

Range avoidance, Arthur-Merlin, and TFNP

Surendra Ghentiyala* Zeyong Li† Noah Stephens-Davidowitz‡

September 20, 2026

Abstract

Range avoidance (Avoid) is the computational problem in which the input is an expanding circuit $C : \{0, 1\}^n \rightarrow \{0, 1\}^{n+1}$ and the goal is to find a string $y \in \{0, 1\}^{n+1}$ that is *not* in the image of C . Avoid was introduced recently by Kleinberg, Korten, Mitropolsky, and Papadimitriou [ITCS 2021] as an example of a total search problem that appears not to live in TFNP but does live in the second level of the total function polynomial hierarchy. Since then, Avoid has found surprising applications throughout complexity theory, and in theoretical computer science more broadly.

Our main results are as follows.

1. We show that any decision problem that efficiently reduces to Avoid is in $\text{AM} \cap \text{coAM}$ (even for promise problems, and even if the reduction is randomized and makes many adaptive queries). This in particular shows that NP-hardness of Avoid would collapse the polynomial hierarchy, answering an open question that has arisen numerous times in the literature.
2. We show an efficient randomized reduction from Avoid to a problem in TFNP that succeeds with probability $1 - \varepsilon$ for any $\varepsilon \geq 1/\text{poly}(n)$ (under complexity-theoretic assumptions). This provides additional evidence that Avoid is unlikely to be NP-hard. And, it shows that, though Avoid itself is almost certainly not in TFNP, it is in some sense extremely close to lying in TFNP. The randomness in our reduction seems necessary, since Chen and Li [STOC 2024] showed (under cryptographic assumptions) that $\text{Avoid} \notin \text{SearchNP}$, while a deterministic reduction from Avoid to a TFNP problem *would* place Avoid in SearchNP.

The high-level idea behind these two results is a rather simple “search Arthur-Merlin-Arthur protocol for Avoid.” And, a key technical tool that we use in all of our results is a novel AM protocol for upper bounding the size of the image of a circuit. This latter protocol can be viewed as a sort of dual of the celebrated set-size lower bound protocol due to Goldwasser and Sipser [STOC 1986]. Both protocols seem likely to be of independent interest.

*Cornell University. Email: sg974@cornell.edu. This work was supported by NSF Grant No. CCF-2312296, a Packard Foundation Fellowship, and a generous gift from Google.

†National University of Singapore. Email: li.zeyong@u.nus.edu. This work was supported by NRF grant NRF-NRFI09-0005.

‡Cornell University. Email: noahsd@gmail.com. This work was supported by NSF Grant No. CCF-2312296, a Packard Foundation Fellowship, and a generous gift from Google.

Contents

1	Introduction	1
1.1	Our results	2
1.2	Our techniques	5
1.3	Related work	12
1.4	Future directions	14
2	Preliminaries	15
2.1	Search problems	15
2.2	Avoid	16
2.3	Pseudorandom generators against $\text{NP}_{ }$ circuits	18
2.4	Checkability	19
2.5	Arthur-Merlin protocols	19
2.6	k -wise independent hashing	21
2.7	The Chernoff-Hoeffding bound	22
3	A range-size upper bound protocol	22
3.1	The Goldwasser-Sipser protocol	22
3.2	An approximation of the range size	24
3.3	An AM protocol for upper bounding range size	26
4	An “AM protocol for Avoid”	27
5	Any decision problem that reduces to Avoid is in $\text{AM} \cap \text{coAM}$	28
5.1	Proof of Proposition 5.1 by a tree of computation	30
6	Avoid is almost in TFNP	35
A	Additional proofs, deferred from Section 5.1	40

1 Introduction

Range avoidance (Avoid) is the computational problem in which the input is an expanding circuit $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$ for $m > n$, and the goal is to find a string $y \in \{0, 1\}^m$ that is *not* in the image of C . We consider the hardest case when $m = n + 1$ (except where we explicitly state otherwise).

Avoid was introduced relatively recently by Kleinberg, Korten, Mitropolsky, and Papadimitriou [KKMP21] as a beautiful example of a total search problem that appears not to be in TFNP but does live in the second level of the total function polynomial hierarchy (formally, $\text{TF}\Sigma_2^P$). Notice in particular that Avoid is in fact a total search problem, since for every expanding function there must be at least one element not in its image. Indeed, Avoid can be thought of as a computational problem that captures the “dual weak pigeonhole principle,” which states that if there are many more holes than pigeons, there must be an empty hole.

In fact, an instance of Avoid always has *many* solutions, in the sense that a random element $y \sim \{0, 1\}^{n+1}$ will be a solution with probability $1 - |\text{Im}(C)|/2^{n+1} \geq 1/2$. This makes Avoid a rather strange computational problem. On one hand, it is widely believed that $\text{Avoid} \notin \text{FP}$, i.e., that there is *not* a deterministic polynomial-time algorithm that solves Avoid. On the other hand, the trivial randomized algorithm that simply outputs a random string $y \sim \{0, 1\}^{n+1}$ solves Avoid with probability at least $1/2$!

Avoid grew in prominence due to Korten’s celebrated work showing that it is in a certain precise sense “the hardest computational problem that is solved by a random string with decent probability” [Kor21]. In particular, Korten showed reductions to range avoidance from the problems of constructing two-source extractors, truth tables with high circuit complexity, pseudorandom strings, rigid matrices, etc. All of these are objects that we do not know how to construct deterministically in polynomial time, but in all cases a suitable random distribution will yield such an object with good probability.

In the few years since [KKMP21] and [Kor21], Avoid has been the subject of intense study (e.g., [Kor22, RSW22, GLW22, ILW23, CHLR23, GGNS23, CGL⁺26, CL24, KP24, GLV24, LORS24, KS25a, Kor25, HV26, KPI25, HLZ26, KS25b, FIM25, RWZ26, GGLS26]), due mostly to its intimate connections with derandomization and circuit lower bounds. In particular, a series of recent exciting breakthroughs on circuit lower bounds by [CHR24, Li24, CLL25] followed from showing non-trivial algorithms for Avoid in exotic models of computation.

In fact, if we could actually prove that $\text{Avoid} \in \text{FP}$, then we would immediately have explicit constructions for many important mathematical and computational objects, *and* we would have proven that $\text{BPP} = \text{P}$, *and* we would have proven incredibly strong new circuit lower bounds! However, as we have already mentioned, Avoid is widely believed to *not* be in FP. Indeed, work of Ilango, Li, and Williams showed that $\text{Avoid} \notin \text{FP}$ under the assumption that $\text{NP} \neq \text{coNP}$ together with a very strong cryptographic assumption [ILW23]. And, later work of Chen and Li and Ren, Wang, and Zhong even showed that $\text{Avoid} \notin \text{SearchNP}$ under similarly strong assumptions from complexity theory and cryptography (even when the input circuit is quite restricted) [CL24, RWZ26]. However, we do not have any evidence for hardness of Avoid that simply works via reduction from a presumed hard problem. (In particular, all of the explicit construction problems that Korten reduced to Avoid are widely believed to be in FP, in spite of the lack of known algorithms.)

It would be particularly nice to reduce SAT to Avoid, thereby proving that Avoid is NP-hard and therefore that $\text{Avoid} \notin \text{FP}$ under the minimal assumption that $\text{P} \neq \text{NP}$. Indeed, the question of whether Avoid is NP-hard or whether NP-hardness of Avoid can be ruled out under standard complexity-theoretic assumptions has arisen repeatedly in the literature (e.g., in [RSW22, ILW23,

CGL⁺26, CL24]), including as Problem 3 in Korten’s survey on range avoidance [Kor25]. Prior work has merely observed that reductions from SAT to Avoid with very large stretch m (or reductions with relatively large stretch m that only make a small number of oracle calls) would imply that $\text{NP} = \text{RP}$ [ILW23, CGL⁺26]. (See Section 1.3.) But, prior work has not managed to rule out NP-hardness of Avoid when the stretch is just $m = n + 1$ and the reduction is unrestricted.

1.1 Our results

1.1.1 Range avoidance is probably not that hard

Here, we briefly present our two main results. In Section 1.1.3, we provide more discussion and interpretation.

Range avoidance is “kind of in $\text{AM} \cap \text{coAM}$.” Our first main contribution essentially settles the question of whether Avoid is NP-hard by showing that such hardness would collapse the polynomial hierarchy. This applies even for the hardest case when $m = n + 1$, and even under adaptive randomized reductions that make many queries to an oracle for Avoid. In fact, we show that any decision problem (or even promise problem) that reduces to Avoid must lie in the complexity class $\text{AM} \cap \text{coAM}$. (Below and throughout the paper, we write prC to represent the promise-problem version of a complexity class C .)

Theorem 1.1 (See Theorem 5.2). $\text{BPP}^{\text{Avoid}} \subseteq \text{AM} \cap \text{coAM}$ and $\text{prBPP}^{\text{Avoid}} \subseteq \text{prAM} \cap \text{prcoAM}$.

(Note that Theorem 1.1 places *decision* problems that reduce to Avoid into $\text{AM} \cap \text{coAM}$, but it does not seem to directly imply that Avoid itself lies in any particularly natural complexity class.)

As a consequence of Theorem 1.1, NP-hardness of Avoid (even under randomized reductions that make many adaptive queries) would collapse the polynomial hierarchy to AM [BHZ87], thus answering [Kor25, Problem 3]. More generally, one can think of Theorem 1.1 as effectively placing Avoid in a rather low level of the polynomial hierarchy. This is in contrast to the closely related problem Empty (which is sometimes called “strong Avoid”) that was also defined in [KKMP21], which is in fact NP-hard.

We actually prove the slightly stronger result that $\text{BPP}^{\text{RAvoid}} \subseteq \text{AM} \cap \text{coAM}$ (and similarly, $\text{prBPP}^{\text{RAvoid}} \subseteq \text{prAM} \cap \text{prcoAM}$). Here, $\text{BPP}^{\text{RAvoid}} \supseteq \text{BPP}^{\text{Avoid}}$ is the class of languages recognized by a BPP machine with access to a *random* oracle for range avoidance. See Theorem 5.2.

Range avoidance is *almost* in TFNP. Our second main contribution gives a more direct upper bound on Avoid by showing that it reduces via a (one-query) randomized reduction to a problem in TFNP, under the assumption that sufficiently strong PRGs exist.

What we mean by this is rather subtle, since in general randomized algorithms for search problems are rather subtle. (See, e.g., [ABK24] for recent discussion of this.) In this context, we mean that there is a single search problem $T \in \text{TFNP}$ such that for any $\varepsilon := \varepsilon(n) \geq 1/\text{poly}(n)$, there is a randomized reduction from Avoid to T that succeeds with probability at least $1 - \varepsilon$ (after making just one oracle query), as in the following theorem. Notice that this success probability is much higher than the probability of $1/2$ achieved by the trivial randomized algorithm.

Theorem 1.2 (Informal; see Theorem 6.1). *Assuming that sufficiently strong PRGs exist, there is a problem $T \in \text{TFNP}$ such that for every $\varepsilon := \varepsilon(n) \geq 1/\text{poly}(n)$, there is a (one-query) randomized reduction from Avoid to T that succeeds with probability at least $1 - \varepsilon$.*

In particular, under this assumption, any decision problem that reduces to Avoid is in BPP^{TFNP} .

The specific version of PRGs that we need for this result are PRGs that fool $\text{NP}_{\parallel}$ circuits¹ (of fixed polynomial size). Such PRGs are believed to exist, and they are even known to exist under the quite likely assumption that E does not have $2^{o(n)}$ -size non-deterministic circuits [SS24]. (See Sections 2.3 and 6 for more discussion.)

Theorem 1.2 can of course be interpreted as additional evidence that Avoid is unlikely to be NP-hard. Specifically, the theorem immediately implies that Avoid is not NP-hard unless $\text{SAT} \in \text{BPP}^{\text{TFNP}}$ (assuming the existence of sufficiently strong PRGs). See Section 6 for more discussion.

1.1.2 Two Arthur-Merlin protocols

A SearchAMA protocol for range avoidance. The high-level idea behind the proofs of both Theorems 1.1 and 1.2 is a simple SearchAMA protocol for Avoid. This protocol may be of independent interest, and we describe it in detail in Section 1.2.1. (Our definition of SearchAMA is rather delicate, but the protocol itself is quite simple. See Section 1.2.1 for a nearly complete description of the protocol and Section 2.5 for our definition of SearchAMA.)

Theorem 1.3 (Informal; see Theorem 4.1). $\text{Avoid} \in \text{SearchAMA}$.

A range-size upper bound protocol. Finally, a key technical tool that we use to prove Theorems 1.1, 1.2 and 1.3 is a novel AM protocol for upper bounding the image size of a circuit $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$, as in the following theorem.

Theorem 1.4 (A range-size upper bound protocol; see Theorem 3.2). *For any $\varepsilon := \varepsilon(n) \geq 1/\text{poly}(n)$, there is an AM protocol for the following promise problem. The input is a circuit $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$ and an estimate $\tau \in [1, 2^n]$ of the size of the image, $|\text{Im}(C)|$. It is a YES instance if $|\text{Im}(C)| \leq \tau$ and it is a NO instance if $|\text{Im}(C)| \geq \tau + \varepsilon 2^n$.*

We expect that Theorem 1.4 will be of independent interest and have additional applications. In particular, Theorem 1.4 is perhaps best viewed as complementing the celebrated Goldwasser-Sipser set-size lower bound protocol [GS86]. Recall that the set-size lower bound protocol allows Merlin to convince Arthur of a *lower* bound on the size of any set $S \subseteq \{0, 1\}^m$ that can be recognized in NP (or even AM; see Section 3.1) up to multiplicative error $1 + 1/\text{poly}(n)$. (Unsurprisingly, we use the set-size lower bound protocol from [GS86] as a key subroutine in our range-size upper bound protocol.)

In particular, notice that the Goldwasser-Sipser protocol allows Merlin to convince Arthur of a *lower* bound on the size of any set $S \subseteq \{0, 1\}^m$ that can be represented as the image of a polynomial-size circuit $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$ (i.e., $S = \text{Im}(C)$). Our range-size upper bound protocol allows Merlin to convince Arthur of an *upper* bound on the size of such a set $S \subseteq \{0, 1\}^m$, up to additive error $2^n/\text{poly}(n)$.

1.1.3 Some additional discussion

Perhaps the most obvious interpretation of Theorems 1.1 and 1.2 is simply as evidence that Avoid is unlikely to be NP-hard. Here, we list a number of additional interpretations and consequences of our results.

¹Circuits with parallel NP-gates.

Between two trivial algorithms for Avoid. We recall two basic facts about Avoid that were already observed when the problem was introduced in [KKMP21]. As we have already discussed, there is a trivial efficient randomized algorithm that solves Avoid with probability $1/2$. A closely related fact is that $\text{Avoid} \in \text{FZPP}^{\text{NP}}$, i.e., Avoid can be solved by a *zero-error* randomized algorithm with access to an NP oracle.²

Theorems 1.2 and 1.3 tell us that something interesting happens between the zero-error regime of the FZPP^{NP} algorithm and the $1/2$ error regime of the trivial algorithm. Specifically, Theorem 1.2 shows that an oracle for a TFNP problem (rather than an oracle for all of NP) is sufficient if one is willing to accept failure probability $1/\text{poly}(n)$ rather than zero error (assuming that sufficiently strong PRGs exist). And, Theorem 1.3 similarly shows that if one is willing to accept a small polynomial failure probability, then interaction with Merlin is sufficient.

Avoid is almost certainly not in TFNP, but it comes pretty close. Notice that Avoid is not in TFNP unless $P = \text{NP}$,³ and under strong cryptographic assumptions, Chen and Li and Ren, Wang, and Zhong showed that Avoid is not even in the (much larger) class SearchNP [CL24, RWZ26]. However, Theorem 1.2 says that Avoid in some sense is “surprisingly close to being contained in TFNP.” Specifically, it reduces to a problem in TFNP (under a reasonable complexity-theoretic assumption) via a randomized reduction. Indeed, it is easy to see that any problem that reduces to a problem in TFNP under *deterministic* reductions must lie in SearchNP . So, the randomness in our result is in fact necessary (under the cryptographic assumptions used in [CL24] or [RWZ26]).

Derandomization and algorithms for Avoid. Recall that a celebrated result of Korten (closely related to much earlier work of Jeřábek [Jeř04]) showed that E^{NP} does not have circuits of size $2^{o(n)}$ *if and only if* Avoid reduces to a problem in NP under deterministic Turing reductions (i.e., if and only if $\text{Avoid} \in \text{FP}^{\text{NP}}$) [Kor21]. Theorem 1.2 can be viewed as a variant of (one direction of) this result. Specifically, Theorem 1.2 shows that an even stronger circuit lower bound (the assumption that E requires exponential-size non-deterministic circuits) implies that Avoid actually reduces to a problem that is not only in NP but is actually in TFNP—albeit under randomized reductions.

Theorem 1.2 can also be viewed as a variant of a beautiful result by Korten and Santhanam [KS25a]. [KS25a] showed that if even stronger PRGs exist, then the *uniform* variant of Avoid with sufficient stretch is actually in FP. (In other words, for any $\text{poly}(n)$ -time computable family of expanding circuits $C_n : \{0, 1\}^n \rightarrow \{0, 1\}^m$, Avoid can be solved in deterministic polynomial time when restricted to this family. See Section 1.3.) We show how to solve a notably harder problem (Avoid with minimal stretch on arbitrary inputs) under a weaker assumption, but only via a randomized algorithm with access to an oracle for a problem in TFNP. (Our proof was also inspired by the ideas in [KS25a], as we discuss in Section 1.3.)

²Given access to an NP oracle, one can simply repeatedly sample a random string, use the NP oracle to check if it is in the image of the circuit, and output it if it is not.

³This is because the formal definition of TFNP requires that there is a verifier that determines whether any candidate solution is valid. Determining whether a given string is a solution to a given Avoid instance is a coNP -complete problem, so an efficient algorithm for this would imply that $P = \text{NP}$. Notice that under this definition, it is easy to find examples of search problems that are trivially in FP but also trivially not in TFNP or even FNP. (For example, on input x , output a Turing machine that halts on input x .) So, the fact that Avoid is not in TFNP unless $P = \text{NP}$ really tells us relatively little about whether Avoid is easy.

Separating Avoid and the Linear Ordering Principle. [Theorem 1.1](#) separates Avoid from LOP (the Linear Ordering Principle), which was recently defined by Korten and Pitassi [[KP24](#)] and is closely related to Avoid. Indeed, [[KP24](#), Problem 2] asked whether LOP can be reduced to Avoid under FP^{NP} reductions, as such a reduction would have very interesting complexity-theoretic consequences. Since LOP is known to be hard for MA and even P^{prMA} [[KP24](#), [HV26](#)], [Theorem 1.1](#) implies that LOP does not reduce to Avoid unless $\text{P}^{\text{prMA}} \subseteq \text{AM} \cap \text{coAM}$. (So, we rule out FP reductions, and even suitable randomized efficient reductions. We do not expect to rule out FP^{NP} reductions, since it is thought that $\text{LOP} \in \text{FP}^{\text{NP}}$.)

Generalization to a wider class of problems. We briefly note that [Theorems 1.1, 1.2](#) and [1.3](#) can be generalized slightly. Specifically, these results apply not only to Avoid but to any search problem such that (1) a random string is a solution with non-negligible probability (where the branching factor of the tree in the protocol in [Section 5](#) depends on this probability); (2) the problem of recognizing a solution is in coAM ; and (3) the probability that a random string is a non-solution can be upper bounded up to small error in AM (which we prove about Avoid in [Theorem 1.4](#)). For example, our results apply to the variant of Avoid in which the circuit takes as input an integer in $[N]$ (represented appropriately) and outputs an integer in $[M]$, provided that $M \geq N(1 + 1/\text{poly}(\log N))$,⁴ and they also apply to the two harder variants of Avoid defined in [[CGL+26](#)]. However, for simplicity, we simply work with Avoid. (Of course, [Theorems 1.1, 1.2](#) and [1.3](#) also apply immediately to all of the problems that are now known to reduce to Avoid, and particularly the explicit construction problems in [[Kor21](#)].)

1.2 Our techniques

1.2.1 “A SearchAMA protocol for Avoid”

As we mentioned above, the high-level idea behind our two main results is a simple “SearchAMA protocol for Avoid.” (See [Section 2.5](#) for the formal definition of SearchAMA.)

In the protocol, polynomially bounded Arthur and computationally unbounded Merlin take as input a circuit $C : \{0, 1\}^n \rightarrow \{0, 1\}^{n+1}$. Let us assume for simplicity that Arthur knows $\alpha := |\text{Im}(C)|/2^{n+1}$, i.e., Arthur knows the fraction $\alpha \leq 1/2$ of elements in $\{0, 1\}^{n+1}$ that are in the image of the circuit C . (In our actual protocol, we use [Theorem 1.4](#) to allow Merlin to convince Arthur of a good upper bound on α , which is sufficient.)

Arthur will sample $y_1, \dots, y_m \sim \{0, 1\}^{n+1}$ for some suitably large $m := m(n) = \text{poly}(n)$ and send them to Merlin. Notice that Arthur expects roughly αm of the y_i to be in the image of C and roughly $(1 - \alpha)m$ of them to lie outside the image. More precisely, the number of y_i in the image of C will be between, say, $\alpha m - \sqrt{nm}$ and $\alpha m + \sqrt{nm}$ with probability at least $1 - 2^{-\Omega(n)}$.

The idea is to have Merlin simply tell Arthur which of the y_i are in $\text{Im}(C)$ by directly providing preimages for these y_i . So, Merlin will respond to Arthur’s message with a list $x_1, \dots, x_m \in \{0, 1\}^n$ that is meant to be a complete list of preimages, in the sense that $C(x_i) = y_i$ for every i such that $y_i \in \text{Im}(C)$. (Of course, when $y_i \notin \text{Im}(C)$, we cannot hope to have $C(x_i) = y_i$.)

Upon receiving the purported preimages x_i , Arthur first computes the number $k := |\{i : C(x_i) = y_i\}|$ of actual preimages provided by Merlin. If $k < \alpha m - \sqrt{nm}$, then Arthur assumes that Merlin has not in fact provided a complete list of preimages, and Arthur simply aborts the

⁴This problem is known to be equivalent to Avoid under FP^{NP} reductions [[KKMP21](#)], but it is not known to be equivalent under efficient reductions without an NP oracle.

protocol by outputting the special symbol $\perp$. Otherwise, Arthur samples a uniformly random index $i \sim \{i' : C(x_{i'}) \neq y_{i'}\}$ and outputs y_i . In other words, Arthur outputs a uniformly random element y_i from the list of elements for which Merlin has *not* provided a preimage.

Notice that if the number of elements y_i in $\text{Im}(C)$ is between $\alpha m - \sqrt{nm}$ and $\alpha m + \sqrt{nm}$ then (1) Arthur will always output a correct answer if Merlin is honest and sends a complete list of preimages; and (2) no matter how Merlin behaves, Arthur will either output $\perp$, or Arthur will output a correct answer with probability

$$\frac{|\{i' : y_{i'} \notin \text{Im}(C)\}|}{|\{i' : y_{i'} \neq C(x_{i'})\}|} \geq \frac{(1 - \alpha)m - \sqrt{nm}}{(1 - \alpha)m + \sqrt{nm}} \geq 1 - 4\sqrt{n/m}.$$

So, honest Merlin causes Arthur to output a correct answer with very high probability (probability at least $1 - 2^{-\Omega(n)}$), and no matter what Merlin does, with probability at least $1 - 4\sqrt{n/m} - 2^{-\Omega(n)}$, Arthur will output either a solution or $\perp$. We can make the latter probability larger than $1 - \varepsilon$ for any $\varepsilon := \varepsilon(n) \geq 1/\text{poly}(n)$ by taking $m \gg n/\varepsilon^2$ to be a sufficiently large polynomial.

So, this protocol allows Arthur to “interact with Merlin to drastically lower the failure probability of the trivial algorithm for Avoid from $1/2$ to any $\varepsilon = 1/\text{poly}(n)$.” See [Section 4](#) for the formal statement. This simple protocol is a first hint that perhaps there is something interesting about range avoidance’s relationship with Arthur-Merlin protocols.

Before moving on, we wish to draw attention to one rather annoying technicality about the above protocol. Formally, it is an AMA protocol, and *not* an AM protocol. Recall that in an AM protocol, Arthur sends a uniformly random message to Merlin, Merlin responds, and then Arthur must decide what to output *deterministically based only on these two messages and the input*. In an AMA protocol, Arthur is allowed to flip additional random coins before deciding what to output. It is known that any AMA protocol *for a decision problem* can be efficiently converted generically into an AM protocol [[Bab85](#), [BM88](#)] (and, in fact, any constant-round protocol can be converted into a two-round AM protocol). But, for search problems we do not know how to convert an AMA protocol into an AM protocol (perhaps because no such conversion is possible).

1.2.2 Reducing Avoid to a problem in TFNP

To reduce Avoid to a problem in TFNP, our idea is to replace the random strings $y_1, \dots, y_m \in \{0, 1\}^{n+1}$ from the AM protocol described above by a fixed list of suitably *pseudorandom* strings. In particular, suppose that for every n and every circuit size s there exists some canonical list of strings $y_1, \dots, y_m \in \{0, 1\}^{n+1}$ that can be computed in time $\text{poly}(n, s)$ such that for every circuit $C : \{0, 1\}^n \rightarrow \{0, 1\}^{n+1}$ of size at most s , we have

$$\alpha m - \varepsilon m \leq |\{i : y_i \in \text{Im}(C)\}| \leq \alpha m + \varepsilon m$$

where $\alpha := |\text{Im}(C)|/2^{n+1}$ as above and, say, $\varepsilon \leq 1/n^{10}$. In other words, the number of elements y_i in the image of C is always close to what we expect.

Then, we can try to define a total search problem in which the input is a circuit $C : \{0, 1\}^n \rightarrow \{0, 1\}^{n+1}$, and the goal is to output x_i such that $C(x_i) = y_i$ for at least $\alpha m - \varepsilon m$ different choices of i . This is in fact a total search problem by the above assumption that the number of y_i in the image is at least $\alpha m - \varepsilon m$. And, just like in the protocol described above, if we choose a random element for which we have *not* received a preimage, it will be a solution to the Avoid instance with probability at least $1 - O(\varepsilon)$.

As described, however, this problem is probably *not* in TFNP even if we accept that such a suitable list of pseudorandom strings $y_1, \dots, y_m$ exists. The problem is that it is not clear how to verify that the number of preimages that we have received is in fact larger than $\alpha m - \varepsilon m$, since we do not know $\alpha := |\text{Im}(C)|/2^{n+1}$. For this, we modify the search problem slightly by also asking for the solution to include a proof π that α is actually small (specifically, small relative to the fraction of preimages provided).

It remains to show how to construct (1) a list $y_1, \dots, y_m \in \{0, 1\}^{n+1}$ of such pseudorandom strings; and (2) a proof π that the range size of a circuit is small. In fact, it is relatively easy to see that such a list of pseudorandom strings can be constructed if there exists a PRG that fools $\text{NP}_{\parallel}$ circuits. (See [Lemma 2.12](#).) Furthermore, the existence of such a PRG implies that $\text{AM} = \text{NP}$ [[KvM99](#)]. Therefore, if such a PRG exists, then the range-size upper bound AM protocol in [Theorem 1.4](#) can be converted into an NP proof π with the desired property. Indeed, such strong PRGs are known to exist under the reasonable assumption that E does not have $2^{o(n)}$ -size non-deterministic circuits [[SS24](#)]. See [Sections 2.3](#) and [6](#) for more detail.

Putting all of this together, we obtain a problem that *is* in TFNP (assuming such strong PRGs exist, which is in turn implied by circuit lower bounds against non-deterministic circuits) and a randomized reduction from Avoid to this problem. Notice that the reason that the reduction is randomized is more-or-less exactly the same as the reason that the protocol that we described in [Section 1.2.1](#) is an AMA protocol rather than an AM protocol.

1.2.3 An $\text{AM} \cap \text{coAM}$ protocol for decision problems that reduce to Avoid

We now describe our proof of [Theorem 1.1](#), which says that any decision problem that reduces to Avoid must be in $\text{AM} \cap \text{coAM}$.

So, suppose that L is some decision problem that reduces to Avoid. In other words, suppose that there exists some polynomial-time oracle algorithm $\mathcal{B}^{\mathcal{O}}$ such that if $\mathcal{O}$ solves Avoid then $\mathcal{B}^{\mathcal{O}}(z)$ accepts if and only if $z \in L$. (We are making some simplifying assumptions here. In particular, we are assuming here that $\mathcal{B}$ is deterministic, and we are assuming that L is a true decision problem, rather than a promise decision problem. In [Section 5](#), we handle randomized reductions and promise problems as well. We even handle the case when the reduction only works with a *random* Avoid oracle $\mathcal{O}$, rather than an arbitrary Avoid oracle.) We wish to show that L is in $\text{AM} \cap \text{coAM}$ by effectively showing how, given as input $z \in \{0, 1\}^*$, Arthur can use Merlin to simulate the behavior of $\mathcal{B}^{\mathcal{O}}$ on input z .

A simple protocol for one-query reductions. We first consider the very simple special case in which the number of oracle queries Q made by $\mathcal{B}^{\mathcal{O}}$ is always just $Q = 1$.

To construct an Arthur-Merlin protocol that simulates the behavior of $\mathcal{B}^{\mathcal{O}}$ on input z , let $C : \{0, 1\}^n \rightarrow \{0, 1\}^{n+1}$ be the one Avoid query made by $\mathcal{B}^{\mathcal{O}}$ on input z . Arthur samples $y \sim \{0, 1\}^{n+1}$, which we think of as a candidate solution to the Avoid instance C . Arthur then sends y to Merlin, and Merlin responds with $x \in \{0, 1\}^n$. If Merlin is honest and $y \in \text{Im}(C)$, Merlin will set x so that $C(x) = y$. (We do not care what value x takes when $y \notin \text{Im}(C)$.) Arthur can then check if $C(x) = y$. If $C(x) = y$, then evidently y was not a valid response to the oracle query C , and Arthur will therefore output the special symbol $?$. We think of $?$ as Arthur's way of saying "My attempt at simulating $\mathcal{B}^{\mathcal{O}}$ on input z failed, so I do not know whether $z \in L$." If $C(x) \neq y$, then Arthur

assumes that y is a valid solution to the oracle query C , and Arthur then simulates the behavior of $\mathcal{B}^\mathcal{O}$ when its input is z and the response to its oracle query is y . Arthur accepts if the simulation accepts and rejects otherwise.

Notice that when Merlin is honest, Arthur never outputs the wrong answer—he either correctly accepts, correctly rejects, or outputs $?$. And, when $y \notin \text{Im}(C)$, no matter how Merlin behaves, Arthur will output the correct answer. (Here, we are relying on the simple but crucial fact that if $y \notin \text{Im}(C)$, then Merlin simply cannot provide a preimage x for y .) So, Arthur outputs the correct answer with probability

$$\Pr[y \notin \text{Im}(C)] = 1 - |\text{Im}(C)|/2^{n+1} \geq 1/2,$$

whether Merlin is malicious or honest. Arthur and Merlin can then repeat this protocol many times in parallel. The result is many subprotocols outputting either accept, reject, or $?$ with the property that (1) with high probability at least one of the subprotocols correctly outputs accept or reject; and (2) if Merlin is honest then no subprotocol outputs the wrong answer (i.e., accepts when $z \notin L$ or rejects when $z \in L$). Arthur can therefore output $\perp$ (i.e., correctly declare that Merlin has cheated) if any of the subprotocols disagree (i.e., if at least one accepts and at least one rejects). Otherwise, Arthur can accept if at least one protocol accepts and reject otherwise. Such a protocol immediately implies that $L \in \text{AM} \cap \text{coAM}$.

A line-based protocol that works for small Q . This one-query protocol more-or-less immediately generalizes to any constant number Q of oracle calls, and even $Q = O(\log |z|)$. To see this, suppose that Arthur sends to Merlin Q candidate solutions $y_1, \dots, y_Q \sim \{0, 1\}^{n+1}$, one for each of the Q oracle queries $C_1, \dots, C_Q$ made by $\mathcal{B}^\mathcal{O}$ on input z . (Here, we are assuming without loss of generality that all of the oracle queries C_i are circuits with n input bits and $n + 1$ output bits for some fixed n .) Notice that C_i can depend on $y_1, \dots, y_{i-1}$ here. Regardless, the C_i are well defined given z and $y_1, \dots, y_Q$. If Merlin is honest, he responds with $x \in \{0, 1\}^n$ taken to be a preimage of y_i under C_i whenever such a preimage exists. If there are many such i , Merlin chooses the smallest such i . Arthur then checks whether $C_i(x) = y_i$ for some i . If so, Arthur again outputs the special symbol $?$. Otherwise, Arthur outputs whatever $\mathcal{B}^\mathcal{O}(z)$ outputs with its oracle query responses set to $y_1, \dots, y_Q$.

Now, notice that if Merlin is honest, then Arthur always either outputs the correct answer or $?$. (He never rejects when he should accept or accepts when he should reject.) And, regardless of Merlin's behavior, Arthur will output the correct answer whenever the event G occurs, where G is the event that $y_i \notin \text{Im}(C_i)$ for all i . Notice that

$$\Pr[G] \geq 2^{-Q}.$$

So if Arthur and Merlin repeat this protocol in parallel roughly 2^Q times and Arthur decides what to output as above (outputting $\perp$ if any subprotocols disagree, and otherwise outputting the unique decision of all subprotocols that made a decision), we will again obtain a valid protocol deciding L . And, if Q is constant (or even $Q = O(\log |z|)$) then the resulting protocol will be efficient and we again obtain that $L \in \text{AM} \cap \text{coAM}$. However, for $Q = \omega(\log |z|)$, this requires a superpolynomial number of repetitions, and it therefore does not yield an efficient $\text{AM} \cap \text{coAM}$ protocol for L .

A tree-based protocol that works for any Q . To obtain a protocol that works for larger Q , it helps to first observe that the protocol described above is actually the most natural special case of a wider class of protocols. In particular, given any tree $\mathcal{T}$ of height $Q - 1$ and a labeling

$Y : \mathcal{T} \rightarrow \{0, 1\}^{n+1}$ of the tree, we can assign to any vertex $v \in \mathcal{T}$ at depth $k - 1$ in the tree a partial simulation of $\mathcal{B}^{\mathcal{O}}(z)$ in the following way. If $v_1, \dots, v_k := v$ is the path from the root v_1 to the vertex $v = v_k$ in the tree, then we assign to v the simulation of $\mathcal{B}^{\mathcal{O}}(z)$ where the first k oracle query responses are $Y(v_1), \dots, Y(v_k)$. Notice in particular that this uniquely determines a circuit C_v , which is the circuit queried by $\mathcal{B}^{\mathcal{O}}(z)$ after receiving $Y(v_1), \dots, Y(v_{k-1})$ in response to its first $k - 1$ oracle queries. Notice, however, that many of these simulated computations are likely to be *invalid*, since we will often have that $Y(v_i) \in \text{Im}(C_{v_i})$. Such invalid simulations correspond to the behavior of $\mathcal{B}^{\mathcal{O}}$ when its oracle query responses are not correct solutions to the given Avoid instances, and there is no reason to expect that $\mathcal{B}^{\mathcal{O}}$ behaves in any sensible way in such invalid simulations.

The leaves $\ell \in \mathcal{T}$ of depth $Q - 1$ correspond to completed simulations (since such leaves correspond to simulations with a complete set of Q oracle queries and responses), in which $\mathcal{B}^{\mathcal{O}}(z)$ either accepts or rejects. We can therefore think of these leaves as either accepting leaves or rejecting leaves. In particular, for the *valid* leaves, the resulting decision to accept or reject will actually correspond to a valid simulation of $\mathcal{B}^{\mathcal{O}}(z)$ in which all of the oracle query responses $Y(v_i)$ are correct responses to the corresponding oracle queries C_{v_i} . So, the leaf will be accepting if and only if $z \in L$. The goal of our protocol is therefore to have Arthur construct some labeling Y and then to have Merlin help Arthur *prune* the tree so that Arthur is left with only valid leaves, and thus with only valid simulations of $\mathcal{B}^{\mathcal{O}}(z)$.

The simple protocol that we discussed above is the very special case in which the tree $\mathcal{T}$ is just the line graph. The line graph is a natural choice in this context, and viewing the line graph as a tree is perhaps unnecessarily complicated. But, to obtain a protocol that works for any $Q \leq \text{poly}(|z|)$, we will have to be more clever. In particular, we will take $\mathcal{T}$ to be the complete binary tree with depth $Q - 1$, which we write as $\mathcal{T}_Q$.

So, suppose Arthur chooses some suitably random labeling $Y : \mathcal{T}_Q \rightarrow \{0, 1\}^{n+1}$ of the complete binary tree with depth $Q - 1$ and sends it to Merlin. (For now, we will not worry about how Arthur chooses Y or how he sends it to Merlin. We will of course eventually need to worry about the fact that Arthur cannot efficiently send all $2^Q - 1$ such labels to Merlin.) As we discussed above, this labeling (together with $\mathcal{B}$ and z) induces a query circuit C_v for each vertex v in $\mathcal{T}_Q$, and every path $v_1, \dots, v_k$ from the root v_1 to a vertex v_k of depth $k - 1$ corresponds to a partial simulation of $\mathcal{B}^{\mathcal{O}}(z)$ in which we set the response to the i th oracle query C_{v_i} to be $Y(v_i)$.

Our goal is therefore to use Merlin to help Arthur learn the subtree $\mathcal{T}^* \subseteq \mathcal{T}_Q$ of valid nodes. I.e., a vertex $v_k \in \mathcal{T}_Q$ with depth $k - 1$ is included in $\mathcal{T}^*$ if the path $v_1, \dots, v_k$ from the root v_1 to v_k satisfies $Y(v_i) \notin \text{Im}(C_{v_i})$ for all i . As we discussed above, any leaves of depth $Q - 1$ in $\mathcal{T}^*$ correspond to complete valid simulation of $\mathcal{B}^{\mathcal{O}}(z)$, so that such a leaf will be accepting if and only if $z \in L$.

To that end, if Merlin is honest, he will send Arthur a list (x_v) of strings $x_v \in \{0, 1\}^n$ such that $C_v(x_v) = Y(v)$, where v ranges over the vertices $v \in \mathcal{T}_Q$ such that $Y(v) \in \text{Im}(C_v)$ but $Y(v') \notin \text{Im}(C_{v'})$ for any ancestor v' of v . $\mathcal{T}^*$ is then the tree obtained by removing each such vertex v from $\mathcal{T}_Q$, together with all of its descendants. (The set of vertices v chosen by Merlin is the minimal set with this property. Its size is at most $2|\mathcal{T}^*| + 1$.)

When Arthur receives Merlin's list (x_v) , he will iteratively construct a tree $\mathcal{T}^\dagger$, which he hopes is equal to $\mathcal{T}^*$. In particular, $\mathcal{T}^\dagger \subseteq \mathcal{T}_Q$ will be the subtree of $\mathcal{T}_Q$ where $v \in \mathcal{T}^\dagger$ if and only if (1) its parent is in $\mathcal{T}^\dagger$ (or v is the root); and (2) Merlin's list does *not* contain x_v such that $C_v(x_v) = Y(v)$. Notice that Arthur can construct $\mathcal{T}^\dagger$ given Merlin's list (x_v) in time proportional to the size $|\mathcal{T}^\dagger|$.

After building the tree $\mathcal{T}^\dagger$, Arthur decides what to output as follows. If $\mathcal{T}^\dagger$ contains no leaves of depth $Q - 1$, then Arthur outputs $?$, as his tree contains no accepting or rejecting nodes. If $\mathcal{T}^\dagger$ contains *both* accepting and rejecting leaves, Arthur outputs $\perp$. (In this case, $\mathcal{T}^\dagger$ must contain some invalid simulations, so Merlin must have lied to Arthur.) Otherwise, Arthur accepts if there is an accepting leaf and rejects otherwise.

It is much harder to analyze the behavior of this protocol. However, notice that this protocol still has the property that Arthur will always either output the right answer or $?$ when Merlin is honest. Furthermore, let G be the event that there exists a leaf ℓ of depth $Q - 1$ in $\mathcal{T}^*$. (Notice that this is the same as the good event G from above if we take $\mathcal{T}$ to be the line graph rather than the complete binary tree.) Observe that when G happens, $\mathcal{T}^\dagger$ will always also contain a valid leaf, regardless of Merlin's behavior. (Again, we are relying on the fact that Merlin cannot convince Arthur that $Y(v)$ has a preimage under C_v when it does not. This in particular implies that $\mathcal{T}^\dagger$ contains $\mathcal{T}^*$ as a subtree.) So, Arthur will always output the right answer (or $\perp$) when G happens, in close analogy with the above.

To finish the analysis, we would like to argue that Arthur can efficiently choose the labeling $Y : \mathcal{T}_Q \rightarrow \{0, 1\}^{n+1}$ in such a way (1) that the event G happens with probability at least $1/\text{poly}(Q)$; and (2) that $|\mathcal{T}^*| \leq \text{poly}(Q)$ with high probability. This first property guarantees that with probability $1/\text{poly}(Q)$ Arthur will output the correct answer (or $\perp$) *regardless of Merlin's behavior*. The second property guarantees that when Merlin is honest the running time of Arthur is polynomially bounded.

In the sequel, we will choose the labeling Y from a k -wise independent hash function for a suitable choice of k . Here in the introduction, we simply pretend that Y is a truly random function. (Of course, Arthur cannot sample a random function on $2^Q - 1$ inputs efficiently. So, this is just for the purposes of exposition.)

Then, let us consider the probability (over Y) that a vertex $v \in \mathcal{T}_Q$ is included in $\mathcal{T}^*$, conditioned on its parent lying in $\mathcal{T}^*$. Notice that this happens if and only if $Y(v) \notin \text{Im}(C_v)$. So, for fixed $C_v : \{0, 1\}^n \rightarrow \{0, 1\}^{n+1}$, this probability is exactly

$$\Pr[Y(v) \notin \text{Im}(C_v)] = 1 - |\text{Im}(C_v)|/2^{n+1} \geq 1/2.$$

Furthermore, this probability will be exactly $1/2$ if $|\text{Im}(C_v)| = 2^n$, i.e., if C_v is injective.

If we assume for the moment that all of the circuits C_v queried by $\mathcal{B}^\mathcal{O}$ are in fact injective, then our tree $\mathcal{T}^*$ is simply a random subtree of $\mathcal{T}_Q$ in which each child of a vertex in $\mathcal{T}^*$ is included with probability exactly $1/2$. In particular, notice that a vertex $v \in \mathcal{T}_Q$ with depth d will be included in $\mathcal{T}^*$ with probability exactly 2^{-d-1} . Since there are 2^d vertices in $\mathcal{T}_Q$ with depth d , the expected number of vertices with depth d is exactly $1/2$. We then prove suitable concentration bounds on these quantities to argue that with high probability the total number of nodes remains small, but with decent probability there will be at least one node of depth $Q - 1$, as needed. (Our analysis mimics that of the closely related Galton-Watson process [WG75].)

However, if the circuits C_v are not necessarily injective, then we encounter additional complications. In particular, recall that the probability of a node $v \in \mathcal{T}_Q$ being included in $\mathcal{T}^*$ when its parent is in $\mathcal{T}^*$ is $1 - |\text{Im}(C_v)|/2^{n+1}$. If the circuit C_v is not injective, this probability can be significantly larger than $1/2$, which can hugely increase the expected size of $\mathcal{T}^*$, making the protocol inefficient.

To solve this issue, for each node v in the tree, we would like to flip a biased coin that lands on heads with probability p_v , where $p_v := 2^n / (2^{n+1} - |\text{Im}(C_v)|)$ is chosen so that

$$p_v \cdot \left(1 - \frac{|\text{Im}(C_v)|}{2^{n+1}}\right) = 1/2.$$

We then only include a node v in $\mathcal{T}^*$ if its parent is in $\mathcal{T}^*$, $Y(v) \notin \text{Im}(C_v)$, and this coin lands on heads. In other words, we will do some *additional pruning* of our tree to maintain the invariant that each node of depth d in the tree is included with probability 2^{-d-1} . The randomness used to flip these coins will again be provided by Arthur (using a suitable k -wise independent function). (Notice that it is absolutely necessary that Arthur provides the randomness for this additional pruning; if we instead trusted Merlin to decide which nodes to prune, a dishonest Merlin could simply remove all valid leaves from the tree.)

Of course, Arthur does not know p_v , since Arthur does not know $|\text{Im}(C_v)|$. Fortunately, we show in [Section 5](#) that it suffices to have Merlin provide Arthur with a good upper bound T_v of $|\text{Im}(C_v)|$ and convince him that this upper bound is valid using our range-size upper bound protocol.⁵

This completes our description of the protocol and its analysis. (See [Section 5](#) for the details.) Stepping back a bit, we observe that this tree-based protocol intuitively allows Arthur to “use his luck more efficiently” than the simpler line-based protocol. In particular, notice that both the line-based protocol and the tree-based protocol treat the first query $C_{v_1} : \{0, 1\}^n \rightarrow \{0, 1\}^{n+1}$ made by $\mathcal{B}^{\mathcal{O}}(z)$ identically. In both cases, Arthur simply samples $Y(v_1) \sim \{0, 1\}^{n+1}$ uniformly at random and hopes that $Y(v_1) \notin \text{Im}(C_{v_1})$. Unfortunately, this can fail up to half of the time. In the simple line-based protocol, Arthur treats the second query in exactly the same way—sampling a single uniformly random candidate response to this second query, potentially failing with probability $1/2$. But, in the tree-based protocol, Arthur effectively decides that if he happens to be lucky enough to sample $Y(v_1) \notin \text{Im}(C_{v_1})$ for the response to the first query, he should make use of this opportunity by allowing himself two independent chances to sample a valid response to the second oracle query, corresponding to the two labels of the two children of the root node. In the simple case when C_{v_1} is injective, these two independent chances will be irrelevant half of the time (since half of the time $Y(v_1) \in \text{Im}(C_{v_1})$), so that in expectation Arthur effectively tries exactly one candidate response for the second query. (When C_{v_1} is not injective, this remains true after we do our additional pruning.) This idea then propagates so that in expectation Arthur has one candidate response for each oracle query.

1.2.4 A range-size upper bound protocol

Finally, we describe the range-size upper bound protocol from [Theorem 1.4](#). In other words, we show a protocol in which Merlin can convince Arthur that $|\text{Im}(C)|$ is *small* for some circuit $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$.

For this, we recall the celebrated Goldwasser-Sipser set-size *lower bound* protocol [[GS86](#)]. This protocol allows Merlin to convince Arthur that a set $S \subseteq \{0, 1\}^m$ is *large*, provided that membership in S can be recognized in AM, i.e., provided that Merlin can convince Arthur that $x \in S$ whenever

⁵In [Section 5](#), we also deal with additional technicalities that we have avoided here. Most notably, we allow for randomized reductions that only output the right answer with probability at least $2/3$ when instantiated with a random Avoid oracle. This means that we must (1) argue that we can boost the success probability of such a reduction; and (2) pay careful attention to the distribution of the $Y(v_i)$ along paths $v_1, \dots, v_Q \in \mathcal{T}^*$ from the root v_1 to a leaf v_Q . Indeed, the additional pruning that we perform is not only needed to keep the size of $\mathcal{T}^*$ manageable. It is also necessary to ensure that the leaves in $\mathcal{T}^*$ have the correct acceptance probability.

x is in fact in S . (The Goldwasser-Sipser protocol is commonly described as working for sets S that are recognizable in NP , but it naturally extends to AM and even to prAM . We include a proof in [Section 3.1](#) for completeness.)

So, in order to build our protocol, we would like some way to reduce the problem of showing that the image of a circuit is *small* to the problem of showing that certain recognizable sets are *large*. To do so, we first note a simple double-counting identity, which says that

$$|\text{Im}(C)| = 2^n - \sum_{i=2}^{2^n} |S_i| ,$$

where

$$S_i := \{y \in \{0,1\}^m : |C^{-1}(y)| \geq i\} .$$

In other words, S_i is the set of all elements with at least i preimages. This identity shows that in order to prove that $|\text{Im}(C)|$ is small, it suffices to show that the sum is large.

We then “apply Goldwasser-Sipser twice.” In particular, notice that membership in $C^{-1}(y)$ can be recognized in $\text{NP} \subseteq \text{AM}$ (i.e., given x and y , one can simply check if $C(x) = y$ to determine whether $x \in C^{-1}(y)$). So, by Goldwasser-Sipser, it follows that membership in S_i can be recognized in AM .⁶ Therefore, by applying Goldwasser-Sipser again, we can obtain a lower bound on $|S_i|$.

Finally, we show that we can replace the sum $\sum_{i=2}^{2^n} |S_i|$ by a much coarser sum

$$\sum_{j=1}^{\text{poly}(n)} \beta_j |S_{i_j}|$$

for carefully chosen indices i_j and weights β_j without incurring much error.

Putting everything together, we can use the Goldwasser-Sipser protocol polynomially many times (in parallel) to obtain lower bounds on S_{i_j} for a suitable choice of polynomially many indices i_j , which in turn yields an upper bound on $|\text{Im}(C)|$.

1.3 Related work

There is already a large body of literature on Avoid. Here, we describe some of the work that is most closely related to the present work.

Korten and Santhanam showed that under very strong derandomization assumptions, certain *uniform* versions of Avoid can be solved by efficient deterministic algorithms [KS25a]. In more detail, they show that for any $\text{poly}(n)$ -time computable family of circuits $C_n : \{0,1\}^n \rightarrow \{0,1\}^m$, there is an efficient deterministic algorithm that solves Avoid on C_n , where depending on the strength of the derandomization assumption their algorithm works for either $m = \text{poly}(n)$, $m = n \cdot \text{poly}(\log n)$, or $m = n + o(n)$. Our [Theorem 1.2](#) requires a weaker derandomization assumption, and it solves a harder problem in that it works for any input (rather than just $\text{poly}(n)$ -time computable families) and for $m = n + 1$ (the hardest choice). (Indeed, the problem that we solve is widely believed to *not* be solvable by efficient deterministic algorithms.) But, our algorithm is randomized and requires access to an oracle for a problem that is in TFNP . In short, we place a weaker upper bound on a harder computational problem under a weaker computational assumption.

⁶Here, we have ignored the fact that Goldwasser-Sipser only allows us to *approximately* estimate set size. (Computing such set sizes exactly is of course $\#\text{P}$ -complete.) This means that Goldwasser-Sipser only lets us distinguish elements that are in S_i from elements that are not even in $S_{(1-1/\text{poly}(n))i}$. This leads to some additional technicalities, but the high-level idea does not change.

Our proof of [Theorem 1.2](#) is heavily inspired by Korten and Santhanam’s work. Indeed, [\[KS25a\]](#) similarly uses a suitably pseudorandom string to effectively derandomize the basic idea of using a random string as a solution to Avoid. However, in [\[KS25a\]](#), they need to argue that a single sufficiently pseudorandom string can itself be used as a solution to an Avoid instance consisting of a sufficiently expanding *uniform* circuit, while we need the much simpler observation that a sufficiently pseudorandom *list* of strings must contain many solutions to any Avoid instance (of a given size). (The technical difficulty in proving [Theorem 1.2](#) is in showing how to use this pseudorandom list of candidate solutions to reduce Avoid to a problem in TFNP.)

To the authors’ knowledge, prior to the present work the only result showing a barrier to proving hardness of Avoid was the following simple observation in [\[ILW23, CGL⁺26\]](#). (To be clear, this observation was not the focus of either [\[ILW23, CGL⁺26\]](#).) Suppose that SAT reduces to Avoid via a reduction that makes q queries to the Avoid oracle on circuits $C : \{0, 1\}^n \rightarrow \{0, 1\}^{n+s}$ with $q \ll 2^s$. Then, we can conclude that $\text{SAT} \in \text{RP}$ (which implies that $\text{NP} = \text{RP}$), since a randomized algorithm can simply simulate solutions to the oracle queries with random strings and succeed with probability at least $1 - q2^{-s}$. Both works asked whether any barriers can be shown to proving NP-hardness of Avoid with a larger number of queries q , smaller stretch s , or both. We resolve this by showing such a barrier for any (polynomial) number of queries and minimal stretch $s = 1$.

Ilango, Li, and Williams [\[ILW23\]](#), Chen and Li [\[CL24\]](#), and Ren, Wang, and Zhong [\[RWZ26\]](#) showed that Avoid is hard under strong cryptographic and complexity-theoretic assumptions. [\[CL24, RWZ26\]](#) even showed that $\text{Avoid} \notin \text{SearchNP}$ under strong cryptographic assumptions (even for restricted classes of input circuits). These proofs are beautiful but rather indirect, and it is natural to ask whether we can simply show a reduction from a presumed hard problem to Avoid, ideally an NP-hard problem. Our [Theorem 1.1](#) effectively rules this out by showing that any reduction from an NP-hard problem to Avoid would collapse the polynomial hierarchy. The results of [\[CL24, RWZ26\]](#) also show that the randomness in [Theorem 1.2](#) is necessary, since any problem that reduces to TFNP deterministically must lie in SearchNP .

Like us, Chen, Li, and Liang also provide “an Arthur-Merlin protocol for Avoid” [\[CLL25\]](#). However, for their application, they must construct a *single-valued* protocol (i.e., a protocol in which Arthur is likely to output some canonical solution, regardless of any randomness or what Merlin says). To obtain a single-valued protocol, they require non-uniform advice and quasi-polynomial time, and they only solve *uniform* Avoid instances for infinitely many different input lengths. ([\[CLL25\]](#) then uses this protocol to prove breakthrough circuit lower bounds in $\text{AMEXP}_{/2^{n^\epsilon}}$!) We do not know of any direct relationship between their protocol and ours, but we do note that both protocols rely on the same fundamental properties of Avoid—that a random string is a solution with decent probability and that a solution can be verified in coAM (in fact, solutions to Avoid can be verified in coNP , but both protocols work with the weaker property of verifiability in coAM).

[\[CGL⁺26\]](#) showed how to build cryptography under the assumption that no efficient randomized algorithm can solve Avoid with probability $1 - 1/\text{poly}(n)$. Our [Theorems 1.2](#) and [1.3](#) show that Avoid can be solved with this probability with access to a TFNP oracle or via interaction with an all-powerful prover Merlin, respectively. This shows that the problem of solving Avoid with probability $1 - 1/\text{poly}(n)$ in some sense lives rather low in the polynomial hierarchy, similar to nearly all known problems whose hardness implies cryptography.

Finally, we note that Fortnow [\[For87\]](#) presented a protocol that is similar to our range-size upper bound protocol from [Theorem 1.4](#). Formally, Fortnow gave a protocol that allows Merlin to convince Arthur of an upper bound on the size of a set S , provided that Arthur has the ability to sample

a random element from S . For example, if the circuit C is somehow known to be regular (i.e., if every element in the image has the same number of preimages), then Fortnow’s protocol allows Merlin to convince Arthur of an upper bound on the size of the image (in fact, with multiplicative error, whereas we only achieve additive error). (We thank Korten for bringing this similarity to our attention.)

1.4 Future directions

We briefly list some possible future directions.

First, recall that we show that $\text{BPP}^{\text{Avoid}} \subseteq \text{AM} \cap \text{coAM}$. In other words, any decision problem that reduces to Avoid must be in $\text{AM} \cap \text{coAM}$. This effectively rules out NP-hardness of Avoid, but it does not rule out the possibility of a reduction from some presumed hard decision problem in $\text{AM} \cap \text{coAM}$ to Avoid. (E.g., cryptographers study many problems that are thought to be hard but are in $\text{AM} \cap \text{coAM}$ and even in much smaller classes.) So, perhaps we can still reduce a well-studied presumed hard problem to Avoid.

Alternatively, perhaps $\text{BPP}^{\text{Avoid}}$ (or P^{Avoid}) lies in an even smaller complexity class. Indeed, since there is no known reduction from a presumed hard decision problem to Avoid, we see no fundamental barrier to proving that decision problems that reduce to Avoid lie in much smaller complexity classes than $\text{AM} \cap \text{coAM}$.⁷ Perhaps a natural first subclass of $\text{AM} \cap \text{coAM}$ to consider is SZK, though our current protocols are certainly not SZK protocols.

In a different direction, recall that we have shown that there is a randomized reduction from Avoid to a problem in TFNP (under a complexity-theoretic assumption). Chen and Li’s proof that $\text{Avoid} \notin \text{SearchNP}$ (under cryptographic assumptions) shows that we are unlikely to derandomize this reduction. However, we can hope to improve on it in a different direction. In particular, the specific problem $T \in \text{TFNP}$ that we reduce to is rather artificial. So, we ask whether there is a (necessarily randomized) reduction from Avoid to a more natural problem in TFNP. For example, as was already observed in [KKMP21], Avoid seems closely related to the subclass $\text{PWPP} \subseteq \text{TFNP}$. Informally, PWPP is the complexity class that captures “computational problems whose totality is proven via the (weak) pigeonhole principle” while Avoid corresponds to “computational problems whose totality is proven via the *dual* (weak) pigeonhole principle.” It is quite natural to ask whether there is a reduction between Avoid and PWPP in either direction or whether one can show an oracle separation. The same problem is interesting for any of the well-studied subclasses of TFNP.

Finally, we note that the range-size upper bound protocol described in Theorem 1.4 seems like quite a general tool that complements the celebrated Goldwasser-Sipser protocol [GS86]. We therefore ask whether it has additional applications. (In particular, our protocol works for any circuit and does not require the circuit to be expanding.)

Acknowledgments

The authors would like to thank Huck Bennett, Sasha Golovnev, and Bobby Kleinberg for helpful discussions. We also thank Oliver Korten for comments on an earlier version of this work.

⁷As far as we know, it is even plausible that $\text{BPP}^{\text{Avoid}} = \text{BPP}$, even if Avoid itself is hard. For example, letting Avoid_m be the version of Avoid in which the input circuit maps n bits to $m := m(n)$ bits, $\text{BPP}^{\text{Avoid}_m} = \text{BPP}$ for any $m > n + \omega(\log n)$ [ILW23, CGL⁺26], but it is still certainly plausible (perhaps likely) that $\text{Avoid}_m \notin \text{FP}$. Similarly, the problem of outputting an n -bit string with large Kolmogorov complexity is provably uncomputable (and therefore certainly not in FP), but for suitable parameters any decision problem that reduces to it is again in BPP.

History and AI disclosure

The present work is a substantial modification of an earlier paper. The earlier paper (which can be found in the version history on ECCC) contained an incorrect proof of [Theorem 1.1](#) (i.e., of the fact that decision problems that reduce to Avoid are in $\text{AM} \cap \text{coAM}$), and this paper contains a substantially different correct proof of the same fact. (Technically, the present work proves a slightly stronger version of this, which holds relative to a random Avoid oracle.) Here, we document the history of the paper and disclose the role of AI in this history.

The original version of this article (in which AI was only used to proofread and produce a figure) was posted publicly to ECCC in December, 2025 and was accepted to STOC 2026. In June, 2026, the authors asked GPT 5.5 Pro to check this version for any logical gaps. GPT identified what it said was a lack of formality in one step of the proof of soundness of our proposed $\text{AM} \cap \text{coAM}$ protocol. In attempting to understand the issue flagged by GPT, we eventually realized that this was not merely a lack of formality but an actual bug. In fact, the protocol was not sound. (Very roughly speaking, the bug had to do with the fact that Merlin could choose his responses for earlier queries in ways that would affect later queries, allowing for an attack on the protocol and violating an implicit assumption of independence in our flawed proof of soundness.)

We eventually asked GPT 5.5 to give a revised protocol that was correct, and it output a tree-based protocol very similar to the one that we present in [Section 5](#). The authors confirmed the correctness of the main idea and used this idea to develop the protocol as presented in [Section 5](#). (This protocol resembles the one proposed by GPT at a high level, but the details and writeup are quite different.)

So, GPT 5.5 identified a part of our proof that seemed informal, which we then realized was actually hiding a major bug. GPT 5.5 then proposed the high-level idea of modifying our original flawed protocol to create the tree-based protocol in [Section 5](#). (We find this to be a startlingly clever idea.) The actual details were written by the authors.

Other parts of the paper (including the range-size upper bound protocol, the SearchAMA protocol for Avoid, and the randomized reduction from Avoid to a problem in TFNP) are essentially unchanged from the earlier version.

Gemini 3.1 Pro, GPT 5.6 Sol, GPT 6, and Claude Fable 5.1 were also used for correctness checks (which found and resolved minor errors/inconsistencies throughout) and copy editing of the entire work.

The authors assume full responsibility for correctness.

2 Preliminaries

2.1 Search problems

A search problem is different from a decision problem in that it may have a multi-bit output and some (or all) inputs may have multiple possible solutions. Search problems are therefore formalized as relations.

Definition 2.1. *A search problem is a relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$. We say that y is a solution to R on input x if $(x,y) \in R$. We write $R(x)$ to denote $\{y : (x,y) \in R\}$.*

An important complexity class in this context is TFNP, which consists of search problems for which (1) there always exists a solution; and (2) solutions are efficiently checkable.

Definition 2.2. A relation R is in TFNP if the following conditions hold:

1. R is total: for all $x \in \{0, 1\}^*$, there exists $y \in \{0, 1\}^*$ such that $(x, y) \in R$.
2. R is polynomial: For all $(x, y) \in R$, $|y| \leq \text{poly}(|x|)$.
3. There exists a polynomial time Turing machine $\mathcal{M}$ such that $(x, y) \in R$ if and only if $\mathcal{M}(x, y) = 1$.

While the notion of a reduction from one decision problem to another is clear, what it means for one *search* problem to reduce to another can be rather tricky since search problems may have multiple solutions. The situation becomes more complicated still when the reduction is a randomized Turing reduction. We will be somewhat informal about this (as is standard in the literature) and simply consider the strongest type of randomized reduction $\mathcal{M}^B$ from A to B that succeed with probability $2/3$ whenever the responses of $\mathcal{O}$ to queries are solutions to B . The weaker type is where the oracle $\mathcal{O}$ is fixed in advance and the reduction should work for all *fixed* oracles $\mathcal{O}$ solving B . We will not dwell on this distinction except to say that our results always hold for whichever type of reduction gives the stronger result.

2.2 Avoid

The following useful lemma tells us that we can always “upward self-reduce” Avoid simply by padding the input and output.

Lemma 2.3. *There is a (simple) linear-time, deterministic, single-query reduction from Avoid on a circuit $C : \{0, 1\}^n \rightarrow \{0, 1\}^{n+1}$ with $|C| = s$ to Avoid on a circuit $C' : \{0, 1\}^{n'} \rightarrow \{0, 1\}^{n'+1}$ where $|C'| \leq n'^2$ for any $n' > s$.*

Proof. The reduction constructs a circuit $C' : \{0, 1\}^n \times \{0, 1\}^{n'-n} \rightarrow \{0, 1\}^{n'+1}$ such that $C'(x, x') = C(x) \circ x'$. It is not hard to see that $|C'| \leq s + O(n') = O(n')$ which is less than n'^2 for sufficiently large s . We feed C' to our range avoidance oracle to get back an avoided element $y \in \{0, 1\}^{n'+1}$. We decompose $y = y_1 \circ y_2$ such that $y_1 \in \{0, 1\}^{n+1}$ and output y_1 . The reduction clearly runs in $O(s + n')$ time since all we need to do to construct C' is pad C with $n' - n$ extra input and output bits. To see that y_1 is not in the range of C , simply observe that if it were and had preimage x_1 in C , then $y_1 \circ y_2$ would be in the range of C' since $C'(x_1, y_2) = C(x_1) \circ y_2 = y_1 \circ y_2$. $\square$

Let AvoidInst denote the set of syntactically valid Avoid instances of stretch 1. I.e., an instance $C \in \text{AvoidInst}$ is a Boolean circuit $C : \{0, 1\}^n \rightarrow \{0, 1\}^{n+1}$

for some positive integer n , and the set of valid answers to C is

$$\text{Sol}(C) := \{y \in \{0, 1\}^{n+1} \mid y \notin \text{Im}(C)\}.$$

A Range Avoidance oracle is a function

$$\mathcal{O} : \text{AvoidInst} \rightarrow \{0, 1\}^*$$

such that

$$\mathcal{O}(C) \in \text{Sol}(C)$$

for every $C \in \text{AvoidInst}$.

Let RAvoid be the collection of all Range Avoidance oracles and we denote $\mathcal{O} \sim \text{RAvoid}$ as drawing a uniformly random oracle from RAvoid . In particular, for every $C \in \text{AvoidInst}$ and $y \in \text{Sol}(C)$, it holds that

$$\Pr_{\mathcal{O} \sim \text{RAvoid}}[\mathcal{O}(C) = y] = \frac{1}{|\text{Sol}(C)|} ,$$

and is independent of the solutions to other queries.

With the definition for Avoid oracles in mind, we specify what we mean by a reduction to Avoid.

Definition 2.4. *Let q, m be polynomials. A promise problem (Π_0, Π_1) is in $\text{prBPP}^{\text{RAvoid}}$ if there is a probabilistic polynomial-time oracle machine $\mathcal{B}^{\mathcal{O}}$ that on input $x \in \Pi_b$ and its internal randomness r , makes exactly $q(|x|)$ distinct adaptive queries to instances from AvoidInst mapping $m(|x|)$ -bits to $m(|x|) + 1$ -bits, and satisfies that:*

$$\Pr_{r, \mathcal{O} \sim \text{RAvoid}}[\mathcal{B}^{\mathcal{O}}(x, r) = b] \geq 2/3 .$$

Remark 2.5. *In the definition above, we assume $\mathcal{B}$ makes exactly $q(|x|)$ distinct queries to instances from AvoidInst and all queries are circuits mapping $m(|x|)$ -bits to $m(|x|) + 1$ -bits. This is without loss of generality because*

1. $\mathcal{B}$ can make dummy queries if it needs less than q queries.
2. If $\mathcal{B}$ needs more than one solution to the same instance C , it could construct C' by adding a dummy gate in C and query its oracle on C' instead.
3. $\mathcal{B}$ makes only queries mapping m -bits to $m + 1$ -bits, due to the 'upward self-reducibility' from [Lemma 2.3](#).

We note that one can also amplify the success probability via careful parallel repetitions.

Lemma 2.6. *Let q, m be polynomials. Let (Π_0, Π_1) be a promise problem in $\text{prBPP}^{\text{RAvoid}}$ where on input $x \in \Pi_b$ the reduction algorithm $\mathcal{B}^{\mathcal{O}}$ makes exactly $q(|x|)$ distinct adaptive queries to instances mapping $m(|x|)$ -bits to $m(|x|) + 1$ -bits.*

Then for $t = O(n)$, there is another polynomial-time oracle algorithm $\mathcal{B}'^{\mathcal{O}}$ that makes qt distinct adaptive queries to instances from AvoidInst mapping $m(|x|)$ -bits to $m(|x|) + 1$ -bits, and satisfies that

$$\Pr_{r, \mathcal{O} \sim \text{RAvoid}}[\mathcal{B}'^{\mathcal{O}}(x, r) = b] \geq 1 - 2^{-n} .$$

Proof. Set $t = 20n$. Consider an alternative algorithm $\tilde{\mathcal{B}}$ that simulates $\mathcal{B}$, but on each oracle query C , adds at most qt dummy gates to C to obtain another equivalent circuit C' and queries on C' . Further assume that all queries made by $\tilde{\mathcal{B}}$ are distinct.

By definition of the randomly drawn oracle $\mathcal{O} \sim \text{RAvoid}$, it holds that

$$\Pr_{r, \mathcal{O} \sim \text{RAvoid}}[\tilde{\mathcal{B}}^{\mathcal{O}}(x, r) = b] = \Pr_{r, \mathcal{O} \sim \text{RAvoid}}[\mathcal{B}^{\mathcal{O}}(x, r) = b] .$$

The algorithm $\mathcal{B}'$ simulates t independent copies of $\mathcal{B}^{\mathcal{O}}$ (using fresh randomness r for each copy) and hence makes at most qt queries. Moreover, whenever it is about to query an instance C that has been queried before, it adds (at most qt) dummy gates to C to generate a fresh query C' and queries C' instead. And it outputs the majority answer of the t copies.

Since both the internal randomness r and oracle queries are independent, $\mathcal{B}'$ effectively makes t independent simulations of $\tilde{\mathcal{B}}^{\mathcal{O}}(x, r)$. Let X be the random variable counting the number of simulations that correctly outputs b . We have $\mathbb{E}[X] \geq 2t/3$. By Chernoff's inequality,

$$\Pr[X \leq t/2] \leq e^{-2(t/6)^2/t} = e^{-t/18} \leq 2^{-n}.$$

Hence, $\mathcal{B}'$ outputs b with probability at least $1 - 2^{-n}$. $\square$

2.3 Pseudorandom generators against $\text{NP}_{\parallel}$ circuits

One of the key assumptions that we will make is the existence of pseudorandom generators against $\text{NP}_{\parallel}$ circuits. We first define non-deterministic circuits and pseudorandom generators.

Definition 2.7. A non-deterministic circuit $C : \{0, 1\}^n \rightarrow \{0, 1\}$ is a circuit which has along with its usual n bit input, m non-deterministic bits as input. We say that the circuit evaluates to 1 on input $x \in \{0, 1\}^n$ if there exists an assignment of the m non-deterministic bits that causes the circuit to output 1, and 0 otherwise.

We note that non-deterministic circuits are different from $\text{NP}_{\parallel}$ circuits, which is how one may initially think to formalize non-deterministic circuits.

Definition 2.8. An $\text{NP}_{\parallel}$ oracle circuit is a circuit with oracle gates to SAT where all oracle queries are made in parallel (i.e. non-adaptively).

Definition 2.9. A distribution X on n bits is ε -pseudorandom for a class $\mathcal{C}$ of functions, if for every $C \in \mathcal{C}$,

$$\left| \Pr_{y \sim X}[C(y) = 1] - \Pr_{y \sim \{0, 1\}^n}[C(y) = 1] \right| \leq \varepsilon.$$

A function $G : \{0, 1\}^d \rightarrow \{0, 1\}^n$ is an ε -PRG for $\mathcal{C}$ if $G(X)$ is ε -pseudorandom for $\mathcal{C}$ where $X \sim \{0, 1\}^d$ is a uniformly random seed.

Our key assumption will be the existence of PRGs against $\text{NP}_{\parallel}$ circuits. This assumption can itself be based on the plausible assumption that $\mathbf{E}$ (the set of problems solvable in $2^{O(n)}$ time) requires exponential size non-deterministic circuits. One should view this as a variant of the classic result of Impagliazzo and Wigderson that if $\mathbf{E}$ requires exponential size circuits, then we have PRGs good enough to get $\text{BPP} = \text{P}$ [IW97], and it follows from a long line of work [IW97, KvM99, SU06, SS24]. The precise version that we use is from [SS24, Theorem 2.8].

Assumption 2.10. For every constant $c > 1$, there exists a constant $\alpha > 1$ such that for every sufficiently large m , there is a $G : \{0, 1\}^{\alpha \log m} \rightarrow \{0, 1\}^m$ that is a $\frac{1}{m^c}$ -PRG for $\text{NP}_{\parallel}$ circuits of size m^c . Furthermore, G is computable in time $\text{poly}(m)$.

Theorem 2.11 ([IW97, KvM99, SU06, SS24]). If $\mathbf{E}$ requires exponential size non-deterministic circuits, then Assumption 2.10 holds.

We now prove the following useful lemma which tells us that the output $G(1), \dots, G(k)$ of a PRG against $\text{NP}_{\parallel}$ circuits will contain roughly as many solutions to an Avoid instance as a random list of strings.

Lemma 2.12. *Let $G : \{0, 1\}^{\alpha \log(n+1)} \rightarrow \{0, 1\}^{n+1}$ be a $\frac{1}{n^3}$ -PRG which is secure against $\text{NP}_{\parallel}$ circuits of size at most n^3 . Let $C : \{0, 1\}^n \rightarrow \{0, 1\}^{n+1}$ be a circuit of size at most n^2 . Then*

$$\left| \Pr_{i \sim [(n+1)^\alpha]}[G(i) \in \text{Im}(C)] - \frac{|\text{Im}(C)|}{2^{n+1}} \right| \leq \frac{1}{n^3}.$$

Proof. Let $D : \{0, 1\}^{n+1} \rightarrow \{0, 1\}$ be the $\text{NP}_{\parallel}$ circuit which on input $y \in \{0, 1\}^{n+1}$ outputs 1 if there exists an x such that $C(x) = y$ and outputs 0 otherwise. There is the minor subtlety that writing the statement $\exists x, C(x) = y$ as a SAT instance incurs some blowup, so D may be slightly larger than C . However, we can safely bound the blowup so that $|D| \leq n^3$.

Notice also that

$$\Pr_{y \sim \{0, 1\}^{n+1}}[D(y) = 1] = \frac{|\text{Im}(C)|}{2^{n+1}}.$$

The result follows by observing that G is a $\frac{1}{n^3}$ -PRG which fools size n^3 circuits, including D . $\square$

2.4 Checkability

Intuitively, a relation R is checkable if we have an efficient algorithm with the following property. When given oracle access to an oracle $\mathcal{O}$ which claims to solve R , our algorithm either solves R or correctly reports that $\mathcal{O}$ in fact does not solve R . Checkability is a notion originally defined in [BK95].

For a relation $R \subseteq \{0, 1\}^* \times \{0, 1\}^*$, we say that $y \in \{0, 1\}^* \cup \{\perp\}$ is a *solution* to $R(x)$ if $y \in R(x)$ or if $R(x) = \emptyset$ and $y = \perp$.

Definition 2.13 ([BK95, MX10]). *Let $R \subseteq \{0, 1\}^* \times \{0, 1\}^*$ be a relation. R is checkable if there exists an efficient randomized oracle algorithm $\mathcal{A}$ such that for all oracles $\mathcal{O}$, the following holds.*

- **Completeness:** *Suppose that for all x , $\mathcal{O}(x)$ is a solution to $R(x)$. Then for all x , $\mathcal{A}^{\mathcal{O}}(x)$ is also a solution to $R(x)$.*
- **Soundness:** *For any $\mathcal{O}$ and x , with overwhelming probability $\mathcal{A}^{\mathcal{O}}(x) \in R(x) \cup \{\perp\}$.*

We are particularly interested in the checkability of SAT. The question of whether SAT is checkable was first proposed in [BK95] and remains open to this day. It seems there is no consensus on whether SAT should be checkable or not. However, [MX10] showed that a randomized Turing reduction from SAT to any TFNP problem would imply checkability of SAT, thereby giving a barrier against showing such a reduction.

Theorem 2.14 ([MX10]). *If $\text{SAT} \in \text{BPP}^{\text{TFNP}}$, then SAT is checkable.*

2.5 Arthur-Merlin protocols

Crucial to our results will be the notion of Arthur-Merlin protocols [Bab85]. Informally, we say that a language $L \subseteq \{0, 1\}^*$ has a k -round Arthur-Merlin protocol if there is a k -round game between a prover $\mathcal{M}$ (Merlin) and a verifier $\mathcal{A}$ (Arthur) where if $x \in L$, then an honest Merlin can convince Arthur of that fact; and if $x \notin L$, even a dishonest Merlin cannot convince Arthur that $x \in L$. The protocol begins with Arthur sampling some randomness r_1 and sending it to Merlin, who sends back a message $m_1 = \mathcal{M}(x, r_1)$ which only depends on the randomness sampled so far. Arthur then

samples some new randomness r_2 which he sends to Merlin and Merlin sends back a new message $m_2 = \mathcal{M}(x, r_1, m_1, r_2)$. This continues until k messages have been sent in total, at which point $\mathcal{A}$, given access to all the messages sent so far decides if $x \in L$. We now define 2-round AM, which by [BM88] is known to be equivalent to any k -round AM for any constant k .

Definition 2.15. We say that a promise problem $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}})$ is in $\text{prAM}_{c(n), s(n)}$ if there exists a polynomial time algorithm $\mathcal{A}$ and some (poly(n))-time computable $m := m(n) \leq \text{poly}(n)$ such that the following conditions hold.

- **Completeness:** there exists a (computationally unbounded) prover $\mathcal{M}$ such that if $x \in \Pi_{\text{YES}}$, then

$$\Pr_{r \sim \{0,1\}^m} [\mathcal{A}(x, r, \mathcal{M}(x, r)) = 1] \geq c(n).$$

- **Soundness:** If $x \in \Pi_{\text{NO}}$, then for all (computationally unbounded) provers $\mathcal{M}^\dagger$,

$$\Pr_{r \sim \{0,1\}^m} [\mathcal{A}(x, r, \mathcal{M}^\dagger(x, r)) = 1] \leq s(n).$$

We say that a promise problem Π is in prAM if Π is in $\text{prAM}_{2/3, 1/3}$. A language $L \subseteq \{0,1\}^*$ is in AM if $(L, \bar{L})$ is in prAM .

We will often work with promise problems in AM and use standard properties of Arthur-Merlin protocols which hold in the promise problem setting as well.

Theorem 2.16 ([FGM⁺89, BGG93, Dru10]). Let $p(n), q(n)$ be any polynomials. If a promise problem Π is in $\text{prAM}_{s(n)+1/p(n), s(n)}$, then Π is also in $\text{prAM}_{1, 2^{-q(n)}}$.

Theorem 2.17 ([Bab85, GS86, BM88]). Let k be any constant. If a promise problem Π has a k -round Arthur-Merlin protocol, then $\Pi \in \text{prAM}$.

We will also be interested in Arthur-Merlin-like protocols that solve search problems (namely Avoid). We therefore define SearchAMA.

Definition 2.18. Let R be a search problem. We say that $R \in \text{SearchAMA}_{c(n), s(n)}$ if there exists an efficient verifier $\mathcal{A}$ and some (poly(n))-time computable $m := m(n) \leq \text{poly}(n)$ such that the following holds.

- **Completeness:** There exists a (computationally unbounded) prover $\mathcal{M}$ with polynomially bounded output length such that for all $x \in \{0,1\}^n$ with $R(x) \neq \emptyset$,

$$\Pr_{r_1, r_2 \sim \{0,1\}^m} [\mathcal{A}(x, r_1, \mathcal{M}(x, r_1), r_2) \in R(x)] \geq c(n).$$

- **Soundness:** For all (computationally unbounded) provers $\mathcal{M}^\dagger$ and all $x \in \{0,1\}^n$,

$$\Pr_{r_1, r_2 \sim \{0,1\}^m} [\mathcal{A}(x, r_1, \mathcal{M}^\dagger(x, r_1), r_2) \in R(x) \cup \{\perp\}] \geq 1 - s(n).$$

We now highlight a few differences between AM and SearchAMA. The first difference is that, unlike in the case of decision problems in AM, where parallel repetition allows us to boost the success probability, we do not know whether the success probability can be boosted for SearchAMA. So, we parameterize $\text{SearchAMA}_{c(n),s(n)}$ by the completeness $c(n)$ and soundness error $s(n)$ respectively. The second difference is that we allow Arthur randomness after he sees the prover's answer in SearchAMA—this is what makes this SearchAMA and not SearchAM. In the case of decision problems, this extra randomness is known to be irrelevant (i.e., $\text{AMA} = \text{AM}$ [Bab85, BM88]). It is unclear if one can show a similar result in the case of search Arthur-Merlin protocols.

Finally, we note that under the strong (but standard) derandomization assumption presented in [Assumption 2.10](#), we can derandomize prAM.

Theorem 2.19 ([KvM99]). *If [Assumption 2.10](#) holds, then $\text{prAM} = \text{prNP}$.*

2.6 k -wise independent hashing

We will also need a family of $2Q$ -wise independent hash functions.

Definition 2.20. *A family of functions $\mathcal{H} = \{h : S \rightarrow [M]\}$ is called a family of k -wise independent hash functions if for all distinct k elements $x_1, x_2, \dots, x_k \in S$ and all $v_1, v_2, \dots, v_k \in [M]$,*

$$\Pr_{h \sim \mathcal{H}} [\forall i \in [k], h(x_i) = v_i] = \frac{1}{M^k}.$$

Theorem 2.21. [Vad12, Corollary 3.34] *For every $n, m, k \in \mathbb{N}$, there is a family of k -wise independent hash functions $\mathcal{H} = \{h : \{0, 1\}^n \rightarrow \{0, 1\}^m\}$ such that choosing a random function from $\mathcal{H}$ takes $k \cdot \max(n, m)$ random bits, and evaluating a function from $\mathcal{H}$ takes time $\text{poly}(n, m, k)$.*

Recall that we write $\mathcal{T}_Q$ for the complete binary tree with depth $Q - 1$, which has $2^Q - 1$ nodes.

Lemma 2.22. *Let Q and m be integers. There exists a family of $2Q$ -wise independent hash functions $\mathcal{H} = \{h : \mathcal{T}_Q \rightarrow \{0, 1\}^m\}$ such that choosing a random function from $\mathcal{H}$ takes $2Q \max\{Q, m\}$ random bits, and evaluating a function from $\mathcal{H}$ takes time $\text{poly}(Q, m)$.*

Proof. Note that $|\mathcal{T}_Q| = 2^Q - 1$. We let $\sigma : \mathcal{T}_Q \rightarrow \{0, 1\}^Q$ be any efficiently computable injection (e.g., the injection obtained by viewing the path to a node of depth k as a bit string of length k , and e.g., appending to this the string 10^{Q-k-1}).

Let $\mathcal{H}_0 = \{h : \{0, 1\}^Q \rightarrow \{0, 1\}^m\}$ be a family of $2Q$ -wise independent hash function from [Theorem 2.21](#). Define

$$\mathcal{H} = \{h \circ \sigma : \mathcal{T}_Q \rightarrow \{0, 1\}^m \mid h \in \mathcal{H}_0\}.$$

To see $2Q$ -wise independence, for any distinct $x_1, \dots, x_{2Q} \in \mathcal{T}_Q$, let $z_i = \sigma(x_i)$. Since σ is injective, it holds that z_i are distinct and hence for all $v_1, v_2, \dots, v_{2Q} \in \{0, 1\}^m$

$$\Pr_{h \sim \mathcal{H}} [\forall i \in [2Q], h(x_i) = v_i] = \Pr_{h_0 \sim \mathcal{H}_0} [\forall i \in [2Q], h_0(z_i) = v_i] = \frac{1}{M^{2Q}}.$$

□

Lemma 2.23. *Let $\mathcal{H} = \{h : [N] \rightarrow [M]\}$ be a family of pairwise independent hash functions and let $S \subseteq [N]$ be an arbitrary subset. Then,*

$$\Pr_{h \sim \mathcal{H}, v \sim [M]} [\exists y \in S, h(y) = v] \geq \frac{|S|}{M} - \frac{|S|(|S| - 1)}{2M^2}.$$

2.7 The Chernoff-Hoeffding bound

We will need the following version of the Chernoff-Hoeffding bound.

Lemma 2.24. *If $X_1, \dots, X_\ell \in \{0, 1\}$ are independent Bernoulli random variables (not necessarily identical) with $\mu := \sum_{i=1}^\ell \mathbb{E}[X_i]$, then for any $m \geq 1$,*

$$\Pr \left[\left| \sum_{i=1}^\ell X_i - \mu \right| \geq \sqrt{m\ell} \right] \leq 2^{-\Omega(m)}.$$

3 A range-size upper bound protocol

In this section, we present a key technical result, showing how to upper bound the size of a given circuit's range in AM. To that end, we consider the following promise problem.

Definition 3.1. *For any $\varepsilon := \varepsilon(n) > 0$, the ε -GapRangeSize problem is a promise problem defined as follows: the input consists of a circuit $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$ and a size threshold $\tau > 0$.*

- $\Pi_{\text{YES}} : |\text{Im}(C)| \leq \tau$
- $\Pi_{\text{NO}} : |\text{Im}(C)| \geq \tau + \varepsilon 2^n$

We will build towards proving the following theorem.

Theorem 3.2. *ε -GapRangeSize is in $\text{prAM} \cap \text{prcoAM}$ for any $\varepsilon := \varepsilon(n) \geq 1/\text{poly}(n)$.*

3.1 The Goldwasser-Sipser protocol

We recall the celebrated Goldwasser-Sipser set-size lower bound protocol from [GS86]. This protocol is typically presented as a protocol in which Merlin can convince Arthur that a certain set S has size at least τ when we in fact have $|S| \geq (1 + \delta)\tau$, provided that membership in S can be recognized in NP. We need a slight generalization of this in which we replace NP-recognizable languages with AM-recognizable promise problems, as follows. The original proof in [GS86] immediately applies to this more general setting, but we include our own proof for completeness.

Definition 3.3. *Let $\Pi' = (\Pi'_{\text{YES}}, \Pi'_{\text{NO}})$ be any promise problem where $\Pi'_{\text{YES}}, \Pi'_{\text{NO}} \subseteq \{0, 1\}^* \times \{0, 1\}^*$. For any $\delta > 0$, the $(1 + \delta)$ -GapSize $_{\Pi'}$ problem, parameterized by the promise problem Π' is a promise problem defined as follows on input $x, 1^n$ and a size threshold τ .*

- $\Pi_{\text{YES}} : |\Pi'_{\text{YES},n}(x)| \geq (1 + \delta)\tau$
- $\Pi_{\text{NO}} : 2^n - |\Pi'_{\text{NO},n}(x)| \leq \tau$

where $\Pi'_{\text{YES},n}(x) := \{y \in \{0, 1\}^n : (x, y) \in \Pi'_{\text{YES}}\}$ and $\Pi'_{\text{NO},n}(x) := \{y \in \{0, 1\}^n : (x, y) \in \Pi'_{\text{NO}}\}$

Theorem 3.4 (The Goldwasser-Sipser protocol [GS86]). *$(1 + \delta)$ -GapSize $_{\Pi'}$ is in prAM for any $\Pi' \in \text{prAM}$ and any $\delta := \delta(n) \geq 1/\text{poly}(n)$.*

Proof. We assume without loss of generality that $\delta \leq 1/10$. We show a 4-round protocol achieving an additive $\text{poly}(\delta)$ gap between completeness and soundness. This is sufficient as the probability can be amplified and the number of rounds can be compressed to 2 due to [Theorems 2.16](#) and [2.17](#).

Since $\Pi' \in \text{prAM}$ by assumption, it follows from [Theorem 2.16](#) that $\Pi' \in \text{prAM}_{1,2^{-n}}$. (I.e., we can boost completeness to 1 and soundness to 2^{-n} .)

Now, on input x , 1^n , and τ , Arthur and Merlin behave as follows.

1. Arthur samples a pairwise independent hash function $h : \{0, 1\}^n \rightarrow [p]$ and a uniformly random $v \sim [p]$ and sends them to Merlin, where p is a prime in $[\frac{2}{\delta} \lceil (1 + \delta)\tau \rceil, \frac{4}{\delta} \lceil (1 + \delta)\tau \rceil]$.
2. Merlin replies with an element $y \in \{0, 1\}^n$ that is meant to satisfy that $y \in \Pi'_{\text{YES},n}(x)$ and $h(y) = v$.
3. Next, Arthur and Merlin perform the protocol for Π' guaranteed by the fact that $\Pi' \in \text{prAM}_{1,2^{-n}}$ on input (x, y) , which takes 2 rounds.
4. Finally, Arthur accepts if and only if both $h(y) = v$ and Arthur accepts in the subprotocol.

Completeness: Let $S := \Pi'_{\text{YES},n}(x)$, and suppose that the input is a YES instance so that $|S| \geq (1 + \delta)\tau$. Then, there exists an $S' \subseteq S$ such that $|S'| = \lceil (1 + \delta)\tau \rceil$. By [Lemma 2.23](#),

$$\begin{aligned} \Pr_{h,v}[\exists y \in S, h(y) = v] &\geq \Pr_{h,v}[\exists y \in S', h(y) = v] \\ &\geq |S'| \cdot \frac{1}{p} - \frac{|S'|(|S'| - 1)}{2} \cdot \frac{1}{p^2} \\ &\geq \varepsilon - \varepsilon^2, \end{aligned}$$

where $\varepsilon := \lceil (1 + \delta)\tau \rceil / p \approx \delta/2$.

The probability that the subprocedure rejects is at most 2^{-n} . Hence the overall completeness is at least $\varepsilon - \varepsilon^2 - 2^{-n}$.

Soundness: Let $S := \{0, 1\}^n \setminus \Pi'_{\text{NO},n}(x)$ and suppose that the input is a NO instance so that $|S| \leq \tau$. By a simple union bound, for any hash function h , we have

$$\Pr_v[\exists y \in S, h(y) = v] \leq \frac{|h(S)|}{p} \leq \frac{|S|}{p} \leq \frac{\varepsilon}{1 + \delta}.$$

So, except with probability at most $\varepsilon/(1 + \delta)$, Merlin will not be able to provide a $y \in S$ satisfying $h(y) = v$. In this case, either $h(y) \neq v$, in which case Arthur rejects, or $y \in \Pi'_{\text{NO},n}(x)$, in which case the subprocedure rejects with probability at least $1 - 2^{-n}$. Hence, the overall soundness (i.e. probability of incorrectly accepting) is at most $2^{-n} + \varepsilon/(1 + \delta)$.

By the above analysis, the gap between the completeness and soundness probabilities is at least

$$\begin{aligned} \varepsilon - \varepsilon^2 - \frac{\varepsilon}{1 + \delta} - 2^{-n+1} &= \varepsilon \left(1 - \varepsilon - \frac{1}{1 + \delta} \right) - 2^{-n+1} \\ &= \varepsilon \left(\frac{\delta}{1 + \delta} - \varepsilon \right) - 2^{-n+1} \\ &\geq \frac{\varepsilon^2}{10}. \end{aligned}$$

The result follows by noting that, e.g., $\varepsilon \geq \delta/4$ (since there is always a prime between x and $2x$). $\square$

3.2 An approximation of the range size

We now show a convenient approximation of the range size of a circuit.

For a circuit $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$ and $r \in \mathbb{R}$, let $s_C(r) := |\{y \in \{0, 1\}^m : |C^{-1}(y)| \geq r\}|$ be the number of elements in the range with at least r preimages. Notice that though the more natural domain for s_C is integers between 1 and 2^n , we allow for arbitrary real number inputs r , which will be convenient. Notice also that $s_C(1) = |\text{Im}(C)|$ is the size of the image of C , the quantity that we would like to upper bound.

In particular, instead of upper bounding $s_C(1)$, we try to lower bound $2^n - s_C(1)$ using the following simple lemma.

Lemma 3.5. *For any circuit $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$,*

$$2^n - s_C(1) = \int_1^{2^n} s_C(r) dr .$$

Proof. Notice that by double counting,

$$\begin{aligned} 2^n &= \sum_{k=1}^{2^n} k |\{y \in \{0, 1\}^m : |C^{-1}(y)| = k\}| \\ &= \sum_{j=1}^{2^n} |\{y \in \{0, 1\}^m : |C^{-1}(y)| \geq j\}| \\ &= s_C(1) + \int_1^{2^n} s_C(r) dr , \end{aligned}$$

as needed. □

It is then natural to define $T := 2^n - s_C(1) = \int_1^{2^n} s_C(r) dr$. In order to approximate T , we further define two (easier to compute) quantities. In particular, for a fixed integer $\ell \geq 1$, we define

$$T_{\text{odd}} := (1 - 2^{-1/\ell}) \sum_{i=1}^{\ell n} 2^{(2i-1)/(2\ell)} s_C(2^{(2i-1)/(2\ell)}) \quad T_{\text{even}} := (1 - 2^{-1/\ell}) \sum_{i=1}^{\ell n} 2^{(2i)/(2\ell)} s_C(2^{(2i)/(2\ell)}) .$$

Notice that these can be thought of as approximations to the integral defining T . These are in fact good approximations of T , as captured in the following lemma.

Lemma 3.6. *The following inequalities hold.*

- $T_{\text{odd}} \geq 2^{n-1/\ell} - s_C(1)$.
- $T_{\text{even}} \leq T$.

Proof. Since $s_C(r)$ is non-increasing in r , we see that for any i ,

$$\begin{aligned} \int_{2^{(2i-1)/(2\ell)}}^{2^{(2i+1)/(2\ell)}} s_C(r) dr &\leq (1 - 2^{-1/\ell}) 2^{(2i+1)/(2\ell)} s_C(2^{(2i-1)/(2\ell)}) , \\ \int_{2^{(i-1)/\ell}}^{2^{i/\ell}} s_C(r) dr &\geq (1 - 2^{-1/\ell}) 2^{i/\ell} s_C(2^{i/\ell}) . \end{aligned} \tag{1}$$

For the first inequality, we know

$$\begin{aligned}
T &= \int_1^{2^n} s_C(r) dr \\
&= \int_1^{2^{1/(2\ell)}} s_C(r) dr + \int_{2^{1/(2\ell)}}^{2^n} s_C(r) dr \\
&\leq \int_1^{2^{1/(2\ell)}} s_C(r) dr + (1 - 2^{-1/\ell}) \sum_{i=1}^{\ell n} 2^{(2i+1)/(2\ell)} s_C(2^{(2i-1)/(2\ell)}) \\
&\leq (2^{1/(2\ell)} - 1) s_C(1) + 2^{1/\ell} (1 - 2^{-1/\ell}) \sum_{i=1}^{\ell n} 2^{(2i-1)/(2\ell)} s_C(2^{(2i-1)/(2\ell)}) \\
&= (2^{1/(2\ell)} - 1) s_C(1) + 2^{1/\ell} T_{\text{odd}} .
\end{aligned}$$

Rearranging, we have

$$T_{\text{odd}} \geq 2^{n-1/\ell} - 2^{-1/(2\ell)} s_C(1) > 2^{n-1/\ell} - s_C(1) .$$

For the second inequality, summing Equation (1) from $i = 1$ to ℓn , we see that

$$T = \sum_{i=1}^{\ell n} \int_{2^{(i-1)/\ell}}^{2^{i/\ell}} s_C(r) dr \geq \sum_{i=1}^{\ell n} (1 - 2^{-1/\ell}) 2^{i/\ell} s_C(2^{i/\ell}) = T_{\text{even}} ,$$

as needed. $\square$

Intuitively, if we have good lower bounds for $s_C(2^{j/(2\ell)})$, we can convert them into a good lower bound for T and hence a good upper bound for $|\text{Im}(C)|$. We now show how to lower bound $s_C(2^{j/(2\ell)})$ by “applying Theorem 3.4 twice.”

Lemma 3.7. *For any circuit $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$, integer $1 \leq \ell \leq \text{poly}(n)$, $1 \leq i \leq \ell n$ and a threshold value $\tau > 0$, the following promise problem Π is in prAM :*

- $\Pi_{\text{YES}} : s_C(2^{i/\ell}) \geq \tau$
- $\Pi_{\text{NO}} : s_C(2^{(i-1)/\ell}) \leq 2^{-1/\ell} \cdot \tau$

Proof. Consider the following promise problem $\Pi' = (\Pi'_{\text{YES}}, \Pi'_{\text{NO}})$:

- $\Pi'_{\text{YES}} : \{(C, y) : |C^{-1}(y)| \geq 2^{i/\ell}\}$
- $\Pi'_{\text{NO}} : \{(C, y) : |C^{-1}(y)| \leq 2^{(i-1)/\ell}\}$

Notice that for any circuit C and $y \in \{0, 1\}^m$, the set $C^{-1}(y) = \{x \in \{0, 1\}^n : C(x) = y\}$ can be decided in P (by simply checking if $C(x) = y$). More specifically, the promise problem (in fact a language) $(C^{-1}(y), \overline{C^{-1}(y)})$ is contained in $\text{P} \subseteq \text{prAM}$. It follows from Theorem 3.4 that Π' is in prAM .

Since Π' is in prAM , again by Theorem 3.4 we know that Π is also in prAM . In particular, by definition we have $s_C(2^{i/\ell}) = |\Pi'_{\text{YES}, m}(C)|$ and $s_C(2^{(i-1)/\ell}) \geq 2^m - |\Pi'_{\text{NO}, m}(C)|$, following the notations of Theorem 3.4. $\square$

3.3 An AM protocol for upper bounding range size

We are now ready to show that upper bounding the range size is contained in AM.

Proof of Theorem 3.2. The fact that GapRangeSize is in prcoAM is immediate from Theorem 3.4. So, we only need to present an AM protocol for GapRangeSize. To that end, we present a 3-round AM protocol for GapRangeSize as follows. One can then compress the number of rounds to 2 via Theorem 2.17.

Set $\ell := \lceil 2/\log(1 + \varepsilon) \rceil$. For each $i \in [1, \ell n]$, Merlin sends a value $s_i \in \{0, 1, \dots, 2^n\}$, claiming that $s_C(2^{(2i-1)/(2\ell)}) = s_i$. Arthur then runs the protocol from Lemma 3.7 (with parameters $\tau' = s_i$, $\ell' = 2\ell$ and $i' = 2i - 1$) for each $i \in [1, \ell n]$ where $s_i \neq 0$ in parallel (note that we would use a protocol in prAM_{1,2⁻ⁿ} guaranteed by Theorem 2.16). Arthur rejects if any of these (at most) ℓn protocols reject.

Arthur then proceeds by computing

$$\tilde{T} := (1 - 2^{-1/\ell}) \sum_{i=1}^{\ell n} 2^{(2i-1)/(2\ell)} s_i .$$

He accepts if and only if $\tilde{T} \geq 2^{n-1/\ell} - \tau$.

Completeness: Suppose that Merlin is honest and $|\text{Im}(C)| \leq \tau$. In particular, $s_i = s_C(2^{(2i-1)/(2\ell)})$ for all $i \in [1, \ell n]$ and all executions of Lemma 3.7 would accept. And we have from Lemma 3.6 that

$$\tilde{T} = T_{\text{odd}} \geq 2^{n-1/\ell} - \tau .$$

Soundness: Suppose that Arthur accepts at the end of the protocol.

By a union bound over all polynomially many executions of the protocol from Lemma 3.7, we know that for each $i \in [1, \ell n]$, $s_C(2^{(2i-2)/(2\ell)}) \geq 2^{-1/(2\ell)} s_i$. Note that this trivially holds when $s_i = 0$. We have

$$\begin{aligned} 2^{1/\ell} \tilde{T} &= (1 - 2^{-1/\ell}) \sum_{i=1}^{\ell n} 2^{(2i-1)/(2\ell)} \cdot 2^{1/\ell} \cdot s_i \\ &\leq (1 - 2^{-1/\ell}) \sum_{i=1}^{\ell n} 2^{(2i)/(2\ell)} \cdot 2^{1/\ell} \cdot s_C(2^{(2i-2)/(2\ell)}) \\ &\leq (1 - 2^{-1/\ell}) \sum_{i=1}^{\ell n} 2^{(2i)/(2\ell)} \cdot 2^{2/\ell} \cdot s_C(2^{(2i)/(2\ell)}) + (1 - 2^{-1/\ell}) 2^{2/\ell} s_C(1) \\ &= 2^{2/\ell} T_{\text{even}} + (2^{2/\ell} - 2^{1/\ell}) s_C(1) \\ &\leq 2^{2/\ell} (T_{\text{even}} + s_C(1)) - 2^{1/\ell} s_C(1) \\ &\leq 2^{2/\ell} \cdot 2^n - 2^{1/\ell} s_C(1) , \end{aligned}$$

where the last inequality follows from Lemma 3.6 and the fact that $T + s_C(1) = 2^n$.

Combining the above with the fact that Arthur accepts so that $\tilde{T} \geq 2^{n-1/\ell} - \tau$, we see that $2^n - 2^{1/\ell} \tau \leq 2^{2/\ell} \cdot 2^n - 2^{1/\ell} s_C(1)$, which after rearranging gives

$$s_C(1) \leq \tau + \varepsilon \cdot 2^{n-1/\ell} < \tau + \varepsilon \cdot 2^n ,$$

as needed. □

4 An “AM protocol for Avoid”

Combining the high-level outline from [Section 1.2.1](#) with the tools from [Section 3](#), we now show an AMA protocol for Avoid that succeeds with probability $1 - 1/\text{poly}(n)$.

Theorem 4.1. *For any $s(n) \geq 1/\text{poly}(n)$, there is a $\text{SearchAMA}_{1-2^{-n}, s(n)}$ algorithm for Avoid on circuits $C : \{0, 1\}^n \rightarrow \{0, 1\}^{n+1}$.*

Proof. Let $\varepsilon = (s - 2^{-n})/32$. By [Theorem 2.16](#), we may assume that the AM protocol from [Theorem 3.2](#) has two rounds, completeness 1, and soundness error 2^{-2n} .

The SearchAMA protocol consists of two parallel parts:

1. For $i \in [\lceil 1/\varepsilon \rceil]$, set $\tau_i = i\varepsilon 2^n$. Arthur runs the protocol from [Theorem 3.2](#) with parameters $(\varepsilon' = \varepsilon, \tau' = \tau_i)$ for all such i in parallel.
2. Arthur samples $y_1, \dots, y_T$ uniformly from $\{0, 1\}^{n+1}$ and sends them to Merlin for $T = \lceil 32n/\varepsilon^2 \rceil$. Merlin is expected to send to Arthur $x_1, \dots, x_T$ where $C(x_i) = y_i$ for any $y_i \in \text{Im}(C)$ (and arbitrary x_i for $y_i \notin \text{Im}(C)$).

Next, we specify Arthur’s behavior upon receiving Merlin’s response from the two parallel parts.

Let $j \in [\lceil 1/\varepsilon \rceil]$ be the smallest index such that Arthur accepts the range-size upper bound protocol.

Let $t := |\{i : C(x_i) = y_i\}|$. Arthur rejects if $t < (j-1)\varepsilon T/2 - \sqrt{nT}$. Otherwise, Arthur outputs a uniformly sampled string from $\{y_i : C(x_i) \neq y_i\}$.

Next, we proceed to analyze the correctness of the protocol. Let Y_i be a random variable such that $Y_i = 1$ if and only if $y_i \in \text{Im}(C)$. Let $\alpha := \frac{|\text{Im}(C)|}{2^{n+1}}$. Let $Y = \sum_{i=1}^T Y_i$. By the Chernoff-Hoeffding bound, we have

$$\Pr_Y[|Y - \alpha T| \geq \sqrt{nT}] \leq 2 \exp\left(-\frac{2nT}{T}\right) = 2e^{-2n}.$$

Completeness: Let k be the smallest index such that $|\text{Im}(C)| \leq \tau_k$. We have $j \leq k$ and hence $\tau_{j-1} \leq \tau_{k-1} < |\text{Im}(C)|$. Therefore, $\alpha T > (j-1)\varepsilon T/2$ and $\Pr[Y < (j-1)\varepsilon T/2 - \sqrt{nT}] \leq 2e^{-2n}$. Therefore, the completeness is at least $1 - 2e^{-2n} \geq 1 - 2^{-n}$, as claimed.

Soundness: By soundness of [Theorem 3.2](#), we know that $|\text{Im}(C)| < \tau_{j+1}$ except with probability $\lceil 1/\varepsilon \rceil 2^{-2n}$. In particular, with probability $(1 - 2e^{-2n})$ we have $Y \leq (j+1)\varepsilon T/2 + \sqrt{nT}$. We also have that $t \geq (j-1)\varepsilon T/2 - \sqrt{nT}$. Moreover, $T - t \geq T - Y \geq T - \alpha T - \sqrt{nT} \geq T/2 - \sqrt{nT}$.

The probability that Arthur outputs a string in the image of C is at most:

$$\frac{Y - t}{T - t} \leq \frac{\varepsilon T + 2\sqrt{nT}}{T/2 - \sqrt{nT}} \leq 10\varepsilon.$$

By union bound, the overall soundness error is at most $\lceil 1/\varepsilon \rceil 2^{-2n} + 2e^{-2n} + 10\varepsilon < s$. $\square$

5 Any decision problem that reduces to Avoid is in $\text{AM} \cap \text{coAM}$

The following rather technical proposition is the main tool that we will use to prove that $\text{BPP}^{\text{Avoid}} \subseteq \text{AM} \cap \text{coAM}$ (in fact, that $\text{BPP}^{\text{RAvoid}} \subseteq \text{AM} \cap \text{coAM}$ and $\text{prBPP}^{\text{RAvoid}} \subseteq \text{prAM} \cap \text{prcoAM}$). To interpret the proposition, one should consider the case when $\mathcal{B}^{\mathcal{O}}$ is very likely to output the bit b on input z , e.g., $p_b(z) \geq 1 - 2^{-|z|}$. Then, the protocol guaranteed by the proposition satisfies that (1) if Merlin is honest, then with high probability Arthur either outputs b or outputs $?$ (i.e., says “I don’t know”); and (2) *no matter how Merlin behaves* (including, of course, if Merlin is honest), Arthur has a not-too-small probability of roughly $1/Q = 1/\text{poly}(|z|)$ of outputting b (or $\perp$, if Merlin misbehaves).

If Arthur and Merlin repeat this protocol $\text{poly}(|z|) \gg Q$ times in parallel (with independent randomness), then (1) if Merlin is honest, then with high probability all of the subprotocols’ outputs will be either b or $?$ (so, never $1 - b$); and (2) no matter what Merlin does, many of the subprotocols’ outputs will be b (or $\perp$ if Merlin misbehaves). So, suppose Arthur outputs $\perp$ if some subprotocols output 0 and others output 1 (or in the very low probability event that all output $?$). Otherwise, we output the unique bit b that we see. Notice that honest Merlin will cause Arthur to output the correct answer with high probability, and dishonest Merlin will either cause Arthur to output $\perp$ or the correct answer with high probability, which is what we want.

Proposition 5.1. *For any randomized polynomial-time oracle algorithm $\mathcal{B}^{\mathcal{O}}$ making at most $Q := Q(n) \leq \text{poly}(n)$ queries to its oracle (where n is the input length) and any positive $\varepsilon := \varepsilon(n) \geq 1/\text{poly}(n)$, there exists a four-round protocol between a polynomial-time interactive algorithm Arthur $\mathcal{A}$ and a (computationally unbounded) interactive algorithm Merlin $\mathcal{M}$ with the following properties.*

1. **(Syntax.)** *Merlin and Arthur take as input a bit string $z \in \{0, 1\}^*$ and exchange at most $\text{poly}(|z|)$ bits, after which Arthur’s output $\langle \mathcal{A}, \mathcal{M} \rangle(z)$ is either 0, 1, ?, or $\perp$. (We think of “?” as “I don’t know the answer” and $\perp$ as “Merlin seems to be lying.”)*
2. **(Correctness.)** *For any input string $z \in \{0, 1\}^N$ and bit $b \in \{0, 1\}$, we have*

$$\Pr[\langle \mathcal{A}, \mathcal{M} \rangle(z) \in \{b, ?\}] \geq 1/2 + p_b(z)/2 - \varepsilon.$$

3. **(Soundness.)** *For any input string $z \in \{0, 1\}^N$ and bit $b \in \{0, 1\}$, and any (possibly malicious) $\mathcal{M}^\dagger$,*

$$\Pr[\langle \mathcal{A}, \mathcal{M}^\dagger \rangle(z) \in \{b, \perp\}] \geq p_b(z)/2 - \frac{Q}{2(Q+1)} - \varepsilon.$$

Here,

$$p_b(z) := \Pr_{\mathcal{O} \sim \text{RAvoid}}[\mathcal{B}^{\mathcal{O}}(z) = b]$$

is the probability that $\mathcal{B}^{\mathcal{O}}$ outputs b on input z , where the randomness is over the random coins of $\mathcal{B}$ and the oracle $\mathcal{O} \sim \text{RAvoid}$.

Before proving the proposition, we show that it implies the main result of this section, as follows.

Theorem 5.2. $\text{BPP}^{\text{RAvoid}} \subseteq \text{AM} \cap \text{coAM}$ and $\text{prBPP}^{\text{RAvoid}} \subseteq \text{prAM} \cap \text{prcoAM}$. *In other words, any (promise) decision problem that efficiently reduces to Avoid (even via randomized reductions that make many adaptive queries and only work with a uniformly random oracle $\mathcal{O} \sim \text{RAvoid}$) is in (promise) $\text{AM} \cap \text{coAM}$.*

In particular, Avoid is not NP-hard (even under randomized reductions that make many adaptive oracle queries) unless the polynomial hierarchy collapses to AM.

Proof. Let $(\mathcal{B}')^\mathcal{O}$ be a polynomial-time randomized oracle algorithm and for any input $z \in \{0, 1\}^*$ and output bit $b \in \{0, 1\}$, let

$$p'_b(z) := \Pr_{\mathcal{O} \sim \text{RAvoid}} [(\mathcal{B}')^\mathcal{O}(z) = b] .$$

Let

$$\Pi_b := \{z \in \{0, 1\}^* : p'_b(z) \geq 2/3\} .$$

We wish to show an Arthur-Merlin protocol for (Π_0, Π_1) and another Arthur-Merlin protocol for (Π_1, Π_0) .

We present a four-round Arthur-Merlin protocol in which Arthur outputs either 0, 1, or $\perp$ with the following correctness and soundness properties. First, if Merlin is honest, then for any $b \in \{0, 1\}$ and any $z \in \Pi_b$, we have that Arthur outputs b with probability at least $2/3$. Second, regardless of Merlin's behavior, on input $z \in \Pi_b$, Arthur either outputs b or $\perp$ with probability at least $2/3$. Notice that this immediately implies that the problem (Π_0, Π_1) accepted by $(\mathcal{B}')^\mathcal{O}$ and its complement (Π_1, Π_0) have four-round Arthur-Merlin protocols. By [Theorem 2.17](#), these can be converted into two-round protocols, and we immediately see that the resulting language (or promise problem) is in $\text{AM} \cap \text{coAM}$ (or $\text{prAM} \cap \text{prcoAM}$).

To that end, by [Lemma 2.6](#), there exists a polynomial-time randomized oracle algorithm $\mathcal{B}^\mathcal{O}$ such that $p_b(z) \geq 1 - 2^{-|z|}$ for any $z \in \Pi_b$, where

$$p_b(z) := \Pr_{\mathcal{O} \sim \text{RAvoid}} [\mathcal{B}^\mathcal{O}(z) = b] .$$

(In other words, $\mathcal{B}^\mathcal{O}(z)$ is very likely to accept if $(\mathcal{B}')^\mathcal{O}(z)$ has probability at least $2/3$ of accepting, and similarly for rejection.)

Let $Q := Q(n) \leq \text{poly}(n)$ be a bound on the number of queries that $\mathcal{B}^\mathcal{O}$ makes to its oracle on input length n , let $\varepsilon := \varepsilon(n) := 1/(100nQ)$, and let $\ell := \ell(n) := 10nQ$. We may assume that the input length n is large enough so that, say, $1 - 2^{-n} - Q/(Q+1) - 2\varepsilon \geq 20/\ell$ and $2\ell 2^{-n} + \ell\varepsilon \leq 1/5$.

Then, on input $z \in \{0, 1\}^n$, Arthur and Merlin together run ℓ copies of the protocol from [Proposition 5.1](#) in parallel. Let $S \subseteq \{0, 1, ?, \perp\}$ be the set of all outputs from Arthur in the subprotocols. If $S = \{b\}$ or $S = \{b, ?\}$ for some bit $b \in \{0, 1\}$, then Arthur outputs b . Otherwise, Arthur outputs $\perp$. (In other words, Arthur outputs $\perp$ if (1) any subprotocol outputs $\perp$, (2) some subprotocols output 0 but others output 1, or (3) all subprotocols output $?$. Otherwise, Arthur outputs the unique bit b returned by every subprotocol that returned a bit.)

Clearly Arthur runs in polynomial time, and Arthur and Merlin exchange at most $\text{poly}(|z|)$ bits. To prove correctness and soundness, let $z \in \Pi_b$ for some bit $b \in \{0, 1\}$, which in particular means that $p_b(z) \geq 1 - 2^{-n}$.

Then, notice that by the soundness of the protocol from [Proposition 5.1](#), we have that

$$\Pr[b, \perp \notin S] \leq \left(1 - p_b(z)/2 + \frac{Q}{2(Q+1)} + \varepsilon\right)^\ell \leq e^{-\ell(p_b(z) - Q/(Q+1) - 2\varepsilon)/2} \leq e^{-\ell(1 - 2^{-n} - Q/(Q+1) - 2\varepsilon)/2} \leq e^{-10} .$$

So, regardless of the behavior of Merlin, we see that with good probability S will contain either b or $\perp$. This immediately implies the soundness of the larger protocol (since if S contains either b or $\perp$, then the output of Arthur must be either b or $\perp$).

To prove correctness of the larger protocol, notice that by the correctness property in [Proposition 5.1](#), we have that if Merlin is honest then

$$\Pr[1 - b \in S \text{ or } \perp \in S] \leq 1 - (1/2 + p_b(z)/2 - \varepsilon)^\ell \leq 1 - (1 - 2^{-n} - \varepsilon)^\ell \leq \ell 2^{-n} + \ell\varepsilon \leq 1/5 .$$

Combining this with the above, we see that when Merlin is honest, then with probability at least $1 - e^{-10} - 1/5 \geq 2/3$, we will simultaneously have that either $b \in S$ or $\perp \in S$ and that $1 - b, \perp \notin S$. Therefore, either $S = \{b, ?\}$ or $S = \{b\}$. Either way, Arthur outputs the correct answer in this case. So, correctness holds. $\square$

5.1 Proof of Proposition 5.1 by a tree of computation

Before proving Proposition 5.1, we make formal the ideas discussed in Section 1.2.3. To that end, we let $\mathcal{A}^\mathcal{O}$ be a deterministic algorithm that takes no input and outputs a single bit $b \in \{0, 1\}$. We assume that $\mathcal{A}^\mathcal{O}$ makes exactly $Q \geq 1$ oracle queries, regardless of the oracle responses, where each oracle query takes the form of a circuit $C : \{0, 1\}^N \rightarrow \{0, 1\}^{N+1}$ for some fixed N , and we think of these queries as instances of Avoid. In particular, the output of $\mathcal{A}^\mathcal{O}$ and all of its oracle queries are uniquely determined by the list $y_1, \dots, y_Q \in \{0, 1\}^{N+1}$. We also assume that the queries made by $\mathcal{A}^\mathcal{O}$ are distinct. (Of course, Proposition 5.1 works with algorithms $\mathcal{B}^\mathcal{O}$ that might be randomized, certainly take some input, and might make a different number of oracle queries depending on the input, the randomness, and the query responses. When we prove Proposition 5.1, we will hard-code an input $z \in \{0, 1\}^n$ and the random coins r into a machine $\mathcal{A}^\mathcal{O} := \mathcal{B}_{z,r}^\mathcal{O}$, and we will use padding arguments to fix the number of oracle queries Q and the input length N of the queried circuits.)

Let $\mathcal{T}_Q$ be the set of nodes of the complete binary tree with height $Q - 1$, and let $Y : \mathcal{T}_Q \rightarrow \{0, 1\}^{N+1}$ be a labeling of the vertices of $\mathcal{T}_Q$. Recall that we can interpret any path $v_1, \dots, v_k$ in this tree starting at the root v_1 as a partial computation of $\mathcal{A}^\mathcal{O}$, where the first k oracle query responses are $Y(v_1), \dots, Y(v_k)$. This in particular means that for every vertex $v \in \mathcal{T}_Q$, there is an associated query circuit

$$C_v := C_v(\mathcal{A}, Y) ,$$

which is simply the circuit queried by $\mathcal{A}^\mathcal{O}$ after receiving $Y(v_1), \dots, Y(v_{k-1})$ as responses to its first $k - 1$ oracle queries, where $v_1, \dots, v_{k-1}, v_k := v$ is the (unique) path from the root v_1 to $v = v_k$. For each leaf node $\ell \in \mathcal{T}_Q$, we can also assign a bit

$$b_\ell := b_\ell(\mathcal{A}, Y) ,$$

which corresponds to the output of $\mathcal{A}^\mathcal{O}$ when its oracle query responses are set to $Y(v_1), \dots, Y(v_Q)$, where $v_1, \dots, v_{Q-1}, v_Q := \ell$ is the path from the root node v_1 to the leaf $\ell = v_Q$.

The key technical definition that we will need is a certain *labeled subtree* $\mathcal{T}^*$ of $\mathcal{T}_Q$ such that all paths $v_1, \dots, v_k$ in $\mathcal{T}^*$ correspond to computations made by $\mathcal{A}^\mathcal{O}$ in which the oracle query responses $Y(v_i)$ are valid solutions to the corresponding Avoid instances C_{v_i} (i.e., $Y(v_i) \notin \text{Im}(C_{v_i})$).

If we were guaranteed that all query circuits $C : \{0, 1\}^N \rightarrow \{0, 1\}^{N+1}$ were injective, then as we discussed in Section 1.2.3, we could simply take $\mathcal{T}^*$ to be the subtree consisting of all such paths, i.e., the subtree consisting of all paths $v_1, \dots, v_k$ where $Y(v_i) \notin \text{Im}(C_{v_i})$ for all $1 \leq i \leq k$. Indeed, if all circuits were guaranteed to be injective and Y is chosen randomly from a suitable family of functions, then the resulting subtree $\mathcal{T}^*$ would have the property that a node $v \in \mathcal{T}_Q$ with depth d is contained in $\mathcal{T}^*$ with probability exactly 2^{-d-1} , which makes the expected number of nodes $\sum_{d=0}^{Q-1} 2^d \cdot 2^{-d-1} = Q/2$. However, without this guarantee, we run into two additional issues: (1) if the circuits C_v have significantly smaller images, then the resulting tree $\mathcal{T}^*$ will tend to be far too large; and (2) if the image size of the circuit C_v can depend on $Y(v')$ for v' an ancestor of v , then the oracle query responses $Y(v_1), \dots, Y(v_Q)$ that lead to leaves in $\mathcal{T}^*$ will not be uniformly

random solutions to the corresponding Avoid instances C_{v_i} . (E.g., if y happens to be an oracle query response that leads to future queries C_{v_j} with small images, then a path leading to a leaf in $\mathcal{T}^*$ will be more likely to contain the label $Y(v_i) = y$.)

For these reasons, we must do some additional random pruning of our tree in order to guarantee that each node appears with the correct probability, as we discussed in [Section 1.2.3](#). (In the proof, we will also need to control the probability of the event that *two* nodes v and v' appear in $\mathcal{T}^*$.) To that end, notice that it suffices to place a node v in $\mathcal{T}^*$ if (1) its parent is in $\mathcal{T}^*$ (or v is the root node of $\mathcal{T}_Q$); (2) $Y(v) \notin \text{Im}(C_v)$; and (3) a biased coin flip comes up heads, where the probability of heads is $P_v := 2^N / (2^{N+1} - |\text{Im}(C_v)|)$ is chosen so that

$$P_v \cdot \left(1 - \frac{|\text{Im}(C_v)|}{2^{N+1}}\right) = 1/2.$$

So, the definition of our tree $\mathcal{T}^*$ will include an additional function $P : \mathcal{T}_Q \rightarrow [0, 1]$ and another function $U : \mathcal{T}_Q \rightarrow [0, 1]$, which we think of as representing the randomness used to flip this biased coin. (In much of our analysis, we will set $P(v) := 2^N / (2^{N+1} - |\text{Im}(C_v)|)$ as above. But, we must in general consider arbitrary functions $P : \mathcal{T}_Q \rightarrow [0, 1]$ because Merlin will be providing P .)

Our definition is then as follows. Given an oracle algorithm $\mathcal{A}^O$ as above and labeling functions $Y : \mathcal{T}_Q \rightarrow \{0, 1\}^{N+1}$, $P : \mathcal{T}_Q \rightarrow [0, 1]$, and $U : \mathcal{T}_Q \rightarrow [0, 1]$, we define

$$\mathcal{T}^* := \mathcal{T}^*(\mathcal{A}, Y, U, P) \subseteq \mathcal{T}_Q$$

as the subtree of $\mathcal{T}_Q$ defined recursively as follows. A vertex $v \in \mathcal{T}_Q$ is in $\mathcal{T}^*$ if and only if (1) the parent of v is in $\mathcal{T}^*$ or v is the root node of $\mathcal{T}_Q$; (2) $Y(v) \notin \text{Im}(C_v)$; and (3) $U(v) \leq P(v)$. Notice that this merely makes formal the informal description from above. (The reader might wish to focus mostly on the special case in which the circuits are injective and $P(v) = 1$ for all v . In that case, the third condition is vacuous, and U is irrelevant.)

It is also convenient to define

$$\mathcal{L}^* := \mathcal{L}^*(\mathcal{A}, Y, U, P)$$

as the set of all leaves $\ell \in \mathcal{T}_Q$ that are also in $\mathcal{T}^*$. In other words, $\mathcal{L}^*$ is the set of nodes in $\mathcal{T}^*$ with depth $Q - 1$. In particular, as we discussed above, if $\ell \in \mathcal{L}^*$, then we can naturally assign an output bit $b_\ell := b_\ell(\mathcal{A}, Y) \in \{0, 1\}$ to ℓ . Specifically, b_ℓ is the output of $\mathcal{A}^O$ when its oracle query responses are set to $Y(v_1), \dots, Y(v_Q)$, where $v_1, \dots, v_Q := \ell$ is the path from the root v_1 to $\ell = v_Q$.

We note that both $\mathcal{T}^*$ and $\mathcal{L}^*$ depend only on the values of $P(v)$ when either v is the root of $\mathcal{T}_Q$ or v 's parent is in $\mathcal{T}^*$. We will therefore sometimes abuse notation and only specify the value of $P(v)$ for such choices of v .

Finally, we write $P_{\mathcal{A}, Y}$ for the “honest” choice of probabilities

$$P_{\mathcal{A}, Y}(v) = \frac{2^N}{2^{N+1} - |\text{Im}(C_v)|}.$$

Notice that these are exactly the acceptance probabilities in the additional pruning process mentioned in the informal description above. (Notice also that $P_{\mathcal{A}, Y}$ depends on Y .)

In our protocol, Arthur will sample Y and U from suitable families of independent hash functions and send them to Merlin. Merlin will respond by sending Arthur a purported tree $\mathcal{T}^*$, and for each child $v \in \mathcal{T}_Q$ of a vertex in $\mathcal{T}^*$ such that $v \notin \mathcal{T}^*$ (or the root node v , if $\mathcal{T}^*$ is empty), Merlin either

sends x_v such that $C_v(x_v) = Y(v)$ or $P_v < U(v)$. We think of these as certificates explaining why $v \notin \mathcal{T}^*$. In the case when Merlin provides P_v , Merlin will have to prove to Arthur that P_v is not much smaller than $P_{\mathcal{A},Y}(v)$ using the range-size upper bound protocol from [Section 3](#).

The following key lemma contains the crucial technical facts about this process that we will use to prove [Proposition 5.1](#). Notice in particular that [Item 1](#) tells us that $\mathcal{T}^*$ will not be too large (which will make our protocol efficient); and [Item 4](#) provides a formal sense in which the output bits b_ℓ of any leaves ℓ in $\mathcal{L}^*$ are a good proxy for the random variable $\mathcal{A}^\mathcal{O}$ when $\mathcal{O} \sim \text{RAvoid}$.

Lemma 5.3. *With the notation described above, suppose that $Y : \mathcal{T}_Q \rightarrow \{0, 1\}^{N+1}$ is sampled from some family of $2Q$ -wise-independent functions and $U : \mathcal{T}_Q \rightarrow [0, 1]$ is similarly sampled independently from a family of $2Q$ -wise-independent hash functions. Let $P := P_{\mathcal{A},Y}$ be the honest choice of probabilities, $\mathcal{T}^* := \mathcal{T}^*(\mathcal{A}, Y, U, P)$, and $\mathcal{L}^* := \mathcal{L}^*(\mathcal{A}, Y, U, P)$. Then, the following all hold.*

1. For any $K > 0$ we have

$$\Pr_{Y,U}[|\mathcal{T}^*| \geq K] \leq \frac{Q}{2K}.$$

2.

$$\Pr_{Y,U}[\mathcal{L}^* = \emptyset] \leq \frac{Q}{Q+1}.$$

3. For any leaf $\ell \in \mathcal{T}_Q$ and bit $b \in \{0, 1\}$,

$$\Pr_{Y,U}[b_\ell = b \text{ and } \ell \in \mathcal{L}^*] = 2^{-Q} \Pr_{\mathcal{O} \sim \text{RAvoid}}[\mathcal{A}^\mathcal{O} = b].$$

4. For any bit $b \in \{0, 1\}$,

$$\frac{1}{2} \cdot \Pr_{\mathcal{O} \sim \text{RAvoid}}[\mathcal{A}^\mathcal{O} = b] - \frac{Q-1}{2(Q+1)} \leq \Pr_{Y,U}[\exists \ell \in \mathcal{L}^* : b_\ell = b] \leq \frac{1}{2} \cdot \Pr_{\mathcal{O} \sim \text{RAvoid}}[\mathcal{A}^\mathcal{O} = b].$$

As the proof of [Lemma 5.3](#) is rather technical, we defer it to [Appendix A](#). At a high level, [Item 1](#) is proven via the first moment method, and [Item 2](#) is proven via the second moment method. Indeed, the proofs of [Items 1](#) and [2](#) are analogous to similar well-known proofs for the closely related Galton-Watson process [\[WG75\]](#). Our proof of [Item 3](#) amounts to carefully showing that conditioned on some leaf v_Q being in the set $\mathcal{L}^*$, the labels $Y(v_1), \dots, Y(v_Q)$ are distributed exactly as the responses made by $\mathcal{O} \sim \text{RAvoid}$ to the corresponding oracle queries $C_{v_1}, \dots, C_{v_Q}$. Finally, [Item 4](#) is a simple corollary of [Items 2](#) and [3](#).

Finally, before we can prove [Proposition 5.1](#), we will need the following two technical lemmas, which intuitively capture the notion that the tree $\mathcal{T}^*$ is unlikely to change significantly if one slightly perturbs U or P . These lemmas will be necessary for our proof because in our actual protocol (1) we will necessarily have to work with a discretized function $U : \mathcal{T}_Q \rightarrow \{0, 1/D, \dots, (D-1)/D\}$ for some very large integer D , rather than a function whose range is the continuous interval $[0, 1]$; and (2) Merlin might not provide us with exactly the right value of $P_{\mathcal{A},Y}$ but might instead provide us with a good approximation of $P_{\mathcal{A},Y}$. (Merlin will then use the range-size upper bound protocol to convince us that it is in fact a good approximation.) We abuse notation and write $P - P' \leq \varepsilon$ if $P(v) - P'(v) \leq \varepsilon$ for all v and $\|U - U'\|_\infty \leq \varepsilon$ if $|U(v) - U'(v)| \leq \varepsilon$ for all v .

Lemma 5.4. *With the notation described above, for any (fixed) $Y : \mathcal{T}_Q \rightarrow \{0, 1\}^{N+1}$ and $P := P_{\mathcal{A}, Y}$, let $U : \mathcal{T}_Q \rightarrow [0, 1]$ be sampled from a family of $2Q$ -wise-independent hash functions. Then, for any $D > 1$,*

$$\Pr_U[\exists U' : \|U - U'\|_\infty \leq 1/D, \mathcal{T}^*(\mathcal{A}, Y, U, P) \neq \mathcal{T}^*(\mathcal{A}, Y, U', P)] \leq 2^{Q+1}/D.$$

Lemma 5.5. *With the notation described above, suppose that $Y : \mathcal{T}_Q \rightarrow \{0, 1\}^{N+1}$ and $U : \mathcal{T}_Q \rightarrow \{0, 1/D, \dots, (D-1)/D\}$ with $D \geq 4Q + 2/\varepsilon$ are sampled independently from families of $2Q$ -wise-independent hash functions, and $\varepsilon \in (0, 1)$, we have*

$$\Pr_{Y, U}[\exists P : P_{\mathcal{A}, Y} - P \leq \varepsilon, \mathcal{T}^*(\mathcal{A}, Y, U, P_{\mathcal{A}, Y}) \not\subseteq \mathcal{T}^*(\mathcal{A}, Y, U, P)] \leq 2Q\varepsilon.$$

We again defer the proofs of [Lemmas 5.4](#) and [5.5](#) to [Appendix A](#). At a high level, the proofs formalize the simple intuition that changing $U(v)$ or $P(v)$ by a small amount will only change $\mathcal{T}^*$ if $U(v) - P(v)$ happens to lie in a small interval, which is unlikely.

With all of the machinery that we have built above, we are now ready to prove [Proposition 5.1](#).

Proof of Proposition 5.1. Suppose without loss of generality that on input $z \in \{0, 1\}^n$, $\mathcal{B}^\mathcal{O}$ flips exactly $R := R(n) \leq \text{poly}(n)$ coins, only makes queries on circuits $C : \{0, 1\}^N \rightarrow \{0, 1\}^{N+1}$ for some $N := N(n) \leq \text{poly}(n)$, and makes exactly $Q := Q(n)$ oracle queries. We may also assume without loss of generality that these queries are distinct. For $r \in \{0, 1\}^R$, we write $\mathcal{B}_{z, r}^\mathcal{O}$ for the deterministic input-less oracle algorithm obtained by running $\mathcal{B}^\mathcal{O}$ with its random coins fixed to r . Let $D := D(n) = 2^{Q + \lceil \log(1/\varepsilon) \rceil + 10} \geq 2^{10}Q/\varepsilon$, which will effectively discretize the space of probabilities $P(v)$, and $K := K(n) = 10Q/\varepsilon$, which will be the limit on the tree size.

In the protocol, the function $\mathcal{A}$ representing Arthur and the function $\mathcal{M}$ representing honest Merlin behave as follows on input $z \in \{0, 1\}^n$. Arthur samples $r \sim \{0, 1\}^R$, two $(2Q)$ -wise independent $\text{poly}(Q, n)$ -time computable hash functions $Y : \mathcal{T}_Q \rightarrow \{0, 1\}^{N+1}$ and $U : \mathcal{T}_Q \rightarrow \{0, 1/D, 2/D, \dots, (D-1)/D\}$ (as in [Lemma 2.22](#)) and sends r , Y , and U to Merlin.

Honest Merlin responds as follows. Merlin sets $P := P_{\mathcal{B}_{z, r}, Y}$ and computes $\mathcal{T}^* := \mathcal{T}^*(\mathcal{B}_{z, r}, Y, U, P)$. If $|\mathcal{T}^*| > K$, then Merlin sends Arthur the special symbol $\perp$. (This will be a low probability event.) Otherwise, $|\mathcal{T}^*| \leq K$, and Merlin sends to Arthur $\mathcal{T}^*$ and for each node $v \in \mathcal{T}_Q \setminus \mathcal{T}^*$ whose parent is in $\mathcal{T}^*$ (or the root if $\mathcal{T}^*$ is empty) Merlin sends $x_v \in \{0, 1\}^N$ such that $C_v(x_v) = Y(v)$ if $Y(v) \in \text{Im}(C_v)$ (and x_v is arbitrary otherwise) and $P_v := P_{\mathcal{B}_{z, r}, Y}(v)$. (We could have Merlin send either a preimage x_v of $Y(v)$ or P_v if $P_{\mathcal{B}_{z, r}, Y}(v) < U(v)$. But it seems simpler to just have Merlin send the same thing for each v .)

Given some tree $\mathcal{T}^\dagger$ and Merlin's list (x_v, P_v) , Arthur checks that the tree $\mathcal{T}^\dagger$ looks to be constructed properly as follows. First, he simply checks that Merlin did not send $\perp$, that $|\mathcal{T}^\dagger| \leq K$, and that $T_v \in \{1, \dots, 2^N\}$, where $T_v := 2^N(2 - 1/P_v)$. (This is chosen so that if $P_v = P_{\mathcal{B}_{z, r}, Y}(v)$ is honest, then $T_v = |\text{Im}(C_v)|$.) Then, for each of the at most $2K + 1$ nodes v in $\mathcal{T}_Q \setminus \mathcal{T}^\dagger$ with a parent in $\mathcal{T}^\dagger$ (or the root node v if $\mathcal{T}^\dagger = \emptyset$), Arthur makes sure that Merlin has sent x_v and P_v such that either $C_v(x_v) = Y(v)$ or $P_v < U(v)$.

Then, Arthur checks the values of P_v by running the range-size upper bound protocol from [Theorem 3.2](#) with range-size estimate $T_v = 2^N(2 - 1/P_v)$. In particular, by applying [Theorem 2.16](#), Arthur can set parameters such that if $|\text{Im}(C_v)| \leq T_v$ and Merlin is honest then Arthur will always accept, but if $|\text{Im}(C_v)| \geq T_v + \varepsilon 2^N/(10Q)$ then Arthur will reject with probability at least, say, $1 - \varepsilon/(100K)$, regardless of Merlin's behavior. Arthur runs this protocol for each of the values P_v sent by Merlin in parallel.

If any one of Arthur's checks fails (either a check on $\mathcal{T}^\dagger$ or a check on P_v), Arthur outputs the special symbol $\perp$.

Otherwise, let $\mathcal{L}^\dagger$ be the set of nodes of depth $Q - 1$ in $\mathcal{T}^\dagger$. Intuitively, each $\ell \in \mathcal{L}^\dagger$ represents a completed computation of $\mathcal{B}_{z,r}^\mathcal{O}$. If $\mathcal{L}^\dagger$ is empty, Arthur outputs $?$ (i.e., says "I don't know"). If $\mathcal{L}^\dagger$ is non-empty, Arthur computes for each such leaf $\ell \in \mathcal{L}^\dagger$ the output bit $b_\ell \in \{0, 1\}$ of $\mathcal{B}_{z,r}^\mathcal{O}$ in the completed computation represented by the leaf ℓ . (In other words, b_ℓ is the output of $\mathcal{B}_{z,r}^\mathcal{O}$ when its oracle query responses are $Y(v_1), \dots, Y(v_Q)$ where $v_1, \dots, v_Q = \ell$ is the path from the root v_1 to ℓ .) If all of these values b_ℓ are the same for all such leaves ℓ , then Arthur outputs the bit b_ℓ . If any of them disagree, Arthur outputs $\perp$.

Clearly the protocol has the appropriate syntax, and clearly Arthur runs in time that is polynomial in n . Notice that it suffices to prove the result for the algorithm $\mathcal{B}_{z,r}^\mathcal{O}$ (i.e., $\mathcal{B}^\mathcal{O}$ with its input fixed to z and its coins fixed to r). The result for $\mathcal{B}^\mathcal{O}$ then follows by taking expectations over r .

To prove correctness and soundness, we first address the fact that $U(v)$ is sampled from a finite set, rather than uniformly from the interval $[0, 1]$ as in Lemma 5.3. To that end, let $\delta : \mathcal{T}_Q \rightarrow [0, 1/D]$ be a truly uniformly random function and let $U'(v) := U(v) + \delta(v)$. Notice that $U' : \mathcal{T}_Q \rightarrow [0, 1]$ is a $2Q$ -wise independent function and by Lemma 5.4, we have that $\mathcal{T}^*(\mathcal{B}_{z,r}, Y, U, P) = \mathcal{T}^*(\mathcal{B}_{z,r}, Y, U', P)$ except with probability at most $\varepsilon/5$.

To prove correctness, we then notice that if Merlin is honest, then except with probability at most $\varepsilon/5$, the tree $\mathcal{T}^\dagger$ that he constructs will be equal to $\mathcal{T}^*(\mathcal{B}_{z,r}, Y, U', P)$. And, by Item 1 of Lemma 5.3, we see that $|\mathcal{T}^*(\mathcal{B}_{z,r}, Y, U', P)| \leq K$ except with probability at most $Q/(2K) < \varepsilon/10$. Therefore, the probability that Merlin sends $\perp$ to Arthur or that $\mathcal{T}^\dagger \neq \mathcal{T}^*(\mathcal{B}_{z,r}, Y, U', P)$ when Merlin is honest is at most ε .

Then, notice that when Merlin does not send $\perp$, if all $\ell \in \mathcal{L}^\dagger$ satisfy that $b_\ell = b$, then Arthur will either output b or $?$. (The $?$ output corresponds to the case when $\mathcal{L}^\dagger$ is empty.) By the upper bound in Item 4, we see that the probability that there exists a $\ell \in \mathcal{L}^\dagger$ such that $b_\ell = 1 - b$ is at most $\Pr_{r \sim \{0,1\}^R, \mathcal{O} \sim \text{RAvoid}}[\mathcal{B}_{z,r}^\mathcal{O} = 1 - b]/2 = (1 - p_b(z))/2$. Correctness then follows by union bound.

To prove soundness, notice that Arthur outputs b or $\perp$ whenever there is some $\ell \in \mathcal{L}^*(\mathcal{B}_{z,r}, Y, U, P)$ such that $b_\ell = b$. (Indeed, notice that if $\ell \in \mathcal{L}^*$, then either Arthur outputs $\perp$ or $\ell \in \mathcal{L}^\dagger$. This crucially relies on the basic fact that Merlin cannot provide x_v such that $C_v(x_v) = Y(v)$ when $Y(v) \notin \text{Im}(C_v)$.) So, we wish to prove that this happens with probability at least $p_b(z)/2 - Q/(2(Q+1)) - \varepsilon$.

To that end, first note that Arthur will reject at least one P_v with probability at least $1 - \varepsilon/10$ unless we have that $|\text{Im}(C_v)| < T_v + \varepsilon 2^N/(10Q)$ for all v for which Merlin provides a P_v . It follows that Arthur will reject with probability at least $1 - \varepsilon/10$ unless $P_{\mathcal{B}_{z,r}, Y}(v) - P_v \leq \varepsilon/(10Q)$.⁸ It therefore suffices to show that for any choice of P_v satisfying $P_{\mathcal{B}_{z,r}, Y}(v) - P_v \leq \varepsilon/(10Q)$ for all v , there must exist a leaf $\ell \in \mathcal{L}^*(\mathcal{B}_{z,r}, Y, U, P)$ with $b_\ell = b$ with probability at least, say, $p_b(z)/2 - Q/(2(Q+1)) - \varepsilon/2$.

Indeed, by Lemma 5.5, we have that

$$\mathcal{L}^*(\mathcal{B}_{z,r}, Y, U, P_{\mathcal{B}_{z,r}, Y}) \subseteq \mathcal{L}^*(\mathcal{B}_{z,r}, Y, U, P)$$

⁸To see this, note that $P_{\mathcal{B}_{z,r}, Y}(v) = f(|\text{Im}(C_v)|)$ and $P_v = f(T_v)$, where

$$f(t) := \frac{2^N}{2^{N+1} - t}.$$

In particular, since f is differentiable with derivative f' satisfying $0 \leq f'(t) \leq 2^{-N}$ for $0 \leq t \leq 2^N$, it follows that $f(|\text{Im}(C_v)|) - f(T_v) \leq 2^{-N}(|\text{Im}(C_v)| - T_v) \leq \varepsilon/(10Q)$, as claimed.

except with probability at most $\varepsilon/5$. Then, by [Lemma 5.4](#), we have that

$$\mathcal{L}^*(\mathcal{B}_{z,r}, Y, U, P_{\mathcal{B}_{z,r}, Y}) = \mathcal{L}^*(\mathcal{B}_{z,r}, Y, U', P_{\mathcal{B}_{z,r}, Y})$$

except with probability at most $\varepsilon/5$. And, by [Item 4](#), we have that there exists $\ell \in \mathcal{L}^*(\mathcal{B}_{z,r}, Y, U', P_{\mathcal{B}_{z,r}, Y})$ such that $b_\ell = b$ with probability at least $p_b(z)/2 - Q/(2(Q+1))$. The result then follows by union bound. $\square$

6 Avoid is almost in TFNP

The Avoid problem is in $\text{TF}\Sigma_2^P$ rather than TFNP; although Avoid is total, one needs an NP oracle to verify any proposed solution. In fact, since checking if 0 is not in the range of a circuit $C : \{0, 1\}^n \rightarrow \{0, 1\}^{n+1}$ is coNP-hard, putting Avoid in TFNP would imply $P = NP$. (See [Footnote 3](#).)

However, to our surprise, we find that under plausible derandomization assumptions, Avoid has a randomized reduction to a TFNP problem that succeeds with very high probability.

Theorem 6.1. *Assuming [Assumption 2.10](#), there exists a TFNP problem A such that for any constant c , Avoid has a randomized one-query reduction to A which succeeds with probability at least $1 - 1/n^c$.*

Proof. Let $G : \{0, 1\}^{\alpha \cdot \log(n+1)} \rightarrow \{0, 1\}^{n+1}$ be a $\frac{1}{n^3}$ -PRG which fools n^3 size $\text{NP}_{||}$ circuits, whose existence we have assumed by [Assumption 2.10](#). By [Theorem 2.19](#), the existence of such a G in particular implies that $\text{prAM} = \text{prNP}$. Since by [Theorem 3.2](#), ε -GapRangeSize is in prAM for, say, $\varepsilon := 1/n^{50}$, it follows that ε -GapRangeSize is in prNP . Formally, there exists a polynomial-time verifier V such that for any circuit $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$, if $|\text{Im}(C)| \geq \tau + \varepsilon 2^n$, then for all π , $V(C, \tau, \pi) = 0$, but if $|\text{Im}(C)| \leq \tau$, then there exists a π such that $V(C, \tau, \pi) = 1$.

The problem A is defined as follows. For sufficiently large n , given a circuit $C : \{0, 1\}^n \rightarrow \{0, 1\}^{n+1}$ where $|C| \leq n^2$, output both of the following.

1. $\tau \in [2^n], \pi \in \{0, 1\}^{\text{poly}(n)}$ such that $V(C, \tau, \pi) = 1$. In other words, an estimate τ of the range size of C and a proof π that $\tau > \tau^* - \varepsilon 2^n$, where $\tau^* := |\text{Im}(C)|$.
2. $w_1, \dots, w_{(n+1)^\alpha} \in \{0, 1\}^n$ such that for at least a $\frac{\tau}{2^{n+1}} - \frac{1}{n^3}$ fraction of $i \in [(n+1)^\alpha]$, $C(w_i) = G(i)$.

We first show that A is a TFNP problem by showing that A is an FNP problem and that it is total. Since V is efficient and since checking if $C(w_i) = G(i)$ for sufficiently many $i \in [(n+1)^\alpha]$ can be done efficiently, there is a simple efficient procedure for verifying solutions to A, and A is therefore in FNP. To see that A is total, we claim that there is always a solution to A with $\tau = \tau^*$. Indeed, if $\tau = \tau^*$, then by the definition of V , there exists a polynomial size witness π such that $V(C, \tau, \pi) = 1$. And, for $\tau = \tau^*$, [Lemma 2.12](#) tells us that at least a

$$\frac{\tau}{2^{n+1}} - \frac{1}{n^3}$$

fraction of the $G(i)$ must have a preimage under C . So, there exist solutions $w_1, \dots, w_{(n+1)^\alpha}$ in this case.

We now show how to reduce Avoid to A. By [Lemma 2.3](#), we can reduce our avoid instance $C : \{0,1\}^n \rightarrow \{0,1\}^{n+1}$ where $|C| = s$ to an Avoid instance $C' : \{0,1\}^{n'} \rightarrow \{0,1\}^{n'+1}$ where $|C'| \leq n'^2$ and $n' \geq n^c$ in polynomial time. Notice that solving Avoid on C' with probability at least $1 - 1/n'$ will yield a solution to Avoid on C with probability at least $1 - 1/n^c$. We will therefore assume without loss of generality that we only wish to solve range avoidance on circuits $C : \{0,1\}^n \rightarrow \{0,1\}^{n+1}$ where $|C| \leq n^2$ with probability at least $1 - 1/n$.

We show a randomized reduction from Avoid on $C : \{0,1\}^n \rightarrow \{0,1\}^{n+1}$ where $|C| \leq n^2$ to A such that the reduction succeeds with probability at least $1 - 1/n$. The reduction feeds our circuit C to an oracle for A to get back τ, π (a witness that $\tau \gtrsim \tau^*$), and $w_1, \dots, w_{(n+1)^\alpha} \in \{0,1\}^n$. Let $I := \{i : C(w_i) \neq G(i)\}$. The reduction samples i^* uniformly from I and outputs $G(i^*)$.

The reduction clearly runs in polynomial time. We now show correctness of the reduction. Let $J := \{i : G(i) \notin \text{Im}(C)\}$. In other words, $G(i)$ is a solution to the Avoid instance if and only if $i \in J$. Since we must have $J \subseteq I$, it follows that the reduction succeeds with probability exactly $|J|/|I|$. So, it remains to prove that $|J|/|I| \geq 1 - 1/n$.

We will first bound $|I|$ and $|J|$ separately. First, by [Lemma 2.12](#), we have

$$\left| \frac{|J|}{(n+1)^\alpha} - \left(1 - \frac{\tau^*}{2^{n+1}}\right) \right| \leq \frac{1}{n^3}.$$

Rearranging gets us

$$\frac{|J|}{(n+1)^\alpha} \geq 1 - \frac{\tau^*}{2^{n+1}} - \frac{1}{n^3}.$$

We now bound $|I|$. By the definition of A,

$$\frac{|I|}{(n+1)^\alpha} \leq 1 - \frac{\tau}{2^{n+1}} + \frac{1}{n^3} < 1 - \frac{\tau^* - \varepsilon 2^n}{2^{n+1}} + \frac{1}{n^3} \leq 1 - \frac{\tau^*}{2^{n+1}} + \frac{2}{n^3}.$$

Let $k = \tau^*/2^{n+1}$. Note that $k \leq 1/2$. We first bound $|I| - |J|$ and will then use that to bound $|J|/|I|$. We have

$$|I| - |J| \leq (n+1)^\alpha \left((1 - k + 2/n^3) - (1 - k - 1/n^3) \right) = 3(n+1)^\alpha/n^3.$$

Therefore,

$$\frac{|J|}{|I|} = 1 - \frac{|I| - |J|}{|I|} \geq 1 - \frac{3(n+1)^\alpha/n^3}{|I|} \geq 1 - \frac{3/n^3}{1 - k - 1/n^3} \geq 1 - \frac{3/n^3}{1/2 - 1/n^3} \geq 1 - 1/n,$$

as needed, where the last inequality holds for sufficiently large n . $\square$

The following corollary follows immediately from combining [Theorem 6.1](#) above with [Theorem 2.11](#) (the fact that sufficiently strong circuit lower bounds imply sufficiently strong PRGs).

Corollary 6.2. *If E does not have $2^{o(n)}$ -size non-deterministic circuits, then there exists a TFNP problem A such that for all constants c , there exists a randomized reduction from Avoid to A which succeeds with probability at least $1 - 1/n^c$.*

We are now able to use [Theorem 6.1](#) to show that if plausible derandomization assumptions hold and SAT is not checkable, then Avoid is not NP-hard. This is an alternative proof (under less standard assumptions than $\text{PH} \neq \text{AM}$) that Avoid is not NP-hard.

Corollary 6.3. *If [Assumption 2.10](#) holds and SAT is not checkable, then Avoid is not NP-hard under randomized Turing reductions. Consequently, if E requires exponential size non-deterministic circuits and SAT is not checkable, then Avoid is not NP-hard under randomized Turing reductions.*

Proof. We will show that if [Assumption 2.10](#) holds, and Avoid is NP-hard, then SAT is checkable. Say there is a randomized Turing reduction from SAT on an instance of length n to Avoid which runs in time n^c for some constant c . We assume without loss of generality (by standard probability amplification techniques) that the randomized reduction succeeds with probability at least 99/100. Since [Assumption 2.10](#) holds, by [Theorem 6.1](#), there exists a TFNP problem A such that Avoid has a randomized reduction to A which succeeds with probability at least $1 - 1/n^{c+1}$. Chaining these two reductions together, we see that we have a Turing reduction from SAT to A which succeeds with the following probability.

$$\frac{99}{100} \left(1 - \frac{1}{n^{c+1}}\right)^{n^c} \geq \frac{99}{100} e^{-\Theta(1/n)} \geq \frac{99}{100} (1 - \Theta(1/n))$$

Therefore, SAT has a randomized reduction to the TFNP problem A that succeeds with probability at least 2/3. It follows from [Theorem 2.14](#) that SAT is checkable. $\square$

References

- [ABK24] Scott Aaronson, Harry Buhrman, and William Kretschmer. A qubit, a coin, and an advice string walk into a relational problem. In *ITCS*, 2024. [2](#)
- [Bab85] László Babai. Trading group theory for randomness. In *STOC*, 1985. [6](#), [19](#), [20](#), [21](#)
- [BG93] Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Randomness in interactive proofs. *Computational Complexity*, 3(4):319–354, 1993. [20](#)
- [BHZ87] Ravi B. Boppana, Johan Håstad, and Stathis Zachos. Does co-NP have short interactive proofs? *Information Processing Letters*, 25:127–132, 1987. [2](#)
- [BK95] Manuel Blum and Sampath Kannan. Designing programs that check their work. *Journal of the ACM*, 42(1):269–291, 1995. [19](#)
- [BM88] László Babai and Shlomo Moran. Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36(2):254–276, 1988. [6](#), [20](#), [21](#)
- [CGL⁺26] Eldon Chung, Alexander Golovnev, Zeyong Li, Maciej Obremski, Sidhant Saraogi, and Noah Stephens-Davidowitz. The hardness of range avoidance for randomized algorithms implies Minicrypt. In *FSTTCS*, 2026. [1](#), [2](#), [5](#), [13](#), [14](#)
- [CHLR23] Yeyuan Chen, Yizhi Huang, Jiayu Li, and Hanlin Ren. Range avoidance, remote point, and hard partial truth table via satisfying-pairs algorithms. In *STOC*, 2023. [1](#)
- [CHR24] Lijie Chen, Shuichi Hirahara, and Hanlin Ren. Symmetric exponential time requires near-maximum circuit size. In *STOC*, 2024. [1](#)

- [CL24] Yilei Chen and Jiatu Li. Hardness of range avoidance and remote point for restricted circuits via cryptography. In *STOC*, 2024. [1](#), [2](#), [4](#), [13](#)
- [CLL25] Lijie Chen, Jiatu Li, and Jingxun Liang. Maximum circuit lower bounds for exponential-time Arthur Merlin. In *STOC*, 2025. [1](#), [13](#)
- [Dru10] Andrew Donald Drucker. PCPs for Arthur-Merlin games and communication protocols. Master’s thesis, Massachusetts Institute of Technology, 2010. [20](#)
- [FGM⁺89] Martin Fürer, Oded Goldreich, Yishay Mansour, Michael Sipser, and Stathis Zachos. On completeness and soundness in interactive proof systems. *Advances in Computing Research*, 5:429–442, 1989. [20](#)
- [FIM25] Noah Fleming, Deniz Imrek, and Christophe Marciot. Provably total functions in the polynomial hierarchy. In *CCC*, 2025. [1](#)
- [For87] Lance Fortnow. The complexity of perfect zero-knowledge. In *STOC*, 1987. [13](#)
- [GGLS26] Karthik Gajulapalli, Surendra Ghentiyala, Zeyong Li, and Sidhant Saraogi. Downward self-reducibility in the total function polynomial hierarchy. In *SODA*, 2026. [1](#)
- [GGNS23] Karthik Gajulapalli, Alexander Golovnev, Satyajeet Nagargoje, and Sidhant Saraogi. Range avoidance for constant depth circuits: Hardness and algorithms. In *RANDOM*, 2023. [1](#)
- [GLV24] Karthik Gajulapalli, Zeyong Li, and Ilya Volkovich. Oblivious complexity classes revisited: Lower bounds and hierarchies. In *FSTTCS*, 2024. [1](#)
- [GLW22] Venkatesan Guruswami, Xin Lyu, and Xiuhan Wang. Range avoidance for low-depth circuits and connections to pseudorandomness. In *RANDOM*, 2022. [1](#)
- [GS86] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *STOC*, 1986. [3](#), [11](#), [14](#), [20](#), [22](#)
- [HLZ26] Shengtang Huang, Xin Li, and Yan Zhong. Range avoidance and remote point: New algorithms and hardness. In *ITCS*, 2026. [1](#)
- [HV26] Edward A. Hirsch and Ilya Volkovich. Upper and lower bounds for the linear ordering principle. In *STACS*, 2026. [1](#), [5](#)
- [ILW23] Rahul Ilango, Jiatu Li, and R. Ryan Williams. Indistinguishability obfuscation, range avoidance, and bounded arithmetic. In *STOC*, 2023. [1](#), [2](#), [13](#), [14](#)
- [IW97] Russell Impagliazzo and Avi Wigderson. P= BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *STOC*, 1997. [18](#)
- [Jeř04] Emil Jeřábek. Dual weak pigeonhole principle, Boolean complexity, and derandomization. *Annals of Pure and Applied Logic*, 129:1–37, 2004. [4](#)
- [KKMP21] Robert Kleinberg, Oliver Korten, Daniel Mitropolsky, and Christos Papadimitriou. Total functions in the polynomial hierarchy. In *ITCS*, 2021. [1](#), [2](#), [4](#), [5](#), [14](#)

- [Kor21] Oliver Korten. The hardest explicit construction. In *FOCS*, 2021. [1](#), [4](#), [5](#)
- [Kor22] Oliver Korten. Derandomization from time-space tradeoffs. In *CCC*, 2022. [1](#)
- [Kor25] Oliver Korten. Range avoidance and the complexity of explicit constructions. *Bulletin of EATCS*, 145(1), 2025. [1](#), [2](#)
- [KP24] Oliver Korten and Toniann Pitassi. Strong vs. weak range avoidance and the linear ordering principle. In *FOCS*, 2024. [1](#), [5](#)
- [KPI25] Oliver Korten, Toniann Pitassi, and Russell Impagliazzo. Stronger cell probe lower bounds via local PRGs. In *FOCS*, 2025. [1](#)
- [KS25a] Oliver Korten and Rahul Santhanam. How to construct random strings. In *CCC*, 2025. [1](#), [4](#), [12](#), [13](#)
- [KS25b] Neha Kuntewar and Jayalal Sarma. Avoiding range via Turan-type bounds. In *RANDOM*, 2025. [1](#)
- [KvM99] Adam R Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. In *STOC*, 1999. [7](#), [18](#), [21](#)
- [Li24] Zeyong Li. Symmetric exponential time requires near-maximum circuit size: Simplified, truly uniform. In *STOC*, 2024. [1](#)
- [LORS24] Zhenjian Lu, Igor C. Oliveira, Hanlin Ren, and Rahul Santhanam. On the complexity of avoiding heavy elements. In *FOCS*, 2024. [1](#)
- [MX10] Mohammad Mahmoody and David Xiao. On the power of randomized reductions and the checkability of SAT. In *CCC*, 2010. [19](#)
- [RSW22] Hanlin Ren, Rahul Santhanam, and Zhikun Wang. On the range avoidance problem for circuits. In *FOCS*, 2022. [1](#), [2](#)
- [RWZ26] Hanlin Ren, Yichuan Wang, and Yan Zhong. Hardness of range avoidance and proof complexity generators from demi-bits. In *ITCS*, 2026. [1](#), [4](#), [13](#)
- [SS24] Ronen Shaltiel and Jad Silbak. Explicit codes for poly-size circuits and functions that are hard to sample on low entropy distributions. In *STOC*, 2024. [3](#), [7](#), [18](#)
- [SU06] Ronen Shaltiel and Christopher Umans. Pseudorandomness for approximate counting and sampling. *Computational Complexity*, 15(4):298–341, 2006. [18](#)
- [Vad12] Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1–3):1–336, 2012. [21](#)
- [WG75] H. W. Watson and Francis Galton. On the probability of the extinction of families. *The Journal of the Anthropological Institute of Great Britain and Ireland*, 4:138–144, 1875. [10](#), [32](#)

A Additional proofs, deferred from Section 5.1

Here, we collect the rather technical proofs about the tree of computation that we deferred in Section 5.1. Our proofs are intentionally quite careful and technical.

Proof of Lemma 5.3. Our proofs of Items 1, 2 and 3 are via careful (and tedious) computation. Item 4 will then be a simple corollary of Items 2 and 3.

To prove Items 1 and 2, for $v \in \mathcal{T}_Q$, let E_v be the event that $v \in \mathcal{T}^*$. We first claim that for two vertices $v, v' \in \mathcal{T}_Q$ with the same depth d and deepest common ancestor with depth $e \leq d$, we have

$$\Pr[E_v \text{ and } E_{v'}] = 2^{-2d+e-1}.$$

(Notice that the subtree in $\mathcal{T}_Q$ consisting of the union of the path to v and the path to v' has size exactly $2d - e + 1$, so that this quantity arises naturally if one imagines that each vertex in this union appears in $\mathcal{T}^*$ independently with probability $1/2$. For Item 1, we will only use the special case when $v = v'$, which is much easier to prove. We prove the more general case in order to prove Item 2.)

Indeed, notice that the distinct vertices in the union of the path from the root to v and the path from the root to v' can be written as $v_1, \dots, v_k$, where $k := 2d - e + 1$, $v_1, \dots, v_{d+1}$ is the path from the root v_1 to $v = v_{d+1}$, and $v_1, \dots, v_{e+1}, v_{d+2}, \dots, v_k$ is the path from v_1 to $v' = v_k$. Then, the events E_v and $E_{v'}$ occur if and only if $Y(v_i) \notin \text{Im}(C_{v_i})$ and $U(v_i) \leq P(v_i)$ for all $1 \leq i \leq k$. So, for fixed $y_1, \dots, y_k \in \{0, 1\}^{N+1}$, for $1 \leq i \leq d+1$, let C_i be the i th circuit queried by $\mathcal{A}^O$ when its first $i-1$ oracle query responses are fixed to $y_1, \dots, y_{i-1}$. Similarly, for $d+2 \leq i \leq k$, let C_i be the $(i-d+e)$ th circuit queried by $\mathcal{A}^O$ if its first $e+1$ oracle query responses are fixed to $y_1, \dots, y_{e+1}$ and the next $i-(d+2)$ oracle query responses are fixed to $y_{d+2}, \dots, y_{i-1}$. Notice in particular that these definitions are chosen so that if $Y(v_i) = y_i$ for all i , then $C_i = C_{v_i}$. Furthermore, notice that C_i is fixed given $y_1, \dots, y_{i-1}$.

Then, we have

$$\begin{aligned} \Pr_{Y,U}[E_v \text{ and } E_{v'}] &= \sum_{y_1 \notin \text{Im}(C_1), \dots, y_k \notin \text{Im}(C_k)} \Pr_Y[\forall i, Y(v_i) = y_i] \cdot \Pr_{Y,U}[\forall i, U(v_i) \leq P(v_i) \mid \forall i, Y(v_i) = y_i] \\ &= 2^{-(N+1)k} \sum_{y_1 \notin \text{Im}(C_1), \dots, y_k \notin \text{Im}(C_k)} \Pr_{Y,U}[\forall i, U(v_i) \leq P(v_i) \mid \forall i, Y(v_i) = y_i], \end{aligned}$$

where in the second equality we have used the fact that Y is $2Q$ -wise independent (which of course implies that Y is k -wise independent since $k = 2d - e + 1 \leq 2(Q-1) - e + 1 < 2Q$).

Using the fact that U is itself $2Q$ -wise independent and chosen independently of Y , and recalling the definition of $P := P_{\mathcal{A},Y}$, we see that

$$\begin{aligned} \Pr_{Y,U}[E_v \text{ and } E_{v'}] &= 2^{-(N+1)k} \sum_{y_1 \notin \text{Im}(C_1), \dots, y_k \notin \text{Im}(C_k)} \left(\prod_{i=1}^k \frac{2^N}{2^{N+1} - |\text{Im}(C_i)|} \right) \\ &= 2^{-k} \sum_{y_1 \notin \text{Im}(C_1), \dots, y_k \notin \text{Im}(C_k)} \left(\prod_{i=1}^k \frac{1}{2^{N+1} - |\text{Im}(C_i)|} \right) \\ &= 2^{-k} \sum_{y_1 \notin \text{Im}(C_1)} \frac{1}{2^{N+1} - |\text{Im}(C_1)|} \cdots \sum_{y_k \notin \text{Im}(C_k)} \frac{1}{2^{N+1} - |\text{Im}(C_k)|} \\ &= 2^{-k}, \end{aligned}$$

as claimed.⁹

Items 1 and 2 now follow from basic techniques. In particular, we first notice that by plugging in the special case when $v = v'$ above (in which case $e = d$), we see that

$$\mathbb{E}_{Y,U}[|\mathcal{T}^*|] = \sum_{v \in \mathcal{T}_Q} \Pr[E_v] = \sum_{v \in \mathcal{T}_Q} 2^{-d-1} = \sum_{d=0}^{Q-1} 2^d \cdot 2^{-d-1} = Q/2,$$

where d is the depth of v and we have used the fact that the number of nodes at depth d is 2^d . Item 1 then follows by Markov's inequality.

Now, to prove Item 2, again taking the special case when $\ell' = \ell$ for some leaf $\ell \in \mathcal{T}_Q$ at depth $d = Q - 1$, we see that

$$\mathbb{E}_{Y,U}[|\mathcal{L}^*|] = \sum_{\ell} \Pr_{Y,U}[E_\ell] = \sum_{\ell} 2^{-Q} = 1/2.$$

And, for the second moment,

$$\mathbb{E}_{Y,U}[|\mathcal{L}^*|^2] = \sum_{\ell, \ell'} \Pr_{Y,U}[E_\ell \wedge E_{\ell'}] = \sum_{\ell, \ell'} 2^{-2Q+e+1},$$

where e is the depth of the deepest common ancestor of ℓ and ℓ' . The number of pairs of leaves $\ell, \ell' \in \mathcal{T}_Q$ with deepest common ancestor at depth e is exactly 2^{2Q-e-3} if $e \leq Q - 2$ (because there are 2^{Q-1} choices for v , and 2^{Q-e-2} choices for v' once v is fixed—accounting for the fact that v and v' must have a different ancestor at depth $e + 1$) and 2^{Q-1} if $e = Q - 1$. So,

$$\mathbb{E}_{Y,U}[|\mathcal{L}^*|^2] = \frac{1}{2} + \sum_{e=0}^{Q-2} \frac{1}{4} = \frac{Q+1}{4}.$$

Item 2 then follows from the second moment method, which tells us that

$$\Pr[|\mathcal{L}^*| = 0] \leq 1 - \frac{\mathbb{E}[|\mathcal{L}^*|]^2}{\mathbb{E}[|\mathcal{L}^*|^2]} = 1 - \frac{1/4}{(Q+1)/4} = \frac{Q}{Q+1},$$

as claimed.

To prove Item 3, let $v_1, \dots, v_Q := \ell$ be the path from the root v_1 to the leaf $\ell = v_Q$. And, for fixed $y_1, \dots, y_Q \in \{0, 1\}^{N+1}$, let C_i be the circuit queried by $\mathcal{A}^\mathcal{O}$ in its i th oracle query if its first $i - 1$ oracle query responses are fixed to $y_1, \dots, y_{i-1}$. Similarly, let $b^* := b^*(y_1, \dots, y_Q)$ be the output of $\mathcal{A}^\mathcal{O}$ with its oracle query responses fixed to $y_1, \dots, y_Q$.

Then,

$$\begin{aligned} \Pr[b_\ell = 1 \text{ and } \ell \in \mathcal{L}^*] &= \sum_{y_1, \dots, y_Q \in \{0, 1\}^{N+1}} b^* \Pr[\forall i, Y(v_i) = y_i] \Pr[\ell \in \mathcal{L}^* \mid \forall i, Y(v_i) = y_i] \\ &= 2^{-Q(N+1)} \cdot \sum_{y_1, \dots, y_Q \in \{0, 1\}^{N+1}} b^* \Pr[\ell \in \mathcal{L}^* \mid \forall i, Y(v_i) = y_i] \\ &= 2^{-Q(N+1)} \cdot \sum_{y_1 \notin \text{Im}(C_1), \dots, y_Q \notin \text{Im}(C_Q)} b^* \Pr[\ell \in \mathcal{L}^* \mid \forall i, Y(v_i) = y_i], \end{aligned}$$

⁹This proof is more delicate than it might appear, which is why we have been so careful. E.g., one might think that for any arbitrary collection $u_1, \dots, u_k$ of distinct vertices with $k \leq 2Q$ we have that $\Pr[\forall i, Y(u_i) \notin \text{Im}(C_{u_i}) \text{ and } U(u_i) \leq P(u_i)] = 2^{-k}$. But, this stronger statement is not necessarily true because of the fact that the circuits C_{u_i} can themselves depend on the values of $Y(v)$ for any ancestor v of any of the u_i . So, k -wise independence is not enough to prove this more general statement. The crucial fact that we are using here is that C_i is defined given $y_1, \dots, y_{i-1}$. Otherwise, the set $\{(y_1, \dots, y_k) : \forall i, y_i \notin \text{Im}(C_i)\}$ that we are summing over would not be well defined.

where for the second equality we have used the fact that Y is Q -wise independent (in fact, $2Q$ -wise independent) and for the last equality we have used the fact that $\ell \in \mathcal{L}^*$ only if $Y(v_1) \notin \text{Im}(C_1), \dots, Y(v_Q) \notin \text{Im}(C_Q)$.

Finally, notice that if $y_1 \notin \text{Im}(C_1), \dots, y_Q \notin \text{Im}(C_Q)$, then $\ell \in \mathcal{L}^*$ if and only if $U(v_i) \leq P(v_i)$ for all i . Recalling again that $P := P_{A,Y}$ satisfies

$$P(v_i) = \frac{2^N}{2^{N+1} - |\text{Im}(C_i)|} ,$$

and since U is chosen independently from Y and is itself Q -wise independent (in fact, $(2Q)$ -wise independent), we see that for $y_1 \notin \text{Im}(C_1), \dots, y_Q \notin \text{Im}(C_Q)$,

$$\Pr[\ell \in \mathcal{L}^* \mid \forall i, Y(v_i) = y_i] = \prod_{i=1}^Q \frac{2^N}{2^{N+1} - |\text{Im}(C_i)|} .$$

Therefore,

$$\begin{aligned} \Pr[b_\ell = 1 \text{ and } \ell \in \mathcal{L}^*] &= 2^{-Q(N+1)} \cdot \sum_{y_1 \notin \text{Im}(C_1), \dots, y_Q \notin \text{Im}(C_Q)} b^* \cdot \prod_{i=1}^Q \frac{2^N}{2^{N+1} - |\text{Im}(C_i)|} \\ &= 2^{-Q} \cdot \sum_{y_1 \notin \text{Im}(C_1), \dots, y_Q \notin \text{Im}(C_Q)} b^* \cdot \prod_{i=1}^Q \frac{1}{2^{N+1} - |\text{Im}(C_i)|} . \end{aligned}$$

It remains to show that this probability is actually equal to $2^{-Q} \Pr_{\mathcal{O} \sim \text{RAvoid}}[\mathcal{A}^\mathcal{O} = 1]$. To that end, notice that

$$\begin{aligned} \Pr_{\mathcal{O} \sim \text{RAvoid}}[\mathcal{A}^\mathcal{O} = 1] &= \sum_{y_1 \notin \text{Im}(C_1)} \frac{1}{2^{N+1} - |\text{Im}(C_1)|} \sum_{y_2 \notin \text{Im}(C_2)} \frac{1}{2^{N+1} - |\text{Im}(C_2)|} \cdots \sum_{y_Q \notin \text{Im}(C_Q)} \frac{1}{2^{N+1} - |\text{Im}(C_Q)|} \cdot b^* \\ &= \sum_{y_1 \notin \text{Im}(C_1), \dots, y_Q \notin \text{Im}(C_Q)} b^* \cdot \prod_{i=1}^Q \frac{1}{2^{N+1} - |\text{Im}(C_i)|} \\ &= 2^Q \Pr[b_\ell = 1 \text{ and } \ell \in \mathcal{L}^*] , \end{aligned}$$

as claimed. (Here, we have used the fact that $\mathcal{A}^\mathcal{O}$ never makes the same query twice, so that for $\mathcal{O} \sim \text{RAvoid}$ we have $\Pr[\mathcal{O}(C_i) = y_i \mid \mathcal{O}(C_j) = y_j] = 1/(2^{N+1} - |\text{Im}(C_i)|)$ for any $y_i \notin \text{Im}(C_i)$.)

Finally, we show that [Item 4](#) is a relatively immediate corollary of what we have already proven. The upper bound in [Item 4](#) follows by taking a union bound over all 2^{Q-1} choices of leaves $\ell \in \mathcal{T}_Q$ of the inequality in [Item 3](#). The lower bound in [Item 4](#) follows by noting that

$$\begin{aligned} \Pr[\exists \ell \in \mathcal{L}^* : b_\ell = b] &\geq 1 - \Pr[\mathcal{L}^* = \emptyset] - \Pr[\exists \ell \in \mathcal{L}^* : b_\ell = 1 - b] \\ &\geq 1 - \frac{Q}{Q+1} - \frac{1}{2} \cdot \Pr_{\mathcal{O} \sim \text{RAvoid}}[\mathcal{A}^\mathcal{O} = 1 - b] \\ &= \frac{1}{2} \cdot \Pr_{\mathcal{O} \sim \text{RAvoid}}[\mathcal{A}^\mathcal{O} = b] + 1/2 - \frac{Q}{Q+1} \\ &= \frac{1}{2} \cdot \Pr_{\mathcal{O} \sim \text{RAvoid}}[\mathcal{A}^\mathcal{O} = b] - \frac{Q-1}{2(Q+1)} , \end{aligned}$$

where the inequality follows by [Item 2](#) together with the upper bound in [Item 4](#) applied with $b' := 1 - b$. $\square$

We now prove [Lemmas 5.4](#) and [5.5](#).

Proof of [Lemma 5.4](#). Notice that if

$$\mathcal{T}^*(\mathcal{A}, Y, U, P) \neq \mathcal{T}^*(\mathcal{A}, Y, U', P) ,$$

then it must be the case that there is some $v \in \mathcal{T}_Q$ such that $|U(v) - P(v)| \leq |U'(v) - U(v)| \leq 1/D$. Since $U(v)$ is uniform in $[0, 1]$ for each $v \in \mathcal{T}_Q$, we see that for any fixed v this happens with probability at most $2/D$. The result then follows by taking a union over all $2^Q - 1 \leq 2^Q$ choices of v . $\square$

Proof of [Lemma 5.5](#). Notice that if

$$\mathcal{T}^*(\mathcal{A}, Y, U, P_{\mathcal{A}, Y}) \not\subseteq \mathcal{T}^*(\mathcal{A}, Y, U, P) ,$$

then there must be a v such that either v is the root node or whose parent is in $\mathcal{T}^*(\mathcal{A}, Y, U, P_{\mathcal{A}, Y})$ such that $P(v) < U(v) \leq P_{\mathcal{A}, Y}(v)$. If this happens for some P with $P_{\mathcal{A}, Y} - P \leq \varepsilon$, then we must have $P_{\mathcal{A}, Y}(v) - \varepsilon < U(v) \leq P_{\mathcal{A}, Y}(v)$ for some such v .

So, let F_v be the event that the *parent* of v is in $\mathcal{T}^*$ (where we say that F_v always happens if v is the root) and let B_v be the event that $P_{\mathcal{A}, Y}(v) - \varepsilon < U(v) \leq P_{\mathcal{A}, Y}(v)$. It suffices to bound the probability that there exists a v such that F_v and B_v both occur. To that end, notice that F_v depends only on the values of $U(v_1), \dots, U(v_{k-1})$ and $Y(v_1), \dots, Y(v_{k-1})$ where $v_1, \dots, v_k := v$ is the path from the root v_1 to $v = v_k$ for $k \leq Q$. Since U is Q -wise independent (in fact, $2Q$ -wise independent) and completely independent of Y , we see that

$$\Pr[B_v \mid F_v] \leq \max_{p \in [0, 1]} \Pr[p - \varepsilon \leq U(v) \leq p] \leq \varepsilon + 1/D .$$

So, we have

$$\Pr[\exists v : F_v \text{ and } B_v] \leq \sum_{v \in \mathcal{T}_Q} \Pr[F_v \text{ and } B_v] \leq (\varepsilon + 1/D) \sum_{v \in \mathcal{T}_Q} \Pr[F_v] .$$

Finally, by design, we have

$$\Pr[F_v] \leq (1/2 + 1/D)^d ,$$

where d is the depth of v . So,

$$\sum_{v \in \mathcal{T}_Q} \Pr[F_v] \leq \sum_{d=0}^{Q-1} (1/2 + 1/D)^d 2^d \leq Q(1 + 2/D)^Q .$$

Putting everything together, we see that the final probability is bounded by

$$Q(\varepsilon + 1/D)(1 + 2/D)^Q \leq Q\varepsilon e^{(2Q+1/\varepsilon)/D} \leq 2Q\varepsilon ,$$

as claimed. $\square$