Electronic Colloquium on Computational Complexity, Report No. 76 (2024)

Strong vs. Weak Range Avoidance and the Linear Ordering Principle

Oliver Korten* and Toniann Pitassi†

April 10, 2024

Abstract

In a pair of recent breakthroughs [CHR24, Li24] it was shown that the classes $\mathsf{S}_2^{\mathsf{E}}, \mathsf{ZPE}^{\mathsf{NP}}$, and Σ_2^{E} require exponential circuit complexity, giving the first unconditional improvements to a classical result of Kannan [Kan82]. These results were obtained by designing a surprising new algorithm for the total search problem *Range Avoidance:* given a circuit $C : \{0,1\}^n \to \{0,1\}^{n+1}$, find an $n+1$-bit string outside its range. Range Avoidance is a member of the class $\mathsf{TF}\Sigma_2^{\mathsf{P}}$ of total search problems in the *second level* of the polynomial hierarchy, analogous to its better-known counterpart TFNP in the first level. $\mathsf{TF}\Sigma_2^{\mathsf{P}}$ was only recently introduced in [KKMP21] and its structure is not well understood. We investigate here the extent to which algorithms of the kind in [CHR24, Li24] can be applied to other search problems in this class, and prove a variety of results both positive and negative.

On the positive side we show that Li's Range Avoidance algorithm [Li24] can be improved to give a reduction from Range Avoidance to a natural total search problem we call the *Linear Ordering Principle* or "LOP": given a circuit $\prec$: $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ purportedly defining a total order on $\{0,1\}^n$, find either a witness that $\prec$ is not a total order or else a minimal element in the ordering. The problem LOP is quite interesting in its own right, as it defines a natural *syntactic subclass* "$\mathsf{L}_2^{\mathsf{P}}$" of $\mathsf{S}_2^{\mathsf{P}}$ which nonetheless maintains most of the interesting properties of $\mathsf{S}_2^{\mathsf{P}}$; in particular we show that $\mathsf{L}_2^{\mathsf{P}}$ contains MA and that its exponential analogue $\mathsf{L}_2^{\mathsf{E}}$ requires $2^n/n$ size circuits. Both of these are consequences of our reduction from Range Avoidance to LOP.

On the negative side we prove that the algorithms developed in [CHR24, Li24] cannot be extended to *Strong Range Avoidance*, a problem considered in the same paper which first introduced Range Avoidance [KKMP21]. In this problem we are given a circuit $C : \{0,1\}^n \setminus \{0^n\} \to \{0,1\}^n$, and once again seek a point outside its range. We give a separation in the decision tree (oracle) model showing that this problem cannot be solved in $\mathsf{FP}_{||}^{\Sigma_2^{\mathsf{P}}}$, which in particular rules out all of the new kinds of algorithms considered in [CHR24, Li24]. This black box separation is derived from a novel depth 3 AC^0 circuit lower bound for a *total search problem*, which we believe is of independent interest from the perspective of circuit complexity: we show that unlike previous depth 3 lower bounds, ours *cannot be proven* by reduction from a decision problem, and thus requires new techniques specifically tailored to total search problems. Proving lower bounds of this kind was recently proposed by Vyas and Williams in the context of the original (Weak) Avoid problem [VW23].

*Columbia University. oliver.korten@columbia.edu
†Columbia University. tonipitassi@gmail.com

ISSN 1433-8092

1 Introduction

One of the central problems in complexity theory is to prove strong lower bounds on the size of boolean circuits computing some explicit function $f : \{0,1\}^n \to \{0,1\}$; by a classical result of Shannon, most such functions f require circuits of size $2^n/n$. In the original formulation of this problem, "explicit" meant $f \in \mathsf{NP}$, since in this case case proving a lower bound $n^{\omega(1)}$ would separate the classes P and NP. It was soon realized that this problem remains difficult even for much weaker definitions of "explicit," i.e. if we broaden our search for high circuit complexity functions to a much larger uniform complexity class than NP.

Obviously there are decidable problems with maximal circuit complexity, since a Turing machine with sufficient resources may search for the hardest function $f_n : \{0,1\}^n \to \{0,1\}$ by brute force given n, and then compute that function f_n on a given input of length n. Kannan [Kan82] was the first to investigate the precise complexity upper bound of this brute-force construction of a hard language: he observed that the class $\mathsf{E}^{\Sigma_2^P}$, which denotes $2^{O(n)}$-time machines with access to an oracle for a $\mathsf{\Sigma_2^P}$ complete language, contains a function of maximum circuit complexity, precisely because this class possesses sufficient resources to diagonalize over all low-complexity functions of a given input length. The question of finding the smallest uniform complexity class containing an exponentially hard boolean function came to more prominence following the seminal works of Nisan, Wigderson and Impagliazzo[NW94, IW97], who showed that if one could improve Kannan's upper bound from $\mathsf{E}^{\Sigma_2^P}$ to E, it would imply $\mathsf{BPP} = \mathsf{P}$, i.e. universal derandomization of all polynomial time algorithms.

A recent line of work on the *Range Avoidance* problem ("Avoid") has reformulated this topic in the language of search problems [Kor21, CHR24]. Consider the so-called "truth table generator" $\mathrm{TT}_n : \{0,1\}^\ell \to \{0,1\}^{2^n}$ which takes as input the description of a circuit $C : \{0,1\}^n \to \{0,1\}$ of size s, and outputs the truth table of the function it computes. If $s << 2^n$, then a standard argument shows that we may encode C in a direct way using $< 2^n$ bits, so that $\ell < 2^n$ and the function TT_n is computable in $2^{O(n)}$ time. Therefore, to produce a function of high circuit complexity, it suffices to find a string $y \in \{0,1\}^{2^n}$ outside the range of C. This general problem, where we are given a circuit with more output bits then input bits and wish to find a string outside its range, is called Range Avoidance. By the pigeonhole principle, it is a *total search problem*, meaning it always has solutions. Viewing TT_n as an instance of the Range Avoidance problem we can observe that if Avoid has a polynomial time algorithm then E requires exponential-size circuits. For larger complexity classes, it was shown in [Kor21] (based on an earlier result in [Jeř04]) that solving Avoid and proving circuit lower bounds are actually *equivalent problems*: for exponential-time classes at least as large as E^{NP}, proving an exponential circuit lower bound is equivalent to solving Avoid in the polynomial-time analogue of that class. Beyond constructing hard boolean functions, algorithms for Range Avoidance have a host of further applications in *explicit constructions* of pseudorandom objects [Kor21, GLW22].

This perspective was crucially used in recent breakthrough works of Chen, Hirahara, Ren and Li [CHR24, Li24], who gave the first unconditional improvement to Kannan's classical result[1]. These works showed that the classes $\mathsf{S_2^P} \subseteq \mathsf{ZPE^{NP}} \subseteq \mathsf{\Sigma_2^E}$ all contain a function of circuit complexity $2^n/n$, which is within a $(1 \pm o(1))$ factor of the maximum possible circuit complexity of a boolean function. The results are established by giving a new algorithm for Avoid which runs in the class $\mathsf{FS_2^P}$, the functional variant of the decision class $\mathsf{S_2^P}$.

[1]This is the first improvement on finding the smallest complexity class with an *exponential* circuit lower bound. If the goal is to prove merely superpolynomial lower bounds, lower bounds for smaller classes can be established using Karp-Lipton theorems [KL80], an approach pioneered in Kannan's original paper [Kan82]. This method can at best show "sub-half-exponential" lower bounds, and only for infinitely many input lengths.

The authors of [CHR24, Li24] made progress on a classical lower bound problem by discovering a new kind of algorithm for the total search problem Range Avoidance. This search problem lies in an unusual complexity class which was only first investigated a few years ago by [KKMP21]: it is a member of $\mathsf{TF\Sigma_2^P}$, the class of total search problems in the *second level* of the polynomial hierarchy. Since the introduction of $\mathsf{TF\Sigma_2^P}$ in [KKMP21], there has been no follow up work developing a structural classification of the problems therein, despite the considerable attention that has been devoted to Range Avoidance [Kor21, CHR24, Kor22, CHLR23, ILW23, GGNS23, CL23, GLW22] and explicit construction problems more generally. The work of [CHR24, Li24] indicates that some problems in $\mathsf{TF\Sigma_2^P}$ admit highly nontrivial algorithms, algorithms which have consequences in seemingly unrelated areas of complexity. Do such algorithms exist for all search problems in $\mathsf{TF\Sigma_2^P}$? What more can we say about the relation of Avoid to other problems in this class? The purpose of this work is to address these two questions in particular, and more broadly to further the systematic study of $\mathsf{TF\Sigma_2^P}$ beyond its introduction in [KKMP21], with an emphasis both on *inclusions* and *black box separations*. Prior to our work, no non-trivial separations between problems in $\mathsf{TF\Sigma_2^P}$ were known.

$\mathsf{TF\Sigma_2^P}$ has a more famous cousin one level down in the polynomial hierarchy, the class TFNP of total NP search problems. In contrast to $\mathsf{TF\Sigma_2^P}$, TFNP has received a thorough investigation over the past three decades and its structure is rather well understood in comparison. A major distinction we reveal between $\mathsf{TF\Sigma_2^P}$ and TFNP is that the former has a plethora of *resource-constained* subclasses, many of which can be separated from one another by explicit and natural search problems. By "resource-constrained subclass" we mean a class of search problems characterized by the existence of some resource-constrained algorithm which can find a solution. One example we will see is the class $\mathsf{psFZPP^{NP}}$, the class of problems where for every input x, there is a canonical solution y_x which is output with high probability by some polynomial time NP-oracle algorithm. Another class is $\mathsf{FP}^{\Sigma_2^{\mathsf{NP}}\cap\Pi_2^{\mathsf{P}}}$, consisting of those problems which can be solved in polynomial time with oracle access to a language in $\mathsf{\Sigma_2^P}\cap\mathsf{\Pi_2^P}$. We will introduce and study several other such subclasses, and exhibit four natural search problems which exhibit separations (in the decision tree model) between them. This situation is in stark contrast to the known structure of TFNP: while FNP does contain some intermediate resource-constrained classes, such as psFZPP and $\mathsf{FP^{NP\cap coNP}}$ (these are roughly analogous to $\mathsf{psFZPP^{NP}}$ and $\mathsf{FP}^{\Sigma_2^{\mathsf{NP}}\cap\Pi_2^{\mathsf{P}}}$ mentioned above), it is known that in the decision tree model all of these classes collapse to FP. As a result in the decision tree model, the only resource-based distinctions amongst the standard TFNP problems is the distinction between FP and the rest of TFNP.

1.1 Overview of Main Results

A search problem is defined by a relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$, where for each "instance" x we say y is a "solution for x" if $(x,y) \in R$; the relevant task is to find a solution given an instance. We say that a search problem is *total* if every instance has a solution. A defining feature of search problems which distinguishes them from decision problems is that a given instance may have many different solutions. Indeed, if a search problem has a unique solution on every instance, then it may be equivalently phrased as a decision problem: given (x,i), output the i^{th} bit of the unique solution for x. For many search problems of interest it is not clear how to reduce them to a decision problem of the same complexity. For example given total search problem R we may define the decision problem LexFirst_R, where given an instance (x,i) we must output the i^{th} bit of the lexicographically first solution to x. While LexFirst_R is clearly at least as hard as R, in many cases it will be much harder and a reduction does not seem to exist in the opposite direction.

Say that $\mathcal{A}$ is a deterministic algorithm solving some search problem R. Then $\mathcal{A}$ naturally

associates to each instance x a *canonical solution* $y_x := \mathcal{A}(x)$ which it outputs. In particular we can think of $\mathcal{A}$ as *defining a second search problem* R' with $(x, y) \in R'$ iff $\mathcal{A}(x) = y$; if $\mathcal{A}$ solves the original search problem R, then R is reducible to R' since $(x, y) \in R' \to (x, y) \in R$. Now, if the algorithm $\mathcal{A}$ lies in some restricted complexity class, then this places the same complexity upper bound derived search problem R'. The point of this is that when we have some nontrivial deterministic algorithm solving a search problem R, we may think of it as giving a reduction from R to a search problem with *unique solutions*.

For most of the classical problems in the class TFNP it is believed that there is *no decision problem* which captures their complexity precisely, a conjecture supported by black box separations [BCE+98a]. More generally it is believed that for most of the standard problems $R \in \mathsf{TFNP}$, any unique-solution problem R' which we can reduce R to must lie *outside of* TFNP[2]. The new results in [Li24, CHR24] reveal that the situation for Range Avoid is different: because their Range Avoidance algorithms are deterministic and are upper bounded inside $\mathsf{TF}\Sigma_2^{\mathsf{P}}$, they imply that Range Avoidance is reducible to a problem in $\mathsf{TF}\Sigma_2^{\mathsf{P}}$ with unique solutions. To make this discussion more formal we need to define the class $\mathsf{TF}\Sigma_2^{\mathsf{P}}$ and its *unique solution* subclass $\mathsf{TFU}\Sigma_2^{\mathsf{P}}$:

Definition 1 ($\mathsf{TF}\Sigma_2^{\mathsf{P}}$ and $\mathsf{TFU}\Sigma_2^{\mathsf{P}}$)**.** *A polynomially-bounded*[3] *search problem R lies in $\mathsf{TF}\Sigma_2^{\mathsf{P}}$ if it is a total search problem, and there exists a coNP verifier V so that $(x, y) \in R$ if and only if $\mathrm{V}(x, y) = 1$. We say $R \in \mathsf{TFU}\Sigma_2^{\mathsf{P}}$ if moreover every instance has a unique solution.*

In this terminology Li's result implies the following: Range Avoidance is polynomial-time reducible to a problem in $\mathsf{TFU}\Sigma_2^{\mathsf{P}}$. His result is in fact a significant strengthening of this, but for now we focus on this particular consequence, which seems quite unintuitive on the surface: given an instance $f : \{0,1\}^n \to \{0,1\}^{n+1}$ of Range Avoidance, there are at least 2^n distinct solutions, and it is not clear how to narrow down to any particular solution which is "more special" then the others. Our first contribution is to clarify this result in the following way: we introduce a *natural* $\mathsf{TF}\Sigma_2^{\mathsf{P}}$ search problem, whose containment in $\mathsf{TFU}\Sigma_2^{\mathsf{P}}$ is obvious from the definition, and then show that Range Avoidance reduces to this problem:

Definition 2 (Linear Ordering Principle (LOP))**.** *Given $\prec$: $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ specified by a boolean circuit, find a witness that $\prec$ does not define a total ordering on $\{0,1\}^n$, or else find the minimal element in the ordering it defines. A witness that $\prec$ does not define a total order consist of $x, y, z \in \{0,1\}^n$ such that one of the following holds: (a) $x \prec x$; (b) $x \neq y$, $x \not\prec y$, and $y \not\prec x$; (c) $x \prec y \prec z$ and $x \not\prec z$.*

While this problem does not literally have unique solutions as stated, it has a property which we show is morally equivalent. Observe that one type of solution in this problem is easier to verify then the other: a witness that $\prec$ fails to define a linear order can be verified in polynomial time, whereas a candidate minimum element can only be verified in coNP. Moreover, if an instance has no easily verifiable solutions, then it has a unique solution. This follows from the fact that every linear order has a unique minimal element. We summarize this by saying that LOP has *essentially unique solutions*; a very easy argument (Section 2, Lemma 3) shows that any problem with essentially unique solutions is reducible to a problem in $\mathsf{TFU}\Sigma_2^{\mathsf{P}}$. We then prove:

Theorem 1. *Range Avoidance is polynomial time reducible to LOP.*

[2]This is equivalent to saying that we believe these standard problems, e.g. PPAD or PPP, not to lie in $\mathsf{FP}^{\mathsf{NP}\cap\mathsf{coNP}}$.

[3]The "polynomially bounded" condition just means we are restricting attention to search problems whose solutions have polynomially bounded length.

We believe this result goes a long way in explaining the new upper bounds for Range Avoidance. In particular our reduction isolates the two key steps in Li's algorithm which allow us to single out a special *low complexity canonical solution* for Range Avoidance: the first step prepares a special subset of solutions using a certain tree-like iteration construction, and the second step singles out a fixed canonical solution among these by defining a certain *total ordering* on these special solutions. Recall our comment under Definition 1 that Li's result is in fact stronger than a reduction from Range Avoidance to a problem in $\mathsf{TFU}\Sigma_2^\mathsf{P}$: more generally he shows that Range Avoidance lies in the complexity class FS_2^P (we will review the definition of this class in Section 2). Our result subsumes this upper bound as well, since another very direct argument shows that LOP lies in FS_2^P. Thus our result gives the current best upper bound on the Range Avoidance problem, and hence on the Kannan's classical problem of constructing the truth table of a hard boolean function.

At this point we have seen that there are search problems in $\mathsf{TF}\Sigma_2^\mathsf{P}$ which seem on the surface to have no distinguished solutions, but which nonetheless can be reduced to problems in $\mathsf{TFU}\Sigma_2^\mathsf{P}$ by some highly non-obvious means. This naturally points to the following question: are all problems in $\mathsf{TF}\Sigma_2^\mathsf{P}$ reducible to $\mathsf{TFU}\Sigma_2^\mathsf{P}$? We give a negative answer in the decision tree model. Our separation is exhibited by the following relative of Range Avoidance which was introduced originally in [KKMP21].

Definition 3 (Strong Avoid)**.** *Given $f : \{0,1\}^n \setminus \{0^n\} \to \{0,1\}^n$, find $y \in \{0,1\}^n \setminus \mathrm{range}(f)$.*

From now on we will refer to Range Avoidance as "Weak Avoid" to distinguish it from Strong Avoid. Our black box separation will be significantly stronger than just showing that Strong Avoid is not reducible to a problem in $\mathsf{TFU}\Sigma_2^\mathsf{P}$; we will show more generally that Strong Avoid cannot be solved by making non-adaptive queries to any language in Σ_2^P, which is equivalent to proving size lower bounds for depth-3 AC_0 circuits solving Strong Avoid:

Theorem 2. *In the decision tree model, Strong Avoid is not in $\mathsf{FP}_{||}^{\Sigma_2^\mathsf{P}}$. More specifically, let $C : \{0,1\}^{(N-1)\log N} \to \{0,1\}^{N^\epsilon}$ be a depth-3 circuit of size 2^{N^ϵ} and let $D : \{0,1\}^{N^\epsilon} \to \{0,1\}^{\log N}$ be an arbitrary postprocessing function, where $N = 2^n = |\{0,1\}^n|$. Then provided ϵ is sufficiently small, $D \circ C$ cannot solve Strong Avoid: there must be some input $f : [N-1] \to [N]$ so that $D(C(f))$ fails to find a $y \notin \mathrm{range}(f)$.*

This immediately implies non-reducibility to $\mathsf{TFU}\Sigma_2^\mathsf{P}$, since any problem $R \in \mathsf{TFU}\Sigma_2^\mathsf{P}$ with unique solutions can be solved with non-adaptive queries to the language $\{(x,i) \mid (x,y) \in R \to y_i = 1\}$ which lies in Σ_2^P. The connection of these kinds of separations to depth 3 AC^0 lower bounds was spelled out in a recent paper of Vyas and Williams [VW23] for the case of *Weak* Avoid: their work established that nontrivial upper bounds for Weak Avoid are equivalent to certain depth 3 circuits solving the so-called "Missing String" problem: given an explicit list of 2^{n-1} n-bit strings, output a string not in the list. The input size here is $\approx N = 2^n$, and the question is whether a depth 3 circuit exists of size polynomial or quasipolynomial in N. Note that the Missing String problem is simply the black-box variant of Weak Avoid. Li's result showed that quasipolynomial size depth-3 circuits for this problem actually *do exist*, solving the original question of Vyas and Williams in the positive. We show that if the problem is modified so that the list of strings has length $N-1$ rather then $\frac{N}{2}$, then depth-3 circuits require exponential size to solve this problem. We note that our lower bound holds against a stronger class of circuits than what was originally considered by Williams and Vyas: in their model the depth 3 circuit is of the form $C : \{0,1\}^{(N-1)\log N} \to \{0,1\}^{\log N}$ and must output the exact solution to Avoid. Here we allow C to output an arbitrary string in $\{0,1\}^{N^\epsilon}$, which can then be postprocessed arbitrarily to construct a solution to Avoid.

By a simple reduction, we also obtain quasipolynomial depth-3 circuit size lower bounds even for *moderately weak* Avoid instances with domain $[N]$ and codomain $[N+N/\log^{O(1)} N]$. This result gives a complete characterization of the degree of "Weakness" necessary to obtain quasipolynomial size depth 3 circuits: if the codomain has size $N + N/\log^{O(1)} N$ then circuits of quasipolynomial size suffice, and if its size is $N + N/\log^{\omega(1)} N$ then they do not; see Lemma 9 for details.

Our last main result exhibits a more fine-grained separation amongst the subclasses of $\mathsf{TF\Sigma_2^P}$. Above we have highlighted one important distinction, between the problems which are reducible to $\mathsf{TFU\Sigma_2^P}$ and those which aren't. However both of the problems we've seen so far which reduce to $\mathsf{TFU\Sigma_2^P}$ also have an additional property: they are solvable by a polynomial time randomized algorithm using a NP oracle. For Weak Avoid this follows from its definition, while for LOP it follows from its containment in the class $\mathsf{FS_2^P}$. We show that this is not possible for all problems reducible to $\mathsf{TFU\Sigma_2^P}$. Our separation is exhibited by the following natural search problem:

Definition 4 (Strong 1-1 Avoid)**.** *Given $f : \{0,1\}^n \setminus \{0^n\} \to \{0,1\}^n$, find a pair $x \neq y$ in $\{0,1\}^n \setminus \{0^n\}$ such that $f(x) = f(y)$, or else $y \in \{0,1\}^n \setminus \mathrm{range}(f)$,*

Observe that Strong 1-1 Avoid enjoys the same property as LOP of having *essentially unique solutions*: it is easy to verify the collision solutions $f(x) = f(y) \wedge x \neq y$, and any instance with no collision solutions has a unique solution. This follows from the fact that any injective function $f : [N] \to [N+1]$ misses exactly one point in its codomain. Hence, like Weak Avoid and LOP, the problem Strong 1-1 Avoid is reducible to $\mathsf{TFU\Sigma_2^P}$. However we prove the following lower bound:

Theorem 3. *In the decision tree model, Strong 1-1 Avoid is not solvable in $\mathsf{FBPP^{NP}}$.*

Aside from revealing further the structure of $\mathsf{TF\Sigma_2^P}$, this result yeilds a new separation for *decision classes* which was not previously known:

Theorem 4. *In the decision tree model, $\mathsf{\Sigma_2^P \cap \Pi_2^P \not\subseteq BPP^{NP}}$. In particular $\mathsf{\Sigma_2^P \cap \Pi_2^P \not\subseteq S_2^P}$.*

In the next two subsections we will describe these main results in some more technical detail. A diagram of the structure of $\mathsf{TF\Sigma_2^P}$ and our main results is given in Figure 1; some classes in this diagram will not be defined until Section 2.1.

1.2 $\mathsf{AC^0}$ Lower Bounds and Class Separations

Our two main separations show that Strong Avoid has no non-trivial upper bound inside of $\mathsf{TF\Sigma_2^P}$ (Theorem 2), and that Strong 1-1 Avoid has no randomized NP-oracle algorithm (Theorem 7). The first lower bound is the more involved of the two, and requires proving a novel depth 3 AC^0 circuit lower bound for a total search problem, which appears to be the first circuit lower bound of this kind.

Theorem 2 yields a very fine-grained separation for the Strong Avoid problem. It is easy to construct a depth 4 circuit $C : \{0,1\}^{(N-1)\lceil \log N \rceil} \to \{0,1\}^{\lceil \log N \rceil}$ solving Strong Avoid, where moreover the bottom fan-in is only $O(\log N)$ [VW23]. More strongly, it is possible to construct a depth $O(\log N)$ decision tree, which at each step queries a depth 3, $\mathrm{poly}(N)$ size circuit on the input f, and at each leaf outputs a correct solution $y \notin \mathrm{range}(f)$. Our lower bound can be interpretted as saying that if such a decision tree is forced to be non-adaptive, then either the circuits it queries at each step must grow to exponential size, or else the number of queries must grow to $N^{\Omega(1)}$. This also contrasts the situation with Weak Avoid, where as mentioned above, Li's construction gives depth-3 $\mathsf{AC_0}$ circuits of size $N^{O(\log N)}$ for solving Weak Avoid.

We believe our lower bound is of independent interest in circuit complexity. In particular, we give a very precise depth-3 lower bound for a *total search problem*. It is of course possible to

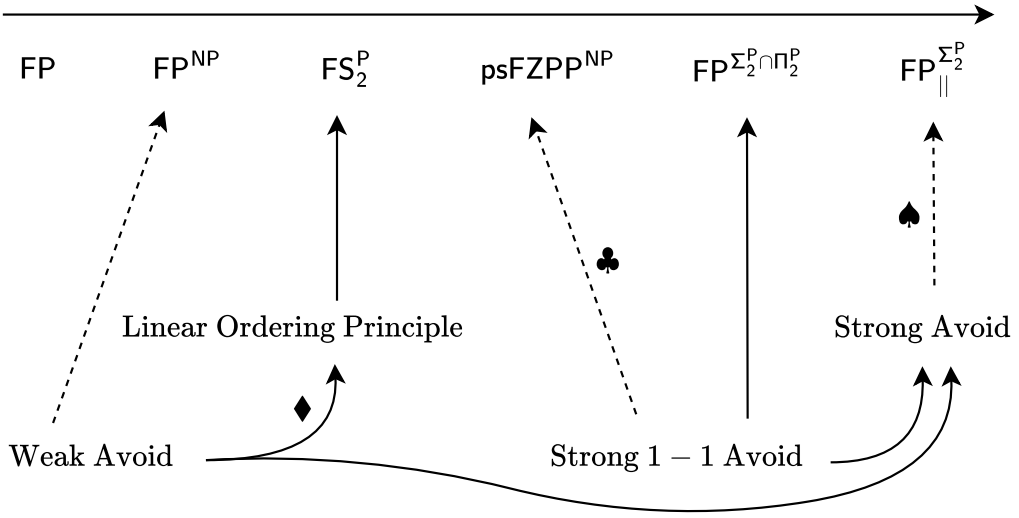

Figure 1: Inclusion diagram of relevant classes and search problems. Solid arrows represent the inclusion of the class at the base of the arrow into the class at its tip. Dotted arrows indicate non-inclusion of the base class into the tip class in the decision tree model. The main linear axis of classes along the top are all included from left to right, indicated by the long solid arrow above them. Our major results are the separations marked with ♠, ♣ and the inclusion marked with ♦. $\mathsf{TFU}\Sigma_2^\mathsf{P}$ essentially corresponds to $\mathsf{FP}^{\Sigma_2^\mathsf{P}\cap\Pi_2^\mathsf{P}}$, see Lemma 2.

construct contrived examples of total search problems which are hard for AC^0 circuits, for example "given $x \in \{0,1\}^n$ output $b \in \{0,1\}$ such that Parity$(x) = b$;" in such examples we can derive hardness of the search problem *by reduction from a decision problem.* In contrast, the lower bound we show here *cannot be established by reduction from any decision problem.* This follows from the fact that any decision problem which is reducible to a $\mathsf{TF}\Sigma_2^\mathsf{P}$ search problem lies in $\Sigma_2^\mathsf{P} \cap \Pi_2^\mathsf{P}$. In the decision tree model, this means that any language which can be solved by a small depth decision tree querying instances of Strong Avoid has both Σ_2^P and Π_2^P circuits. However, the lower bound we are trying to show for Strong Avoid rules out the existence of any $\mathsf{FP}^{\Sigma_2^\mathsf{P}\cap\Pi_2^\mathsf{P}}$ algorithm for Strong Avoid. Phrased more succinctly, our lower bound establishes that Strong Avoid is harder then *any* decision problem which can be reduced to it, which by definition means we cannot establish the lower bound itself by reduction from a decision problem. In light of this, to prove Theorem 2 we must develop new AC^0 lower bound techniques which are specially tailored to total search problems.

Our second main lower bound (Theorem 3) places Strong 1-1 Avoid outside of $\mathsf{FBPP}^\mathsf{NP}$. Recall that Strong 1-1 Avoid has the property of having *essentially unique solutions* and is thus reducible to a problem in $\mathsf{TFU}\Sigma_2^\mathsf{P}$. We will see (Lemma 2) that this means it is reducible to a *decision problem* in $\Sigma_2^\mathsf{P} \cap \Pi_2^\mathsf{P}$. Combining this with the above lower bound we obtain Theorem 4, which separates the decision tree class $\Sigma_2^\mathsf{P} \cap \Pi_2^\mathsf{P}$ from BPP^NP and in particular from S_2^P. This is in contrast to the situation for $\mathsf{NP} \cap \mathsf{coNP}$, which is known to collapse to P in the decision tree model. This improves a previous result of Fortnow and Yamakami [FY96] who showed that $\Sigma_2^\mathsf{P} \cap \Pi_2^\mathsf{P} \not\subseteq \mathsf{P}^\mathsf{NP}$ in the decision tree model.

1.2.1 Lower Bound Methods

A main technical ingredient in both lower bounds is a new Switching Lemma (Lemma 6) specialized for Avoid. Switching lemmas have been used for both circuit lower bounds for computing functions,

and in proof complexity to prove lower bounds on the size of proofs of hard tautologies (e.g., [Bea94, Raz93, BIK$^+$92]). Between the two, our argument bears a stronger resemblance to the second, however there are some key conceptual differences. The basic idea behind all switching lemmas is to show that under a random restriction ρ (from a suitable distribution) a low-width DNF is likely to be represented by a low-depth decision tree. Since a low-depth decision tree representation for a function f implies that both f as well as its negation can be represented by low-width DNFs, this in turn allows us to collapse an AND of low-width DNFs into a single low-width DNF, thus reducing the circuit depth by 1.

A major difference between various switching lemmas is the choice of distribution over restrictions, and the way in which the decision tree *represents* a DNF. In the original Switching Lemmas used to prove AC_0 lower bounds for parity, the restrictions are simply uniformly random partial restrictions, and the notion of represents is with respect to *every* input. That is, the decision tree computes the same function as the DNF.

In the case of switching lemmas used in proof complexity to prove AC^0-Frege lower bounds for the $[N] \to [N+1]$ pigeonhole principle, we think of the input as specifying a purported 1-1 function from $[N]$ to $[N+1]$ (which cannot actually exist if N is finite). The chosen distribution over restrictions are partial 1-1 matchings from $[N]$ to $[N+1]$, and a low-depth "PHP decision tree" in this context can make queries to a pigeon or to a hole at each vertex, and every path in the tree corresponds to a partial matching. Since, in reality, no 1-1 function from $[N]$ to $[N+1]$ exists, these trees do not represent the original DNF in any standard way. However, if the input variables instead corresponded to a total 1-1 assignment from $[N]$ to $[N]$ (which do exist), then we can apply the same PHP Switching Lemma to prove that under a random partial 1-1 restriction, a low-width DNF is likely to convert to a low-depth PHP decision tree, which now represents the DNF in the sense that it agrees with the DNF on all input assignments that correspond to 1-1 mappings from $[N]$ to $[N]$.

In the case of Avoid, we have to modify the way of constructing a decision tree associated with a DNF so that the decision tree represents the original DNF in the sense that they are truth-functionally equivalent with respect to all 1-1 input functions from $[N]$ to $[M]$, where now M is strictly larger than N. To achieve this, we modify the notion of pigeonhole decision trees as follows. As in the original PHP Switching Lemma in proof complexity, each node of our decision tree will query either a pigeon or a hole. When a pigeon is queried at a node, we allow edges for all possible holes that it could be mapped to. But when a hole is queried, now we have to allow for the possibility that this hole is unmapped: in addition to allowing edges for each pigeon that could map to this hole, we allow an extra edge corresponding to the case where nothing maps to this hole. With this modification, our pigeonhole decision trees will represent the original DNF with respect to all 1-1 inputs from $[N]$ to $[M]$.

Another crucial distinction is how we use the Switching Lemma to reduce the depth of the circuit by one. In the proof complexity setting for the pigeonhole principle lower bounds, we think of N as infinite, and therefore with respect to 1-1 inputs, a DNF can be written as a low-width *matching disjunction*, where each term in the disjunction corresponds to a partial 1-1 function (or matching) from $[N]$ to $[N]$. After applying the PHP Switching Lemma and a union bound, each matching disjunction f (under ρ) becomes a low-depth "matching decision tree", enabling a reduction in the overall circuit depth by one. To summarize, in the classical PHP Switching Lemma, the underlying depth-2 subcircuits are always low-width matching disjunctions, both before and after each application of the PHP Switching Lemma. In our case, the underlying depth-2 subcircuits are not of the same type before and after applying our Pigeonhole Switching Lemma; a consequence is that our Switching Lemma *cannot be applied twice*. This is not a defect of our method, but rather a necessary feature of any technique here, since our search problem *can be*

solved by circuits of one higher depth. More specifically, we show that initially the bottom depth-2 subcircuits of $\mathcal{C}$ can be expressed as low-width matching disjuncdtions. But after applying the Pigeonhole Switching Lemma, and subsequent depth reduction, the new depth-2 subcircuits become *hole disjunctions* which are a generalization of matching disjunctions, where now each term in the DNF can specify not only a partial 1-1 matching, but also a subset of holes that are unmapped. With these appropriate modifications, our proof of the switching lemma is similar to previous proofs.

Equipped with the Pigeonhole Switching Lemma, we can give the high level view of both proofs. We start with Theorem 2. To prove Theorem 2, we would like to restrict attention to the class of 1-1 input functions from $[N]$ to $[N+1]$. However if we truly restrict ourselves to 1-1 functions, then a lower bound is not possible by Lemma 4: for these inputs, there is a unique solution, and therefore a polynomial-size depth-3 circuits can easily check whether the unique solution has its i^{th} bit equal to 1 or to 0 and hence solve the problem unconditionally. To circumvent this barrier, we will prove a strengthening of Theorem 2, by giving a lower bound for Avoid on input functions $f : [N] \to [M]$, where M is larger than $N+1$. By enlarging the range of f, we can focus our attention of f's that are 1-1 since now for every input, there are at least $M - N$ distinct solutions. Note that this implies the lower bound stated above (with $M = N+1$), since there is a direct reduction from Avoid on instances $[N] \to [M]$ to instances $[N] \to [N+1]$: we simply map every element of $[M] \setminus [N+1]$ to the element $N+1$. Observe that this reduction does not preserve injectivity.

Now assume there exists small-size s depth-3 circuit $\mathcal{C}$ computing Strong Avoid on 1-1 functions from $[N]$ to $[M]$, $M >> N$. We can first apply a standard argument (the Width Reduction Lemma 8) so that we can assume that the bottom-level fanin of $\mathcal{C}$ is at most $O(\log s)$. After this step, we can assume that $\mathcal{C}$ is a size-s, depth-3 circuit, where the bottom depth-2 subcircuits are low-width matching disjunctions. Next we apply our Pigeonhole Switching Lemma (as discussed above) which will guarantee that there exists a matching restriction ρ such that under ρ, all depth-2 matching disjunctions in $\mathcal{C}$ will convert to low-depth pigeonhole decision trees. This will allow us to reduce the overall circuit depth by 1, and afterwards each output bit of $\mathcal{C}$ will be computed by a low-width *hole disjunction*. As discussed above, a hole disjunction is a type of DNF that generalizes matching disjunctions: each term t in the hole disjunction can be viewed as partial information about the input f. The partial information consists of two parts: (i) first, t_1 specifies a small partial matching, pairing up some pigeons in $[N]$ to some holes in $[N+1]$; (ii) secondly, t_2 specifies a small set of holes (disjoint from the holes mentioned in t_1 that are not in the range of f). It remains to prove a lower bound for circuits $\mathcal{C}$ for solving Avoid, where each output bit is specified by a low-width hole disjunction. This is also accomplished using a kind of restriction, but rather then choosing it at random we apply a careful deterministic process involving a novel covering argument. This step is somewhat reminiscent of early proofs of the Switching Lemma (e.g., [FSS84, Ajt83]).

The proof of Theorem 3 again uses the Pigeonhole Switching Lemma, together with a direct argument. We want to prove depth lower bounds for $\mathsf{FBPP}^{\mathsf{NP}}$, which informally are randomized decision trees of small height which, instead of querying variables, are allowed to query the value of an arbitrary low-width DNF over the inputs. By Yao's minimax principle, it suffices to prove that any low-depth P^{NP} decision tree cannot solve Avoid with probability 2/3, with respect to the uniform distribution of 1-1 functions. We think of this distribution in the following way: first sample a uniform *partial* 1-1 assignment ρ, then sample a uniform extension of ρ to a total assignment. Applying our Pigeonhole Switching Lemma and a union bound, we can argue that with high probability over the first choice of ρ, all of the NP queries in our P^{NP} decision tree T can be simplified to small depth pigeonhole decision trees, which overall allows T to be replaced by a low depth pigeonhole decision tree. It then remains only to argue that a pigeonhole decision tree

of low depth cannot solve 1-1 Strong Avoid with non-trivial probability on a uniform extension f of ρ, which can be accomplished with a direct argument.

1.3 Linear Ordering Principle

We now discuss in more detail our results on the newly defined Linear Ordering Principle problem, abbreviated LOP. Recall that our main result here is Theorem 1, which says that Weak Avoid is polynomial time reducible to LOP. The proof follows much of the high level structure of Li's result placing Weak Avoid in FS_2^P, with some key modifications. Roughly speaking, Li's proof shows that given an instance of Avoid $f : \{0,1\}^n \to \{0,1\}^{n+1}$, we can define a comparison relation $\sqsubset$ on $\{0,1\}^{\mathrm{poly}(n)}$, so that for some unique distinguished element $\pi^* \in \{0,1\}^{\mathrm{poly}(n)}$ we have $\pi^* \sqsubset \pi$ for all $\pi \neq \pi^*$, and π^* contains a solution to the original Avoid instance. In our case we need to define a similar comparison relation, which in addition *globally* acts as a total order on $\{0,1\}^{\mathrm{poly}(n)}$. To explain the argument more clearly we split the reduction into two parts. We first introduce an intermediate search problem called Forest Termination and reduce Weak Avoid to this problem, then we reduce Forest Termination to LOP. We note that our proof, as well as Li's, also bears a strong resemblance to the work of [PWW88] who gave the first proof of the weak pigeonhole principle in the bounded arithmetic theory T_2.

Our subsequent results show some appealing structural properties of the complexity class defined by reducibility to LOP. We start by proving closure under a broad class of reductions:

Theorem 5. *Any search problem which has a polynomial time P^{NP} Turing reduction to LOP also has a polynomial time many-one reduction to LOP.*

Combining this closure property with the fact that LOP has essentially unique solutions, we are able to conclude that LOP is *equivalent* in complexity to a decision problem. In particular we can define a decisional complexity class L_2^P for which LOP is the "complete problem" (despite being a search problem and not a language). We start by presenting a machine-based definition of L_2^P:

Definition 5. *A language L is in the complexity class L_2^P if there is a polynomial time relation $R : (\{0,1\}^*)^3 \to \{0,1\}$ and a polynomial p, so that for all x, $R(x,\cdot,\cdot)$ defines a total order on $\{0,1\}^{p(n)}$ whose minimal element a has $a_1 = L(x)$.*

We then have the following equivalent characterizations:

Theorem 6. *The following are equivalent for a language L:*

1. *$L \in \mathsf{L}_2^\mathsf{P}$*
2. *L is polynomial time many-one reducible to LOP.*
3. *L is P^{NP}-Turing reducible to LOP.*

Conversely, the search problem LOP is polynomial time truth table reducible[4] to a language in L_2^P.

We mentioned in passing before that LOP is easily shown to lie in the class FS_2^P, the functional analogue of the decision class S_2^P (this will be shown in Lemma 5. The same reason shows that $\mathsf{L}_2^\mathsf{P} \subseteq \mathsf{S}_2^\mathsf{P}$. For those unfamiliar with the somewhat unconvential class S_2^P, this is a complexity class introduced independently by Russell-Sundaram and Canetti [RS98, Can96]. Their goal was to identify the smallest class in the polynomial hierarchy which is sufficient to capture randomized

[4]This is essentially the most restrictive reduction possible when reducing a search problem to a language.

algorithms, in particular BPP and MA. Beyond this purpose the class rarely appears, and so far no natural problems have been exhibited which lie in S_2^P and not one of its more traditional subclasses (such as BPP or NP). In addition it seems that S_2^P does not have a complete problem, due to its definition involving a promise. We have identified here a subclass L_2^P of S_2^P, which is characterized exactly by a simple and natural total search problem, and which nonetheless maintains the interesting properties that motivated the original definition of S_2^P:

Theorem 7.

1. $\mathsf{P}^{\mathsf{NP}} \subseteq \mathsf{L}_2^\mathsf{P}$ *and* $\mathsf{BPP} \subseteq \mathsf{MA} \subseteq \mathsf{L}_2^\mathsf{P}$
2. $\mathsf{L}_2^{\mathsf{E}}$[5] *contains a language of circuit complexity* $2^n/n$.

In each case, the result stated for L_2^P was previously known to hold for S_2^P and is now shown to be inherited by the more natural subclass L_2^P. The only interesting property that is known of S_2^P which we were unable to prove for L_2^P is the Karp-Lipton theorem; we discuss this further Sections 1.4 and 4.

1.4 Open Problems

We conclude our introduction with a few interesting problems which remain open. The first is rather broad:

Problem 1. *Show any additional inclusions or black-box separations which are not implied by the arrows in Figure 1.*

We specifically highlight the following:

Problem 2. *Is there an* $\mathsf{FP}^{\mathsf{NP}}$ *Turing reduction from LOP to Weak Avoid?*

Our interest in this problem is the following observation:

Observation 1. *Say that Linear Ordering Principle is* $\mathsf{FP}^{\mathsf{NP}}$ *Turing reducible to Weak Avoid. Then there is a particular language* L *of circuit complexity* $\geq 2^n/n$*, and a deterministic* NP *oracle algorithm* $\mathcal{A}$ *running in time* $2^{O(n)}$*, such that given oracle access to any language* L' *of circuit complexity* $2^{\Omega(n)}$*,* $\mathcal{A}^{L'}$ *computes* L*.*

This follows by composing the reduction of Weak Avoid to LOP, which produces a unique solution, with the (purported) second reduction from LOP back to Weak Avoid. Such a consequence would be rather surprising and interesting purely from the perspective of circuit complexity.

Next we highlight the problem of better clarifying the relationship between L_2^P and S_2^P:

Problem 3. *Does* $\mathsf{L}_2^\mathsf{P} = \mathsf{S}_2^\mathsf{P}$*? Can they be separated in the decision tree model? Does* L_2^P *satisfy a Karp-Lipton theorem?*

For this problem, we would say L_2^P "satisfies a Karp-Lipton theorem" if one could unconditionally prove the implication "$\mathsf{NP} \in \mathsf{P/poly} \to \mathsf{PH} = \mathsf{L}_2^\mathsf{P}$." A notable property of S_2^P is that it is the smallest complexity class for which this statement is known to hold.

The next problem we highlight is in the realm of depth 3 circuits:

Problem 4. *Does Weak Avoid have depth 3 circuits of polynomial size?*

[5] $\mathsf{L}_2^{\mathsf{E}}$ is the exponential-time analogue of L_2^P, where we replace "polynomial time" with $2^{O(n)}$ time" in its definition

Recall that Li's upper bound is only quasipolynomial, of size around $N^{\log N}$. This problem seems intimately connected to the long-standing open question in proof complexity of whether the Weak Pigeonhole Principle has polynomial size bounded depth Frege proofs; a *quasipolynomial* upper bound was shown by Paris Wilkie and Woods [PWW88] using a very similar technique to Li's, and [MPW02] give a different bounded-depth Frege upper bound of lower depth, but still quasipolynomial size. Despite the strong aesthetic similarities we do not know a formal connection in either direction between these problems.

Lastly, the search problems discussed here have other connections to bounded arithmetic. In particular, the LOP principle has been studied in several papers, within the context of characterizing the strength of Jerebek's bounded arithmetic theory of approximate counting relative to weaker theories, and also as a new avenue for approaching the longstanding problem of separating Buss' T_2 hierarchy by sentences of fixed complexity. Buss, Kolodziejczyk and Thapen [BKT14] observe that the LOP principle is provable in both T^2_2 and in APC_2, and ask whether or not LOP is provable in the weaker theory $T^1_2 + sWPHP$, where $sWPHP$ is the surjective weak pigeonhole principle, and corresponds to the search problem Weak Avoid. Atserias and Thapen [AT13] resolve this question, proving that in the relativized setting, $sWPHP$ does not prove the LOP principle over T^1_2. In fact they prove a stronger result, that $sWPHP$ cannot prove the HOP principle over T^1_2, where HOP is a Σ^b_1 version of LOP. It seems possible that the techniques here could be used to give a negative answer to Problem 2. A relatively unexplored area that is likely to be fruitful is to discover more relationships between natural search problems lying in the second level of the polynomial hierarchy (and higher) and corresponding systems of bounded arithmetic.

2 Preliminaries

2.1 Search Problems, Complexity Classes, and Basic Inclusions

We define here a variety of subclasses inside of $\mathsf{TF}\Sigma^\mathsf{P}_2$, classified according to the computational resources necessary to solve a search problem. We then prove some of the more basic results relating these classes to eachother and to the four main search problems of interest in this work.

Definition 6 (FP^NP, $\mathsf{FP}^{\Sigma^\mathsf{P}_2\cap\Pi^\mathsf{P}_2}$, $\mathsf{FP}^{\Sigma^\mathsf{P}_2}_{||}$)**.** *Let R be a search problem and $\mathcal{C}$ a class of decision problems. We say $R \in \mathsf{FP}^\mathcal{C}$ if there is a language $L \in \mathcal{C}$ and a polynomial time algorithm making queries to L which "solves R:" given x it outputs y such that $(x, y) \in R$. We say $R \in \mathsf{FP}^\mathcal{C}_{||}$ if there is such an algorithm, which moreover makes its queries* nonadaptively*: given an input x it computes in polynomial time a list of queries $z_1, \ldots, z_m$, uses its L-oracle to test in unit time the membership of each z_i in L, and then uses the oracle responses to output an answer in polynomial time.*

We include for reference the following class considered in [Li24, CHR24]:

Definition 7 ($\mathsf{svF}\Sigma^\mathsf{P}_2$)**.** $\mathsf{svF}\Sigma^\mathsf{P}_2$ *is the class of search problems having "singled-valued" $\mathsf{F}\Sigma^\mathsf{P}_2$ algorithms. We say R lies in this class if there is a choice of canonical solution $\{(x, y_x) \in R \mid x\}$ for each input x, a second relation $R' \in \mathsf{TF}\Sigma^\mathsf{P}_2$, and a polynomial time function f so that whenever $(x, z) \in R'$, $f(z) = y_x$.*

In Lemma 1 we relate this class to the others we have defined here. Beyond this we will not need to reference this class further: every time we prove an upper bound for a search problem it will be in a class lower then $\mathsf{svF}\Sigma^\mathsf{P}_2$, and our separation for Strong Avoid will hold even against the larger class $\mathsf{FP}^{\Sigma^\mathsf{P}_2}_{||} \supseteq \mathsf{svF}\Sigma^\mathsf{P}_2$. We next review the randomized classes:

Definition 8 ((ps)$\mathsf{FZPP}^{\mathsf{NP}}$, (ps)$\mathsf{FBPP}^{\mathsf{NP}}$)**.** *A relation R is in $\mathsf{FBPP}^{\mathsf{NP}}$ if there exists a randomized polynomial time algorithm with access to a* SAT *oracle which, given x, outputs y such that $(x, y) \in R$ with probability $\geq 2/3$. If the algorithm always outputs a valid answer or $\perp$ and answers $\perp$ with probability $< \frac{1}{3}$ this places R in the subclass $\mathsf{FZPP}^{\mathsf{NP}}$.*

For a search problem R where each x may have many solutions y, it is possible that a randomized algorithm outputs different correct answers on the same input x as a function of its random coin tosses. If for each x there exists a canonical y_x with $(x, y_x) \in R$ and some randomized algorithm computes $x \mapsto y_x$ with high probability, we say that algorithm is pseudodeterministic; we use the prefix ps$-$ *to denote the pseudodeterministic analogue of a randomized class.*

The last standard complexity class we examine is the functional analogue of $\mathsf{S}_2^{\mathsf{P}}$, defined as follows:

Definition 9 ($\mathsf{FS}_2^{\mathsf{P}}$)**.** *A search problem R is in $\mathsf{FS}_2^{\mathsf{P}}$ if there exists polynomial time algorithm* V *taking three inputs, so that for all x there exists y_x with $(x, y_x) \in R$ so that:*

1. *There exists π_1 such that for all π_2, $\mathrm{V}(x, \pi_1, \pi_2) = y_x$.*
2. *There exists π_2 such that for all π_1, $\mathrm{V}(x, \pi_1, \pi_2) = y_x$.*

Search Problems vs. Function Problems: As discussed in the introduction, some kinds of algorithms for search problems have the property that they associate to each input a *fixed solution* which the algorithm produces on that input. With the exception of the non-pseudodeterministic randomized classes $\mathsf{FBPP}^{\mathsf{NP}}$ and $\mathsf{FBPP}^{\mathsf{NP}}$, each of the classes we have just defined describes algorithms of this sort. These classes are arranged nicely in the following hierarchy, as indicated in Figure 1:

Lemma 1. $\mathsf{FP} \subseteq \mathsf{FP}^{\mathsf{NP}} \subseteq \mathsf{FS}_2^{\mathsf{P}} \subseteq \mathsf{psFZPP}^{\mathsf{NP}} \subseteq \mathsf{FP}^{\Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}} \subseteq \mathsf{svF}\Sigma_2^{\mathsf{P}} \subseteq \mathsf{FP}_{||}^{\Sigma_2^{\mathsf{P}}}$

Proof. All inclusions follow directly from the definition, with the exceptions of $\mathsf{FP}^{\mathsf{NP}} \subseteq \mathsf{FS}_2^{\mathsf{P}}$ which is due to Russell-Sundaram [RS98] and $\mathsf{FS}_2^{\mathsf{P}} \subseteq \mathsf{psZPP}^{\mathsf{NP}}$ which is due to Cai [Cai07]. □

Recall the class $\mathsf{TFU}\Sigma_2^{\mathsf{P}}$, referenced heavily in the introduction, consisting of those $\mathsf{TF}\Sigma_2^{\mathsf{P}}$ search problems with unique solutions. This class is directly associated to $\mathsf{FP}^{\Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}}$ in this hierarchy:

Lemma 2. *The following are equivalent for any $R \in \mathsf{TF}\Sigma_2^{\mathsf{P}}$:*

1. *R is polynomial time reducible to a problem in $\mathsf{TFU}\Sigma_2^{\mathsf{P}}$.*
2. *$R \in \mathsf{FP}^{\Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}}$.*

Proof. Say R is polynomial time reducible to $R' \in \mathsf{TFU}\Sigma_2^{\mathsf{P}}$. So there are polynomial time functions so that for all x, if $(f(x), y) \in R'$ then $(x, g(x, y)) \in R$. Consider the language L defined as follows: $(x, i) \in L$ iff for the unique y with $(f(x), y) \in R'$, we have $y_i = 1$. Then $L \in \Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}$ and g yeilds a reduction from R to L.

In the other direction note that any language $L \in \Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}$ defines a search problem in $\mathsf{TFU}\Sigma_2^{\mathsf{P}}$: given x find $b \in \{0, 1\}$ so that $L(x) = b$. The result follows directly from this fact. □

Recall from the introduction that the problems LOP and Strong 1-1 Avoid do not quite have unique solutions, but come very close. We define this property of having *essentially unique solutions* as follows:

Definition 10 (Essentially Unique Solutions)**.** *We say that a total search problem $R \in \mathsf{TF\Sigma_2^P}$ has "essentially unique solutions" if there are verifiers $\mathrm{V}_1, \mathrm{V}_2$ such that:*

1. *V_1 is testable in polynomial time, while V_2 is testable in coNP.*
2. *For all x, either there exists y so that $\mathrm{V}_1(x, y) = 1$ and $(x, y) \in R$, or else there exists a unique y such that $\mathrm{V}_2(x, y) = 1$ and $(x, y) \in R$.*

We then have:

Lemma 3. *If R has essentially unique solutions then it is polynomial time reducible to a search problem $R' \in \mathsf{TFU\Sigma_2^P}$ which actually has unique solutions. By Lemma 2 this is equivalent to the statement $R \in \mathsf{FP}^{\mathsf{\Sigma_2^P \cap \Pi_2^P}}$.*

Proof. Let $\mathrm{V}_1, \mathrm{V}_2$ witness that R has essentially unique solutions. Consider the search problem R': given x, output either the lexicographically first y such that $\mathrm{V}_1(x, y) = 1$, or else the unique y such that $\mathrm{V}_2(x, y) = 1$. By the definition of $\mathrm{V}_1, \mathrm{V}_2$ we see that R' is a total search problem with unique solutions. Clearly R is polynomial time reducible to R'. We need to show that $R' \in \mathsf{F\Sigma_2^P}$. For a fixed x, say that there exists y with $\mathsf{V}_1(x, y) = 1$, and let y_0 be the lexicographically first such y. Then we can confirm that $(x, y_0) \in R'$ in coNP by confirming $\mathsf{V}_1(x, y_0) = 1$ and that for all $y' < y_0$ we have $\mathsf{V}_1(x, y') = 0$. On the other hand say that there is no y with $\mathsf{V}_1(x, y) = 1$, and let y^* be the unique element with $\mathsf{V}_2(x, y^*) = 1$. Then we can confirm that $(x, y^*) \in R'$ in coNP by checking that for all y we have $\mathsf{V}_1(x, y) = 0$, and using V_2 to confirm $\mathsf{V}_2(x, y^*) = 1$ (recall that V_2 is in coNP by definition). □

We now state formally the claim made in the introduction that LOP and Strong 1-1 Avoid have essentially unique solutions:

Lemma 4. *Linear Ordering Principle and Strong 1-1 Avoid have essentially unique solutions; hence both problems lie in $\mathsf{FP}^{\mathsf{\Sigma_2^P \cap \Pi_2^P}}$.*

Proof. Any injective function $f : [N] \to [N + 1]$ leaves a unique point in $[N + 1]$ out of its range. Every total linear order on a finite set has a unique minimal element. □

We next prove formally that LOP is contained in the class $\mathsf{FS_2^P}$, which follows quite directly from the definitions:

Lemma 5. *Linear Ordering Principle is in $\mathsf{FS_2^P}$.*

Proof. Let $\prec$: $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be an instance of Linear Ordering Principle. We construct the $\mathsf{FS_2^P}$ solver V for $\prec$ as follows. Let $X = \{0,1\}^{3n+1}$ be partitioned so that the first 2^n elements A are identified with $\{0,1\}^n$, and the remaining $> 2^{3n}$ elements B are identified with potential witnesses that $\prec$ fails to define a linear order. Given $\pi_1, \pi_2 \in X$, $\mathrm{V}(\prec, \pi_1, \pi_2)$ behaves as follows:

1. If $\pi_i \in B$ codes a witness that $\prec$ does not define a linear order for some $i \in \{1, 2\}$, we output π_i; if both do then we output the lexicographically first between them.
2. Say $\pi_i \in B$ and $\pi_{i'} \in A$ for $\{i, i'\} = \{1, 2\}$, and π_i is not a witness that $\prec$ fails to define a linear order. In this case we output $\pi_{i'}$.
3. If both $\pi_1, \pi_2 \in A$, we think of them as representing elements a_1, a_2 of $\{0,1\}^n$ and compare them according to $\prec$. If $a_1 \prec a_2$ we output π_1, otherwise we output π_2.

First say that $\prec$ is not a total order. Among all witnesses to this let $\pi \in B$ be the lexicographically first. Then for all $\pi' \in X$, we have $V(\prec, \pi, \pi') = V(\prec, \pi', \pi) = \pi$. On the other hand say $\prec$ is a total order and let $\pi \in A$ correspond to its unique minimal element. Then again for all $\pi' \in X$, we have $V(\prec, \pi, \pi') = V(\prec, \pi', \pi) = \pi$. Thus in all cases V gives an $\mathsf{FS}_2^{\mathsf{P}}$ algorithm solving the Linear Ordering Principle problem on input $\prec$. □

Finally, amongst the 4 search problems studied here we have the two following obvious inclusions which we haven't mentioned yet:

Observation 2. *Weak Avoid and Strong 1-1 Avoid are polynomial time reducible to Strong Avoid.*

2.2 Oracle Separations and the Decision Tree Model

All of the upper bounds and inclusions we show in this paper are unconditional and hold relative to every oracle. Since showing any unconditional separations amongst the classes we have identified would imply $\mathsf{P} \neq \mathsf{NP}$, we can only hope to establish separations in a restricted model. As is standard, our restricted model will correspond to the "decision tree model" of complexity classes, which can be framed either in terms of oracles and Turing machines, or more directly in terms of decision trees and bounded depth circuits, c.f. [CIY97]. We choose here the latter terminology.

All of our search problems are defined in terms of a function or relation specified by a boolean circuit. Take for example the Weak Avoid problem, whose instance is a boolean circuit computing $f : \{0,1\}^n \to \{0,1\}^{n+1}$. Clearly this problem remains total if f were not represented by a small circuit, but instead was an arbitrary function $f : [2^n] \to [2^{n+1}]$. In the decision tree model, the relevant search problem has as its input an arbitrary function $f : [N] \to [2N]$ where $N = 2^n$. For example, the decision tree variant of the simplest complexity class FP then corresponds to algorithms which can access f only by querying its on $\text{poly}(n) = \text{poly}(\log N)$ different inputs $x \in [N]$. If we imagine $f : [N] \to [2N]$ is specified by an assignment $\alpha : \{0,1\}^{[2^n] \times [n+1]}$, where $\alpha_{x,i} = f(x)_i$, then this is equivalent to allowing the algorithm to query $\text{poly}(n) = \text{poly}(\log N)$ variables of the assignment α.

Decision tree analogues of other complexity classes can be defined similarly. It is a standard result that a decision tree separation implies a a separation of the associated Turing machine classes relative to an oracle; more specifically it is equivalent to a separation relative to a so-called "generic oracle" [BCE+98b].

Most of the definitions in our decision tree models will be standard, e.g. bounded depth formulae and DNFs. One decision tree model we study whose definition and notation is less standardized is a $\mathsf{FP}^{\mathsf{NP}}$ decision tree:

Definition 11 ($\mathsf{FP}^{\mathsf{NP}}$ Decision Trees)**.** *Let $f : \{0,1\}^n \to A$ be a function where A is some set. A P^{NP} decision tree T computing f is defined by a binary tree, with each internal node labeled by a DNF formula on the variables $\{x_i \mid i \leq n\}$ and each leaf labeled by a value $y \in A$. On an input $x \in \{0,1\}^n$, we traverse T starting at the root. At each internal node associated to a DNF D, we test if $D(x) = 1$; if so we proceed to the right child of the current node, otherwise we proceed to the left. When we reach a leaf we output the value associated to it; T computes f if the value reached is $f(x)$. We say that T has complexity $\leq r$ if its depth as a tree is at most r, and each DNF associated to its nodes has width at most r.*

We next define the decision tree variant of $\mathsf{FP}_{||}^{\Sigma_2^{\mathsf{P}}}$ which is the subject of our main lower bound Theorem 2:

Definition 12. *Let $R \subseteq \{0,1\}^N \times [M]$ black-box (relativized) search problem. We say that R is in $\mathsf{FP}_{||}^{\Sigma_2^\mathsf{P}}$ if there exist Σ_2^P circuits $\Phi_1, \ldots, \Phi_k$ with $k \leq \text{poly}\log(N)$ and an arbitrary post-processing function $S : \{0,1\}^k \to [M]$ so that for all inputs $f \in \{0,1\}^N$, $(f, S(\Phi_1(f), \ldots, \Phi_k(f))) \in R$. A Σ_2^P circuit is a $2^{\text{poly}\log N}$-size depth 3 circuit with bottom fanout $\leq \text{poly}\log N$.*

In our case the input will be some $f \in \{0,1\}^{N \log N}$ representing a function $f : [N] \to [N+1]$, and the relevant search problem is to output some $y \in [N+1]$ outside its range. The fact that this captures the relativized version of $\mathsf{FP}_{||}^{\Sigma_2^\mathsf{P}}$ is based on the well known equivalence between relativized levels of the polynomial hiearchy and quasipolynomial size bounded depth circuits [FSS84]; in particular the Σ_2^P circuits $\Phi_1, \ldots, \Phi_k$ above correspond to a sequence of non-adaptive Σ_2^P oracle queries.

Note on bottom fanin: The depth 3 circuit model which captures $\mathsf{FP}_{||}^{\Sigma_2^\mathsf{P}}$ has the additional restriction that the bottom level of each depth 3 circuit has fanin $\text{poly}\log N$. Our lower bound will apply also to the stronger model in which the depth 3 circuits are only constrained in their size and not their bottom fanin.

Finally, we make note of one separation indicated in Figure 1 which was shown prior to our work:

Lemma (Wilson and Vyas-Williams[Wil83][VW23])**.** *In the decision tree model, Weak Avoid is not in FP^NP.*

3 Lower Bounds for Pigeonhole Principles

3.1 Pigeonhole Principle Basics

We will be concerned here with search problems where the input is a function $f : [N] \to [M]$ with $M > N$, and the goal is to find an empty pigeonhole of f.

Definition 13. *We use $[M]^{[N]}$ to refer to the set of all functions $f : [N] \to [M]$. We define a set of propositional variables, referred to as "bit variables", given by*

$$\text{BITS}_{N,M} := \{f_{x,i} \mid x \in [N], i \in [\lceil \log M \rceil]\}$$

We associate each truth assignment $\alpha : \text{BITS}_{N,M} \to \{0,1\}$ with the function $f_\alpha : [N] \to [M]$, where where the value α assigns to $f_{x,i}$ indicates the i^{th} bit of $f_\alpha(x)$ in binary; if M is not a power of two then we think of all strings in $\{0,1\}^{\lceil \log M \rceil}$ of binary value exceeding M as being redundant representations of the element M. Similarly we associate every function $f : [N] \to [M]$ with the assignment $\alpha_f : \text{BITS}_{N,M} \to \{0,1\}$ using the same correspondence.

From now on we will often not refer explicitly to assignments $\alpha : \text{BITS}_{N,M} \to \{0,1\}$, only to functions $f : [N] \to [M]$. The relevance of this definition is that, when defining circuits/computational devices whose input is a function $f : [M] \to [N]$, we must specify how the device is able to access/read the input f: typically this will correspond to the ability to read the bits $\text{BITS}_{N,M}$.

For the vast majority of this section we will restrict our attention to functions $f : [N] \to [M]$ which are 1-1 (injective): consequently when we speak about the evaluation of a formula which takes f as an input, we will typically only care about the behavior of that formula on this special class of inputs. We use the following notation to express this:

Definition 14. *Let $\mathcal{F}_{N,M} \subset [M]^{[N]}$ denote the set of all 1-1 functions $[N] \to [M]$. For two predicates $F, G : [M]^{[N]} \to \{0,1\}$, we say that F is equivalent to G (with respect to all 1-1 functions), denoted by $F \equiv G$, if $F(f) = G(f)$ for all $f \in \mathcal{F}_{N,M}$.*

We now introduce notation for describing partial information about an input f. A natural such unit of partial information is a partial assignment $\rho : \mathrm{BITS}_{N,M} \to \{0,1,\star\}$, however for our purposes it will be useful to define additional types of partial information as well:

Definition 15 (Matchings and Hole Restrictions)**.** *A partial matching π is a partially defined 1-1 function from $[N]$ to $[M]$. We use $\mathrm{dom}(\pi), \mathrm{range}(\pi)$ to refer to the domain and range of π, and $\mathrm{nodes}(\pi) = \mathrm{dom}(\pi) \sqcup \mathrm{range}(\pi)$. More generally a "hole restriction" $\tau = \langle \pi, E \rangle$ consists of a partial matching π and a set $E \subseteq [M]$ satisfying $\mathrm{range}(\pi) \cap E = \emptyset$. Let $\mathrm{nodes}(\tau) = \mathrm{nodes}(\pi) \cup E$. We use $|\tau|$ to refer to the value $|\pi| + |E|$. We think of the hole restriction $\langle \pi, E \rangle$ as describing the following partial information about a function $f : [N] \to [M]$: "$f(x) = \pi(x)$ for all $x \in \mathrm{dom}(\pi)$, and $y \notin \mathrm{range}(f)$ for all $y \in E$." For a total assignment $f \in \mathcal{F}_{N,M}$, we say f is compatible with $\langle \pi, E \rangle$, written $f \| \langle \pi, E \rangle$, if the above statement holds. For two hole restrictions $\tau_1 = \langle \pi_1, E_1 \rangle, \tau_2 = \langle \pi_2, E_2 \rangle$, we say they are consistent, written $\tau_1 \| \tau_2$, if there is a total assignment f which is compatible with both. We say τ_1 extends τ_2, written $\tau_1 \supseteq \tau_2$, if $\pi_1 \supseteq \pi_2$ and $E_1 \supseteq E_2$. We will think of a partial matching π as a special kind of hole restriction of the form $\langle \pi, \emptyset \rangle$, and a total 1-1 assignment f as a special hole restriction of the form $\langle f, \overline{\mathrm{range}}(f) \rangle$, where $\overline{\mathrm{range}}(f) = [M] \setminus \mathrm{range}(f)$. In this way we will use the above terminology to define relations between matchings, hole restrictions and total 1-1 assignments, e.g. $f \| \pi$ and $\pi \subseteq \tau$.*

The main subject of this section is circuits $C : \{0,1\}^{|\mathrm{BITS}_{N,M}|} \to \{0,1\}^{\lceil \log M \rceil}$ which solve Avoid, in the sense that $C(f) \notin \mathrm{range}(f)$ for all f, where $C(f)$ denotes feeding C the representation of f as an assignment to its bit variables. We thus need some notation for basic kinds of circuits computing a function of f, which will be hole and matching disjunctions:

Definition 16 (Matching and Hole Disjunctions)**.** *A hole disjunction $\phi = \vee_i \tau_i$ over $[M]^{[N]}$ is defined by a collection of hole restrictions $[N] \to [M]$, which we refer to as the "terms" of ϕ. We say $\phi(f) = 1$, or "f satisfies ϕ," if $f \| \tau_i$ for some term τ_i of ϕ. In this way each hole disjunction is associated with a boolean function $\mathcal{F}_{N,M} \to \{0,1\}$. If $|\tau_i| \leq w$ for all i we say that ϕ has width w. We call ϕ a matching disjunction if all τ_i are partial matchings.*

Recall that we are primarily concerned here with the values a formula takes on 1-1 assignments $f \in \mathcal{F}_{N,M}$. In such a setting we can simplify any circuit so that the bottom two logical layers are a set of matching disjunctions:

Observation 3. *Let D be a DNF formula on the bit variables $\mathrm{BITS}_{N,M}$. Then there is a matching disjunction ϕ so that $D \equiv \phi$. Moreover if D has s terms and width w, then ϕ will have $\leq Ms$ terms and width $\leq \lceil \log M \rceil w$.*

Proof. Say that $D = \vee_i \wedge_j \ell_{i,j}$ is a DNF where each $\ell_{i,j}$ is a literal on $\mathrm{BITS}_{N,M}$. For each term $t_i = \wedge_j \ell_{i,j}$ we may replace it by the matching disjunction:

$$\theta_i = \bigvee_{\pi \in \mathrm{match}(t_i)} \pi$$

where $\mathrm{match}(t_i)$ is the set of minimal partial matchings π so that for each literal $\ell_{i,j} = (\neg)^{\xi} f_{x,b}$ in t_i (with $\xi \in \{0,1\}$), there is an edge $(x,y) \in \pi$ with $y_b = \neg\xi$. Clearly we have that $\theta_i(f) = t_i(f)$ for all 1-1 assignments f, and thus we may replace D with the matching disjunction $\vee_i \theta_i$ without affecting its behavior on 1-1 assignments. Observe that $|\mathrm{match}(t_i)| \leq M|t_i|$ and its width is larger by a factor of at most $\lceil \log M \rceil$. □

Definition 17. *Let $\phi = \tau_1 \vee \ldots \vee \tau_s$ be a hole disjunction with $\tau_i = \langle \pi_i, E_i \rangle$ and let $\kappa = \langle \sigma, U \rangle$ be a hole restriction. We define ϕ restricted by τ, denoted by $\phi \restriction \kappa$, as follows: (i) First, any term $\tau_i \in \phi$ that is inconsistent with κ is set to 0 by κ and thus these terms disappear from $\phi \restriction \kappa$ (we also say they are "killed" by κ). If all terms are set to 0, then $\phi \restriction \kappa = 0$. (ii) Otherwise for any term τ_i consistent with κ, we replace τ_i with $\tau_i \restriction \kappa := \langle \pi_i \setminus \sigma, E_i \setminus U \rangle$. If $\tau_i \restriction \kappa = \langle \emptyset, \emptyset \rangle$ this means κ already satisfies this term; if this happens for any of the τ_i then we set $\phi \restriction \kappa = 1$.*

The last basic computational model acting on inputs $f \in \mathcal{F}_{N,M}$ we consider is a "pigeonhole decision tree":

Definition 18. *A pigeonhole decision tree T over $\mathcal{F}_{N,M}$ is defined by a rooted tree with fanout $\leq M$, with leaves labeled by values from some finite set $\mathcal{Z}$. Each internal node $v \in T$ is labelled by either a "pigeon query" or a "hole query". Pigeon nodes (those that make a pigeon query) are labeled by a query $q(v) \in [N]$ and have $\leq M$ outgoing edges each labeled by a distinct element $(q(v), y) \in [N] \times [M]$. Hole nodes are labeled by a query $q(v) \in [M]$ and have $\leq N + 1$ outgoing edges, which are either labelled by a distinct element $(x, q(v)) \in [N] \times [M]$ or by $q(v) \in [M]$. (The label $q(v)$ corresponds to nothing mapping to pigeon $q(v)$.) In this way, we can associate each node $v \in T$ to some $\langle \pi, E \rangle$ with $\pi \subseteq [N] \times [M]$, $E \subseteq [M]$, consisting of the labels of all edges on the path from the root to v in T. We then require that this pair $\langle \pi, E \rangle$ associated to v is a hole disjunction.*

T is said to be a complete pigeonhole decision tree if for every $f \in \mathcal{F}_{N,M}$ there is a (unique) root to leaf path in T that is consistent with f. In the case that T is complete we may associate T with a function $T : \mathcal{F}_{N,M} \to \mathcal{Z}$, where $T(f)$ is the value at the leaf of T consistent with f. The depth of T is the length of the longest root to leaf path. We will sometimes define pigeonhole decision trees without having specific associated leaf values in mind; in this case we refer to the tree as "unlabeled."

Note: The queries in a pigeonhole decision tree *do not* correspond directly to queries of the underlying bit variables $\mathrm{BITS}_{N,M}$. While it is possible to query $\log M$ of the variables $\mathrm{BITS}_{N,M}$ to determine $f(x)$ for some $x \in [N]$, if we want to determine the preimage of $y \in [M]$ under f (or determine it has no preimage), we would need to query $\approx N \log M$ variables of $\mathrm{BITS}_{N,M}$ to make this determination directly. Instead, a pigeonhole decision tree corresponds more directly to a special kind of P^{NP} decision tree on the bit variables.

We next observe that if T is a complete depth d pigeonhole decision tree with binary leaf values, then we may represent both T and $\neg T$ by a width d hole disjunction:

Observation 4. *Let $F : \mathcal{F}_{N,M} \to \{0, 1\}$ be some predicate. If $F \equiv T$ for some complete pigeonhole decision tree T of depth d, then there exist hole disjunctions ϕ_1, ϕ_2 of width $\leq d$ so that $F \equiv \phi_1$ and $\neg F \equiv \phi_2$.*

We will need the following variant of the Switching Lemma, which says that for any low-width matching disjunction ϕ, is we sample a random partial matching ρ, then with high probability $\phi \restriction \rho$ will have a low-depth, complete pigeonhole decision tree. We defer the proof of the Switching Lemma to Section A.

Definition 19 (Distribution of Partial Restrictions)**.** *Let $\mathcal{M}_K^{N,M}$ be the set of all partial matchings $[N] \to [M]$ with exactly $N - K$ edges. We use $\rho \sim \mathcal{M}_K^{N,M}$ to denote a sample from the uniform distribution on this set. When N, M are clear from context we write $\mathcal{M}_K$ as shorthand for $\mathcal{M}_K^{N,M}$.*

Lemma 6 (Pigeonhole Switching Lemma)**.** *Let $M, N, d \in \mathbb{N}$. Let ϕ be a width-w matching disjunction over $[M]^{[N]}$. If $M - N \leq K \leq \frac{N}{4}$ and N, K, d, w sufficiently large, then*

$$\Pr_{\rho \sim \mathcal{M}_K}[\phi \restriction \rho \text{ has no depth } \leq d \text{ pigeonhole decision tree }] \leq \exp\Big(d(\log wK^5 - \log N^{1/2} + O(1))\Big)$$

3.2 Depth 3 Lower Bound for Strong Avoid

In this section we prove the following circuit lower bound, which is a restatement of Theorem 2.

Theorem 2 (Restated). *There is some absolute constant $\epsilon > 0$ so that the following holds. Let $\Phi_1, \ldots, \Phi_k$ be depth 3, size $\leq s$ unbounded fanout formulas over the bit variables $\mathrm{BITS}_{N,N+1}$. Provided $k \leq N^\epsilon$ and $s \leq 2^{N^\epsilon}$, there exists a string $z \in \{0,1\}^k$ so that for all $y \in [N+1]$, there exists an assignment $f : [N] \to [N+1]$ such that:*

1. *$\Phi_i(f) = z_i$ for all i*
2. *$y \in \mathrm{range}(f)$*

In particular it is not possible to determine an empty pigeonhole of f by reading the values of $\Phi_1(f), \ldots, \Phi_k(f)$.

Overview of Proof. As mentioned in Section 1.2.1, we want to prove Theorem 2 by restricting attention to the class of inputs that correspond to 1-1 functions, and in order to do this we will need to prove a strengthening of Theorem 2, by proving the lower bound for Avoid on input functions $f : [N] \to [M]$, where M is larger than $N+1$. As discussed in Section 1.2.1, this implies the lower bound stated above (with $M = N+1$).

Thus we assume for sake of contradiction that there exist depth-3 AND-of-OR-of-AND circuits $\Phi_1, \ldots, \Phi_k$ of total size at most s that solves Avoid on all one-to-one instances $f : [N_0] \to [M_0]$, where $M_0 = N_0 + N_0^\epsilon$, for some $0 < \epsilon < 1$. Since we will prove the lower bound with respect to all 1-1 inputs, by Observation 3 we can assume without loss of generality that the bottom level of ANDs are partial matchings. Therefore, each circuit Φ_i is without loss of generality an AND of matching disjunctions.

1. The first step is to argue that there exists a matching restriction ρ such that after applying ρ, all depth-2 subcircuits (which are matching disjunctions) have width at most $w \approx \log s$. This is accomplished in the Width Reduction Lemma (8) by a standard Chernoff argument and a union bound. Thus after applying Lemma 8, we are left with depth-3 circuits $\Phi_1, \ldots, \Phi_k$ which are ANDs of matching disjunctions of width at most $w = c \log s$ for some constant $c > 0$. These circuits still solve Avoid but now with respect to 1-1 functions on the reduced domain and range, $[N_1]$, $[M_1]$, where $N_1 = \Omega(N_0)$, and $M_1 \approx N_1 + N_1^\epsilon$.

2. At this point each Φ_i is an AND of low-width matching disjunctions, i.e. we have $\Phi_i = \wedge_j \phi_{i,j}$ where the $\phi_{i,j}$ are width w matching disjunctions. The next step is to apply our Pigeonhole Switching Lemma (6) which will tell us that there exists a matching restriction ρ such that for all i, j, $\phi_{i,j} \restriction \rho$ simplifies to a depth d pigeonhole decision tree. We will be able to choose the parameters in our switching lemma so that after the second restriction, there are N_2 remaining pigeons and $M_2 = 2N_2$ remaining holes, with $d^2k^2 < N_2$. By Observation 4, this allows us to rewrite $\neg\phi_{i,j}$ as a width=d hole disjunction, and thus $\neg\Phi_i$ as width d hole disjunction. The simplified circuits $\neg\Phi_1, \ldots, \neg\Phi_k$ still solve Avoid, but now with respect to 1-1 functions on the reduced domain and range leftover after this restriction, i.e. on instances $[N_2] \to [2N_2]$.

3. The final step is to prove that if $\phi_1, \ldots, \phi_k$ are width-d hole disjunctions that solve Avoid with respect to all 1-1 functions from $[N_2]$ to $[2N_2]$, then we must have $d^2k^2 > N_2$. This is accomplished by Lemma 7. The proof is a novel argument based on coverings. Recall that our goal is to give some sequence of values $z_1, \ldots, z_k \in \{0,1\}$, so that for all $y_0 \in [2N_2]$ we can find a 1-1 instance $f : [N_2] \to [2N_2]$ so that $\phi_1(f) = z_1, \ldots, \phi_k(f) = z_k$, but y_0 is not an Avoid solution for f. This indicates that the values $\phi_1(f), \ldots, \phi_k(f)$ are insufficient to

determine an Avoid solution. To construct $z_1, \ldots, z_k$, we start by repeatedly searching for some ϕ_i whose underlying terms have a *small hitting set*, which is a small set of pigeons and holes so that every term of ϕ_i mentions at least one of them. If some ϕ_i is found, we apply a partial restriction ρ to the hitting set variables, which reduces the width of ϕ_i by 1. Since there are k ϕ_i's, and each has width at most d, after dk iterations, each ϕ_i either has been set to a constant, or is promised to have no small hitting set. We set $z_i = 1$ or $z_i = 0$ for all the ϕ_i which have been forced to a constant by our restriction ρ in this process, and set $z_i = 1$ for all the unkilled ϕ_i. Then to finish the proof, we need to show that for any y_0 we can find an instance $f\|\rho$ so that $y_0 \in \text{range}(f)$, and $\phi_i(f) = 1$ for every ϕ_i which was unkilled in the first step. Let $y_0 \in [2N_2]$ be given. Starting with the partial assignment ρ, we extend it so some unmapped pigeon x_0 goes to y_0, and then we iterate over each unkilled ϕ_i and try extend ρ to satisfy one of its terms greedily. The correctness will follow from the fact that none of the unkilled ϕ_i have a small hitting set.

We now formalize the above, which relies on the three Lemmas mentioned in the proof overview. Two of these Lemmas (Lemmas 8 and 7) will then be proven, while the more involved Lemma 6 is postponed to Appendix A.

Proof of Theorem 2. Fix some ϵ, δ to be specified later. Let N_0 be sufficiently large, and let $M_0 = N_0 + N_0^\epsilon$. Assume towards a contradiction that $\Phi_1, \ldots, \Phi_k$ are depth-3, size $s = 2^{N_0^\epsilon}$ circuits so that for all assignments $f : [N_0] \to [M_0]$, we may determine an empty pigeonhole for f by reading the values $\Phi_1(f), \ldots, \Phi_k(f)$, and that $k \leq N_0^{\epsilon/5}$. Without loss of generality we may assume:

$$\Phi_i = \bigwedge_j D_{i,j}$$

where $D_{i,j}$ is a DNF over the variables BITS_{N_0,M_0}. Obviously if $\Phi_1(f), \ldots, \Phi_k(f)$ could determine an empty pigeonhole on all assignments f then it would do so on all 1-1 assignments; we will derive a contradiction already from this fact. Therefore by Observation 3 we may assume

$$\Phi_i = \bigwedge_j \phi_{i,j}$$

where $\phi_{i,j}$ is a matching disjunction, at the cost of increasing the overall size of Φ from s to $M_0 s$, while preserving the behavior of the Φ_i on 1-1 assignments.

Applying the width reduction lemma (Lemma 8) with $w = c \log s$ for a sufficiently large constant $c > 1$, and a union bound over i, j, we can find a partial matching ρ_0 leaving N_1 pigeons unmapped, where $N_1 = \delta N_0$, such that for all i, j, $\phi_{i,j} \restriction \rho_0$ has width w. Thus we have reduced the bottom gate fanin of each circuit Φ_i to w, and the reduced circuits still solve Avoid with respect to all 1-1 functions f with domain $N_1 = \delta N_0$ and range $M_1 = N_1 + N_0^\epsilon$.

Next we apply our Switching Lemma (Lemma 6) with the following choice of parameters:

$$N := N_1$$

$$M := M_1 = N_1 + N_0^\epsilon$$

$$K := N_0^\epsilon$$

$$w = c \log s = c N_0^\epsilon$$

$$d := K^{1/5} = N_0^{\epsilon/5}$$

Choosing ϵ sufficiently small, it follows that for all i, j:

$$Pr_{\rho \sim \mathcal{M}_K}[\mathcal{T}_{\phi_{i,j} \restriction \rho} \text{ has depth } \geq d] < \frac{1}{ks}$$

Thus by a union bound over i, j, we conclude that there exists a partial matching ρ so that for each i, j there is a pigeonhole decision tree $T_{i,j}$ of depth d such that $\phi_{i,j} \equiv_\rho T_{i,j}$.

Applying Observation 4 we can find hole disjunctions $\psi_{i,j}$ of width d so that $\neg\phi_{i,j} \equiv_\rho \psi_{i,j}$, and therefore $\neg\Phi_i \equiv_\rho \bigvee_j \psi_{i,j}$ which is a width d hole disjunction. By the conditions on the original $\Phi_1, \ldots, \Phi_k$, for all 1-1 assignments f extending ρ we must be able to determine an empty pigeonhole of f by reading the values $\neg\Phi_1(f), \ldots, \neg\Phi_k(f)$.

After applying ρ, we are left with a reduced domain and range $[N_2]$ and $[M_2]$, where $N_2 = K = N_0^\epsilon$, and $M_2 = M_1 - (N_1 - K) = 2K = 2N_2$. Therefore on the smaller input size $[2N_2]^{[N_2]}$, we have found a sequence of k width-d hole disjunctions such that on all 1-1 assignments $g : [N_2] \to [2N_2]$, we can determine an empty pigeonhole of g by reading the values of these hole disjunctions applied to g. At this point we reach a contradiction with Theorem 7, since by our choice of parameters we have $4k^2d^2 < N_2$ is satisfied (recall that $k \leq N_0^{\epsilon/5}$ by assumption).

□

Equipped with the Pigeonhole Switching Lemma (Lemma 6, proof in Section A), it is left to prove Lemmas 8 and 7. We will first prove Lemma 7 followed by a proof of Lemma 8.

Lemma 7. *Let $\phi_1, \ldots, \phi_k$ be hole disjunctions of width w over $[2N]^{[N]}$. Then provided $N > 4k^2w^2$, there exists a string $z \in \{0,1\}^k$ so that for all $y \in [2N]$, there exists a 1-1 function $f : [N] \to [2N]$ such that:*

1. *$\phi_i(f) = z_i$ for all i*
2. *$y \in \text{range}(f)$*

In particular it is not possible to determine an empty pigeonhole of f from $\phi_1(f), \ldots, \phi_k(f)$ for all 1-1 assignments f.

To prove this we need the following definition:

Definition 20 (Hitting Sets)**.** *If ϕ is a hole disjunction and $A \subseteq [N] \sqcup [M]$, we say that A is a hitting set for ϕ if $A \cap \text{nodes}(\tau) \neq \emptyset$ for all terms $\tau \in \phi$.*

We can observe the following property of hitting sets:

Observation 5. *Let $\pi : [N] \to [M]$ be a partial matching, ϕ a nonempty hole disjunction, and $A \subseteq [N] \sqcup [M]$ a non-empty hitting set for $\phi \restriction \pi$. Say that π' is an extension of π such that $A \subseteq \text{nodes}(\pi')$. Then for each term $\tau \in \phi \restriction \pi$, τ has strictly smaller width in $\phi \restriction \pi'$ then it originally did in $\phi \restriction \pi$.*

We can now prove the main claim:

Proof. To prove this we will construct partial matching $\pi : [N] \to [2N]$, a set $I \subseteq [k]$, and values $\{z_i \in \{0,1\} \mid i \in I\}$ so that:

1. For all total 1-1 assignments f extending π, $\phi_i(f) = z_i$ for all $i \in I$.
2. For all $y_0 \in [2N]$, there exists a total 1-1 assignment f extending π so that $\phi_i(f) = 1$ for all $i \notin I$, and $y_0 \in \text{range}(f)$.

We will initialize $\pi = \emptyset$ and expand it in stages by the following procedure:

1. If there exists a nonempty $A \subseteq [N] \sqcup [2N], |A| \leq 2kw + 2$ and some $i \in [k]$ so that A is a hitting set for $\phi_i \restriction \pi$, then extend π to a 1-1 map π' so that $A \subseteq \text{nodes}(\pi')$. If none exist then halt the procedure and output the current π.

2. Set $\pi := \pi'$ and return to the previous step.

We claim that we will always be able to extend π in the appropriate way until no further hitting sets can be found, and that at the end we will have $|\pi| \leq kw(2kw + 2) = 2k^2w^2 + 2kw$. To see this, observe that if ϕ is a hole disjunction of width w and $\text{nodes}(\pi)$ is a hitting set for ϕ, then every term in ϕ will either be killed in $\phi \restriction \pi$ or else have its width decreased by 1 by Observation 5. Therefore the above procedure can only repeat kw times before all the ϕ_i have been killed. In each step at most $2kw + 2$ edges need to be added to π in order to cover A, so the total size of π at any step is at most $kw(2kw + 2) = 2k^2w^2 + 2kw$, and therefore there are always enough available pigeons/holes to extend π in the appropriate way at each step since $N > k^2w^2$.

Now we can choose $I \subseteq [k]$, which will consist of those indices i so that $\phi_i \restriction \pi$ is forced to a constant (i.e. one of its terms is already satisfied by π in which case it is forced to one, or else all of its terms are killed and it is forced to 0). For $i \in I$ we denote by $z_i \in \{0, 1\}$ the value it is forced to. It remains to show that for any given $y_0 \in [2N]$, we can find a total 1-1 assignment f extending π so that $y_0 \in \text{range}(f)$ and $\phi_i(f) = 1$ for all $i \notin I$. To do this we will construct a hole restriction $\langle \rho, U \rangle$, where ρ is another partial matching disjoint consistent with π, and $U \subseteq [2N]$ satisfying:

1. $y_0 \in \text{range}(\pi \cup \rho)$

2. $|\rho| \leq kw$, $|U| \leq kw$

3. $U \cap \text{range}(\pi \cup \rho) = \emptyset$

4. For each $i \in [k] \setminus I$ there is a term $\langle \sigma, E \rangle \in \phi_i$, so that for all $x \in \text{dom}(\sigma)$ we have $\pi \cup \rho(x) = \sigma(x)$, and $E \subseteq U$.

If we can accomplish this then the proof is complete. We simply extend $\pi \cup \rho$ to a total 1-1 assignment f which leaves U out of its range. By the assumption $N > 4k^2w^2$ such a total assignment exists; here we are using the fact that the total number of holes $2N$ is larger then the size of U plus the number of holes filled thus far by $\pi \cup \rho$. By construction we then have $\phi_i(f) = 1$ for all $i \notin I$ and $y \in \text{range}(f)$.

Observe that by construction of π, for each $i \notin I$ we have that $\phi_i \restriction \pi$ has no hitting set of size $\leq 2kw + 2$. We will construct ρ, U in stages. Initially we check if $y_0 \in \text{range}(\pi)$, if not we initialize $\rho = \{(x_0, y_0)\}$ where x_0 is an arbitrary element unmapped in π, otherwise we initialize $\rho = \emptyset$. In addition we initialize $U = \emptyset$. Now we go through each $i \in [k] \setminus I$ in order and do the following. We search for some term $\tau = \langle \sigma, E \rangle \in \phi_i \restriction (\pi \cup \rho)$ so that $\text{nodes}(\tau) \cap \text{nodes}(\langle \rho, U \rangle) = \emptyset$. If found, then we add the nodes in E to U and the edges in σ to ρ. We claim that it is always possible to find such a term while maintaining that $\pi \cup \rho$ is 1-1 and $U \cap \text{range}(\pi \cup \rho) = \emptyset$. In particular say that we have gotten to some step $i \in [k] \setminus I$ where this is not possible. Let $A = \text{nodes}(\langle \rho, U \rangle)$. Observe that by construction $|A| \leq 2kw + 2$. Recall that by construction of π we have that A cannot be a hitting set for $\phi_i \restriction \pi$. By definition of a hitting set this implies the existence of the required term $\tau \in \phi_i$. $\square$

Next, we prove the Width Reduction Lemma:

Lemma 8 (Width Reduction)**.** *Let $\epsilon > 0$ be a sufficiently small constant. Let ϕ be a matching disjunction with s terms over $[M]^{[N]}$. Say a partial matching ρ is sampled as follows: sample $A \subseteq [N]$ by including each element independently with probability $1-\epsilon$; now choose a uniform 1-1 function $A \to [M]$.*

$$\Pr_{\rho}[\phi \restriction \rho \text{ has width } \geq w] < \exp(\log s - \Omega(w))$$

Proof. Let $\phi = \lambda_1, \ldots, \lambda_s$. We will ignore each term whose width is already $< w$; its width cannot increase under the restriction. Let $\lambda \in \phi$ be a term of width $\geq w$. We bound the probability that ρ does not kill the term λ as:

$$\begin{aligned}
\Pr_{\rho}[\lambda \text{ survives } \rho] &= \sum_{U \subseteq \mathrm{dom}(\lambda)} \Pr[\mathrm{dom}(\lambda) \cap \mathrm{dom}(\rho) = U] \frac{1}{|U|!\binom{M}{|U|}} \\
&\leq \sum_{t=1}^{w} \Pr[|\mathrm{dom}(\lambda) \cap \mathrm{dom}(\rho)| = t] \frac{1}{t!\binom{M}{t}} \\
&\leq \frac{1}{(w/2)!\binom{M}{w/2}} + \Pr[|\mathrm{dom}(\lambda) \cap \mathrm{dom}(\rho)| < \frac{w}{2}] \\
&\leq \frac{1}{(w/2)!\binom{M}{w/2}} + \Pr_{\xi_1,\ldots,\xi_w \sim \mathrm{Bern}(1-\epsilon)}[\sum_i \xi_i \leq \frac{w}{2}] \leq \exp(-\Omega(w))
\end{aligned}$$

Where the last inequality follows from Chernoff's bound and $\mathrm{Bern}(1-\epsilon)$ denotes the Bernoulli distribution with expected value $1-\epsilon$. Thus by a union bound

$$\Pr_{\rho}[\phi \restriction \rho \text{ has width } \geq w] \leq \exp(\log s - \Omega(w))$$

□

Lower Bounds for Moderately Weak Avoid: Above we showed that depth-3 circuits cannot solve the Strong Avoid problem unless their size is exponential. In contrast, Li's construction gives *quasipolynomial* size depth 3 circuits which solve the Weak Avoid problem, i.e. Avoid on instances $[N] \to [2N]$ when the codomain is at least twice as large as the domain. By a standard iteration construction (the same argument which lets us reduce $[N] \to [2N]$ Avoid to $[N] \to [N^2]$ Avoid as in [Kor21]), for every constant d we can obtain quasipolynomial size depth 3 circuits solving Avoid on instances $[N] \to [N + N/\log^d N]$. We show here that this is essentially optimal: for every constant c there exists another constant d so that $[N] \to [N + N/\log^d N]$ Avoid cannot be solved by depth 3 circuits of size $2^{\log^c N}$. The proof is a simple reduction to Theorem 2 which is originally due to Razborov [Raz].

Lemma 9. *For every $c \in \mathbb{N}$ there exists $d \in \mathbb{N}$ so that the following holds. Let $\Phi_1, \ldots, \Phi_k$ be depth 3 circuits of size s such that for every assignment $f : [N] \to [N + N/\log^d N]$, an Avoid solution can be determined from the values $\Phi_1(f), \ldots, \Phi_k(f)$. Then either $k \geq \log^c N$ or $s \geq 2^{\log^c N}$.*

Proof. Let $c \in \mathbb{N}$ be fixed. We will choose $M = \log^d N$ for an appropriate choice of d which will be specified later and which will depend only on c. We then reduce the Avoid problem on instances $[M] \to [M+1]$ to instances $[N] \to [N + N/\log^d N]$ as follows: given $g : [M] \to [M+1]$, let $g^{\otimes \ell} : [M\ell] \to [(M+1)\ell]$ be the map which sends each block $[1, M], \ldots, [M(\ell-1), M\ell]$ to its corresponding block in $[1, M+1], \ldots [(M+1)(\ell-1), (M+1)\ell]$ according to g. Setting $\ell = \frac{N}{M}$ we see that g is a map $[N] \to [N + \ell]$, where $\ell = N/M = N/\log^d N$.

Obviously given g we can generate $g^{\otimes \ell}$ without any computational overhead by simply substituting variables. In addition, any Avoid solution to $g^{\otimes \ell}$ uniquely determines an Avoid solution to g, obtained by simply forgetting the block number and outputting the position within the relevant block. Thus if the circuits $\Phi_1(f), \ldots, \Phi_k(f)$ could solve Avoid on inputs $[N] \to [N + N/\log^d N]$ then they can also solve Avoid on inputs $[M] \to [M+1]$. By Theorem 2 there is some absolute constant ϵ, such that for this to be possible we must have $k \geq M^\epsilon$ and $s \geq 2^{M^\epsilon}$. Setting $d > \frac{c}{\epsilon}$ the theorem follows. □

3.3 $\mathsf{BPP}^{\mathsf{NP}}$ Lower Bound for 1-1 Strong Avoid

We restate the main result to be proved:

Theorem 3 *In the decision tree model, Strong 1-1 Avoid is not solvable in* $\mathsf{FBPP}^{\mathsf{NP}}$.

Recalling the definition of $\mathsf{BPP}^{\mathsf{NP}}$ (in the decision tree model), if 1-1 Strong Avoid has $\mathsf{BPP}^{\mathsf{NP}}$ decision trees of complexity w, then there is a distribution $\mathcal{T}$ over P^{NP} decision trees of complexity r, so that for all $f : [N] \to [N+1]$ $\mathcal{T}$ outputs a 1-1 Strong Avoid Solution with probability $\geq 2/3$. By Yao's minimax principle, this implies the existence of a fixed tree $T \in \text{support}(\mathcal{T})$, so that $T(f)$ succeeds in finding an empty pigeonhole of f with probability $\geq 2/3$ when f is sampled uniformly from the space of 1-1 functions $\mathcal{F}_{N,N+1}$. It thus suffices to rule out the existence of such a deterministic tree T. We will use here the notation $\mathcal{F}_N := \mathcal{F}_{N,N+1}$, and $f \sim \mathcal{F}_N$ to denote a uniform sample from this set. Thus Theorem 3 follows from the following Theorem:

Theorem 8. *There is an absolute constant $\epsilon > 0$ so the following holds: if T is a P^{NP} decision tree of complexity N^ϵ over the bit variables $\text{BITS}_{N,N+1}$ and leaves labeled by elements of $[N+1]$, then:*

$$\Pr_{f \sim \mathcal{F}_N}[T(f) \notin \text{range}(f)] \leq N^{-\Omega(1)}$$

The high level idea of the proof is as follows. Observe that a random $f \sim \mathcal{F}_{N,M}$ can be sampled by first choosing a uniform partial matching ρ of a certain size, and then choosing a uniform completion of ρ to a total assignment. Applying our Switching Lemma, we can argue that with high probability, after we sample ρ we can replace all of the NP queries in our P^{NP} decision tree T with small depth pigeonhole decision trees, which overall allows T to be replaced by a low depth pigeonhole decision tree. It then remains only to argue that a pigeonhole decision tree of low depth cannot solve 1-1 Strong Avoid with non-trivial probability on a uniform extension f of ρ, which can be accomplished with a direct argument.

Proof of Theorem 8. By definition of a P^{NP} decision tree, T is a binary tree of depth $r = N^\epsilon$ with each internal node branching on the value of some DNF of width r over the bit variables $\text{BITS}_{N,N+1}$. Let $D_1, \ldots, D_s$ be the set of all DNF associated to the nodes of T; so $s \leq 2^r$. As in the proof of Theorem 2 we may apply Observation 3 and replace all D_i by width $r \log N$ matching disjunctions ϕ_i since we only care about their behavior on assignments in $\mathcal{F}_N$. Fix some parameter K to be specified later. We will consider sampling $f \sim \mathcal{F}_N$ in the following way: first choose a uniform partial matching ρ with $N - K$ edges, then a uniform extension $f \supseteq \rho$ of ρ to a total 1-1 assignment. We define $\mathcal{B}$ as the event that $\phi_i \restriction \rho$ does not have a pigeonhole decision tree of depth $\leq r$ for some $i \leq s$. Observe in the case $\rho \notin \mathcal{B}$, we may construct a pigeonhole decision tree $\mathcal{T}$ of depth r^2 so that $\mathcal{T} \equiv_\rho T$; we simply simulate the original computation of T, and each time a DNF D_i was originally queried, we instead simulate the depth $\leq r$ pigeonhole decision which represents ϕ_i. Let $\mathbb{P}_d(J)$ be the maximum, over all pigeonhole decision trees Q of depth d on $g : [J] \to [J+1]$, of $\Pr_{g \sim \mathcal{F}_J}[Q(g) \notin \text{range}(g)]$. Then we have:

$$\begin{aligned}
\Pr_{f\sim\mathcal{F}_N}[T(f) \notin \text{range}(f)] &= \Pr_{\rho,\ f\supseteq\rho}[T(f) \notin \text{range}(f)] \\
&\leq \Pr_{\rho}[\rho \in \mathcal{B}] + \max_{\rho\notin\mathcal{B}} \Pr_{f\supseteq\rho}[T(f) \notin \text{range}(f)] \\
&\leq \Pr_{\rho}[\rho \in \mathcal{B}] + \mathbb{P}_{r^2}(K) \\
&\leq 2^r \max_i \Pr_{\rho}[\phi_i \restriction \rho \text{ has no pigeonhole decision tree of depth } \leq r] + \mathbb{P}_{r^2}(K)
\end{aligned}$$

We will prove in the next lemma that $\mathbb{P}_{r^2}(K) \leq \frac{r^2}{K+1}$. Setting ϵ sufficiently small and $K = r^3$, we can apply Lemma 6 and complete the proof. □

It remains to prove the bound on $\mathbb{P}_{r^2}(K)$. At the cost of increasing the depth of the tree by 1, we assume that a pigeonhole decision tree always queries a hole before outputting it as a solution.

Lemma 10. $\mathbb{P}_d(N) = \frac{d}{N+1}$

Proof. Let T be a tree witnessing $\mathbb{P}_d(N)$.

Say that T first queries a hole $y \in N$, so that

$$T(f) = \begin{cases} T_1(f) & \text{if } f(1) = y \\ \cdots & \\ T_N(f) & \text{if } f(N) = y \\ y & \text{if } y \notin \text{range}(f) \end{cases}$$

then we have:

$$\begin{aligned}
\mathbb{P}_d(N) &= \Pr_f[T(f) \notin \text{range}(f)] \\
&= \Pr_f[y \notin \text{range}(f)] + \sum_x \Pr_f[f(x) = y] \cdot \Pr_f[T_x(f) \notin \text{range}(f) \mid f(x) = y] \\
&\leq \frac{1}{N+1} + \frac{N}{N+1}\mathbb{P}_{d-1}(N-1)
\end{aligned}$$

Here we are using that fact that a uniform f conditioned on $f(x) = y$ is equivalent, up to relabeling, to a uniform member of $\mathcal{F}_{N-1}$, and the labeling of pigeons and holes has no effect on the value of $\mathbb{P}_{d-1}(N-1)$. By the same reasoning if T first queries a pigeon x then we have the simpler bound:

$$\Pr_f[T(f) \notin \text{range}(f)] = \sum_y \Pr_f[f(x) = y] \cdot \Pr_f[T_y(f) \notin \text{range}(f) \mid f(x) = y] \leq \mathbb{P}_{d-1}(N-1)$$

so overall

$$\mathbb{P}_d(N) \leq \max\{\mathbb{P}_{d-1}(N-1), \frac{1}{N+1} + \frac{N}{N+1}\mathbb{P}_{d-1}(N-1)\}$$

Therefore by induction on d we conclude that the optimal pigeonhole decision tree achieving $\mathbb{P}_d(N)$ only queries holes, and without loss of generality queries them in order $1, \ldots, d$ (in the base case we use the assumption that T must query a hole before using it as an answer), then outputs the index of the first empty hole that was queried (if any). Therefore we have:

$$\mathbb{P}_d(N) = \Pr_f[\{1, \ldots, d\} \nsubseteq \text{range}(f)] = \frac{d}{N+1}$$

□

Next we show that Theorem 4, separating the decision class $\Sigma_2^P \cap \Pi_2^P$ from $\mathsf{BPP}^{\mathsf{NP}}$, follows as a corollary of Theorem 3.

Proof of Theorem 4. Lemma 4 says that Strong 1-1 Avoid is solvable in $\mathsf{FP}^{\Sigma_2^P \cap \Pi_2^P}$. This result holds relative to every oracle (and thus in the decision tree model). Hence relative to every oracle, if $\Sigma_2^P \cap \Pi_2^P \subseteq \mathsf{BPP}^{\mathsf{NP}}$ then Strong 1-1 Avoid collapses into $\mathsf{BPP}^{\mathsf{NP}}$. The result then follows from Theorem 3. □

4 Linear Ordering Principle

In this section we investigate the complexity of the Linear Ordering Principle. We restate the definition in a slightly different terminology:

Definition (Linear Ordering Principle, Restatement of 2)**.** *The input to LOP is a binary relation $\prec$ on $[N] \times [N]$ specified by a circuit. The following solutions are sought:*

1. *A witness that $\prec$ does not define a total order on $[N]$ (trivial solution)*
2. *An element $a_0 \in [N]$ so that $a_0 \prec a$ for all $a \neq a_0 \in [N]$ (nontrivial solution)*

We say an instance $\prec$ is nontrivial if it has no trivial solutions. The trivial solutions are enumerated formally as follows:

Antireflexivity Violation: *x such that $x \prec x$*

Transitivity Violation: *x, y, z such that $x \prec y \prec z$ and $x \not\prec z$*

Totality Violation: *$x \neq y$ such that $x \not\prec y$ and $y \not\prec x$*

Note that in the introduction we defined $\prec$ to be a relation on $\{0,1\}^n \times \{0,1\}^n$. We state it in this more general form here, since the space we define a linear ordering on in our main reduction will not have size being a power of two. It is straightforward to pad an instance of LOP which is defined on some subset of $\{0,1\}^n$ to the whole space without affecting the value of the solution.

Recall from Section 2.1 that LOP has "essentially unique solutions:" the trivial solutions are checkable in polynomial time, and any instance without trivial solutions has a unique nontrivial solution. Our primary interest in this problem stems from the following reduction (stated in the Introduction):

Theorem 1 *Weak Avoid is polynomial-time many-one reducible to LOP.*

We will describe the reduction in two parts, using an intermediate search problem called Forest Termination:

Definition 21 (Forest Termination)**.** *The input consists of:*

1. *A function* $\mathrm{Pred} : [M] \times [N] \times [N] \to [M]$
2. *A function* $\mathrm{Col} : [M] \times [N] \to \{0,1\}$

specified by boolean circuits. We think of the input as representing a layered rooted forest F of depth N with nodes partitioned into sets $L_1, \ldots, L_N$, each of size M, with nodes in L_1 being roots and all other nodes in $\{L_j\}_{j>1}$ having a unique parent in L_{j-1}. We say that the function Pred *"validly represents a forest" if the following holds: for each $u \in [M]$, $k \leq j \leq i \in [N]$, we have*

that $\mathrm{Pred}(\mathrm{Pred}(u,i,j),j,k) = \mathrm{Pred}(u,i,k)$. *If this holds then we may think of* Pred *as representing some forest* F *in the following strong sense: for any node* $u \in L_i$ *with* $i > 1$*, the nodes* $\mathrm{Pred}(u,i,1),\ldots,\mathrm{Pred}(u,i,i)$ *form the unique path from* L_1 *to* u *in* F*. Finally for each each* $u \in F$ *with think of* $\mathrm{Col}(u,i)$ *as coloring the node* $u \in L_i$ *red or blue.*

We say that $u \in F \cup \{\perp\}$ *is a termination point of* F *if the following holds:*

1. $u = \{\perp\}$ *and* L_1 *contains no blue node.*
2. u *is a node in* F*, the path from* L_1 *to* u *in* F *uses only blue nodes, and* u *does not have a blue child in* F*.*

Now the search problem is: given $\mathrm{Pred}, \mathrm{Col}$*, find a witness that* Pred *does not validly represent a forest, or else find a termination point in the forest it represents.*

The totality of this problem follows by starting at at a blue node in L_1 (if one exists) and finding a maximal blue path through its descendants in F. We start by giving a reduction from Weak Avoid to Forest Termination. This is the aspect of the proof which is very similar to arguments by Li and by Paris Wilkie and Woods [Li24, PWW88].

Lemma 11. *Weak Avoid with* $2n$ *stretch is polynomial-time reducible to Forest Termination.*

Proof. Let $f : \{0,1\}^n \to \{0,1\}^{2n}$ be given; we denote by $f_0, f_1 : \{0,1\}^n \to \{0,1\}^n$ the functions obtained by restricting to the first and last n bits of output respectively. Let $\mathcal{S}_n$ denote the set of all binary strings of length at most n, including the empty string ϵ. Naturally $\mathcal{S}_n$ is associated with the nodes of a full binary tree of depth n; however since the Forest Termination instance we construct involves a forest whose structure and depth differ from that of $\mathcal{S}_n$, we will use the terminology of binary strings to refer to elements of $\mathcal{S}_n$ in order to avoid confusion. We say $s \sqsubseteq s'$ if s is a prefix of s', and $s \sqsubset s'$ if it is a proper prefix. We use $s \cdot s'$ for concatenation. If $s \sqsubseteq s'$, with $s' = s \cdot p$, we use $s' - s$ to denote p. We define the "preorder" $<$ on $\mathcal{S}_n$ recursively with respect to prefixes: for a prefix p and two distinct extensions $p \cdot s, p \cdot s'$ with $s < s'$ (with the relative order of s, s' defined recursively), we put $p \cdot s < p \cdot s' < p$. In the view of $\mathcal{S}_n$ as a depth n perfect binary tree, this corresponds to the recursive subtree ordering "left subtree $<$ right subtree $<$ root." Note that this is a total ordering with ϵ being the greatest element and 0^n the least. Finally, for an element $s \in \mathcal{S}_n$, we say that s' is a "left outlet" of s if either $s' = s$ or $s' = p0$ for some $p \sqsubseteq s$. We use $\mathrm{LO}(s)$ to denote its set of left outlets; note that $|\mathrm{LO}(s)| \leq n$. Observe that for any $s' \leq s$, there exists a unique $p \in \mathrm{LO}(s)$ such that $p \sqsubseteq s'$.

For some $s \in \mathcal{S}_n$ and some value $x \in \{0,1\}^n$, define $f_s(x)$ recursively as follows: if $s = \epsilon$ then $f_s(v) = v$. If $s = bs'$ for some bit b and substring s', $f_s = f_{s'}(f_b(x))$. We define a "transcript" as an element $\pi \in (\{0,1\}^n)^n$. We say the transcript π is valid for some $s \in \mathcal{S}_n$ if $\pi_i = 0^n$ for all $i > |\mathrm{LO}(s)|$. In this way the valid transcripts for s are in one-to-one correspondence with functions $\mathrm{LO}(s) \to \{0,1\}^n$. Thus for a valid transcript π for s and some $p \in \mathrm{LO}(s)$, we use $\pi(p) \in \{0,1\}^n$ to denote the value the transcript associates to p. For a transcript π, and strings $s' < s \in \mathcal{S}_n$, we define the transcript $\pi' = \mathrm{Pred}(\pi, s, s')$ as follows. We first check if π is valid for s; if not we set π' to be some fixed canonical choice of invalid transcript for s'. Otherwise, for each $p \in \mathrm{LO}(s')$, we set $\pi'(p) = f_{p-q}(\pi(q))$ where $q \in \mathrm{LO}(s)$ is the unique left outlet of s which is an ancestor of p.

We now construct an instance of Forest Termination with $[N] = [2^{n+1} - 1] = \mathcal{S}_n$ and $M = [2^{n^2}] = (\{0,1\}^n)^n$. $[N]$ is associated naturally to $\mathcal{S}_n$, so that each layer corresponds to an element of $\mathcal{S}_n$ and the layers are ordered according to $<$, i.e. L_1 is associated to $0^n \in \mathcal{S}_n$ and L_N to $\epsilon \in \mathcal{S}_n$. $[M]$ is associated to the set of possible transcripts, so that L_s will be the set of all possible transcripts for the string s (some valid and others invalid). We use Pred defined above to give

the Pred function in the instance of Forest Termination. It should be noted that the depth of the forest defined here is $N = 2^{n+1} - 1$, which is much larger then the depth of tree naturally associated to $\mathcal{S}_n$ (whose depth is n); this is why we have used the notation of binary strings rather then nodes in a tree to refer to elements of $\mathcal{S}_n$. It remains to define the node coloring function Col. If $(\pi, s) \in [M] \times [N]$ are such that π is not a valid transcript for s, then $\mathrm{Col}(\pi, s)$ is red; if π is a valid transcript for s and $|s| < n$ then it is colored blue ($|s|$ denotes its length as a binary string). Otherwise if π is a valid transcript for s and $|s| = n$, we check that $\pi(s) = \neg f_s(s)$, where $\neg$ denotes the string obtained by flipping all bits. If so we color it blue, otherwise we color it red. Note that since $|s| = n$, $f(s)$ is well defined, and thus so is $f_s(s)$.

It follows by construction that Pred validly defines a forest in the sense of Definition 21. It remains to show that given any termination point we may determine a solution to the original instance of range avoidance. We first claim that there exists a blue node in the first layer L_{0^n}, so that the solution to Forest Termination cannot be $\perp$. Observe that $\mathrm{LO}(0^n) = \{0^n\}$, thus the transcript π which sets $\pi(0^n) = \neg f_{0^n}(0^n)$ will be blue. Now, say that $\pi \in L_s$ is a termination node. We claim that L_s cannot be the last layer, i.e. $s \neq \epsilon$. Say towards a contradiction that π is a termination node in layer L_ϵ. So π is a valid transcript for ϵ, which stores a single value $x = \pi(\epsilon)$. By the assumption π is a termination point all of its predecessors are blue. In particular if we view x as a length-n element of $\mathcal{S}_n$, then we have that $\mathrm{Pred}(\pi, \epsilon, x)$ is blue, which by construction means that $f_x(x) = \neg f_x(x)$, which is a direct contradiction.

Now say that π is a termination node in layer L_s where $s \neq \epsilon$; let s' be the successor of s in the ordering $<$. First say that $|s'| = n$; in this case (π, s) cannot be a termination point, since we may construct a blue transcript π' for s' such that $\mathrm{Pred}(\pi', s', s) = \pi$ by simply appending the value $\neg f_{u'}(u')$ as $\pi'(u')$ to the original transcript π for s. Otherwise we have $s = s'1$. Let $p = s'0$; so p is a left-outlet of s, and therefore π stores some values $x_0 = \pi(p), x_1 = \pi(s)$. We claim that x_0x_1 must be a solution to the Avoid instance f. Say this is not true and there is some $x \in \{0,1\}^n$ such that $f(x) = x_0x_1$; i.e. $f_0(x) = x_0$ and $f_1(x) = x_1$. Then we can generate a blue transcript π' for s' as follows. Observe that $\mathrm{LO}(s') = \mathrm{LO}(s) \setminus \{p\} \cup \{s'\}$. Thus if we remove p from the domain of π and set $\pi'(s') = x$ to obtain π', then π' will be a blue transcript for s' and we will have $\mathrm{Pred}(\pi', s', s) = \pi$, which means (π, s) cannot be a termination point. To see this, note that in the computation of $\mathrm{Pred}(\pi', s', s)$, the only modification made to π' to generate π will be to compute $z_0 = f_0(\pi'(s')), z_1 = f_1(\pi(s'))$, and add z_0 as the value π associates to p and z_1 as the value it associates to s. By construction $z_0 = x_0$ and $z_1 = x_1$ so we end up with the original transcript π. □

Next we reduce Forest Termination to LOP, which is simpler:

Lemma 12. *Forest Termination is polynomial-time many-one reducible to LOP.*

We will prove shortly that any search problem which is P^{NP}-reducible to LOP is also polynomial-time many-one reducible to LOP (Theorem 5); therefore it suffices to give a P^{NP} reduction.

Proof. Let Pred, Col an instance of Forest Termination. We start by using an NP oracle to search for a witness that Pred fails to validly represent a forest; if found we output this as our solution, otherwise we know that the instance indeed represents a forest F on the nodes $L_1 \sqcup \cdots \sqcup L_N$. For conceptual simplicity, we modify F into a tree T by adding a virtual blue root node $\perp$ whose children consist of all nodes in L_1. We use $<$ for the lexicographical ordering on each L_i. We now define a new ordering $\prec$ on the nodes of T. Its definition is given recursively with respect to subtrees of T as follows: say that u is a node with blue-rooted subtrees $B_1 < \ldots < B_K$ and red-rooted subtrees $R_1 < \ldots < R_{K'}$ ordered lexicographically by their respective roots. Define $\prec$ recursively inside

each B_i, R_j; now between them use the ordering $B_1 \prec \ldots \prec B_K \prec \{u\} \prec R_1 \prec \ldots \prec R_{K'}$. By induction on the depth of T we see that $\prec$ is a total ordering. The relation $\prec$ will be our instance of LOP output by the reduction; it remains to show that $\prec$ is efficiently computable and that its minimal element yeilds a Forest Termination solution.

We first show how to compute $\prec$. Given $u_0 \neq u_1 \in T$:

1. If u_0 is an ancestor of u_1 then let v be the child of u_0 from which u_0 can be reached. If v is blue then set $u_1 \prec u_0$, else if it is red set $u_0 \prec u_1$.

2. Otherwise let v be the least common ancestor of u_0 and u_1, and let w_0, w_1 be the children of w from which u_0 and u_1 can be reached respectively. We define the relative ordering of w_0, w_1 in $\prec$, which is then inherited by the pair u_0, u_1: if w_b is blue and $w_{\neg b}$ is red for some $b \in \{0, 1\}$, then set $w_b \prec w_{\neg b}$. Otherwise use the lexicographical order $<$ to order the pair w_0, w_1.

The only nontrivial step here is to compute a least common ancestor of u_1, u_2, which can be accomplished with binary search. Say u_1, u_2 lie in layers $L_i, L_{i'}$ respectively, with $i' > i$. The using Pred we find the ancestor of u_2 in L_i; call it $\hat{u}_2$. If $\hat{u}_2 = u_1$ then u_1 is the ancestor of u_2. Otherwise let $j = \lceil i/2 \rceil$, and compute the ancestors v_1, v_2 of $u_1, \hat{u}_2$ in layer L_j. If they are distinct then we repeat the same procedure starting from v_1, v_2. Otherwise if $v_1 = v_2$ we prune all the layers $L_1, \ldots, L_{j-1}$ from consideration and repeat the procedure from $u_1, \hat{u}_2$, taking L_j to be the root layer with $v_1 = v_2$ as its root, and renumbering the layers accordingly. Overall the number of steps required is $O(\log N)$.

Finally it remains to show that any solution to LOP on the instance $\prec$ yields a solution to the given Tree Termination instance, i.e. if $u \in \{\perp\} \sqcup L_1 \sqcup \cdots \sqcup L_N$ is minimal in the ordering $\prec$ then it must correspond to a termination point. Say that u is not a termination point, then either:

1. There is an edge (v, w) in the path from the root to u so that w is colored red. In this case we must have $v \prec u$.

2. u has a blue child v. In this case we have $v \prec u$.

In each case we see that u cannot be the minimal element. An example of a Forest Termination instance and its corresponding order $\prec$ is given in Figure 2. □

We now observe some special properties held by the search problem LOP, the first of which is its strong closure properties under a wide class of reductions:

Theorem (Theorem 5)**.** *Let R be any search problem which has a polynomial-time, NP-oracle Turing reduction to LOP. Then R is polynomial-time many-one reducible to LOP.*

Proof. Let A be a polynomial time P^{NP} Turing reduction from some search problem R to LOP. Let x be an instance of the search problem of length n. We first claim that we can modify A so that:

1. $A(x)$ only calls the LOP oracle on instances $\prec$ which are nontrivial, i.e. they define a true total order.

2. $A(x)$ does not use its NP oracle.

To achieve the first condition, before each call to the LOP oracle on an instance $\prec$, we first use the NP oracle to check if $\prec$ defines a total order, and to find a violation if not. If a violation is found we no longer need to use the LOP oracle on this instance; otherwise we know that $\prec$ is a nontrivial

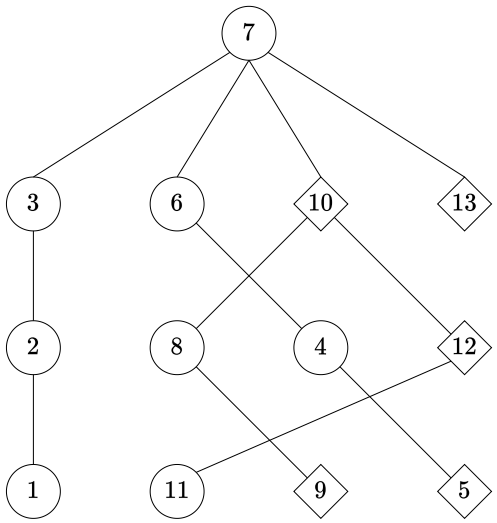

Figure 2: Example instance of Forest Termination with $M = 4$ and $N = 3$, with virtual root added at the top. Blue nodes are circles and red nodes are diamonds. We have sorted each layer left to right so that blue nodes come first, and within each color class the nodes are ordered left to right lexicographically. The number on each node represents its position in the ordering $\prec$. The nodes numbered 1 and 4 are the termination points of this instance.

instance and we use the oracle to solve it. Now to achieve the second condition, if ϕ is an instance of SAT with m variables, we define the ordering $\prec_\phi$ on $\{0,1\}^m$ by: $y \prec_\phi z$ if $\phi(y) = 1 \wedge \phi(z) = 0$ or vice versa, otherwise $\prec_\phi$ orders them lexicographically. Clearly $\prec_\phi$ is a nontrivial LOP instance, and the minimal element of $\prec_\phi$ will tell us if ϕ is satisfiable.

At this point we have a polynomial time reduction A which makes adaptive oracle calls to LOP on nontrivial instances, each of which has a unique solution. We next modify $A(x)$ to make exactly m calls on all computation paths, and have each call of the form $\prec$: $\{0,1\}^m \times \{0,1\}^m \to \{0,1\}$ for some fixed value $m = \text{poly}(n)$. We can accomplish this easily by padding the algorithm with dummy queries and padding any instance of LOP to a larger bit-length. We now define a new instance $\prec_x$: $\{0,1\}^{m^2} \times \{0,1\}^{m^2} \to \{0,1\}$ as follows. An element of $\{0,1\}^{m^2}$ is given by a sequence $(u_1, \ldots, u_m)$ with $u_i \in \{0,1\}^m$. To compare two elements $\bar{u} = (u_1, \ldots, u_m), \bar{v} = (v_1, \ldots v_m)$, we find the first index $i \in [m]$ so that $u_i \neq v_i$. We simulate $A(x)$ through the first $i-1$ queries, plugging in $u_j = v_j$ as the answer to the j^{th} LOP oracle query. Once we get to the i^{th} query, we look at the instance $\prec_i$ of LOP defining the next query, and compare u_i, v_i using $\prec_i$; we then order $\bar{u}, \bar{v}$ accordingly, e.g. $\bar{u} \prec_x \bar{v}$ if $u_i \prec_i v_i$. It follows by construction that $\prec_x$ is a total order, and its least element corresponds to the unique sequence of correct oracle responses in the computation $A(x)$. Therefore by the correctness of A we may read off in polynomial time from this sequence the solution to the search problem R on input x. □

At this point we can prove Theorem 1 that Weak Avoid is polynomial time many-one reducible to LOP:

Proof of Theorem 1. By [Kor21] there is a P^{NP} reduction from the general Weak Avoid problem to the special case when the stretch is $2n$. Composing this with the reductions in Lemmas 11 and 12 we obtain a P^{NP} reduction from Weak Avoid to LOP. Applying Theorem 5 this in turn yields a polynomial-time many-one reduction. □

Recall that LOP enjoys the special property of having essentially unique solutions. Together with its closure under P^{NP} reductions, this endows LOP with the unusual property of being equivalent to a decision problem. We define now a decisional complexity class $\mathsf{L}_2^{\mathsf{P}}$, possessing a few equivalent definitions, whose complete problem is equivalent to the search problem LOP:

Definition (Reminder of Definition 5)**.** *A language L is in the complexity class $\mathsf{L}_2^{\mathsf{P}}$ if there is a polynomial time relation $R : (\{0,1\}^*)^3 \to \{0,1\}$ and a polynomial p, so that for all x, $R(x,\cdot,\cdot)$ defines a total order on $\{0,1\}^{p(n)}$ whose minimal element a has $a_1 = L(x)$.*

Recalling Lemma 5, we have the following upper bound for this class:

Observation 6. $\mathsf{L}_2^{\mathsf{P}} \subseteq \mathsf{S}_2^{\mathsf{P}}$.

At first sight $\mathsf{L}_2^{\mathsf{P}}$ seems to crucially involve a promise, namely that $\prec_x$ defines a total order for all x, just like the promise defining the class $\mathsf{S}_2^{\mathsf{P}}$. However it turns out that this promise can be eliminated and LOP can be given a purely syntactic characterization. This is summarized in Theorem 6 which we restate here:

Theorem 6 *The following are equivalent for a language L:*

1. $L \in \mathsf{L}_2^{\mathsf{P}}$
2. *L is P^{NP}-Turing reducible to LOP.*
3. *L is polynomial time many-one reducible to LOP.*

Conversely, the search problem LOP is polynomial time truth table reducible to a language in $\mathsf{L}_2^{\mathsf{P}}$.

Proof. If $L \in \mathsf{L}_2^{\mathsf{P}}$ then by definition it is many-one reducible to LOP which gives (1) $\to$ (2). Theorem 5 gives (2) $\to$ (3); inspecting the proof, for each x we may define $\prec_x$ so that the value $L(x)$ is the first bit of the LOP solution to $\prec_x$, which actually gives (2) $\to$ (1). □

Finally we prove Theorem 7 from the introduction, which tells us that $\mathsf{L}_2^{\mathsf{P}}$ satisfies two of the three most interesting properties of the larger class $\mathsf{S}_2^{\mathsf{P}}$:

Theorem 7

1. $\mathsf{P}^{\mathsf{NP}} \subseteq \mathsf{L}_2^{\mathsf{P}}$ *and* $\mathsf{BPP} \subseteq \mathsf{MA} \subseteq \mathsf{L}_2^{\mathsf{P}}$
2. $\mathsf{L}_2^{\mathsf{E}}$[6] *contains a language of circuit complexity $2^n/n$.*

Proof. The inclusion $\mathsf{P}^{\mathsf{NP}} \subseteq \mathsf{L}_2^{\mathsf{P}}$ follows directly from the closure of $\mathsf{L}_2^{\mathsf{P}}$ under P^{NP} reductions. For the second inclusion, we know that every language in MA is P^{NP}-reducible to the construction of a $O(\log n)$-seed length PRG for $O(n)$-size circuits [NW94]. The explicit construction of such PRGs is in turn reducible to Weak Avoid [Kor21], and Weak Avoid is reducible to LOP by Theorem 1. Applying Theorem 5 the result follows.

For the second part, the proof is identical to the case of $\mathsf{S}_2^{\mathsf{E}}$ shown in [Li24] given our reduction from Weak Avoid to LOP. Given $x \in \{0,1\}^n$ we may produce in $2^{O(n)}$ time an instance of Weak Avoid $C_n : \{0,1\}^\ell \to \{0,1\}^{2^n}$ so that any solution is a truth table of a function $f : \{0,1\}^n \to \{0,1\}$ of circuit complexity $2^n/n$. By Theorem 1 we may then construct in $2^{O(n)}$ time a nontrivial instance of LOP $\prec$ so that from the unique minimal element of $\prec$ we may read off a uniquely defined Weak Avoid solution f_n for C_n. We then accept/reject x based on $f_n(x)$. Using the closure properties of $\mathsf{L}_2^{\mathsf{E}}$ the language $\{f_n\}_{n\in\mathbb{N}}$ thus defined lies in $\mathsf{L}_2^{\mathsf{E}}$. □

[6] $\mathsf{L}_2^{\mathsf{E}}$ is the exponential-time analogue of $\mathsf{L}_2^{\mathsf{P}}$, where we replace "polynomial time" with "$2^{O(n)}$ time" in its definition

The only interesting property of S_2^P which we cannot prove is inherited by its subclass L_2^P is the Karp-Lipton theorem:

Theorem ([Cai07], credited to Sengupta)**.** *If* $\mathsf{NP} \in \mathsf{P/poly}$ *then* $\mathsf{PH} = \mathsf{S}_2^\mathsf{P}$.

The proof of this result does not seem to generalize to L_2^P.

References

[Ajt83] Miklós Ajtai. $\sum^1_1$-formulae on finite structures. *Ann. Pure Appl. Log.*, 24(1):1–48, 1983.

[AT13] Albert Atserias and Neil Thapen. The ordering principle in a fragment of approximate counting. *Electron. Colloquium Comput. Complex.*, TR13-149, 2013.

[BCE+98a] Paul Beame, Stephen Cook, Jeff Edmonds, Russell Impagliazzo, and Toniann Pitassi. The relative complexity of np search problems. *Journal of Computer and System Sciences*, 57(1):3–19, 1998.

[BCE+98b] Paul Beame, Stephen A. Cook, Jeff Edmonds, Russell Impagliazzo, and Toniann Pitassi. The relative complexity of NP search problems. *J. Comput. Syst. Sci.*, 57(1):3–19, 1998.

[Bea94] Paul Beame. A switching lemma primer. *Technical Report, Department of Computer Science, University of Washington*, 1994.

[BIK+92] Paul Beame, Russell Impagliazzo, Jan Krajícek, Toniann Pitassi, Pavel Pudlák, and Alan R. Woods. Exponential lower bounds for the pigeonhole principle. In S. Rao Kosaraju, Mike Fellows, Avi Wigderson, and John A. Ellis, editors, *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pages 200–220. ACM, 1992.

[BKT14] Samuel R. Buss, Leszek Aleksander Kolodziejczyk, and Neil Thapen. Fragments of approximate counting. *J. Symb. Log.*, 79(2):496–525, 2014.

[Cai07] Jin-Yi Cai. S2p in zppnp. *Journal of Computer and System Sciences*, 73(1):25–35, 2007.

[Can96] Ran Canetti. More on bpp and the polynomial-time hierarchy. *Information Processing Letters*, 57(5):237–241, 1996.

[CHLR23] Yeyuan Chen, Yizhi Huang, Jiatu Li, and Hanlin Ren. Range avoidance, remote point, and hard partial truth table via satisfying-pairs algorithms. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC 2023, page 1058–1066, New York, NY, USA, 2023. Association for Computing Machinery.

[CHR24] Lijie Chen, Shuichi Hirahara, and Hanlin Ren. Symmetric exponential time requires near-maximum circuit size. In *56th Annual Symposium on Theory of Computing*, 2024.

[CIY97] Stephen Cook, Russell Impagliazzo, and Tomoyuki Yamakami. A tight relationship between generic oracles and type-2 complexity theory. *Information and Computation*, 137(2):159–170, 1997.

[CL23] Yilei Chen and Jiatu Li. Hardness of range avoidance and remote point for restricted circuits via cryptography. Cryptology ePrint Archive, Paper 2023/1894, 2023. `https://eprint.iacr.org/2023/1894`.

[FSS84] Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Math. Syst. Theory*, 17(1):13–27, 1984.

[FY96] Lance Fortnow and Tomoyuki Yamakami. Generic separations. *Journal of Computer and System Sciences*, 52(1):191–197, 1996.

[GGNS23] Karthik Gajulapalli, Alexander Golovnev, Satyajeet Nagargoje, and Sidhant Saraogi. Range Avoidance for Constant Depth Circuits: Hardness and Algorithms. In Nicole Megow and Adam Smith, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023)*, volume 275 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 65:1–65:18, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[GLW22] Venkatesan Guruswami, Xin Lyu, and Xiuhan Wang. Range Avoidance for Low-Depth Circuits and Connections to Pseudorandomness. In Amit Chakrabarti and Chaitanya Swamy, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022)*, volume 245 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:21, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[ILW23] Rahul Ilango, Jiatu Li, and R. Ryan Williams. Indistinguishability obfuscation, range avoidance, and bounded arithmetic. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC 2023, page 1076–1089, New York, NY, USA, 2023. Association for Computing Machinery.

[IW97] Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '97, page 220–229, New York, NY, USA, 1997. Association for Computing Machinery.

[Jeř04] Emil Jeřábek. Dual weak pigeonhole principle, boolean complexity, and derandomization. *Annals of Pure and Applied Logic*, 129(1):1–37, 2004.

[Kan82] R. Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55(1):40–56, 1982.

[KKMP21] Robert Kleinberg, Oliver Korten, Daniel Mitropolsky, and Christos Papadimitriou. Total Functions in the Polynomial Hierarchy. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*, volume 185 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 44:1–44:18, Dagstuhl, Germany, 2021. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[KL80] Richard M. Karp and Richard J. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing*, STOC '80, page 302–309, New York, NY, USA, 1980. Association for Computing Machinery.

[Kor21] Oliver Korten. The hardest explicit construction. In *62nd Annual Symposium on Foundations of Computer Science*, 2021.

[Kor22] Oliver Korten. Derandomization from time-space tradeoffs. In *Proceedings of the 37th Computational Complexity Conference*, CCC '22, Dagstuhl, DEU, 2022. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[Li24] Zeyong Li. Symmetric exponential time requires near-maximum circuit size: Simplified, truly uniform. In *56th Annual Symposium on Theory of Computing*, 2024.

[MPW02] Alexis Maciel, Toniann Pitassi, and Alan R. Woods. A new proof of the weak pigeonhole principle. *J. Comput. Syst. Sci.*, 64(4):843–872, 2002.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994.

[PWW88] J. Paris, A. Wilkie, and Alan R. Woods. Provability of the pigeonhole principle and the existence of infinitely many primes. *J. Symb. Log.*, 53:1235–1244, 1988.

[Raz] Alexander Razborov. Personal communication.

[Raz93] A. Razborov. *Bounded Arithmetic and Lower Bounds in Boolean Complexity, In Feasible Mathematics II*. Birkhauser, 1993.

[RS98] Alexander Russell and Ravi Sundaram. Symmetric alternation captures bpp. *computational complexity*, 7:152–162, 1998.

[VW23] Nikhil Vyas and Ryan Williams. On Oracles and Algorithmic Methods for Proving Lower Bounds. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference (ITCS 2023)*, volume 251 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 99:1–99:26, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[Wil83] Christopher B. Wilson. Relativized circuit complexity. In *24th Annual Symposium on Foundations of Computer Science (sfcs 1983)*, pages 329–334, 1983.

A Proof of the Pigeonhole Switching Lemma

As in the standard proof of the switching Lemma, we we need to define a procedure for representing a matching disjunction by a *canonical* complete pigeonhole decision tree. We start with some preliminary definitions.

Definition 22 (Complete Pigeonhole Trees on a subset A)**.** *For a partial matching π and a hole restriction τ, we say that τ covers π if* $\text{nodes}(\tau) \supseteq \text{nodes}(\pi)$*. If $\tau = \langle \sigma, E \rangle$, we say that τ is a minimal cover of π if it fails to be a cover after removing any edge from σ or any element from E.*

For every tuple (A, N, M) where $A \subseteq [N] \sqcup [M]$, we define a canonical complete pigeonhole decision tree, denoted denoted by $\mathcal{C}_A(N, M)$, as follows. This tree is unlabelled, and is defined with respect to some fixed ordering of the elements of A. So lets assume the ordering $a_1, \ldots, a_r$ of A. We build $\mathcal{C}_A(N, M)$ in r steps:

1. *In step 1, if $a_1 \in [N]$, we query pigeon a_1 at the root, with all possible outedges: (a_i, y), $y \in [M]$. Otherwise if $a_1 \in [M]$, we query hole a_1 with all possible outedges: $(x, a_i), x \in [N] \sqcup \{\perp\}$.*

2. *In step $i > 1$, for each leaf node l in T, let $\tau_l = \langle \sigma_l, E_l \rangle$ denote the hole restriction associated with the path to l. If a_i is a pigeon or hole in $nodes(\tau_l)$ (that is, if a_i has already been "determined" by the hole restriction thus far), then we do nothing. Otherwise, we query a_i at the leaf l, with all possible outedges that are consistent with τ_l.*

Observe that the leaves of $\mathcal{C}_A(N, M)$ are in one-to-one correspondence with the set of all hole restrictions that minimally cover A, and thus $\mathcal{C}_A(N, M)$ is a complete pigeonhole decision tree.

Definition 23 (Canonical Pigeonhole Trees)**.** *Let $\phi = \pi_1 \vee \ldots \vee \pi_s$ be a matching disjunction over $[N] \sqcup [M]$ where we have fixed this ordering on the terms. The canonical pigeonhole tree for ϕ, $\mathcal{T}_\phi(N, M)$, is defined recursively below. When N, M is clear from context, we will abbreviate $\mathcal{T}_\phi(N, M)$ by $\mathcal{T}_\phi$.*

1. *If $s = 0$, then ϕ is a constant function. If $\phi = 1$ then $\mathcal{T}_\phi(N, M)$ consists of a single node labelled by 1, and similarly if ϕ is 0, then the decision tree is a single node labelled by 0.*

2. *Otherwise, let π_1 be the first term in ϕ according to the ordering of terms in ϕ. First, we create the complete pigeonhole tree $\mathcal{C}_{nodes(\pi_1)}(N, M)$. For each leaf l in the tree constructed so far, let $\tau_l = \langle \sigma_l, E_l \rangle$ be the hole restriction associated with the path to l. Then for each leaf l:*

   (a) *Let $P \subseteq [N]$ be the subset of pigeons that have not been matched by π_l, and let $H \subseteq [M]$ be the subset of holes not matched by π_l and not in E_l. P and H define the pigeons and holes that are still undetermined after applying τ_l. If the size of P is N' and the size of H is M', then we can rename the vertices so that we identify P with $[N']$ and H with $[M']$.*

   (b) *Recursively construct $\mathcal{T}_{\phi \restriction \tau_l}(N', M')$. and attach a copy of this subtree to the leaf l.*

It follows from the definition that $\mathcal{T}_\phi \equiv \phi$. Next, we restate the Pigeonhole Switching Lemma.

Lemma 6 *Let $M, N, d \in \mathbb{N}$. Let ϕ be a width-w matching disjunction over $[M]^{[N]}$. If $M - N \leq K \leq \frac{N}{4}$ and N, K, d, w sufficiently large, then:*

$$\Pr_{\rho \sim \mathcal{M}_K}[\mathcal{T}_{\phi \restriction \rho} \text{ has depth } \geq d] \leq \exp\Big(d(\log wK^5 - \log N^{1/2} + O(1))\Big)$$

Proof. Let $\phi = \lambda_1 \vee \ldots \vee \lambda_s$ be a matching disjunction of width w. Let $\mathcal{B}$ be the set of partial matchings ρ with $N - K$ edges, such that $\mathcal{T}_{\phi \restriction \rho}$ has depth $\geq d$. If $\rho \in \mathcal{B}$, there exists a total assignment $f \supseteq \rho$ yielding a path P of length d from the root in $\mathcal{T}_{\phi \restriction \rho}$. We define a sequence of hole restrictions $\tau_1, \ldots, \tau_\ell$ and a sequence of partial matchings $\sigma_1, \ldots, \sigma_\ell$, and set $S \subseteq [\ell] \times [w]$ as follows. Define $\tau_0 = \emptyset$. Once τ_i has been defined, let a_{i+1} be the index of the first term in ϕ so that $\lambda_{a_{i+1}} \restriction \rho \cup \tau_1 \cup \cdots \cup \tau_i$ is unkilled. Then let τ_{i+1} be the minimal map $\tau_{i+1} \subseteq f$ which covers $\lambda_{a_{i+1}} \restriction \rho \cup \tau_1 \cup \cdots \cup \tau_i$, and let $\sigma_i = \lambda_{a_{i+1}} \restriction \rho \cup \tau_1 \cup \cdots \cup \tau_i$. Finally let $S \subseteq [\ell] \times [w]$ contain each index $(i, j) \in [\ell] \times [w]$ such that the j^{th} edge of λ_i occurs in σ_i. We claim that such a sequence can

be constructed so that at the end $\ell \leq d$, and if we take $\sigma = \cup_i \sigma_i$, then σ is a partial matching with $\text{nodes}(\sigma) \cap \text{nodes}(\rho) = \emptyset$ and $|\sigma| = t$ for some $\frac{d}{2} \leq t \leq dw$. This follows from the definition of $\mathcal{T}_{\phi \restriction \rho}$ and and the fact that f achieves a path of length $\geq d$ in this decision tree.

Thus, $\rho \in \mathcal{M}_K$, while $\rho \cup \sigma \in \mathcal{M}_{K-t}$ for some $\frac{d}{2} \leq t \leq dw$. We will define a map

$$\text{Dec} : \bigcup_{t \geq \frac{d}{2}} \mathcal{M}_{K-t} \times \Gamma_t \times \Delta_t \to \mathcal{M}_K$$

for some finite sets $\{\Delta_t\}_{t \leq N}$ so that whenever $\rho \in \mathcal{B}$ and σ, S are constructed from ρ as above, there exists some "advice" $\delta \in \Delta_t$ so that $\text{Dec}(\rho \cup \sigma, S, \delta) = \rho$. This will imply that

$$\Pr_{\rho \sim \mathcal{M}_K}[\rho \in \mathcal{B}] \leq \frac{1}{|\mathcal{M}_K|} \sum_{t \geq \frac{d}{2}} |\mathcal{M}_{K-t}||\Gamma_t||\Delta_t|$$

which will yield the theorem provided we can choose the sets Γ_t, Δ_t sufficiently small.

Let $\rho \in \mathcal{B}$, τ_i, a_i, σ_i, S be as above. Define $\gamma_i := \rho \cup \tau_1 \cup \ldots \cup \tau_i \cup \sigma_{i+1} \cup \ldots \cup \sigma_\ell$, where $\gamma_0 = \rho \cup \sigma$. Let $\sigma = \cup_i \sigma_i$ and $t = |\sigma_i|$. First, we claim that if we are given $\tau_1, \ldots, \tau_{i-1}$, $\sigma_1, \ldots, \sigma_{i-1}$, and and γ_{i-1} then we may decode from these the index a_i. To do this, we simply search for the first satisfied term in $\phi \restriction \gamma_{i-1}$. Next, observe that if we have knowledge of a_i, then using S we may reconstruct σ_i. Once we have $\gamma_{i-1}, \sigma_i, \tau_i$, we may construct γ_i directly, by removing σ_i in γ_{i-1} and replacing it with τ_i. So it remains to show how to specify τ using the extra information Δ_t, so that we may recover τ_i from σ_i. Obverse that $\text{nodes}(\tau)$ is contained in $\text{nodes}(\sigma) \cup ([N] \sqcup [M] \setminus \text{nodes}(\rho \cup \sigma))$. Therefore if we construct a map $\delta : [K] \to [K + L]$, where $M - N = L$ and $|\delta| \leq 2d$ which specifies, for each $u \in \text{nodes}(\sigma)$ the preimage/image of u in τ (or the lack of preimage if u is marked empty), then from σ_i we may reconstruct τ_i by searching for the images/preimages of the nodes in $[K] \sqcup [K + L]$ corresponding to $\text{nodes}(\sigma_i)$. Thus for every t we can take $\Delta_t = \Delta$ to be the set of all hole restrictions $[K] \to [K + L]$ of size at most d, so that $|\Delta| \leq \binom{K}{2d}\binom{K+L}{2d}(2d)!$. Finally it remains to show that the sets S lie in Γ_t for some suitably small set Γ_t. Observe that the set $S \subseteq [\ell] \times [w]$ constructed in this argument has the following form: $|S \cap \{i\} \times [w]| \geq 1$ for all $i \in \ell$, and $|S| \leq t$. There are at most $(2w)^t$ sets of this form (ranging over all possible values of ℓ), thus we can take $|\Gamma_t| = (2w)^t$.

It remains to estimate the value

$$\begin{aligned}\Pr_{\rho \sim \mathcal{M}_K}[\rho \in \mathcal{B}] &\leq \frac{1}{|\mathcal{M}_K|} \sum_{t \geq \frac{d}{2}} |\mathcal{M}_{K-t}||\Gamma_t||\Delta_t| \\ &= \sum_{t \geq \frac{d}{2}} \frac{|\mathcal{M}_{K-d}|}{|\mathcal{M}_K|} |\Gamma_t||\Delta_t|\end{aligned}$$

Observe that for each $\kappa \in \mathcal{M}_{K-t}$ there are at least $\binom{N-K+t}{t}$ elements $\alpha \in \mathcal{M}_K$ such that $\alpha \subseteq \kappa$, and on the other hand for any $\alpha \in \mathcal{M}_K$ there are at most $\binom{K}{t}\binom{K+L}{t}t!$ elements $\kappa \in \mathcal{M}_{K-t}$ so that $\alpha \subseteq \kappa$. Thus

$$\frac{|\mathcal{M}_{K-t}|}{|\mathcal{M}_K|} \leq \frac{\binom{K}{t}\binom{K+L}{t}t!}{\binom{N-K+t}{t}}$$

So overall we have:

$$\begin{aligned}\Pr_{\rho\sim\mathcal{M}_K}[\rho\in\mathcal{B}] &\leq \sum_{t\geq\frac{d}{2}} \frac{\binom{K}{t}\binom{K+L}{t}t!}{\binom{N-K+t}{t}}|\Gamma_t||\Delta_t| \\ &\leq |\Delta|\sum_{t\geq\frac{d}{2}} \frac{\binom{K}{t}\binom{2K}{t}t!}{\binom{N-K+t}{t}}(2w)^t \\ \leq \binom{K}{2d}\binom{K+L}{2d}(2d)!\sum_{t\geq\frac{d}{2}} \frac{\binom{K}{t}\binom{2K}{t}t!}{\binom{N-K+t}{t}}(2w)^t &\leq (\frac{2K}{\sqrt{2d}})^{4d}\sum_{t\geq\frac{d}{2}}\left(\frac{8wK^2}{N}\right)^t\end{aligned}$$

Where we apply the assumptions $K \geq L$ and $K \leq \frac{N}{4}$. We may safely assume $\frac{wK^2}{N} = o(1)$, otherwise the conclusion of the theorem holds trivially. Thus by geometric decay of terms in the above sum we have, for some absolute $C \in \mathbb{N}$, the bounds:

$$\begin{aligned}\Pr_{\rho\sim\mathcal{M}_K}[\rho\in\mathcal{B}] &\leq C(\frac{2K}{\sqrt{2d}})^{4d}\left(\frac{8wK^2}{N}\right)^{\frac{d}{2}} \\ &\leq (2K)^{4d}\left(\frac{8wK^2}{N}\right)^{\frac{d}{2}}\end{aligned}$$

provided K, d are sufficiently large with respect to C. So overall

$$\begin{aligned}\Pr_{\rho\sim\mathcal{M}_K}[\rho\in\mathcal{B}] &\leq \exp(4d\log(2K) + \frac{d}{2}\log(8wK^2) - \frac{d}{2}\log N) \\ \leq \exp(d(\log(16K^4) + \log(8wK) + \log N^{1/2})) &= \exp\Big(d(\log wK^5 - \log N^{1/2} + O(1))\Big)\end{aligned}$$

□

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