2^{2c_0 n}$.
- Let $\mathcal{H} : \{0,1\}^* \to \{0, 1, \perp\}$ be the nondeterministic machine that decides $L_{hard} \in \mathbf{USPACE}[n] \cap \mathbf{coUSPACE}[n]$ unambiguously. That is, for every x, there is exactly one guess sequence g_x such that $\mathcal{H}(x, g_x) \neq \perp$, and $\mathcal{H}(x, g_x) = \mathbb{I}[x \in L_{hard}]$.
- Let $\mathcal{U}$ be the logspace machine of Theorem 6.6 that unambiguously decides if a graph has unique shortest paths (and if so decides connectivity).
- Let $\mathcal{P}$ be the machine that on input $(G = (V, E), i, x)$ runs in space $c_p \cdot \log V$ where $c_p \ge 1$ and unambiguously computes $P_i(x)$, where P_i is the predictor of Theorem 6.8.

Let ε be as in our hypothesis, and let $c_{\mathsf{NW}} > 1$ be the universal constant from Theorem 6.15. Finally, let

$$\varepsilon_{\mathsf{NW}} = \frac{\varepsilon}{2c_{\mathsf{NW}}}, \qquad c = \frac{8c_0(c_{\mathsf{NW}} + c_p)}{\varepsilon}, \qquad N = c \cdot n.$$

Derandomization algorithm. Recall that we are given $x \in \{0,1\}^n$, and our goal is to decide $\mathcal{N}(x)$, i.e., $s \to t$ connectivity in (G_x, s, t), where s and t represent the initial and accept configurations, respectively.

To produce our sequence of weight functions, we use the PRG from Theorem 6.15, instantiated with parameter $\varepsilon_{\mathsf{NW}}$ and with a hard truth-table f given by L_{hard} on inputs of length N. The output length of the PRG is then

$$2^{\varepsilon_{\mathsf{NW}} \cdot N} \geq 2^{4c_0 \cdot n} \geq |E| \cdot 10 \log |V|$$

and the generator's seed length is $\ell = O(N)$. We then compute as follows. We enumerate over $y \in \{0,1\}^\ell$ in sequence, and let $w_y : E \to [V^{10}]$ be the weight function induced by $\mathsf{NW}^f(y)$. We then call the machine $\mathcal{U}((G_x, s, t), w_y)$ and:

- If $\mathcal{U}$ outputs $\perp$, we reject.
- If $\mathcal{U}$ returns `CONN`, we return `CONN`.
- If $\mathcal{U}$ returns `BAD`, we proceed to the next seed y.

If we exhaust all y's (i.e. every $\mathcal{U}$ always returns `BAD`) we reject, but we will show this does not occur given the assumption.

We first prove that the algorithm as described is unambiguous:

Claim 6.16. *The prior procedure can be implemented by an unambiguous space $O(N)$ machine.*

Proof. First, note that NW can be computed in deterministic space $O(N)$ with oracle calls to f. For each oracle query at index q, we use $\mathcal{H}$ to decide L_{hard} on q, and by assumption there is one guess sequence where we return the value f_q (and otherwise we reject). Note that we may query the same input bit multiple times, but the procedure remains unambiguous. Finally, for each weight function w_y that we construct, the machine $\mathcal{U}$ unambiguously computes `BAD` or `CONN`. Thus, there is at most one guess sequence where we do not reject. □

Next, we argue that the algorithm succeeds.

Correctness. We claim that (assuming the derandomization hypothesis) if $x \in L_0$ there will always be at least one seed y such that the call with weight function w_y either returns $\perp$ or `CONN`.

Assume for contradiction that there is an $x \in \{0,1\}^n$ where (G_x, s, t) is connected, but every guess sequence causes the machine to reject. Since there exist guesses for which the derandomization algorithm correctly computes all queries it makes to f, it must be that even with these guesses, all guesses made by $\mathcal{U}$ cause it to reject. By Theorem 6.6, in this case, for every $y \in \{0,1\}^\ell$ we have that $w_y = \mathsf{NW}^f(y)$ does not induce unique shortest paths in G, and therefore

$$T_G(\mathsf{NW}^f(\mathbf{U}_\ell)) = 0, \qquad \mathbb{E}[T_G(\mathbf{U})] = 1 - o(1) \tag{3}$$

where T is the function of Definition 6.7.

In this case, we show that L_{hard} can be decided by ULU and ULE circuits of size $2^{\varepsilon \cdot n}$. Specifically, we will rely on the following claim:

Claim 6.17. *There is an unambiguous space $O(N)$ algorithm $\mathcal{R}$ that gets input G such that Equation (3) holds, and prints a $(\mathsf{TC}^0)^{\mathsf{USPACE}[\varepsilon N] \cap \mathsf{coUSPACE}[\varepsilon N]}$ circuit of depth d (for some universal constant $d \in \mathbb{N}$) and size $2^{\varepsilon \cdot N}$, whose truth-table is f. Moreover, the circuit is ULE.*

Proof. We construct a nondeterministic machine that prints the entire circuit to a write-only output tape, where if we reject before printing the entire circuit, we halt and write a trailing $\perp$ to the tape (and there is at most one guess sequence where this abort symbol is not printed). To see that this is equivalent to the prior definition of ULU, consider an algorithm which gets an index i in the output circuit, and maintains a counter for the number of bits output so far (and does not write them to a tape). If we reach the ith output bit, store it, and continue to attempt to print the circuit. If we print $\perp$ before completing this task, reject, and otherwise return i.

We use the D2P transformation of Theorem 6.8 and the reconstruction algorithm R_{NW} from Theorem 6.15. First,

$$(P_1, \ldots, P_{d=\widetilde{O}(|V|^4)}) = \mathcal{T}(G)$$

be the $(1/2)$-distinguish to $(1/10|E|\log|V|)$-predict transformation of Theorem 6.8 for T_G. Note that every circuit P_i can be constructed in space $O(N)$ (given access to G and i) and can be evaluated in unambiguous space $O(N)$. By the correctness of the transformation, there exists an $i \in [d]$ such that

$$\mathrm{adv}_{\mathsf{NW}^f(\mathbf{U}_\ell)}(P_i) > \alpha. \tag{4}$$

We enumerate over $i \in [d]$ and, for each, compute the advantage of P_i over NW^f. Note that we can compute this quantity in unambiguous space $O(N)$ by essentially the same strategy of Claim 6.16. For the first P_i on which the advantage is at least α, we call the reconstruction algorithm R of Theorem 6.15 (which runs in deterministic space $O(N)$), and give it oracle access to P_i and f. Note that every time this machine queries P_i, we use the fact that P_i is ULE to compute the answer in unambiguous space $O(N)$, and use the machine $\mathcal{H}$ to answer queries to f in unambiguous space $O(N)$. Since the machine R is deterministic, and all oracles are unambiguous machines, there is exactly one guess sequence where we output the circuit. Finally, when we successfully print the circuit, we then print the description of the predictor P_i, which is an unambiguous logspace machine that takes hardwired input (G, i) and computes the map $x \to P_i(x)$ in

$$\mathbf{USPACE}[c_p \log|V|] \cap \mathbf{coUSPACE}[c_p \log|V|] \subseteq \mathbf{USPACE}[\varepsilon N/2] \cap \mathbf{coUSPACE}[\varepsilon N/2].$$

We can hardwire (G, i) (and the code for the computation of the machine) using $|V|^2 + \widetilde{O}(|V|^4) \leq 2^{\varepsilon N/2}$ bits.

Circuit properties. Observe that the printed circuit C is a constant-depth $\mathbf{TC}^0$ oracle circuit such that

$$C^{P_i}(x) = f_x.$$

The circuit C has size $2^{\varepsilon_{\mathsf{NW}} c_{\mathsf{NW}} N} \leq 2^{\varepsilon N/2}$ and has fixed constant depth by Theorem 6.15. The circuit that corresponds to the computation of P_i has size at most $2^{\varepsilon N/2}$, and hence the complete circuit has size at most $2^{\varepsilon N}$. Finally, the fact that the resulting circuit is ULE follows from the standard simulation of $\mathbf{TC}^0$ in logspace, where we replace the oracle calls to P_i with an actual evaluation of the unambiguous nondeterministic machine. If such a call returns $\perp$ we likewise abort, and otherwise we successfully compute the circuit. □

Finally, let us explain how to use Claim 6.17 to contradict the hardness of L_{hard}. Assume towards a contradiction that for a large enough $n \in \mathbb{N}$, Equation (3) holds for the configuration graph G_x generated by $\mathcal{N}$ on some input $x \in \{0,1\}^n$, and let $N = c \cdot n$. On input 1^N, we construct a circuit for $L_{hard} \cap \{0,1\}^N$ as follows:

1. Enumerate over $x \in \{0,1\}^n$.

2. Determine if Equation (3) fails for the configuration graph G_x of $\mathcal{N}$ on x.

3. If this holds for G_x, run the unambiguous machine of $\mathcal{R}$ and print its output (unless it rejects, in which case write a trailing $\perp$ to the tape), and then halt.

By assumption, such an x exists. Thus, it suffices to verify the algorithm runs in unambiguous space $O(N)$. For the second step, we enumerate over $s \in \{0,1\}^\ell$ and call the machine of Lemma 6.10 with graph G_x and weight function $w = \mathsf{NW}^f(s) \in [|V|^{10}]^{|E|}$ (to determine $T_{G_x}(w)$). Every time that machine queries the weight function, we compute f via the unambiguous machine $\mathcal{H}$. If for any s the machine of Lemma 6.10 returns 1, we proceed to the next x. Otherwise, if the machine returns 0 (i.e. every weight function output by the generator does not induce unique shortest paths), we have that Equation (3) holds for G_x. This algorithm that tests if the equation holds can be seen to run in unambiguous space $O(N)$ as claimed. Moreover, note that every time we call Lemma 6.10 the output is either $\perp$ (in which case we abort) or a fixed value b, so if at least one x exists such that Equation (3) holds, we always identify the first such x. We then use the algorithm $\mathcal{R}$ as established in Claim 6.17 with this x. Thus, the overall algorithm runs in space $O(N)$, and there is exactly one guess sequence where we output the circuit. □

6.4 Making Nondeterministic Logspace Umambiguous

We now show how to obtain our scaled-down result. Essentially the same approach as Theorem 5.7 gives the following targeted hitting set reduction for unambiguous logspace:[32]

Theorem 6.18. *Suppose that Assumption 1.10 holds. Then, for every class of circuits $\mathcal{C}$ that is ULE and has a polynomial time $(1/10n)$-PCAPP algorithm there is a $\mathbf{UL} \cap \mathbf{coUL}$ algorithm that, on input $C \in \mathcal{C}_n$ with $\mathbb{E}[C] \geq 1/2$, outputs x such that $C(x) = 1$.*

Proof Sketch. The proof is identical to that of Theorem 5.7, except that we modify the generator and reconstruction as follows.

The generator. For every $x \in L_i$, we determine if $C(x) = 1$ using our $\mathbf{UL} \cap \mathbf{coUL}$ algorithm to evaluate C, and return the first x where this occurs. As for every x this computation is unambiguous, so is the overall algorithm.

The reconstruction. We pad the input size n such that the circuit C is of linear size, and so that the algorithm that evaluates C from its description runs in linear time (note that there is a polynomial time algorithm that computes $(C, x) \to C(x)$ as $\mathbf{UL} \subseteq \mathbf{P}$), and the PCAPP algorithm runs in linear time. Then the generator works without modification. □

Applying Theorem 6.18 with $\mathcal{C}$ being the class of functions T_G of Definition 6.7, we obtain the following corollary:

Corollary 6.19. *Suppose that Assumption 1.10 holds. Then, there is a $\mathbf{UL} \cap \mathbf{coUL}$ algorithm that, on input $G = (V, E)$, outputs a weight function $w : E \to [n^{10}]$ such that $T_G(w) = 1$.*

Proof. We let the circuit family be $\{T_G\}_G$ as defined in Definition 6.7. This circuit family has a PCAPP algorithm with $\varepsilon = 1/10n$ by Lemma 6.10. This algorithm is in $\mathbf{P}$ by the fact that $\mathbf{UL} \subseteq \mathbf{P}$, and the circuits are ULE by Lemma 6.10. We then apply Theorem 6.18 and conclude. □

[32] It likewise gives a targeted PRG, but we do not use this property.

We can now prove the scaled-down result.

Theorem 1.11. *Suppose that Assumption 1.10 holds. Then* $\mathbf{NL} = \mathbf{UL}$.

Proof. Recall that (Theorem 6.14) a complete problem for $\mathbf{NL}$ is to, given (G, s, t) of size n, decide $s \to t$ connectivity in G. Given such input, we run the algorithm of Theorem 6.6 on (G, w) (and every time the algorithm queries the input weight function w, we query the unambiguous machine of Corollary 6.19). Correctness follows as w induces USPs, and hence Theorem 6.6 will return $\texttt{CONN}$ on exactly one guess sequence if $s \to t$ is connected and always return $\perp$ otherwise, and the machine is unambiguous as desired. □

7 Certified Derandomization

In this section, we explore *certified derandomization*, a notion that was originally introduced by Pyne, Ran, and Zhan [PRZ23] in the context of derandomizing $\mathbf{BPL}$.

We first formally define certified derandomization and its variants in Section 7.1. In Section 7.2, we prove Theorem 1.15, i.e., the equivalence between certified derandomization and $\mathbf{prBPP} = \mathbf{prZPP}$. In Section 7.3, we explore the connection between a variant of certified derandomization (that is more similar to the original definition in [PRZ23]) and the derandomization of $\mathbf{prBPP}$. Finally, we compare certified derandomization with a relaxed notion called *property-aided derandomization* and prove an oracle separation between them in Section 7.4.

7.1 Definitions of Certified Derandomization and its Variants

We start by giving the formal version of the more general definition of certified derandomization that we suggested in Section 1.4.

Definition 7.1 (certified derandomization)**.** Let $\ell = \ell(n) = 2^{o(n)}$,[33] $s = s(n)$, $\delta_0 = \delta_0(n)$, $\delta_1 = \delta_1(n)$, and $\mathcal{P} = \{\mathcal{P}_n \subseteq \{0,1\}^\ell\}_{n\in\mathbb{N}}$ be a property such that $\mathcal{P} \in \mathbf{coNP}$ and $\mathcal{P}_n \neq \varnothing$ for every n. An algorithm A is said to be a $\mathsf{certified}$ (δ_0, δ_1)-$\mathsf{derandomization}$ algorithm for s-size circuits using $\mathcal{P}$ if there exists a verifier V for $\mathcal{P}$ such that for every size-s circuit $C : \{0,1\}^n \to \{0,1\}$ and every $\tau \in \{0,1\}^\ell$,

$$A(C,\tau) = \begin{cases} 1 & \tau \in \mathcal{P} \wedge \Pr_x[C(x) = 1] \geq 1 - \delta_1; \\ 0 & \tau \in \mathcal{P} \wedge \Pr_x[C(x) = 1] \leq \delta_0; \\ (\perp, w) \text{ s.t. } V(\tau, w) = 0, \text{ or } 1 & \tau \notin \mathcal{P} \wedge \Pr_x[C(x) = 1] \geq 1 - \delta_1; \\ (\perp, w) \text{ s.t. } V(\tau, w) = 0, \text{ or } 0 & \tau \notin \mathcal{P} \wedge \Pr_x[C(x) = 1] \leq \delta_0; \\ 0,\ 1,\ \text{or } (\perp, \{0,1\}^*) & \text{otherwise.} \end{cases}$$

For simplicity, we may drop the parameter s if $s(n) = O(n)$, and drop the parameter δ_i if $\delta_i = 1/6$.

For a typical circuit class $\mathcal{C}$, a certified derandomization algorithm for $\mathcal{C}$ is defined as in Definition 7.1 while the input circuit C is restricted to a $\mathcal{C}$-circuit instead of a general circuit.

[33] The reason that we force $\ell = 2^{o(n)}$ is to ensure that the certified derandomization algorithm A runs in non-trivial time: A brute-force enumeration over all 2^n inputs trivially solves CAPP on n-input circuits in $2^{O(n)}$ time.

Two illustrative examples. The formulation of certified derandomization in [PRZ23] uses the specific property of hard truth-tables. That is, it uses $\mathcal{P} = \mathcal{P}^{\mathsf{cc}}_{\varepsilon}$, where

$$\mathcal{P}^{\mathsf{cc}}_{\varepsilon} \triangleq \{f : \{0,1\}^m \to \{0,1\} \mid f \text{ requires } 2^{\varepsilon m} \text{ size circuits}\},$$

for some $m = O(\log n)$.

As a more general example, we can also define $\mathcal{P}$ as the solutions to a *uniform range avoidance problem* [Kor21; RSW22]. Specifically, let $\ell = \text{poly}(n)$ and let $g\colon \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time computable function mapping $|x|$ bits to $|x|+1$ bits. We define $\mathcal{P}^g = \{\mathcal{P}_n \subseteq \{0,1\}^\ell\}_{n\in\mathbb{N}}$ as the strings outside of the range of g, i.e.,

$$\mathcal{P}^g_n \triangleq \{y \in \{0,1\}^\ell \mid g^{-1}(y) = \varnothing\}.$$

Note that to verify that $y \notin \mathcal{P}^g_n$, it suffices to find a string $x \in \{0,1\}^{\ell-1}$ such that $g(x) = y$.

This is a generalization of $\mathcal{P}^{\mathsf{cc}}$, since if we choose $\ell = 2^m$ and $g : \{0,1\}^{\ell-1} \to \{0,1\}^\ell$ as the function that outputs the truth table of a circuit of size at most $2^{\varepsilon m}$, then $\mathcal{P}^{\mathsf{cc}}_{\varepsilon} = \mathcal{P}^g$.[34] Moreover, by instantiating g with different functions, we can obtain other properties $\mathcal{P}^g$ consisting of objects typically considered in natural explicit construction problems, such as rigid matrices and strongly explicit two-source extractors (see [Kor21] for more details).

Certified derandomization without printing certificates. In Definition 7.1, the certified derandomization algorithm is required to output a certificate w witnessing the fact that $\tau \notin \mathcal{P}$ (i.e., w such that $V(\tau, w) = 0$). One can also consider a relaxed version, in which finding an explicit certificate is not necessary – i.e., if $\tau \notin \mathcal{P}$, the algorithm is allowed to simply output $\perp$.

Definition 7.2 (certified derandomization without printing certificates)**.** Let $\ell = \ell(n) = 2^{o(n)}$, $s = s(n)$, $\delta_0 = \delta_0(n)$, $\delta_1 = \delta_1(n)$, and $\mathcal{P} = \{\mathcal{P}_n \subseteq \{0,1\}^\ell\}_{n\in\mathbb{N}}$ be a property such that $\mathcal{P}_n \neq \varnothing$ for every n. An algorithm A is said to be a decision certified (δ_0, δ_1)-derandomization algorithm for size-s circuits using $\mathcal{P}$ if for every size-s circuit $C : \{0,1\}^n \to \{0,1\}$ and every $\tau \in \{0,1\}^\ell$,

$$A(C,\tau) = \begin{cases} 1 & \tau \in \mathcal{P} \wedge \Pr_x[C(x)=1] \geq 1-\delta_1; \\ 0 & \tau \in \mathcal{P} \wedge \Pr_x[C(x)=1] \leq \delta_0; \\ \perp \text{ or } 1 & \tau \notin \mathcal{P} \wedge \Pr_x[C(x)=1] \geq 1-\delta_1; \\ \perp \text{ or } 0 & \tau \notin \mathcal{P} \wedge \Pr_x[C(x)=1] \leq \delta_0; \\ 0,\ 1,\ \text{or } \perp & \text{otherwise.} \end{cases}$$

As in Definition 7.1, we may drop s when it is linear, drop δ_i if $\delta_i = 1/6$, and we define a decision certified derandomization for a typical class $\mathcal{C}$ in the natural way.

One may think of the notion in Definition 7.2 as a relaxed version of certified derandomization, since the algorithm guarantees that a certificate w *exists* (even if it does not explicitly print a certificate). Indeed, this relaxed notion does not depend on a particular verifier V for $\mathcal{P}$.[35] Alternatively, one may compare the notion in Definition 7.2 to the definition of **prZPP**, and think

[34] This is essentially the reduction from the problem of finding hard truth tables to the range avoidance problem; see [Kor21; RSW22]. Note that we only need $O(2^{\varepsilon m} \cdot \varepsilon m)$-bits to encode a circuit of size $2^{\varepsilon m}$ in the input, and the function g will simply ignore the last $\ell - 1 - 2^{\varepsilon m}$ input bits.

[35] In fact, the definition does not even require that $\mathcal{P} \in$ **coNP**. However, the definition implies that a certain superset $\mathcal{P}' \supseteq \mathcal{P}$ satisfies $\mathcal{P}' \in$ **coNP**. To see this, let A and $\mathcal{P}$ that satisfy Definition 7.2, and assume for a moment that for every $\tau \notin \mathcal{P}$ there exists C for which $A(C,\tau) = \perp$. Then, $\mathcal{P} \in$ **coNP**, since to certify that $\tau \notin \mathcal{P}$ it suffices to provide C such that $A(C,\tau) = \perp$. Now, if the assumption (that for every $\tau \notin \mathcal{P}$ there is C for which $A(C,\tau) = \perp$) does not hold, we can extend $\mathcal{P}$ to also include the strings τ that violate the assumption, and this yields the superset $\mathcal{P}' \in$ **coNP** such that A is a decision certified derandomization algorithm with $\mathcal{P}'$. Indeed, it is possible that $\mathcal{P}'$ is trivial (i.e., includes all strings), but this means that A is just a CAPP algorithm. The definition is meaningful (i.e., does not collapse to solving CAPP) when $\mathcal{P}' \in$ **coNP** is not trivial.

of the notion as imposing stricter conditions than in the definition of **prZPP**. Specifically, for a problem $\Pi = (\Pi^{\mathsf{YES}}, \Pi^{\mathsf{NO}}) \in \mathbf{prZPP}$, for every input $x \in \{0,1\}^*$ there is a dense set $\mathcal{P}_x$ such that an algorithm can guess τ, test whether $\tau \in \mathcal{P}_x$, and if the test passes, use τ to decide whether $x \in \Pi^{\mathsf{YES}}$ or $x \in \Pi^{\mathsf{NO}}$. The main difference between this view Definition 7.2 is that in the latter we require a single "global" property $\mathcal{P}$ that does not depend on x.

We will show (in Theorem 7.4) that the two notions of certified derandomization are in fact equivalent, and both are equivalent to **prBPP** = **prZPP**.

7.2 Certified Derandomization and prBPP = prZPP

In this subsection, we show that **prBPP** = **prZPP** if and only if there is a certified derandomization algorithm using some dense property. The notion of a dense property $\mathcal{P} = \{\mathcal{P}_n \subseteq \{0,1\}^\ell\}_{n\in\mathbb{N}}$ here means that there is a constant $\delta > 0$ such that for every $n \in \mathbb{N}$ it holds that $\Pr_{z\in\{0,1\}^\ell}[z \in \mathcal{P}_n] \geq \delta$.[36]

Proof idea. A certified derandomization algorithm using any dense property $\mathcal{P} \in \mathbf{coNP}$ immediately implies that **prBPP** = **prZPP**: This is since one can randomly generate strings $\tau \in \{0,1\}^\ell$ until the certified derandomization algorithm fails to provide a witness for $\tau \notin \mathcal{P}$.[37]

The main interesting direction is that **prBPP** = **prZPP** implies certified derandomization. Since we have the freedom of choosing the property $\mathcal{P}$ for certified derandomization, a natural idea is to define a property that consists of a *pseudorandom distribution* $\mathbf{D}$ over $\{0,1\}^n$ of polynomial size (say $|\mathbf{D}| = n^5$) that fools every linear-size circuit C. However, it is unclear how the deterministic certified derandomization algorithm could identify that a given distribution fails to fool the circuit it needs to derandomize, let alone provide a witness for the failure (i.e. the non-membership of $\mathcal{P}$).

The key idea to deal with the challenge utilizes the precondition **prBPP** = **prZPP**: Observe that the decision problem of verifying whether a distribution $\mathbf{D}$ fools a circuit C is in **prBPP**, and thus it is also in **prZPP**. This means that any random tape that makes the **prZPP** machine halt and accept is a witness that $\mathbf{D}$ fools C, and the witness can be verified by a deterministic algorithm. If we define $\mathcal{P}$ as the set of pseudorandom distributions $\mathbf{D}$ attached with such a witness of its pseudorandomness, we can then resolve the aforementioned challenge and obtain a certified derandomization algorithm.

The only remaining issue is that the witness depends on the circuit C, while in certified derandomization the property $\mathcal{P}$ has to be *independent* of the circuit we want to derandomize. Nevertheless, as the witness is the random tape of a **prZPP** algorithm, a random string is likely to be a correct witness that $\mathbf{D}$ fools C for any specific $\mathbf{D}$ and C. Therefore, if we randomly sample $w_1, w_2, \ldots, w_t$ for some large $t = n^{O(1)}$, with high probability, there is an $i \in [t]$ such that w_i is a witness that $\mathbf{D}$ fools C for every $\mathbf{D}$ (of size n^5) and every linear-size circuit C; that is, $(w_1, \ldots, w_t)$ serves as a *universal witness* of pseudorandomness that works for every $\mathbf{D}$ and C. By defining $\mathcal{P}$ as the set of pseudorandom distribution $\mathbf{D}$ (of size n^5) together with such a universal witness, we can obtain a certified derandomization algorithm using the property $\mathcal{P}$.

Proof of the equivalence. For the proof we will need the following standard encoding of sparse Boolean strings (see, e.g., [GII+19; Kor21]).

[36] We could equivalently define a dense property as one with $\delta = 1/\operatorname{poly}(n)$.

[37] Indeed, Pyne, Raz, and Zhan [PRZ23] gave a new proof of Nisan's [Nis93] result that **BPL** $\subseteq$ **ZP*L**, based on this idea and on their certified derandomization for read-once branching programs (where **ZP*L** is the class of zero-error randomized logspace algorithms with two-way access to the random tape).

Lemma 7.3. *For any $\varepsilon \in (0, 1/2)$, there is a pair of maps $\Phi_d : \{0,1\}^{n-\varepsilon^2 n+\log n} \to \{0,1\}^n$ and $\Phi_e : \{0,1\}^n \to \{0,1\}^{n-\varepsilon^2 n+\log n}$ computable in $\mathrm{poly}(n)$ time such that for every string x of Hamming weight at most $n/2 - \varepsilon n$, $\Phi_d(\Phi_e(x)) = x$.*

Theorem 7.4 (certified derandomization vs **prBPP** = **prZPP**)**.** *The following statements are equivalent.*

(1) **prBPP** = **prZPP**.

*(2) There is a dense property $\mathcal{P} \in$ **coNP** and a deterministic polynomial-time certified derandomization algorithm using $\mathcal{P}$ with parameter $\ell(n) = \mathrm{poly}(n)$.*

(3) There is a dense property $\mathcal{P}$ and a deterministic polynomial-time decision certified derandomization algorithm using $\mathcal{P}$ with parameter $\ell(n) = \mathrm{poly}(n)$.

Proof. (1)⇒(2). Suppose that **prBPP** = **prZPP**, then there is a zero-error polynomial-time algorithm A that solves the following promise problem $\Pi = (\Pi^{\mathsf{YES}}, \Pi^{\mathsf{NO}}) \in$ **prBPP**: Given a circuit $C : \{0,1\}^n \to \{0,1\}$ of linear size and a multiset $D \subseteq \{0,1\}^n$,

- $(C, D) \in \Pi^{\mathsf{YES}}$ if $|\mathbb{E}_{x\in\{0,1\}^n}[C(x)] - \mathbb{E}_{x\in D}[C(x)]| \le 0.01$.
- $(C, D) \in \Pi^{\mathsf{NO}}$ if $|\mathbb{E}_{x\in\{0,1\}^n}[C(x)] - \mathbb{E}_{x\in D}[C(x)]| \ge 0.02$.

By the definition of zero-error algorithms and Markov's inequality, we know that there is a polynomial $t = t(n)$ such that for every input $(C, D) \in \Pi^{\mathsf{YES}} \cup \Pi^{\mathsf{NO}}$ of length $\mathrm{poly}(n)$, all but a $1/10$-fraction of random tapes $w \in \{0,1\}^t$ will make $A(C, D; w)$ halt in time t.

Let $\ell(n) = n^6 + n^{10} \cdot t = \mathrm{poly}(n)$. We define the property $\mathcal{P} \in$ **coNP** as follows. Let $D \in (\{0,1\}^n)^{n^5}$ be parsed as a sequence $d_1, \ldots, d_{n^5}$ of n^5 length-n strings, and $W \in (\{0,1\}^t)^{n^{10}}$ be parsed as a sequence $w_1, \ldots, w_{n^{10}}$ of length-t strings. We define $(D, W) \in \mathcal{P}$ if and only if for every circuit $C : \{0,1\}^n \to \{0,1\}$ of linear size, there is some $i \in [n^{10}]$ such that $A(C, D; w_i)$ halts in time t and $A(C, D; w_i) = 1$.

Claim 7.5. *$\mathcal{P}$ is a $4/5$-dense property.*

Proof. Fix any circuit $C : \{0,1\}^n \to \{0,1\}$. Let $\gamma = \mathbb{E}_{x\in\{0,1\}^n}[C(x)]$, and $D = (d_1, \ldots, d_{n^5}) \in (\{0,1\}^n)^{n^5}$ be sampled uniformly at random. By Hoeffding's inequality (i.e., Theorem 3.10), we know that

$$\Pr_D\left[\left|\mathop{\mathbb{E}}_{x\leftarrow D}[C(x)] - \gamma\right| > 0.01\right] \le \exp\left(-\Omega(n^5)\right).$$

By union bounding over all circuits of linear size, all but a $1/10$-fraction of choices for D satisfy the following: For every linear-size circuit C, it holds that $(C, D) \in \Pi^{\mathsf{YES}}$.

Now, fix any (C, D) in the promise of Π. If we choose n^{10} random tapes $w_1, w_2, \ldots, w_{n^{10}} \in \{0,1\}^t$ uniformly at random, the probability that $A(C, D; w_i)$ does not halt in time t for every $i \in [n^{10}]$ is at most $\exp(-\Omega(n^{10}))$. By union bounding over all circuits C of size n^2 and all multisets D of size n^5, we deduce that all but a $1/10$-fraction of sequences of random tapes $w_1, w_2, \ldots, w_{n^{10}} \in \{0,1\}^t$ satisfy the following: For every linear-size circuit $C : \{0,1\}^n \to \{0,1\}$ and every multiset $D \subseteq \{0,1\}^n$ of size n^5, there is an $i \in [n^{10}]$ such that $A(C, D; w_i)$ halts in time t.

By union bounding over a choice of W and a choice of D, we deduce that at least a $4/5$-fraction of (D, W) are in $\mathcal{P}$. □

Given any linear-size circuit $C : \{0,1\}^n \to \{0,1\}$ and $\tau \in \{0,1\}^\ell$, the certified derandomization algorithm using $\mathcal{P}$ works as follows.

- Parse $\tau = (D, W)$ as described above, where $D = (d_1, \ldots, d_{n^5})$ and $W = (w_1, \ldots, w_{n^{10}})$.
- If for some $i \in [n^{10}]$, $A(C, D; w_i)$ halts in time t and $A(C, D; w_i) = 1$, the algorithm outputs 1 if and only if $\mathbb{E}_{x \in D}[C(x)] \geq 1/2$.
- Otherwise, the algorithm outputs $(\perp, C)$.

The correctness of the algorithm follow from a case analysis:

- If C either accepts at least a 2/3-fraction of its inputs or rejects at least a 2/3-fraction of its inputs, and $A(C, D; w_i)$ halts in time t and $A(C, D; w_i) = 1$ for some $i \in [n^{10}]$, we know by the perfect correctness of the algorithm A that $(C, D) \notin \Pi^{\mathsf{NO}}$. In other words, $|\mathbb{E}_{x \in \{0,1\}^n}[C(x)] - \mathbb{E}_{x \in D}[C(x)]| < 0.02$, and therefore the algorithm correctly solves CAPP for C based on whether $\mathbb{E}_{x \in D}[C(x)] \geq 1/2$.
- If C either accepts a 2/3-fraction of its inputs or rejects a 2/3-fraction of its inputs, and there is no $i \in [n^{10}]$ such that $A(C, D; w_i)$ halts in time t and $A(C, D; w_i) = 1$, then C is a witness for $(D, W) \notin \mathcal{P}$.

(2)⇒(3). Trivial.

(3)⇒(1). Suppose that there is a dense property $\mathcal{P}$ and a decision certified derandomization algorithm using $\mathcal{P}$ with parameter $\ell(n) = \mathrm{poly}(n)$. For every promise problem $\Pi = (\Pi^{\mathsf{YES}}, \Pi^{\mathsf{NO}}) \in \mathbf{prBPP}$ and probabilistic Turing machine M that solves Π, the following probabilistic algorithm will solve Π with zero error in expected polynomial time:

- Given any $x \in \Pi^{\mathsf{YES}} \cap \Pi^{\mathsf{NO}}$, construct a linear-size circuit $C_x : \{0,1\}^m \to \{0,1\}$ for some $m = \mathrm{poly}(n)$ such that

$$x \in \Pi^{\mathsf{YES}} \Rightarrow \Pr_r[C(r) = 1] \geq 2/3,$$
$$x \in \Pi^{\mathsf{NO}} \Rightarrow \Pr_r[C(r) = 0] \geq 2/3.$$

- Randomly generate $\tau \in \{0,1\}^{\ell(m)}$ and simulate the certified derandomization algorithm that uses $\mathcal{P}$. Repeat until we generate some τ such that the certified derandomization algorithm outputs $b \in \{0,1\}$.
- The algorithm accepts if and only if $b = 1$.

The correctness follows from the correctness of the certified derandomization algorithm, and the running time follows from the density of the property $\mathcal{P}$. □

7.3 LossyCode and Certified Derandomization

We recall the definition of the problem $\mathsf{LossyCode}$, introduced by Korten [Kor22]. Intuitively, the problem $\mathsf{LossyCode}$ calls for finding *incompressible strings*, with respect to a fixed efficient compression algorithm and a fixed efficient decompression algorithm (that tests whether or not the compression algorithm succeeded).

Problem 1.16 ($\mathsf{LossyCode}$)**.** Given a pair of circuits $C : \{0,1\}^n \to \{0,1\}^m$, $D : \{0,1\}^m \to \{0,1\}^n$, where $m < n$, find a string $x \in \{0,1\}^n$ such that $D(C(x)) \neq x$.

The promise problems reducible to LossyCode are in **prZPP**, as since $m < n$ a random string $x \in \{0,1\}^n$ satisfies $D(C(x)) \neq x$ with probability at least $1/2$, and this can be verified deterministically. Korten [Kor22] observed that if LossyCode is hard for **prBPP** (under deterministic polynomial-time reductions), then **prBPP** = **prZPP**.[38] We show a stronger result, coupled with a converse.

We define the complexity class **LOSSY** (resp. **FLOSSY**) as the set of languages (resp. search problems) that are reducible in deterministic polynomial time to LossyCode. The promise version **prLOSSY** is defined analogously.

Definition 7.6 (**LOSSY**)**.** The class **LOSSY** (resp., **prLOSSY**) it the set of languages (resp., promise problems) decidable in deterministic polynomial time with oracle access to a function solving the search problem LossyCode. The class **FLOSSY** is the set of search problems that can be solved in deterministic polynomial time with oracle access to a function solving LossyCode.

Definition 7.6 does not place any restrictions on the type of reduction to LossyCode (other than being in deterministic polynomial time), but it turns out that even a single query to LossyCode suffices to capture the general case; see Lemma 7.9.

The question of whether LossyCode is **prBPP**-hard can thus be phrased as asking if **prBPP** = **prLOSSY**. The following theorem asserts that **prBPP** = **prLOSSY** *if and only if* there is an algorithm for certified derandomization using the property of truth tables with high circuit complexity.

Theorem 7.7 (**prBPP** = **prLOSSY** $\iff$ certified derandomization with hard truth-tables)**.** *The following statements are equivalent.*

(1) **prBPP** = **prLOSSY**.

(2) There is $\varepsilon \in (0,1)$ *and a deterministic polynomial-time certified derandomization algorithm using* $\mathcal{P}^{\mathsf{cc}}_{\varepsilon}$.

(3) There is a deterministic polynomial-time certified derandomization algorithm using $\mathcal{P}^{g}$ *for some polynomial-time computable* g.

Note that the equivalence between (2) and (3) shows that certified derandomization using hard truth-tables is implied by certified derandomization using any uniform range avoidance property. The latter is more general, as explained in Section 7.1. In fact, many well-known explicit construction problems reduce to the uniform range avoidance problem (see, e.g., [Kor21; RSW22]). The equivalence between Items (2) and (3) can be viewed as analogous to the $\mathbf{FP}^{\mathbf{NP}}$ reduction in [Kor21] of range avoidance to the problem of finding hard truth tables.[39]

Remark 7.8. Alternatively, we show that **prBPP** = **prLOSSY** is also equivalent to "non-black-box" derandomization of Yao's transformation in **FLOSSY** using an idea in [Kor22]; see Appendix C.2 for the definitions and the proof of the result.

Proof of Theorem 7.7. Note that (2) $\Rightarrow$ (3) is trivial. We will prove that (1) $\Rightarrow$ (2) and that (3) $\Rightarrow$ (1), in this order.

[38] Korten also proved that a version of LossyCode where the compression algorithm is allowed to be randomized, called R-LossyCode, is indeed hard for **prBPP**.

[39] Korten [Kor21] phrases the result as showing that circuit lower bounds (i.e. the problem of constructing hard truth tables) is the hardest instance for range avoidance, and thus implies solutions to many well-known explicit construction problems. Certified derandomization models derandomization using a property with a *deterministic reconstructive argument*, and thus the equivalence between (2) and (3) shows that in terms of derandomization with deterministic reconstructive argument, hard truth tables are as useful as any range avoidance property.

(1) $\Rightarrow$ (2). Suppose that there is a polynomial-time oracle machine A that solves CAPP given oracle access to $\mathsf{LossyCode}$. Consider the following certified derandomization algorithm using $\mathcal{P}^{\mathsf{cc}}_{\varepsilon}$. Given any linear-size circuit $C : \{0,1\}^n \to \{0,1\}$ and a candidate hard truth table $\tau \in \{0,1\}^{2^k}$, where $k = O(\log n)$, we will simulate $A^{\mathsf{LossyCode}}$, while trying to answer queries to $\mathsf{LossyCode}$ using τ (in a way that will be specified below). The point is that if we succeed in answering all queries, then we successfully simulated $A^{\mathsf{LossyCode}}$; and if we fail to answer a query, then we will be able to output a small circuit whose truth-table is τ.

Let us now explain how to use τ in order to try and answer a query $q = (C', D')$ to $\mathsf{LossyCode}$. Since we can always pad the input and output of C', D' with dummy bits, we can assume without loss of generality that $C' : \{0,1\}^{t+1} \to \{0,1\}^t$ and $D' : \{0,1\}^t \to \{0,1\}^{t+1}$ are linear-size circuits for some $t = \mathrm{poly}(n)$ being a power of two.

Single-bit to length-doubling compression. Our first step is to reduce $\mathsf{LossyCode}$ for C' and D' compressing and decompressing with one bit, to $\mathsf{LossyCode}$ for C' and D' that halve and double the input length (respecitvely).

Let $C'_1 \triangleq C'$, $D'_1 \triangleq D'$. We define $C'_i : \{0,1\}^{t+i} \to \{0,1\}^t$ and $D'_i : \{0,1\}^t \to \{0,1\}^{t+i}$ inductive as follows: for $x \in \{0,1\}^{t+i}$, $b \in \{0,1\}$, and $z \in \{0,1\}^t$,

$$\begin{aligned} C'_{i+1}(x \circ b) &\triangleq C'(C'_i(x) \circ b), \\ D'_{i+1}(z) &\triangleq D'_i(D'(z)_{\le t}) \circ D'(z)_{t+1}. \end{aligned}$$

Let $C'' \triangleq C'_t$ and $D'' \triangleq D'_t$. Note that for every string x and $b \in \{0,1\}$ such that $D'_{i+1}(C'_{i+1}(x \circ b)) \neq x \circ b$, denoting $\hat{x} \triangleq C'_i(x) \circ b$, either $D'(C'(\hat{x})) \neq \hat{x}$, or $D'_i(C'_i(x)) \neq x$. This is because if none of the two cases hold, then

$$\begin{aligned} D'_{i+1}(C'_{i+1}(x \circ b)) &= D'_i(D'(C'(C'_i(x) \circ b))_{\le t}) \circ D'(C'(C'_i(x) \circ b))_{t+1} \\ &= D'_i((C'_i(x) \circ b)_{\le t}) \circ (C'_i(x) \circ b)_{t+1} \\ &= D'_i(C'_i(x)) \circ b \\ &= x \circ b. \end{aligned}$$

Note that, given C' and D', we can compute C'' and D'' in time $O(t^2)$. Also, relying on the property above, given (C', D') and $y \in \{0,1\}^{2t}$ such that $D''(C''(y)) \neq y$, we can find in polynomial time a string $x \in \{0,1\}^t$ such that $D'(C'(x)) \neq x$.

Reduction to exponentially long strings via a binary tree. Next, we reduce $\mathsf{LossyCode}$ on (C'', D'') to finding a string of length 2^k on which an exponential compression and decompression algorithm (based on C'', D'') fails.

Let $C''_0 \triangleq C''$, $D''_0 \triangleq C''$, and let $k' = k - \log(t)$. For $i \in [k']$, we define $C''_i : \{0,1\}^{t \cdot 2^i} \to \{0,1\}^t$ and $D''_i : \{0,1\}^t \to \{0,1\}^{t \cdot 2^i}$ inductively as follows: for all $x, y \in \{0,1\}^{t \cdot 2^i}$ and $z \in \{0,1\}^t$,

$$\begin{aligned} C''_{i+1}(x \circ y) &\triangleq C''(C''_i(x) \circ C''_i(y)), \\ D''_{i+1}(z) &\triangleq D''_i(D''(z)_{\le t}) \circ D''_i(D''(z)_{>t}). \end{aligned}$$

Note that, given (C'', D'') and i and $x \in \{0,1\}^{t \cdot 2^i}$, we can compute $C''_i(x)$ in polynomial time; similarly, given (C'', D'') and i and $z \in \{0,1\}^t$, we can compute $D''_i(z)$ in polynomial time. Also note that for every $x, y \in \{0,1\}^{t \cdot 2^i}$ such that $D''_{i+1}(C''_{i+1}(x \circ y)) \neq x \circ y$, letting $z \triangleq C''_i(x) \circ C''_i(y)$, either $D''(C''(z)) \neq z$, or $D''_i(C''_i(x)) \neq x$, or $D''_i(C''_i(y)) \neq y$. Hence, there is a polynomial-time

algorithm that gets input (C'', D'') and $i \in [k]$ and $x \circ y \in \{0,1\}^{t \cdot 2^i}$ satisfying $D''_i(C''_i(x \circ y)) \neq x \circ y$, and finds a string $z \in \{0,1\}^t$ such that $D''(C''(z)) \neq z$ (i.e., the algorithm applies the property above iteratively, and uses the fact that C''_i and D''_i are efficiently computable).

Simulating the LossyCode oracle with τ. Now we describe how we try to answer a query (C', D') to LossyCode. We construct $\hat{C}' \triangleq C''_{k'}$ and $\hat{D}' \triangleq D''_{k'}$, where the input length of $\hat{C}'$ (i.e. the output length of $\hat{D}'$) is 2^k, and guess $\tau \in \{0,1\}^{2^k}$.

If $\hat{D}'(\hat{C}'(\tau)) \neq \tau$, as explained above, we can find in polynomial time a string $x \in \{0,1\}^t$ such that $D(C(x)) \neq x$; then, we answer the oracle query by x.

Otherwise, we will find in time $\mathrm{poly}(n)$ a circuit of size $2^{\varepsilon \cdot k}$ whose truth table is τ. As explained above, given (C', D') we can compute D'' and D'' in time $O(t^2)$; thus, our algorithm can construct a circuit of size $\tilde{O}(t^2)$ (with D' hard-wired) computing D''.

Consider the following circuit $E : \{0,1\}^k \to \{0,1\}$: Given any $u = (u_1, u_2, \ldots, u_k) \in \{0,1\}^k$, it parses $u = v \circ j$, where $v \in \{0,1\}^{k'}$ and $j \in [t]$, and for $i \in [k']$ it iteratively computes the following function. Denoting $\hat{\tau}_0 \triangleq \hat{C}'(\tau) \in \{0,1\}^t$, we define

$$\hat{\tau}_{i+1} \triangleq \begin{cases} D''(\hat{\tau}_i)_{\leq t} & v_i = 0, \\ D''(\hat{\tau}_i)_{>t} & v_i = 1, \end{cases}$$

Finally, the circuit E outputs the j-th bit of $\hat{\tau}_{k'}$. Since $\hat{D}'(\hat{C}'(\tau)) = \tau$ and the circuit simulates the computation of $\hat{D}'$, we know that $E(u) = \tau_u$ for every $u \in \{0,1\}^k$. Moreover, the circuit E is of size $\tilde{O}(k' \cdot t^2) = \tilde{O}(t^2)$, and can be constructed in polynomial time. If we choose $k = O(\log n)$ such that $2^k \geq t^{5/\varepsilon}$, then for sufficiently large t, the size of E is at most $2^{\varepsilon k}$.

(3) $\Rightarrow$ (1). Assume that for some polynomial-time computable function $g : \{0,1\}^{\ell-1} \to \{0,1\}^\ell$, there is a deterministic polynomial-time certified derandomization algorithm A using $\mathcal{P}^g$. We now describe a reduction of CAPP to LossyCode.

Let $C : \{0,1\}^n \to \{0,1\}$ be a linear-size circuit that either accepts at least $2/3$ of its inputs or rejects at least a $2/3$ of its inputs. Consider the following circuits $C' : \{0,1\}^\ell \to \{0,1\}^{\ell-1}$ and $D' : \{0,1\}^{\ell-1} \to \{0,1\}$.

- $C'(\tau)$ simulates the certified derandomization algorithm $A(C, \tau)$. If $A(C, \tau) \in \{0,1\}$, then $C'(\tau)$ outputs $0^{\ell-1}$. Otherwise, $A(C, \tau)$ outputs a witness $w \in \{0,1\}^{\ell-1}$ that $\tau \notin \mathcal{P}^g$ (i.e. $g(w) = \tau$); in this case $C'(\tau)$ outputs w.
- $D'(w) \triangleq g(w)$.

Let τ be a solution to the LossyCode instance (C', D'), i.e., $D'(C'(\tau)) \neq \tau$. It follows that $A(C, \tau) \in \{0,1\}$ (since whenever $A(C, \tau) \notin \{0,1\}$ we have that $D'(C'(\tau)) = D'(w) = g(w) = \tau$). Since certified derandomization is always correct if it the circuit is in the promise and it outputs 0 or 1, we can figure out whether $\Pr_x[C(x) = 1] \geq 2/3$ or $\Pr_x[C(x) = 1] \leq 1/3$. □

A single query suffices for the definition of LOSSY. It is implicit in the proof of Theorem 7.7 that if CAPP polynomial-time reduces to LossyCode, then CAPP reduces to LossyCode in polynomial-time with a single query.[40] As mentioned above, this generalizes to *every* problem: Every problem that is polynomial-time reducible to LossyCode is also reducible to LossyCode with a single query.

[40] Notice that in the proof of (1) $\Rightarrow$ (2) the reduction can have multiple queries, while in the proof of (3) $\Rightarrow$ (1) the reduction we constructed has only one query.

Lemma 7.9. *For every language L (resp. search problem P), L (resp. P) can be solved in deterministic polynomial time with oracle access to a function solving* LossyCode *if and only if it can be solved in deterministic polynomial time with a single oracle query to a function solving* LossyCode.

Proof. We only consider the case of a language L, and the proof clearly generalizes to any search problem. Suppose that there is a polynomial-time Turing machine M deciding L with a LossyCode oracle. We describe a reduction $\mathcal{R}$ from L to LossyCode with a single query.

Given any string x, $\mathcal{R}(x)$ outputs a compression circuit $C : \{0,1\}^\ell \to \{0,1\}^{\ell-1}$ and a decompression circuits $D : \{0,1\}^{\ell-1} \to \{0,1\}^\ell$ that works as follows.

- Let $m = m(n) = \mathrm{poly}(n)$ be the maximum input length of the LossyCode oracle queries made by $M(x)$. Let $\ell = \ell(n) \triangleq m^3$.
- Let $\tau_1, \tau_2, \ldots, \tau_{m^2} \in \{0,1\}^\ell$ be the input to the compression circuit C. It simulates the oracle Turing machine $M(x)$. Once there is a query (C', D') to the LossyCode oracle, without loss of generality say $C' : \{0,1\}^m \to \{0,1\}^{m-1}, D' : \{0,1\}^{m-1} \to \{0,1\}^m$ are linear-size circuits, it considers two cases.
  - If $D'(C'(\tau_i)) = \tau_i$ for every $i \in [m^2]$, it stops simulating $M(x)$ and outputs $(D', C'(\tau_1) \circ \cdots \circ C'(\tau_{m^2}))$, which can be encoded in $O(m) + m^3 - m^2 \le \ell - 1$ bits.
  - Otherwise, it continues simulating $M(x)$ by answering τ_i to the oracle query, where i is the smallest index such that $D'(C'(\tau_i)) \neq \tau_i$.
- The decompression circuit D parses its input as $(D', y_1 \circ y_2 \circ \cdots \circ y_{m^2})$, where D' is a linear-size circuit and $y_1, \ldots, y_{m^2} \in \{0,1\}^{m-1}$. It outputs $D'(y_1) \circ D'(y_2) \circ \cdots \circ D'(y_{m^2}) \in \{0,1\}^\ell$.

If we are given a solution $\tau \in \{0,1\}^\ell$ such that $D(C(\tau)) \neq \tau$, then it means if we simulate $M(x)$ by answering the oracle calls as described in the definition of the compression circuit C, the simulation will be successful.[41] □

As a consequence, the definition of **LOSSY** does not change if we limit the type of reduction to LossyCode even to polynomial-time reduction with only a single query.

7.4 Certified Derandomization and Property-Aided Derandomization

A property-aided derandomization is an algorithm for CAPP given a string that is *guaranteed* to have a certain property $\mathcal{P}$; in particular, standard hardness-vs-randomness framework [NW94; IW97] can be interpreted as a property-aided derandomization algorithm where $\mathcal{P}$ is the set of hard truth tables. Formally:

Definition 7.10. Let $\ell = \ell(n) = 2^{o(n)}$ and $s = s(n) = \mathrm{poly}(n)$ be parameters, and $\mathcal{P} = \{\mathcal{P}_n \subseteq \{0,1\}^\ell\}_{n\in\mathbb{N}}$ be a property such that $\mathcal{P} \neq \varnothing$ for every n. An algorithm A is said to be a $\mathcal{P}$-aided CAPP algorithm if for every size-$s(n)$ circuit $C : \{0,1\}^n \to \{0,1\}$ and every $\tau \in \mathcal{P}_n$,

$$A(C,\tau) = \begin{cases} 1 & \mathbb{E}[C(\mathbf{U}_n)] \ge 2/3; \\ 0 & \mathbb{E}[C(\mathbf{U}_n)] \le 1/3; \\ 0 \text{ or } 1 & \text{otherwise.} \end{cases}$$

[41] The proof is more or less utilizing Krajíček's gadget generator [Kra07]. Alternatively, one can also apply the binary tree approach in the proof of Theorem 7.7.

Property-aided derandomizaiton is similar to, but (possibly) easier than certified derandomization, as it does not need to deal with the situation where the given string does not have the property $\mathcal{P}$. Given that we already know how to construct property-aided derandomization algorithms, while certified derandomization is equivalent to $\mathbf{prBPP} = \mathbf{prZPP}$, it is natural to ask whether there is a barrier preventing us from improving the property-aided derandomization algorithm into a certified derandomization algorithm.

In this subsection, we will prove an oracle separation: There is an oracle relative to which certified derandomization is impossible using any property, while property-aided derandomization is possible (indeed, the standard hardness-vs-randomness framework [NW94; IW97; KM02] relativizes).

Property-aided derandomization in relativized worlds. We first introduce the notion of property-aided derandomization in ideal worlds by relativizing both the CAPP algorithm and the input circuits.

Definition 7.11. Let $\ell = \ell(n) = 2^{o(n)}$ and $s = s(n) = \mathrm{poly}(n)$ be parameters, $\mathcal{O} : \{0,1\}^* \to \{0,1\}$ be an oracle, and $\mathcal{P} = \{\mathcal{P}_n \subseteq \{0,1\}^\ell\}_{n\in\mathbb{N}}$ be a property such that $\mathcal{P} \neq \varnothing$ for every n. An oracle algorithm A is said to be a $\mathcal{P}$-aided CAPP algorithm relative to the oracle $\mathcal{O}$ if for every size-$s(n)$ $\mathcal{O}$-oracle circuit $C : \{0,1\}^n \to \{0,1\}$ and every $\tau \in \mathcal{P}_n$,

$$A^{\mathcal{O}}(C,\tau) = \begin{cases} 1 & \mathbb{E}[C^{\mathcal{O}}(\mathbf{U}_n)] \geq 2/3; \\ 0 & \mathbb{E}[C^{\mathcal{O}}(\mathbf{U}_n)] \leq 1/3; \\ 0 \text{ or } 1 & \text{otherwise.} \end{cases}$$

In addition, we can define the (relativized) circuit complexity property $\mathcal{P}^{\mathcal{O}}_{\varepsilon\text{-}\mathsf{cc}}$ as

$$\tau \in \mathcal{P}^{\mathcal{O}}_{\varepsilon\text{-}\mathsf{cc}} \iff \tau \text{ requires } \mathcal{O}\text{-oracle circuits of size } |\tau|^{\varepsilon}$$

for some fixed constant $\varepsilon \in (0,1)$. It is proved in [KM02] that the constructions of the Nisan-Wigderson [NW94] and Impagliazzo-Wigderson [IW97] PRGs relativize. Formally:

Theorem 7.12 ([KM02])**.** *The following holds for every oracle $\mathcal{O}$. For every $s(n) = \mathrm{poly}(n)$ and every constant $\varepsilon \in (0,1)$, there is an $\ell(n) = \mathrm{poly}(n)$ and a polynomial-time deterministic algorithm A that is a $\mathcal{P}^{\mathcal{O}}_{\varepsilon\text{-}\mathsf{cc}}$-aided CAPP algorithm relative to $\mathcal{O}$ with parameters (ℓ, s).*

Certified derandomization in relativized worlds. Similarly, we can define certified derandomization in relativized worlds.

Definition 7.13. Let $\ell = \ell(n) = 2^{o(n)}$ and $s = s(n) = \mathrm{poly}(n)$ be parameters, $\mathcal{O} : \{0,1\}^* \to \{0,1\}$ be an oracle, and $\mathcal{P} = \{\mathcal{P}_n \subseteq \{0,1\}^\ell\}_{n\in\mathbb{N}}$ be a property such that $\mathcal{P}_n \neq \varnothing$ for every n. An oracle algorithm A is said to be a decisional certified derandomization algorithm using the property $\mathcal{P}$ with respect to the oracle $\mathcal{O}$ if for every size-s $\mathcal{O}$-oracle circuit $C : \{0,1\}^n \to \{0,1\}$ and every $\tau \in \{0,1\}^\ell$,

$$A^{\mathcal{O}}(C,\tau) = \begin{cases} 1 & \tau \in \mathcal{P}^{\mathcal{O}} \wedge \mathbb{E}[C^{\mathcal{O}}(\mathbf{U}_n)] \geq 2/3; \\ 0 & \tau \in \mathcal{P}^{\mathcal{O}} \wedge \mathbb{E}[C^{\mathcal{O}}(\mathbf{U}_n)] \leq 1/3; \\ \perp \text{ or } 1 & \tau \notin \mathcal{P}^{\mathcal{O}} \wedge \mathbb{E}[C^{\mathcal{O}}(\mathbf{U}_n)] \geq 2/3; \\ \perp \text{ or } 0 & \tau \notin \mathcal{P}^{\mathcal{O}} \wedge \mathbb{E}[C^{\mathcal{O}}(\mathbf{U}_n)] \leq 1/3; \\ 0,\ 1,\ \text{or } \perp & \text{otherwise.} \end{cases}$$

The oracle separation. The following theorem shows that in a relativized world, certified derandomization is impossible.

Theorem 7.14. *There is an oracle $\mathcal{O}$ such that the following holds. For every $\ell = \ell(n) = 2^{o(n)}$, $s = s(n) \geq n^2$, and any property $\mathcal{P} = \{\mathcal{P}_n \subseteq \{0,1\}^\ell\}_{n\in\mathbb{N}}$ such that $\mathcal{P}_n \neq \varnothing$ for every n, there is no deterministic polynomial-time algorithm $A^{\mathcal{O}}$ that is a decisional certified derandomization algorithm using $\mathcal{P}$ relative to the oracle $\mathcal{O}$.*

We will use the following oracle construction due to Heller.

Lemma 7.15 (Heller [Hel86])**.** *There is an oracle $\mathcal{O}$ such that $L \in \mathbf{BPTIME}^{\mathcal{O}}[n]$ for some $\mathbf{E}^{\mathbf{NP}^{\mathcal{O}}}$-complete language L.*

Proof of Theorem 7.14. Let $\mathcal{O}$ be the oracle and L be the language in Lemma 7.15. Suppose, towards a contradiction, that for some $\ell = \ell(n) = 2^{o(n)}$ and property $\mathcal{P}$ satisfying $\mathcal{P} \cap \{0,1\}^{\ell(n)} \neq \varnothing$ for all n, there is a deterministic decision certified derandomization algorithm with respect to $\mathcal{O}$ using $\mathcal{P}$. We will show that $L \in \mathbf{NTIME}^{\mathcal{O}}[2^{o(n)}]$, leading to the collapse

$$\mathbf{NTIME}^{\mathcal{O}}[2^{O(n)}] \subseteq \mathbf{E}^{\mathbf{NP}^{\mathcal{O}}} \subseteq \mathbf{NTIME}^{\mathcal{O}}[2^{o(n)}],$$

and thus contradicts the nondeterministic time hierarchy that holds relative to any oracle (see, e.g., the proof in [AB09, Chapter 3.3].

Recall that $L \in \mathbf{BPTIME}^{\mathcal{O}}[n]$. Given any input $x \in \{0,1\}^n$, the nondeterministic machine works as follows. It first constructs an oracle circuit $C : \{0,1\}^m \to \{0,1\}$ of size $\tilde{O}(m) \leq s(m)$ for some $m = O(n)$ such that

$$x \in L \iff \mathbb{E}[C^{\mathcal{O}}(\mathbf{U}_m)] \geq 2/3, \quad x \notin L \iff \mathbb{E}[C^{\mathcal{O}}(\mathbf{U}_m)] \leq 1/3.$$

It nondeterministically guesses a string $\tau \in \{0,1\}^{\ell(m)}$, and accepts if and only if the certified derandomization algorithm given (C, τ) outputs 1. It is clear that the nondeterministic algorithm decides L if $\mathcal{P}$ is non-empty on every length ℓ, and it runs in time $\mathrm{poly}(2^{o(m)}) = 2^{o(n)}$. □

By combining Theorem 7.12 and Theorem 7.14, we prove the following oracle separation:

Corollary 7.16. *There is an oracle $\mathcal{O}$ such that the following properties holds.*

- *For every $\ell = \ell(n) = 2^{o(n)}$, $s = s(n)$, and every property $\mathcal{P} = \{\mathcal{P}_n \subseteq \{0,1\}^\ell\}_{n\in\mathbb{N}}$ such that $\mathcal{P}_n \neq \varnothing$ for every n, there is no deterministic polynomial-time algorithm for decisional certified derandomization using $\mathcal{P}$ with respect to $\mathcal{O}$ with parameters (ℓ, s)*
- *For every $s = s(n) = \mathrm{poly}(n)$ and every constant $\varepsilon \in (0,1)$, there is an $\ell = \mathrm{poly}(n)$ and a polynomial-time $\mathcal{P}^{\mathcal{O}}_{\varepsilon\text{-}\mathsf{cc}}$-aided CAPP algorithm relative to $\mathcal{O}$ with parameters (ℓ, s).*

Acknowledgements

The authors thank Oded Goldreich for a useful conversation about [GW00], and Lijie Chen, Dean Doron, and Ryan Williams for helpful conversations.

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: A modern approach*. Cambridge University Press, Cambridge, 2009.

[ACR98] Alexander E. Andreev, Andrea E. F. Clementi, and José D. P. Rolim. "A new general derandomization method". In: *Journal of the ACM* 45.1 (1998), pp. 179–213.

[ACR+99] Alexander E. Andreev, Andrea E. F. Clementi, José D. P. Rolim, and Luca Trevisan. "Weak Random Sources, Hitting Sets, and BPP Simulations". In: *SIAM J. Comput.* 28.6 (1999), pp. 2103–2116.

[AM08] Vikraman Arvind and Partha Mukhopadhyay. "Derandomizing the Isolation Lemma and Lower Bounds for Circuit Size". In: *RANDOM 2008, Boston, MA, USA, August 25-27*. Ed. by Ashish Goel, Klaus Jansen, José D. P. Rolim, and Ronitt Rubinfeld. 2008.

[ARZ99] Eric Allender, Klaus Reinhardt, and Shiyu Zhou. "Isolation, Matching, and Counting Uniform and Nonuniform Upper Bounds". In: *J. Comput. Syst. Sci.* 59.2 (1999), pp. 164–181.

[Bar89] David A. Mix Barrington. "Bounded-Width Polynomial-Size Branching Programs Recognize Exactly Those Languages in NC^1". In: *Journal of Computer and System Sciences* 38.1 (1989), pp. 150–164.

[BCK+14] Harry Buhrman, Richard Cleve, Michal Koucký, Bruno Loff, and Florian Speelman. "Computing with a full memory: catalytic space". In: *Proc. 46 Annual ACM Symposium on Theory of Computing (STOC)*. 2014, pp. 857–866.

[BDCG+92] Shai Ben-David, Benny Chor, Oded Goldreich, and Michel Luby. "On the theory of average case complexity". In: *Journal of Computer and System Sciences* 44.2 (1992), pp. 193–219.

[BF99] Harry Buhrman and Lance Fortnow. "One-Sided Versus Two-Sided Error in Probabilistic Computation". In: *Proc. 16th Symposium on Theoretical Aspects of Computer Science (STACS)*. 1999, pp. 100–109.

[BKL+18] Harry Buhrman, Michal Koucký, Bruno Loff, and Florian Speelman. "Catalytic Space: Non-determinism and Hierarchy". In: *Theory Comput. Syst.* 62.1 (2018), pp. 116–135. DOI: 10.1007/S00224-017-9784-7.

[BKT14] Samuel R. Buss, Leszek Aleksander Kolodziejczyk, and Neil Thapen. "Fragments of Approximate Counting". In: *J. Symb. Log.* 79.2 (2014), pp. 496–525. DOI: 10.1017/JSL.2013.37.

[Bra10] Mark Braverman. "Polylogarithmic independence fools AC^0 circuits". In: *Journal of the ACM* 57.5 (2010).

[BTV09] Chris Bourke, Raghunath Tewari, and N. V. Vinodchandran. "Directed Planar Reachability Is in Unambiguous Log-Space". In: *ACM Trans. Comput. Theory* 1 (2009).

[Bus97] Samuel R Buss. "Bounded arithmetic and propositional proof complexity". In: *Logic of computation*. Springer, 1997, pp. 67–121.

[CH22] Kuan Cheng and William M. Hoza. "Hitting Sets Give Two-Sided Derandomization of Small Space". In: *Theory Comput.* 18 (2022), pp. 1–32.

[CHL+23] Yeyuan Chen, Yizhi Huang, Jiatu Li, and Hanlin Ren. "Range Avoidance, Remote Point, and Hard Partial Truth Table via Satisfying-Pairs Algorithms". In: *STOC*. ACM, 2023, pp. 1058–1066. DOI: 10.1145/3564246.3585147.

[CHR24] Lijie Chen, Shuichi Hirahara, and Hanlin Ren. "Symmetric Exponential Time Requires Near-Maximum Circuit Size". In: *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*. Ed. by Bojan Mohar, Igor Shinkar, and Ryan O'Donnell. ACM, 2024, pp. 1990–1999. DOI: 10.1145/3618260.3649624. URL: https://doi.org/10.1145/3618260.3649624.

[CJS+21] Lijie Chen, Ce Jin, Rahul Santhanam, and Ryan Williams. "Constructive Separations and Their Consequences". In: *Proc. 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2021, pp. 646–657.

[CL20] Eshan Chattopadhyay and Jyun-Jie Liao. "Optimal Error Pseudodistributions for Read-Once Branching Programs". In: *Proc. 35th Annual IEEE Conference on Computational Complexity (CCC)*. 2020, 25:1–25:27.

[CL24] Yilei Chen and Jiatu Li. "Hardness of Range Avoidance and Remote Point for Restricted Circuits via Cryptography". In: *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*. 2024, pp. 620–629.

[CLM+24] James Cook, Jiatu Li, Ian Mertz, and Edward Pyne. "The Structure of Catalytic Space: Capturing Randomness and ..." In: *Electron. Colloquium Comput. Complex.* TR24-106 (2024). ECCC: TR24-106. URL: https://eccc.weizmann.ac.il/report/2024/106.

[CLO+23] Lijie Chen, Zhenjian Lu, Igor Carboni Oliveira, Hanlin Ren, and Rahul Santhanam. "Polynomial-Time Pseudodeterministic Construction of Primes". In: *arXiv preprint arXiv:2305.15140* (2023).

[CM20] James Cook and Ian Mertz. "Catalytic approaches to the tree evaluation problem". In: *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020*. ACM, 2020, pp. 752–760.

[CM23] James Cook and Ian Mertz. "Tree Evaluation is in Space O(log n · log log n)". In: *Electron. Colloquium Comput. Complex.* TR23-174 (2023). ECCC: TR23-174. URL: https://eccc.weizmann.ac.il/report/2023/174.

[CN10] Stephen A. Cook and Phuong Nguyen. *Logical Foundations of Proof Complexity*. Cambridge University Press, 2010.

[CRS95] Suresh Chari, Pankaj Rohatgi, and Aravind Srinivasan. "Randomness-optimal unique element isolation with applications to perfect matching and related problems". In: *SIAM Journal on Computing* 24.5 (1995), pp. 1036–1050.

[CRT+20] Lijie Chen, Ron D. Rothblum, Roei Tell, and Eylon Yogev. "On Exponential-Time Hypotheses, Derandomization, and Circuit Lower Bounds". In: *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2020, pp. 13–23.

[CRT22] Lijie Chen, Ron D. Rothblum, and Roei Tell. "Unstructured Hardness to Average-Case Randomness". In: *Proc. 63rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2022, pp. 429–437.

[CT21a] Lijie Chen and Roei Tell. "Hardness vs Randomness, Revised: Uniform, Non-Black-Box, and Instance-Wise". In: *Proc. 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2021, pp. 125–136.

[CT21b] Lijie Chen and Roei Tell. "Simple and fast derandomization from very hard functions: Eliminating randomness at almost no cost". In: *Proc. 53st Annual ACM Symposium on Theory of Computing (STOC)*. 2021, pp. 283–291.

[CT23a] Lijie Chen and Roei Tell. "Guest column: New ways of studying the **BPL** = **P** conjecture". In: *ACM SIGACT News* 54.2 (2023), pp. 44–69.

[CT23b] Lijie Chen and Roei Tell. "When Arthur has Neither Random Coins nor Time to Spare: Superfast Derandomization of Proof Systems". In: *Proc. 55th Annual ACM Symposium on Theory of Computing (STOC)*. 2023, pp. 60–69.

[CTW23] Lijie Chen, Roei Tell, and Ryan Williams. "Derandomization vs Refutation: A Unified Framework for Characterizing Derandomization". In: *Proc. 64 Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. To appear. 2023.

[CW19] Lijie Chen and R. Ryan Williams. "Stronger Connections Between Circuit Analysis and Circuit Lower Bounds, via PCPs of Proximity". In: *Proc. 34th Annual IEEE Conference on Computational Complexity (CCC)*. 2019, 19:1–19:43.

[DGJ+12] Samir Datta, Chetan Gupta, Rahul Jain, Vimal Raj Sharma, and Raghunath Tewari. "Randomized and Symmetric Catalytic Computation". In: *Computer Science - Theory and Applications - 15th International Computer Science Symposium in Russia, CSR 2020*. 2012.

[DKM+13] Holger Dell, Valentine Kabanets, Dieter van Melkebeek, and Osamu Watanabe. "Is Valiant-Vazirani's isolation probability improvable?" In: *Computational Complexity* 22.2 (2013), pp. 345–383.

[DMO+22] Dean Doron, Dana Moshkovitz, Justin Oh, and David Zuckerman. "Nearly Optimal Pseudorandomness From Hardness". In: *Journal of the ACM* 69.6 (2022), pp. 1–55.

[DPT24] Dean Doron, Edward Pyne, and Roei Tell. "Opening Up the Distinguisher: A Hardness to Randomness Approach for BPL = L that Uses Properties of BPL". In: *Proc. 56th Annual ACM Symposium on Theory of Computing (STOC)*. 2024.

[DT23] Dean Doron and Roei Tell. "Derandomization with Minimal Memory Footprint". In: *Proc. 38 Annual IEEE Conference on Computational Complexity (CCC)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[FSU+13] Bill Fefferman, Ronen Shaltiel, Christopher Umans, and Emanuele Viola. "On beating the hybrid argument". In: *Theory of Computing* 9 (2013), pp. 809–843.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. "How to construct random functions". In: *J. ACM* 33.4 (1986), pp. 792–807.

[GGN+23] Karthik Gajulapalli, Alexander Golovnev, Satyajeet Nagargoje, and Sidhant Saraogi. "Range Avoidance for Constant-Depth Circuits: Hardness and Algorithms". In: *CoRR* abs/2303.05044 (2023). DOI: 10.48550/arXiv.2303.05044. arXiv: 2303.05044. URL: https://doi.org/10.48550/arXiv.2303.05044.

[GII+19] Alexander Golovnev, Rahul Ilango, Russell Impagliazzo, Valentine Kabanets, Antonina Kolokolova, and Avishay Tal. "AC^0[p] Lower Bounds Against MCSP via the Coin Problem". In: *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*. 2019.

[GJS+19] Chetan Gupta, Rahul Jain, Vimal Raj Sharma, and Raghunath Tewari. "Unambiguous Catalytic Computation". In: *39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2019*. Vol. 150. LIPIcs. 2019, 16:1–16:13.

[GL89] Oded Goldreich and Leonid A. Levin. "A Hard-core Predicate for All One-way Functions". In: *Proc. 21st Annual ACM Symposium on Theory of Computing (STOC)*. 1989, pp. 25–32.

[GLW22] Venkatesan Guruswami, Xin Lyu, and Xiuhan Wang. "Range Avoidance for Low-Depth Circuits and Connections to Pseudorandomness". In: *APPROX/RANDOM*. Vol. 245. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 20:1–20:21. DOI: 10.4230/LIPIcs.APPROX/RANDOM.2022.20.

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1: Basic Techniques*. Cambridge University Press, 2001. ISBN: 0-521-79172-3. DOI: 10.1017/CBO9780511546891.

[Gol08] Oded Goldreich. *Computational Complexity: A Conceptual Perspective*. New York, NY, USA: Cambridge University Press, 2008.

[Gol11a] Oded Goldreich. "A Sample of Samplers: A Computational Perspective on Sampling". In: 2011.

[Gol11b] Oded Goldreich. "In a World of $\mathbf{P} = \mathbf{BPP}$". In: *Studies in Complexity and Cryptography. Miscellanea on the Interplay Randomness and Computation*. 2011, pp. 191–232.

[Gol11c] Oded Goldreich. "Two Comments on Targeted Canonical Derandomizers". In: *Electronic Colloquium on Computational Complexity: ECCC* (2011).

[Gol18] Oded Goldreich. "On doubly-efficient interactive proof systems". In: *Foundations and Trends® in Theoretical Computer Science* 13.3 (2018).

[GRZ23] Uma Girish, Ran Raz, and Wei Zhan. "Is Untrusted Randomness Helpful?" In: *Proc. 14 Conference on Innovations in Theoretical Computer Science (ITCS)*. Vol. 251. LIPIcs. 2023, 56:1–56:18.

[GST19] Chetan Gupta, Vimal Raj Sharma, and Raghunath Tewari. "Reachability in O(log n) Genus Graphs is in Unambiguous Logspace". In: *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019*. 2019.

[GVW11] Oded Goldreich, Salil Vadhan, and Avi Wigderson. "Simplified derandomization of BPP using a hitting set generator". In: *Studies in complexity and cryptography*. Vol. 6650. Lecture Notes in Computer Science. Springer, Heidelberg, 2011, pp. 59–67.

[GW00] Oded Goldreich and Avi Wigderson. "On Pseudorandomness with respect to Deterministic Observers". In: *Electron. Colloquium Comput. Complex.* TR00-056 (2000). ECCC: TR00-056. URL: https://eccc.weizmann.ac.il/eccc-reports/2000/TR00-056/index.html.

[GW96] Anna Gál and Avi Wigderson. "Boolean complexity classes vs. their arithmetic analogs". In: *Random Struct. Algorithms* 9.1-2 (1996), pp. 99–111.

[GZ11] Oded Goldreich and David Zuckerman. "Another Proof That BPP subset PH (and More)". In: *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*. Ed. by Oded Goldreich. Vol. 6650. Lecture Notes in Computer Science. Springer, 2011, pp. 40–53.

[Hel86] Hans Heller. "On Relativized Exponential and Probabilistic Complexity Classes". In: *Inf. Control.* 71.3 (1986), pp. 231–243.

[HH23] Pooya Hatami and William M. Hoza. "Theory of Unconditional Pseudorandom Generators". In: *Electronic Colloquium on Computational Complexity: ECCC* (2023).

[HIL+99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. "A Pseudorandom Generator from any One-way Function". In: *SIAM Journal on Computing* 28.4 (1999), pp. 1364–1396.

[Hoe63] Wassily Hoeffding. "Probability inequalities for sums of bounded random variables". In: *Journal of the American Statistical Association* 58 (1963), pp. 13–30.

[Hoz19] William M. Hoza. "Typically-correct derandomization for small time and space". In: *Proc. 34th Annual IEEE Conference on Computational Complexity (CCC)*. 2019, 9:1–9:39.

[HU22] William M. Hoza and Chris Umans. "Targeted Pseudorandom Generators, Simulation Advice Generators, and Derandomizing Logspace". In: *SIAM J. Comput.* 51.2 (2022), pp. 17–281.

[ILW23] Rahul Ilango, Jiatu Li, and R. Ryan Williams. "Indistinguishability obfuscation, range avoidance, and bounded arithmetic". In: *Proc. 55th Annual ACM Symposium on Theory of Computing (STOC)*. [2023] ©2023, pp. 1076–1089.

[IW97] Russell Impagliazzo and Avi Wigderson. "$\mathsf{P} = \mathsf{BPP}$ if E requires exponential circuits: derandomizing the XOR lemma". In: *Proc. 29th Annual ACM Symposium on Theory of Computing (STOC)*. 1997, pp. 220–229.

[Jeř04] Emil Jeřábek. "Dual weak pigeonhole principle, Boolean complexity, and derandomization". In: *Ann. Pure Appl. Log.* 129.1-3 (2004), pp. 1–37.

[Jeř07] Emil Jeřábek. "Approximate counting in bounded arithmetic". In: *J. Symb. Log.* 72.3 (2007), pp. 959–993.

[KM02] Adam R. Klivans and Dieter van Melkebeek. "Graph Nonisomorphism Has Subexponential Size Proofs Unless the Polynomial-Time Hierarchy Collapses". In: *SIAM Journal on Computing* 31.5 (2002), pp. 1501–1526.

[Kor21] Oliver Korten. "The Hardest Explicit Construction". In: *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*. IEEE, 2021, pp. 433–444.

[Kor22] Oliver Korten. "Derandomization from time-space tradeoffs". In: *Proc. 37th Annual IEEE Conference on Computational Complexity (CCC)*. 2022.

[Kra07] Jan Krajıcek. "A proof complexity generator". In: *Proc. from the 13th International Congress of Logic, Methodology and Philosophy of Science (Beijing*. 2007.

[Kra95] Jan Krajíček. *Bounded Arithmetic, Propositional Logic, and Complexity Theory*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1995. ISBN: 978-0-521-45205-2.

[KT16] Vivek Anand T. Kallampally and Raghunath Tewari. "Trading Determinism for Time in Space Bounded Computations". In: *Proc. 41st International Symposium on Mathematical Foundations of Computer Science*. Ed. by Piotr Faliszewski, Anca Muscholl, and Rolf Niedermeier. 2016.

[KV10] Jan Kyncl and Tomás Vyskocil. "Logspace Reduction of Directed Reachability for Bounded Genus Graphs to the Planar Case". In: *ACM Trans. Comput. Theory* 1 (2010).

[Lau83] Clemens Lautemann. "BPP and the polynomial hierarchy". In: *Information Processing Letters* 17.4 (1983), pp. 215–217.

[Lev87] Leonid A. Levin. "One-way functions and pseudorandom generators". In: *Combinatorica* 7.4 (1987), pp. 357–363.

[Li24] Zeyong Li. "Symmetric Exponential Time Requires Near-Maximum Circuit Size: Simplified, Truly Uniform". In: *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*. Ed. by Bojan Mohar, Igor Shinkar, and Ryan O'Donnell. ACM, 2024, pp. 2000–2007. DOI: 10.1145/3618260.3649615. URL: https://doi.org/10.1145/3618260.3649615.

[LP20] Yanyi Liu and Rafael Pass. "On one-way functions and Kolmogorov complexity". In: *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. [2020] ©2020, pp. 1243–1254.

[Mer23] Ian Mertz. "Reusing Space: Techniques and Open Problems". In: *Bulletin of EATCS* 141.3 (2023).

[MP19] Dieter van Melkebeek and Gautam Prakriya. "Derandomizing Isolation in Space-Bounded Settings". In: *SIAM J. Comput.* 48.3 (2019), pp. 979–1021.

[MP20] Moritz Müller and Ján Pich. "Feasibly constructive proofs of succinct weak circuit lower bounds". In: *Ann. Pure Appl. Log.* 171.2 (2020).

[MP23] Noam Mazor and Rafael Pass. "Counting Unpredictable Bits: A Simple PRG from One-Way Functions". In: *Theory of Cryptography - 21st International Conference, TCC 2023, Taipei, Taiwan, November 29 - December 2, 2023, Proceedings, Part I*. Ed. by Guy N. Rothblum and Hoeteck Wee. Vol. 14369. Lecture Notes in Computer Science. Springer, 2023, pp. 191–218. DOI: 10.1007/978-3-031-48615-9_7.

[MS23a] Dieter van Melkebeek and Nicollas Sdroievski. "Instance-Wise Hardness versus Randomness Tradeoffs for Arthur-Merlin Protocols". In: *Proc. 38 Annual IEEE Conference on Computational Complexity (CCC)*. 2023.

[MS23b] Dieter van Melkebeek and Nicollas M. Sdroievski. "Leakage Resilience, Targeted Pseudorandom Generators, and Mild Derandomization of Arthur-Merlin Protocols". In: *43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*. Ed. by Patricia Bouyer and Srikanth Srinivasan. Vol. 284. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023, 29:1–29:22.

[MVV87] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. "Matching is as easy as matrix inversion". In: *Comb.* 7.1 (1987), pp. 105–113.

[Nis91] Noam Nisan. "Pseudorandom bits for constant depth circuits". In: *Combinatorica* 11.1 (1991), pp. 63–70.

[Nis93] Noam Nisan. "On Read-Once vs. Multiple Access to Randomness in Logspace". In: *Theoretical Computer Science* 107.1 (1993), pp. 135–144.

[Nis94] Noam Nisan. "$\mathsf{RL} \subseteq \mathsf{SC}$". In: *Computational Complexity* 4 (1994), pp. 1–11.

[NW94] Noam Nisan and Avi Wigderson. "Hardness vs. randomness". In: *Journal of Computer and System Sciences* 49.2 (1994), pp. 149–167.

[PR23] Rafael Pass and Oren Renard. "Characterizing the Power of (Persistent) Randomness in Log-space". In: *Electronic Colloquium on Computational Complexity: ECCC* (2023).

[PRZ23] Edward Pyne, Ran Raz, and Wei Zhan. "Certified Hardness vs. Randomness for Log-Space". In: *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023*. 2023.

[PS21] Ján Pich and Rahul Santhanam. "Strong co-nondeterministic lower bounds for NP cannot be proved feasibly". In: *STOC '21*. Ed. by Samir Khuller and Virginia Vassilevska Williams. 2021.

[Pyn24] Edward Pyne. "Derandomizing Logspace with a Small Shared Hard Drive". In: *39th Computational Complexity Conference, CCC 2024, July 22-25, 2024, Ann Arbor, MI, USA*. Ed. by Rahul Santhanam. Vol. 300. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, 4:1–4:20.

[RA00] Klaus Reinhardt and Eric Allender. "Making Nondeterminism Unambiguous". In: *SIAM J. Comput.* 29.4 (2000), pp. 1118–1131.

[RR97] Alexander A. Razborov and Steven Rudich. "Natural proofs". In: *Journal of Computer and System Sciences* 55.1, part 1 (1997), pp. 24–35.

[RS98] Alexander Russell and Ravi Sundaram. "Symmetric Alternation Captures BPP". In: *Comput. Complex.* 7.2 (1998), pp. 152–162.

[RSW22] Hanlin Ren, Rahul Santhanam, and Zhikun Wang. "On the Range Avoidance Problem for Circuits". In: *Proc. 63rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2022.

[Sha49] Claude E. Shannon. "The synthesis of two-terminal switching circuits". In: *Bell System technical journal* 28.1 (1949), pp. 59–98.

[Sip83] Michael Sipser. "A complexity theoretic approach to randomness". In: *Proc. 15th Annual ACM Symposium on Theory of Computing (STOC)*. 1983, pp. 330–335.

[Sip88] Michael Sipser. "Expanders, randomness, or time versus space". In: *Journal of Computer and System Sciences* 36.3 (1988), pp. 379–383.

[Siv02] D. Sivakumar. "Algorithmic derandomization via complexity theory". In: *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002*. Ed. by John H. Reif. ACM, 2002, pp. 619–626.

[STV01] Madhu Sudan, Luca Trevisan, and Salil Vadhan. "Pseudorandom generators without the XOR lemma". In: *Journal of Computer and System Sciences* 62.2 (2001), pp. 236–266.

[SU05] Ronen Shaltiel and Christopher Umans. "Simple extractors for all min-entropies and a new pseudorandom generator". In: *Journal of the ACM* 52.2 (2005), pp. 172–216.

[Sud97] Madhu Sudan. "Decoding of Reed Solomon Codes beyond the Error-Correction Bound". In: *J. Complex.* 13.1 (1997), pp. 180–193. DOI: 10.1006/jcom.1997.0439. URL: https://doi.org/10.1006/jcom.1997.0439.

[SV22] Ronen Shaltiel and Emanuele Viola. "On Hardness Assumptions Needed for "Extreme High-End" PRGs and Fast Derandomization". In: *Proc. 13 Conference on Innovations in Theoretical Computer Science (ITCS)*. 2022.

[SW13] Rahul Santhanam and R. Ryan Williams. "On medium-uniformity and circuit lower bounds". In: *Proc. 28th Annual IEEE Conference on Computational Complexity (CCC)*. IEEE, 2013, pp. 15–23.

[Tha02] Neil Thapen. "The weak pigeonhole principle in models of bounded arithmetic". PhD thesis. University of Oxford, 2002.

[Tod91] Seinosuke Toda. "PP is as hard as the polynomial-time hierarchy". In: *SIAM Journal on Computing* 20 (1991).

[Tre01] Luca Trevisan. "Extractors and Pseudorandom Generators". In: *Journal of the ACM* 48.4 (2001), pp. 860–879.

[TSUZ07] Amnon Ta-Shma, Christopher Umans, and David Zuckerman. "Lossless condensers, unbalanced expanders, and extractors". In: *Combinatorica* 27.2 (2007), pp. 213–240.

[TSZS06] Amnon Ta-Shma, David Zuckerman, and Shmuel Safra. "Extractors from Reed-Muller codes". In: *Journal of Computer and System Sciences* 72.5 (2006), pp. 786–812.

[Uma03] Christopher Umans. "Pseudo-random generators for all hardnesses". In: *Journal of Computer and System Sciences* 67.2 (2003), pp. 419–440.

[Vad12] Salil P. Vadhan. *Pseudorandomness*. Foundations and Trends in Theoretical Computer Science. Now Publishers, 2012.

[Val76] Leslie G. Valiant. "Relative complexity of checking and evaluating". In: *Information Processing Letters* 5.1 (1976/77), pp. 20–23.

[VV86] Leslie G. Valiant and Vijay V. Vazirani. "NP is as Easy as Detecting Unique Solutions". In: *Theor. Comput. Sci.* 47.3 (1986), pp. 85–93.

[Wil13] R. Ryan Williams. "Improving Exhaustive Search Implies Superpolynomial Lower Bounds". In: *SIAM Journal on Computing* 42.3 (2013), pp. 1218–1244.

[Wil16] Richard Ryan Williams. "Strong ETH breaks with Merlin and Arthur: short non-interactive proofs of batch evaluation". In: *Proc. 31st Annual IEEE Conference on Computational Complexity (CCC)*. Vol. 50. 2016, 2:1–2:17.

[Yao82] Andrew C. Yao. "Theory and Application of Trapdoor Functions". In: *Proc. 23rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 1982, pp. 80–91.

A D2P Centric Proofs of Classical Results

We present new proofs for two classical results in complexity theory. Our proofs are interesting for two reasons: Conceptually, the main notion in the proofs is a D2P transformation; and technically, the new proofs are simple and appealing.

Specifically, we present proofs of the following:

1. Derandomization of **prBPP** reduces to derandomization of **prRP**. We provide alternative proofs for both known "flavors" of this result.
2. $\mathbf{MA} \subseteq \mathbf{S_2P}$ (and hence also $\mathbf{BPP} \subseteq \mathbf{S_2P}$).

A.1 Proof overviews

One-sided to two-sided derandomization. Recall that $\mathbf{prBPP} \subseteq \mathbf{prRP}^{\mathbf{prRP}}$, as proved by [Sip83; Lau83; BF99; GZ11]. More generally, there are two known types of reductions of derandomization of **BPP** to derandomization of **RP**: The first only relies on the assumption that **RP** can be derandomized, but incurs a large time overhead (in the derandomization of **BPP**); and the second relies on the stronger assumption that **RP** can be derandomized in a black-box way (i.e., using hitting-set generators), and only incurs a polynomial time overhead. In more detail:

Theorem A.1 (derandomization of **BPP** using hitting-sets; see [ACR98; ACR+99; GVW11], and also [NW94; IW97; Uma03])**.** *There is a constant $c \geq 1$ such that the following holds for every time bound $T(n)$ satisfying that $T(n)^k \leq T(n^{O(k)})$. If there exists a hitting set for circuits of size n computable in time $T(n)$, then* $\mathbf{prBPTIME}[n] \subseteq \mathbf{prDTIME}[T(n^c)]$.[42]

Theorem A.2 (derandomization of **BPP** from derandomization of **RP**; see [Sip83; Lau83; BF99; GZ11; CH22])**.** *There is a constant $c \geq 1$ such that for every time bound $T(n)$ satisfying that $T(n)^k \leq T(n^{O(k)})$, if* $\mathbf{prRTIME}[n] \subseteq \mathbf{prDTIME}[T(n)]$ *then* $\mathbf{prBPTIME}[n] \subseteq \mathbf{prDTIME}[T(T(n^c))]$.

The main idea behind our alternative proofs for both Theorem A.1 and Theorem A.2 is very simple, as explained next.

(*Proof idea of Theorem A.1*). Assume that there is a hitting-set generator as in the hypothesis. Then, given any circuit C, the HSG contains "suffixes" that derandomize Yao's transformation of C to predictors (as in Lemma C.4). We can then construct a distribution that isn't predicted by these predictors, using efficient diagonalization (i.e., as in Lemma 4.2, following [GW00]). This distribution is pseudorandom for C by the security of the D2P transformation.

(*Proof idea of Theorem A.2*). Consider a probabilistic algorithm P_1 that gets as input a circuit C and a distribution $\mathbf{D}$, and tests whether or not Yao's transformation (applied to C) yields, whp, a predictor for $\mathbf{D}$. (In particular, when C is a distinguisher for $\mathbf{D}$, the algorithm P_1 accepts whp.)

Now, consider inputs $(C, \mathbf{D})$ with the following promise: Either C is a distinguisher for $\mathbf{D}$, or $\mathbf{D}$ is unpredictable by any efficient predictor. Note that in the latter case, P_1 rejects with probability 1;[43]hence, on inputs satisfying this promise, P_1 has one-sided error. Hence, by our assumption, there is a deterministic algorithm $\tilde{P}_1$ that gets as input $(C, \mathbf{D})$ and solves this promise problem.

Given as input a circuit C and a bit $\sigma \in \{0, 1\}$, which we think of as indicating whether the acceptance probability of C is high, we design a probabilistic algorithm A that decides whether σ is indeed correct, whp, and that has one-sided error (i.e., when σ is incorrect A always rejects). Since A has one-sided error, the foregoing problem can also be solved deterministically.

[42]The reason for crediting [NW94; IW97; Uma03] for this result is that the "hardness vs randomness" framework can be used to prove the result. Specifically, if there are polynomial-time computable hitting-set generators for linear-sized circuits, then there is a function in $\mathbf{E} = \mathbf{DTIME}[2^{O(n)}]$ that is hard for circuits of size $2^{\varepsilon \cdot n}$ (for some $n \in \mathbb{N}$), and hence there exist polynomial-time computable pseudorandom generators for linear-sized circuits (and thus $\mathbf{prBPP} = \mathbf{prP}$). Using the results of [Uma03], this statement also scales smoothly quantitatively (i.e., slower hitting-set generators yield slower pseudorandom generators).

Specifically, given (C, σ), the algorithm A draws a random $\mathbf{D}$ and outputs

$$\mathbb{I}\Big[\,|\,\mathbb{E}[C(\mathbf{D})] - \sigma| < 1/10\Big] \wedge \neg \tilde{P}_1(C, \mathbf{D}).$$

For any input (C, σ), almost all distributions $\mathbf{D}$ will be both pseudorandom for C and unpredictable by any efficient predictor, and thus A outputs the correct answer whp. On the other hand, when σ is incorrect, A always rejects: This is since if $|\,\mathbb{E}[C(\mathbf{D})] - \sigma| < 1/10$, then Yao's transformation succeeds with high probability (since C is a distinguisher for $\mathbf{D}$), in which case $\tilde{P}_1(C, \mathbf{D}) = 1$.

Note that the alternative proof of Theorem A.2 above incurs the same overhead as in the previous proofs of this result. Specifically, the algorithm A runs $\tilde{P}_1$, whose running time is T (by the derandomization assumption), and hence A runs in time $\mathrm{poly}(T)$; after invoking the derandomization hypothesis on A, we get a deterministic algorithm that runs in time $\mathrm{poly}(T(\mathrm{poly}(T)))$.

Upper bounds for MA. Finally, we give two simple proofs of the best-known upper bound for **MA** (which also yields an upper bound for **BPP**), originally due to Russell and Sundaram [RS98]:

Proposition A.3. *We have that* (**BPP** $\subseteq$)**MA** $\subseteq$ $\mathbf{S_2P}$.

For clarity, we recall the standard definitions of $\mathbf{S_2P}$ and of **MA**.

Definition A.4. A language $L \in \mathbf{S_2P}$ if there is a polynomial-time algorithm $V(x, g_0, g_1)$ (called the verifier) and a polynomial $p(n)$ such that:

- For $x \in L$, there exists $g_1 \in \{0,1\}^{p(n)}$ (i.e. the strategy of the YES-player) such that for every $g_0 \in \{0,1\}^{p(n)}$ (i.e. the strategy of the NO-player), $V(x, g_0, g_1) = 1$.
- For $x \notin L$, there exists $g_0 \in \{0,1\}^{p(n)}$ (i.e. the strategy of the NO-player) such that for every $g_1 \in \{0,1\}^{p(n)}$ (i.e. the strategy of the YES-player), $V(x, g_0, g_1) = 0$.

Definition A.5. A language $L \in \mathbf{MA}$ if there is a polynomial-time algorithm $V(x, g, r)$ (called the verifier) and a polynomial $p(n)$ such that:

- **Completeness.** For $x \in L$, there exists $g \in \{0,1\}^{p(n)}$ (which we call a **witness**) such that $\mathbb{E}[V(x, g, \mathbf{U}_{p(n)})] > 9/10$.
- **Soundness.** For $x \notin L$, for every $g \in \{0,1\}^{p(n)}$, we have $\mathbb{E}[V(x, g, \mathbf{U}_{p(n)})] < 1/10$.

(*Idea of the the **first** proof*). Given a statement x, the "yes" prover P^{YES} is supposed to provide a proof π and a D2P transformation (i.e., predictors) $\mathcal{P}$ for the corresponding distinguisher $V_{x,\pi}(\cdot) = V(x, \pi, \cdot)$, and the "no" prover P^{NO} is supposed to provide a distribution $\mathbf{D}$ unpredictable by all efficient procedures. The verifier checks that the predictors $\mathcal{P}$ do not predict $\mathbf{D}$, and if this is true, it uses $\mathbf{D}$ as pseudorandom coins for $V_{x,\pi}$.

When $x \in L$, P^{YES} can provide π and a correct D2P $\mathcal{P}$, in which case any $\mathbf{D}$ that is unpredictable by $\mathcal{P}$ is pseudorandom for $V_{x,\pi}$ (and the verifier can recognize $\mathbf{D}$'s that are predictable by $\mathcal{P}$, and accept). When $x \notin L$, P^{NO} can provide an unpredictable distribution, which is necessarily pseudorandom for $V_{x,\pi}$ (and thus the verifier will not accept any proof of P^{YES}).

[43] This is because P_1 first runs Yao's transformation (using randomness), and then tests *deterministically* whether the resulting procedure is a predictor for $\mathbf{D}$.

(*Idea of the **second** proof*). Our second proof does not use the notion of D2P, but it was motivated by the idea of certifying one derandomization with another derandomization. For the latter, the verifier expects both P^{YES} and P^{NO} to provide distributions $\mathbf{D}_Y, \mathbf{D}_N$ that are pseudorandom for all small circuits. The key observation is that at least one player will always want the verifier to accurately estimate the acceptance probability of the verifier circuit $M(x, \cdot)$. If the two distributions disagree on that expectation, we can *duel* the distributions against each other using downward-self reducibility of CAPP on the circuit $M(x, \cdot)$, and eventually determine which distribution was not in fact pseudorandom.

A.2 Formal Proofs

We now formalize each argument. We will need the following probabilistic construction:

Fact A.6. There is a constant $c > 0$ such that for every $n, S \in \mathbb{N}$ and $\varepsilon > 0$, a random multiset $\mathbf{D} \subseteq \{0,1\}^n$ of size $(nS/\varepsilon)^c$ satisfies the following with probability $1 - \exp(-\Omega(S))$: No circuit of size S on at most n bits predicts $\mathbf{D}$ with advantage ε, nor distinguishes $\mathbf{D}$ from the uniform distribution $\mathbf{U}_n$ with advantage ε.

Proof. Let c be sufficiently large and $\ell \triangleq (nS/\varepsilon)^c$. Fix any $n, S \in \mathbb{N}$ and $\varepsilon > 0$. Let $\mathbf{D} \subseteq \{0,1\}^n$ be a random multiset of size ℓ, and $x^{(i)} \in \{0,1\}^n$ be the i-th string in $\mathbf{D}$.

- For every $i \in [n]$ and every circuit $C : \{0,1\}^{i-1} \to \{0,1\}$ of size S, we know by the Chernoff bound (see Theorem 3.9) that the probability that C predicts the i-th bit of $\mathbf{D}$ with advantage ε is at most $2\exp(-\varepsilon^2\ell/40)$. By the union bound, we know that with probability at most
$$2\exp(-\varepsilon^2\ell/40) \cdot 2^{O(S \log S)} \cdot n \leq \exp(-\Omega(S)),$$
for every $i \in [n]$ and every circuit C of size S, the i-th bit of $\mathbf{D}$ cannot be predicted by C with advantage ε.
- Similarly, we can prove using the Chernoff bound and the union bound that with probability $\exp(-\Omega(S))$, $\mathbf{D}$ cannot be distinguished from $\mathbf{U}_n$ with advantage ε by any circuit of size S.

Thus, by the union bound, we know that with probability $1 - 2\exp(-\Omega(S)) = 1 - \exp(-\Omega(S))$, the distribution $\mathbf{D}$ is neither predictable nor distinguishable from $\mathbf{U}_n$ by circuits of size S. □

A.2.1 Black-Box One-Sided to Two-Sided: Proof of Theorem A.1

By standard reductions, we may assume we are given a circuit $C : \{0,1\}^n \to \{0,1\}$ of size n, and wish to estimate $\mathbb{E}[C]$ to error $(1/3)$. Let $C : \{0,1\}^n \to \{0,1\}$ be the size n circuit to solve CAPP for. By standard error-reduction results, we have a $T(n^c)$-time computable $(1/10n)$-hitting set for circuits of size n^4 that read n bits, for some constant $c > 0$. Let this set be $H \subseteq \{0,1\}^n$. Applying Lemma C.4, we have that H is a $(m = 100n^3, \alpha = 1/10n)$-BB Yao family for general circuits. Therefore, observe that

$$\left\{P_{z,i,\sigma}(x_{<i}) = C(x_{<i} \circ \sigma_1 \circ z_{>i}) \oplus \sigma_2 : z \in H, i \in [n], \sigma \in \{0,1\}^2\right\} \leftarrow C$$

is a $(m = 100n^3, \alpha = 1/10n)$-D2P transformation for circuits of size n, and moreover given C we can output the predictors $\mathcal{P} = (P_1, \ldots, P_t)$ in time $\mathrm{poly}(T(n^c))$. Finally, applying Lemma 4.2 and using that $m \geq n/\alpha^2$, we have that there is a deterministic polytime algorithm that, given $\mathcal{P}$, outputs a distribution $\mathbf{D}$ of size $\mathrm{poly}(T(n^c))$ such that $|\mathbb{E}[C(\mathbf{D})] - \mathbb{E}[C(\mathbf{U}_n)]| \leq 1/3$ (by the security of the D2P transformation). Thus, we compute $\mathbb{E}[C(\mathbf{D})]$ and return this estimate. The runtime follows as $\mathrm{poly}(T(n^c)) \leq T(n^{c'})$ for some constant c'.

A.2.2 Non-Black-Box One-Sided to Two-Sided: Proof of Theorem A.2

We first define the one-sided promise derandomization problem we consider:

Definition A.7. Let $\mathcal{A}$ be a one-sided derandomization algorithm that works as follows. On input a circuit $T : \{0,1\}^n \to \{0,1\}$ of size n:

- If $\mathbb{E}[T(\mathbf{U}_n)] = 0$, then $\mathcal{A}(T) = 0$.
- If $\mathbb{E}[T(\mathbf{U}_n)] \geq 1/10n$, then $\mathcal{A}(T) = 1$.
- Otherwise, $\mathcal{A}(T)$ has arbitrary behavior.

Next, we create a test machine that takes in a circuit C of size n and a distribution $\mathbf{D}$ of size n^c and attempts to find a predictor for $\mathbf{D}$ by applying the one-sided derandomization algorithm to Yao's lemma. Then we ask our one-sided derandomization algorithm to find $\mathbf{D}$ that *cannot* be broken by this machine. Since almost all distributions will have no small predictors, most $\mathbf{D}$ satisfy this property, but any $\mathbf{D}$ that induces a bad estimate of the expectation will fail the test.

Algorithm 1: TryBreakD$(C, \mathbf{D}, e)$

```
if |𝔼[C(D)] − e| > 1/5 then
    return 0
end
for i ∈ [n], σ ∈ {0,1}^2 do
    Let T_{C,D,σ,i}(z) be the circuit that accepts if adv_D(P_{z,σ,i}) ≥ 1/10n.
    if 𝒜(T_{C,D,σ,i}) = 1 then
        return 0
    end
end
return 1
```

Let $\mathcal{M}$ be the machine implementing Algorithm 1. We first prove some useful claims.

Claim A.8. *For every C with $\mathbb{E}[C] < 1/10$ and $\mathbf{D}$ supported over $\{0,1\}^n$ that has no advantage $(1/10n)$ predictors of size $\text{size}(C) + 2n$, we have $\mathcal{M}(C, \mathbf{D}, 0) = 1$.*

Proof. First, by Yao's lemma (see Lemma 3.8) we have that $\mathbf{D}$ has no $(1/10)$-distinguishers of size $\text{size}(C)$. Thus, we must have $|\mathbb{E}[C(\mathbf{D})] - \mathbb{E}[C(\mathbf{U}_n)]| \leq 1/10$, and hence

$$|\mathbb{E}[C(\mathbf{D})] - e| \leq |\mathbb{E}[C(\mathbf{D})] - \mathbb{E}[C(\mathbf{U}_n)]| + |\mathbb{E}[C(\mathbf{U}_n)] - e| \leq \frac{1}{10} + \frac{1}{10},$$

so we do not reject on the first line. Furthermore, every circuit $T_{C,\mathbf{D},\sigma,i}$ produced in the loop has expectation exactly 0 (as any z and $T_{C,\mathbf{D},\sigma,i}$ such that $T_{C,\mathbf{D},\sigma,i}(z) = 1$ would imply a $(1/10n)$-predictor for $\mathbf{D}$ of size $\text{size}(C) + 2n$), and hence $\mathcal{A}$ will return 0 and we will not reject. □

Next, depending on e (which we think of as a guess of the expectation of C) the algorithm either accepts almost all distributions, or no distributions:

Claim A.9. *The following holds for an absolute constant c. For every $C : \{0,1\}^n \to \{0,1\}$, if $\mathbb{E}[C(\mathbf{U}_n)] < 1/10$ then, letting $\mathbf{D}$ be a random distribution over $\{0,1\}^n$ of size n^c uniformly sampled as $\mathbf{D} \leftarrow \mathbf{U}((\{0,1\}^n)^{n^c})$, we have:*

- $\mathbb{E}[\mathcal{M}(C, \mathbf{D}, 0)] \geq 1 - 1/(10n^{c+1})$
- $\mathbb{E}[\mathcal{M}(C, \mathbf{D}, 1)] = 0$

And vice-versa if $\mathbb{E}[C(\mathbf{U}_n)] > 9/10$.

Proof. We only demonstrate the case that $\mathbb{E}[C(\mathbf{U}_n)] < 1/10$, while it is easy to verify that an analogous argument holds if $\mathbb{E}[C] > 9/10$.

- For the $e = 0$ case, we have the following. Choosing c sufficiently large as in Fact A.6, we have that almost all distributions $\mathbf{D}$ of size n^c satisfy the condition of Claim A.8, so we obtain the desired result.
- For the $e = 1$ case, we assume that $\mathbb{E}[C(\mathbf{D})] \geq 8/10$ since otherwise $\mathcal{M}(C, \mathbf{D}, e)$ returns 0 in its first line. Then we have

  $$|\mathbb{E}[C(\mathbf{U}_n)] - \mathbb{E}[C(\mathbf{D})]| \geq 8/10 - 1/10,$$

  so by Lemma 3.8 there is $i \in [n]$ and $\sigma \in \{0,1\}^2$ such that with probability $1/10n$ over z, the Yao predictor $P_{z,\sigma,i}$ satisfies

  $$\mathrm{adv}_{\mathbf{D}}(P_{z,\sigma,i}) \geq \frac{1}{10n}.$$

  But then the test circuit $T_{C,\mathbf{D},\sigma,i}$ has expectation at least $1/10n$, so the inner algorithm $\mathcal{A}$ will evaluate to 1 on this circuit, and the overall algorithm will return 0. □

We can then prove Theorem A.2. By the assumption and standard error reduction results, there is a machine $\mathcal{A}$ running in time $T(n^{O(1)})$ with the behavior of Definition A.7, where $O(1)$ hides an absolute constant. We instantiate the algorithm $\mathcal{M}$ of Line 1 with this machine, and observe that $\mathcal{M}$ can be represented as a circuit of size $\mathrm{poly}(T(n^{O(1)})) = T(n^{O(1)})$.

Let $C : \{0,1\}^n \to \{0,1\}$ be a circuit of size n that we need to estimate the expectation of to error $(1/3)$. By standard error reduction results, we may assume without loss of generality that either $\mathbb{E}[C(\mathbf{U}_n)] < 1/10$ or $\mathbb{E}[C(\mathbf{U}_n)] > 9/10$. Let $T_2(D) \triangleq \mathcal{M}(C, \mathbf{D} = \cdot, e = 0)$ be the circuit obtained from $\mathcal{M}$ by fixing C and e as inputs. Our derandomization algorithm runs the machine $\mathcal{A}$ on this circuit, and return the negation of $\mathcal{A}(T_2)$. The correctness of our algorithm follows from Claim A.9:

- if $\mathbb{E}[C(\mathbf{U}_n)] < 1/10$, T_2 will accept all but a $1/|\mathbf{D}|$ fraction of its inputs, and thus $\mathcal{A}(T_2)$ will accept and our algorithm will reject;
- if $\mathbb{E}[C(\mathbf{U}_n)] > 9/10$, T_2 will reject all its inputs, and thus $\mathcal{A}(T_2)$ will reject and our algorithm will accept.

Moreover, by the runtime bound on $\mathcal{A}$ and the bound on the size of T_2, our overall runtime becomes $\mathrm{poly}(T(\mathrm{poly}(T(n^{O(1)})))) \leq T(T(n^c))$ for an absolute constant c.

A.2.3 Upper Bounds for MA

Recall that we give two proofs, where the first relies on D2P, and the second is more direct (though inspired by the same notions).

First proof of Proposition A.3 Fix an arbitrary **MA** language L. For its **MA** verifier V, we define $T_{x,g}(r) \triangleq V(x,g,r)$ be the circuit, and without loss of generality (by padding) we assume that $T : \{0,1\}^n \to \{0,1\}$ is of size n. We now give the $\mathbf{S_2P}$ protocol.

- The YES-player submits a witness g and an $(\infty, 1/10n)$-D2P transformation $\mathcal{P}$ for the circuit $T_{x,g}(r) = V(x,g,r)$, where the size of each predictor is at most $s = n^c$ for a constant c. Note that such a transformation always exists by Theorem B.1.
- The NO-player submits a distribution $\mathbf{D}_N$ over $\{0,1\}^n$ of size $\mathrm{poly}(n)$ such that $\mathbf{D}_N$ has no $(1/10n)$-predictors of size s, which always exists by Fact A.6.

Then the verifier works as follows. It first checks if there is a $P \in \mathcal{P}$ such that $\mathrm{adv}_{\mathbf{D}_N}(P) > 1/10n$. In this case, we know that the NO-player did not submit a truly unpredictable distribution, and the verifier immediately accepts. Otherwise, the verifier uses $\mathbb{E}_{r \leftarrow \mathbf{D}_N}[T_{x,g}(r)]$ to estimate the acceptance probability of $T_{x,g}(r)$, and accepts if and only if the estimated acceptance probability of $T_{x,g}(r)$ is greater than $2/3$.

We now argue correctness of the protocol.

- For $x \in L$, if the YES-player submits a witness g and a D2P transformation $\mathcal{P}$ for $T_{x,g}$, for any strategy $\mathbf{D}_N$ of the NO-player, either $\mathbf{D}_N$ is predictable by $\mathcal{P}$ and the verifier accepts, or the circuit C fails to $(1/10)$-distinguish $\mathbf{D}_N$ and the uniform distribution by Yao's lemma (see Lemma 3.8) and then

$$\mathop{\mathbb{E}}_{r \leftarrow \mathbf{D}_N}[T_{x,g}(r)] \geq \mathop{\mathbb{E}}_{r \leftarrow \mathbf{U}_N}[T_{x,g}(r)] - \frac{1}{10} \geq \frac{2}{3}.$$

  (The second inequality follows from the completeness of the **MA** protocol.)

- For $x \in L$, if the NO-player submits a distribution $\mathbf{D}_N$ with no small predictors, the verifier will use $\mathbb{E}[T_{x,g}(\mathbf{D}_N)]$ to estimate the acceptance probability. Note that C fails to $(1/10)$-distinguish $\mathbf{D}_N$ and the uniform distribution by Yao's lemma (see Lemma 3.8), and then

$$\mathop{\mathbb{E}}_{r \leftarrow \mathbf{D}_N}[T_{x,g}(r)] \leq \mathop{\mathbb{E}}_{r \leftarrow \mathbf{U}_N}[T_{x,g}(r)] + \frac{1}{10} \geq \frac{2}{3}.$$

  (The second inequality follows from the soundness of the **MA** protocol.)

Second proof of Proposition A.3. We rely on the following lemma, which takes two competing algorithms, one of which is guaranteed to solve PCAPP, and "duels" the algorithms to find the correct one.

Lemma A.10 (dueling PCAPP algorithms)**.** *There is a deterministic polynomial-time algorithm* DUEL *that takes in a parameter $\varepsilon > 0$, a circuit $C : \{0,1\}^n \to \{0,1\}$ of size n, and two circuits E_0, E_1 such that at least one of them is an ε-PCAPP algorithm for circuits of size n, and returns ρ such that $|\rho - \mathbb{E}[C]| \leq (4n+2)\varepsilon$.*

Proof. Recall that for $x \in \{0,1\}^{\leq n}$ and $b \in \{0,1\}$, $E_b(C,x)$ represents the estimate of E_b on C with the first bits set to x. Let

$$g_x \triangleq |E_0(C,x) - E_1(C,x)|$$

be the gap between the estimates under the two machines. In the event that the machines disagree on the expectation, we recursively identify a prefix under which they *continue* to disagree, eventually

enabling us to falsify the bad machine. Our algorithm works as follows. First, if $g_\emptyset \le (4n+1)\varepsilon$, we return $\rho = E_0(C, \emptyset)$ and the triangle inequality gives the desired error bound. Otherwise, we initialize $x = \emptyset$ and proceed in n stages. At the current stage with $x \in \{0,1\}^i$, we first check that for both b,

$$\left| E_b(C,x) - \frac{E_b(C, x\circ 0) + E_b(C, x\circ 1)}{2} \right| \le 2\varepsilon.$$

If this does not occur for E_b, it is easy to see that E_b cannot actually be an ε-PCAPP algorithm, so we can reject it and return $E_{\neg b}(C, \emptyset)$. Otherwise, if this holds for both E_0 and E_1, we have:

$$\begin{aligned} g_x &= |E_0(C,x) - E_1(C,x)| \\ &\le \left| \frac{E_0(C, x\circ 0) + E_0(C, x\circ 1)}{2} - \frac{E_1(C, x\circ 0) + E_1(C, x\circ 1)}{2} \right| + 4\varepsilon \\ &\le \frac{g_{x\circ 0} + g_{x\circ 1}}{2} + 4\varepsilon \end{aligned}$$

so there is some $p \in \{0,1\}$ such that $g_{x\circ p} \ge g_x - 4\varepsilon$ (and we can identify this value by checking both). We set $x \leftarrow x \circ p$ and continue. Thus, after n steps we have either identified the correct algorithm, or there is $x \in \{0,1\}^n$ such that $g_x > \varepsilon$, at which point we can compute $C(x)$ and determine which machine is wrong. Thus, we must at some point reject one of the algorithms, and hence we can return the estimation of $E_c(C, \emptyset)$ the correct algorithm E_c. □

We can then prove the result. Fix any language $L \in \mathbf{MA}$. For its **MA** verifier V, we define $T_{x,g}(r) \triangleq V(x,g,r)$ be the circuit, and without loss of generality (by padding) we assume that $T : \{0,1\}^n \to \{0,1\}$ is of size n. We now give the $\mathbf{S_2P}$ protocol for L.

- The YES-player is supposed to send a witness g and a distribution $\mathbf{D}_1$ of size n^c that cannot be $(1/50n)$-distinguished by circuits of size n (which we call a **good distribution**). Such a distribution always exists per Fact A.6 for a sufficiently large constant c.
- The NO-player likewise is supposed to send a good distribution $\mathbf{D}_0$.

Then the verifier $\mathcal{V}$ constructs the circuit $T_{x,g}(r)$ and applies Lemma A.10, where the machine E_b simply takes the empirical average over $\mathbf{D}_b$, i.e., $E_b(C, x_{\le i}) \triangleq \mathbb{E}_{z \leftarrow \mathbf{D}_b}[C(x_{\le i} \circ z_{>i})]$. Then:

$$\mathcal{V}(x, (g, \mathbf{D}_1), \mathbf{D}_0) = \mathbb{I}\left[\texttt{DUEL}(T_{x,g}, \mathbf{D}_1, \mathbf{D}_0) > 1/2\right]$$

As long as *one* of the players submits a good distribution, say $\mathbf{D}_b$, then E_b will be an $(1/50n)$-PCAPP algorithm for circuits of size n. The verifier then estimates $\mathbb{E}[T_{x,g}(\mathbf{U}_n)]$ to error $(1/10)$ using the dueling algorithm and thus correctly decide the language. Formally:

- For $x \in L$, let g be a witness of the **MA** protocol and $\mathbf{D}_1$ a good distribution. Then the YES-player submitting $(g, \mathbf{D}_1)$ will result in

  $$\texttt{DUEL}(T_{x,g}, \mathbf{D}_1, \mathbf{D}_0) \ge \frac{2}{3} - 4(n+2)\frac{1}{50n} > \frac{1}{2}$$

  so $\mathcal{V}$ will return 1.
- For $x \notin L$, the NO-player submitting a good distribution $\mathbf{D}_0$ will result in

  $$\texttt{DUEL}(T_{x,g}, \mathbf{D}_1, \mathbf{D}_0) \le \frac{1}{3} + 4(n+2)\frac{1}{50n} < \frac{1}{2}$$

  so $\mathcal{V}$ will return 0.

Thus, the $\mathbf{S_2P}$ protocol correctly decides the language.

B The Parameters of D2P

In this section we prove several extensions and corollaries complementing our main equivalence for D2P. In particular, we focus on understanding the parameters that we can hope to obtain for D2P transformations.

In Appendix B.1, we prove lower bounds on the parameters achievable by D2P transformations (even for weak classes of circuits), and provide a nearly-matching non-explicit construction. In Appendix B.2, we show that a slightly non-trivial D2P transformation (or a slightly non-trivial certified derandomization) would already lead to circuit lower bounds, following Williams' algorithmic method [Wil13]. Finally, in Appendix B.3, we study the best possible parameters possible for explicit D2P transformations, and obtain an equivalence between optimal D2P and superfast derandomization, assuming the existence of OWFs.

B.1 Existential Bounds on D2P

We first prove that D2P exists for every circuit, following Lemma 4.6. Recall that a majority gate $\texttt{MAJ}(x_1, x_2, \ldots, x_m)$ outputs 1 if and only if $x_1 + \cdots + x_m \geq \beta$, where $x_1, \ldots, x_m \in \{0,1\}$ and β is the parameter of the gate. Then:

Theorem B.1. *For every circuit $C : \{0,1\}^n \to \{0,1\}$ of size S, there is a $(1/3)$-distinguish to $(1/10n)$-predict transformation $\mathcal{P} = (P_1, \ldots, P_{O(n^2)})$ for C such that each P_i is an $S \cdot O(n^4)$ size circuit. Moreover, each P_i consists of a top $\texttt{MAJ}$ gate fed by $O(n^3)$ circuits obtained from C by fixing the last $n - i$ input bits. (In particular, each P_i is a $(\mathbf{TC}^0)^C$ circuit of size $O(n^4)$.)*

Proof. By Lemma 4.6, to construct a D2P transformation for a circuit $C : \{0,1\}^n \to \{0,1\}$ of size S, it suffices to give a circuit E that on input $x_< \in \{0,1\}^{\leq n}$, estimates $\mathbb{E}_r[C(x_< \circ r)]$ up to error $1/10n$. Note that there are at most $n \cdot 2^n$ such prefixes, and hence by the Chernoff bound (see Theorem 3.9) there is a distribution $\mathbf{D}$ over $\{0,1\}^n$ of support size $O(n^3)$ such that for every prefix $x_<$, we have

$$\left| \mathop{\mathbb{E}}_{r \leftarrow \mathbf{U}}[C(x_< \circ r)] - \mathop{\mathbb{E}}_{y \leftarrow \mathbf{D}}[C(x_< \circ y_>)] \right| \leq \frac{1}{10n}$$

where $\mathbf{U} \triangleq \mathbf{U}_{n-|x_<|}$ and $y_> \triangleq y_{>|x_<|}$. Let E be the circuit such that

$$E(C, x_<) = \mathop{\mathbb{E}}_{y \leftarrow \mathbf{D}}[C(x_< \circ y_>)],$$

then by Lemma 4.6 we have that there exists a $(1/3)$-distinguish to $(1/10n)$-predict transformation for C. The number of predictors is $O(n^2)$. Moreover, since the predictors in Lemma 4.6 output 1 if and only if $E(C, x_<)$ is larger than a threshold, each predictor P_i consists of a top $\texttt{MAJ}$ gate fed by $O(n^3)$ circuits $C(x_{<i} \circ y_{\geq i})$, where $y \in \mathbf{D}$ is of size n, so the size is as claimed. □

Note that since we only need our pseudorandom set to fool every restriction of a fixed circuit C of size S, rather than every restriction of *every circuit of size* S, we obtain a size blowup (and number of predictors) that is independent of the size of C.

Next, we show that this construction is essentially tight, up to the polynomial dependence on n. In particular, a generically valid D2P transformation for a class of circuits that contains $\mathbf{AC}^0_2$[44] on n input bits with size S must have the following three properties:

1. It must output $\widetilde{\Omega}(n)$ predictors.

[44] $\mathbf{AC}^0_d$ refers to $\mathbf{AC}^0$ circuits of depth d.

2. The size of the predictors must be S^δ, for a constant δ.

3. The guaranteed advantage must be at most $\widetilde{O}(1/n)$.

This essentially generalizes the "hybrid argument barrier" for constructing PRGs from hard truth tables (see, e.g., [FSU+13; SV22]), as every reconstructive argument using D2P must pay for the cost of each overhead. Finally, we remark that lower bounds hold even if we only consider D2P with respect to natural and efficiently samplable distributions (as will be evident from the proofs).

For the lower bounds, we relax the definition of D2P transformation, by allowing each predictor to read all but a single bit and then try to predict that bit.

Definition B.2. We say a set of functions $(P_1, \ldots, P_t)$ are a weak D2P transformation for C if each P_i gets as input $x_{[n]\setminus j_i}$ and attempts to predict the bit x_{j_i}.

We prove these properties as follows. For the first and third property, we use the TRIBES function, which we now define:

Definition B.3. Let $\texttt{TRIBES}_{m,\ell} : \{0,1\}^{m\cdot\ell} \to \{0,1\}$ be the function where, on input $(x^1, \ldots, x^m)$ where $x^i \in \{0,1\}^\ell$, computes:

$$\texttt{TRIBES}_{m,\ell}(x^1, \ldots, x^m) = \bigvee_{i\in[m]} \bigwedge_{j\in[\ell]} x^i_j.$$

We let V_i denote the bits corresponding to the i-th *term* of the tribes function for $i \in [m]$. Observe that

$$\mathbb{E}[\texttt{TRIBES}_{m,\ell}(\mathbf{U})] = 1 - (1 - 2^{-\ell})^m.$$

Observe that TRIBES can be computed by an $\mathbf{AC}^0_2$ circuit with $m + 1$ gates and $O(m\ell)$ wires.

A lower bound on the number of predictors. We show that any D2P (even a weak one) must output $\tilde{\Omega}(n)$ predictors.

Proposition B.4. *For every $n \in \mathbb{N}$, there is an $\mathbf{AC}^0_2$ circuit $C : \{0,1\}^n \to \{0,1\}$ of size n such that any weak $(1/3)$-distinguish to α-predict transformation for C with $\alpha > 0$ must output at least $n/2\log(n)$ predictors.*

Proof. Let $C = \texttt{TRIBES}_{m,2\log m}$ where $m = n/2\log(n)$ (note that indeed $2m\log(m) \le n$).[45] Observe that $\mathbb{E}[C(\mathbf{U}_n)] = o(1)$. Fix an arbitrary weak D2P transformation $(P_1, \ldots, P_t)$ for C, and assume towards a contradiction that $t < m$.

Recall that $j_i \in [n]$ is the bit that P_i attempts to predict. Since $t < m$, there is some term V such that $j_i \notin V$ for every $i \in [t]$. Then let $\mathbf{D}$ be the distribution that is uniform on $[m] \setminus V$ and equal to $1^{2\log m}$ on V. It is easy to see that $C(\mathbf{D}) = 1$ (and hence C $(1/3)$-distinguishes $\mathbf{D}$ from $\mathbf{U}_n$), yet no predictor attempts to predict an index where $\mathbf{D}$ is not uniform, so the transformation cannot obtain nonzero advantage. □

An upper bound on the prediction advantage. We show that any D2P (even a weak one) cannot output predictors that succeed with advantage more than $\tilde{O}(1/n)$.

Proposition B.5. *For every $n \in \mathbb{N}$, there is an $\mathbf{AC}^0_2$ circuit $C : \{0,1\}^n \to \{0,1\}$ such that any weak $(1/3)$-distinguish to α-predict transformation for C must have $\alpha \le 2\log n/n$.*

[45] For simplicity, we ignore rounding issues, as these do not substantially impact the proof.

Proof. We now let $C = \texttt{TRIBES}_{2m,\log m}$ where $m = n/\log(n)$ (note that again, $2m\log(m) \leq n$). Observe that now we have $\mathbb{E}[C(\mathbf{U}_n)] = 1 - o(1)$. Let $\ell = \log m$. In this case, let $\mathbf{D}$ be the distribution that is sampled as follows. We sample the bits of each term independently. For each term i, sample $x^i \leftarrow \mathbf{U}_\ell$ uniformly at random. If $x^i = \vec{1}^\ell$, set the final bit to 0, and otherwise make no changes. Observe that with this distribution we have $\mathbb{E}[C(\mathbf{D})] = 0$ and hence there must be a predictor with advantage $1/\alpha$. However, even just information-theoretically, every bit of $\mathbf{D}$ can only be predicted with advantage at most $1/2^{\ell-1} = 2/m = 2\log(n)/n$. □

A lower bound on the size of predictors. Finally, we show that a D2P transformation must output predictors of size that scale with the size of the circuit C. This follows from the standard analysis of the Nisan-Wigderson generator (see, e.g., [AB09, Chapter 18]). For simplicity, we will use the stronger version Theorem 6.15 from [DPT24] that has been discussed in Section 6.3.

Fact B.6 ([Sha49])**.** For every $n \in \mathbb{N}$, there is a truth table h on n input bits such that every circuit computing h has size greater than $2^n/O(n)$.

Proposition B.7. *There is a fixed constant $\delta > 0$ such that for every $n, S \in \mathbb{N}$ where $S \leq 2^n$, there is an $\mathbf{AC}^0_2$ circuit $C : \{0,1\}^n \to \{0,1\}$ of size S such that every weak $(1/3)$-distinguish to $(1/10n)$-predict transformation for C must contain a circuit of size at least S^δ.*

Proof. Let $s = \log(S)$. Let c_{NW} be the constant of Theorem 6.15, and choose $\varepsilon_{\mathsf{NW}} = 1/(4c_{\mathsf{NW}})$. Without loss of generality, we assume $S \leq 2^{n/8{c_{\mathsf{NW}}}^2}$ (as we can simply set S to this smaller size and change the constant δ appropriately).

Next, let $f \in \{0,1\}^S$ be a truth table on $\log S$ bits that is hard for circuits of size $S^{1/2}$, which exists per Fact B.6. Let $\mathsf{NW}^f : \{0,1\}^{t=4{c_{\mathsf{NW}}}^2 \log S} \to \{0,1\}^n$ be the PRG of Theorem 6.15 with the chosen parameters, where we furthermore truncate the output to n bits.[46] Observe that the seed length is $t = 4{c_{\mathsf{NW}}}^2 \log S \leq n/2$. Next, let C be the $\mathbf{AC}^0_2$ circuit of size $O(2^t) = O(S^{4{c_{\mathsf{NW}}}^2})$ such that

$$C(x) = \bigwedge_{y \in \mathrm{Im}(\mathsf{NW}^f)} (x = y).$$

Observe that $\mathbb{E}[C] \leq 2^{-n/2}$, yet letting $\mathbf{D} = \mathsf{NW}^f(\mathbf{U}_t)$ we have $\mathbb{E}[C(\mathbf{D})] = 1$. Thus, every D2P transformation for C must contain a predictor for $\mathbf{D}$. Now fix an arbitrary weak $(\infty, 10n)$-D2P transformation $\mathcal{P}$ for C. By the above, there must be $P \in \mathcal{P}$ such that $\mathrm{adv}_D(P) \geq 1/10n$. Applying the reconstruction claim of Theorem 6.15, we have that there is an oracle circuit R of size $S^{c_{\mathsf{NW}} \cdot \varepsilon_{\mathsf{NW}}} = S^{1/4}$ such that $R^P(x) = f_x$. Thus, as f does not have circuits of size $S^{1/2}$, we have

$$\mathrm{size}(R)\,\mathrm{size}(P) \geq \mathrm{size}(R^P) \geq S^{1/2}$$

and hence $\mathrm{size}(P) \geq S^{1/4}$. Finally, we take $S \leftarrow S^{1/4{c_{\mathsf{NW}}}^2}$ (and adjust δ appropriately) and so we obtain the final result, with $\delta = 1/16{c_{\mathsf{NW}}}^2$. □

Remark B.8. We suspect that, under the assumption that one-way functions exist, the constant $\delta > 0$ in Proposition B.7 can be made arbitrarily close to 1.

B.2 Slightly Non-Trivial D2P and Certified Derandomization Imply Circuit Lower Bounds

In this section, we show that slightly non-trivial D2P or certified derandomization suffice to imply circuit lower bounds.

[46] If the output is less than n bits, we can simply set n to be the output length and apply the subsequent argument.

Slightly non-trivial D2P implies circuit lower bounds. The equivalence between D2P transformation and derandomization is tight in parameters. Following Williams' algorithmic method for circuit lower bounds [Wil13], we prove that even a *slightly non-trivial* D2P transformation for a *restricted circuit class* would imply lower bounds for that circuit class. Since we want a statement that holds even for very weak circuit classes (i.e., every typical circuit class; see Section 3), we will use a version of Williams' method in [CW19] that is tight with respect to the class.

Theorem B.9 (non-trivial derandomization implies circuit lower bounds; [CW19])**.** *There is an absolute constant $\delta > 0$ such that for every typical circuit class $\mathcal{C}$, if there is a* CAPP *algorithm for $\mathcal{C}$ circuits $C : \{0,1\}^n \to \{0,1\}$ of size $\mathrm{poly}(n)$ with error δ that runs in $2^n/n^{\omega(1)}$ time, then* $\mathbf{NEXP} \not\subseteq \mathcal{C}[\mathrm{poly}(n)]$.

Relying on Theorem B.9, we prove the aforementioned connection between non-trivial D2P and circuit lower bounds:

Theorem B.10 (non-trivial D2P implies circuit lower bounds)**.** *There is an absolute constant $\delta > 0$ such that the following holds for every typical circuit class $\mathcal{C}$. Assume that there is a deterministic algorithm that gets a polynomial-size $\mathcal{C}$-circuit $C : \{0,1\}^n \to \{0,1\}$, runs in time $2^n/n^{\omega(1)}$, and outputs a δ-distinguish to α-predict transformation against m-size distributions for C, denoted $\mathcal{P} = (P_1, \ldots, P_t)$, where $\alpha = 2^{-o(n)}$, $t = 2^{o(n)}$, $\sum_i \mathrm{size}(P_i) \leq \alpha^2 \cdot 2^n/n^{\omega(1)}$ and $m \geq n/\alpha^2$. Then* $\mathbf{NEXP} \not\subseteq \mathcal{C}[\mathrm{poly}(n)]$.

Proof. Let δ be the constant in Theorem B.9. Suppose that there is a $2^n/n^{\omega(1)}$ time deterministic algorithm as described above, by Theorem B.9, it suffices to have a CAPP algorithm with error δ for $\mathcal{C}$ circuits of size $\mathrm{poly}(n)$.

Our CAPP algorithm works as follows. We run the algorithm described above to obtain a D2P transformation $\mathcal{P} = (P_1, \ldots, P_t)$ for C in $2^n/n^{\omega(1)}$-time. We then call the algorithm in Lemma 4.2 with the parameter values m and α to produce a distribution $\mathbf{D}$ of size $n/\alpha^2 = 2^{o(n)}$ such that for every $i \in [t]$, $\mathrm{adv}_{\mathbf{D}}(P_i) < \alpha$. Since $\mathcal{P}$ is a δ-distinguish to α-predict transformation against m-size distributions for C, it follows that $\mathbf{D}$ must δ-fool C. Our CAPP algorithm then outputs $\mathbb{E}_{x \leftarrow \mathbf{D}}[C(x)]$.

It remains to verify that the algorithm runs in $2^n/n^{\omega(1)}$ time. Our CAPP algorithm first calls the algorithm in the assumption, which takes $2^n/n^{\omega(1)}$ time. The algorithm in Lemma 4.5 runs in time

$$\tilde{O}\left(\frac{n}{\alpha^2}\sum_{i=1}^{t}\mathrm{size}(P_i)\right) + \mathrm{poly}(tn\alpha) \leq \frac{2^n}{n^{\omega(1)}} \quad ,$$

and outputs a distribution of size $|\mathbf{D}| = n/\alpha^2 = 2^{o(n)}$. Finally, computing $\mathbb{E}_{x \leftarrow \mathbf{D}}[C(x)]$ only takes $2^{o(n)}$ time as C is of polynomial size. □

This implies, for instance, that a slightly non-trivial D2P transformation for even depth-two majority circuits would imply $\mathbf{NEXP} \not\subseteq \mathtt{THR} \circ \mathtt{THR}$, a frontier open problem in circuit complexity (see, e.g., [CW19]).

Slightly non-trivial certified derandomization implies circuit lower bounds. Similarly, we will show that a slightly non-trivial certified derandomization – even for restricted circuit classes, and even in the relaxed notion of decision certified derandomization – implies circuit lower bounds.

Let $\mathcal{C}$ be a typical circuit class, let $s = s(n)$ be a size bound, and let $\delta \in (0, 1/2)$ be a constant. Recall that the problem $\mathcal{C}$-$\mathsf{GapUNSAT}$ with parameters (s, δ) is defined as follows: Given a $\mathcal{C}$-circuit $C : \{0,1\}^n \to \{0,1\}$ of size $s(n)$, we should accept if $\Pr_{r \in \{0,1\}^n}[C(r) = 1] = 0$, and reject if $\Pr[C(\mathbf{U}_n) = 1] \geq 1 - \delta$. Then:

Proposition B.11 (certified derandomization ⇒ GapUNSAT)**.** *Let $\mathcal{C}$ be a typical circuit class, let $\delta \in (0,1)$ be constant, and let $s(n) = \text{poly}(n)$. Assume that for some $\ell(n) = 2^{o(n)}$,*[47] *there is a property $\mathcal{P} = \{\mathcal{P}_n \subseteq \{0,1\}^\ell\}_{n\in\mathbb{N}}$ satisfying $\mathcal{P}_n \cap \{0,1\}^{\ell(n)} \neq \varnothing$ for every $n \in \mathbb{N}$, and a decision certified derandomization algorithm for s-size $\mathcal{C}$-circuits using $\mathcal{P}$ with parameters $\delta_0 = 0$ and $\delta_1 = \delta$ that, when given (C,τ) such that C has n input bits, runs in $2^n/n^{\omega(1)} \cdot \text{poly}(|\tau|)$.*

Then, there is a nondeterministic $2^n/n^{\omega(1)}$-time algorithm solving $\mathcal{C}$-GapUNSAT with parameters (s,δ). Moreover, for every input, for every non-deterministic guess w the algorithm either outputs the correct answer or outputs $\perp$, and there is at least one w such that the algorithm outputs the correct answer.

Proof. Fix $\delta, s(n) = \text{poly}(n)$ and $\ell(n) = 2^{o(n)}$ as described above. Let A be the certified derandomization algorithm using $\mathcal{P}$ with parameters (ℓ, s, δ). The nondeterministic algorithm works as follows. Given any $\mathcal{C}$-circuit $C : \{0,1\}^n \to \{0,1\}$ of size $s(n)$ that either rejects all inputs or accepts at least a $(1-\delta)$-fraction of the inputs, it nondeterministically guesses a string $\tau \in \{0,1\}^{\ell(n)}$, and if $A(C,\tau) = \perp$ then outputs $\perp$, and otherwise outputs $\neg A(C,\tau)$.

Observe that for every circuit $C\colon \{0,1\}^n \to \{0,1\}$ in the promise of $\mathcal{C}$-GapUNSAT:

- When $\tau \in \mathcal{P}_n$, then $A(C,\tau) = \begin{cases} 1 & \Pr_r[C(r)=1] \geq 1-\delta \\ 0 & \Pr_r[C(r)=1] = 0 \end{cases}$. Thus, whenever the algorithm guesses $\tau \in \mathcal{P}_n$, it outputs the correct answer.
- When $\tau \notin \mathcal{P}_n$, then $A(C,\tau) \in \begin{cases} \{1,\perp\} & \Pr_r[C(r)=1] \geq 1-\delta \\ \{0,\perp\} & \Pr_r[C(r)=1] = 0 \end{cases}$. Thus, whenever the algorithm guesses $\tau \notin \mathcal{P}_n$, it either outputs the correct answer or $\perp$.
- There is at least one $\tau \in \mathcal{P}_n$.

The running time of this algorithm is $2^n/n^{\omega(1)} \cdot \text{poly}(|\tau|) = 2^n/n^{\omega(1)}$. □

Using the algorithmic method (specifically, the results of Chen and Williams [CW19]), a non-trivial algorithm for certified derandomization implies circuit lower bounds. Specifically, we use the following result:

Theorem B.12 (Theorem 5 and Remark 7 of [CW19])**.** *There is an absolute constant $\delta \in (0,1)$ such that the following holds for every typical circuit class $\mathcal{C}$. Assume that there is a non-deterministic algorithm that gets as input a $\mathcal{C}$-circuit $C\colon \{0,1\}^n \to \{0,1\}$ of size $\text{poly}(n)$, runs in time $2^n/n^{\omega(1)}$, and satisfies the following: For every non-deterministic guess the algorithm either outputs $\perp$ or solves GapUNSAT for C ; and there is at least one non-deterministic guess such that the algorithm solves GapUNSAT for C. Then, **NEXP** does not have polynomial-sized $\mathcal{C}$-circuits.*

Corollary B.13 (certified derandomization implies circuit lower bounds)**.** *There is an absolute constant $\delta \in (0,1)$ such that the following holds for every typical circuit class $\mathcal{C}$. If for some $\ell(n) = 2^{o(n)}$ and every $s(n) = \text{poly}(n)$, there is a property $\mathcal{P} = \{\mathcal{P}_n \subseteq \{0,1\}^\ell\}_{n\in\mathbb{N}}$ satisfying $\mathcal{P}_n \cap \{0,1\}^{\ell(n)} \neq \varnothing$ for every $n \in \mathbb{N}$, and a decision certified derandomization algorithm for s-size $\mathcal{C}$-circuits using $\mathcal{P}$ with parameters $\delta_0 = 0$ and $\delta_1 = \delta$ that, when given (C,τ) such that C has n input bits, runs in $2^n/n^{\omega(1)} \cdot \text{poly}(|\tau|)$. Then, **NEXP** does not have polynomial-sized $\mathcal{C}$-circuits.*

[47] The only reason for the assumption that $\ell(n) = 2^{o(n)}$ is since we assume that the certified derandomization algorithm runs in time $\text{poly}(|\tau|)$.

B.3 Near Optimal D2P and Superfast Derandomization

What are the best parameters for derandomized D2P that we can hope for? In Appendix B.1 we prove lower bounds asserting that, when the number of bits in the unpredictable/pseudorandom distribution is r, the number of predictors must be at least $\tilde{\Omega}(r)$ and the advantage can be at most $\tilde{O}(1/r)$ (even if we only focus on natural subclasses of distributions).

Note that the foregoing lower bounds hold with respect to the number of unpredictable or pseudorandom bits r, but do not take into account the running time (or size) of the distinguisher or predictor. Moreover, when derandomizing **prBPP** (or related class), there is usually also an input x to the problem; that is, the distinguisher is a probabilistic machine that takes an input $x \in \{0,1\}^n$ and random coins $s \in \{0,1\}^r$, and tries to predict or distinguish s.

Since we now care about the best possible D2P, we will carefully distinguish between these parameters. In particular, we think of D2P transformations with respect to problem $\Pi \in \mathbf{prBPTIME}[T]$ (rather than with respect to a fixed distinguisher). We define a notion of near optimal D2P as one that essentially meets the lower bounds in Appendix B.1 while adding essentially no runtime overhead:

Definition B.14 (near optimal D2P)**.** Let Π be a promise problem that can be decided by a probabilistic machine M running in time $T = T(n)$ and using $r = r(n)$ random coins. We say that Π is admits a near optimal D2P transformation with respect to M if for every constant $\varepsilon > 0$, there is an algorithm that gets input $x \in \{0,1\}^n$, runs in time $T(n) \cdot n^{1+\varepsilon} \cdot \mathrm{poly}(r(n))$, and outputs a $(1/3)$-distinguish to $(1/10r(n))$-predict transformation $\mathcal{P} = \{P_1, \ldots, P_t\}$ for the distinguisher $M(x, \cdot) : \{0,1\}^r \to \{0,1\}$, where $t = \mathrm{poly}(r(n))$.

Indeed, our definition allows overhead of $\mathrm{poly}(r)$ in the running time and in the number of predictors (which is why we refer to it as "near optimal"). The reason is that our equivalence result below tolerates this overhead (since it relies on one-way functions, which allow to reduce r to $r = T^{o(1)}$; see below). However, we believe that a stricter definition also makes sense and is worth investigating.

One might wonder, though, why we allow running time $\approx T \cdot n$ rather than require running time $\approx T$. The reason is that there is a tight equivalence between D2P and derandomization, and under reasonable assumptions the latter incurs such overhead. In particular, assuming one-way functions, we prove a tight equivalence between *near-optimal D2P* and *near-optimal derandomization* (i.e., superfast derandomization). As shown by Chen and Tell [CT21b], following Williams [Wil16], under the assumption #NSETH, worst-case derandomization of **BPP** must incur an overhead of $T \mapsto n \cdot T$. Thus, we do not expect D2P that does not incur this overhead.

Following recent results studying superfast derandomization (see, e.g., [DMO+22; CT21b; CT21a; CT23b; DT23; DPT24]), we will need to rely on an assumption. In particular, we rely on the following assumption, which is implied by the existence of (non-uniformly secure) one-way functions (see, e.g., [CT21b] for a standard explanation):

Assumption B.15 (near-linear-time PRGs with polynomial stretch)**.** For every $k \in \mathbb{N}$ and $\varepsilon > 0$, there exists a function $G_{cry} : \{0,1\}^n \to \{0,1\}^{n^k}$ such that for every circuit $C : \{0,1\}^{n^k} \to \{0,1\}$ of size $\mathrm{poly}(n)$, we have that G_{cry} $(1/n^k)$-fools C, and $G_{cry}(x)$ runs in time $n^{k+\varepsilon}$.

Theorem B.16 (near-optimal D2P is equivalent to near-optimal derandomization, assuming OWFs)**.** *Suppose that Assumption B.15 is true. Then, the following statements are equivalent:*

(1) For every polynomial $T(n)$, problem $\Pi \in \mathbf{prBPTIME}[T(n)]$, and probabilistic $T(n)$-time algorithm M for Π, Π admits a near optimal D2P transformation with respect to M.

(2) For every polynomial $T(n)$ and every $\varepsilon > 0$, we have $\mathbf{prBPTIME}[T(n)] \subseteq \mathbf{DTIME}[T(n) \cdot n^{1+\varepsilon}]$.

Proof. (1) $\Rightarrow$ (2). Fix any polynomial $T = T(n)$ and $\varepsilon > 0$. Let $\Pi \in \mathbf{prBPTIME}[T(n)]$, let M be a probabilistic algorithm using $r(n) = T(n)$ random bits, and fix an input $x \in \{0,1\}^n$. Using naive error reduction, it suffices to distinguish between $\mathbb{E}[M(x, \mathbf{U}_r)] \geq 0.9$ and $\mathbb{E}[M(x, \mathbf{U}_r)] \leq 0.1$.

Let $\varepsilon' \triangleq \varepsilon/c'$ and $r' \triangleq T^{\varepsilon'}$ for some sufficiently large constant $c' > 1$ to be determined later, and $M'(x, \cdot) : \{0,1\}^{r'} \to \{0,1\}$ be the Turing machine defined as $M'(x, u) \triangleq M(x, G_{cry}(u))$, where $G_{cry} : \{0,1\}^{r'} \to \{0,1\}^T$ is the cryptographic PRG in Assumption B.15. Since the cryptographic PRG G_{cry} is $(1/T)$-pseudorandom for any circuit of size $\mathrm{poly}(T)$, the machine M' solves the same promise problem as M.[48] Also, since G_{cry} runs in time $T^{1+\varepsilon'}$, the running time of M' is $O(T^{1+\varepsilon'})$. Since M' uses only $r' = T^{\varepsilon'}$ random coins, by (1), there is a near optimal D2P transformation for Π with respect to M', and therefore we can obtain in time $T(n)^{1+\varepsilon'} \cdot n^{1+\varepsilon'} \cdot \mathrm{poly}(r')$ a $(1/3)$-distinguish to $(1/10r')$-predict transformation $\mathcal{P} = \{P_1, \ldots, P_t\}$ for $M'(x, \cdot)$, where $t = \mathrm{poly}(r')$.

We then apply the result of Lemma 4.2 with the transformation $\mathcal{P}$ and the parameter value α. From this, we obtain a distribution $\mathbf{D}$ of size $\mathrm{poly}(r')$ such that $\mathbf{D}$ $(1/3)$-fools $M'(x, \cdot)$ in time

$$\tilde{O}(T(n) \cdot n^{1+\varepsilon'} \cdot \mathrm{poly}(r')/\alpha^2) + \mathrm{poly}(tr'/\alpha) \leq T(n) \cdot n^{1+O(\varepsilon')}.$$

We then compute $\rho = \mathbb{E}[M(x, G_{cry}(\mathbf{D}))]$ in deterministic time $T^{1+\varepsilon'} \cdot n^{1+\varepsilon'} \cdot \mathrm{poly}(r') \leq T(n) \cdot n^{1+O(\varepsilon')}$ and return ρ. The desired outcome follows by a sufficiently large choice of c'.

(2) $\Rightarrow$ (1). Fix any polynomial $T = T(n)$ and $\varepsilon > 0$. Let $\Pi \in \mathbf{prBPTIME}[T(n)]$ and M be an $T(n)$-time probabilistic algorithm for Π using $r = r(n) \leq T(n)$ random bits.

Consider the following problem Π': Given any $x \in \{0,1\}^n$, a prefix $u \in \{0,1\}^{\leq r}$, and $\tau \in [10r]$, decide whether $\mathbb{E}[M(x, u \circ \mathbf{U}_{r-|u|})] \geq \tau/10r$ or $\mathbb{E}[M(x, u \circ \mathbf{U}_{r-|u|})] < (\tau - 1)/10r$. (In other words, the $(1/10r)$-PCAPP problem for $M(x, \cdot)$.) Observe that the problem Π' can be solved in probabilistic time $O(T(n) \cdot \mathrm{poly}(r))$, by sampling sufficiently many strings of length $r - |u|$ and simulating M. The length of an input to Π' is $m = n + r + O(\log(10r)) = O(n + r)$, and therefore $\Pi' \in \mathbf{prBPTIME}[T(n) \cdot \mathrm{poly}(r(n))]$ (where we use m to denote the input length of Π'). By our assumption, $\Pi' \in \mathbf{DTIME}[T(n) \cdot \mathrm{poly}(r(n)) \cdot m^{1+\varepsilon}]$ (where m is the input length of Π').

Now we describe the near optimal D2P transformation with respect to M. Fix any input $x \in \{0,1\}^n$ and consider the circuit $C(\cdot) \triangleq M(x, \cdot) : \{0,1\}^r \to \{0,1\}$. Using Lemma 4.6, we can construct a $(1/3)$-distinguish to $(1/10r)$-predict transformation for C of size $t = \mathrm{poly}(r)$ in time

$$T(n) \cdot m^{1+\varepsilon} \cdot \mathrm{poly}(r(n)) \leq T(n) \cdot n^{1+\varepsilon} \cdot \mathrm{poly}(r(n))$$

by plugging the deterministic algorithm for Π' (as the $(\delta/3r)$-PCAPP algorithm E for $\delta = 1/3$) into each predictor in Lemma 4.6. □

Universal superfast derandomization. The proof of Theorem B.16 has an interesting corollary: Assuming OWFs, there is a *universal* two-sided error superfast derandomization algorithm for any problem. That is, for any $\Pi \in \mathbf{prBPP}$, there is a *fixed* algorithm such that if OWFs exist and superfast derandomization is possible, the algorithm is a correct superfast derandomization algorithm for Π when the input length is sufficiently large.

Note that universal derandomization (not necessarily a superfast one) is implied by the standard reduction from two-sided error derandomization to one-sided error derandomization. Specifically,

[48] Recall that the computation of M on any input x as a function of the random coins (i.e., $M(x, \cdot)$) can be modeled as a circuit of size $\tilde{O}(T) \leq \mathrm{poly}(T)$.

recall that there is a universal one-sided error derandomization algorithm,[49] and there is a reduction of two-sided error derandomization to one-sided error derandomization [Sip83; Lau83; BF99; GZ11].

Relying on the ideas in the proof of Theorem B.16, we show a universal two-sided error derandomization algorithm that does not incur overheads. The crucial observation is that we can reduce CAPP to D2P, and construct a universal D2P algorithm.

Corollary B.17 (universal superfast derandomization)**.** *For every polynomial $T(n)$, $\varepsilon > 0$, and problem $\Pi = (\Pi^{\mathsf{YES}}, \Pi^{\mathsf{NO}}) \in \mathbf{prBPTIME}[T(n)]$, there is a deterministic Turing machine D_Π running in time $T(n) \cdot n^{1+O(\varepsilon)}$ such that the following holds (where $O(\cdot)$ hides an absolute constant).*

There is a polynomial $T'(n) = T(n) \cdot n^{O(\varepsilon)}$ such that if cryptographic OWFs secure against polynomial-size circuits exist and $\mathbf{prBPTIME}[T'(n)] \subseteq \mathbf{DTIME}[T'(n) \cdot n^{1+\varepsilon}]$, then for every sufficiently large n and $x \in \{0,1\}^n$, $D_\Pi(x) = 1$ (resp. $D_\Pi(x) = 0$) if $x \in \Pi^{\mathsf{YES}}$ (resp. $x \in \Pi^{\mathsf{NO}}$).

Proof. Fix any polynomial $T(n)$, $\varepsilon > 0$, and $\Pi \in \mathbf{prBPTIME}[T(n)]$. The algorithm $D_\Pi(x)$ follows "(1) $\Rightarrow$ (2)" in the proof of Theorem B.16 with two exceptions.

- We instantiate a universal pseudorandom generator (i.e. a pseudorandom generator that is secure as long as OWFs exist). For instance, we can use a universal OWF [Lev87; LP20] and apply the standard transformation to construct a pseudorandom generator [HIL+99; MP23]. Moreover, we can assume without loss of generality (see, e.g., [Gol01]) that the PRG satisfies the seed length and running time requirement in Assumption B.15.

- Let M be a probabilistic algorithm for Π, $r \leq T$ be randomness complexity of M. Assume without of loss of generality that $r = T$. Let $r' = r^\varepsilon$ be the seed length of the cryptographic PRG $G_{cry} : \{0,1\}^{r'} \to \{0,1\}^r$, and M' be defined as the proof of Theorem B.16.

  To obtain a D2P for M', we enumerate the first $O(\log \log n)$ Turing machines (under any reasonable encoding), simulate them on input x for $T(n) \cdot n^{1+\varepsilon} \cdot r'^d$ step for some constant d to be determined later, and parse the outputs of the Turing machines as candidate $(1/3)$-distinguish to $(1/10r')$-predict transformations of size $t = r'^d$.

- We then apply Lemma 4.2 with $\mathcal{P}$ being the union of all candidate D2P transformations and with parameter α, to obtain a distribution $\mathbf{D}$ over $\{0,1\}^{r'}$ of size $\mathrm{poly}(r')$ that $(1/3)$-fools $M'(x, \cdot)$ (as long as one of the candidate D2P transformations is correct).

- Our algorithm accepts if and only if $\mathbb{E}[M'(x, \mathbf{D})] > 0.51$.

It is clear that the algorithm runs in time $T(n) \cdot n^{1+\varepsilon} \cdot r'^{O(d)} = T(n) \cdot n^{1+O(\varepsilon)}$ (where the constant in $O(\cdot)$ on the exponent depends on d that will be determined later), and therefore it remains to argue the correctness. Following "(2) $\Rightarrow$ (1)" in the proof of Theorem B.16, we know that under the assumptions of OWFs and superfast derandomization (for $T'(n) = T(n) \cdot \mathrm{poly}(r') = T(n) \cdot n^{O(\varepsilon)}$ time randomized algorithms), there is a deterministic $T'(n) \cdot n^{1+\varepsilon} \cdot \mathrm{poly}(r')$ time algorithm such that given x, outputs a $(1/3)$-distinguish to $(1/10r')$-predict transformation for the distinguisher $M'(x, \cdot) : \{0,1\}^{r'} \to \{0,1\}$ of size $t = \mathrm{poly}(r')$. Moreover, it can be verified that $\mathrm{poly}(r')$ above hides an absolute constant on the exponent that is independent of the assumption $\mathbf{prBPTIME}[T'(n)] \subseteq \mathbf{DTIME}[T'(n) \cdot n^{1+\varepsilon}]$.

Therefore, for sufficiently large n, this algorithm will appear as one of the first $O(\log \log n)$ Turing machines. By setting d to be a sufficiently large constant (depending on the constant hidden

[49] Specifically, if $\mathbf{prRP} = \mathbf{prP}$ then $\mathbf{prBPP} = \mathbf{prP}$, which implies the existence of a targeted PRG (see [Gol11b; Gol11c]). Given input x, the universal derandomization algorithm enumerates the outputs of the first (say) $\log(n)$ machines, trying to find a random string that will make the relevant machine M accept x.

in poly(r') above), $D_\Pi(x)$ will generate a correct D2P transformation and thus the distribution $\mathbf{D}$ will (1/3)-fool $M'(x,\cdot)$. The correctness of D_Π then follows from the security of the pseudorandom generator G_{cry}. □

Remark B.18. Indeed, it is implicit in the proof of Corollary B.17 that the pseudorandom distribution $\mathbf{D}$ generated by the universal superfast derandomization algorithm D_Π is *independent* of the problem Π.

C Alternate Derandomizations of Yao's Transformation

In this section, we study two alternate notions of derandomization of Yao's transformation that are different from D2P and certified derandomization. We prove in Appendix C.1 that "black-box" derandomization of Yao's transformation is equivalent to the existence of hitting sets, which implies (but may not be implied by) **prBPP** = **prP**. In Appendix C.2, we prove that a weaker "non-black-box" derandomization of Yao's transformation is equivalent to **prBPP** = **prLOSSY**, which is (possibly) stronger than **prBPP** = **prZPP** but weaker than **prBPP** = **prP**.

C.1 Black-Box Yao Derandomization

We observe that a "maximally black box" derandomization of the Yao transformation is equivalent to constructing hitting sets. We will first define this notion, which refers to finding a suitable suffix for Yao's transformation of distinguishers to predictors (see Lemma 3.8) that works for *every* distribution $\mathbf{D}$ of fixed size.

Definition C.1 (black-box derandomization of Yao's transformation)**.** We say $\{S_n \subseteq \{0,1\}^n\}_{n\in\mathbb{N}}$ is an (m,α)-black-box Yao family (BB-Yao family) for $\mathcal{C}$ circuits if for every $C \in \mathcal{C}$ of size $O(n)$, and distribution $\mathbf{D}$ over $\{0,1\}^n$ of size at most m that does not (1/3)-fool C, there is $i \in [n], \sigma \in \{0,1\}^2$, and $z \in S_n$ such that

$$\Pr_{x\leftarrow\mathbf{D}}[C(x_{<i} \circ \sigma_1 \circ z_{>i}) \oplus \sigma_2 = x_i] > \frac{1}{2} + \alpha.$$

It is easy to show that (m,α)-BB-Yao family of size poly$(m, 1/\alpha)$ exists for every $m = m(n)$ and $\alpha = \alpha(n) \geq 1/(10n)$ using the probabilistic method. We note, however, that it is unclear whether there is a $(m = \infty, \alpha)$-BB-Yao family (beyond the trivial construction of including all possible prefixes).[50]

This definition captures the most restricted strategy of derandomizing Yao's transformation, that of simply producing a set of suffixes that are good for every (fixed-size) bad distribution. We remark that derandomizing Yao in this fashion has some advantages. In particular, the added complexity of the predictor is essentially as low as possible (a negation of a projection circuit that makes a single call to C).

We show that derandomizing Yao in this restricted fashion is equivalent to explicit constructions of hitting sets.

Theorem C.2. *The followings are equivalent for any typical class of circuits $\mathcal{C}$ such that* $(\mathtt{MAJ} \circ \mathcal{C})[s] \subseteq \mathcal{C}[\text{poly}(s)]$. [51]

[50] Our existence proof for a BB-Yao family relies on a union bound over all bad distributions, of which there could be doubly-exponentially many if m was unbounded.

[51] Note that MAJ denotes a majority gate with a threshold: Given $x = x_1 \circ \cdots \circ x_n$, it outputs 1 if and only if $x_1 + \cdots + x_n \geq \beta$ for a parameter β of the gate.

(1) There is an explicit family of $(1/2)$*-hitting sets* $H_n \subseteq \{0,1\}^n$ *of size* $\mathrm{poly}(n)$ *for* $\mathcal{C}[n]$.

(2) There is an explicit $(m = 2, \alpha = 1/10n)$*-BB-Yao family for* $\mathcal{C}[n]$.

(3) For every $k \in \mathbb{N}$*, there is an explicit* $(m = n^k, \alpha = 1/10n)$*-BB-Yao family for* $\mathcal{C}[n]$.

Observe that (3) immediately implies (2), so to complete the proof it suffices to show (1) $\Rightarrow$ (3) and (2) $\Rightarrow$ (1). Note that our assumption that $\mathcal{C}$ is typical allows us to assume we have a $(1/\mathrm{poly}(n))$-HSG (i.e. reduce the error).

Remark C.3. We note that it is known (following a classic trick, see, e.g., [HH23, Theorem 4.1.2]) that the existence of an explicit family of $(1/2)$-hitting sets H_n for general Boolean circuits is equivalent to $\mathbf{E} \not\subseteq$ i.o. $\mathbf{SIZE}[2^{\Omega(n)}]$, which itself implies an explicit family of $(1/2-\varepsilon)$-pseudorandom sets for general Boolean circuits [NW94; IW97]. Thus, an explicit BB-Yao family for general circuit is essentially equivalent to the existence of PRGs. Moreover, this fact holds also for weaker circuit classes, in fact even for $\mathbf{TC}^0$ (see [DT23, Theorem 5.1]).

Proof of the equivalence. We first show that an HSG gives a BB-Yao family.

Lemma C.4. *Let* $\mathcal{C}$ *be typical. Suppose* $H \subseteq \{0,1\}^n$ *is a* $(1/10n)$*-hitting set for* $\mathtt{MAJ} \circ \mathcal{C}$ *circuits of size* $2 \cdot n^{k+1}$*. Then* H *is an* $(m = n^k, \alpha = 1/10n)$*-BB-Yao set for* $\mathcal{C}$ *circuits.*

Proof. Fix an arbitrary distribution $\mathbf{D} \subseteq \{0,1\}^n$ of size n^k such that $\mathbf{D}$ does not $(1/3)$-fool a linear-size $\mathcal{C}$ circuit C. By Lemma 3.8, there exists $i \in [n]$ and $\sigma \in \{0,1\}^2$ such that the following holds:

$$\Pr_{z \leftarrow \mathbf{U}_{n-i}}\left[\mathrm{adv}_{\mathbf{D}}(P_{z,\sigma,i}) \geq \frac{1}{10n}\right] \geq \frac{1}{10n}$$

where $P_{z,\sigma,i}$ is the Yao predictor defined as $P_{z,\sigma,i}(x) = C(x_{<i} \circ \sigma_1 \circ z) \oplus \sigma_2$. Thus, the circuit $T_{C,\mathbf{D}}$ that takes in z and computes $a = \mathrm{adv}_{\mathbf{D}}(P_{z,\sigma,i})$ and accepts if $a \geq 1/10n$ satisfies $\mathbb{E}_z[T_{C,\mathbf{D}}(z)] \geq 1/10n$. Moreover, the circuit has low complexity:

Claim C.5. $T_{C,\mathbf{D}}$ *is a* $\mathtt{MAJ} \circ \mathcal{C}$ *circuit of size* $m \cdot \mathrm{size}(C) + 1 \leq 2n^{k+1}$.

Proof. Let $\mathbf{D} = (x^1, \ldots, x^{n^k})$. The circuit first computes $\{C(x^j_{<i} \circ z)\}_{j \in [n^k]}$ in parallel in size $n^k \cdot \mathrm{size}(C)$. Then the circuit uses the top $\mathtt{MAJ}$ gate to compute the advantage obtained by z (and accepts if it is sufficiently high). □

Thus, by the properties of the hitting-set there is $z \in H$ such that $T_{C,\mathbf{D}}(z) = 1$. But then the Yao predictor induced by this z predicts $\mathbf{D}$ with advantage at least $(1/10n)$, and hence (as $\mathbf{D}$ was arbitrary) we have that H induces a BB-Yao family with the claimed parameters. □

Note that this proof requires that H hits circuits that we cannot explicitly construct (as we do not have access to the bad distribution $\mathbf{D}$), which is the reason why we require a hitting set. Next, we show that any black-box Yao set implies a (closely related) hitting set.

Lemma C.6. *Suppose* S_n *is a* $(m = 2, \rho)$*-black-box Yao family for* $\mathcal{C}$ *circuits, where* $\rho > 0$ *is arbitrary. Then*

$$H = \left\{\vec{0}_{<i} \circ \sigma_1 \circ z_{>i}, \vec{1}_{<i} \circ \sigma_1 \circ z_{>i} : i \in [n], \sigma_1 \in \{0,1\}, z \in S_n\right\}$$

is a $(1/2)$*-hitting set for* $\mathcal{C}[n]$.

Proof. Assume for contradiction this is not the case, i.e., there is a $\mathcal{C}$ circuit $C : \{0,1\}^n \to \{0,1\}$ of size n where $\mathbb{E}[C(\mathbf{U}_n)] \geq 1/2$ but $C(u) = 0$ for every $u \in H$. By construction, this implies $C(\vec{1}) = C(\vec{0}) = 0$, and hence the distribution

$$\mathbf{D} \stackrel{\text{def}}{=} \left\{\vec{1}, \vec{0}\right\}$$

likewise fails to 0.49-fool C. Therefore, by the correctness of the BB-Yao set, there must be $i \in [n]$, $\sigma \in \{0,1\}^2$, and $z \in S$ such that:

$$\Pr_{x \leftarrow \mathbf{D}}[C(x_{<i} \circ \sigma_1 \circ z_{\geq i}) \oplus \sigma_2 = x_i] \geq \frac{1}{2} + \rho.$$

However, note that we have

$$C\left(\vec{1}_{<i} \circ \sigma_1 \circ z_{>i}\right) = C\left(\vec{0}_{<i} \circ \sigma_1 \circ z_{>i}\right) = 0$$

since otherwise H would hit C by construction. Therefore the predictor must predict $P(\vec{1}_{<i}) = P(\vec{0}_{<i}) = \sigma_2$, i.e. output the same value on both prefixes. But over $x \leftarrow \mathbf{D}$, the marginal distribution of the i-th bit is exactly uniform, and hence such a predictor cannot achieve any nonzero advantage, which contradicts the validity of the BB Yao family. Therefore, H must hit C. □

We can then prove the equivalence:

Proof of Theorem C.2. (1) ⇒ (3). Fix arbitrary k and assume that there is an explicit family of (1/2)-hitting sets H_n of size $\text{poly}(n)$ for $\mathcal{C}[n]$. We will show that there is an explicit $(1/10n)$-hitting set $\hat{H}_n \subseteq \{0,1\}^n$ of size $\text{poly}(n)$ for $\texttt{MAJ} \circ \mathcal{C}[2n^{k+1}]$. If this is true, then by applying Lemma C.4 we will obtain the desired result.

To construct $\hat{H}_n$, we first apply the hypothesis to $n' = n^c \cdot t$ for $t = O(n)$ and some sufficiently large constant c to be determined later to obtain a HSG $H_{n'} \subseteq \{0,1\}^{n'}$ that fools n'-input $\mathcal{C}[n']$ circuits. Let $\hat{H}_n \subseteq \{0,1\}^n$ be the set of size $|H'| \cdot t$ obtained as follows: For each string $x \in \{0,1\}^{n^c \cdot t}$ in H', we split $x_{\leq nt}$ into t pieces of length-n strings and put them into $\hat{H}_n$.

Now we prove that $\hat{H}_n$ is a desired hitting set. Towards a contradiction, we assume that there is a $\texttt{MAJ} \circ \mathcal{C}[2n^{k+1}]$ circuit $C : \{0,1\}^n \to \{0,1\}$ that breaks the HSG $\hat{H}_n$, i.e., $\mathbb{E}[C(\mathbf{U}_n)] \geq 1/10n$ but $\hat{H}_n$ does not hit C. Since $\texttt{MAJ} \circ \mathcal{C}[2n^{k+1}] \subseteq \mathcal{C}[\text{poly}(n)]$, we can construct a $\mathcal{C}[\text{poly}(n)]$ circuit $C' : \{0,1\}^n \to \{0,1\}$ that is equivalent to C. Let $C'' : \{0,1\}^{n'} \to \{0,1\}$ be the following circuit:

- Given any input $x \in \{0,1\}^{n'}$, it splits $x_{\leq nt}$ input t pieces of length-n strings $u_1, u_2, \ldots, u_t$ and outputs 1 if $C(u_i) = 1$ for some $i \in [t]$.

Clearly, $\mathbb{E}[C''(\mathbf{U}_{n'})] \geq 1/2$ for some $t = O(n)$.

Moreover, this circuit can be implemented by an OR over t copies of C', and since an OR gate can be simulated by a $\texttt{MAJ}$ gate, C'' is a $\texttt{MAJ} \circ \mathcal{C}$ circuit of size $\text{poly}(n)$, thus it is also a $\mathcal{C}$ circuit of size $\text{poly}(n)$. Note that $\text{poly}(\cdot)$ hides an absolute constant independent of c, and C'' only depends on its first nt inputs. By setting c to be a sufficiently large constant, C'' will be an $O(n')$-size n'-input $\mathcal{C}$ circuit, and hence is hit by $H_{n'}$ by the assumption. By the definition of $\hat{H}_n$ and C'', we can see that $\hat{H}_n$ hits C, which leads to a contradiction.

(3) ⇒ (2). Trivial.

(2) ⇒ (1). Let S be the explicit black-box Yao family for $\mathcal{C}$ circuits of size n. By Lemma C.6, we have an explicit construction of a hitting set for $\mathcal{C}$ circuits of size n, so we are done. □

C.2 Non-Black-Box Yao Derandomization

We now show that **prBPP** = **prLOSSY** if and only if a "non-black-box" derandomization of Yao's transformation (i.e., of Lemma 1.3) is possible. We think of the latter as the problem of transforming a circuit C into a predictor $P(x_{<i}) = C(x_{<i} \circ z_{\geq i})$ for a *given* distribution $\mathbf{D}$, with the main challenge being finding a suitable suffix z. More formally:

Definition C.7. Let $\varepsilon = \varepsilon(n) \in (0,1)$ be a parameter. A non-black-box Yao's transformation is the following search problem. Given any $O(n)$-size circuit $C : \{0,1\}^n \to \{0,1\}$ and a distribution $\mathbf{D}$ over $\{0,1\}^n$ that fails to ε-fool C, the solution to the search problem is $z \in \{0,1\}^n$ such that there is an $i \in [n]$ and $\sigma_1, \sigma_2 \in \{0,1\}$ satisfying that

$$\Pr_{x \leftarrow \mathbf{D}}[C(x_{<i} \circ \sigma_1 \circ z_{>i}) \oplus \sigma_2 = x_i] \geq \frac{1}{2} + \frac{\varepsilon}{5n}.$$

By default, we use the parameter value $\varepsilon = 1/3$.

This problem serves as an intermediate problem in Korten's proof for the **prBPP**-hardness of R-LossyCode (see [Kor22] for the definition). Korten observed that if there is a deterministic polynomial-time algorithm for non-black-box Yao transformation, then **prBPP** $\subseteq$ **prNP** (see [Kor22, Corollary 41]). Given that we have nice characterizations for black-box derandomization (see Theorem C.2) and D2P transformation (see Theorem 4.1), it is natural to ask whether there is a characterization for non-black-box derandomization of Yao's transformation.

It turns out that non-black-box derandomization of Yao's transformation is polynomial-time reducible to LossyCode (i.e. it is in **FLOSSY**) if and only if **prBPP** = **prLOSSY**.

Theorem C.8. *The following statements are equivalent.*

(1) **prBPP** = **prLOSSY**.

(2) Non-black-box derandomization of Yao's transformation is in **FLOSSY**.

Proof. (1) $\Rightarrow$ (2). Assume that **prBPP** = **prLOSSY**, CAPP reduces to LossyCode. Notice that non-black-box Yao derandomization is a **prBPP**-search problem (see [Gol11b] for definition). More formally, we define a promise problem $\Pi = (\Pi^{\mathsf{YES}}, \Pi^{\mathsf{NO}}) \in$ **prBPP** as follows.

- (*YES-instance*). $(C, \mathbf{D}, z) \in \Pi^{\mathsf{YES}}$ if $\mathbf{D}$ fails to ε-fools C, and there is $i \in [n]$ and $\sigma_1, \sigma_2 \in \{0,1\}$ such that
$$\Pr_{x \leftarrow \mathbf{D}}[C(x_{<i} \circ \sigma_1 \circ z_{>i}) \oplus \sigma_2 = x_i] \geq \frac{1}{2} + \frac{\varepsilon}{3n}.$$
- (*NO-instance*). $(C, \mathbf{D}, z) \in \Pi^{\mathsf{NO}}$ if $\mathbf{D}$ fails to ε-fool C, but for every $i \in [n]$ and $\sigma_1, \sigma_2 \in \{0,1\}$,
$$\Pr_{x \leftarrow \mathbf{D}}[C(x_{<i} \circ \sigma_1 \circ z_{>i}) \oplus \sigma_2 = x_i] < \frac{1}{2} + \frac{\varepsilon}{5n}.$$

Note that for every $(C, \mathbf{D})$ such that $\mathbf{D}$ fails to ε-fool C, $(C, \mathbf{D}, z) \in \Pi^{\mathsf{YES}}$ for a constant fraction of $z \in \{0,1\}^n$ (see Lemma 3.8). By [Gol11b, Theorem 3.5], there is a deterministic algorithm with a CAPP oracle that (given $(C, \mathbf{D})$) outputs a $z \in \{0,1\}^n$ such that $(C, \mathbf{D}, z) \notin \Pi^{\mathsf{NO}}$, or equivalently, outputs a solution z for non-black-box Yao derandomization. Recall that CAPP reduces to LossyCode, and therefore there is also an algorithm to output such z using a LossyCode oracle.

(2) $\Rightarrow$ (1). The proof idea is a modification of Korten's proof of the **prBPP**-hardness of R-LossyCode [Kor22, Corollary 41]. Suppose that non-black-box derandomization of Yao's transformation reduces to LossyCode. By Lemma 7.9, a single oracle query to LossyCode suffices. Let $\mathcal{R}$ be such a reduction.

Now we describe a reduction of CAPP to LossyCode. Given any circuit $C : \{0,1\}^n \to \{0,1\}$, we construct an instance $C' : \{0,1\}^\ell \to \{0,1\}^{\ell-1}, D' : \{0,1\}^{\ell-1} \to \{0,1\}^\ell$ as follows.

- For any distribution $\mathbf{D}$ of size n^4, the reduction generates some LossyCode instance $C'' : \{0,1\}^m \to \{0,1\}^{m-1}, D'' : \{0,1\}^{m-1} \to \{0,1\}^m$ given the instance $(C, \mathbf{D})$ of derandomizing Yao, where $m = \mathrm{poly}(n, \mathrm{size}(C))$. Let $\ell \triangleq n^5 + 2m$.

- C' parses its input as two components: a distribution $\mathbf{D} \subseteq \{0,1\}^n$ of size n^4 encoded as a string of length n^5, and two strings $\tau_1, \tau_2 \in \{0,1\}^m$. It simulates the reduction $\mathcal{R}$ on the instance $(C, \mathbf{D})$ of non-black-box Yao derandomization, which gives a LossyCode instance (C'', D''). It then outputs at most $n^5 + 2m$ bits.

  - If $D''(C''(\tau_j)) = \tau_j$ for $j \in [2]$, C' outputs $0 \circ \mathbf{D} \circ C''(\tau_1) \circ C''(\tau_2)$.
  - Otherwise, for some $j \in [2]$, $D''(C''(\tau_j)) \neq \tau_j$. In this case, we solve the LossyCode instance (C'', D''), and thus by running the reduction $\mathcal{R}$, C'' can find a string z for derandomizing Yao. It then search for an $i \in [n]$ and $\sigma_1, \sigma_2 \in \{0,1\}$ such that

    $$\Pr_{x \leftarrow \mathbf{D}}[C(x_{<i} \circ \sigma_1 \circ z_{>i}) \oplus \sigma_2 = x_i] \geq \frac{1}{2} + \frac{\varepsilon}{5n}.$$

    If none of valid (i, σ_1, σ_2) was found, C' outputs $1^{\ell-1}$. Otherwise, it works as follows. Let $\mathbf{D} = (d^1, d^2, \ldots, d^{n^4})$, then the vector $v \in \{0,1\}^{n^4}$ defined as $v_k \triangleq C(x_{<i} \circ \sigma_1 \circ z_{>i}) \oplus \sigma_2 \oplus d_i^k$ has Hamming weight at most $(1/2 - \varepsilon/(5n)) \cdot n^4$ and thus can be encoded as a string $\hat{v}$ of length $n^4 - \Omega(n^2)$ by Lemma 7.3. Therefore, we can encode D as a string $\hat{D}$ of length $n^5 - \Omega(n^2) + O(n)$ by removing $\{d_i^k\}_{k \in [n^4]}$ and adding $(z, i, \sigma_1, \sigma_2, \hat{v})$. We can pad $\hat{D}$ so that it has $n^5 - 2$ bits. The final output of C' will be $1 \circ \hat{D} \circ \tau_1 \circ \tau_2$.

- D' first simulates the reduction $\mathcal{R}$, generates C'', D'', and considers the following two cases.

  - If the first bit of its input is 0, D' parses its input as $0 \circ \mathbf{D} \circ z_1 \circ z_2$, where $D \subseteq \{0,1\}^{n^5}$, $z_1, z_2 \in \{0,1\}^{m-1}$. It then outputs $0 \circ \mathbf{D} \circ D''(z_1) \circ D''(z_2)$.
  - Otherwise, D' parses its input as $1 \circ \hat{D} \circ \tau_1 \circ \tau_2$, where $\hat{D}$ is a succinct encoding of a distribution $\mathbf{D}$ over $\{0,1\}^n$ of size n^4, as described above. If D' fails to parse its input as the form (i.e., its input is $1^{\ell-1}$), it outputs 0^ℓ. Otherwise, it recovers the distribution $\mathbf{D}$ and outputs $1 \circ \mathbf{D} \circ \tau_1 \circ \tau_2$.

Now it suffices to prove that given a solution $\pi = (\mathbf{D}, \tau_1, \tau_2) \in \{0,1\}^{n^5+2m}$ of the LossyCode instance (C', D'), we can solve the CAPP instance C in polynomial time. Note that since $D'(C'(\pi)) \neq \pi$, we know that $D''(C''(\tau_j)) \neq \tau_j$ for some $j \in [2]$, and there is no $i \in [n]$ and $\sigma_1, \sigma_2 \in \{0,1\}$ such that

$$\Pr_{x \leftarrow \mathbf{D}}[C(x_{<i} \circ \sigma_1 \circ z_{>i}) \oplus \sigma_2 = x_i] \geq \frac{1}{2} + \frac{\varepsilon}{5n}.$$

This implies that $\mathbf{D}$ must ε-fool C: If it is not the case, we know by the correctness of the reduction $\mathcal{R}$ that such i and σ_1, σ_2 must exist. Therefore, we can approximate the acceptance probability of C by $\mathbb{E}[C(\mathbf{D})]$ up to an additive error ε. $\square$

Combining Theorem 7.7 and Theorem C.8, we obtain the following characterization of **prBPP** = **prLOSSY** by certified derandomization and non-black-box derandomization of Yao's transformation.

Corollary C.9. *Let $\varepsilon \in (0,1)$ be any constant. The following statements are equivalent.*

- **prBPP** = **prLOSSY**.
- *There is a deterministic polynomial-time certified derandomization using $\mathcal{P}^{\mathsf{cc}}_{\varepsilon}$.*
- *There is a deterministic polynomial-time certified derandomization algorithm using range avoidance, i.e., using property $\mathcal{P}^g$ for some polynomial-time computable g.*
- *Non-black-box Yao's transformation is in* **FLOSSY**.

D Bounded Arithmetic and FLOSSY

In this section, we show that if a sentence of form $\forall x\ \exists y\ \varphi(x,y)$ can be proved in the bounded theory $\mathbf{APC}_1$ (defined in [Jeř04; Jeř07], also see [BKT14]), then the search problem defined by the relation φ is reducible to LossyCode. This is observed by Jeřábek [Jeř04], while the main idea has already appeared in earlier work of Wilkie (unpublished, see [Kra95]) and Thapen [Tha02].

We refer interested readers to standard textbooks [Kra95; Bus97; CN10] for the background of bounded arithmetic (e.g., the theories **PV** and $\mathbf{PV}_1$), and [Jeř04; Jeř07; BKT14] (and the references therein) for the definition of the theory $\mathbf{APC}_1$.

Theorem D.1 (Implicit [Jeř04, Proposition 1.14])**.** *Let $\phi(x,y)$ be a quantifier-free formula in the language of $\mathbf{APC}_1$ that only has x and y as its open variables. If $\mathbf{APC}_1 \vdash \forall x\ \exists y\ \phi(x,y)$, then the following search problem is in* **FLOSSY***: Given any $x \in \mathbb{N}$, output a $y \in \mathbb{N}$ such that $\phi(x,y)$ holds.*

Proof Sketch. Proposition 1.14 of [Jeř04] shown that if $\mathbf{APC}_1 \vdash \forall x\ \exists y\ \phi(x,y)$, then there are **PV** functions G, g, h and a constant k such that **PV** proves:

$$\forall x\ \forall b \geq 2^{|x|^k}\ \forall w < b^2\ \ (g(x,w,b) < b \wedge (G(g(x,w,b)) = w \vee \phi(x, h(x,w,b)))) \,.$$

By the soundness of **PV**, we know that there are polynomial-time algorithms G, g, h and a constant k such that for every $x \in \mathbb{N}$, $b \geq 2^{|x|^k}$, and $w < b^2$, we will have $g(x,w,b) < b$, and either $G(g(x,w,b)) = w$ or $\phi(x, h(x,w,b))$.

Our **FLOSSY** algorithm works as follows. Given any $x \in \mathbb{N}$ input, let b be the smallest power of two such that $b \geq 2^{|x|^k}$, it generates the LossyCode instance where $G : [b] \to [b^2]$ is the decompression circuit and $g(x,\cdot,b) : [b^2] \to [b]$ is the compression circuit. For every solution w of the LossyCode instance, we know that $G(g(x,w,b)) \neq w$, and therefore $\phi(x, h(x,w,b))$ is true. The algorithm then simply outputs $h(x,w,b)$. It runs in polynomial time as $|b| = |x|^k$ and G, g, h are all polynomial-time algorithms. □

We remark that Theorem D.1 can be used as a tool to design **FLOSSY** algorithms. There are several weak circuit lower bounds known to be provable in $\mathbf{APC}_1$ while it is unclear whether they are provable in $\mathbf{PV}_1$ (see, e.g., Table 1 of [PS21]). As an example, we use the $\mathbf{APC}_1$ formalization of the parity lower bound against $\mathbf{AC}^0$ due to Müller and Pich [MP20].

Theorem D.2 (Theorem 3.7 of [MP20], informal)**.** *For all constants $k, d \in \mathbb{N}$, there is a constant $n_0 \in \mathbb{N}$ such that $\mathbf{APC}_1$ proves the following statement: for all $n > n_0$ and every $\mathbf{AC}^0$ circuit $C_n : \{0,1\}^n \to \{0,1\}$ of depth d and size n^k, there is a $y \in \{0,1\}^n$ such that $C_n(y) \neq \sum_i y_i \bmod 2$.*

By combining Theorem D.2 and Theorem D.1, we conclude that the following search problem (usually called the *refutation problem* of the lower bound, see, e.g., [CJS+21]) is in **FLOSSY**. Note that the best known deterministic algorithm runs in quasi-polynomial time (see, e.g., [Bra10]).

Corollary D.3. *Let $k, d \in \mathbb{N}$ be any constants. The following search problem is in* **FLOSSY***: Given an* $\mathbf{AC}^0$ *circuit $C_n : \{0,1\}^n \to \{0,1\}$ of size n^k and depth d, output a string $y \in \{0,1\}^n$ such that $C_n(y) \neq \sum_i y_i \bmod 2$.*

Proof Sketch. The $\mathbf{AC}^0$ lower bound proved in Theorem D.2 is formalized in the form $\forall x\, \exists y\, \phi(x, y)$ for a quantifier-free $\phi(x, y)$, where x consists of n and (the description of) an $\mathbf{AC}^0$ circuit $C_n : \{0,1\}^n \to \{0,1\}$, and y is supposed to be an input on which C_n fails to compute parity.[52] Thus, by Theorem D.1, the corresponding search problem is in **FLOSSY**. □

E A Targeted Generator with Derandomized Reconstruction

In this section we prove Theorem 5.6. At a high-level, our construction follows the original strategy of Chen and Tell [CT21b], with two main differences explained next. (We assume that the reader has basic familiarity with this construction; for explanations and expositions, see [CT21a, Section 2.2], or the follow-up works [CRT22; CTW23; CLO+23; DPT24].)

- We show that if the hard function is in logspace-uniform $\mathbf{NC}^1$, then the generator can be computed in logspace. This is *not* immediate from any known version of the generator. In particular, the original construction uses a preprocessing transformation of the circuit by Goldreich [Gol18], which incurs a significant overhead: Even for functions computable by (logspace-uniform) constant-depth circuits, this transformation yields circuits of depth $\Omega(\log^2 n)$, which do not seem to be evaluable in logspace.

  Our key observation is that almost all of the layers added by this transformation are very simple (i.e., they consist of repeatedly powering the transition matrix of the logspace machine that outputs the hard function). Hence, we can compute the value of *each gate* in these levels in logspace (i.e., without evaluating the entire $\log^2 n$ depth circuit directly), and this allows us to compute each output of the targeted generator in logspace.

- Our reconstruction argument takes in a family of candidate next-bit-predictors, rather than a single distinguisher, and uses only $\mathrm{polylog}(n)$ random coins in total. To do so, we must show that each of the multiple steps in [CT21a] in which randomness is used to decode a code, or weed a list of possible candidates, can be done using few random bits.

  We do so by designing a modified reconstruction procedure. Following the idea of [PRZ23], who used samplers to obtain a randomness-efficient reconstruction procedure. As a consequence of this change, our reconstruction is not implemented as a small-depth circuit (i.e., unlike [CTW23; DPT24]).

The rest of this section is organized as follows. First, in Appendix E.1, we present a construction of bootstrapping systems for logspace-uniform $\mathbf{NC}^1$ circuits. Then, in Appendix E.2, we present a construction of a targeted generator with randomness-efficient reconstruction from bootstrapping systems. Finally, in Appendix E.3 we combine these two to prove Theorem 5.6.

[52] We note that since bounded theories only define efficiently computable functions, one needs to be careful about formalizing sentences in bounded theories. Nevertheless, one can verify that the formalization in [MP20] is compatible with Theorem D.1.

Throughout the proofs in this section, we will use standard technical tools, such as local list-decoders for the Hadamard code and for the Reed-Muller code. We defer the (standard) statements of these tools to Appendix E.4, for readability.

E.1 Bootstrapping Systems Computable in Logspace for Logspace-Uniform NC^1

We first define query-aided worst-case to rare-case reducibility. In contrast to the notion in prior works, we no longer assume that worst-case to rare-case reduction algorithm is given uniform samples. Instead, we think of the algorithm as having a preprocessing phase where it makes few queries to f, and then outputs a circuit that is correct on all inputs whp.

Definition E.1. For $\rho, \varepsilon : \mathbb{N} \to (0,1)$ and $T : \mathbb{N} \to \mathbb{N}$ and sequence of alphabets $\Sigma_1, \ldots$ and output lengths $w_1, \ldots$, we say that a function $f : (\Sigma_n)^n \to (\Sigma_n)^{w_n}$ is query-aided worst-case to ρ-rare-case reducible by logspace uniform circuits of size T, randomness complexity r, and query size s if there is a randomized algorithm M that works as follows.

- M is given oracle access to f_n and $\widetilde{f_n} \in (\Sigma_n)^n$ with agreement ρ with f_n, where agreement denotes relative Hamming distance.
- M runs in time T, uses r random coins, and makes at most s non-adaptive queries to f.
- M outputs a deterministic circuit C such that with probability at least $1 - \varepsilon$, $C(x) = f_n(x)$ for every $x \in (\Sigma_n)^n$.

Consider a function computable by a logspace-uniform family of $\mathbf{NC}^1$ circuits $\{C_n\}$. We will encode the computation of C_n on any fixed input x as a sequence of polynomials. As in [Gol18; CT21a], the first step will be to transform the family $\{C_n\}$ into a circuit family $\{C'_n\}$ that has even better uniformity properties, at the cost of increasing the depth (the transformation is identical to [Gol18; CT21a]). We underline the new properties that we prove for clarity.

Proposition E.2 (polynomial decomposition)**.** *There exist two universal constants $c, c' \in \mathbb{N}$ such that the following holds. Let $\{C_n\}$ be a logspace uniform family of $\mathbf{NC}^1$ circuits of size $T(n) \geq n$ and depth $\log(n) \leq d(n) \leq O(\log T)$, and let $\gamma \in (0,1)$ be a constant. Then there exists a logspace-uniform family of circuits $\{C'_n\}$ of size $T' = O(T(n)^c)$ and depth $d' = O(\log^2(T))$ that computes the same function as $\{C_n\}$, such that for every $x \in \{0,1\}^n$ there is a polynomial decomposition satisfying:*

1. ***Arithmetic setting.*** *The polynomials are defined over $\mathbb{F} \triangleq \mathbb{F}_p$, where p is the smallest prime in the interval $[T^{\gamma c}, T^{2\gamma c}]$. Set $H = [h] \subseteq \mathbb{F}$, there h is the smallest power of 2 of size at least $T^{\gamma/6}$, and let m be the minimal integer such that $h^m \geq 2T^c$.*
2. ***Faithful representation.*** *For every $i \in [d']$ and $\vec{w} \in H^m$ representing a gate in the ith layer, it holds that $\hat{\alpha}_i(\vec{w})$ is the value of the gate $\vec{w}$ in $C'_n(x)$.*
3. ***Layer logspace computability.*** *There is a space $O(\log T)$ algorithm that, given $x \in \{0,1\}^n$ and indices i, j of polynomials, computes $\hat{\alpha}_{i,j}$ (also, given index 0 it computes $\hat{\alpha}_0$).*
4. ***Base Layer Computability.*** *There is a time $\max\{n, t\} \cdot t$ algorithm that computes $\hat{\alpha}_0$.*
5. ***Layer DSR.*** *There is a time $h^{c'}$ algorithm that computes $\hat{\alpha}_{i,0}$ while querying $\hat{\alpha}_{i-1}$ twice on inputs in H^m. Lastly, $\hat{\alpha}_{i,2m} = \hat{\alpha}_i$*

6. ***Sumcheck DSR.*** *There is a time $h^{c'}$ and algorithm that gets input $\vec{w} \in \mathbb{F}^m$ and $(\sigma_1, \ldots, \sigma_{2m}) \in \mathbb{F}^{2m}$ and $j \in [2m]$ and oracle access to $\hat{\alpha}_{i,j-1}$ and outputs $\hat{\alpha}_{i,j}(\vec{w}, \sigma_1, \ldots, \sigma_{2m-j})$.*

7. ***Query-aided worst-case to rare-case reducibility.*** *For each $i \in [d']$ and $j \in [2m]$, the boolean function representing $\hat{\alpha}_{i,j}$ is query-aided worst-case to ρ-rare-case reducible with error h^{-100} in time $h^{c'}$ with $O(\log T)$ random coins, where $\rho = h^{-\alpha}\,\mathrm{polylog}(T)$. The same claim holds for $\hat{\alpha}_0$.*

Proof Sketch. Our construction of the polynomial decomposition is precisely identical to that in [CT21a, Proposition 4.7]. The only new properties that are claimed are Item (3) and Item (7), and we now explain why these are true.

Layer logspace computability. For Item (3), we use a more careful analysis of the transformation of $\{C_n\}$ to $\{C'_n\}$. Specifically, we strengthen [CT21a, Claim 4.7.1] to assert the following:

Claim E.3. *There exists a circuit C'_n as above of depth $d'(n) = O(d + \log^2 T) = O(\log^2 T)$ and size $T' = 2^{c\lceil \log T \rceil}$ for an integer $c > 1$ such that:*

- ***Adjacency Computability.*** *The layered adjacency relation function $\Phi' : [d'] \times \{0,1\}^{3\log T'} \to \{0,1\}$ of C'_n can be decided by a formula that can be constructed in time $\mathrm{polylog}(T)$ and space $O(\log T)$. Moreover, Φ' can be evaluated in space $O(\log T)$.*

- ***Logspace Evaluability.*** *There is an $O(\log T)$ space algorithm that on input $x \in \{0,1\}^n$ and $j \in [d']$, prints the values of the gates of $C'(x)$ at layer j.*

Proof. The first property is exactly the same as [CT21a, Claim 4.7.1], (and the fact that Φ' is logspace-computable is immediate from examining their proof). For the second property, recall from [Gol18] that the circuit C'_n has the following properties. It can be divided into $d_1 = O(\log^2 T)$ initial layers and $d_2 = O(\log(n) + d(n))$ final layers.

- In the first d_1 layers, the circuit C'_n constructs and repeatedly squares the $(T(n))^c \times (T(n))^c$ transition matrix M of the space $c\log(T)$ machine that, on input $\langle n \rangle \langle i \rangle$, outputs the ith gate of C_n. This squaring is performed $c \log T(n)$ times, where each square is computed by a depth $O(\log T)$ $\mathbf{NC}^1$-circuit that is extremely uniform.

- In the final d_2 layers, the circuit computes the map $(\langle C_n \rangle, x) \mapsto C_n(x)$ via the universal $\mathbf{NC}_1$ circuit, which has depth $O(d(n))$.[53]

On input (x, j) where $j \in [d']$, our logspace evaluation algorithm works as follows.

- If $j \leq d_1$, we first determine the number of repeated squaring steps have occurred prior to layer j. Supposing this number was r, the $O(\log T)$ layers that contain j correspond to computing the map $M^{2^r} \to M^{2^{r+1}}$. To compute M^{2^r}, for every initial state σ we run the machine from σ for 2^r steps and obtain what state τ it reaches, and then determine $\left(M^{2^r}\right)_{\sigma,\tau} = 1$. Since the machine has $\mathrm{poly}(T)$ configurations, we can perform this in logspace. We then determine the values in the next $O(\log T)$ layers of C'_n via the DFS simulation of $\mathbf{NC}^1$, where each time the simulation reaches the layer holding M^{2^r}, we generate the answer as described above.

[53] Here we assume that the logspace-uniform circuit for C_n outputs a representation in the form of a width-5 branching program [Bar89], which is without loss of generality since the reduction of [Bar89] can be computed in logspace. In this case, it is clear that such a universal circuit exists and has depth $O(\log n)$.

- If $j > d_1$ corresponds to the k-th layer of the universal $\mathbf{NC}^1$ circuit $\texttt{Eval}$ on $(\langle C_n \rangle, x)$, we use the fact that we can construct the input $\langle C_n \rangle$ using the logspace machine (i.e., the machine that prints C_n). Then we use the standard DFS-style simulation of $\mathbf{NC}^1$ to compute the value in layer j. □

Relying on Claim E.3, we now argue that the extension of the circuit-structure function of $\{C'_n\}$ is evaluable in logspace. Specifically, we strengthen [CT21a, Claim 4.7.2] as follows. (A similar strengthening appeared in [CRT22].)

Claim E.4. *For $i \in [d']$ there exists $\hat{\Phi}_i : \mathbb{F}^3 \to \mathbb{F}$ that satisfies the following:*

- *For every $(\vec{w}, \vec{u}, \vec{v}) \in H^{3m}$ we have that $\hat{\Phi}(\vec{w}, \vec{u}, \vec{v}) = 1$ iff the gate $\vec{w}$ in the i-th layer of C'_n is fed by gates $\vec{u}, \vec{v}$ in the $(i-1)$-th layer.*
- *The degree of $\hat{\Phi}_i$ is at most $h \cdot \mathrm{polylog}(T)$.*
- *For a universal constant $c_1 > 1$, there is a time h^{c_1} algorithm that computes $\hat{\Phi}_i$.*
- ***Logspace computability.*** *There is a space $O(\log T)$ algorithm that computes $\hat{\Phi}_i$.*

Proof. The construction is identical to [CT21a, Claim 4.7.2]. The only new property follows immediately from the map $\pi_j : H \to \{0,1\}$ that projects a to the j-th bit in the binary expansion of a being computable in logspace, and then using Claim E.3. □

We now prove that the polynomials $\hat{\alpha}_{i,j}$ and $\hat{\alpha}_i$ and $\hat{\alpha}_0$ are computable in logspace. We will do so separately for each case:

- Recall that $\alpha_0 : H^m \to \{0,1\}$ represents the string $x0^{h^m - n}$, and to compute $\hat{\alpha}_0(\vec{w})$, it suffices to evaluate
$$\hat{\alpha}_0(\vec{w}) = \sum_{\vec{0} \in H^{m'} \times \{0\}^{n'-m'}} \delta_{\vec{0}}(\vec{w}) \hat{\alpha}_0(\vec{0})$$
and then the result follows from standard results on arithmetic in logspace.

- For $i \geq 1$, recall that to compute $\hat{\alpha}_i(\vec{w})$ it suffices to compute
$$\hat{\alpha}_i(\vec{w}) = \sum_{\vec{v},\vec{u}} \hat{\Phi}(\vec{w}, \vec{u}, \vec{v})(1 - \hat{\alpha}_{i-1}(\vec{u})\hat{\alpha}_{i-1}(\vec{v}))$$
We can compute $\hat{\Phi}(\vec{w}, \vec{u}, \vec{v})$ in space $O(\log T)$ by Claim E.4. By the faithful representation property, we have that $\hat{\alpha}_{i-1}(\vec{u})$ is the value of gate $\vec{u}$ in layer $i-1$, which we can compute in space $O(\log T)$ by Claim E.3. Then the result follows from standard results on arithmetic in logspace.

- Finally, for (i,j), recall that to compute $\hat{\alpha}_{i,j}(\vec{w}, \sigma_1, \ldots, \sigma_{2m-j})$ it suffices to compute (letting $\vec{\sigma} = \sigma_1, \ldots, \sigma_{2m-j}$):
$$\begin{aligned}\hat{\alpha}_{i,j}(\vec{w}, \sigma_1, \ldots, \sigma_{2m-j}) &= \sum_{\beta_1,\ldots,\beta_j \in H^j} \hat{\alpha}_{i,0}(\vec{w}, \vec{\sigma} \circ \vec{\beta}) \\ &= \sum_{\beta_1,\ldots,\beta_j \in H^j} \hat{\alpha}_{i-1}(\vec{w}, \vec{\sigma} \circ \vec{\beta})\end{aligned}$$
Using the fact that we can compute $\hat{\alpha}_{i-1}$ in space $O(\log T)$, we can thus compute this value in space $O(j \log H + \log T) = O(m \log H + \log T) = O(\log T)$.

Query-aided worst-case to rare-case reducibility. The proof of this result is exactly analogous to the sample-aided property in [CT21a], except that we appeal to the decoder of Proposition E.13. Note that the error probability is now $|\mathbb{F}|^{-100} \leq h^{-100}$. □

Given the polynomial decomposition, we now construct bootstrapping systems for logspace-uniform $\mathbf{NC}^1$ circuits with randomness-efficient reconstruction, as follows.

Proposition E.5 (Bootstrapping system for uniform $\mathbf{NC}^1$ circuits)**.** *There exist two constants $\alpha \in (0,1)$ and $k > 1$ such that the following holds. Let $f : \{0,1\}^* \to \{0,1\}^*$ be a length-preserving function computable by logspace-uniform $\mathbf{NC}^1$ circuits $\{C_n\}$ of size $T(n) \geq n$ and depth $d = d(n)$ with $\log(n) \leq d \leq O(\log T)$. Then there exists $d' = O(\log^2 T)$ and $T' \leq T^k$ such that for every $\mu \in (0,1)$ and $t = T^\mu$, the function has $(d' \times T')$-bootstrapping systems with the following properties:*

1. *There is a logspace algorithm that, given $x \in \{0,1\}^n$ and $i \in [d']$, prints $P_i(x)$.*
2. *There is a time $\max\{n,t\} \cdot n$ algorithm that computes $P_0(x)$.*
3. *The downward self-reducibility algorithm runs in time t.*
4. *The query-aided worst-case to ρ-rare-case algorithm supports agreement $\rho = t^{-\alpha}\,\mathrm{polylog}(t)$, has error $t^{-100\alpha}$, and is computable by a time t algorithm.*

Proof. Let C_n be the logspace-uniform circuit for f on inputs of length n of depth $d = d(n) = O(\log T)$ and of size $T = T(n)$. Let c, c' be the universal constants from Proposition E.2, let $\gamma = 5\mu/c'$, and let C'_n be the corresponding logspace-uniform circuit with

$$\text{depth } d'_0 = O(d\log(T) + \log^2 T) = O(\log^2 T), \qquad \text{and size } T' = O(T(n)^c) \leq T^k \text{ for any } k > c.$$

Note that $|\mathbb{F}| \leq \mathrm{poly}(T)$ and $h = O(T^{5\mu/6c'})$ and $m = O(1/\mu)$. Denote $t = h^{c'} \leq T^\mu$. Next, let

$$\{\hat{\alpha}_i : \mathbb{F}^m \to \mathbb{F}\}_{i \in [d']}, \qquad \{\hat{\alpha}_{i,j} : \mathbb{F}^{3m-j} \to \mathbb{F}\}_{i \in [d'], j \in \{0..2m-1\}}$$

be the corresponding layer polynomials and sumcheck polynomials in the decomposition. We now view each $\hat{\alpha}_i$ and $\hat{\alpha}_{i,j}$ as a Boolean function from $3m\log(|\mathbb{F}|) = O(\log T)$ bits to $\mathbb{F} = O(\log T)$ bits (where we pad all functions to have the same domain).

Defining the bootstrapping system. The bootstrapping system has $d' = d'_0(2m+1) = O(d'_0/\mu)$ layers, each of length $|\mathbb{F}|^{3m} = \mathrm{poly}(T)$ and over alphabet $\mathbb{F}$. The base layer P_0 is the truth table of $\hat{\alpha}_0$, and hence there is an algorithm that computes it in the claimed space bound by Item 3. Then for $(i,j) \in [d'_0] \times \{0,\ldots,2m\}$, the (i,j)th layer is the truth table of the function $\hat{\alpha}_{i,j}$ (where we order first in increasing order of i, then increasing order of j).

By Item 3 of Proposition E.2, there is an algorithm that prints each non-base layer in space $O(\log T)$ by Item 3, and thus Item 1 of the current claim follows. Furthermore, the query-aided worst-case reduction follows from Item 7 of Proposition E.2. All other properties follow without modification from [CT21a]. □

E.2 From Bootstrapping to a Targeted HSG

We can now transform the bootstrapping system into a targeted somewhere-PRG. We again closely follow the transformation of [CT21a, Proposition 4.4], except that the generator is now computable in logspace, and the reconstruction algorithm is now given a set of predictors and uses only $\mathrm{polylog}(n)$ random coins.

Proposition E.6. *There exist universal constants $c', c'' > 1$ such that the following hold.*
Assumption. *For $d', t, T', A : \mathbb{N} \to \mathbb{N}$ and $\alpha \in (0,1)$ such that $d' \leq n$ and $\max\{n, t, A, d'\} \leq T'$, let f be a length-preserving function that has $(d' \times T')$-bootstrapping systems with alphabet size A satisfying the following:*

1. *There is an $O(\log T')$ space algorithm that, given $x \in \{0,1\}^n$ and $i \in [d']$, prints $P_i(x)$.*
2. *There is a time $\max\{n,t\} \cdot t$ algorithm to print $P_0(x)$.*
3. *The downward self-reduction runs in time t.*
4. *The query-aided worst-case to rare-case algorithm supports agreement t^α, has error $t^{-100\alpha}$, and is computable in time t.*

Conclusion. *Then, for every $M : \mathbb{N} \to \mathbb{N}$ and $\gamma \in (0,1)$ such that $\log(T') \leq M \leq \min\{T'^{\gamma/c''}, t^{\alpha/c''}\}$ there exists an algorithm G_f and an algorithm R that together obey the following for every $x \in \{0,1\}^n$:*

1. *The algorithm $G_f(x)$ runs in space $O(\log T')$ and computes $(L_1, \ldots, L_{d'})$, where for every $i \in [d']$, L_i is a list of M bit strings.*
2. *The algorithm R runs in time $(d' \cdot t \cdot T'^{\gamma} \cdot M \cdot n)^{c'}$ and uses $O(d' \cdot \log(T'))$ random coins.*
3. *When R is given x and oracle access to $\mathcal{P} = (P_1, \ldots, P_{M^3}) : \{0,1\}^{\leq M} \to \{0,1\}$ such that for every $i \in [d']$, there is some $j \in [M^3]$ such that*

$$\mathrm{adv}_{G_{f,i}(x)(\mathbf{U})}(P_j) \geq 1/M^2,$$

then R outputs $f(x)$ with probability at least $1 - 3d'/M^{-100}$.

The rest of this section will be devoted to the proof of Proposition E.6. For convenience, throughout the section we denote the size of the bootstrapping system as $(d \times T)$ (instead of $(d' \times T')$).

The Generator G_f. Given $x \in \{0,1\}^n$, the algorithm enumerates over $i \in [d]$. Fixing i, note that we can compute $P_i(x)$ in space $O(\log T)$ via Item 1. Thinking of $P_i(x) \in [A]^T$ as a truth table of a function $p_i : \{0,1\}^{\log T} \to [A]$, note that we can compute $h_i = \mathsf{Had}(p_i)$ in space $O(\log T)$, where Had is the encoding of Proposition E.16. Finally, the algorithm outputs

$$L_i = \mathsf{NW}^{h_i}(\mathbf{U})$$

where NW is the generator of Theorem E.19 with parameters $(\ell = O(\log T), M, \gamma)$, answering queries to h_i via composition of space-bounded algorithms. The fact that the overall generator is computable in space $O(\log T)$ is then direct.

The reconstruction argument. Fix $x \in \{0,1\}^n$ and

$$\mathcal{P} = (P_1, \ldots, P_{M^3}) : \{0,1\}^{\leq M} \to \{0,1\}$$

such that for every $i \in [d']$ there is $j \in [M^3]$ such that $\mathrm{adv}_{\mathsf{NW}^{h_i}(\mathbf{U})}(P_j) \geq 1/M^2$. The algorithm R gets input $x \in \{0,1\}^n$ and for $i = 0, \ldots, d$, finds a small circuit that computes a function with truth table P_i. We now describe this iterative process.

First, the algorithm constructs a circuit C_0 that computes the function with truth table P_0. By assumption, this circuit is of size $nt + t^2$ and can be constructed in time $\mathrm{poly}(nt)$. Next, for $i \in [d]$, we find a small circuit that computes $p_i = P_i(x)$. Our proof closely follows that of [CT21a, Lemma 4.10], except that we show that each iteration requires only $O(\log T)$ random coins, by applying the randomness-efficient reconstruction of Theorem E.19 and Propositions E.13 and E.16.

Lemma E.7 (Iteratively Finding a Small Circuit)**.** *There exists a universal constant $c_0 > 1$ such that the following holds. Given the circuit C_{i-1} and oracle access to $\mathcal{P}$, we can compute with probability at least $1 - 3 \cdot M^{-100}$ an oracle circuit C_i that computes p_i given oracle access to D. This can be done in time $t^2 \cdot T^{2\gamma} \cdot (M \cdot \mathrm{size}(C_{i-1}))^{c_0}$ using $O(\log T)$ many random coins, and the circuit C_i is of size $t \cdot M^{c_0} \cdot T^{2\gamma}$.*

Proof. The proof follows that of [CT21a, Lemma 4.10], with the following structure:

Claim E.8 (The NW Reconstruction)**.** *Given the circuit C_{i-1} and oracle access to $\mathcal{P}$, we can compute with probability at least $1 - M^{-100}$ an oracle circuit $C_{i,1}$ such that $C_{i,1}^{\mathcal{P}}$ computes h_i correctly on at least a $1/2 + 1/M^3$ fraction on inputs. Moreover, this step can be performed in time $\mathrm{poly}(M, \mathrm{size}(C_{i-1})) \cdot T^{2\gamma} t$ using $O(\log T)$ random coins, and the circuit $C_{i,1}$ is of size $\mathrm{poly}(M) \cdot T^{2\gamma}$.*

Proof Sketch. The proof is identical to that of [CT21a], except that we use the reconstruction algorithm of Theorem E.19 that uses few random coins. □

Next, we decode the Hadamard code:

Claim E.9 (The GL Reconstruction)**.** *Given the circuits C_{i-1} and $C_{i,1}$ and oracle access to $\mathcal{P}$, we can compute with probability at least $1 - M^{-100}$ an oracle circuit $C_{i,2}$ such that $C_{i,2}^{C_{i,1}}$ computes p_i on at least $\mu_{GL} = \mathrm{poly}(1/M)$ of the inputs. Moreover, this step can be performed in time $t \cdot \mathrm{poly}(M, \mathrm{size}(C_{i-1}))$ using $O(\log T)$ random coins, and the circuit $C_{i,2}$ is of size $\mathrm{poly}(M)$.*

Proof Sketch. The proof mirrors that of [CT21a]. We instantiate the decoder of Proposition E.16, with $\varepsilon = 1/M^3$. Let $r = O(\log T)$ be the number of random coins used by GL, and let $\tau = \mathrm{poly}(1/M \log(T)) = \mathrm{poly}(1/M)$ be the value τ in Proposition E.16, and let $\mu_{GL} = \tau/4$. We instantiate the sampler of Theorem E.12 with

$$n = r \qquad \varepsilon = \tau/2, \qquad \delta = M^{-101}$$

and note that the seed length is $O(\log T)$ and the number of samples is $\mathrm{poly}(M/\delta) = \mathrm{poly}(M)$. Letting the samples be S_1, for every $s \in S_1$ we let C_s be the circuit of Proposition E.16 with random coins s. Note that with probability at least $1 - M^{-101}$, at least one of these circuits $C = C_s$ satisfies

$$\Pr_{x \in \{0,1\}^{\log T}}[C^{C_{i,1}}(x) = p_i(x)] \geq \tau/2.$$

Finally, we weed the list by instantiating Theorem E.12 with

$$n = \log T, \qquad \varepsilon = \tau/8, \qquad \delta = M^{-101}/|S_1|$$

Note that the seed is of length $O(\log T)$ and the number of points is $\mathrm{poly}(M)$. Then we estimate the agreement of each circuit C_s with p_i by computing the empirical accuracy on the queried points. With probability M^{-101}, all of these estimates will be correct to accuracy $\tau/8$, and so if we return the most accurate it will satisfy the desired property. Finally, the bounds on the size of the circuit and runtime follow from Proposition E.16. □

Finally, we consider the worst-case to rare-case step:

Claim E.10. *Given the circuits C_{i-1} and $C_{i,2}$ and oracle access to $\mathcal{P}$, we can compute with probability at least $1 - M^{-100}$ a circuit $C_{i,3}$ such that $C_{i,3}^{C_{i,2}}$ computes p_i. Moreover, this step can be performed in time $t^2 \operatorname{poly}(M, \operatorname{size}(C_{i-1}))$ with $O(\log T)$ random coins, and $C_{i,3}$ is of size $t \cdot \operatorname{poly}(M)$.*

Proof Sketch. Recall that $p_i : \{0,1\}^{\log T} \to [A]$ is sample-aided worst-case to ρ-rare-case reducible (with $|\mathbb{F}| = A$) with $\rho = h^{-\alpha} \operatorname{polylog}(T)$, and note that

$$\rho = h^{-\alpha} \operatorname{polylog}(T) \leq M^{-k} = \mu_{GL}$$

by our assumption on M, and finally we use that $A \leq T$. Then the proof is again identical to [CT21a], except that we the query-aided worst-case to rare-case reduction follows from Proposition E.13. □

Lemma E.7 follows by combining Claims E.8 to E.10 in a straightforward way. Each of the three steps articulated in the latter claims uses $O(\log T)$ coins, the final circuit that is produced is of size $t \cdot \operatorname{poly}(N) \cdot T^{2\gamma}$, and the runtime of the three steps combined is at most $\operatorname{poly}(M, t, \operatorname{size}(C_{i-1})) \cdot T^{2\gamma}$. □

Finally, the reconstruction applies Lemma E.7 for d' times, and thus its randomness complexity, running time, and success probability are as claimed.

E.3 Putting it all together

Finally, let us recall the result statement, and present the proof:

Theorem 5.6. *There exists a universal constant $c > 1$ such that the following holds. Let $f : \{0,1\}^n \to \{0,1\}^n$ be computable by logspace-uniform NC^1 circuits of depth $d(n) = O(\log T)$ and size $T(n) \leq 2^{d(n)}$. Let $\delta > 0$ and let $M(n)$ be such that $c \log T(n) \leq M(n) \leq T(n)^{\delta/c}$. Then there is a deterministic algorithm G_f and a probabilistic algorithm R that satisfy the following. For every $x \in \{0,1\}^n$:*

- ***Generator.*** *The generator G_f gets input x, runs in space $O(\log T)$, and outputs $d' = \operatorname{polylog}(T)$ lists of M-bit strings $L_1, \ldots, L_{d'}$. Moreover, each list can be computed in space $O(\log T)$.*
- ***Reconstruction.*** *When R gets input x and oracle access to $(P_1, \ldots, P_{M^3}) : \{0,1\}^{\leq M} \to \{0,1\}$ such that for every $i \in [d']$, there is $j \in [M^3]$ such that*

$$\operatorname{adv}_{L_i}(P_j) \geq \frac{1}{M^2}$$

then R outputs $f(x)$ with probability at least $1 - 1/M$. The procedure R runs in time $T^\delta \cdot n^c$ and uses at most $\operatorname{polylog}(T)$ random coins.

Proof. Let c', c'' be the universal constants from Proposition E.6, and let α, k be the universal constants from Proposition E.5. We instantiate the bootstrapping system for f from Proposition E.5 with $\gamma = \delta/5c'$. This yields a bootstrapping system of dimension

$$(d' \times T'), \qquad d' = O(\log^2 T) \text{ and } T' = T^k$$

where the algorithm that prints each $P_i(x)$ runs in space $O(\log T)$, the algorithms computing the two reductions (i.e., downward self-reducibility and worst-case to rare-case reducibility) are logspace-uniform circuits of size $t = T^{\delta/5c'}$, and there is a logspace uniform circuit of size $\max\{n, t\} \cdot t$ that computes $P_0(x)$.

We now plug this system into Proposition E.6 with parameters n, T', d', t and with

$$\gamma = \delta/(5kc')$$

and observe that we have $A(n) = O(\log T) \leq T$ as required. The constraint in its conclusion that

$$\log(T') \leq M \leq \min\{T^{k\gamma/c''}, T^{\alpha\delta/5c'c''}\}$$

is satisfied by our choice of a sufficiently large universal constant c.

Generator. By the conclusion of Proposition E.6, we have that the lists output by the generator are computable in space $O(\log T') = O(\log T)$.

Reconstruction. The reconstruction algorithm R runs in time

$$(tT^{k\gamma}Mnd')^{c'} = (T^{(\delta/5c')}T^{k(\delta/(5kc'))}Mnd')^{c'} \leq T^{\delta} \cdot n^{c}$$

for a sufficiently large constant c (so that $M^{c'} \leq T^{\delta}$), and the number of random coins used is $\text{polylog}(T') = \text{polylog}(T)$. The probability that R fails to compute $f(x)$ is at most $O(\log^2(T)/M^{100}) \leq 1/M$ as claimed. □

E.4 Technical Tools

The proofs in this section relied on randomness-efficient versions of standard technical results. We now recall these results, and for completeness include proofs of the randomness-efficient versions.

E.4.1 Samplers

We recall the standard definition of a sampler:

Definition E.11 (sampler)**.** A function $\mathsf{Samp}\colon \{0,1\}^m \times [t] \to \{0,1\}^n$ is an (ε, δ) (oblivious) sampler if for any $H \subseteq \{0,1\}^n$ it holds that

$$\Pr_{x \leftarrow \{0,1\}^m}\left[\left|\Pr_{i \leftarrow [t]}[\mathsf{Samp}(x,i) \in H] - \frac{|H|}{2^n}\right| \leq \varepsilon\right] \geq 1 - \delta.$$

The parameter ε is the *accuracy* parameter of the sampler, and δ is its *confidence* parameter. We recall the following strong sampler.

Theorem E.12 ([Gol11a; CL20])**.** *For every $n \in \mathbb{N}$ and $\varepsilon, \delta > 0$, there exists an explicit (ε, δ) sampler $\mathsf{Samp} : \{0,1\}^m \times [t] \to \{0,1\}^n$ where $t = \text{poly}(\log(1/\delta)/\varepsilon)$ and $m = n + O(\log(1/\varepsilon\delta))$. Moreover, given $x \in \{0,1\}^m$ and $y \in [t]$, $\mathsf{Samp}(x,y)$ is computable in space $O(m)$ and time $\text{poly}(m)$.*

E.4.2 Reed-Muller Decoding

We recall that Reed-Muller codes are list-decodable with the following parameters. Note that [CT21a] constructed a logspace-uniform probabilistic **NC** circuit for this task, whereas we compute the function (not with a low depth circuit) using polylog(n) coins.

Proposition E.13. *Let $q : \mathbb{N} \to \mathbb{N}$ be a function mapping integers to primes, let $\ell : \mathbb{N} \to \mathbb{N}$ be such that $n \geq \ell(n)\log(q(n))$, and let $d : \mathbb{N} \to \mathbb{N}$. Let $\{f_n\}_{n\in\mathbb{N}}$ be a sequence of functions such that f computes a polynomial $\mathbb{F}_n^{\ell(n)} \to \mathbb{F}$ of degree $d(n)$ where $|\mathbb{F}| = q(n)$. Then f is query-aided worst-case to ρ-rare-case reducible by a time $\mathrm{poly}(q, \ell)$ algorithm that uses $O(\ell \log q)$ random coins, makes $\mathrm{poly}(q)$ queries, and has error $1 - q^{-100}$, where $\rho = 10\sqrt{d/q}$.*

We prove this following the proof of [CT21a] (using the randomness reduction idea of [PRZ23]), again proving that all steps can be implemented randomness-efficiently. We first recall the unique decoder from agreement (.97), where use the the decoder of [PRZ23] to do the final error-reduction via majority step efficiently.

Theorem E.14 (Lemma 4.6 [PRZ23])**.** *For every prime field $\mathbb{F}$ and integer $\ell \geq 1$ there is an algorithm* RMUniqueDec *that gets as input a degree parameter $d \leq |\mathbb{F}|/2$ and a vector $x \in \mathbb{F}^\ell$, and gets oracle access to $f : \mathbb{F}^\ell \to \mathbb{F}$, and satisfies the following:*

1. *If f agrees with a degree-d polynomial $P : \mathbb{F}^\ell \to \mathbb{F}$ on a (.97) fraction of the inputs, then*

$$\Pr[\mathrm{RMUniqueDec}^f(x) = P(x)] \geq 1 - |\mathbb{F}|^{-100}. \tag{5}$$

2. *The algorithm runs in time $\mathrm{poly}(|\mathbb{F}|, \ell)$ and uses $O(\ell \log |\mathbb{F}|)$ random coins.*

Next, we recall the RM list decoder (with a modification to use fewer random coins):

Theorem E.15. *For every prime field $\mathbb{F}$ and integer $\ell \geq 1$, there is an algorithm* RMListDec *that gets as input a degree parameter $d \geq 1$, a pair $(x_0, y_0) \in \mathbb{F}^\ell \times \mathbb{F}$, a vector $x \in \mathbb{F}^\ell$, and oracle access to a function $f : \mathbb{F}^\ell \to \mathbb{F}$, and satisfies the following:*

1. *If f agrees with a degree-d polynomial $P : \mathbb{F}^\ell \to \mathbb{F}$ on $10\sqrt{d/|\mathbb{F}|}$ of inputs, then with probability at least $\delta = 1/\mathrm{poly}(|\mathbb{F}|)$ over the random pair (x_0, y_0) ,*

$$\Pr[\mathrm{RMListDec}^f((x_0, y_0), x) = P(x)] \geq 1 - |\mathbb{F}|^{-100} \text{ for every } x \in \mathbb{F}^\ell. \tag{6}$$

2. *The algorithm runs in time $\mathrm{poly}(|\mathbb{F}|, \ell)$ and uses $O(\ell \log |\mathbb{F}|)$ random coins.*

Proof. By inspection, the proof of Theorem B.15 [CT21a] uses $O(\ell \log |\mathbb{F}|)$ random coins except in two places. First, their algorithm uses randomness in a final step of using a unique decoder for the Reed-Muller code; we implement this step using the unique decoder of Theorem E.14 from [PRZ23]. Second, it uses a randomized list-decoding algorithm for Reed-Solomon codes (to obtain a decoder in **NC**). Since we do not aim to obtain an **NC** circuit and can tolerate runtime $\mathrm{poly}(|\mathbb{F}|)$, we can use the deterministic decoder of [Sud97]. □

We then combine the two decoders to prove the final result, in exactly the same fashion as [CT21a].

Proof of Proposition E.13. We construct a probabilistic oracle algorithm that gets input 1^n and oracle access to f_n and $\widetilde{f_n}$ with agreement $\rho(n)$ with f_n, uses $O(\ell \log |\mathbb{F}|)$ random coins, makes $\mathrm{poly}(1/\rho)$ queries to f_n, and with probability $1 - q^{-100}$ outputs a deterministic circuit $C : \mathbb{F}^\ell \to \mathbb{F}$ such that $C^{\widetilde{f}}$ computes f. The algorithm D works as follows:

1. We use the sampler of Theorem E.12 with

$$\varepsilon = 1/2\,\mathrm{poly}(|\mathbb{F}|), \qquad \delta = q^{-101}, \qquad n = O(\ell \log |\mathbb{F}|),$$

where $2\varepsilon = 1/\mathrm{poly}(|\mathbb{F}|)$ is the value δ in Theorem E.15, so the seed length is $O(\log q + \ell \log |\mathbb{F}|)$ and the number of queried points is $\mathrm{poly}(q)$. Letting the set of sampled points be S_1, for every $s \in S_1$ we let C_s be a circuit implementing the algorithm of of Theorem E.15, where we use the random string s to hardwire (x_0, y_0) and the internal coins of C.

2. By Theorem E.15 and Markov, at least a $\delta = 1/\mathrm{poly}(|\mathbb{F}|)$ fraction of these circuits C_s (over a random s) satisfy

$$\Pr_x \left[C_s^{\widetilde{f}}(x) = f(x) \right] \geq .99$$

we call such a circuit a good circuit. By our choice of sampler error, with probability at least $1 - q^{-101}$, there will be $s \in S_1$ such that C_s is good.

3. Next, we use a second application of Theorem E.12 with

$$\varepsilon = .01, \qquad \delta = q^{-101}/|S_1|, \qquad n = \ell \cdot \log |\mathbb{F}|,$$

so the seed length is $O(\log q + \ell \log |\mathbb{F}|)$ and the number of queried points is $\mathrm{poly}(q)$. Letting the set of sampled points be $S_2 \subseteq \mathbb{F}^\ell$, for every $s \in S_1$ we compute an estimate of the agreement of C_s with f, i.e.

$$a(C_s) \stackrel{\text{def}}{=} \Pr_{x \in S_2} \left[C_s^{\widetilde{f}}(x) = f(x) \right].$$

With probability at least $1 - q^{-101}$, all these empirical estimates are within .01 of the true value, and if this occurs the circuit with the largest empirical agreement has agreement at least .97 with f.

4. Finally, we take this best circuit $C = C_s$ and compose it with the decoder of Theorem E.14, which is a probabilistic circuit D_v using $|v| = O(\ell \log |\mathbb{F}|)$ random coins that computes f on each point with probability $1 - |\mathbb{F}|^{-100}$. For the third time, we instantiate Theorem E.12, now with

$$\varepsilon = 0.1, \qquad \delta = q^{-100}/|\mathbb{F}|^\ell = q^{-100-\ell}, \qquad n = |v|$$

so the seed length is $O(\log q + \ell \log |\mathbb{F}|)$ (and note that the number of sampled points is now $\mathrm{poly}(\ell \log |\mathbb{F}|)$, which we can tolerate as we do not use this sampler to query f). Letting the sampled points be S_3, our final algorithm D computes:

$$D(x) = \mathtt{MAJ}_{v \in S_3}\{D_v^C(x)\}.$$

With probability at least q^{-100}, for every $x \in \mathbb{F}^\ell$ the sampler will output a set of points where taking the average over the decoding circuits correctly computes $f(x)$, and hence we will have $D(x) = f(x)$. Note that the final algorithm runs in time $|S_3| \cdot \mathrm{size}(D)\,\mathrm{size}(C) = \mathrm{poly}(|\mathbb{F}|, \ell)$ as claimed.

Finally, observe that the number of queries to f is bounded by $\mathrm{poly}(q)$ as claimed, and the number of random coins used is $O(\ell \log |\mathbb{F}|)$. Furthermore, all queries to f are simply obtained by enumerating over sampler seeds, so the queries are non-adaptive. □

E.4.3 The Hadamard Code

We require the Hadamard code, where we use (essentially) the randomness-efficient reconstruction of [DPT24, Theorem 5.16].

Proposition E.16 ([GL89; PRZ23; DPT24])**.** *For every $a : \mathbb{N} \to \mathbb{N}$ satisfying $a(n) \leq n$ and $\varepsilon : \mathbb{N} \to (0, 1/2)$, there exists a function Had that maps $g : \{0,1\}^\ell \to \{0,1\}^{a(\ell)}$ to a function $\mathsf{Had}(g) : \{0,1\}^{\ell + a(\ell)} \to \{0,1\}$ such that the following holds.*

1. *For every $x \in \{0,1\}^\ell$ and $z \in \{0,1\}^{a(\ell)}$, we have $\mathsf{Had}(g)(x,z) = \langle g(x), z \rangle$.*
2. *There exists a logspace algorithm GL that gets input 1^ℓ, uses $O(\ell)$ random coins, and outputs a deterministic oracle circuit C of size $\mathrm{poly}(\ell/\varepsilon)$ such that for every $\widetilde{g}$ that agrees with $\mathsf{Had}(g)$ in $1/2 + \varepsilon$ of positions, the probability over $x \in \{0,1\}^\ell$ and the coins used to produce C that $C^{\widetilde{g}}(x) = g(x)$ is at least $\tau = \mathrm{poly}(\varepsilon/\ell)$.*

Proof. We can assume $\log(1/\varepsilon) \leq \ell$ as otherwise the decoding algorithm can output a circuit with a fixed random output and be correct with sufficient probability. We will use the decoder $\mathsf{Dec}_{\mathsf{GL}}$ of [DPT24, Theorem 5.16]. Specifically, for every fixed $x \in \{0,1\}^\ell$, we consider $g(x) \in \{0,1\}^k$ where $k = a(\ell)$, and consider the string $H(x) \in \{0,1\}^{2^k}$ such that $H(x)_z = \mathsf{Had}(g)(x,z)$. The string $H(x)$ is the encoding of $g(x)$ by the (Hadamard) code referred to in [DPT24, Theorem 5.16].

Now, fix an arbitrary $\widetilde{g}$ that has agreement $1/2 + \varepsilon$ with $\mathsf{Had}(g)$. We call $x \in \{0,1\}^\ell$ **good** if

$$\Pr_{p \in a(\ell)}[\widetilde{g}_{x,a} = \mathsf{Had}(g)_{x,a}] \geq 1/2 + \varepsilon^2$$

and note that at least a $\mathrm{poly}(\varepsilon)$ fraction of x are good. For an arbitrary good x, the decoder $\mathsf{Dec}_{\mathsf{GL}}$, instantiated with confidence $\delta = 1/2$ and accuracy ε^2, uses $O(k + \log(1/\delta)) = O(\ell)$ random coins, and outputs $L = O(k \cdot \log(1/\delta)/\varepsilon^2) = \mathrm{poly}(\ell/\varepsilon)$ oracle circuits $C_1, ..., C_L \colon [k] \to \{0,1\}$ such that with probability at least $1 - \delta = 1/2$ there is $i \in [L]$ for which $C_i^{\widetilde{g}}(z) = H(x)_z$ for all $z \in [k]$.

Note that the circuits $C_1, ..., C_L$ do not depend on x. Our algorithm GL runs $\mathsf{Dec}_{\mathsf{GL}}$ with parameters as above, chooses $i \in [L]$ at random, and outputs C_i.[54] With probability $\mathrm{poly}(\varepsilon)$ we have that x is good, and with probability $1/2L = \mathrm{poly}(\varepsilon/\ell)$ we chose the correct $i \in [L]$. □

E.4.4 The NW Generator

Let us recall the standard definition of combinatorial designs, and the fact that logspace-uniform designs exist with good parameters:

Definition E.17. A family of sets $S_1, \ldots, S_M \subseteq [d]$ is an (ℓ, ρ) design if each of the sets is of size ℓ, and any two distinct sets satisfy $|S_i \cap S_j| \leq \log \rho$. The computational complexity of the design is the complexity of the function that gets input $i \in [M]$ and outputs the set S_i.

Theorem E.18 ([CT21a], Lemma A.3)**.** *There is a universal constant $c \geq 1$ such that for any $\alpha \in (0,1)$ the following holds for sufficiently large ℓ. There is an algorithm that outputs an (ℓ, ρ) design $S_1, ..., S_M \subseteq [d]$, where $M \geq 2^{\alpha\ell/c}$ and $d \leq c\ell/\alpha$ and $\log(\rho) = \alpha\ell$, and the algorithm runs in space $O(\log M)$.*

The following version of the Nisan-Wigderson generator [NW94] uses samplers to increase the success probability of the reconstruction procedure in a randomness-efficient way.

[54] A minor technical issue is that C_i – as defined and analyzed in [DPT24] with respect to a fixed x – gets oracle access not to $\widetilde{g}$, but only to the part of $\widetilde{g}$ that refers to x (i.e., to the partial function $\widetilde{g}_x(z) = \widetilde{g}(x,z)$). The final circuit that GL outputs gets input x and oracle $\widetilde{g}$, and simulates C_i when the latter is given oracle access to $\widetilde{g}_x$.

Theorem E.19. *There exists a universal constant $c_{\mathsf{NW}} > 1$, an oracle machine NW, and a probabilistic oracle machine R that work as follows for every constant γ.*

1. ***Generator.*** *When given input $(1^\ell, 1^M, \gamma)$ such that $M \leq 2^{2(\gamma/c_{\mathsf{NW}})\ell}$ and oracle access to $h : \{0,1\}^\ell \to \{0,1\}$ the machine NW runs in space $O(\ell)$ and outputs a set of strings in $\{0,1\}^M$.*

2. ***Reconstruction.*** *When given input $(1^\ell, 1^M, \gamma)$ and oracle access to h and $\mathcal{P} = (P_1, \ldots, P_{M^3})$, where there is $i \in [M^3]$ such that*

$$\mathrm{adv}_{\mathsf{NW}^h(\mathbf{U})}(P_i) \geq \frac{1}{M^2},$$

the reconstruction R runs in time $(\ell M)^{c_{\mathsf{NW}}} 2^{\gamma\ell}$, uses $O(\ell)$ random coins, makes non-adaptive queries, and outputs a circuit C such that with probability at least $1 - M^{-100}$:

$$\Pr_{i \in \{0,1\}^\ell}[C^{\mathcal{P}}(i) = h(i)] \geq \frac{1}{2} + \frac{1}{M^3}.$$

Moreover, the circuit that R outputs is deterministic and makes just one oracle query.

Proof Sketch. The generator is exactly the standard NW generator (with a logspace-computable design), which we recall so that we have consistent terminology.

The generator. The machine constructs the design of Theorem E.18, with M sets $S_1, \ldots, S_M \subseteq [d]$ with size ℓ and pairwise intersection at most $\gamma\ell$, with $d = O(\ell/\gamma)$. Then for $z \in \{0,1\}^d$ we define

$$G(z) = (h(\mathsf{Des}(z,1)), \ldots, h(\mathsf{Des}(z,M)))$$

and the fact that the generator is computable in the claimed space bound follows from the explicitness of Theorem E.18.

The reconstruction. We define a trial reconstruction which uses $O(\ell)$ random bits and succeeds with probability at least $1/\operatorname{poly}(M)$.

1. Draw $j \in [M^3]$ and $z' \in \{0,1\}^{[d]\setminus S_i}$, where $P_j : \{0,1\}^i \to \{0,1\}$ predicts bit i. (We denote the randomness used as $r = (j, z')$.) We then query h on all points in $S_i \cap S_l$ for every $l \neq i$, which takes $M \cdot 2^{\alpha\ell}$ queries.

2. Create the circuit C_r that, on input $x \in \{0,1\}^\ell$, combines x with z' to form $z \in \{0,1\}^d$ (where x specifies the points in S_i and z' specifies the other points), and outputs

$$P_j(h(\mathsf{Des}(z,1), \ldots, h(\mathsf{Des}(z,i-1)).$$

By the standard NW reconstruction argument (and the fact that there is at least one good predictor), on at least a $\rho = 1/\operatorname{poly}(M)$ fraction of strings $r \in \{0,1\}^{O(\ell)}$, the above procedure produces a circuit that computes h on at least a $1/2 + 2/M^3$ fraction of inputs (which we call a good circuit). To perform this construction in a randomness-efficient fashion, we apply the sampler of Theorem E.12 with

$$n = O(\ell) \qquad \varepsilon = \rho/2 \qquad \delta = M^{-101}.$$

With these parameters, the seed is of length $O(\ell)$ and the number of queries is $\mathrm{poly}(M)$. Letting the queried points be S_1, the probability that for every $s \in S_1$, C_s is not a good circuit is at most M^{-101}.

We next probabilistically weed this list using a second sampler. We apply the sampler of Theorem E.12 with

$$n = \ell, \qquad \varepsilon = 1/2M^3, \qquad \delta = M^{-101}/|S_1|$$

where the seed is again $O(\ell)$. Letting the queried points be $S_2 \subseteq \{0,1\}^\ell$, we test the agreement of C_s for every $s \in S_1$ and return the circuit with largest estimated agreement. The probability that we fail to estimate the agreement of any circuit up to error $1/2M^3$ is at most $|S_1| \cdot M^{-101}/|S_1| = M^{-101}$, and if this failure does not occur (and a good circuit is present in the list) we succeed. The time is dominated by constructing the circuits C_s, of which there are $\mathrm{poly}(M)$ for a fixed polynomial, so the runtime is as claimed by taking c_{NW} sufficiently large. □

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il