Electronic Colloquium on Computational Complexity, Report No. 55 (2026)

Res(log) Proves Bounded-Depth Frege Lower Bounds

Ben Davis and Robert Robere*
School of Computer Science
McGill University

April 10, 2026

Abstract

Given a propositional proof system P, we may define a formula $\mathsf{Prf}_s^P(F)$ which is satisfiable if and only if the formula F has a length $\leq s$ refutation in P. These formulas have received much attention in recent years due to their fundamental nature — if a powerful proof system P cannot refute $\mathsf{Prf}_s^Q(F)$, and P can formalize a set of techniques, then we cannot expect these techniques to prove lower bounds on Q proofs — as well as due to their relationship to proving non-automatability of proof systems, as pioneered by Atserias and Müller [AM20].

We provide, to our knowledge, one of the first *positive* results regarding these principles. Let $\mathsf{Prf}_{s,l}^{\mathrm{Frege}_d}$ denote the Prf formula for depth-d Frege proofs of length s and in which each line has size bounded by l. We show that in the regime of exponential length $s = 2^{n^\varepsilon}$, and polynomial-line size $l = n^{O(1)}$, depth-d Frege proofs can refute $\mathsf{Prf}_{s,l}^{\mathrm{Frege}_d}(\mathsf{PHP}_n^{n+1})$, and hence we give an example of a proof system P which seemingly can prove its own lower bounds. In fact, our upper bounds can already be implemented in the fragment Res(log), and hence even an apparently limited subsystem of depth-d Frege is already capable of proving depth-d Frege lower bounds. To the best of our knowledge, this is the first example of a propositional proof system which is capable of proving strong lower bounds against itself. We prove our results by using a constructive proof of the switching lemma due to Razborov [Raz95], particularly its presentation by Urquhart and Fu [UF96], and combining it with a recent study of $\mathsf{Prf}^{\mathrm{Res}}$ by Li, Li, and Ren [LLR24], in order to reduce the problem of refuting $\mathsf{Prf}_{s,l}^{\mathrm{Frege}_d}(\mathsf{PHP}_n^{n+1})$ to refuting the weak pigeonhole principle.

1 Introduction

Separating NP from coNP is equivalent to proving superpolynomial lower bounds on the lengths of proofs for every propositional proof system [CR79]. We are far from this goal, being only capable of proving strong lower bounds on proof length in relatively weak proof systems, particularly, those in which we can adopt lower bound techniques from circuit complexity (e.g. [Ajt94, Hak85]). Why has progress stalled here? Is it possible to devise some formal barrier results, like the relativization [BGS75], algebrization [AW09], or natural proof [Raz15] barriers in classical complexity, which can explain this lack of progress?

Meta-Proof-Complexity, Refutation Principles, and Automatability. A natural approach to these questions is to understand if we can formalize proof complexity lower bounds inside propositional proof systems. For any propositional proof system P, we can define a CNF formula $\mathsf{Prf}_s^P(F)$ which is satisfiable if and only if there is a size $\leq s$ *refutation*[1] of the CNF formula F (see Section 2.4 for formal definitions). If F is unsatisfiable, but requires superpolynomial-length P-refutations, then the formula $\mathsf{Prf}_{n^{O(1)}}^P(F)$ will itself be unsatisfiable. In this case, we can ask: what proof systems can *efficiently refute* $\mathsf{Prf}_{n^{O(1)}}^P(F)$? If we believe that proving lower bounds on P-proofs is hard, then we should believe that it would require a powerful proof

*Both authors supported by NSERC.

[1] This formula is typically defined with respect to *proofs*, not *refutations*, but as we will be primarily considering refutation systems in this paper we fix this definition.

ISSN 1433-8092

system to refute $\mathsf{Prf}^P_{n^{O(1)}}(F)$ efficiently. Even a conditional result in this direction would be interesting, as if we choose a proof system P which can formalize a set of lower bound techniques, and can prove that P is not able to efficiently refute $\mathsf{Prf}^P_{n^{O(1)}}(F)$ under some believable assumption, then this is evidence that these techniques are insufficient to prove these proof complexity lower bounds.

Open Problem 1.1. For any proof system P and unsatisfiable formula F, does P efficiently refute $\mathsf{Prf}^P_s(F)$?

Pudlák suggested that the above problem is a fundamental open question regarding proof systems [Pud20]. However, the question of whether P can prove its own lower bounds is closely related to other central problems, such as the question of automating P. Recall that a propositional proof system P is *automatable* [BPR00] if there is an algorithm A which, given a propositional tautology T as input, outputs a P-proof of T in time proportional to the length of the optimal P-proof. Automatability captures the complexity of *proof search*, as many practical heuristic algorithms for NP search problems can be viewed as automating certain underlying proof systems[2]. We can now prove that certain "strong-enough" proof systems are not automatable under cryptographic assumptions [KP98, BPR00] by formalizing the totality of cryptographic primitives inside the systems themselves — for instance, this can be used to show that the powerful *Extended Frege* system is not automatable unless Factoring is efficiently computable in polynomial time [KP98]. However, for weaker proof systems which cannot reason about these primitives, such as Resolution, only weaker results were known regarding their automatability [AR08] until the breakthrough result of Atserias and Müller [AM20], who proved Resolution is not automatable unless $\mathsf{P} = \mathsf{NP}$.

Let us briefly sketch how the Atserias-Müller argument proceeds, as it crucially uses the Prf principles. For a proof system P and a formula F, let $s_P(F)$ denote the size of the shortest P-refutation of F, where $s_P(F) = \infty$ if there is no P-refutation. Suppose there is an automating algorithm A for Resolution which runs in time polynomial in $s_{\text{Res}}(F)$ for every unsatisfiable formula F. Consider the following "candidate reduction" from 3-SAT to the automatability problem: given a 3-CNF formula F, output the new CNF formula $\mathsf{Prf}^{\text{Res}}_{n^3}(F)$. If F is satisfiable then $\mathsf{Prf}^{\text{Res}}_{n^3}(F)$ is unsatisfiable, and in this case Pudlák [Pud03] proved that there is a polynomial-size refutation of $\mathsf{Prf}^{\text{Res}}_{n^3}(F)$. On the other hand, if F is unsatisfiable, then either $\mathsf{Prf}^{\text{Res}}_{n^3}(F)$ is satisfiable, in which case there is no Res refutation of $\mathsf{Prf}^{\text{Res}}_{n^3}(F)$, or $\mathsf{Prf}^{\text{Res}}_{n^3}(F)$ is unsatisfiable. Therefore, if we can prove superpolynomial lower bounds against Resolution refutations of $\mathsf{Prf}^{\text{Res}}_{n^3}(F)$ when F is unsatisfiable, then we can use our hypothesized automating algorithm to determine if F is satisfiable, simply by running it for long enough to guarantee that we are not in the satisfying case. The above argument generalizes beyond Resolution, so we record it as an observation here:

Observation 1.2. *Let P be any proof system with the property that for any satisfiable formula F, P has polynomial-size refutations of $\mathsf{Prf}^P_{n^{O(1)}}(F)$. If P requires superpolynomial-length refutations of $\mathsf{Prf}^P_{n^{O(1)}}(F)$ whenever F is unsatisfiable, then P is not automatable in polynomial time unless $\mathsf{P} = \mathsf{NP}$.*

Hence, beyond being a question of purely fundamental interest, understanding when proof systems can prove their own lower bounds may also have theoretical implications regarding their automation.

On the Hardness of Refuting Refutation Principles. After Atserias and Müller's breakthrough, several follow-up works extended their results to other weak proof systems by proving lower bounds for Prf principles. The work of de Rezende et. al. [dRGN+21] gave an alternative proof of Atserias and Müller's result via a reduction to a formula called the *retraction weak pigeonhole principle*, introduced by Jeřábek [Jeř07], and here denoted[3] Lossy (cf. Section 2.5). The work of [dRGN+21] showed that any proof system with efficient refutations of $\mathsf{Prf}^{\text{Res}}_{n^3}(F)$ for unsatisfiable F also has efficient refutations of Lossy. This allowed them to extend the non-automatability results to other proof systems which cannot refute Lossy, such as the Nullstellensatz and Polynomial Calculus systems. Afterwards, other extensions were shown for *cutting planes* [GKMP20] and *Res(k)* systems [Gar24], each of which are now known to be hard to automate unless $\mathsf{P} = \mathsf{NP}$.

[2]For instance, modern CDCL SAT solvers can, under certain configurations, be viewed as automating the classical *Resolution* proof system, though not necessarily in polynomial time.

[3]Formally, our Lossy formula is slightly different than the retraction weak pigeonhole principle, as we use a binary encoding instead of a unary encoding. The terminology Lossy was introduced by Korten [Kor22] in the setting of TFNP. We use TFNP language prominently in this work, so we opt to use this name of the formula so as not to introduce confusion.

Despite this progress, some notable proof systems remain without fully satisfactory results: the *Sherali-Adams* proof system[4], and the *bounded-depth Frege* proof systems. The Sherali-Adams proof system is known to be able to efficiently refute the Lossy formulas, and hence the reduction argument of de Rezende et al [dRGN+21] does not apply. Currently, it is still an open question if Sherali-Adams is automatable (assuming $\mathsf{P} \neq \mathsf{NP}$), or if Sherali-Adams can refute $\mathsf{Prf}_s^{\mathrm{Res}}(F)$ when the formula is unsatisfiable. For bounded-depth Frege, the work of Papamakarios [Pap24] showed that depth-d Frege systems are not automatable on depth-d tautologies via a modification of the Atserias-Müller argument. However, the argument of Papamakarios does not proceed by proving bounded-depth Frege lower bounds for Prf formulas directly, and instead considers a depth-d substitution of these formulas. Hence, it is also unknown whether or not, say, depth-d Frege can efficiently refute $\mathsf{Prf}_s^{\mathrm{Frege}_d}(F)$ whenever $s_{\mathrm{Frege}_d}(F) > s$.

1.1 Our Results

Given the above results, a natural hypothesis is that Open Problem 1.1 never has a positive answer: namely, proof systems can never prove their own lower bounds efficiently. Our main contribution is that this hypothesis is too strong in all parameter regimes, and in some regimes bounded-depth Frege *can* prove strong lower bounds on its own length. Moreover, these proofs can even be carried out in Res(log), the fragment of Frege whose lines are DNFs with $\log(S)$-width $\wedge$s, where S is the size of the proof [Kra19, Section 5.7]. In the statement below, let $\mathsf{Prf}_{L,S}^d(F)$ denote the formula which is satisfiable if there is a depth-d Frege refutation of F with L lines, and in which each line is a depth-d formula of size at most S. Let PHP_n^{n+1} denote the propositional pigeonhole principle, stating (falsely) that there is an injective mapping from $n+1$ pigeons to n holes. We note that it is known [Ajt94, BIK+92, UF96, Hås23] that PHP_n^{n+1} requires exponential size to refute in bounded-depth Frege.

Theorem 1.3. *For any $d \in \mathbb{N}, 0 < \delta < (1/5)^d$, and $c \in \mathbb{N}$, there is a size-$2^{n^{c+O(\delta)}}$ Res(log) refutation of $\mathsf{Prf}_{1.01^{n^\delta},n^c}^d(\mathsf{PHP}_n^{n+1})$.*

We give a few remarks on this theorem. First, the length of the Res(log) refutation is $2^{n^{c+O(\delta)}}$, but the formula $\mathsf{Prf}_{1.01^{n^\delta},n^c}^d(\mathsf{PHP}_n^{n+1})$ itself has size $\exp(O(n^\delta))$, and hence the Res(log) refutation is quasi-polynomial size. As we will later see, bringing the length of this refutation down to polynomial size is closely related to how efficiently bounded-depth Frege can refute the weak pigeonhole principle $\mathsf{PHP}_{n/2}^n$. Second, we note that Res(log), which is equivalent to depth-0.5 Frege, is capable of proving exponential lower bounds against depth-d Frege for *any* d, provided the sizes of the lines are bounded to be a polynomial. In other words: not only can Res(log) prove lower bounds against the fragment of its *own* proofs with polynomial-size formulas, it can even prove lower bounds against proofs with *arbitrary depth*, polynomial-size formulas. (That being said, we note it is a notorious open problem to separate the bounded-depth Frege hierarchy on CNF contradictions, and even giving more than a quasi-polynomial separation between Res(log) and depth-$o(\log n)$ Frege is wide open.) Third, and finally, we note that in our $\mathsf{Prf}_{1.01^{n^\delta},n^c}^d(\mathsf{PHP}_n^{n+1})$ formula we are required to bound the line size to be polynomial. This restriction seems to be an inherent limitation of our proof technique. We will discuss it further in our technical overview next, and remark that improving it is an intriguing open problem.

We prove our main result by formalizing the standard proof of depth-d Frege lower bounds of PHP_n^{n+1} as a reduction to the weak pigeonhole principle in Res(log). Thanks to the work of Maciel, Pitassi, and Woods [MPW00], the weak pigeonhole principle can be refuted in Res(log) in quasipolynomial size, and so chaining these two proofs together finishes the argument. Unfortunately, the best known upper bound for refuting the weak pigeonhole principle in bounded-depth Frege is quasi-polynomial in size, and this quasi-polynomial is one of the main reasons we obtain a quasi-polynomial upper bound. In fact, any improvement to refuting the weak pigeonhole principle in bounded-depth Frege will directly translate to an improvement to our main theorem, as we show in our next theorem:

Theorem 1.4. *Let k, n be positive integers. If there is a size-$S(n)$, depth-k Frege refutation of $\mathsf{PHP}_{n/2}^n$, then for any $d \in \mathbb{N}, 0 < \delta < (1/5)^d$, and $c \in \mathbb{N}$, there is a depth-$(k+2)$ Frege refutation of $\mathsf{Prf}_{1.01^{n^\delta},n^c}^d(\mathsf{PHP}_n^{n+1})$*

[4]The initial version of [dRGN+21] claimed that Sherali-Adams was not automatable unless $\mathsf{P} \neq \mathsf{NP}$, but this was later retracted.

of size $2^{O(n^c)} \cdot (2^{O(n^\delta)} + S(2^{n^\delta}))$.

We find it useful to view the previous theorem as a "proof-complexity reduction" from Prf^d to $\mathsf{PHP}^n_{n/2}$: improving the size of refutations of $\mathsf{PHP}^n_{n/2}$ translates immediately to an improved size on refutations of $\mathsf{Prf}^d(\mathsf{PHP}^{n+1}_n)$. On the other hand, the results of de Rezende et al [dRGN+21], further tightened by Li, Li, and Ren [LLR24], can be seen as providing a matching reduction from Lossy (which bounded-depth Frege can prove, by a modification of the techniques in Section 3, is equivalent to $\mathsf{PHP}^n_{n/2}$) to $\mathsf{Prf}^{\mathrm{Res}}_{1.01^n}(F)$. Hence in the regime of proving exponential-length lower bounds, and from the perspective of bounded-depth Frege, the complexity of $\mathsf{Prf}^d_{1.01^{n^\delta},n^c}$ is equivalent to that of refuting $\mathsf{PHP}^n_{n/2}$, up to the 2^{n^c} price paid for the line size. This suggests the possibility that proving non-automatability of bounded-depth Frege on CNF formulas by the Atserias-Müller framework in this parameter regime may necessarily require proving lower bounds on bounded-depth Frege proofs of the weak pigeonhole principle, which is a well-known open problem in proof complexity.

Technical Overview. We prove our main theorem by formalizing the proof that PHP^{n+1}_n requires exponential-size bounded-depth Frege lower bounds, particularly via the use of a switching lemma. This argument (originated by Ajtai [Ajt94], although we follow the presentation of Urquhart and Fu [UF96]) proceeds by proving a variation of the *switching lemma* used to prove AC^0 circuit lower bounds [Hås86], called the *matching switching lemma*. (It will be particularly important for us to use the *constructive* proof of the switching lemma, due to Razborov [Raz95] and applied in proof complexity by Urquhart and Fu [UF96], which proceeds by encoding all partial restrictions which cause the switching lemma to fail into a small set of codes.)

Let us overview this proof first. Given a purported depth-d Frege refutation of PHP^{n+1}_n, the argument proceeds by using the switching lemma to construct a non-standard truth assignment, called a *k-evaluation*, for every formula that occurs in the proof. The k-evaluations will be defined so that the axioms of the pigeonhole principle PHP^{n+1}_n are "true" under the evaluation, the final contradiction is "false", and yet the rules of bounded-depth Frege are still sound with respect to the propagation of truth under the k-evaluation. We build the k-evaluations for each subformula in the proof inductively by depth, using the switching lemma to build k-evaluations for depth $i+1$ subformulas from the k-evalutions for formulas of depth i subformulas.

To formalize this in bounded-depth Frege, we generalize an argument of Li, Li, and Ren [LLR24, Theorem 3.16], who showed how to reduce the problem of refuting $\mathsf{Prf}^{\mathrm{Res}}_{1.01^n}$ to that of refuting a particular formula denoted $\mathsf{Lossy}(\mathsf{Iter})$. This formula is built as a composition of two formulas, Lossy_N and Iter_L:

- The Iter_L formula embodies a simple local search principle. The input variables s encode a map $s : [L] \to [L]$ which must be monotone non-decreasing, in the sense that $s(u) \geq u$ and $s(1) > 1$, and the principle claims that there is a node u with $s(s(u)) = s(u)$.
- The Lossy_N formula, as we referenced earlier, embodies a weaker version of the pigeonhole principle. Namely, the input variables encode two maps $f : [N] \to [N/2]$ and $g : [N/2] \to [N]$, and the principle claims that there is a witness x that g is not the left-inverse of f (i.e. $g(f(x)) \neq x$).

The $\mathsf{Lossy}(\mathsf{Iter})$ formula uses the Iter_L formula to add a layer of indirection to Lossy: now the forward map f is encoded by solutions to an inner Iter_L instance. See Section 2 for more details.

The main idea of our argument is to use a supposed refutation of PHP^{n+1}_n in bounded-depth Frege, combined with the constructive proof of the switching lemma, in order to compress a certain set of random partial matchings perfectly into a strictly smaller set of perfect matchings. We can then use any matching *not* in this compressed set to construct a k-evaluation for the set of all formulas appearing in the proof, and we will use the Lossy problem to find this matching. More formally, the domain of our Lossy instance will be the space of *sequences* of partial matchings of pigeons to holes, and the range of the Lossy instance will be a small set of "codes", as defined in the proof of the constructive switching lemma. Given such a sequence of partial matchings M of $n - n^{\varepsilon^d}$ pigeons to holes, we apply the switching lemma to either (a) construct a k-evaluation for all lines in the proof restricted by M, or (b) discover some sub-formula in the proof on which this construction fails. On the one hand, if we successfully constructed a k-evaluation supported on all lines of the proof, then we simply map the matchings M to some arbitrary code. On the other hand, if an application of the switching lemma failed at some line of the proof, we detect this by using the inner Iter_L instance to perform a local search through the proposed Frege proof, restricted under the sequence M, such

that the solutions of the Iter instance correspond to sub-formulas in which the switching lemma fails. At the same time, we can verify each step of the proof to locate any errors. If we arrive at a bad sub-formula, we can use the constructive proof of the switching lemma to encode this "bad" partial matching into its corresponding code. If we locate an error in the proof, however, we encode it arbitrarily. Since this is the only case where we cannot uniquely encode an Iter solution, the solutions to the entire $\mathsf{Lossy}(\mathsf{Iter})$ instance correspond exactly to errors in the refutation.

The main technical difficulty is in encoding d iterative applications of the switching lemma into a single Lossy instance. To do this, we (roughly speaking) must formalize all these switches as happening "in parallel", and our argument to this effect is somewhat reminiscent of the fact that the search problem corresponding to Lossy is Turing-Closed [LPT24].

Afterwards, there are two main hurdles remaining. First, we must formalize the above reduction in bounded-depth Frege (indeed, in Res(log)). To do this, we show how to implement the reduction purely in the language of search problems using the theory of black-box TFNP. It will turn out that the reduction we sketched above can be implemented by a low-depth decision-tree reduction between the false-clause search problems associated $\mathsf{Prf}^d_{1.01^n,n^c}(\mathsf{PHP}^{n+1}_n)$ and $\mathsf{Lossy}(\mathsf{Iter})$, and this implementation is easy to formalize directly in Res(log) (cf. Section 2.2). After this, the second hurdle is showing how to refute the $\mathsf{Lossy}(\mathsf{Iter})$ formula inside of Res(log). We do this by a second reduction to the weak pigeonhole principle — essentially, we show how to "remove" the inner Iter instance at the cost of losing the retraction pointers g in our Lossy instance, all inside of Res(log). Fortunately, we have good upper bounds for the weak pigeonhole principle [MPW00] in Res(log), and composing these two proofs together completes the argument.

Potential Improvements. The most obvious open problem left by our work is improving the bound on the length of our refutation of Prf^d from quasi-polynomial to polynomial. There are three main obstacles to this improvement.

The first obstacle is that all known bounded-depth Frege proofs of the weak pigeonhole principle are quasipolynomial size, and improving this is a fascinating open question [MPW00]. We view our work as added motivation for such an improvement, as the weak pigeonhole principle seemingly plays a central role in formalizing bounded-depth Frege lower bounds.

The second, and somewhat less significant, obstacle involves the particulars of the matching switching lemma. Crucially, our reduction to the $\mathsf{Lossy}(\mathsf{Iter})$ instance via the switching lemma requires a large space of matchings, since the known proofs of the matching switching lemma in proof complexity require sampling from the space of all partial matchings that leave $\ell \ll n/2$ pigeons alive. Perhaps the most natural way to surmount this obstacle is via a derandomized switching lemma, for instance, by porting the techniques of Trevisan and Xue [TX13] to the proof complexity setting. This would introduce an added difficulty, as we would then need to be able to reason about the derandomized switching lemma inside bounded-depth Frege.

The final obstacle is that our proof requires us to have an exponential gap between the number of lines L and the size of each line S in $\mathsf{Prf}^d_{L,S}$. Indeed, while our upper bound is quasipolynomial in the size of the formula $\mathsf{Prf}^d_{L,S}$, it is *exponential* in the line-size parameter S. This blow-up occurs when we are searching for a line where the switching lemma fails in the proof. In this case, we brute-force over all possible subformulas at each line, causing a blow-up of size 2^S. This brute-force search is actually crucial, as it helps us simplify the argument in two significant ways.

The first simplification is that it allows us to avoid implementing the complete proof of the constructive switching lemma (i.e. by building the canonical decision tree) inside bounded-depth Frege directly. This seems quite difficult to do, since the standard algorithm which produces the canonical decision tree does not seem implementable inside of AC^0, and hence it is unclear how to argue about its correctness inside of bounded-depth Frege.

The second simplification is that it allows us to sidestep a subtle "renaming" argument in the proof of the pigeonhole principle lower bound[5]. Essentially, each time we restrict a proposed depth-d Frege refutation of PHP^{n+1}_n by a random partial matching that leaves ℓ holes alive, we are left with a smaller copy of the pigeonhole principle $\mathsf{PHP}^{\ell+1}_\ell$, but on some *random* set of $\ell+1$ pigeons and a *random* set of ℓ holes. In order to continue the lower-bound argument inside depth-d Frege in a consistent way, we must be able to rename

[5]We thank an anonymous reviewer for pointing out this subtlety.

the $\ell+1$ unrestricted pigeons and ℓ holes to the sets $[\ell+1]$ and $[\ell]$, respectively, but doing this is tantamount to proving $\mathsf{PHP}_{\ell}^{\ell+1}$ inside of depth-d Frege, which we know to be impossible.

Therefore, to avoid this brute-force step, we must deal with the problems underlying these simplifications directly, by coming up with a "parallelizable" proof of the switching lemma, and also by determining a way around the renaming issue we just described. Given all three of these obstacles, it is a very interesting question whether or not our upper bound can be improved from quasi-polynomial to polynomial, which we leave for future work.

1.2 Related Work

There are a few important works which are related to our results, but which we have not yet reviewed. Perhaps the most prominent is the work of Razborov [Raz95], who investigated which theories of bounded arithmetic are necessary to formalize lower bound methods in circuit complexity. These questions led him to produce the constructive form of the switching lemma which we exploit here. We note that Razborov's regime is different from our own, as he is primarily interested in circuit complexity lower bounds, as opposed to proof complexity lower bounds. Later work of Razborov and others used *lower bounds* on the weak pigeonhole principle to prove lower bounds on the circuit-size tautologies, which can be viewed as the circuit complexity analogues of the Prf formulas [Raz98, ABRW02, Kra04, Raz15].

Santhanam and Tzameret [ST21] considered the $\mathsf{Prf}^{\mathrm{IPS}}$ formulas for the *Ideal Proof System* (IPS). They proved the striking result that proving separations in algebraic complexity (namely, $\mathsf{VP} \neq \mathsf{VNP}$) holds if and only if IPS does not efficiently refute $\mathsf{Prf}^{\mathrm{IPS}}_{n^c}(\text{“}\mathsf{VP} = \mathsf{VNP}\text{”})$. In other words, $\mathsf{VP} \neq \mathsf{VNP}$ holds if and only if IPS cannot prove $\mathsf{VP} = \mathsf{VNP}$. They also studied iterations of the formulas $\mathsf{Prf}(\mathsf{Prf}), \mathsf{Prf}(\mathsf{Prf}(\mathsf{Prf}))$, and conjectured that these sequences provide an example of hard tautologies for *every* proof system.

Another recent and closely related family of results is due to Arteche et al [AAdRK25]. There, they show that the *entire* Atserias-Mueller argument for Resolution can itself be formalized in Extended Frege. When combined with win-win arguments by studying iterations of the Prf formula, their constructive proof has numerous interesting consequences. Of particular interest to us is a bounded-depth Frege lower bound for *some* $\mathsf{Prf}^{\mathrm{Res}}_{n^{O(1)}}(F)$ formula. In fact, they show that Resolution lower bounds are actually universal, in the sense that if a lower bound for a (strong enough) propositional proof system P holds, then a lower bound holds in P for some $\mathsf{Prf}^{\mathrm{Res}}(F)$ formula. (Although, we note that this was already implicitly shown by [Iwa97], who proved that calculating the Resolution refutation size exactly is NP-Complete).

1.3 Open Problems

Finally, we review a few interesting problems left open by our work.

- Is it possible to efficiently refute $\mathsf{Prf}^{d}_{L,S}(\mathsf{PHP}^{n+1}_{n})$ where L and S are polynomially related, in bounded-depth Frege (or any standard propositional proof system)? As discussed above, this would require actually analyzing the formulas comprising the proof themselves, since we cannot just semantically read them in anymore.

- As an easier goal, can we simply reduce the exponential lower bound on the length parameter L? One could hope to approach this by formalizing a *derandomized* switching lemma, in the style of Trevisan and Xue [TX13].

- Can the weak pigeonhole principle $\mathsf{PHP}^{n}_{n/2}$ be refuted in polynomial-size in bounded-depth Frege? Or, can we prove *any* lower bound (even polynomial) for the weak pigeonhole principle in the same system?

- Essentially the only other prominent weak proof systems for which non-automatability results remain open are the semi-algebraic systems Sherali-Adams and Sums-of-Squares. Both of these systems can refute Lossy in polynomial size, and hence the reduction of de Rezende et al [dRGN+21] does not apply. Can these systems efficiently refute $\mathsf{Lossy}(\mathsf{Iter})$?

- We have not attempted to formalize our argument in bounded arithmetic, opting instead to formalize it using the language of search problems and TFNP. Given the connection between Res(log) and the bounded arithmetic theory $T^2_2(\alpha)$ [Kra01], our argument is very likely formalizable in $T^2_2(\alpha)$, and it may be instructive to do so.

2 Definitions

2.1 Proof Systems

We will deal with binary boolean formulas over the connectives $\vee, \neg$. We can nest these connectives to simulate connectives of higher fan-in in the natural way, and we will use $A_1 \wedge \cdots \wedge A_m$ to denote $\neg(\neg A_1 \vee \cdots \vee \neg A_m)$. The *depth* of a formula is the maximum number of sequences $\neg(\vee)$ occurring along any of its root-literal branches, coinciding with the number of alternations between $\wedge$ and $\vee$ (for example, $a \wedge (b \vee c)$ and $a \vee (b \wedge c)$ have depth 1, while a and $a \vee \neg b$ have depth 0). The *size* of a formula is the number of variables and connectives appearing in it. The *width*, also called *bottom fan-in*, of a formula is the maximum fan-in of its bottom-most sub-formulas.

A *literal* is either a variable or its negation, a *clause* is an $\vee$ of literals, and a *term* is an $\wedge$ of literals. A *CNF* (Conjunctive Normal Form) formula is an $\wedge$ of clauses, and a *DNF* (Disjunctive Normal Form) formula is an $\vee$ of terms. If a CNF or DNF has width k, we call it a k-CNF/DNF.

We will often talk about a conjunction $\bigwedge \mathcal{F}$ and the set of formulas $\mathcal{F}$ interchangeably, except when we are required to make a distinction.

Definition 2.1. A *Resolution* proof of a set of clauses $\mathcal{D}$ from another set of clauses $\mathcal{C}$ is a sequence of clauses $D_1, ..., D_m$, which we call the *lines* of the proof, such that $\mathcal{D} \subseteq \{D_1, ..., D_m\}$ and for each $i \in [m]$, D_i is either a clause from $\mathcal{C}$ or is derived from previous lines according to the following inference rules:

$$\textbf{\textit{Weakening}}\ \frac{C}{C \vee \ell} \qquad\qquad \textbf{\textit{Cut}}\ \frac{C \vee \ell \qquad D \vee \neg \ell}{C \vee D}$$

where C and D may be any clauses, and ℓ is any literal. The *length* of the proof is m, the *width* is the maximum width of its lines, and the *size* is the total size of its lines. If $\bot \in \mathcal{D}$, we call the proof a *refutation* of $\mathcal{C}$.

Definition 2.2. Let $\mathcal{F}$ be a finite set of inference rules, of the form:

$$\frac{A_1 \quad \cdots \quad A_k}{B}$$

where $A_1, \ldots, A_k, B$ are boolean formulas over $\vee, \neg$. An *$\mathcal{F}$-proof* of a set of formulas $\mathcal{H}$ from another set of formulas $\mathcal{G}$ is a sequence of formulas $F_1, \ldots, F_m$ such that $\mathcal{H} \subseteq \{F_1, \ldots, F_m\}$ and for each $i \in [m]$, F_i is either from $\mathcal{G}$ or is derived from previous lines by a substitution of an inference rule of $\mathcal{F}$. The *length* of the proof is m, the *depth* of the proof is the maximum depth of $F_1, \ldots, F_m$, the *line size* of the proof is the maximum size of any of $F_1, \ldots, F_m$, and the total *size* of the proof is the sum of the sizes of $F_1, \ldots, F_m$. Again, if $\bot \in \mathcal{H}$, the proof is a *refutation* of $\mathcal{G}$. We will say that a proof Π has depth $d + 0.5$ to mean that all lines have logical depth d but width at most $\log S$, where S is the total size of Π.

A proof system $\mathcal{F}$ is *sound* if whenever there is an $\mathcal{F}$-proof of $\mathcal{H}$ from $\mathcal{G}$, $\mathcal{G} \vDash \mathcal{H}$, and *implicationally complete* if whenever $\mathcal{G} \vDash \mathcal{H}$, there is an $\mathcal{F}$-proof of $\mathcal{H}$ from $\mathcal{G}$. The system $\mathcal{F}$ is a *Frege system* if both are true.

We will work with *constant-depth* proofs, where we additionally require that the depths of lines in the proof are bounded by a constant. While all (implicationally complete) Frege systems are polynomially-equivalent [CR79] and therefore the lower bound techniques we will present also work for constant-depth proofs in any Frege system, the transformation does not preserve depth, so when making claims about the exact depths of proofs (such as our upper bound) it is important to specify the system. For this work, we will use Shoenfield's system [Sho67, p21]:

$$\textbf{\textit{Excluded Middle}}\ \frac{}{A \vee \neg A} \qquad \textbf{\textit{Weakening}}\ \frac{A}{A \vee B} \qquad \textbf{\textit{Cut}}\ \frac{A \vee C \qquad B \vee \neg C}{A \vee B}$$

$$\textbf{\textit{Contraction}}\ \frac{A \vee (B \vee B)}{A \vee B} \qquad \textbf{\textit{Associativity}}\ \frac{A \vee (B \vee (C \vee D))}{A \vee ((B \vee C) \vee D)} \qquad \textbf{\textit{Commutativity}}\ \frac{(A \vee B) \vee C}{(B \vee A) \vee C}$$

We will usually omit any explicit mention of Contraction, Associativity, and Commutativity, so that we can treat connectives as having arbitrary arity.

2.2 TFNP Background

A useful tool for analyzing propositional proofs is the theory of *black-box search problems*. These are sequences of relations $R_n \subseteq \{0,1\}^n \times \mathcal{O}_n, n \in \mathbb{N}$, where given $x \in \{0,1\}^n$, a solution to the search problem is any $o \in \mathcal{O}_n$ such that $(x,o) \in R_n$. In this setting, "efficient computation" is carried out by polylogarithmic-depth (in n) decision trees.

Definition 2.3. TFNP^{dt} is the class of search problems $\{R_n \subseteq \{0,1\}^n \times \mathcal{O}_n\}_n$ which are *total* and *efficiently-verifiable*, meaning that:

- For each $x \in \{0,1\}^n$, there is some $o \in \mathcal{O}_n$ for which $(x,o) \in R_n$, and
- For each $n \in \mathbb{N}$, there is a collection of depth-$\mathsf{polylog}(n)$ decision trees $\{T_o\}_{o \in \mathcal{O}_n}$ so that for each pair $(x,o) \in \{0,1\}^n \times \mathcal{O}_n$, $T_o(x) = 1 \iff (x,o) \in R_n$.

When working with a decision tree T, we will use the notation $\mathrm{Br}_o(T)$ to denote the set of branches of T ending in a leaf labelled by o, and $\mathrm{Br}(T)$ to denote the set of all branches of T. We can compare the power of problems in TFNP^{dt} using the following notion of reducibility.

Definition 2.4. Let $R \subseteq \{0,1\}^n \times \mathcal{O}_R$ and $S \subseteq \{0,1\}^m \times \mathcal{O}_S$ be search problems. A *decision tree reduction* from R to S is a collection of decision trees $f_i : \{0,1\}^n \to \{0,1\}$ for each $i \in [m]$ and $g_o : \{0,1\}^n \to \mathcal{O}_R$ for each $o \in \mathcal{O}_S$ such that:

$$((f_1(x), ..., f_m(x)), o) \in S \implies (x, g_o(x)) \in R$$

The *decision tree depth* of the reduction is the maximum depth of these decision trees.

The connection between search problems and proofs is exhibited by the following family of search problems.

Definition 2.5. Let $F(x_1, \ldots, x_n) = C_1 \wedge \cdots \wedge C_m$ be a CNF formula. The search problem $\mathrm{Search}(F)$ is defined as

$$\mathrm{Search}(F) = \{(x,i) \in \{0,1\}^n \times [m] : C_i(x) = 0\}.$$

We note $\mathrm{Search}(F) \in \mathsf{TFNP}^{dt}$ whenever F is unsatisfiable and has width $\mathsf{polylog}(n)$.

Lemma 2.6. *For every search problem $\{R_n\} \in \mathsf{TFNP}^{dt}$, there is a sequence of width-$\mathsf{polylog}(n)$ CNF formulas F_{R_n} such that R_n is reducible to $\mathrm{Search}(F_{R_n})$ in decision-tree depth 1.*

Proof. Fix $n \in \mathbb{N}$. Let $\mathcal{O}_{R_n}$ be the solution set of R_n, and $\{T_o\}_{o \in \mathcal{O}_{R_n}}$ the set of polylog-depth decision trees verifying R_n. Define:

$$F_{R_n} = \bigwedge_{o \in \mathcal{O}_{R_n}} \bigwedge_{\pi \in \mathrm{Br}_1(T_o)} C_\pi$$

where for a branch π of a decision tree T, C_π is the natural conjunction encoding that π is followed when evaluating T. The reduction assigns variables of F_{R_n} identically to input variables of R_n, and maps each solution C_π of $\mathrm{Search}(F_{R_n})$ to the solution $o \in \mathcal{O}_{R_n}$ for which $\pi \in \mathrm{Br}_1(T_o)$. □

Because of this correspondence between CNF formulas and NP search problems, we will usually abuse notation and use F to refer to both the CNF formula F and the search problem $\mathrm{Search}(F)$. The reason that this perspective on unsatisfiable CNFs will be useful to us is that efficient bounded-depth Frege refutations are "closed" under decision tree reductions, in the sense of the following theorems:

Theorem 2.7. *Let F and G be unsatisfiable CNF formulas and suppose there is both a size-S, depth-1 Frege refutation of G where each formula has width at most w, and a decision tree depth-k reduction from $\mathrm{Search}(F)$ to $\mathrm{Search}(G)$. Then there is also a size-$(S \cdot 2^{O(wk)})$, depth-1 Frege refutation of F where each formula has width at most wk.*

Theorem 2.8. *Let F and G be unsatisfiable CNF formulas and suppose there is both a size-S, depth-d Frege refutation of G and a decision tree depth-k reduction from $\mathrm{Search}(F)$ to $\mathrm{Search}(G)$. Then there is also a size-$(S \cdot 2^{O(k)})$, depth-$(d+1)$ Frege refutation of F.*

Before proving this, we will need the following lemmas which allow us to prove basic facts about the propositionalizations of decision tree paths.

Lemma 2.9. *Let T be a depth-k decision tree, and let $\{A_\pi\}_{\pi \in \mathrm{Br}(T)}$ be a set of size-s, depth-d formulas of the form $A_\pi = F_\pi \vee \neg C_\pi$. Then there is a size-$O(s \cdot 2^{2k})$, depth-$d$ proof of the formula $\bigvee_{\pi \in \mathrm{Br}(T)} F_\pi$ from $\{A_\pi\}_{\pi \in \mathrm{Br}(T)}$.*

Proof. Consider any length-k branch π of T. Suppose x was the last query made along π and the response was b. Let π' be the branch of T which is identical to π, except that the response to this query is instead $\neg b$. We can cut the formulas A_π and $A_{\pi'}$ on x to obtain the formula $A_{\pi_0} = F_\pi \vee F_{\pi'} \vee C_{\pi_0}$, where π_0 is the common sub-branch of π and π' which omits the final query. Iterating this, we can similarly derive $A_\pi = \bigvee_{\pi \subseteq \pi' \in \mathrm{Br}(T)} F_\pi \vee C_\pi$ for each length-i sub-branch π of T, for $i = k-1, ..., 0$. After at most 2^k steps, we obtain $A_\emptyset = \bigvee_{\pi \in \mathrm{Br}(T)} F_\pi$ as desired. □

Lemma 2.10. *If T is a depth-k decision tree, then there is a size-$O(k \cdot 2^{3k})$ depth-1, and width-k proof of the tautology $\bigvee_{\pi \in \mathrm{Br}(T)} C_\pi$*

Proof. By applying excluded middle, we can derive $C_\pi \vee \neg C_\pi$ for all $\pi \in \mathrm{Br}(T)$. Applying Lemma 2.9 completes the proof. □

We prove Theorem 2.7 first.

Proof of Theorem 2.7. Let $\Pi = D_1, ..., D_m$ be the promised refutation of G. We will replace each sub-formula H in the proof with a substituted formula H^+ and show that each line D_i^+ can still be efficiently derived from $\{D_j^+ \mid j < i\}$. For a single literal ℓ, ℓ^+ is the disjunction $\bigvee_{\pi \in \mathrm{Br}_1(T_\ell)} C_\pi$, where for a literal ℓ, T_ℓ is the decision tree in the reduction computing it (negate the tree for x if $\ell = \neg x$). For a conjunction C of the form $\ell_1 \wedge \cdots \wedge \ell_s$, C^+ is the disjunction:

$$\bigvee_{\pi_1 \in \mathrm{Br}_1(T_{\ell_1}), ..., \pi_s \in \mathrm{Br}_1(T_{\ell_s}} (C_{\pi_1} \wedge \cdots \wedge C_{\pi_s})$$

Finally, for a disjunction (of any depth) $D = A_1 \vee \cdots \vee A_s$, $D^+ = A_1^+ \vee \cdots \vee A_s^+$.

We will handle the lines D_i which are not clauses of G first. We have multiple cases depending on how D_i was derived:

- ***Excluded Middle.*** If D_i was derived by excluded middle, then it is of the form:

$$\bigvee_i \ell_i \vee \bigwedge_i \neg \ell_i$$

  Therefore D_i^+ is:

$$\bigvee_i \bigvee_{\pi \in \mathrm{Br}_1(T_{\ell_i})} C_\pi \vee \bigvee_{\pi_1 \in \mathrm{Br}_0(T_{\ell_1}), ..., \pi_s \in \mathrm{Br}_0(T_{\ell_s}} (C_{\pi_1} \wedge \cdots \wedge C_{\pi_s})$$

  To derive this, we can instead apply excluded middle on each term $C_{pi_1} \wedge \cdots \wedge C_{\pi_s}$ of this, obtaining $\neg C_{\pi_1} \vee \cdots \vee \neg C_{\pi_s} \vee (C_{\pi_1} \wedge \cdots \wedge C_{\pi_s})$ for each choice of $\pi_1 \in \mathrm{Br}_1(T_{\ell_1}), ..., \pi_s \in \mathrm{Br}_1(T_{\ell_s})$. Fix some choice of $\pi_2, ..., \pi_s$, then use Lemma 2.10 to derive $\bigvee_{\pi \in \mathrm{Br}(T_{\ell_1})} C_\pi$ and cut it with the corresponding formulas to obtain:

$$\bigvee_{\pi \in \mathrm{Br}_1(T_{\ell_1})} C_\pi \vee \neg C_{\pi_2} \vee \cdots \vee \neg C_{\pi_s} \vee \bigvee_{\pi_1 \in \mathrm{Br}_0(T_{\ell_1})} (C_{\pi_1} \wedge \cdots \wedge C_{\pi_s})$$

  Repeating this, we derive this formula for each choice of $\pi_2, ..., \pi_s$, and iterating for $\ell_2, ..., \ell_s$ derives D_i^+.

- ***Cut.*** If $D_i = A \vee B$ was derived from previous lines $A \vee C$ and $B \vee \neg C$ by cutting on C, then C is of the form $\ell_1 \vee \cdots \vee \ell_s$ and $\neg C = \neg\ell_1 \vee \cdots \neg\ell_s$. Therefore $(A \vee C)^+$ is:

$$A^+ \vee \bigvee_i \bigvee_{\pi \in \mathrm{Br}_1(T_{\ell_i})} C_\pi$$

and $(B \vee \neg C)^+$ is:

$$B^+ \vee \bigvee_{\pi_1 \in \mathrm{Br}_0(T_{\ell_1}), \ldots, \pi_s \in \mathrm{Br}_0(T_{\ell_s})} (C_{\pi_1} \vee \cdots \vee C_{\pi_s})$$

Fix some choice of $\pi_1 \in \mathrm{Br}_0(T_{\ell_1}), ..., \pi_s \in \mathrm{Br}_0(T_{\ell_s})$. For each $\pi \in \mathrm{Br}_1(T_{\ell_1})$, $\neg C_\pi \vee \neg C_{\pi_1}$ is a tautology which can be derived by applying excluded middle on any query on which π and π_1 disagree, then weakening. Cutting $\neg C_\pi \vee \neg C_{\pi_1}$ with $(A \vee C)^+$ for each $\pi \in \mathrm{Br}_1(T_{\ell_1})$ derives:

$$A^+ \vee \neg C_{\pi_1} \vee \bigvee_{i>1} \bigvee_{\pi \in \mathrm{Br}_1(T_{\ell_i})} C_\pi$$

Iterating this for $\ell_2, ..., \ell_s$, we derive $A^+ \vee \neg C_{\pi_1} \vee \cdots \vee \neg C_{\pi_s}$, and after repeating for each choice of $\pi_1, ..., \pi_s$ we can cut the resulting formulas with $(B \vee \neg C)^+$ to derive $A^+ \vee B^+ = D_i^+$.

- ***Weakening, Contraction, Associativity.*** Applications of the remaining rules are still valid without modification.

Finally, we need to handle lines D_i which are clauses of G. For this, define for each clause $C = \ell_1 \vee \cdots \ell_s$ another substitution $C^- = \{\neg C_{\pi_1} \vee \cdots \vee \neg C_{\pi_s} \mid \pi_1 \in \mathrm{Br}_0(T_{\ell_1}), ..., \pi_s \in \mathrm{Br}_0(T_{\ell_s})\}$. It is straightforward to derive C^+ from this using Lemma 2.10, and observe that by the definition of a reduction, $D \vee \neg C_\pi$ must a weakening of some clause of F for each $D \in C^-$ and $\pi \in \mathrm{Br}(T_C)$, where T_C is the return tree corresponding to C in the reduction. Therefore, since Lemma 2.9 allows us to derive C^- from the set of such clauses, we can also derive C^+ from F. □

The proof of Theorem 2.8 is similar, except that we can be less careful with how we substitute.

Proof of Theorem 2.8. Let $\Pi = F_1, ..., F_m$ be the refutation of G. For each literal ℓ, define the formulas $\ell^+ = \bigvee_{\pi \in \mathrm{Br}_1(T_\ell)} C_\pi$ and $\ell^- = \bigwedge_{\pi \in \mathrm{Br}_0(T_\ell)} \neg C_\pi$. For each line F_i of Π, define $F_i' = F_i[\ell_1^-/\ell_1, ..., \ell_{2n}^+/\ell_{2n}]$ if d is odd, where $\ell_1, ..., \ell_{2n}$ are all literals over the variables of G. If d is even, then we obtain F' by instead substituting ℓ^+ for each literal ℓ, in either case increasing the depth of the formula by 1. As before, we will show that for each i, we can derive F_i' from the substitutions of the previous lines $\{F_j' \mid j < i\}$ by simulating the inference used to derive F_i:

- ***Excluded Middle.*** If $F_i = A \vee \neg A$, then unless A is a literal this is still a valid application of excluded middle. Otherwise, $F_i = \ell \vee \neg\ell$ for some literal ℓ. We begin by deriving the tautology $\bigvee_{\pi \in \mathrm{Br}(T_\ell)} C_\pi$ by Lemma 2.10. If d is even, this is already F_i'. Otherwise, we can derive from this the formula $\neg C_\pi \vee \bigvee_{\pi \in \mathrm{Br}_0(T_\ell)} C_\pi$ for each $\pi \in \mathrm{Br}_0(T_\ell)$ by cutting it with each formula $\neg C_\pi \vee \neg C_{\pi'}, \pi' \in \mathrm{Br}_1(T_\ell)$, which in turn can be derived by applying excluded middle and weakening. We can then cut this set of formulas with $\bigvee_{\pi \in \mathrm{Br}_0(T_\ell)} C_\pi \vee \bigwedge_{\pi \in \mathrm{Br}_0(T_\ell)} \neg C_\pi$ to obtain $\bigwedge_{\pi \in \mathrm{Br}_0(T_\ell)} \neg C_\pi \vee \bigvee_{\pi \in \mathrm{Br}_0(T_\ell)} C_\pi$. The rest of the formula can be converted in a symmetric way.

- ***Cut.*** If $F_i = A \vee B$ was derived by cutting previous lines $A \vee C$ and $B \vee \neg C$ on C, then again unless C is a literal this is still a valid application of cut. Otherwise, let $C = \ell$. Then we can derive $\ell^- \vee (\neg\ell)^-$ (if d is odd) or $\ell^+ \vee (\neg\ell)^+$ (if d is even) as in the previous case, then cut this with both $(A \vee \ell)'$ and $(B \vee \neg\ell)'$ to obtain $A' \vee B'$.

- ***Weakening, Contraction, Associativity.*** As before, applications of the remaining rules are still valid.

Finally, for clauses $C = \ell_1 \vee \cdots \ell_s$ of G, we can derive C' from the set of clauses $C^- = \{\neg C_{\pi_1} \vee \cdots \vee \neg C_{\pi_s} \mid \pi_1 \in \mathrm{Br}_0(T_{\ell_1}), ..., \pi_s \in \mathrm{Br}_0(T_{\ell_s})\}$ as follows. If d is odd, then for each choice of $\pi_2, ..., \pi_s$, cut the set of corresponding clauses from this set with $\bigwedge_{\pi \in \mathrm{Br}_0(T_{\ell_1})} \neg C_\pi \vee \bigwedge_{\pi \in \mathrm{Br}_1(T_{\ell_1})} \neg C_\pi$, which we already showed how to derive. Repeating this for each $i = 2, ..., s$ yields:

$$\bigwedge_{\pi \in \mathrm{Br}_0(T_{\ell_1})} \neg C_\pi \vee \cdots \vee \bigwedge_{\pi \in \mathrm{Br}_0(T_{\ell_s})} \neg C_\pi = C'$$

Now if d is even, again for each choice of $\pi_2, ..., \pi_s$, cut the corresponding clauses with $\bigvee_{\pi \in \mathrm{Br}(T_{\ell_1})} C_\pi$ instead to obtain the formula

$$\bigvee_{\pi \in \mathrm{Br}_1(T_{\ell_1})} C_\pi \vee \neg C_{\pi_2} \vee \cdots \neg C_{\pi_s}$$

for each $\pi_2, ..., \pi_s$. Iterating for each $i = 2, ..., s$ derives C'. From here, we can use the same argument as in the proof of Theorem 2.7 to derive C^- from F. □

2.3 Binary Variables

When defining our search problems (and their corresponding unsatisfiable CNF formulas), it will be convenient to encode values in binary, so that each variable x with domain D is encoded by coordinates $x_1, ..., x_k$, where we assume $|D| = 2^k$ for some $k \in \mathbb{N}$ (this assumption will be left implicit from now on). To express the value of x with boolean formulas, we use the notation $[\![x = v]\!]$ for $v \in D$ (with binary encoding $v_1, ..., v_k$) to mean:

$$\bigwedge_{i \in [k], v_i = 1} x_i \wedge \bigwedge_{i \in [k], v_i = 0} \neg x_i$$

and similarly, we use $[\![x \neq v]\!]$ to mean the logical negation of $[\![x = v]\!]$:

$$\bigvee_{i \in [k], v_i = 1} \neg x_i \vee \bigvee_{i \in [k], v_i = 0} x_i$$

When dealing with variables encoded in binary, the following special cases of Lemmas 2.9 and 2.10 will be useful.

Lemma 2.11. *Let x be a variable with domain D. Let $\{A_v\}_{v \in D}$ be a set of size-s, depth-d formulas, each of the form $A_v = F_v \vee [\![x \neq v]\!]$. There is a size-$O(s|D|^2)$ depth-d proof of $\bigvee_{v \in D} F_v$ from $\{A_v\}_{v \in D}$.*

Lemma 2.12. *Let x be a binary variable with domain D. There is a size-$O(|D|^3 \log |D|)$, depth-1, and width-$(\log^{O(1)} |D|)$ proof of the formula $\bigvee_{v \in D} [\![x = v]\!]$.*

2.4 Refuter Principles

In this section we carefully define the refuter principles for depth-k Frege refutations. We note that the notation of these principles has not been fully standardized in the literature. Many papers, particularly in the literature on automatability, use Ref (representing "refutation") instead of Prf, although this clashes with the notation more typically used in logic where Ref represents a *reflection principle* for a proof system. We opt for Prf instead, but emphasize that $\mathsf{Prf}^d(F)$ will be satisfiable when there is a *refutation* of F in depth-d Frege.

Definition 2.13. For size parameters $N, L, k \in \mathbb{N}$ and a CNF formula $F = C_1 \wedge \cdots \wedge C_m$ on n variables, the refuter problem $\mathsf{Prf}^k_{N,L}(F)$ takes as input a (purported) depth-k Frege refutation of F. This is encoded as follows:

- The lines are specified by the following variables for each $i \in [N]$ and $j \in [L]$ (indexing the j^{th} gate of line i):
  - A connective $\mathsf{conn}_{i,j} \in \{\vee, \neg, \mathsf{inp}, \bot\}$ specifying the gate, with inp meaning that the gate is an input gate and $\bot$ meaning that the gate is not used,

- A pointer $\mathsf{lit}_{i,j} \in [n]$, specifying the variable used as input for gate j, if $\mathsf{conn}_{i,j} = \mathsf{inp}$,
- A pair of pointers $\mathsf{in}^{\ell}_{i,j}, \mathsf{in}^{r}_{i,j} \in [L]$, specifying the inputs of gate j (ignore one or both if the gate is not binary), and
- A pointer $\mathsf{out}_{i,j} \in [L]$, specifying the unique gate of line i which has gate j as an input

- The inferences are specified by the following for each $i \in [N]$:
  - A rule type $\mathsf{rule}_i \in \{\mathsf{excl.\ middle}, \mathsf{weaken}, \mathsf{cut}, \mathsf{contract}, \mathsf{assoc.}\}$, specifying the rule used to derive line i,
  - Pointers hyp^{ℓ}_i and $\mathsf{hyp}^{r}_i \in [N]$, specifying the (at most two) hypotheses of this rule application,
  - Pointer $\mathsf{act}_i, \mathsf{act}^{\ell}_i, \mathsf{act}^{r}_i \in [L]$ specifying the location, within the conclusion and both hypotheses, of the active formula in the rule application (for example, the cut-formula if line i was derived by a cut), and
  - Pointers $\mathsf{corr}_{i,j} \in [2L]$ and $\mathsf{corr}^{\ell}_{i,j}, \mathsf{corr}^{r}_{i,j} \in [2L]$ for each index $j \in [L]$. These specify mappings of gates in the hypotheses and conclusion to corresponding gates in the other two lines, so that we can locally verify that the sub-formulas in the conclusion have the expected structure.

We interpret these as encoding a proof in the natural way given the variables' descriptions above, with the exception that for $i \in [m]$, the i^{th} line is assumed to be equal to C_i, disregarding the variables specifying the corresponding inference. The goal is to find a pair $(i, j) \in [N] \times [L]$ such that either:

- The j^{th} gate of line i is not well-formed,
- $i \in [m]$ and the j^{th} gate of line i specified by the variables does not match the j^{th} gate of C_i,
- $i > m$ and either $\mathsf{hyp}^{\ell}_i \geq i$ or $\mathsf{hyp}^{r}_i \geq i$ (ie. a circular inference), or
- $i > m$ and the j^{th} gate specified by the variables does not match the inference used to derive line i.

We will omit the full definition of the corresponding formula, but it should be clear that these solutions can be verified in depth $O(\log N + \log L)$ and therefore such a formula exists.

2.5 Coding Principles

In this section we introduce our some intermediate principles which are used in our bounded-depth Frege refutations. The principal one is the *Lossy Code* formula [Kor21], although we will also use a composed formula of Lossy_N with the classical Iteration problem, as introduced by [LLR24]. Note that the Lossy formula is a binary-coded version of the *retraction weak pigeonhole principle* [Jeř07].

Definition 2.14. For a size parameter $N \in \mathbb{N}$, the *Lossy Code* problem, Lossy_N, is a search problem given by:

- Encoding pointers $f(i) \in [N/2]$ for each $i \in [N]$, and
- Decoding pointers $g(j) \in [N]$ for each $j \in [N/2]$

The encoding pointers specify a function $f : [N] \to [N/2]$, and the decoding pointers a function $g : [N/2] \to [N]$. Clearly the image of g must exclude many elements of $[N]$, so it cannot correctly invert f. The goal is to find a witness of this, ie. an element $i \in [N]$ so that $g(f(i)) \neq i$.

When N is a power of 2, the corresponding CNF formula consists of each clause:

$$[\![f(i) \neq j]\!] \vee [\![g(j) \neq k]\!]$$

for $i \neq k \in [N]$ and $j \in [N/2]$.

Our upper bound will use the composition of Lossy with the following classical search problem.

Definition 2.15. For a size parameter L, Iter_L is given by successor pointers $s(u) \in [L]$ for each $u \in [L]$. We interpret $[L]$ as a set of nodes, and we call their labels their *potential*. Each node u must have $s(u) \geq u$, and the node (1) must have a successor of strictly greater potential. The goal is to find either a node which violates this, or else a witness of a proper sink, meaning a node u such that $s(u) > u$ but $s(s(u)) = s(u)$. The corresponding CNF formula has the following clauses:

1. $[\![s_i(1) \neq 1]\!]$ *((1) must be a source)*
2. $[\![s_i(u) \neq v]\!] \qquad \forall v < u \in [L]$ *(potential increases)*
3. $[\![s_i(u) \neq v]\!] \vee [\![s_i(v) \neq v]\!] \qquad \forall u < v \in [L]$ *(no proper sinks)*

The next definition is from [LLR24].

Definition 2.16. For size parameters N and $L \in \mathbb{N}$, the search problem $\mathsf{Lossy}(\mathsf{Iter})_{N,L}$ is given by:

- Successor pointers $s_i(u) \in [L]$ for each $i \in [N]$ and $u \in [L]$,
- Return pointers $h_i(u) \in [N/2]$ for each $i \in [N]$ and $u \in [L]$, and
- Decoding pointers $g(j) \in [N]$ for each $j \in [N/2]$

Each encoding pointer $f(i)$ is replaced by a call to Iter_L, specified by the successor pointers $s_i(u)$. The return pointers $h_i(u)$ specify a function $h_i : [L] \to [N/2]$, mapping solutions of this Iter instance to encodings of i (Note that each element of $[N]$ can now have many encodings). The image of g must still exclude many elements of $[N]$, so it still cannot uniquely decode these encodings. A solution is a a witness of this, which is now a pair $(i, u) \in [N] \times [L]$ such that u is a solution to the Iter instance associated with i, but $g(h_i(u)) \neq i$.

The corresponding CNF formula contains the following groups of clauses, corresponding to the different forms of Iter solutions:

1. $[\![s_i(1) \neq 1]\!] \vee [\![h_i(1) \neq j]\!] \vee [\![g(j) \neq k]\!] \qquad \forall i \neq k \in [N], j \in [N/2]$ *((1) must be a source)*
2. $[\![s_i(u) \neq v]\!] \vee [\![h_i(v) \neq j]\!] \vee [\![g(j) \neq k]\!] \qquad \forall i \neq k \in [N], v < u \in [L], j \in [N/2]$ *(potential increases)*
3. $[\![s_i(u) \neq v]\!] \vee [\![s_i(v) \neq v]\!] \vee [\![h_i(v) \neq j]\!] \vee [\![g(j) \neq k]\!] \qquad \forall i \neq k \in [N], u < v \in [L], j \in [N/2]$ *(no proper sinks)*

2.6 The Pigeonhole Principle

The final formula we will be interested in is the pigeonhole principle. We will need the unary version, which has high-width, so rather than presenting it as a search problem we simply give the formula.

Definition 2.17. For size parameters $n > m \in \mathbb{N}$, the (unary) *pigeonhole principle* PHP^n_m is an unsatisfiable family of clauses given by variables $p_{i,j} \in \{0, 1\}$ for each $i \in [n]$ and $j \in [m]$, specifying a multi-valued function $f : [n] \to [m]$ which we assert to be injective. The clauses of this family are:

- $\bigvee_{j \in [m]} p_{i,j} \qquad \forall i \in [n]$, and
- $\neg p_{i,j} \vee \neg p_{k,j} \qquad \forall i \neq k \in [n], j \in [m]$

For a family of DNF formulas $\mathcal{P} = \{P_{i,j}\}_{i \in [n], j \in [m]}$, the *substituted* pigeonhole principle $\mathsf{PHP}^n_m(\mathcal{P})$ is the family of DNF formulas obtained from PHP^n_m by substituting $P_{i,j}$ for $p_{i,j}$ for all $i \in [n], j \in [m]$, then distributing $\vee$ over $\wedge$.

In the regime where $m \leq n/2$, we call this the *weak pigeonhole principle*.

3 Upper Bounds for Refuting Lossy(Iter)

To prove Theorem 1.3, we will first show that Res(log) can quasi-polynomially refute Lossy(Iter), and then show that $\mathsf{Prf}^{d}_{1.01^{n^\delta},n^c}(\mathsf{PHP}^{n+1}_n)$ can be efficiently reduced to Lossy(Iter). In this section, we first give a quasi-polynomial-size refutation for Lossy(Iter), in Res(log). We then give a second upper bound (used for Theorem 1.4) which we cannot formalize in Res(log), but for which the size is parametrized by the minimum size of a bounded-depth refutation of the weak pigeonhole principle. This has the benefit that any improvements to the currently known upper bounds for constant-depth proofs of the weak pigeonhole principle would also improve our upper bound.

3.1 Depth-0.5 Frege (Res(log)

We begin with the optimal depth lower bound. Our goal for this section is to show:

Theorem 3.1. *There is a size-*$\left(O(NL^2)+N^{O(\log^2 N)}\right)$ *depth-*0.5 *Frege refutation of* $\mathsf{Lossy}(\mathsf{Iter})_{N,L}$.

We will describe this refutation in multiple parts. In the first stage, we follow the natural low-width Resolution refutation of Iter_L to derive a formula related to Lossy_N:

Definition 3.2. For $N < M \in \mathbb{N}$, Avoid^N_M is an unsatisfiable family of DNF formulas over the variables $f(i) \in [M]$ for each $i \in [N]$, encoding a function $f : [N] \to [M]$. The family contains the formulas:

$$\bigvee_{i\in[N]} [\![f(i) = j]\!]$$

for each $j \in [M]$, asserting that f is a surjection.

Theorem 3.3. *There is a size-*$O(N^6+N^3L^2)$ *depth-*0.5 *Frege proof of* $\mathsf{Avoid}^{N/2}_N$ *(up to relabelling of variables) from* $\mathsf{Lossy}(\mathsf{Iter})_{N,L}$.

Proof. Before we can emulate the Resolution refutation, we need to eliminate the pointers $h_i(u)$. At the same time, we will convert the terms $[\![g(j) \neq k]\!]$ to their positive form, deriving the depth-0.5 formulas:

1. $[\![s_i(1) \neq 1]\!] \vee \bigvee_{j\in[N/2]} [\![g(j) = i]\!]$ $\quad \forall i \in [N]$
2. $[\![s_i(u) \neq v]\!] \vee \bigvee_{j\in[N/2]} [\![g(j) = i]\!]$ $\quad \forall i \in [N], v < u \in [L]$
3. $[\![s_i(u) \neq v]\!] \vee [\![s_i(v) \neq v]\!] \vee \bigvee_{j\in[N/2]} [\![g(j) = i]\!]$ $\quad \forall i \in [N], u < v \in [L]$

This step is identical for each of the three forms of formulas, so we show it for the formulas of the first form as an example. We begin by fixing $i \in [N]$ and $j \in [N/2]$, then applying Lemma 2.12 to derive the formula $\bigvee_{k\in[N]} [\![g(j) = k]\!]$. Cutting this on $[\![g(j) = k]\!]$ with the formula $[\![s_i(1) \neq 1]\!] \vee [\![h_i(1) \neq j]\!] \vee [\![g(j) \neq k]\!]$ for each $k \neq i$, we obtain $[\![s_i(1) \neq 1]\!] \vee [\![h_i(1) \neq j]\!] \vee [\![g(j) = i]\!]$. Repeating this derives the same formula for each $i \in [N]$ and $j \in [N/2]$. Finally, we fix i again and apply Lemma 2.11 on the set of clauses $[\![s_i(1) \neq 1]\!] \vee [\![h_i(1) \neq j]\!] \vee [\![g(j) = i]\!]$, $j \in [N/2]$, to complete the derivation of $[\![s_i(1) \neq 1]\!] \vee \bigvee_{j\in[N/2]} [\![g(j) = i]\!]$. This stage of the proof contributes size at most $O(N^6)$.

Once we have repeated the first step to derive the entire family of formulas above, we can finally run the Resolution refutation of Iter_L as promised. We do this separately for each i, so fix $i \in [N]$. The goal of this step is to prove the formula:

$$[\![s_i(u) \neq v]\!] \vee \bigvee_{j\in[N/2]} [\![g(j) = i]\!]$$

for all $u < v \in [L]$. For $u = L$, we have nothing to do since there is no $v > u$. Otherwise, suppose we have shown this for all $u' > u$ and consider u. Intuitively if no $u' > u$ has a successor, then neither can u, or else it would witness a proper sink. Therefore we can use Lemma 2.11 to cut the corresponding formula $[\![s_i(u) \neq u']\!] \vee [\![s_i(u') \neq u']\!] \vee \bigvee_{j\in[N/2]} [\![g(j) = i]\!]$ with the set of formulas $[\![s_i(u') \neq v]\!] \vee \bigvee_{j\in[N/2]} [\![g(j) = i]\!]$,

$v \neq u'$, which we have already derived by assumption for $v > u'$ and which are formulas (2) otherwise. This derives $[\![s_i(u) \neq u']\!] \vee \bigvee_{j\in[N/2]}[\![g(j) = i]\!]$ as desired.

Finally, once we have derived $[\![s_i(1) \neq v]\!] \vee \bigvee_{j\in[N/2]}[\![g(j) = 1]\!]$ for all $v > 1$, we can apply Lemma 2.11 to cut this set of formulas with (1), completing the proof of $\bigvee_{j\in[N/2]}[\![g(j) = i]\!]$. Repeating this for all $i \in [N]$ proves the remaining axioms of $\mathsf{Avoid}_N^{N/2}$, over the variables $g(j)$ rather than $f(j)$, and brings the total size contribution for this step to at most $O(N^3L^2)$. □

The fact that we obtain Avoid here rather than Lossy itself should not be surprising, as Lossy is a Herbrandization of Avoid and the Iter substitution takes the place of the Herbrand variables. It now remains to show that depth-0.5 Frege can efficiently refute $\mathsf{Avoid}_N^{N/2}$. Rather than doing this directly, we will use a well-known upper bound for the substituted weak pigeonhole principle.

Theorem 3.4. *[MPW00] Let $n \in \mathbb{N}$ and $\mathcal{P} = \{P_{i,j}\}_{i\in[n],j\in[n/2]}$ be a set of size-s, width-$(\log^{O(1)} n)$ DNF formulas. Then there is a size-$(ns)^{O(\log n)^2}$, depth-0.5 refutation of $\mathsf{PHP}^n_{n/2}(\mathcal{P})$.*[6]

To see why this relates to what we have derived so far, choose $\mathcal{P} = \{[\![g(j) = i]\!]\}_{i\in[N],j\in[N/2]}$. Then $\mathsf{PHP}^N_{N/2}(\mathcal{P})$ consists of the clauses:

- $\bigvee_{i\in[N/2]}[\![g(i) = j]\!]$ $\quad \forall j \in [N]$
- $[\![g(i) \neq j]\!] \vee [\![g(i) \neq k]\!]$ $\quad \forall i \in [N/2], j \neq k \in [N]$

The first form of formulas are axioms of $\mathsf{Avoid}_N^{N/2}$, which we have shown how to derive, while the second are tautologies which are simple to derive — apply excluded middle on any coordinate on which j and k differ, then weaken. Therefore we can combine Theorem 3.4 with Theorem 3.3 to complete the proof of Theorem 3.1.

3.2 Depth-d Frege

If we do not care about obtaining an optimal-depth refutation, there is an alternative way to use the bound of Maciel, Pitassi, and Woods in the refutation above. This also has the benefit that it works for any bounded-depth Frege refutation of the weak pigeonhole principle, using the observation that the pigeonhole principle is simply the *unary* encoding of Avoid.

Definition 3.5. A formula F is *block-respecting* if for any variable x, with domain D, appearing in it, x only appears as part of sub-formulas $[\![x \neq v]\!]$ or $[\![x = v]\!]$ for some $v \in D$.

Let $\mathcal{F}$ be a family of block-respecting formulas over binary variables $x_1, ..., x_n$, with domains $D_1, ..., D_n$, respectively. The analogous *unary encoding* of $\mathcal{F}$, $\mathsf{u}\mathcal{F}$, is a family of formulas over new variables $x_{i,j} \in \{0, 1\}$ for all $i \in [n]$ and $j \in [D_i]$, which we interpret as indicating whether $x_i = j$. To obtain the formulas of this family, we begin by taking each $F \in \mathcal{F}$ and replacing each sub-formula of F of the form $[\![x_i = j]\!]$ with $x_{i,j}$ and each sub-formula of the form $[\![x_i \neq j]\!]$ with $\neg x_{i,j}$, obtaining a new formula F'. $\mathsf{u}\mathcal{F}$ then contains:

- F' $\quad \forall F \in \mathcal{F}$,
- $\bigvee_{j\in D_i} x_{i,j}$ $\quad \forall i \in [n]$, and
- $\neg x_{i,j} \vee \neg x_{i,k}$ $\quad \forall i \in [n], j \neq k \in D_i$

where the last two forms of formulas encode that for each i, the variables $x_{i,j}$ indeed specify a unique value for x_i.

We will rely on the following fact about Frege refutations of the two encodings.

Theorem 3.6. *If $\mathcal{F}$ is a family of formulas over n binary variables, each of which has domain of size at most m, and there is a size-S, depth-d Frege refutation of $\mathsf{u}\mathcal{F}$, then there is also a size-$(S + nm^2 + nm^2)\lceil\log m\rceil^2$ depth-$(d + 1)$ refutation of $\mathcal{F}$.*

[6] Their final result is not stated for the substituted pigeonhole principle, but the key parts of their upper bound are stated for this variant and it can be easily verified that the remaining steps also work for this version.

Proof. Let $\Pi = G_1, ..., G_S$ be the refutation of $\mathsf{u}\mathcal{F}$, and let $x_1, ..., x_n$ be the variables of $\mathcal{F}$, with domains $D_1, ..., D_n$, respectively. We simply substitute each occurrence of a unary variable $x_{i,j}$ $(i \in [n], j \in D_i)$ by the formula $[\![x_i = j]\!]$, and each occurrence of $\neg x_{i,j}$ by $[\![x_i \neq j]\!]$ to obtain a new proof $\Pi' = G'_1, ..., G'_S$. The only rule applications which this could affect are instances of the cut rule, but since $[\![x_i = j]\!]$ and $[\![x_i \neq j]\!]$ are logical negations, the corresponding inferences in Π' are still valid cuts. However, Π' is now a refutation of the family consisting of all formulas of $\mathcal{F}$, as well as the formulas:

1. $\bigvee_{j \in D_i} [\![x_i = j]\!] \qquad \forall i \in [n]$
2. $[\![x_i \neq j]\!] \vee [\![x_i \neq k]\!] \forall i \in [n], j \neq k \in D_i$

Note that these are tautologies, and both forms of formula are straightforward to derive — for each formula (1), we can appeal to Lemma 2.12. For each formula $[\![x_i \neq j]\!] \vee [\![x_i \neq k]\!]$, $i \in [n], j \neq k \in D_i$, let y be the boolean variable corresponding to a coordinate of x on which j and k differ. Then this formula is a weakening of $y \vee \neg y$, so we can derive it by applying Excluded middle and weakening. By appending these derivations to the beginning of Π', we obtain a refutation of $\mathcal{F}$ as desired, only increasing the length by at most $nm^2 + nm^2 \lceil \log m \rceil$. □

Applying the above definition to Avoid, we get a new family of formulas, $\mathsf{uAvoid}_N^{N/2}$, containing the following clauses:

- $\bigvee_{i \in [N/2]} f_{i,j} \qquad \forall j \in [N]$
- $\bigvee_{j \in [N]} f_{i,j} \qquad \forall i \in [N/2]$
- $\neg f_{i,j} \vee \neg f_{i,k} \qquad \forall i \in [N/2], j \neq k \in [N]$

where the variables are now $f_{i,j}$ for each $i \in [N/2]$ and $j \in [N]$. As promised, this contains all clauses of $\mathsf{PHP}_{N/2}^N$ (over the variables $f_{j,i}$ instead of $p_{i,j}$), so combining Theorem 3.6 with Theorem 3.3, we can conclude the following.

Theorem 3.7. *If there is a size-$S(n)$, depth-d refutation of $\mathsf{PHP}_{n/2}^n$, then there is also a depth-$(d+1)$ Frege refutation of $\mathsf{Lossy}(\mathsf{Iter})_{N,L}$ with size $O(N^3L^2 + N^6) + S(N)$.*

4 Frege Lower Bounds

With the upper bound for $\mathsf{Lossy}(\mathsf{Iter})$ in hand, we are now ready to formalize the proofs of the switching lemma in proof complexity. To back up the intuition for the reductions, we will first present an overview of the argument of Urquhart and Fu [UF96], who proved a lower bound on the size of constant-depth Frege proofs of PHP_n^{n+1}. The high level idea is similar to the use of the switching lemma to prove bounded-depth circuit lower bounds. We want to show that for any (sufficiently small) bounded-depth refutation of PHP_n^{n+1}, there is a partial assignment to the variables of PHP_n^{n+1} under which the refutation simplifies to a proof in a slightly weaker system, but the formula itself remains difficult to refute. A key difference with bounded-depth circuit bounds is that formulas with many low-width clauses, like PHP, are extremely easy to simplify, as we only need to falsify a single clause to make the formula trivially false, and thus also trivial to refute. To avoid this, we only sample restrictions which will avoid falsifying any clause of PHP. However, certain formulas will never be simplified in the usual sense by such restrictions, so we also need to relax our notion of a "simplified refutation." For this we must define some basic machinery.

4.1 Matchings and Matching Trees

To be as general as possible, the following definitions will deal with a *pigeon* set P and a *hole* set H, with $|P| > |H|$. For example, to match PHP_n^{n+1} we will usually use $P = [n+1]$ and $H = [n]$.

Definition 4.1. A *matching* π over P, H is a set of pairs $(i, j) \in P \times H$ such that for each $(i, j) \neq (i', j') \in \pi$, $i \neq i'$ and $j \neq j'$. We interpret π as a list of mappings $i \mapsto j$, so that its *domain* $\mathrm{dom}(\pi)$ is the set of all $i \in P$ such that $(i, j) \in \pi$ for some $j \in H$, and its *image* $\mathrm{im}(\pi)$ is the set of all $j \in H$ such that $(i, j) \in \pi$

for some $i \in P$. We say that π *fixes* each $i \in \mathrm{dom}(\pi)$ and $j \in \mathrm{im}(\pi)$, and that the remaining pigeons and holes are left *live* by π. We say that two matchings π, σ are *compatible* if $\pi \cup \sigma$ is also a matching, and we say that σ *extends* π if $\pi \subseteq \sigma$.

We use the notation $P \restriction_\pi$ to mean $P \backslash \mathrm{dom}(\pi)$, and similarly for $H \restriction_\pi$. We can also restrict compatible restrictions π, σ by each other, with $\pi \restriction_\sigma = \pi \backslash \sigma$.

Note that each matching π over $[n+1], [n]$ corresponds to a unique partial assignment ρ to the variables of PHP_n^{n+1}, so we will abuse notation and use π to refer both to the matching and the partial assignment. This means that we can also apply π to a boolean formula F or set Γ of boolean formulas over the variables of PHP_n^{n+1}, obtaining a new formula $F \restriction_\pi$ or set $\Gamma \restriction_\pi$ in the natural way. However, rather than working directly with boolean formulas and decision trees as in the circuit lower bounds, the switching lemma will instead work with analogues which are tailored to matchings.

Definition 4.2. A *matching disjunction* over P, H is a disjunction of matchings, which we call the *terms* of the disjunction. The *width* of a disjunction D is the maximum size of its terms, and if this is at most k then D is a *matching k-disjunction*.

We can restrict a matching disjunction $D = \pi_1 \vee \cdots \vee \pi_m$ by a matching σ in the natural way, obtaining a new disjunction $D \restriction_\sigma$ which is the disjunction of $\pi_i \restriction_\sigma$ for all π_i compatible with σ. If σ extends any of $\pi_1, ..., \pi_m$, then we instead set $D \restriction_\sigma \equiv 1$, and if all terms are incompatible with σ, then $D \restriction_\sigma \equiv 0$.

Definition 4.3. A *matching tree* over P, H is a $\leq |P|$-ary tree where, rather than querying boolean variables, each internal node is labelled with a query to either a pigeon $i \in P$ or a hole $j \in H$. If a node was labelled by a pigeon i, then its outgoing edges are labelled by holes $j \in H$, representing the matching (i, j). Similarly, nodes labelled by holes have outgoing edges labelled by pigeons. We require that no labels are repeated along any branch of the tree, so that the matchings represented by the edges along the branch are compatible. We can therefore think of each branch in a matching tree as a matching, including each pair (i, j) encoded by its edges. We also require that all valid responses to a query are present at each node, so that every matching is an extension of some branch of the tree. Finally, each leaf node ℓ is labelled by a boolean value $b \in \{0, 1\}$. We call the branch π ending in ℓ a b-branch, and interpret the tree as assigning the value b to any matching extending π. For a matching tree T and $b \in \{0, 1\}$, we use $\mathrm{Br}_b(T)$ to denote the set of b-branches of T. The *depth* of T, $d(T)$, is the maximum size of its branches.

Similarly to matching disjunctions, we can also restrict a matching tree T by a matching σ, obtaining a new tree $T \restriction_\sigma$ which has all branches incompatible with σ removed, and skips any queries to pigeons or holes which are fixed σ. We say that a matching tree T *represents* a matching disjunction $D = \pi_1 \vee \cdots \vee \pi_m$ if for every $\pi \in \mathrm{Br}_1(T)$, $D \restriction_\pi \equiv 1$, and for each $\pi \in \mathrm{Br}_0(T)$, $D \restriction_\pi \equiv 0$. The *matching query complexity* of D, denoted $d^{\mathrm{match}}(D)$, is the minimum depth of any matching tree representing D.

We will often be concerned with representing a disjunction of matching trees, of the form $\bigvee_i T_i$, which we use to mean the matching disjunction $\bigvee_i \bigvee_{\pi \in \mathrm{Br}_1(T_i)} \pi$. The most obvious way to do this, which turns out to be enough for our needs, is using the canonical decision tree.

Definition 4.4. Let $D = \pi_1 \vee \cdots \vee \pi_m$ be a matching disjunction. The *canonical decision tree*, $\mathsf{CDT}(D)$, of D is the tree which does the following for each term π_k of D. For each entry $(i, j) \in \pi_k$ in some fixed order, the tree queries both the pigeon i and the hole j (skipping pigeons/holes whenever this is impossible). If the responses to all queries are compatible with π_k, then the tree outputs 1. Otherwise, we continue to π_{k+1}. Once we have exhausted all terms without satisfying any, we output 0.

It is straightforward to see that $\mathsf{CDT}(D)$ indeed represents D.

4.2 k-Evaluations

As outlined above, rather than hoping to simplify the proof in the usual sense, we will instead use the simplifications build an object (a *k-evaluation*) which still allows us to witness a contradiction.

Definition 4.5. Let Γ be a set of boolean formulas over the variables of PHP_n^{n+1} for some $n \in \mathbb{N}$, closed under sub-formulas. A *k-evaluation* supported on Γ is an assignment of each formula $F \in \Gamma$ to a matching tree $T(F)$ such that:

- $d(T(F)) \leq k$ for all $F \in \Gamma$,
- $T(\bot)$ is a single node labelled 0, and $T(\top)$ is a single node labelled 1,
- $T(p_{i,j})$ is the tree which queries pigeon i and hole j (if possible), then outputs 1 exactly when i is matched with j,
- $T(\neg F) = \neg T(F)$, where $\neg T(F)$ is obtained by negating the value of each leaf of $T(F)$, and
- $T(\bigvee_i F_i)$ represents the matching disjunction $\bigvee_i T(F_i)$.

We can think of T as encoding truth values for each formula $F \in \Gamma$, evaluated under the branches of $T(F)$. Since $T(F)$ is a matching tree, k-evaluations only define values for formulas evaluated under valid partial matchings, and it is easy to verify that every axiom of PHP is therefore a "tautology" relative to a k-evaluation.

Lemma 4.6. *Let C be an axiom of PHP_n^{n+1} and T a k-evaluation whose support includes C. Then $T(C) \equiv 1$.*

Proof. By definition, no branch of a matching tree can falsify any clause $\neg p_{i,j} \vee \neg p_{k,j}$, $k \neq i$, and each pigeon i in the support of a matching is necessarily matched to some hole, so each clause $\bigvee_{j \in [M]} p_{i,j}$ is also satisfied under every matching fixing i. Thus the only way that $T(C) \not\equiv 1$ is if $T(C)$ does not query pigeon i (or pigeon k, for the first form of clause) at all. It is easy to verify that this contradicts the definition of a k-evaluation, using the fact that the support is closed under sub-formulas and the last three bullet points above. □

The second crucial property is that for sufficiently small k, Frege proofs are "sound" relative to k-evaluations, in the following sense.

Lemma 4.7. *[UF96] Let $\mathcal{F}$ be a Frege system and let Π be a Frege proof of a formula G from formulas $F_1, ..., F_m$ over n variables. There is a constant c depending only on $\mathcal{F}$ such that for any (n/c)-evaluation whose support includes all formulas and sub-formulas of Π, if $T(F_i) \equiv 1$ for all i, then $T(G) \equiv 1$.*

In particular, if we can construct an (n/c)-evaluation for the set of formulas and sub-formulas in Π, then it cannot be a refutation of PHP (since $T(\bot)$ is always identically 0). The strategy is then to show that we can construct such a k-evaluation for any sufficiently-small constant-depth refutation, and therefore no small constant-depth proof can refute PHP_n^{n+1}.

4.3 The Lower Bound

The key tool which allows Urquhart and Fu to construct a k-evaluation for constant-depth proofs is a switching lemma, which states that for any matching k-disjunction D, there are many matchings π for which $D \restriction_\pi$ has a shallow canonical decision tree. For this we define several sets involving matchings.

Definition 4.8. Let $\ell \leq n \in \mathbb{N}$. $\mathcal{M}_n^\ell$ is the set of all matchings from $[n+1]$ to $[n]$ which leave ℓ pigeons alive.

For convenience, after applying a matching π from $\mathcal{M}_n^\ell$, we will usually reindex the remaining pigeon and holes sets $[n+1] \restriction_\pi$ and $[n] \restriction_\pi$ so that we can apply further restrictions from $\mathcal{M}_\ell^{\ell'}$ for $\ell' \leq \ell$. In other words, we will interpret each entry (i,j) in a matching $\sigma \in \mathcal{M}_\ell^{\ell'}$ as mapping the i^{th} pigeon in $[n+1] \restriction_\pi$ to the j^{th} hole in $[n] \restriction_\pi$ rather than the usual interpretation, allowing us to treat σ as a matching over $[n+1] \restriction_\pi, [n] \restriction_\pi$.

We want to show that for appropriate settings of ℓ and k, and for each matching k-disjunction D, the matchings π from $\mathcal{M}_n^\ell$ which result in a deep canonical decision tree for $D \restriction_\pi$ form a much smaller set, so we also give this set a name.

Definition 4.9. Let $n, \ell, t \in \mathbb{N}$ be such that $\ell \leq n$. For any matching disjunction D over $[n+1], [n]$. $\mathcal{BAD}_{n,\ell}^t(D)$ is the set of matchings $\pi \in \mathcal{M}_n^\ell$ such that $d(\mathsf{CDT}(D \restriction_\pi)) \geq t$.

Finally, we define a third set which we use to uniquely encode elements of $\mathcal{BAD}_{n,\ell}^t(D)$. The size of this set is much simpler to calculate, giving us a handle on the size of $\mathcal{BAD}_{n,\ell}^t(D)$.

Definition 4.10. Let $n, \ell, k, t \in \mathbb{N}$ be such that $\ell \leq n$. $Code_{k,t}$ is the set of all sequences $\mathcal{C}_1, ..., \mathcal{C}_m$, where each $\mathcal{C}_i \in \{0,1\}^k \backslash 0^k$ and the total number of 1's in the sequence is t. $\mathcal{ENC}^{k,t}_{n,\ell}$ is then the set:

$$\bigcup_{t/2 \leq j \leq t} \mathcal{M}_n^{\ell - j} \times Code_{k,t} \times [2\ell + 1]^t$$

As promised, we can uniquely encode elements of $\mathcal{BAD}^t_{n,\ell}(D)$ by elements of this set.

Theorem 4.11. *[UF96] Let $n, \ell, k, t \in \mathbb{N}$ be such that $\ell \leq n$, and let D be a matching k-disjunction over $[n+1], [n]$. There is an injective mapping from the set $\mathcal{BAD}^t_{n,\ell}(D)$ to $\mathcal{ENC}^{k,t}_{m,\ell}$.*

Comparing the size of this set to that of $\mathcal{M}^\ell_n$ concludes the switching lemma.

Theorem 4.12. *[UF96] Let $n, \ell, k, t \in \mathbb{N}$ with $\ell \geq 10$, $k \leq \ell$, and $\ell^4/n \leq 1/10$. Then:*

$$\frac{\left|\mathcal{ENC}^{k,2t}_{n,\ell}\right|}{|\mathcal{M}^\ell_n|} \leq (11k\ell^4/n)^t$$

Therefore, for any matching k-disjunction D over $[n+1], [n]$, we also have:

$$\frac{\left|\mathcal{BAD}^{2t}_{n,\ell}(D)\right|}{|\mathcal{M}^\ell_n|} \leq (11k\ell^4/n)^t$$

We are finally ready to give a sketch of the full lower bound argument. We want to show the following.

Theorem 4.13. *Let $d > 0 \in \mathbb{N}$ and $0 < \delta < (1/5)^d$. Then for every Frege system $\mathcal{F}$ and for sufficiently large n, any depth-d $\mathcal{F}$-refutation of PHP^{n+1}_n requires size at least 2^{n^δ}.*

Proof Sketch. Suppose for contradiction that there is a depth-d refutation Π of size smaller than 2^{n^δ}. Our goal is to find a matching ρ for which:

- $\Pi \restriction_\rho$ is still a refutation of PHP^{m+1}_m, for some $m = \mathsf{poly}(n)$, but
- There is a $2n^\delta$-evaluation for the set of formulas and sub-formulas appearing in $\Pi \restriction_\rho$

which contradicts Lemmas 4.6 and 4.7.

Let Γ be the set of all formulas and sub-formulas appearing in Π. We will iteratively choose matchings $\pi_0, ..., \pi_d$ which allow us to construct a sequence of k-evaluations for $\Gamma_{\mathsf{lit}}, \Gamma_0 \restriction_{\pi_0}, ..., \Gamma_d \restriction_{\pi_0 \cdots \pi_d} = \Gamma \restriction_{\pi_0 \cdots \pi_d}$, where Γ_{lit} is the set of all literals in Γ and for $i = 0, ..., d$, Γ_i is the subset of Γ containing all formulas of depth at most i.

For Γ_{lit} the $2n^\delta$-evaluation is trivial, as the definition of a k-evaluation already determines depth-2 trees for literals. For the remaining subsets, we appeal to the switching lemma to simplify the evaluation we have constructed so far. Each matching π_i will leave $\ell_i = n^{\varepsilon^{i+1}}$ pigeons live, where ε is chosen so that $\delta < \varepsilon^d < (1/5)^d$, and as previously discussed we reindex after each matching so that we can think of π_i as being from the set $\mathcal{M}^{\ell_i}_{\ell_{i-1}}$. Now for $i = 0, ..., d$ in order, let $\rho = \pi_0 \cdots \pi_{i-1}$ be the composition of the matchings chosen so far. Given a $2n^\delta$-evaluation T_{i-1} for $\Gamma_{i-1} \restriction_\rho$, we extend it to depth-$i$ formulas as follows. For any $F \in \Gamma_i$ of the form $\bigvee_j F_j$, consider the $2n^\delta$-disjunction:

$$D_F = \bigvee_j T_{i-1}(F_j) \restriction_\rho$$

In general, $d^{\text{match}}(D_F)$ might be very large, even after restricting by ρ, so we would like to apply another matching π_i from $\mathcal{M}^{\ell_i}_{\ell_{i-1}}$ to further simplify D_F. If we set $k = \lfloor 2n^\delta \rfloor$ and $t = n^\delta$, then by Theorem 4.12 we have:

$$\frac{\left|\mathcal{BAD}^{2t}_{\ell_{i-1},\ell_i}(D_F)\right|}{\left|\mathcal{M}^{\ell_i}_{\ell_{i-1}}\right|} \leq (11k{\ell_i}^4/\ell_{i-1})^t$$

By our settings of ℓ_i and ℓ_{i-1}, this is at most $(22N^\delta/n^{\varepsilon^i/5})^{n^\delta}$ for sufficiently large n, which is in turn at most 2^{-n^δ} since $\delta < \varepsilon^d < \varepsilon^i/5$. Therefore, since $|\Gamma_i| < 2^{n^\delta}$, there is a choice of $\pi_i \in \mathcal{M}^{\ell_i}_{\ell_{i-1}}$ such that $d(\mathsf{CDT}(D_F)\restriction_{\pi_i}) \leq 2n^\delta$ for all depth-i formulas $F \in \Gamma_i$ of the form $\bigvee_i F_i$. Assuming that for any matching tree T representing a formula F, $T\restriction_{\pi_i}$ still represents $F\restriction_{\pi_i}$, we can use this to safely extend T to a new evaluation T' for $\Gamma_i\restriction_{\rho\pi_i}$, assigning:

- $T'(F\restriction_{\pi_i}) = T(F)\restriction_{\pi_i}$ for $F \in \Gamma_{i-1}\restriction_\rho$ (or Γ_{lit} if $i = 0$)
- $T'(F\restriction_{\pi_i}) = \mathsf{CDT}(D_F)\restriction_{\pi_i}$ for depth-i formulas $F \in \Gamma_i\restriction$ of the form $\bigvee_i F_i$, and
- $T'(F) = \neg T'(F_0)$ for formulas $F \in \Gamma_i\restriction_{\rho\pi_i}$ of the form $F = \neg F_0$.

After repeating this process for all i, we are left with a $2n^\delta$-evaluation for $\Gamma\restriction_{\pi_0\cdots\pi_d}$ as promised. Assuming $\Pi\restriction_{\pi_0\cdots\pi_d}$ is still an $\mathcal{F}$-refutation of the polynomially-smaller instance $\mathsf{PHP}^{n+1}_n\restriction_{\pi_0\cdots\pi_d} = \mathsf{PHP}^{\ell_d}_{\ell_d-1}$ of the pigeonhole principle, this contradicts Lemmas 4.6 and 4.7, so Π could not have been a valid $\mathcal{F}$ refutation. □

Note that we have omitted the proofs of some technical assumptions used in this sketch. These details and the full argument are available in [UF96].

Observe that rather than showing there is a good choice of π_i at each iteration i by counting "bad" choices of π_i, we could have shown a similar bound by instead counting entire sequences — for each iteration i and any formula $F \in \Gamma_i$, the fraction of sequences $\pi_0, ..., \pi_d$ for which $\pi_i \in \mathcal{BAD}^{2n^\delta}_{\ell_{i-1},\ell_i}(D_F)$ is at most 2^{-n^δ}. Since there are $d+1$ iterations, unless $|\Gamma| \geq 2^{n^\delta}/(d+1)$, there must be a sequence of matchings which works. We will do this counting more carefully in the proof of our reduction, which relies on this observation.

5 The Reduction

In this section we finally conclude our main upper bounds by giving a reduction from $\mathsf{Prf}^d_{1.01^{n^\delta},n^c}(\mathsf{PHP}^{n+1}_n)$ to $\mathsf{Lossy}(\mathsf{Iter})$, for which we will closely follow the lower bound argument from the previous section. The full version of that argument involves iteratively applying restrictions, which can be difficult to follow without understanding the main idea, so for clarity we will prove two separate reductions — to present the main idea behind our reduction, we will first give a warm-up reduction which only uses one iteration of the lower bound argument, but which only works for $\mathsf{Prf}^{0.5}_{1.01^{n^\delta},n^c}(\mathsf{PHP}^{n+1}_n)$. Then we will give the full reduction, showing how to handle the iteration. Both reductions are inspired by [LLR24, Theorem 3.16].

5.1 Warm-up: Depth-0.5 Refuter Principles

We begin with the following simpler reduction.

Theorem 5.1. *For any $0 < \delta < (1/5)^2$ and $c \in \mathbb{N}$, the false-clause search problem* $\mathrm{Search}(\mathsf{Prf}^{0.5}_{1.01^{n^\delta},n^c}(\mathsf{PHP}^{n+1}_n))$ *is reducible to* $\mathrm{Search}(\mathsf{Lossy}(\mathsf{Iter})_{N,L})$ *in decision-tree depth $O(n^c)$, for some $N = 2^{O(n^\delta)}$ and $L = 1.01^{n^\delta}$.*

Note that n^c is polylogarithmic in the size of the refuter formula, which is $\Omega\left(1.01^{n^\delta}\right)$, so this reduction is efficient enough for our needs. The reduction will use the observation that if the width of all formulas was already $\mathsf{polylog}(n)$ in the lower bound argument above, then even before applying any restrictions we would already have $d(\mathsf{CDT}(F)) = O(\log n)$ for all depth-0 conjunctions appearing in the proof, so we could skip the first iteration. For depth-0.5 Frege, this means that only one iteration is necessary.

Proof. Let $\Pi = F_1, ..., F_L$ denote the purported depth-0.5 refutation encoded by the variables of the input refuter instance, and Γ the set of all formulas and sub-formulas appearing in it. To avoid obfuscating the bigger picture, we will talk about this refutation directly rather than referring to individual variables of the refuter principle, and it should be clear that the information we need can be queried in depth $O(n^c)$.

Choose ε as before, so that $\delta < \varepsilon^2 < (1/5)^2$. We identify the domain $[N]$ of the $\mathsf{Lossy}(\mathsf{Iter})$ instance with $\mathcal{M}^{\lceil n^\varepsilon\rceil}_n$, which has size at most $2^{O(n^\delta)}$. For each Iter instance associated with some matching $\pi \in \mathcal{M}^{\lceil n^\varepsilon\rceil}_n$, we identify each node $i \in [L]$ with the $(L-i+1)^{th}$ line of the restricted proof $\Pi\restriction_\pi$, so that $F_L\restriction_\pi = \bot$ is always

associated with the node (1), $F_{L-1} \restriction_\pi$ is associated with (2), and so on. The range $[N/2]$ is identified with $[L] \times \mathcal{ENC}^{\lfloor 2n^\delta \rfloor, \lfloor 2n^\delta \rfloor}_{n, \lceil n^\varepsilon \rceil}$, representing the encodings of matchings which do not simplify the canonical decision tree of some line F of Π. We include the index of this line so that we can invert the encoding. By Theorem 4.12, this is possible, as

$$\frac{\left|[L] \times \mathcal{ENC}^{\lfloor 2n^\delta \rfloor, \lfloor 2n^\delta \rfloor}_{n, \lceil n^\varepsilon \rceil}\right|}{\left|\mathcal{M}_n^{\lceil n^\varepsilon \rceil}\right|} \leq 1.01^{n^\delta} \cdot (22n^\delta / n^{\varepsilon/5})^{n^\delta} \leq \frac{1}{2}$$

for sufficiently large n. Each extra element $j \in [N/2]$ can simply be ignored by setting the decoding pointer $g(j)$ arbitrarily and preventing any $h_i(u)$ from pointing to j to avoid them giving rise to unwanted solutions. See Figure 5.1 for a diagram of this setup.

Now we define the behaviour of the Iter instances associated with each matching $\pi \in \mathcal{M}_n^{\lceil n^\varepsilon \rceil}$. Assume for now that π is a "good" matching, in the sense that for each $F = \bigvee_i F_i \in \Gamma$, $d(\mathsf{CDT}(D_F) \restriction_\pi) \leq \lfloor 2n^\delta \rfloor$, so we can follow the proof of the lower bound to find a $2n^\delta$-evaluation T for $\Gamma \restriction_\pi$. We will use T to locate the invalid derivation in Π. By definition of a k-evaluation we have $T(F_L) = T(\bot) \equiv 0$, and by Lemma 4.7 we know that for some hypothesis G of the rule application used to derive F_L, $T(G \restriction_\pi) \not\equiv 1$, so we set $s_\pi(F_L) = G$ (if there are multiple we can choose arbitrarily). We continue this for all lines F of the proof, setting $s_\pi(F) = F$ if $T(F \restriction_\pi) \equiv 1$ (so that no such line is ever the successor of any other line), and $s_\pi(F) = G$ if $T(F \restriction_\pi) \not\equiv 1$, where G is any hypothesis of the rule application used to derive F such that $T(G \restriction_\pi) \not\equiv 1$ (guaranteed by Lemma 4.7). If we ever fail to locate such a G (including if F was a purported axiom), then we intentionally cause a solution by setting $s_\pi(F) = F$. Now unless some node points to a preceding node (in which case we have already found an error in the proof), the only solution that could arise is if some node F was the successor of some other node, ie. $T(F \restriction_\pi) \not\equiv 1$, but either F was a purported axiom of PHP_n^{n+1} or $T(G \restriction_\pi) \equiv 1$ for each hypothesis G of the rule application used to derive F. The first option contradicts Lemma 4.6, and the second contradicts Lemma 4.7, so in either case we have located an invalid derivation and we can set the corresponding pointer $h_\pi(F)$ arbitrarily to intentionally cause a solution. Since we only ever need to query a single line of the proof and its (constantly-many) hypotheses, this can all be done in depth $O(n^c)$.

Of course, there are many matchings π which are not "good", and for these we have no way to construct

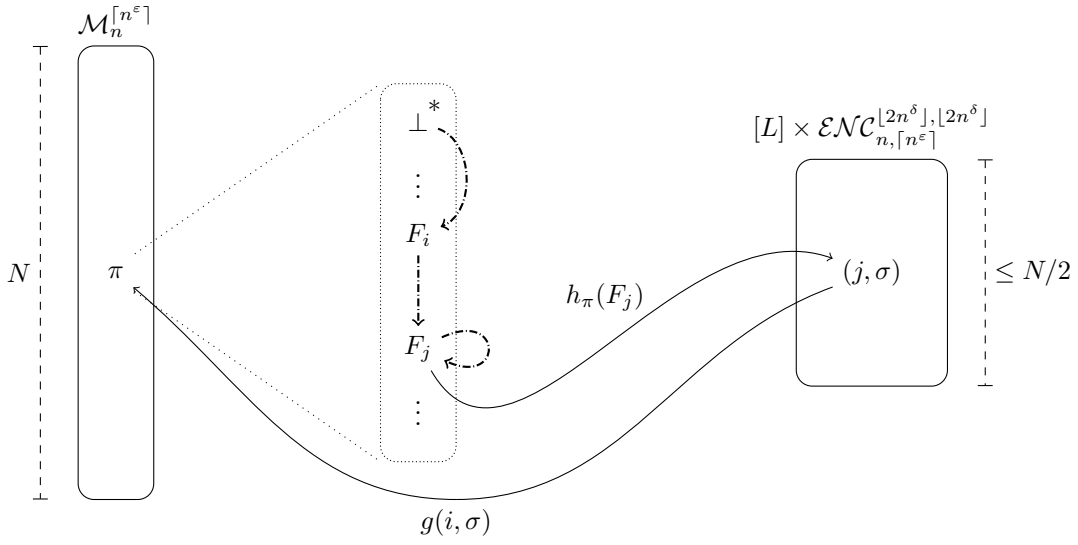

Figure 5.1: An example with a "bad" matching π. Here F_i is the first hypothesis of $F_L = \bot$ for which $T(F_i \restriction_\pi) \not\equiv 1$, F_j is the first hypothesis of F_i for which $d(\mathsf{CDT}(D_{F_j}) \restriction_\pi) > 2n^\delta$, and $\sigma = \mathsf{enc}_{F_j}(\pi)$. The Iter instance for π is shown in the dotted box, with s_π shown as dashed arrows. The relevant coordinates of h_π and g are shown by the labelled solid arrows. Since $g(h_\pi(F_j)) = \pi$, there is no solution in this picture.

a $2n^\delta$-evaluation. Intuitively, to handle these we instead take the largest subset $\Gamma' \subset \Gamma$, closed under sub-formulas, for which $d(\mathsf{CDT}(D_F)\restriction_\pi) \leq 2n^\delta$ for all $F = \bigvee_i F_i \in \Gamma'$, then construct a k-evaluation T supported on Γ' as above. What this changes in the reduction itself is that we now must check at each line $F_i = \bigvee_j C_j$ whether $d(\mathsf{CDT}(D_{F_i}\restriction_\pi)) \leq 2n^\delta$ (ie. whether $F_i \in \Gamma'$). If not, then we intentionally create a solution in the Iter instance by setting $s_\pi(F_i) = F_i$, which we can prevent from causing a solution to $\mathsf{Lossy}(\mathsf{Iter})$ by setting $h_i(F_i) = \left(i, \mathsf{enc}_{D_{F_i}}(\pi)\right)$, where for a matching k-disjunction D, $\mathsf{enc}_D : \mathcal{BAD}^{\lfloor 2n^\delta \rfloor}_{n,\lceil n^\varepsilon \rceil}(D) \to \mathcal{ENC}^{\lfloor 2n^\delta \rfloor, \lfloor 2n^\delta \rfloor}_{n,\lceil n^\varepsilon \rceil}$ is the mapping promised by Theorem 4.11. Since this mapping is injective, we can set $g(i,\sigma) = \left(\mathsf{enc}_{D_{F_i}}\right)^{-1}(\sigma)$ for all pairs $(i,\sigma) \in [L] \times \mathcal{ENC}^{\lfloor 2n^\delta \rfloor, \lfloor 2n^\delta \rfloor}_{P,H,\lceil n^\varepsilon \rceil}$ for which this is well-defined. $g(i,\sigma)$ can be set arbitrarily for the remaining pairs. If for any hypothesis G of F, $d(\mathsf{CDT}(G\restriction_\pi)) > 2n^\delta$, we also cannot proceed as before (since we had to check $T(G\restriction_\pi)$ for all hypotheses G), so we set $s_\pi(F) = G$ to cause the Iter solution to occur at G instead. Otherwise, if F and all its hypotheses are in the support of T, we can proceed as above. After doing this, the only remaining solutions to the $\mathsf{Lossy}(\mathsf{Iter})$ instance are the ones above, corresponding to invalid derivations in Π as desired (we will verify this more carefully in the proof of the full reduction). Furthermore, since we already queried the line F_i above to compute $s_\pi(F_i)$ and $h_\pi(F_i)$, we can check the depth of $\mathsf{CDT}(D_F\restriction_\pi)$ without any additional queries.

□

5.2 General Refuter Principles

We are finally ready to iterate this. Our goal is now the following:

Theorem 5.2. *For any $d \in \mathbb{N}, 0 < \delta < (1/5)^d$, and $c \in \mathbb{N}$,* $\mathrm{Search}(\mathsf{Prf}^d_{1.01^{n^\delta},n^c}(\mathsf{PHP}^{n+1}_n))$ *is reducible to* $\mathrm{Search}(\mathsf{Lossy}(\mathsf{Iter})_{N,L})$ *in decision-tree depth $O(n^c)$, for some $N = 2^{O(n^\delta)}$ and $L = 1.01^{n^\delta}$.*

We will use the observation mentioned at the end of the lower bound sketch, that the number of sequences $\pi_0, ..., \pi_d$ for which we cannot construct a $2n^\delta$-evaluation for the restricted proof is still a small fraction of all sequences. The way we handle this is similar to the argument seen in [LLR24].

Proof. The main idea of the reduction is the same as before, except that now we have to be a bit more careful about how we encode the bad restrictions. Let $\Pi = F_1, ..., F_L$ again denote the purported depth-d

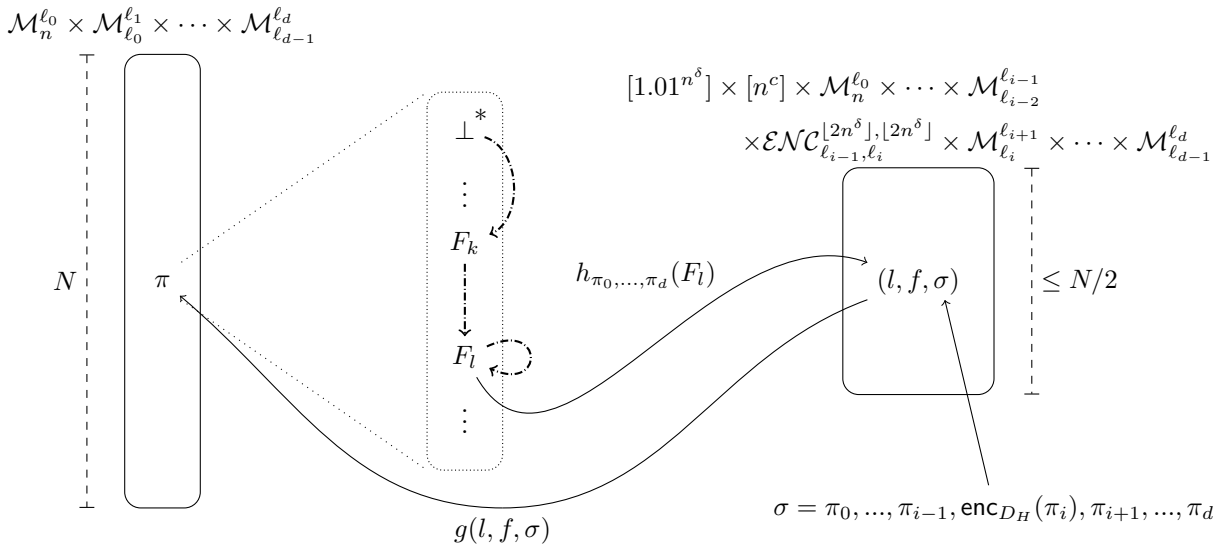

Figure 5.2: An example with a "bad" sequence $\pi_0, ..., \pi_d$, demonstrating that it does not cause an undesired solution. The situation depicted is the same as in Figure 5.1, except that F_l is instead the first hypothesis of F_k for which any iteration i fails, H is the first depth-i sub-formula of F_l for which $d(T_i(H\restriction_{\pi_0 \cdots \pi_i})) > 2n^\delta$, and f is its index within F_l.

refutation, Γ the set of all formulas and sub-formulas appearing in it, and choose ε so that $\delta < \varepsilon^d < (1/5)^d$. The domain $[N]$ of the $\mathsf{Lossy}(\mathsf{Iter})$ instance is identified with $\mathcal{M}_n^{\ell_0} \times \mathcal{M}_{\ell_0}^{\ell_1} \times \cdots \times \mathcal{M}_{\ell_{d-1}}^{\ell_d}$, the set of all sequences of matchings we could choose in the lower bound argument, where for each $i = 0, ..., d$, $\ell_i = \lceil n^{\varepsilon^{i+1}} \rceil$. The size of this set is still at most $2^{O(n^\delta)}$. The encodings are more tricky, since we will only encode one matching from such a sequence at a time. Therefore we identify $[N/2]$ with the set:

$$\bigcup_{0 \le i \le d} \left([1.01^{n^\delta}] \times [n^c] \times \mathcal{M}_n^{\ell_0} \times \cdots \times \mathcal{M}_{\ell_{i-2}}^{\ell_{i-1}} \times \mathcal{ENC}_{\ell_{i-1},\ell_i}^{\lfloor 2n^\delta \rfloor, \lfloor 2n^\delta \rfloor} \times \mathcal{M}_{\ell_i}^{\ell_{i+1}} \times \cdots \times \mathcal{M}_{\ell_{d-1}}^{\ell_d} \right)$$

In other words, "bad" sequences will map to a new sequence containing all matchings both before and after the problematic matching, as well as the encoding of the bad matching and the index of the sub-formula which it failed to simplify. We put off the proof that this set has size at most $N/2$ for now.

Concretely, for each sequence $\pi_0, ..., \pi_d$ and each line F of Π, we will attempt to construct a sequence of $2n^\delta$-evaluations $T_{lit}, T_0, ..., T_d$ and a matching $2n^\delta$-disjunction D_G for each sub-formula G of F as follows. T_{lit} is the trivial $2n^\delta$-evaluation for the set of literals appearing in F, and for depth-0 disjunctions of the form $G = \bigvee_i \ell_i$, we define $D_G = \bigvee_i T_{lit}(\ell_i)$ as before. We define $T_0(G \restriction_{\pi_0}) = \mathsf{CDT}(D_G) \restriction_{\pi_0}$ for depth-0 disjunctions, $T_0(\neg G \restriction_{\pi_0}) = \neg T_0(G)$ for depth-0 negations, and $T_0(\ell \restriction_{\pi_0}) = T_{lit}(\ell) \restriction_{\pi_0}$ for literals ℓ. Iterating this for $i > 0$, we define:

$$D_F = \bigvee_j T_{i-1}(F_j \restriction_{\pi_0 \cdots \pi_{i-1}})$$

for depth-i disjunctions $F = \bigvee_j F_j$, and:

$$T_i(G \restriction_{\pi_0 \cdots \pi_i}) = \begin{cases} T_{i-1}(G \restriction_{\pi_0 \cdots \pi_{i-1}}) \restriction_{\pi_i} & \text{if } G \text{ has depth at most } i-1 \\ \mathsf{CDT}(D_G) \restriction_{\pi_i} & \text{if } G \text{ is a depth-}i \text{ disjunction} \\ \neg T_i(G_0) & \text{if } G = \neg G_0 \text{ is a depth-}i \text{ negation} \end{cases}$$

for all sub-formulas G of F with depth at most i. We say that iteration i *succeeds* if $d(T_i(G \restriction_{\pi_0 \cdots \pi_i})) \le 2n^\delta$ for all depth-i sub-formulas G of F, and fails otherwise.

If all iterations succeed for some line F of Π and all its hypotheses, then we define its successor in the Iter instance corresponding to $\pi_0, ..., \pi_d$ as in the depth-0.5 case — if $T_d(F \restriction_{\pi_0 \cdots \pi_d}) \not\equiv 1$, then $s_{\pi_0,...,\pi_d}(F) = G$ for some hypothesis G of F with $T_d(G \restriction_{\pi_0 \cdots \pi_d}) \not\equiv 1$, and if $T_d(F \restriction_{\pi_0 \cdots \pi_d}) \equiv 1$ or no such G was found, $s_{\pi_0,...,\pi_d}(F) = F$. Otherwise, let i be the first iteration which fails for F or its hypotheses, and let G be the line of the proof for which it fails, meaning that there was a depth-i sub-formula $H = \bigvee_j H_j$ of G for which $d(\mathsf{CDT}(D_H) \restriction_{\pi_i}) > 2n^\delta$. If $G \neq F$, then set $s_{\pi_0,...,\pi_d}(F) = G$. Otherwise, we set $s_{\pi_0,...,\pi_d}(F) = F$ to intentionally create a solution in the Iter instance, then set $h_{\pi_0,...,\pi_d}(F) = (l, f, \pi_0, ..., \pi_{i-1}, \mathsf{enc}_{D_H}(\pi_i), \pi_{i+1}, ..., \pi_d)$, where l is the index of the line G in Π, f is the index of the sub-formula H within G, and enc_{D_H} is again the injective mapping promised by Theorem 4.11. To prevent this from causing a solution, we also set $g(l, f, \pi_0, ..., \pi_{i-1}, \sigma, \pi_{i+1}, ..., \pi_d) = (l, f, \pi_0, ..., \pi_{i-1}, \mathsf{enc}_{D_H}^{-1}(\sigma), \pi_{i+1}, ..., \pi_d)$ for all elements of the range for which this is well-defined (the rest can be set arbitrarily).

To verify that the only solutions which can occur are the ones discussed above, recall that solutions to this instance of $\mathsf{Lossy}(\mathsf{Iter})$ have the form $(\pi_0, ..., \pi_d, F) \in \mathcal{M}_n^{\ell_0} \times \mathcal{M}_{\ell_0}^{\ell_1} \times \cdots \times \mathcal{M}_{\ell_{d-1}}^{\ell_d} \times L$, where either:

- $F = \bot$, $s_{\pi_0,...,\pi_d}(F) = F$, and $g(h_{\pi_0,...,\pi_d}(i)) \neq i$,
- $s_{\pi_0,...,\pi_d}(F) < F$ and $g(h_{\pi_0,...,\pi_d}(F)) \neq F$, or
- $s_{\pi_0,...,\pi_d}(F) = G$, $s_{\pi_0,...,\pi_d}(G) = G$, and $g(h_{\pi_0,...,\pi_d}(G)) \neq G$

Clearly, the first two cases cannot happen, since we only choose earlier lines of Π as successors. For the third case, we have further cases:

- If all iterations succeeded for F and its hypotheses, then this solution could only occur if $T_d(F) \not\equiv 1$ and $T_d(G) \not\equiv 1$ for some hypothesis G of F, but $T_d(H) \equiv 1$ for all hypotheses H of G. Then by Lemma 4.7, the rule application used to derive G was invalid and the error can be located by a depth-$O(n^c)$ decision tree.

- If all iterations succeeeded for F but some iteration i failed for one of its hypotheses G, we would set $h_{\pi_0,...,\pi_d}(G) = (l, f, \pi_0, ..., \pi_{i-1}, \mathsf{enc}_{D_H}(\pi_i), \pi_{i+1}, ..., \pi_d)$, where l is the index of G, H is the subformula of G on which iteration i failed, and f is its index within G. But then by our definition of g, $g(h_{\pi_0,...,\pi_d}(G)) = G$, so this solution cannot occur.
- If any iteration failed for F itself, $T_d(F) \equiv 1$, or $T_d(G) \equiv 1$ for all hypotheses G of F, then $s_{\pi_0,...,\pi_d}(F) = F$ and this solution cannot occur.

Finally, we meed to verify that it is actually possible to identify $[N/2]$ with the range we are using. To see this, fix some depth $i \leq d$ and some sequence $\pi_0, ..., \pi_{i-1}, \pi_{i+1}, ..., \pi_d$ from the set:

$$\mathcal{M}_n^{\ell_0} \times \cdots \times \mathcal{M}_{\ell_{i-2}}^{\ell_{i-1}} \times \mathcal{M}_{\ell_i}^{\ell_{i+1}} \times \cdots \times \mathcal{M}_{\ell_{d-1}}^{\ell_d}$$

ie. we are fixing some sequence of matchings from the domain, but omitting the matching for depth i. Each such sequence can be extended to either $\left|\mathcal{M}_{\ell_{i-1}}^{\ell_i}\right|$ distinct elements of the domain, or $n^c \cdot 1.01^{n^\delta} \cdot \left|\mathcal{ENC}_{\ell_{i-1},\ell_i}^{\lfloor 2n^\delta \rfloor, \lfloor 2n^\delta \rfloor}\right|$ distinct elements of the range, all of which are unique to this choice of $\pi_0, ...\pi_{i-1}, \pi_{i+1}, ..., \pi_d$. By Theorem 4.12:

$$\frac{\left|\mathcal{ENC}_{\ell_{i-1},\ell_i}^{\lfloor 2n^\delta \rfloor, \lfloor 2n^\delta \rfloor}\right|}{\left|\mathcal{M}_{\ell_{i-1}}^{\ell_i}\right|} \leq (22n^\delta / n^{\varepsilon^i/5})^{n^\delta} \leq 2^{-n^\delta}$$

for sufficiently large n. Therefore for any depth $0 \leq i \leq d$, the size of the set:

$$[1.01^{n^\delta}] \times [n^c] \times \mathcal{M}_n^{\ell_0} \times \cdots \times \mathcal{M}_{\ell_{i-2}}^{\ell_{i-1}} \times \mathcal{ENC}_{\ell_{i-1},\ell_i}^{\lfloor 2n^\delta \rfloor, \lfloor 2n^\delta \rfloor} \times \mathcal{M}_{\ell_i}^{\ell_{i+1}} \times \cdots \times \mathcal{M}_{\ell_{d-1}}^{\ell_d}$$

is at most $1.01^{n^\delta} \cdot n^c \cdot 2^{-n^\delta} \cdot N = o(N)$. Since the range is the union of $d+1 = O(1)$ such sets, for sufficiently large n the range has size at most $N/2$ as desired. □

The main upper bounds follow — Theorem 1.3 by combining this with Theorems 2.7 and 3.1, and Theorem 1.4 by combining this with Theorems 2.8 and 3.7.

References

[AAdRK25] Noel Arteche, Albert Atserias, Susanna F. de Rezende, and Erfan Khaniki. The proof analysis problem. *arXiv*, 2025.

[ABRW02] Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Space complexity in propositional calculus. *SIAM J. Comput.*, 31(4):1184–1211, 2002.

[Ajt94] Miklós Ajtai. The complexity of the pigeonhole principle. *Comb.*, 14(4):417–433, 1994.

[AM20] Albert Atserias and Moritz Müller. Automating resolution is np-hard. *J. ACM*, 67(5):31:1–31:17, 2020.

[AR08] Michael Alekhnovich and Alexander A. Razborov. Resolution is not automatizable unless W[P] is tractable. *SIAM J. Comput.*, 38(4):1347–1363, 2008.

[AW09] Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *ACM Trans. Comput. Theory*, 1(1):2:1–2:54, 2009.

[BGS75] Theodore P. Baker, John Gill, and Robert Solovay. Relativizations of the P =? NP question. *SIAM J. Comput.*, 4(4):431–442, 1975.

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

[BPR00] Maria Luisa Bonet, Toniann Pitassi, and Ran Raz. On interpolation and automatization for frege systems. *SIAM J. Comput.*, 29(6):1939–1967, 2000.

[CR79] Stephen Cook and Robert Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44(1):36–50, 1979.

[dRGN+21] Susanna F. de Rezende, Mika Göös, Jakob Nordström, Toniann Pitassi, Robert Robere, and Dmitry Sokolov. Automating algebraic proof systems is NP-Hard. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 209–222. ACM, 2021.

[Gar24] Michal Garlík. Failure of feasible disjunction property for k-dnf resolution and np-hardness of automating it. In *39th Computational Complexity Conference, CCC 2024, July 22-25, 2024, Ann Arbor, MI, USA*, volume 300 of *LIPIcs*, pages 33:1–33:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[GKMP20] Mika Göös, Sajin Koroth, Ian Mertz, and Toniann Pitassi. Automating cutting planes is NP-Hard. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 68–77. ACM, 2020.

[Hak85] Armin Haken. The intractability of resolution. *Theor. Comput. Sci.*, 39:297–308, 1985.

[Hås86] Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 6–20. ACM, 1986.

[Hås23] Johan Håstad. On small-depth frege proofs for PHP. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 37–49. IEEE, 2023.

[Iwa97] Kazuo Iwama. Complexity of finding short resolution proofs. In *Mathematical Foundations of Computer Science 1997, 22nd International Symposium, MFCS'97, Bratislava, Slovakia, August 25-29, 1997, Proceedings*, volume 1295 of *Lecture Notes in Computer Science*, pages 309–318. Springer, 1997.

[Jeř07] Emil Jeřábek. On independence of variants of the weak pigeonhole principle. *Journal of Logic and Computation*, 17(3):587–604, 2007.

[Kor21] Oliver Korten. The hardest explicit construction. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 433–444. IEEE, 2021.

[Kor22] Oliver Korten. Derandomization from time-space tradeoffs. In *37th Computational Complexity Conference, CCC 2022, Philadelphia, PA, USA, July 20-23, 2022*, volume 234 of *LIPIcs*, pages 37:1–37:26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[KP98] Jan Krajícek and Pavel Pudlák. Some consequences of cryptographical conjectures for s^1_2 and EF. *Inf. Comput.*, 140(1):82–94, 1998.

[Kra01] Jan Krajíček. On the weak pigeonhole principle. *Fundamenta Mathematicae*, 170(1-3):123–140, 2001.

[Kra04] Jan Krajíček. Dual weak pigeonhole principle, pseudo-surjective functions, and provability of circuit lower bounds. *Journal of Symbolic Logic*, 69(1):265–286, 2004.

[Kra19] Jan Krajíček. *Proof Complexity*. Cambridge University Press, 2019.

[LLR24] Jiawei Li, Yuhao Li, and Hanlin Ren. Metamathematics of resolution lower bounds: A TFNP perspective. *CoRR*, abs/2411.15515, 2024.

[LPT24] Jiatu Li, Edward Pyne, and Roei Tell. Distinguishing, predicting, and certifying: On the long reach of partial notions of pseudorandomness. In *65th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2024, Chicago, IL, USA, October 27-30, 2024*, pages 1–13. IEEE, 2024.

[MPW00] Alexis Maciel, Toniann Pitassi, and Alan R. Woods. A new proof of the weak pigeonhole principle. *Proceedings of the thirty-second annual ACM symposium on Theory of computing - STOC 00*, 2000.

[Pap24] Theodoros Papamakarios. Depth-d frege systems are not automatable unless P = NP. In *39th Computational Complexity Conference, CCC 2024, July 22-25, 2024, Ann Arbor, MI, USA*, volume 300 of *LIPIcs*, pages 22:1–22:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[Pud03] Pavel Pudlák. On reducibility and symmetry of disjoint NP pairs. *Theor. Comput. Sci.*, 295:323–339, 2003.

[Pud20] Pavel Pudl'ak. Reflection principles, propositional proof systems, and theories. *arXiv: Logic*, 2020.

[Raz95] Alexander A. Razborov. Bounded arithmetic and lower bounds in boolean complexity. In *Feasible Mathematics II*, pages 344–386, Boston, MA, 1995. Birkhäuser Boston.

[Raz98] Alexander A. Razborov. Lower bounds for the polynomial calculus. *Comput. Complex.*, 7(4):291–324, 1998.

[Raz15] Alexander A. Razborov. Pseudorandom generators hard for k-dnf resolution and polynomial calculus resolution. *Annals of Mathematics*, 181:415–472, 2015.

[Sho67] Joseph R. Shoenfield. *Mathematical Logic*. Addison-Wesley Series in Logic. Addison-Wesley, 1967.

[ST21] Rahul Santhanam and Iddo Tzameret. Iterated lower bound formulas: a diagonalization-based approach to proof complexity. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 234–247. ACM, 2021.

[TX13] Luca Trevisan and Tongke Xue. A derandomized switching lemma and an improved derandomization of AC0. In *Proceedings of the 28th Conference on Computational Complexity, CCC 2013, K.lo Alto, California, USA, 5-7 June, 2013*, pages 242–247. IEEE Computer Society, 2013.

[UF96] Alasdair Urquhart and Xudong Fu. Simplified lower bounds for propositional proofs. *Notre Dame J. Formal Log.*, 37(4):523–544, 1996.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il