The Proof Analysis Problem

Noel Arteche
Lund University &
University of Copenhagen
noel.arteche@cs.lth.se

Albert Atserias
Universitat Politècnica de Catalunya
& Centre de Recerca Matemàtica
atserias@cs.upc.edu

Susanna F. de Rezende
Lund University
susanna.rezende@cs.lth.se

Erfan Khaniki
University of Oxford
erfan.khaniki@cs.ox.ac.uk

Abstract*—Atserias and Müller (*JACM*, 2020) proved that for every unsatisfiable CNF formula φ, the formula $\textsc{Ref}(\varphi)$—stating that "φ has small Resolution refutations"—does not have subexponential-size Resolution refutations. Conversely, when φ is satisfiable, Pudlák (*TCS*, 2003) showed how to construct a polynomial-size Resolution refutation of $\textsc{Ref}(\varphi)$ given a satisfying assignment of φ. A question that had remained open is: *do all short Resolution refutations of* $\textsc{Ref}(\varphi)$ *explicitly leak a satisfying assignment of* φ*?

We answer this question affirmatively by providing a polynomial-time algorithm that extracts a satisfying assignment for φ given any short Resolution refutation of $\textsc{Ref}(\varphi)$. The algorithm follows from a new feasibly constructive proof of the Atserias–Müller lower bound, formalizable in Cook's theory PV_1 of bounded arithmetic. This implies that Extended Frege can efficiently prove (a suitable formalization of the statement) that automating Resolution is $\mathbf{NP}$-hard.

Motivated by this algorithm, we introduce a new meta-computational problem concerning Resolution lower bounds: the *Proof Analysis Problem* (PAP). For a fixed proof system Q, the Proof Analysis Problem for Q asks, given a CNF formula φ and a Q-proof of a Resolution lower bound for φ, encoded as $\neg\textsc{Ref}(\varphi)$, whether φ is satisfiable. In contrast to the Proof Analysis Problem for Resolution, which is in $\mathbf{P}$, we prove that PAP for Extended Frege (EF) is $\mathbf{NP}$-complete. In particular, EF can prove Resolution lower bounds on satisfiable formulas without necessarily revealing a satisfying assignment.

Our results yield new insights into proof search and the meta-mathematics of Resolution lower bounds: (i) for every proof system that simulates EF as well as for Resolution, the system is (weakly) automatable if and only if it can be (weakly) automated exclusively on formulas stating Resolution lower bounds; (ii) we provide explicit REF formulas that are exponentially hard for bounded-depth Frege systems; and (iii) for every strong enough theory of arithmetic T we construct explicit unsatisfiable CNF formulas that are exponentially hard for Resolution but for which T cannot prove even a quadratic Resolution lower bound. This latter result applies to arbitrarily strong theories like PA or ZFC, and does not require any complexity-theoretic assumptions.

***Index Terms*—proof complexity, automatability, Resolution, Frege systems, bounded arithmetic**

Noel Arteche was supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. Albert Atserias was supported by the Spanish Research Agency through grant PID2022-138506NB-C22 (PROOFS BEYOND) and the Severo Ochoa and María de Maeztu Program for Centers and Units of Excellence in R&D (CEX2020-001084-M). Susanna F. de Rezende received funding from the Knut and Alice Wallenberg grant KAW 2023.0116, ELLIIT, and the Swedish Research Council grant 2021-05104. Erfan Khaniki was supported by the Royal Society University Research Fellowship URF\R1\211106 "Proof complexity and circuit complexity: a unified approach".
The full version of this paper is available at https://arxiv.org/abs/2506.16956.

I. Introduction

The most natural computational problem arising in proof complexity is that of *proof search*: what is the complexity of finding proofs? In the late 90s, the notion of *automatability*, defined by Bonet, Pitassi, and Raz [1], emerged as a central concept in the theory of propositional proof complexity. A proof system Q is *automatable* if there is a deterministic algorithm that finds a Q-proof of a formula φ in time polynomial in the shortest one available. Except for Tree-like Resolution, which is automatable in quasi-polynomial time [2], no other non-trivial proof system is known to be automatable in polynomial or quasi-polynomial time.

Krajíček and Pudlák [3] and Bonet, Pitassi, and Raz [1] proved that under standard worst-case number-theoretic assumptions in cryptography, strong proof systems like $\mathbf{TC}^0$-Frege and Extended Frege are not automatable. These results can be transferred to $\mathbf{AC}^0$-Frege under slightly stronger hardness assumptions [4], but it seems hard to push them further. Essentially, their proof techniques require some amount of basic number theory to be formalized in the system, something that is likely unworkable for Resolution. Since then, efforts focused on showing the hardness of automating Resolution and related weak systems [5]–[12], culminating in the final answer by Atserias and Müller [13], who proved that Resolution is not automatable unless $\mathbf{P} = \mathbf{NP}$. This is the optimal hardness assumption since $\mathbf{P} = \mathbf{NP}$ implies the automatability of any proof system.

The technique used in [13] relies on the insight that Resolution *cannot reason about its own lower bounds*. To every CNF formula φ, they associate a new formula $\textsc{Ref}_s(\varphi)$ that encodes the statement "there is a size-s Resolution refutation of φ". As a tautology, $\neg\textsc{Ref}_s(\varphi)$ is a natural propositional encoding of a Resolution lower bound. (We postpone to the preliminaries the details of the encoding of the REF formula we use, where we also discuss previously studied variations.)

Pudlák [5] had already shown in 2003 that whenever φ is satisfiable the formula $\textsc{Ref}_s(\varphi)$ is easily refutable by Resolution. On the other hand, Atserias and Müller [13] proved that whenever φ is unsatisfiable, Resolution will require exponential size to refute $\textsc{Ref}_s(\varphi)$, for s being some fixed polynomial in the number of variables of φ, which we omit in the subscript for the rest of this introduction for the sake of clarity.

As a consequence, an automating algorithm running on

2575-8454/25/$31.00 ©2025 IEEE
DOI 10.1109/FOCS63196.2025.00133

formulas of the form $\text{REF}(\varphi)$ can be used to decide SAT in polynomial time: if $\varphi \in \text{SAT}$, then the algorithm must find a short refutation of $\text{REF}(\varphi)$ that Pudlák guarantees must exist; on the other hand, if $\varphi \notin \text{SAT}$, then there are no short refutations of $\text{REF}(\varphi)$, so we can stop the automating algorithm after a polynomial number of steps and be certain that φ is unsatisfiable.

The proof strategy behind the Resolution lower bound on REF formulas was soon adapted to a variety of weak proof systems (those where size lower bounds are known) [14]–[20], although the REF-like formulas used in these spin-off results are no longer natural lower-bound statements for these systems. In general, as pointed out by Pudlák, the question of whether a proof system can prove any of its own lower bounds "is widely open, except for Resolution, and we consider it more important than automatability" [21, p. 3]. It is currently open, for example, whether systems like constant-depth Frege have polynomial-size proofs of any of their own lower bounds.

The feat of the Resolution lower bound on REF formulas, combined with the upper bound for satisfiable formulas, implies that Resolution can *only* reason about "trivial" Resolution lower bounds (i.e., lower bounds on satisfiable formulas, which do not have refutations of any size). This highlights the upper bound construction as something even more remarkable, given that Resolution cannot efficiently argue about its own soundness [22]. Intriguingly, the known upper bound for $\text{REF}(\varphi)$ for satisfiable φ crucially relies on Resolution guessing a satisfying assignment and using it as the backbone of the refutation. It is then natural to ask whether this is necessary:

(Q_1) *Is it the case that whenever there is a short Resolution refutation of* $\text{REF}(\varphi)$*, the proof must* leak *a satisfying assignment?*

By "leaking" we mean that a satisfying assignment is always readable in polynomial time from the given refutation. It is important to note that given a refutation π of $\text{REF}(\varphi)$, one cannot simply restrict π in a way that corresponds to $\text{REF}(\varphi_{\restriction x_1=0})$ and $\text{REF}(\varphi_{\restriction x_1=1})$ to extract a satisfying assignment. This is because the variables of φ are *not* variables of $\text{REF}(\varphi)$. In principle, such a self-reducibility trick seems to require access to an automating algorithm, so that one could successively look for refutations of $\text{REF}(\varphi_{\restriction x_1=0})$ or $\text{REF}(\varphi_{\restriction x_1=1})$, then $\text{REF}(\varphi_{\restriction x_1=b_1,x_2=0})$ or $\text{REF}(\varphi_{\restriction x_1=b_1,x_2=1})$, and so on for all variables. Without access to an automating algorithm, it is not at all clear whether satisfying assignments can be extracted efficiently.

Yet another way of phrasing the lower bound on REF formulas is to see it as the correctness proof of a *lower bound analysis algorithm*. Namely, the result proves that there is an algorithm that given a Resolution refutation π of $\text{REF}(\varphi)$ decides whether φ is satisfiable. The algorithm consists simply of checking whether π is correct and short enough. The correctness of this procedure requires the proof of the lower bound, and this framing naturally leads to the following second natural question regarding REF formulas:

(Q_2) *Is there an algorithm that given an Extended Frege proof* π *of a Resolution lower bound* $\neg\text{REF}(\varphi)$ *decides in polynomial time whether* φ *is satisfiable?*

If the answer were affirmative, this would settle the long-standing open problem of the **NP**-hardness of automating Extended Frege: given a CNF formula φ, construct the formula $\text{REF}(\varphi)$ and run the automating algorithm to find a short Extended Frege refutation. If an algorithm as the one asked for in (Q_2) existed, then we could apply it on this refutation to *analyze* whether φ is satisfiable. This distills the main idea in [13], and the framing of the question in terms of algorithm design suggests that such an algorithm might well be possible *without the need for unconditional Extended Frege lower bounds*.

Overall, the two questions (Q_1) and (Q_2) above hint at the central role of meta-mathematical lower bound statements in the theory of proof search. We believe this calls for a deeper structural understanding that could lead to much-needed conceptual insights in automatability.

A. Contributions

Motivated by questions (Q_1) and (Q_2) above, we introduce a new meta-computational problem relating proofs and computation: the *Proof Analysis Problem*.

For every propositional proof system Q, the *Proof Analysis Problem for* Q (PAP_Q) consists in analyzing Resolution lower bounds proven by Q. More formally, given a CNF formula φ and a Q-proof of the Resolution lower bound encoded by the formula $\neg\text{REF}_s(\varphi)$, the task is to decide whether φ is satisfiable.

The problem can be seen as the computational task of distinguishing "true" Resolution lower bounds (those where φ is actually unsatisfiable) from "trivial" ones (those where the lower bound trivially holds because φ is satisfiable and there is therefore no Resolution refutation, of any size). For those proof systems for which $\text{PAP}_Q \in \mathbf{P}$, we say that Q is *analyzable*. We remark that the REF formula in the definition of PAP_Q is always referring to Resolution refutations, while the proof system Q where $\text{REF}(\varphi)$ is being derived can be arbitrarily strong.

For the case of Resolution itself, the problem $\text{PAP}_{\mathsf{Res}}$ is easy to compute thanks to the lower bound on REF formulas [13]: if π is a correct refutation of $\text{REF}(\varphi)$ and it is small, then φ must be satisfiable. Until now, however, to the best of our knowledge this was the extent of what could be said about PAP-like problems. In particular, we are not aware of any other upper or lower bounds on this problem for proofs systems other than Resolution.

In the language of PAP, questions (Q_1) and (Q_2) can be neatly rephrased as follows:

(Q_1) Does $\text{PAP}_{\mathsf{Res}}$ admit a search-to-decision reduction?

(Q_2) Is Extended Frege analyzable? Namely, is PAP_{EF} in $\mathbf{P}$?

In this work we kick-start the systematic study of these Proof Analysis Problems and settle questions (Q_1) and (Q_2) above. This in turn yields a series of interesting consequences for the meta-mathematics of proof complexity lower bounds as well as proof search. We outline our results next.

1) An algorithm for assignment extraction: On the topic of question (Q_1), our main result is that the search version of $\mathrm{PAP}_{\mathsf{Res}}$ can be solved deterministically in polynomial time.

Theorem I.1 (Assignment extraction algorithm, informal)**.** *The search version of the Proof Analysis Problem for Resolution can be solved in deterministic polynomial time whenever the size parameter s is at least n^3. That is, there is an algorithm that, given a CNF formula φ over n variables and $\mathrm{poly}(n)$ clauses and a Resolution refutation π of $\textsc{Ref}_s(\varphi)$ with $s \geq n^3$, extracts a satisfying assignment for φ in time polynomial in n, s and the size $|\pi|$ of π, whenever φ is satisfiable.*

The question can be stated more formally in terms of *Levin reductions*. A Levin reduction between search problems R_1 and R_2 is a Karp-style many-one reduction that maps instances of R_1 to instances of R_2, with the additional property that it also maps solutions of R_1 to solutions of R_2, and back. The reduction $\varphi \mapsto \textsc{Ref}(\varphi)$ showing that SAT reduces to the Proof Size Problem for Resolution with an exponential gap is clearly Levin in one direction: given a satisfying assignment of φ, Pudlák's construction can craft a refutation of $\textsc{Ref}(\varphi)$. However, it had remained open whether this Levin reduction could be made two-way: given a refutation π of $\textsc{Ref}(\varphi)$, can one always extract a satisfying assignment to φ in polynomial time?

For most if not all natural **NP**-complete languages, the corresponding search problems tend to be complete under Levin reductions. However, the same decision problem could admit different search problems associated to it, and it is known that if $\mathbf{P} \neq \mathbf{NP} \cap \mathbf{coNP}$, then there are **NP** search problems that do not reduce to each other under Levin reductions, while their decision versions are **NP**-complete (and hence do reduce to each other) under Karp reductions (see, for example, [23], [24]). To the best of our knowledge, until now the only natural examples of candidates to be **NP**-hard search problems without Levin reductions were precisely certain problems arising in the context of meta-complexity. One is the Minimum Circuit Size Problem (MCSP), for which Mazor and Pass [25] recently proved that a certain gap version is *not* **NP**-complete under Levin reductions, assuming the existence of indistinguishability obfuscation (iO). The other candidate was precisely the reduction from SAT to the Proof Size Problem for Resolution. Theorem I.1 settles this, giving a two-way Levin reduction.

The existence of the extraction algorithm answers question (Q_1) in the affirmative: Resolution refutations of $\textsc{Ref}_s(\varphi)$ *must leak a satisfying assignment*. This has a certain information-theoretic flavor: the fact that satisfying assignments can *always* be efficiently extracted implies that the most succinct description of a refutation of $\textsc{Ref}_s(\varphi)$ must include the description of a satisfying assignment for φ. We can make this precise in the language of Kolmogorov complexity using the framework of *information efficiency* of Krajíček [26], who studied the minimum time-bounded Kolmogorov complexity (Kt) of propositional proofs.

Theorem I.2 (Assignment extraction as information efficiency, informal)**.** *For every satisfiable CNF formula φ over n variables and $\mathrm{poly}(n)$ clauses,*

$$\mathsf{info}_{\mathsf{Res}}(\neg\textsc{Ref}(\varphi)) \approx \min\{\mathrm{Kt}(\alpha \mid \varphi) \mid \varphi(\alpha) = 1\},$$

where $\mathsf{info}_Q(\psi) \coloneqq \min\{\mathrm{Kt}(\pi \mid \psi) \mid \pi : Q \vdash \psi\}$ is Krajíček's information efficiency function.

To the best of our knowledge, this is one of the first applications of Krajíček's framework.

2) The Proof Analysis Problem for strong proof systems: Motivated by (Q_2), we ask whether PAP is in **P** for strong proof systems. We conclude that the answer is likely negative by proving optimal conditional lower bounds in the form of **NP**-hardness for every proof system that p-simulates Extended Frege (EF).

Theorem I.3 (**NP**-hardness of $\mathrm{PAP}_{\mathsf{EF}}$, informal)**.** *For every propositional proof system S that p-simulates Extended Frege, the Proof Analysis Problem for S is **NP**-complete.*

This means that, unlike Resolution, strong proof systems are seemingly able to prove "trivial" Resolution lower bounds on satisfiable formulas without having to first prove that the underlying formula is satisfiable. In particular, this means Extended Frege is strong enough to *obfuscate* the satisfying assignments. As a consequence, for strong proof systems like Extended Frege, one cannot hope to prove they are **NP**-hard to automate following a strategy similar to that of [13].

3) Formalization of the Atserias–Müller lower bound in PV_1: The inspiration for why the extraction algorithm in Theorem I.1 might exist in the first place comes from *witnessing theorems* in bounded arithmetic. We work here with Cook's theory PV_1 and Buss's S^1_2, which are first-order theories of arithmetic formalizing polynomial-time reasoning. In these theories, if a statement of the form $\forall x \exists y \varphi(x, y)$ with a low-complexity $\varphi(x, y)$ is provable in the theory, then there exists a polynomial-time algorithm that *witnesses* y given x. This implies, in particular, that if a problem is proven **NP**-hard in one of these theories, then the reduction will be a Levin reduction.

The key observation for us is that the statement of the lower bound is itself of this form, a $\forall\Sigma^b_1$ sentence:

> "*for every* formula φ and *every* Resolution refutation π of $\textsc{Ref}(\varphi)$, *there exists* a satisfying assignment for φ, or else π is large."

Thus, if the previous statement were provable in PV_1, we would get a polynomial-time function extracting satisfying assignments given φ and π.

While the extraction algorithm presented in Theorem I.1 is given directly in natural language, it is still worth formalizing the lower bound in bounded arithmetic to obtain a variety of applications.

Theorem I.4 (Atserias–Müller lower bound [13] in PV_1, informal)**.** *The theory PV_1 proves the statement that for every CNF formula φ over n variables and every size parameter*

$s \in \mathbb{N}$, if φ is unsatisfiable and π is a correct Resolution refutation of $\text{REF}_s(\varphi)$, then $|\pi| \geq 2^{\Omega(s/n^2)}$.

Formalizations in bounded arithmetic tend to be particularly interesting when they lead to new proofs of known statements. This has been the case, for example, with Razborov's formalizations of circuit lower bounds leading to the now-famous proof of Håstad's switching lemma via a simpler counting argument [27]. Remarkably, the method introduced by Razborov to formalize the switching lemma is recognized for enabling proofs to at least two major conjectures in combinatorics [28], [29]. Another example is a recent new proof of the Schwartz-Zippel lemma [30], proven via a hybrid argument formalizable in S^1_2. Our formalization of the lower bound on REF formulas also relies on a new proof. We elaborate on this in the technical overview.

We remark that our bound of the form $2^{\Omega(s/n^2)}$ is slightly worse than the original one, which we state here for convenience.

Theorem I.5 (Atserias–Müller lower bound [13])**.** *For every CNF formula φ over n variables and every size parameter $s \in \mathbb{N}$, if φ is unsatisfiable and π is a correct Resolution refutation of $\text{REF}_s(\varphi)$, then $|\pi| \geq 2^{\Omega(s/n)}$.*

The difference in the bound means that we can only show that PV_1 proves hardness of $\text{REF}_s(\varphi)$ for $s \geq n^3$. We leave it open whether PV_1 can achieve the original $2^{\Omega(s/n)}$ bound via a different argument. In any case, this is not particularly important for our applications. We comment on this further in the technical overview.

4) Formalization of Pudlák's upper bound in Resolution: We complement the formalization of the lower bound with a formalization of the upper bound [5], showing that there are short refutations of $\text{REF}(\varphi)$ whenever φ is satisfiable. This construction can be carried out by a constant-depth circuit and could be formalized in S^1_2, but certainly also in much weaker theories. We prove the somewhat surprising fact that the construction can be proven correct in Resolution itself.

Theorem I.6 (Pudlák's upper bound [5] in Resolution, informal)**.** *There is a polynomial-size depth-2 Boolean circuit $P(\alpha, \varphi, s)$ of fan-in 2 that given a CNF formula φ, a satisfying assignment α, and $s \in \mathbb{N}$, outputs a Resolution refutation π of $\text{REF}_s(\varphi)$. Furthermore, the correctness of this circuit P has polynomial-size proofs in Resolution.*

That is, not only Resolution has short refutations of $\text{REF}(\varphi)$ when φ is satisfiable: Resolution can show that the circuits generating these refutations from satisfying assignments are correct. This, again, is in striking contrast with the fact that Resolution does not have small proofs of its own soundness [22].

5) Propositional fragments of Atserias–Müller: automatability in terms of REF *formulas:* The main consequence of the extraction algorithm together with its formalization in bounded arithmetic is the following precise characterization theorem relating the provability of a formula $\neg\varphi$ to the provability of the formula $\neg\text{REF}(\text{REF}(\varphi))$. (For the sake of clarity, we ignore for now the exact size parameters of the REF formulas, which are always some fixed polynomials; in general, when we write $S \vdash_{\text{poly}} \varphi$ we mean that S has polynomial-size proofs of φ, and by "reasonable proof system" we mean essentially that the system is closed under *modus ponens*.)

Theorem I.7 (Propositional fragments of Atserias–Müller, informal)**.** *Let S be a reasonable propositional proof system that simulates Extended Frege. Then, for every sequence $\{\varphi_n\}_{n\in\mathbb{N}}$ of unsatisfiable CNF formulas,*

$$S \vdash_{\text{poly}} \neg\varphi_n \quad \textit{if and only if} \quad S \vdash_{\text{poly}} \neg\text{REF}(\text{REF}(\varphi_n)).$$

The lower bound on REF formulas says that for every unsatisfiable φ, the corresponding $\text{REF}(\varphi)$ is hard for Resolution, making $\text{REF}(\text{REF}(\varphi))$ unsatisfiable. The latter encodes the statement "$\text{REF}(\varphi)$ is hard for Resolution", and our theorem shows that when restricted to the reasoning power of a specific proof system S, such a lower bound has small proofs if, and only if, S has short proofs of the unsatisfiability of φ in the first place. That is, the *fragment* of the Atserias–Müller lower bound that has short proofs in S is precisely the one corresponding to the formulas that S can prove unsatisfiable with short proofs.

This characterization is surprisingly tight and has consequences for automatability and proof search. Since we can relate the proof size of φ in S to the proof size of $\text{REF}(\text{REF}(\varphi))$, this means that looking for proofs of $\text{REF}(\text{REF}(\varphi))$ can be a proxy for searching for proofs of φ.

Theorem I.8 (Automatability in terms of REF formulas, informal)**.** *For every reasonable proof system S that simulates Extended Frege as well as for Resolution itself,*

(i) S is automatable if, and only if, S is automatable exclusively on REF *formulas;*
(ii) S is weakly automatable if, and only if, S is weakly automatable exclusively on REF *formulas.*

We remark again that these REF formulas are always talking about Resolution, not about S. That is, for every strong enough proof system, *efficient proof search over all tautologies is equivalent to efficient proof search over Resolution lower bounds*.

Until now no such general structural result was known that related proof search generally to proof search for a particular class of formulas. This goes in line with a question of Pich and Santhanam [31], who asked whether automating a proof system on truth-table tautologies (i.e., formulas stating circuit lower bounds) implies the automatability of the system on all tautologies. We have proved that this is the case for the class of formulas stating Resolution lower bounds in place of truth-table tautologies.

6) Unprovability of Resolution lower bounds: Theorem I.6, together with Theorem I.5, further imply that true Resolution lower bounds can be essentially arbitrarily hard to prove. Namely, if S is a propositional proof system where $\{\varphi_n\}_{n\in\mathbb{N}}$ is a sequence of formulas that S cannot refute in polynomial size, then S cannot refute $\{\text{REF}(\text{REF}(\varphi_n))\}_{n\in\mathbb{N}}$ either.

Theorem I.9 (Propositional unprovability of Resolution lower bounds, informal)**.** *Let Q be a reasonable propositional proof system that simulates Resolution. If $\{\varphi_n\}_{n\in\mathbb{N}}$ is a sequence of hard unsatisfiable CNF formulas for Q, where φ_n has n variables and size $|\varphi_n| = \mathrm{poly}(n)$, then*

(i) the formulas $\mathrm{REF}_{n^2}(\varphi_n)$ over $N = \mathrm{poly}(n)$ variables are all unsatisfiable and require size $2^{N^{\Omega(1)}}$ to be refuted in Resolution;

(ii) yet, Q does not have polynomial-size refutations of the formulas $\mathrm{REF}_{N^2}(\mathrm{REF}_{n^2}(\varphi_n))$ stating quadratic lower bounds on $\mathrm{REF}_{n^2}(\varphi_n)$.

There is nothing special about quadratic lower bounds being unprovable—one can get arbitrarily small polynomial lower bounds by tweaking the encoding. See the discussion after Theorem I.11.

We note that Iwama showed in 1997 that the Proof Size Problem for Resolution is **NP**-complete [32]. This means, in particular, that its complement in **coNP**-complete and hence, unless $\mathbf{NP} = \mathbf{coNP}$, no propositional proof system can efficiently derive *all* tautological REF formulas (i.e., all true Resolution lower bounds), or else there would be a polynomially bounded proof system. While this has a similar flavor to our result, our theorem is different in at least two aspects. First, from an explicit family of hard tautologies we obtain an explicit family of hard REF formulas for the system, in a generic way. Second, the parameters are essentially optimal: we identify a sequence of unsatisfiable formulas for which an exponential (and hence maximal) Resolution lower bound holds —while the system Q in question cannot even prove a quadratic lower bound.

As a corollary of Theorem I.9, for example, we get the first explicit lower bounds for REF formulas in bounded-depth Frege systems.

Corollary I.10 (Hard REF formulas for bounded-depth Frege, informal)**.** *For every $d \le O(\log n/\log\log n)$, the formulas $\mathrm{REF}(\mathrm{REF}(\mathrm{PHP}_n))$ are all unsatisfiable but require size $\exp\big(\Omega(n^{1/(2d+1)})\big)$ to be refuted in depth-d Frege systems.*

We build here on the best-known PHP lower bounds by Håstad [33], but even better bounds could be obtained by applying the same argument to Tseitin formulas using the results of Håstad and Risse [34]. This corollary contrasts again with the open question of Pudlák [21] about whether constant-depth Frege proves any of its own lower bounds.

Finally, using similar ideas, we obtain unconditional independence results for first-order theories of arithmetic.

Theorem I.11 (First-order unprovability of Resolution lower bounds, informal)**.** *Let T be a consistent first-order theory extending Robinson Arithmetic by a set of polynomial-time recognizable axioms. Then, there exists a sequence of unsatisfiable propositional formulas $\{\psi_N\}_{N\in\mathbb{N}}$ described uniformly by a polynomial-time algorithm, where ψ_N has N variables, such that*

(i) Resolution refutations of the formula ψ_N require size $2^{N^{\Omega(1)}}$;

(ii) there exists $c > 0$ such that the theory T cannot prove $\Omega(N^c)$ lower bounds on the Resolution size of these refutations; that is, there is $N_0 \in \mathbb{N}$ such that the lower bound expressed by the first-order sentence $\forall N \forall \pi\,(N > N_0 \wedge \mathrm{Ref}_{\mathsf{Res}}(\psi_N, \pi) \to |\pi| > N^c)$ is unprovable in T.

We remark that the theory T in this theorem can be arbitrary strong. This implies that, unconditionally, theories like Peano Arithmetic (PA) cannot prove all true Resolution lower bounds. The same ideas apply to Zermelo-Fraenkel Set Theory (ZFC) and similarly powerful formal systems.

We also note that the constant $c > 0$ in the exponent of the unprovable lower bound depends on the definition of ψ_N. In general, one can alter ψ_N to get an unprovable lower bound of the form $\Omega(N^c)$ for any fixed constant $c > 0$.

B. Technical overview

Next we provide a technical overview of the main proof ideas and how these are combined to yield our main results and corollaries.

1) Assignment extraction: We obtain the extraction algorithm in Theorem I.1 by derandomizing the proof of the Resolution lower bound for the REF formulas. The original proof revolves around the concept of *block-width* (called *index-width* in [13]) in Resolution refutations of $\mathrm{REF}_s(\varphi)$. The variables of the formula are arranged into s *blocks*, each encoding a clause in the purported refutation of size s. The block-width of a refutation π is then the largest number of blocks mentioned in a clause of the refutation π. The proof proceeded in two steps:

1) derive a *block-width lower bound*, showing that if $\varphi \notin$ SAT, then the block-width of any refutation of $\mathrm{REF}_s(\varphi)$ must be large;
2) by a *random restriction argument*, argue that if the refutation π is small, there exists a restriction that makes the block-width of the restricted refutation small, contradicting the previous point.

Our algorithm works by following these steps in reverse. First, given a refutation π from which we want to extract a satisfying assignment, instead of sampling a restriction at random from a specific distribution, we construct a restriction *deterministically* in a greedy fashion, tailored to the specifics of π. This is reminiscent of the kind of greedy deterministic restrictions used by Cook and Pitassi [35] to formalize Haken's lower bound for the pigeonhole principle in bounded arithmetic, and more broadly in the style of Beame and Pitassi [2], Clegg, Edmonds, and Impagliazzo [36] and Ben-Sasson and Wigderson [37]. Our algorithm runs in deterministic polynomial time and always succeeds in finding a restriction that reduces the block-width to $O(\sqrt{s \log |\pi|})$.

In the second step, we look at the proof of the block-width lower bound and interpret it as a Prover-Delayer game in the style of Atserias and Dalmau [38]. The Prover issues queries about the values of the variables of $\mathrm{REF}_s(\varphi)$, or forgets previously recorded such values, and the Delayer replies

following a concrete strategy that allows them to keep playing until a large number of blocks appear queried. Our algorithm traverses the Resolution refutation guided by the Delayer's strategy in the Prover-Delayer game. We can then prove that this Delayer's strategy will, after a polynomial number of steps, reach either

(a) a clause of *high block-width*, or
(b) a clause encoding a *satisfying assignment* to φ.

Since the greedy deterministic restriction in the first step made sure the block-width is small, the Delayer will be guaranteed to find a satisfying assignment.

We note that our deterministic restriction only achieves a reduction of block-width to $O(\sqrt{s \log |\pi|})$, while using a random restriction one could achieve up to $O(\log |\pi|)$. In fact, if one allows randomness in the extraction algorithm, then an argument similar to the random restriction of [13] yields a zero-error probabilistic polynomial-time extraction algorithm that works even when the refutation π being analyzed is for the formula $\textsc{Ref}_{n^2}(\varphi)$. In contrast, the price to pay for determinism is that the size parameter s should be at least n^3.

2) $\mathbf{NP}$-hardness of $\mathsf{PAP}_{\mathsf{EF}}$: The idea behind the hardness proof in Theorem I.3 is best explained as a reduction from the Minimum Circuit Size Problem (MCSP)—although given that MCSP is not known to be $\mathbf{NP}$-hard, the actual proof in the main text is a bit more technical and goes instead via a reduction from VERTEX COVER. In 2004, Razborov [39] proved that Resolution cannot efficiently prove circuit size lower bounds. This statement is captured by the the well-known *truth-table tautologies* $\textsc{tt}(f, s)$ stating that a truth-table f can be computed by a circuit of size s. Then, the formula $\textsc{Ref}_t(\textsc{tt}(f, s))$ states that there exists a size-t Resolution refutation of $\textsc{tt}(f, s)$. By Razborov's lower bound there are no such Resolution refutations, meaning that $\textsc{Ref}_t(\textsc{tt}(f, s))$ is unsatisfiable for values of t polynomial in $|\textsc{tt}(f, s)|$.

An interesting feature of Razborov's lower bound is that it is agnostic about whether f is actually hard for circuits of size s. Namely, even if f was computable by size-s circuits, making $\textsc{tt}(f, s)$ satisfiable and hence $\textsc{Ref}_t(\textsc{tt}(f, s))$ unsatisfiable for a trivial reason, Razborov's argument can still prove the unsatisfiabilty of this REF formula without exhibiting a satisfying assignment for $\textsc{tt}(f, s)$.

Now, suppose that Razborov's lower bound was formalizable in, say, Extended Frege[1] in a uniform manner. That is, suppose there is a polynomial-time algorithm that given a truth-table f and size parameters s and t outputs an EF proof π such that

$$\pi : \mathsf{EF} \vdash \neg \textsc{Ref}_t(\textsc{tt}(f, s)).$$

Then, if $\mathsf{PAP}_{\mathsf{EF}}$ happened to be in $\mathbf{P}$, there would be a polynomial-time algorithm that given π would decide whether $\textsc{tt}(f, s) \in \text{SAT}$, which is the same as deciding MCSP. Hence, $\mathsf{PAP}_{\mathsf{EF}}$ (or PAP for whatever system capable of formalizing Razborov's proof) would be at least as hard as MCSP.

[1]We note that it is not known nor clear at all that Razborov's argument goes through in EF. The assumption is only for the sake of exposition.

For our proof we do not formalize Razborov's lower bounds, and instead instantiate this idea for a specific propositional encoding of VERTEX COVER for which Resolution lower bounds follow from Haken's lower bound for the pigeonhole principle. Here we leverage the formalization of Cook and Pitassi, who showed that Haken's lower bound is provable in EF [35].

3) Formalization of the upper and lower bounds: A large part of our technical contribution consists in formalizing the proofs leading to the $\mathbf{NP}$-hardness of automating Resolution. We summarize below some of the challenges encountered, and the solutions devised.

a) The Atserias–Müller lower bound in PV_1: Different proofs of the lower bound exist in the literature, but none of them seem directly formalizable in PV_1. The original proof has the caveat of the random restriction argument, which might be formalizable in Jeřábek's theory APC_1, but likely not in PV_1. In addition, the block-width lower bound is proven by relating small refutations to the canonical exponential-size tree-like refutation of any formula, which could be hard to reason about in bounded theories.

In the work of de Rezende, Göös, Nordström, Pitassi, Robere, and Sokolov [18] two alternative proofs were presented. The first proof consists of a random restriction followed by a block-width lower bound proven via a reduction to the retraction weak pigeonhole principle. The random restriction presents the same formalization issues as the original proof, and the block-width reduction is equally problematic: the decision tree reduction they use has low depth, which is necessary to transfer the lower bounds on (block-)width, but the size of the decision tree itself seems to be superpolynomial, and hence cannot be reasoned about in PV_1. The second proof of de Rezende, Göös, Nordström, Pitassi, Robere, and Sokolov uses this same block-width lower bound followed by the size-width trade-offs of Ben-Sasson and Wigderson [37]. The block-width lower bound is still problematic, of course, but in addition to this, the statement of Ben-Sasson and Wigderson—"for every small Resolution refutation, there exists another Resolution refutation in small width"—is itself impossible to formalize in bounded arithmetic. The reason for this is that, as demonstrated by Thapen [40], in general these narrow proofs can require superpolynomial size, and therefore the statement of Ben-Sasson and Wigderson cannot possibly be a bounded formula.

Finally, Garlík [41] has proven lower bounds on the REF formulas for the so-called non-relativized encoding. Unfortunately for us, his proofs encounter the same barrier: they rely on random restriction arguments, which are in any case more involved than the original ones.

We resolve these issues by coming up with new proof, inspired by the extraction algorithm, that modifies both ingredients in the original proof, yielding Theorem I.4. The random restriction argument is replaced by a greedy deterministic restriction just like the one used in the extraction algorithm. For the block-width lower bound, we show that the argument can be completely described by a Prover-Delayer game without referring to the exponential-size canonical tree-like refutation,

making the entire proof formalizable in PV_1.

We remark that moving from the random restriction to the deterministic one comes at the cost of a slightly worse lower bound. The original size bound on REF formulas is of the form $2^{\Omega(s/n)}$ and hence yields $2^{\Omega(n)}$ Resolution size lower bounds for all REF_s formulas with $s \geq n^2$. Our deterministic restriction, in line with the parameters of the extraction algorithm, achieves a lower bound of $2^{\Omega(s/n^2)}$ which is exponential in $\Omega(n)$ for $s \geq n^3$. It seems reasonable that the original proof with the random restriction can be formalized in APC_1, but we have not carried out this formalization.

b) Pudlák's upper bound in Resolution: For the upper bound in Resolution (Theorem I.6), our proof is based on a careful analysis of the construction that makes it possible to describe the construction by a low-depth circuit. Carrying out the proof of the correctness of this circuit in Resolution is tedious, but ultimately clear once the right description of the circuit is provided.

An interesting technical ingredient is the fact that the correctness statement itself ("if α is a satisfying assignment, then the circuit outputs a correct refutation") is an implication that cannot be immediately expressed as a polynomial-size CNF formula. To deal with this, we devise a construction using extension variables that simulates negations of CNF formulas in Resolution, which we name *pseudo-negations*. With the aid of this pseudo-negation operators, Resolution can carry out *modus ponens* inferences.

4) Consequences:

a) Characterization of the propositional fragments of Atserias–Müller: Our characterization theorem (Theorem I.7) is a consequence of the formalization of the lower bound in PV_1 (Theorem I.4) and the upper bound in Resolution itself (Theorem I.6). Those results, in the propositional setting, imply that

1) for the extraction algorithm E, we have

$$\mathsf{EF} \vdash \text{REF}(\text{REF}(\varphi), \pi) \to \text{SAT}(\varphi, E(\varphi, \pi));$$

2) for Pudlák's algorithm P, we have

$$\mathsf{Res} \vdash \text{SAT}(\varphi, \alpha) \to \text{REF}(\text{REF}(\varphi), P(\varphi, \alpha)).$$

Here, we highlight the variables of the REF formula encoding a refutation π as the second argument of REF. In particular, the Resolution proof of the correctness of P is also possible in EF. Then, simple use of contraposition allows us to go from $\neg\varphi$ to $\neg\text{REF}(\text{REF}(\varphi))$, and vice versa. That is, if EF can prove $\neg\varphi$, then it can also prove it in the encoding $\neg\text{SAT}(\varphi, \alpha)$, and when substituting $E(\varphi, \pi)$ for α, where π are the only free variables, contraposition on item (1) gives us that EF derives $\neg\text{REF}(\text{REF}(\varphi), \pi)$. The other direction is analogous.

b) Automatability in terms of REF *formulas:* For the characterization of automatability in Theorem I.8 to go through we build on Theorem I.7 and additionally show that the characterization given there is not only in terms of proof size, but it is actually constructive. Given a proof of $\neg\varphi$ in S we can efficiently construct a proof of $\neg\text{REF}(\text{REF}(\varphi))$, and vice versa. In this way, searching for proofs of $\text{REF}(\text{REF}(\varphi))$ is a proxy for the proofs of φ.

Remarkably, our proof techniques fail for proof systems strictly between Extended Frege and Resolution. The upper bound in Resolution does imply that from a refutation of $\text{REF}(\text{REF}(\varphi))$ we can obtain a refutation of φ. Unfortunately, it is our extraction algorithm (Theorem I.1) what guaranteed that if φ has a refutation of size t, then $\text{REF}(\text{REF}(\varphi))$ has a refutation of size $\text{poly}(t)$. In Extended Frege this is true thanks to the extraction algorithm, but it seems conceivable that weaker systems might be able to easily prove $\neg\varphi$ without being able to prove $\neg\text{REF}(\text{REF}(\varphi))$ efficiently. (For Resolution itself this result does go through, for the more ad-hoc reason that REF formulas talk about Resolution itself).

c) Unprovability of Resolution lower bounds: For Theorem I.9 we exploit Theorem I.6: if there is a short refutation of $\text{REF}(\text{REF}(\varphi))$, then there is a short refutation of φ. Since we formalized the upper bound construction in Resolution, the result applies to any proof system that contains Resolution (and behaves naturally in the sense that it is closed under *modus ponens*). Then, if φ is a hard formula for Q and Q simulates Resolution, we have that $\text{REF}(\varphi)$ is unsatisfiable. By the lower bound on REF formulas (Theorem I.5), this formula is exponentially hard for Resolution, making $\text{REF}(\text{REF}(\varphi))$ unsatisfiable as well—but hard to refute for Q.

In the first-order setting, Theorem I.11 relies again on the formalization of the upper bound on REF formulas. This time, instead of starting from a sequence of hard propositional formulas, we can leverage Gödel's second incompleteness theorem to start from a sentence (the consistency of T) that is unconditionally unprovable in T. From this follows that T cannot prove the soundness of a certain propositional system based on T (the so-called *strong proof system of* T [21]). We then consider the $\text{REF}(\cdot)$ formula around these soundness statements. We conclude that if T could derive the lower bound on the $\text{REF}(\cdot)$ formulas in question, it would also be able to prove the soundness of the strong proof system of T and, as a consequence, T would derive its own consistency. Since Gödel's incompleteness theorem gives us sentences that are unconditionally independent of T, the corresponding Resolution lower bounds are also unconditionally unprovable in T. This works essentially for any theory of arithmetic subject to Gödel's incompleteness phenomenon, and does not rely on any complexity-theoretic assumptions.

C. Related work

Our work fits into a trend in complexity theory concerned with the meta-mathematics of computational complexity, which has gained remarkable momentum in recent years. Most of this work has been primarily concerned with the formalization of cornerstone results of computational complexity in bounded arithmetic and establishing unprovability and logical independence as barrier results. The literature is too vast to review here, so refer the reader to the recent survey of Oliveira [42]. In parallel, there has been a growing body of work deploying tools and ideas from mathematical logic to prove

complexity-theoretic statements (see, e.g., [31], [43]–[56]). Our work continues in this direction.

Two recent works conceptually related to our investigations on REF formulas merit further discussion. Santhanam and Tzameret [57] initiated a general study of REF formulas for arbitrarily strong proof systems. In particular, they studied *iterations* of these formulas, which are reminiscent of the nested $\text{REF}(\text{REF}(\varphi))$ formulas that feature in our work. Their REF formulas are not limited to Resolution, and they consider the iterated version of REF^Q when REF^Q talks about an arbitrarily strong proof system Q. While we are unable to connect our work on analyzability to their results, our characterization of proof size in terms of REF formulas (Theorem I.7) has conceptual ties to their Iterated Lower Bounds Hypothesis.

The other relevant work is the research of Li, Li, and Ren [58], who studied the provability of Resolution lower bounds in relativized theories of bounded arithmetic in the context of **TFNP**. Until their work, the only formalization of proof complexity lower bounds that we are aware of is that of Cook and Pitassi [35]. Li, Li, and Ren studied so-called *refuter problems* in proof complexity: given a purported Resolution refutation of, say, PHP_n, which is smaller than the known lower bounds, find a mistake in the proof (which must certainly exist, due to these very lower bounds). They connect the provability of lower bounds to the complexity of solving these refuter problems in subclasses of **TFNP**. While their results yield formalizations of some proof complexity lower bounds, our results are essentially incomparable. First, their provability results are for *relativized* theories of bounded arithmetic, where the given Resolution refutation is accessed through an oracle, while our proofs are in the non-relativized theories, where we can quantify over the objects in question. Second, they consider the provability of lower bounds for explicit families of tautologies like the pigeonhole principle or the Tseitin formulas. In contrast, the lower bound we are concerned is a sort of *meta lower bound*: it tells us that the $\text{REF}(\varphi)$ formulas are hard whenever φ is unsatisfiable. We believe, however, that the **TFNP** perspective on analyzability might shed light on some of our open questions.

D. Open problems

a) Analyzability of constant-depth Frege and other weak proof systems: Similar techniques to those of [13] have been employed to prove the **NP**-hardness of automating other weak proof systems like Regular and Ordered Resolution [14], [15], k-DNF Resolution [16], Cutting Planes [17], Nullstellensatz and Polynomial Calculus [18], the OBDD proof system [19] and, more recently, even $\mathbf{AC}^0$-Frege [20]. All proof systems weaker than Resolution are analyzable just because Resolution is (i.e, their corresponding PAP problems are in **P**), since analyzability is downwards closed under p-simulations. For the stronger systems, the question remains open. Are these systems analyzable? What about their search versions?

We highlight the analyzability of constant-depth Frege as a particularly interesting problem. While we have proven some unconditional lower bounds on REF formulas here, it is open whether $\mathbf{AC}^0$-Frege can prove any true Resolution lower bounds at all. It has been conjectured in the past that the PHP lower bound might be formalizable in these systems, at least in quasi-polynomial size. If this was possible, the **NP**-hardness of PAP_{EF} in Theorem I.3 could be improved all the way to these systems.

b) **FP***-completeness of the search version of* $\text{PAP}_{\mathsf{Res}}$*:* While we have shown that assignment extraction can be performed in polynomial time, our algorithm does not seem to be possible anywhere below **P**. The algorithm seems hard to parallelize, which raises the question of whether the search version of $\text{PAP}_{\mathsf{Res}}$ is in **NC** or even below. This is related to the question of whether the formalization of the lower bound on REF formulas is provable in theories weaker than PV_1. If the statement was provable in, say, VNC^1, witnessing theorems would give us a extraction in $\mathbf{FNC}^1$. We conjecture that this improvement is in fact impossible, and that the search problem of $\text{PAP}_{\mathsf{Res}}$ is complete for **FP** under $\mathbf{AC}^0$-reductions, but we are unable to prove it. The reduction, if true, likely requires some new technical idea. This would imply, amongst other things, that V^0 does not prove the lower bound on REF formulas, unconditionally.

c) On the weak automatability of Resolution: Recall that a proof system is *weakly automatable* if there exists a proof system that p-simulates it and is automatable. By our Theorem I.8, the weak automatability of Resolution is equivalent to a proof system Q simulating Resolution and being automatable on REF formulas. If $Q \geq \mathsf{EF}$, then our theorem would imply that Q itself would be automatable on all formulas, hitting cryptographic hardness results [1], [3], [4], [52]. However, if Q is strictly weaker than EF, our statement does not apply and the automatability of Q on REF formulas does not imply automatability on all formulas. This does not seem to contradict any hardness assumptions. Of course, no such Q is known to be efficiently automatable on Resolution lower bound statements, but this raises again the question of whether some non-trivial algorithm weakly automating Resolution might be plausible.

E. Structure of the paper

The paper is structured as follows. After the preliminaries in Section II, we dedicate Section III to formally defining the Proof Analysis Problem and stating some basic facts about it. Section IV proves Theorem I.1, describing the algorithm for the search version of $\text{PAP}_{\mathsf{Res}}$, while the proof of Theorem I.2 is omitted in this version, but can be found in the full version [59]. Section V proves Theorem I.3, giving **NP**-hardness of PAP_{EF} and stronger systems. The remaining results stated earlier can all be found in the full version of the paper [59].

II. PRELIMINARIES

We assume the reader to be familiar with the central concepts of computational complexity theory. Below, we review the essential definitions and facts involving proof complexity and bounded arithmetic that feature in the paper. For a more comprehensive treatment of proof complexity, we refer to

Krajíček [60]. For bounded arithmetic, the recent survey of Oliveira [42] covers all the necessary material in the style of the meta-mathematics of computational complexity, which aligns with the style of our work. Other classical texts in logic and bounded arithmetic also cover these contents (see, e.g., [61]–[64]).

A. Proof complexity

Following the classical definition of Cook and Reckhow [65], a *propositional proof system* S for the set TAUT of propositional tautologies is a polynomial-time function $S : \{0,1\}^* \to \text{TAUT}$ whose range is exactly TAUT. We think of S at the polynomial-time verifier mapping proofs to the statements they prove; i.e., if $S(\pi) = \varphi$, then we say π is an S-proof of φ. It is often convenient to think of a proof system as establishing *unsatisfiability*; thus, if φ is an unsatisfiable formula and π is an S-proof of the tautology $\neg\varphi$, then we say that π is an *S-refutation* of φ, or an S-proof of the unsatisfiability of φ. Since we deal exclusively with classical logic, here and below we tacitly gloss over the distinction between the formulas $\neg\neg\varphi$ and φ; this is particularly useful for literals ℓ, where $\neg\ell$ is sometimes used to denote the complementary literal.

For a tautology φ and a proof system S, we denote by $\mathsf{size}_S(\varphi) \coloneqq \min_{\pi : S(\pi)=\varphi} |\pi|$ the *size* of its smallest S-proof. A proof system S is *polynomially bounded* if there exists a constant $c \in \mathbb{N}$ such that for all $\varphi \in \text{TAUT}$ we have $\mathsf{size}_S(\varphi) \leq |\varphi|^c$. For a sequence $\varphi = \{\varphi_n\}_{n\in\mathbb{N}}$ of tautologies, we write $S \vdash_{\text{poly}} \varphi$ or simply $S \vdash_{\text{poly}} \varphi_n$ to express that $\mathsf{size}_S(\varphi_n) = |\varphi|^{O(1)}$ as n grows. When we want to emphasize that it is via a specific proof π that S proves φ_n, we write $\pi : S \vdash \varphi_n$. More generally, for $s \in \mathbb{N}$, we write $S \vdash_s \varphi_n$ to express that there exists an S-proof π of size $|\pi| \leq s$ such that $\pi : S \vdash \varphi_n$.

We say that a proof system S *simulates* another system Q, written $S \geq Q$, if there exists a constant $c \in \mathbb{N}$ such that for every $\varphi \in \text{TAUT}$ we have $\mathsf{size}_S(\varphi) \leq \mathsf{size}_Q(\varphi)^c$. We additionally say that S *p-simulates* Q and write $S \geq_{\text{p}} Q$ if there exists a polynomial-time computable function sending Q-proofs to S-proofs of the same formula; i.e., there exists a polynomial-time computable function f such that for every $\varphi \in \text{TAUT}$ and every $\pi : Q \vdash \varphi$, we have $f(\pi) : S \vdash \varphi$. We say that two proof systems S and Q are *polynomially equivalent* if $S \geq_{\text{p}} Q$ and $Q \geq_{\text{p}} S$. A proof system S is *optimal* if $S \geq Q$ for every propositional proof system Q, and respectively *p-optimal* if $S \geq_{\text{p}} Q$ for every propositional proof system Q.

A *literal* is a propositional variable or its negation. Given a formula $\varphi(x_1, \ldots, x_n)$, a *literal substitution* is a mapping of the form $\rho : \{x_1, \ldots, x_n\} \to \{x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n, 0, 1\}$ that replaces variables by other literals or substitutes constants in their place. We denote by $\varphi_{\restriction\rho}$ the substituted formula $\varphi(\rho(x_1), \ldots, \rho(x_n))$, with the convention that every resulting occurrence of $\neg\neg x_i$ is replaced by x_i. A *restriction* is a particular case of a variable substitution, where all variables are mapped to either 0, 1, or themselves. We say that a proof system S is *closed under substitutions* (respectively, *closed under restrictions*) if there exists a constant $d \in \mathbb{N}$ such that for every tautology φ and every literal substitution (respectively, restriction) ρ, it holds that $\mathsf{size}_S(\varphi_{\restriction\rho}) \leq \mathsf{size}_S(\varphi)^d$. All the explicit proof systems dealt with in this work (i.e., the ones described below, like Resolution or Frege or Extended Frege systems) are closed under literal substitutions. In these cases, a proof of the substituted formula can be obtained directly by applying the substitution line by line to every formula appearing in a proof π of φ, and we hence denote by $\pi_{\restriction\rho}$ the corresponding substituted proof of $\varphi_{\restriction\rho}$.

1) Resolution: A central proof system in this work is Resolution (Res). We usually see this as a refutation system for CNF formulas. Accordingly, we sometimes write $\pi : \mathsf{Res} \vdash \neg\varphi$ for a CNF formula φ, to mean that π is a Resolution refutation of φ, hence a proof of the tautology $\neg\varphi$. In this way, Res is a Cook-Reckhow proof system for the fragment of TAUT made of the formula of the form $\neg\varphi$, where φ is an unsatisfiable CNF formula. Through the standard Tseitin transformation of an arbitrary propositional formula into equisatisfiable CNF form, Res can also be seen as a Cook-Reckhow proof system for TAUT itself; we do not need the details of this in this paper.

A *literal* is a propositional atom or its negation, a *clause* is a disjunction of literals, and a *CNF formula* is a conjunction of clauses. We see clauses as sets of literals, and write simply $C \subseteq D$ to express that C is a subclause of D. A *Resolution refutation* of an unsatisfiable CNF formula $\varphi = C_1 \wedge \cdots \wedge C_m$ over variables $x_1, \ldots, x_n$ is a sequence $D_1, \ldots, D_s$ of clauses over $x_1, \ldots, x_n$ such that $D_s = \bot$, denoting the empty clause, and for every $i \in [s-1]$, the clause D_i either (a) is one of the clauses $C_1, \ldots, C_m$ of φ, or (b) is a *weakening* of a previous clause, meaning that $D_i \supseteq D_j$ for some $1 \leq j < i$, or (c) has been obtained from two previous clauses $D_j = A \vee x$ and $D_k = B \vee \neg x$, for $j, k < i$, by an application of the *Resolution rule*:

$$\frac{A \vee x \qquad B \vee \neg x}{A \vee B} \ \text{(Res)}$$

We say that $A \vee B$ is obtained by *resolving* over x. The *length* of π, denoted by $\mathrm{length}(\pi)$, is s.

To every Resolution refutation π we can associate a directed acyclic graph in a natural way, and we often do so implicitly. We denote by $\mathrm{depth}(\pi)$ the length of the longest path in the dag, starting from the root labeled by the empty clause $\bot$. The number of vertices in this graph is precisely $\mathrm{length}(\pi)$.

We will also deal with a mild extension of the Resolution system, known as *k-DNF Resolution* [66], denoted $\mathsf{Res}(k)$ for $k \geq 1$. The system $\mathsf{Res}(k)$ is also a refutational system, but lines are *k-DNF* formulas, which are unbounded fan-in disjuctions of *k-terms*, conjunctions of up to k literals. A clause is a 1-disjunction. The system consists of a weakeaning and an introduction rule, together with a Cut rule,

$$\frac{A}{A \vee B}\ \text{(Weak)} \qquad \frac{A \vee \ell_1 \qquad B \vee (\ell_2 \wedge \cdots \wedge \ell_s)}{A \vee B \vee (\ell_1 \wedge \cdots \wedge \ell_s)}\ \text{(}\wedge\text{-Intro)}$$

$$\frac{A \vee (\ell_1 \wedge \cdots \wedge \ell_s) \qquad B \vee \neg\ell_1 \vee \cdots \vee \neg\ell_s}{A \vee B}\ \text{(Cut)}$$

where A and B are k-DNF formulas and $s \leq k$.

It is easy to see that Resolution (Res) corresponds to $\mathsf{Res}(1)$.

2) Frege systems: Through this work we reason about Resolution refutations within much stronger systems for propositional logic. A *Frege system* [65] consists of a finite set of axiom schemas and inference rules that are sound and implicationally complete for the language of propositional tautologies built from the Boolean connectives negation ($\neg$), conjunction ($\wedge$), and disjunction ($\vee$). A *Frege proof* is then a sequence of formulas where each formula is obtained by either substitution of an axiom schema or by application of an inference rule on previously derived formulas. The specific choice of rules does not affect proof size up to polynomial factors, as long as there are only finitely many rules and these are sound and implicationally complete [65]. We refer to Cook and Reckhow [65] or Krajíček [60, §2.1] for specific examples of choices for these rules and axioms. One can alternatively define Frege systems in the formalism of Natural Deduction or the Sequent Calculus for classical propositional logic, but we will not be concerned with these syntactic details.

Of central importance for us is the *Extended Frege* (EF) system [65], in which proof lines can be succinctly written as Boolean circuits rather than formulas [67]. In general, for a circuit class $\mathbf{C}$, one can consider the proof system $\mathbf{C}$-Frege, in which lines are restricted to be Boolean circuits of that type. We are particularly interested in the $\mathbf{AC}^0_d$-Frege systems, in which lines are restricted to be Boolean circuits of unbounded fan-in and constant depth d. We also consider more generally *bounded-depth Frege systems*, where the depth d is bounded, but not necessarily a constant.

For bounded-depth Frege systems, we have strong lower bounds available. The most famous such lower bound is the one for the *Pigeonhole Principle* (PHP). For every $m \in \mathbb{N}$ and $n \in \mathbb{N}$ such that $m > n$, the formula PHP^m_n stands for the CNF formula over variables $p_{i,j}$ for $i \in [m]$ and $j \in [n]$ consisting of the clauses

$$\bigvee_{j \in [n]} p_{i,j} \qquad \text{for all } i \in [m], \tag{PHP-1}$$

$$\neg p_{i,j} \vee \neg p_{i,j'} \quad \text{for all } i \in [m] \text{ and } j, j' \in [n], j \neq j' \tag{PHP-2}$$

$$\neg p_{i,j} \vee \neg p_{i',j} \quad \text{for all } i, i' \in [m], i \neq i' \text{ and } j \in [n] \tag{PHP-3}$$

We sometimes denote by PHP_n the formula PHP^{n+1}_n.

Strong lower bounds are known on the proof complexity of the pigeonhole principle for bounded-depth Frege systems [68]–[70]. Here we state only a simplified version of the best such lower bound, proven by Håstad [33].

Theorem II.1 ([33])**.** *For every $d \leq O(\log n / \log\log n)$, it holds that depth-$d$ Frege systems require size at least $\exp\big(\Omega(n^{1/(4d-2)})\big)$ to prove $\neg\mathrm{PHP}_n$.*

Finally, we often consider extensions of Extended Frege by sets of additional axioms. For a set $A \subseteq \textsc{Taut}$ of tautologies that is recognizable in polynomial time, the system $\mathsf{EF} + A$ refers to Extended Frege extended with the axiom schemas that allow (formula) substitution instances of any formula in A.

3) Automatability and proof search: The notion of of automatability, introduced by Bonet, Pitassi, and Raz [1], is a natural formalization of efficient proof search in propositional proof system. We say that a proof system S is *automatable* if there exists a constant $c \in \mathbb{N}$ and an algorithm that given a propositional tautology φ, outputs an S-proof of φ in time $(|\varphi| + \mathsf{size}_S(\varphi))^c$, meaning that the proof search algorithm succeeds in finding a proof of size polynomial in the size of the shortest one.

Even when a system might not be automatable, it seems natural to ask whether there exists a system Q that p-simulates S and is itself automatable. In this case, we say that S is *weakly automatable* [22]. Weak automatability is equivalent to the existence of an automating algorithm where the output proof belongs to a system $Q \geq_{\mathrm{p}} S$ rather than S itself. In particular, weak automatability is closed downwards under p-simulation.

A more restrictive notion of proof search is given by the *Proof Size Problem*. Associated to any propositional proof system S we can define the *Proof Size Problem for S* (PSP_S), defined as the language

$$\mathrm{PSP}_S \coloneqq \{(\varphi, 1^s) \mid \text{there is an } S\text{-proof of } \varphi \text{ in size } s\}.$$

Automating S entails approximating minimum proof-size to a polynomial, in polynomial time.

B. Bounded arithmetic

We heavily rely on the connection between propositional proof complexity and (weak) theories of arithmetic. We assume familiarity with basic knowledge of first-order logic and introduce the main theories we are concerned with.

1) The theories PV_1 and S^1_2: Theories of bounded arithmetic capture various forms of feasible reasoning and act as a uniform counterpart of propositional proof systems. The main tool to capture feasibility in mathematical reasoning is to bound the complexity of formulas over which one can apply induction.

a) Cook's PV_1: Cook's theory PV_1 [71], [72] is an attempt to make precise the idea of polynomial-time reasoning. It is a universal theory whose vocabulary $\mathcal{L}_{\mathsf{PV}}$ consists of a function symbol for each polynomial-time function, and the axioms are precisely the recursive definitions of these functions via composition and limited recursion on notation, in the style of Cobham's functional definition of $\mathbf{FP}$ [73]. The theory further admits induction on quantifier-free formulas, which define precisely polynomial-time predicates.

The formal definition of PV_1 is rather technical and the details are not particularly relevant to our proofs, so we refer the reader to Krajíček's textbook [62, Definition 5.3] for the details. The reason we rarely care about the technicalities of PV_1 is that we often work instead in the theory S^1_2 of Buss, which happens to be conservative over PV_1 for the classes of formulas we are interested in. We discuss this next.

b) Buss's S^1_2: We see S^1_2 as a theory sitting in between Robinson's Arithmetic Q and Peano Arithmetic PA. Let $\mathcal{L}_{\mathsf{PA}}$ denote the language of Peano Arithmetic, $\mathcal{L}_{\mathsf{PA}} \coloneqq \{0, 1, +, \cdot, <\}$. The axioms of PA consist first of the axioms of Robinson's arithmetic Q, which define the basic behavior of the symbols

of $\mathcal{L}_{\mathsf{PA}}$ (see, for example, [60, §7.4.3] for a definition), together the *Induction Scheme*

$$(\varphi(0) \wedge \forall x(\varphi(x) \to \varphi(x+1)) \to \forall x \varphi(x), \qquad (\text{IND}_\varphi)$$

available for every formula φ.

The language of S^1_2 is the first-order language of bounded arithmetic, $\mathcal{L}_{\text{BA}} := \{0, 1, +, \cdot, <, |\cdot|, \lfloor \cdot/2 \rfloor, \#\}$. This extends the language of Peano Arithmetic $\mathcal{L}_{\mathsf{PA}}$ above by the symbols $|x|$, $\lfloor x/2 \rfloor$ and $x\#y$. The standard interpretation of $\lfloor x/2 \rfloor$ is clear. The notation $|x|$ denotes the length of the binary encoding of the number x, $\lceil \log(x+1) \rceil$, while the *smash symbol* $x\#y$ stands for $2^{|x| \cdot |y|}$.

For a term t in the language of bounded arithmetic and a variable x that does not appear in t, a formula of the form $\forall x(x < t \to \varphi(x))$ or $\exists x(x < t \wedge \varphi(x))$ is called a *bounded formula*. The quantifiers guarded by the bounds on x are called *bounded quantifiers* and we simply write $\forall x < t(\varphi(x))$ and $\exists x < t(\varphi(x))$. If the bounded quantifiers are of the form $\forall x < |s|$ of $\exists x < |s|$ for some term s, then they are called *sharply bounded* quantifiers. The *hierarchy of bounded formulas* consists of the classes Σ^b_n (and Π^b_n), for $n \geq 1$, which are defined by counting the alternations of bounded quantifiers ignoring the sharply bounded ones, starting with an existential (respectively, universal) one. The class Δ^b_n consists of all formulas that admit an equivalent definition in both Σ^b_n and Π^b_n. In particular, the class Δ^b_0 stands for all formulas with sharply bounded quantifiers only.

The theory S^1_2 of Buss [74] extends Robinson's arithmetic Q by a set BASIC of simple axioms for the new function symbols (see, e.g., [62, Definition 5.2.1] for the complete list). On top of this, the theory has the *Polynomial Induction* scheme (PIND) for Σ^b_1-formulas: for every $\varphi \in \Sigma^b_1$, the theory contains the axiom

$$\varphi(0) \wedge \forall x(\varphi(\lfloor x/2 \rfloor) \to \varphi(x)) \to \forall x \varphi(x). \qquad (\text{PIND}_\varphi)$$

When working over S^1_2, we often invoke instead the schema for *Length Induction*,

$$\varphi(0) \wedge \forall x(\varphi(x) \to \varphi(x+1)) \to \forall x \varphi(|x|), \qquad (\text{LIND}_\varphi)$$

made available for all Σ^b_1-formulas. This form of induction is provable from (PIND_φ) for $\varphi \in \Sigma^b_1$ [62, Lemma 5.2.5].

Unlike $\mathcal{L}_{\mathsf{PV}}$, the language $\mathcal{L}_{\text{BA}}$ of bounded arithmetic does not contain a function symbol for every function in **FP**. However, every such $f \in \mathbf{FP}$ is Σ^b_1-definable in S^1_2, meaning that there exists a Σ^b_1 formula whose interpretation over the standard model $\mathbb{N}$ defines f and such that S^1_2 proves the totality of this definition. Thus, in the rest of the paper we choose to use the theory $\mathsf{S}^1_2(\mathcal{L}_{\mathsf{PV}})$, which is the theory S^1_2 in the language of bounded arithmetic extended by all PV function symbols, meaning that we have a fresh symbol for each function in **FP**, and induction is now available for all $\Sigma^b_1(\mathsf{PV})$ formulas. The theory $\mathsf{S}^1_2(\mathcal{L}_{\mathsf{PV}})$ is fully conservative over S^1_2. In what follows we abuse notation and denote this simply as S^1_2.

The final key fact for us is that all $\forall\Sigma^b_1$ formulas provable in S^1_2 are already provable in PV_1. That is, the theory S^1_2 is $\forall\Sigma^b_1$-conservative over PV_1 [74]. We use this in some of the formalizations, where we carry out arguments in S^1_2 but later appeal to its proof in PV_1. For a proof of this fact, see, for example Krajíček's textbook [62, Thm. 5.3.4 and Cor. 7.2.4].

2) Cook's propositional translation: Following Krajíček [60, §8.6], we say that a theory of arithmetic T *corresponds* to a propositional proof system S if (i) T can prove the soundness of S and (ii) every universal consequence $\forall x \varphi(x)$ of T, where φ is quantifier-free, admits polynomial-size proofs in S when suitably grounded into a sequence of propositional formulas. (Pudlák [21] alternatively says that S is a *weak system* of the theory T.)

We are interested in the correspondence between PV_1 and Extended Frege (EF). In this case, the process used to turn first-order formulas into propositional ones is known as (Cook's) *propositional translation*, introduced in his seminal paper on PV [71]. Given a quantifier-free formula $\varphi(x)$, Cook's translation is a polynomial-time construction sending φ to a sequence of polynomial-size propositional formulas $\{[\![\varphi]\!]_n\}_{n \in \mathbb{N}}$ such that for every $n \in \mathbb{N}$, the formula $[\![\varphi]\!]_n \in \text{TAUT}$ if and only if $\mathbb{N} \models \varphi(n)$. See [60, §12.3] or [63, §6.1] for a complete definition of the construction. Cook then observed that, under this translation, PV_1 and EF do indeed correspond to each other.

Theorem II.2 (Cook's correspondence theorem [71])**.** *The theory PV_1 and the proof system Extended Frege correspond to each other. That is,*

(i) PV_1 proves the soundness of EF;

(ii) if $\varphi(x)$ is a quantifier-free formula in the language $\mathcal{L}_{\text{BA}}(\mathsf{PV})$ and $\mathsf{PV}_1 \vdash \forall x \varphi(x)$, then there exists a polynomial-time computable function f such that for every $n \in \mathbb{N}$, it holds that $f(1^n) : \mathsf{EF} \vdash [\![\varphi]\!]_n$.

A proof of the theorem can be found in Krajíček's textbook [60, Theorem 12.4.2].

C. The REF formulas

The main character in this paper is the so-called REF *formula*. Given a propositional CNF formula φ and a size parameter $s \in \mathbb{N}$, the formula $\text{REF}_s(\varphi)$ states that φ has a Resolution refutation consisting of at most s clauses.

It is important to choose an encoding that is simultaneously natural from a modeling point of view while not making the formulas artificially hard to refute. At a basic level, the main property that such an encoding should satisfy is that for a concrete φ and s, the formula $\text{REF}_s(\varphi)$ should be satisfiable if and only if there is a Resolution refutation of φ in at most s clauses. It also seems natural to require that a Resolution refutation should be readable in polynomial time from a satisfying assignment to REF.

While different encodings have appeared in the literature, they tend to agree on a few basic ideas. The formula $\text{REF}_s(\varphi)$ consists of s so-called *blocks* of variables, each representing a clause in the purported Resolution refutation. Each block has variables to represent the literals that appear in this block, how it was obtained (resolved or weakened from an axiom), and

it contains *pointer variables* to indicate from which blocks it was derived.

a) The unary encoding of Pudlák: Pudlák [5] uses the seemingly most standard encoding, which we refer to as the *unary encoding* for REF. He used it to prove that the canonical pair of Resolution is symmetric. This encoding employs pointers in unary, meaning that for every block $B \in [s]$, there are up to s additional variables to point at the blocks from which B was derived.

b) The relativized unary encoding of Atserias and Müller: Atserias and Müller [13] start by studying Pudlák's encoding. They proved suitable so-called *index-width* lower bounds for it in Resolution, but they were unable to prove a *size* lower bound for it. They then introduced a *relativized* version, in which each block can be possibly *enabled* or *disabled*. If it is disabled, then the block is not used towards the refutation, and its associated clauses are immediately satisfied. These additional *enabling variables* made it possible to prove the size lower bound from the index-width lower bound for the unrelativized encoding. We refer to this second encoding as the *relativized unary encoding*.

We note, however, that the change of encoding is not the source for hardness. Garlík [41] proved that even when using the original encoding of Pudlák, the formulas are hard for Resolution whenever the underlying CNF is unsatisfiable.

c) The binary encoding of de Rezende et al.: In their alternative proof of the lower bound on REF formulas, de Rezende, Göös, Nordström, Pitassi, Robere, and Sokolov [18] introduce an encoding of REF where pointers are encoded in binary. Informally, for every block $B \in [s]$, there are $O(\log s)$ variables used encode the value $B' \in [s]$ of the block(s) from which B was derived. We refer to this as the *binary encoding*. While this encoding also includes the enabling variables of the relativized encoding, these are inessential, since one can always assign the pointers in a dummy fashion to effectively disable a block.

We contend that the unary relativized encoding is both the most natural as well as the most versatile. We see three reasons for this:

1) thanks to the enabling variables, one can naturally turn a Resolution refutation of $t < s$ clauses into a satisfying assignment to $\text{REF}_s(\varphi)$ simply by disabling $s - t$ blocks that are not needed, while in the relativized encoding one needs to fill in the remaining $s - t$ clauses with some dummy content;
2) the enabling variables make the random restriction argument leading to the size lower bound much simpler to prove, and Garlík has shown that the hardness does not comes from this change in syntax;
3) when using a unary encoding rather than the binary one of de Rezende, Göös, Nordström, Pitassi, Robere, and Sokolov, one can easily restrict some pointers to get an instance of $\text{REF}_t(\varphi)$ for every $t < s$, while in the binary encoding, after disabling a block, the binary pointers might still be able to point to it, making the formulas more delicate to handle after applying a restriction.

We remark that the choice between unary and binary encodings is ultimately inessential, and all the results in this paper can be reproven for the binary encoding. We choose the unary encoding mainly for reason (3) above, which simplifies the write-up.

We now define the formula in detail.

d) The variables of $\text{REF}_s(\varphi)$*:* Here, we assume φ is a CNF formula over n variables $x_1, \dots, x_n$ and m clauses and define the following variables, where $\text{Lit}_n \coloneqq \{x_1, \dots, x_n, \neg x_1, \dots, \neg x_n\}$ and each variable has the following intended meaning:

- $\mathsf{a\text{-}lit}_\ell^A$: literal $\ell \in \text{Lit}_n$ is present in the clause $A \in [m]$ of φ;
- enable^B: block $B \in [s]$ is enabled;
- $\mathsf{derived}^B$: block $B \in [s]$ is obtained by a Resolution step;
- weak_A^B: block $B \in [s]$ is obtained by weakening from clause $A \in [m]$ of φ;
- lit_ℓ^B: literal $\ell \in \text{Lit}_n$ is present in the block $B \in [s]$;
- $\mathsf{res}_{x_i}^B$: block $B \in [s]$ is obtained by resolving over the variable x_i;
- $\mathsf{lpoint}_{B'}^B$: block $B \in [s]$ is resolved on the left from block $B' \in [s]$, $B' < B$;
- $\mathsf{rpoint}_{B'}^B$: block $B \in [s]$ is resolved on the right from block $B' \in [s]$, $B' < B$.

Building on these variables, the $\text{REF}_s(\varphi)$ formula is defined as follows. We write the clauses as implications for the sake of readability.

Definition II.3 (The REF formulas)**.** Let $n, m, s \in \mathbb{N}$, and let $\text{Lit}_n \coloneqq \{x_1, \dots, x_n, \neg x_1, \dots, \neg x_n\}$ denote the of possible literals over n variables. The $\text{REF}_s(\varphi)$ formula is built from the variables defined above, together with the conjunction of the following clauses:

For every $B, B' \in [s], B' < B, i \in [n], \ell \in \text{Lit}_n, \ell \neq x_i$,
$$\left(\mathsf{enable}^B \wedge \mathsf{res}_{x_i}^B \wedge \mathsf{lpoint}_{B'}^B \wedge \mathsf{lit}_\ell^{B'}\right) \to \mathsf{lit}_\ell^B; \qquad \text{(REF-1)}$$
for every $B, B' \in [s], B' < B, i \in [n], \ell \in \text{Lit}_n \ell \neq \neg x_i$,
$$\left(\mathsf{enable}^B \wedge \mathsf{res}_{x_i}^B \wedge \mathsf{rpoint}_{B'}^B \wedge \mathsf{lit}_\ell^{B'}\right) \to \mathsf{lit}_\ell^B; \qquad \text{(REF-2)}$$
for every $B \in [s], A \in [m], \ell \in \text{Lit}_n$,
$$\left(\mathsf{enable}^B \wedge \mathsf{weak}_A^B \wedge \mathsf{a\text{-}lit}_\ell^A\right) \to \mathsf{lit}_\ell^B; \qquad \text{(REF-3)}$$
for every $B \in [s]$,
$$\left(\mathsf{enable}^B \wedge \mathsf{derived}^B\right) \to \bigvee_{i \in [n]} \mathsf{res}_{x_i}^B; \qquad \text{(REF-4)}$$
for every $B \in [s]$,
$$\left(\mathsf{enable}^B \wedge \mathsf{derived}^B\right) \to \bigvee_{\substack{B' \in [s] \\ B' < B}} \mathsf{lpoint}_{B'}^B; \qquad \text{(REF-5)}$$
for $B \in [s]$,
$$\left(\mathsf{enable}^B \wedge \mathsf{derived}^B\right) \to \bigvee_{\substack{B' \in [s] \\ B' < B}} \mathsf{rpoint}_{B'}^B; \qquad \text{(REF-6)}$$
for every $B \in [s]$,

$$\left(\mathsf{enable}^B \wedge \neg\mathsf{derived}^B\right) \to \bigvee_{A\in[m]} \mathsf{weak}^B_A; \qquad \text{(REF-7)}$$

for every $B, B' \in [s], B' < B$,

$$\left(\mathsf{enable}^B \wedge \mathsf{lpoint}^B_{B'}\right) \to \mathsf{enable}^{B'}; \qquad \text{(REF-8)}$$

for every $B, B' \in [s], B' < B$,

$$\left(\mathsf{enable}^B \wedge \mathsf{rpoint}^B_{B'}\right) \to \mathsf{enable}^{B'}; \qquad \text{(REF-9)}$$

for every $\ell \in \mathrm{Lit}_n$,

$$\neg\mathsf{lit}^s_\ell; \qquad \text{(REF-10)}$$

$$\mathsf{enable}^s. \qquad \text{(REF-11)}$$

Remark II.4. Our encoding of REF has fewer axioms than that of [13]. For example, we do not require that if a block B is resolved on the left by variable x from block B', then B' should contain x, or we do not require that for every resolution step there is a unique resolved variable. We remark that soundness still holds and REF is satisfiable if and only if there exists a refutation of length at most s, which can easily be read from the satisfying assignment to the REF formula. We also remark that the lack of these axioms does not affect the extraction algorithm or the lower bound in any way: while removing axioms could in principle make the lower bound easier to prove, the algorithm works just as well if we added the missing axioms, and our lower bound proof still goes through with the additional axioms. This more succinct encoding, however, makes it easier to formalize the upper bound construction in Resolution.

Remark II.5 (Number of variables). The formula $\mathrm{REF}_s(\varphi)$ is defined over $N = \Theta(s^2 + sm + sn + mn)$ variables and $M = \Theta(s^2n^2 + smn)$ clauses. For the case when the a-lit variables are restricted to encode a k-CNF formula over n variables and $s = n^c$ for some constant $c \geq 1$, we have $m = O(n^k)$ and $N = O(n^{\max\{2c,c+k\}})$.

e) Blocks and block-width: If a variable is part of a block B_i, we say that it *mentions* B_i. An important measure for us will be the *block-width* of a given clause C over the variables of $\mathrm{REF}_s(\varphi)$. This is defined as the number of different blocks mentioned by the variables of the literals in C, not counting the root block $B_\perp$. We denote this measure by $\mathrm{bw}(C)$, and generalize it to refutations by taking $\mathrm{bw}(\pi)$ to be the maximum block-width over all the clauses in π.

III. The Proof Analysis Problem: definitions and basic facts

For a CNF formula $\varphi(x_1, \ldots, x_n)$, we denote by $\mathrm{REF}_s(\varphi)$ the propositional formula claiming that there exists a Resolution refutation of φ in size s. Different encodings of this formula have been considered in the literature. For our purposes, REF consists of s of *blocks* of variables, each of them describing a clauses in a purported Resolution refutation of φ of size s. (See Section II-C for a full rendering of the variables and clauses involved in $\mathrm{REF}_s(\varphi)$.)

We are interested in the following decision problem.

Definition III.1 (The Proof Analysis Problem, PAP_Q)**.** Let Q be a propositional proof system. We define the *Proof Analysis Problem for* Q to be the language

$$\mathrm{PAP}_Q \coloneqq \{(\varphi, \pi, 1^s) \mid \varphi \in \mathrm{SAT} \text{ and } \pi : Q \vdash \neg\mathrm{REF}_s(\varphi)\}.$$

We denote by $\mathrm{PAP}_Q[s(n)]$ the problem where the size parameter s is restricted to be at least $s(n)$ and n denotes the number of variables of φ.

The problem asks, given the proof of a Resolution lower bound in a fixed proof system Q, to decide whether the underlying formula is satisfiable or not. Note that whenever φ is satisfiable there is no Resolution refutation and thus any lower bound holds, so the problem is well-defined.

Analogous to the notion of whether a proof system is automatable, PAP naturally induces a notion of whether, for a given proof system, its Resolution lower bounds are "analyzable".

Definition III.2 (Analyzability)**.** We say that a propositional proof system Q is *analyzable* if there exists some constant $c > 0$ such that $\mathrm{PAP}_Q[n^c] \in \mathbf{P}$.

Remark III.3. It might seem more intuitive to define a proof system Q to be analyzable if $\mathrm{PAP}_Q \in \mathbf{P}$, without restrictions on the size parameter. Note, however, that for most reasonable proof systems, the language PAP_Q taken as a whole contains some degenerate instances that make the problem trivially **NP**-hard. For example, if the size parameter is set to $s = 1$, then certainly proving a Resolution lower bound against φ is easy already for Resolution itself, and we can map a CNF formula φ to the PAP_Q-instance $(\varphi, \pi, 1)$ for some easy to construct Q-proof π that checks there is no Resolution refutation of φ in one clause.

It is easy to see that for every Cook-Reckhow system Q, the problem PAP_Q is in **NP**. Similarly, we note that unlike automatability, analyzability is naturally downwards-closed under p-simulations. Namely, if S is p-simulated by Q and Q is analyzable, so is S; this is not the case with automatability, where a search algorithm for Q may not be used to search for proofs in a weaker S.

The following is a corollary of the results of Atserias and Müller [13]. Here, by **P**-uniform we mean the standard notion of uniformity by which there is a polynomial-time descriptor Turing machine that on input 1^ℓ outputs the circuit solving the problem for inputs of size ℓ (see, e.g., [75, Definition 6.12] or [76]).

Proposition III.4. *It holds that* $\mathrm{PAP}_{\mathsf{Res}}[n^2]$ *is in* **P***-uniform* $\mathbf{AC}^0$*. That is, Resolution is analyzable.*

Proof. Let us first describe the general polynomial-time algorithm that puts $\mathrm{PAP}_{\mathsf{Res}}[n^2]$ in **P**, and we later elaborate on how this can be computed in **P**-uniform $\mathbf{AC}^0$. Indeed, by the Resolution lower bound on REF formulas (Theorem I.5), there exists $\varepsilon > 0$ such that for every $s \in \mathbb{N}$, if a formula φ over n variables is unsatisfiable, then a correct Resolution refutation π of $\mathrm{REF}_s(\varphi)$ must have size $|\pi| > 2^{\varepsilon \cdot s/n}$. Given

an input $(\varphi, \pi, 1^s)$ to $\text{PAP}_{\mathsf{Res}}[n^2]$, to decide if the instance belongs in the language, it suffices to check (i) that π is a correct Resolution refutation of $\text{REF}_s(\varphi)$ and (ii) that $|\pi|$ is smaller than the lower bound $2^{\varepsilon \cdot s/n}$. If (i) fails, we immediately reject, and otherwise, if (ii) fails, the input size is large enough to brute-force SAT in polynomial time. Here we use the fact that $s \geq n^2$, hence $|\pi| \geq 2^{\varepsilon \cdot s/n} \geq 2^{\varepsilon n}$, and thus the input size is large enough.

Let us now argue that this entire computation is possible within $\mathbf{P}$-uniform $\mathbf{AC}^0$. For the sake of precision, let us fix the following natural binary encoding for $\text{PAP}_{\mathsf{Res}}$. An input $(\varphi, \pi, 1^s)$ will be of the form $(1^n, 1^m, C_1, \ldots, C_m, 1^t, \pi, 1^s)$. Here, the first part of the tuple corresponds to the encoding of φ, a CNF formula over n variables and m clauses $C_1, \ldots, C_m$, and we assume that these clauses are initially represented as strings of length $2n$ with indicators for every possible literal. The Resolution refutation of $\text{REF}_s(\varphi)$ is encoded by 1^t and π, where π is an assignment to the $N := N(n, m, t, s) = \text{poly}(n, m, t, s)$ variables of $\text{REF}_t(\text{REF}_s(\varphi))$, as per Section II-C, and the number of variables N can be easily computed in polynomial time. For the purpose of unique decoding, we assume that the tuple $(1^n, 1^m, C_1, \ldots, C_m, 1^t, \pi, 1^s)$ is encoded by bit-doubling: each bit is duplicated and 01 is used as separators. We assume that there are no separators between the clauses $C_1, \ldots, C_m$, so a correct input contains only five separators.

Now, when dealing with binary strings of even length ℓ, there are at most $O(\ell^4)$ possible ways of interpreting such strings as a tuple of the form $(1^n, 1^m, C_1, \ldots, C_m, 1^t, \pi, 1^s)$. This is because we can choose values for n, m, t and s in the interval $[\ell/2 - 5]$, where $\ell/2 - 5$ comes from the fact that we duplicated every bit and introduced 10 bits for the five separators between 1^n, 1^m, $C_1, \ldots, C_m$, 1^t, π, and 1^s, not counting separators between C_1 and C_m. One can then check that this choice of n, m, t and s conforms to the desired pattern: the segment for the clauses C_1 to C_m has length exactly $2nm$, and the segment for π has length exactly $N = N(n, m, t, s)$, as per Remark II.5. That is, it must hold that $\ell = 2(n + m + 2nm + t + N(n, m, t, s) + s) + 10$ and $s \geq n^2$. There are at most $O(\ell^4)$ such choices for $(n, m, t, s) \in [\ell/2 - 5]^4$, hence the upper bound. Furthermore, it is easy to see that, due to the bit-doubling, every string can only encode correctly one input of the form $(1^n, 1^m, C_1, \ldots, C_m, 1^t, \pi, 1^s)$, so the decoding is unique.

The $\mathbf{P}$-uniform descriptor machine for inputs of even length ℓ now works as follows. On input 1^ℓ, for ℓ even, it tries all possible $O(\ell^4)$ ways of separating the lengths, and for each interpretation (n, m, t, s) of the lengths it constructs a different constant-depth Boolean circuit $D_{n,m,t,s}$, as follows.

(a) If the interpretations of the lengths is inconsistent, in the sense that the string cannot correspond to something of the form $(1^n, 1^m, C_1, \ldots, C_m, 1^t, \pi, 1^s)$, then it outputs the constant circuit 0.

(b) If the interpretation of the lengths is valid and $|\pi| < 2^{\varepsilon \cdot s/n}$, then it simply constructs the circuit that checks that π is a correct Resolution refutation of $\text{REF}_s(\varphi)$ in at most t clauses; that is, it outputs the formula $\text{REF}_t(\text{REF}_s(\varphi))$, which itself depends on the variables encoding φ. Since REF formulas are in CNF, nesting these together with φ will result in a total depth of 5.

(c) If the interpretation of the lengths is valid and $|\pi| \geq 2^{\varepsilon \cdot s/n}$, then the descriptor outputs the conjunction of two circuits: one is the same as before, checking the correctness of π as a refutation of $\text{REF}_s(\varphi)$ in at most t clauses, and the other is the trivial circuit of size $\text{poly}(n, m) \cdot 2^n$ that brute-forces the satisfiability of φ. More formally, this is a big disjunction of fan-in 2^n, where each wire goes to the formula $\text{SAT}(\varphi, \alpha)$ for different hard-wired values of $\alpha \in \{0, 1\}^n$. Since $\text{SAT}(\varphi, \alpha)$ is a CNF formula, this brute-forcing circuit has depth 3, and combined with the circuit checking the correctness of π, the entire circuit has depth 5 in this case.

For each interpretation of the lengths there is also a circuit $\text{Correct}_{n,m,t,s}$ that verifies that the input correctly encodes a $\text{PAP}_{\mathsf{Res}}$ instance of the right size. This amounts to checking that the separators are in the right place and the double-bit encoding is correctly implemented, which can all be verified in depth 3.

Finally, the descriptor machine outputs the circuit

$$R_\ell := \bigvee_{(n,m,s,t) \in [\ell/2-5]^4} D_{n,m,t,s} \wedge \text{Correct}_{n,m,t,s} \qquad (1)$$

consisting of the disjunction of all the circuits above for every interpretation of the lengths.

The final circuit R_ℓ correctly computes $\text{PAP}_{\mathsf{Res}}[n^2]$ on inputs of even length ℓ, has depth 6 and polynomial size. Indeed, the constructions (a) and (b) above both have polynomial size, and whenever we construct the exponential-size circuit in (c), it is with respect to a segment of the string that has itself size exponential in n. Finally, the descriptor machine runs in polynomial time given only the length ℓ of the input string x, so we can conclude that $\text{PAP}_{\mathsf{Res}}[n^2]$ is in $\mathbf{P}$-uniform $\mathbf{AC}^0$. □

The fact that $\text{PAP}_{\mathsf{Res}}$ is so easy makes it natural to ask whether the same is true for the *search version* of the problem.

Definition III.5 (Search version of PAP)**.** For a propositional proof system Q, we denote by FPAP_Q the *search version of the Proof Analysis Problem for* Q, defined as follows. Given a CNF formula φ, a size parameter s in unary and a proof π such that $\pi : Q \vdash \neg\text{REF}_s(\varphi)$, output either a satisfying assignment for φ, if $\varphi \in \text{SAT}$, or 0 otherwise. Similarly to $\text{PAP}_Q[s(n)]$, we define $\text{FPAP}_Q[s(n)]$ to be the search problem where the size parameter is at least $s(n)$.

If we impose a polynomial upper bound on the size of π, then by the lower bound on REF formulas, there is always a satisfying assignment for φ, and the problem $\text{FPAP}_{\mathsf{Res}}[n^c]$ for any $c > 0$ is in $\mathbf{TFNP}$. The fact that $\text{PAP}_{\mathsf{Res}}[n^2] \in \mathbf{P}$, does not, however, directly imply that $\text{FPAP}_{\mathsf{Res}}[n^c] \in \mathbf{FP}$ for any $c > 0$. Namely, it is not clear that given a polynomial-size proof π of $\text{REF}_s(\varphi)$ one can extract a satisfying assignment of φ, even if one can conclude that φ is satisfiable. We show

in Section IV that $\text{FPAP}_{\mathsf{Res}}$ is in $\mathbf{FP}$—although the algorithm is not quite as straightforward as the one for the decision problem.

We see PAP and the analyzability of a proof system as closely related to automatability. The following proposition captures this idea and underlines the relevance on PAP in showing hardness of automatability.

Proposition III.6. *Let $Q \geq \mathsf{Res}$. If Q is both analyzable and automatable, then $\mathbf{P} = \mathbf{NP}$.*

Proof. If Q is analyzable and automatable, this means that $\text{PAP}_Q[n^c] \in \mathbf{P}$ for some constant $c > 0$, and that there is an automating algorithm A for Q. We call the polynomial-time algorithm for $\text{PAP}_Q[n^c]$ an *analyzer* and observe that these two combined can solve 3SAT in polynomial time as follows. Given a 3-CNF formula φ, construct the formula $\text{REF}_{n^c}(\varphi)$, stating that φ does not have Resolution refutations of size n^c. Since $Q \geq \mathsf{Res}$, by the upper bound construction as in [5, Theorem 4.1] or [13, Lemma 11], whenever φ is satisfiable, there will be size-$n^{O(1)}$ refutations in Resolution, and hence also in Q, and the automating algorithm A will succeed in finding some refutation in polynomial time. Feed this refutation to the Q-analyzer to decide whether $\varphi \in \text{SAT}$. If the automating algorithm failed to output a polynomial-size proof, then we would already know that $\varphi \notin \text{SAT}$. □

The previous proposition can be seen as an abstract way of stating the $\mathbf{NP}$-hardness of automating Resolution too. Since $\text{PAP}_{\mathsf{Res}}[n^2] \in \mathbf{P}$, that means that Resolution cannot be automatable unless $\mathbf{P} = \mathbf{NP}$. In Section V we study the possibility of analyzing algorithms for strong proof systems actually leading to the hardness of their automatability—and establish that this is highly unlikely.

IV. The extraction algorithm

This section proves that the search version of the Proof Analysis Problem for Resolution is in $\mathbf{FP}$. Our algorithm (in fact, two algorithms) arise from closely observing the lower bound on the REF formulas and attempting to make it fully constructive, in a style that would make them amenable to formalization in weak theories of arithmetic like PV_1 (in the style of Cook and Pitassi [35]).

Recall that the lower bound can be presented in two steps: first, a random restriction argument takes a small refutation and produces a low block-width refutation of a restricted formula, followed by a block-width lower bound for this restricted formula, which overall bounds the size of the original refutation.

Our algorithm works analogously. On input a refutation π of $\text{REF}_s(\varphi)$, it first finds a restriction ρ such that $\pi_{\restriction\rho}$ is a refutation of a restricted version of $\text{REF}_s(\varphi)$ and has low block-width. Then, we have a second algorithm that, inspired by the proof of the width lower bound, analyses this low block-width refutation and extracts a satisfying assignment.

We present the algorithm in two steps. First, two alternatives to perform block-width reduction are described in Section IV-A. These correspond, respectively, to a random restriction and a deterministic restriction argument similar to the one in the proof of the lower bound for the REF formulas. In Section IV-B we explain how to design the algorithm that analyzes low block-width refutations, which is essentially the Prover-Delayer strategy behind the block-width lower bound for the REF formulas. Putting them together yields the desired procedure.

A. The block-width reduction algorithm

Recall that we crucially assume that in our definition of the REF formula there is a variable enable^i for every block $i \in [s]$ that allows us to *disable* that block (see Definition II.3). For succinctness, we often denote enable^i simply by e_i and refer to is as an *enabling variable*.

Let us first define a kind of restriction that will come up a lot in our arguments.

Definition IV.1 (Disabling restrictions)**.** We say that a restriction $\rho \in \{0, 1, *\}^N$ to the N variables of $\text{REF}_s(\varphi)$ is *d-disabling* if it satisfies that (i) exactly d blocks are disabled, and the rest are all enabled, (ii) every variable belonging to a disabled block is assigned a value, and (iii) no other variable is assigned.

A key property of disabling assignments is that they can never falsify any axioms of $\text{REF}_s(\varphi)$, all the enabling variables disappear after the restriction and $\text{REF}_s(\varphi)_{\restriction\rho}$ is essentially an instance of $\text{REF}_{s-d}(\varphi)$, except there are some pointer variables pointing to the d disabled blocks that are still hanging.

A first approach to perform block-width reduction in inspired by random restriction arguments, and requires randomness.

Lemma IV.2 (Randomized block-width reduction)**.** *Let $p \in [0, 1)$ and let N be the number of variables of the $\text{REF}_s(\varphi)$ formula for some CNF formula φ over n variables. There exists a randomized algorithm R taking as input 1^N, and outputting an $\lfloor s/2 \rfloor$-disabling restriction $\rho \in \{0, 1, *\}^N$, such that for every Resolution refutation π of the formula $\text{REF}_s(\varphi)$, the following properties hold:*

(i) the restriction ρ does not falsify $\text{REF}_s(\varphi)$;
(ii) with probability at least p, the block-width of $\pi_{\restriction\rho}$ is at most $O\left(\log|\pi| - \log(1-p)\right)$;
(iii) the running time of $R(1^N)$ is $O(N)$.

Proof. The proof follows closely the random restriction argument of [18, Section 6.3]. For simplicity, let us assume s is even. (Note that, without loss of generality s can be even, since if π is a correct refutation of $\text{REF}_s(\varphi)$ and s is odd, then π can be turned into a refutation of essentially $\text{REF}_{s-1}(\varphi)$ by hitting π with the restriction that disables and fully restricts one block).

Assume the root block corresponds to block $B_\perp$, which is not counted towards block-width, and consider the following random restriction: pair all s blocks into $s/2$ pairs, and for each pair, with probability $1/2$, decide which block in the pair is going to be disabled. Now, if a block is disabled, all of its remaining variables are assigned uniformly at random. We denote by ρ the restriction obtained in this way, which the algorithm outputs.

We claim that with probability at least p, the restriction succeeds in lowering the block-width of any Resolution derivation π to $O(\log|\pi| - \log(1-p))$. Indeed, if ℓ is a literal corresponding to the variable e_i determining whether a certain block is disabled, then $\Pr_\rho[\ell_{\restriction\rho} = 1] = 1/2$. For every other literal ℓ in a block B_i,

$$\Pr_\rho[B_i \text{ is disabled and } \ell_{\restriction\rho} = 1\ ] \tag{2}$$

$$= \Pr_\rho[B_i \text{ is disabled}] \cdot \Pr_\rho[\ell_{\restriction\rho} = 1 \mid B_i \text{ is disabled}] \tag{3}$$

$$= \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}\,. \tag{4}$$

Hence, for every literal ℓ not from $B_\perp$, $\Pr_\rho[\ell_{\restriction\rho} = 1] \geq 1/4$.

Now, if C is a clause of block-width at least w, we have that

$$\Pr_\rho[C_{\restriction\rho} \neq 1] \leq (3/4)^{w/2}\,, \tag{5}$$

where the $1/2$ in the exponent comes from the fact that if two consecutive blocks are present, meaning that they were paired together and only one of them was enabled, their values depend on each other.

Then, if π was indeed a Resolution derivation of $\textsc{Ref}_s(\varphi)$, by a union bound,

$$\Pr_\rho[\mathrm{bw}(\pi_{\restriction\rho}) \geq w] \leq \mathrm{length}(\pi)\cdot(3/4)^{w/2} \leq |\pi|\cdot(3/4)^{w/2}, \tag{6}$$

which is the failure probability for property (ii) in the statement. For success probability at least p, we want to choose w such that $|\pi|\cdot(3/4)^{w/2} \leq 1-p$. This bound is met by choosing $w \geq 2(\log(|\pi|/(1-p))/(\log 4/3))$, meaning that with probability p, the restriction ρ will satisfy all clauses of at least this width.

It suffices to argue that properties (i) and (iii) are also satisfied. Indeed, by the way we designed the restriction, after applying ρ there are no disabling variables left and all variables in the disabled blocks have been restricted, to this is exactly $s/2$-disabling.

As for the running time, the algorithm is simply sampling the restriction, which takes time $O(N)$. □

We now move on to a fully deterministic algorithm that takes as input an actual refutation π and outputs a restriction that *always* manages to reduces the block-width.

Lemma IV.3 (Deterministic block-width reduction)**.** *There exists a constant $c > 0$ and a deterministic algorithm taking as input a Resolution refutation π of the formula $\textsc{Ref}_s(\varphi)$ over N variables and outputting a d-disabling restriction $\rho \in \{0,1,*\}^N$ with $d \leq c/2 \cdot \left(\sqrt{s\log|\pi|}\right)$ such that*

(i) the restriction ρ does not falsify $\textsc{Ref}_s(\varphi)$;

(ii) the block-width of $\pi_{\restriction\rho}$ is at most $c \cdot \left(\sqrt{s\log|\pi|}\right)$;

(iii) the algorithm runs in time $\mathrm{poly}(|\pi|, s)$.

Proof. We employ a greedy strategy to construct the restriction, meaning that we look at all the clauses of high block-width and we iteratively choose to restrict a literal that kills a significant fraction of these clauses.

More formally, let w be a parameter to be optimized later, and given π, let W denote the set of all clauses in π with block-width at least w. Through the following iterative process we will enable and disable some blocks. Whenever we enable a block, we will also mark it as not active by keeping track of a set $\mathsf{ActiveBlocks} \subseteq [s]$, meaning that when choosing greedily the next literal to restrict, inactive blocks are not a valid choice; and whenever we disable a block, we add it to a set $D \subseteq [s]$ to keep track of it.

Algorithm IV.1: Deterministic block-width reduction

Repeat the following procedure iteratively, starting with $\rho \coloneqq \emptyset$, $\mathsf{ActiveBlocks} \coloneqq [s]$, and $D \coloneqq \emptyset$, and stop whenever W is empty:

1) Find the most frequent block $i \in \mathsf{ActiveBlocks}$ among the ones mentioned in the clauses in W.
2) Look at the literal e_i of the variable used to disable block i.
   a) If e_i appears positively in at least $1/3$ of all the clauses in W that mention block i, then set $\rho \coloneqq \rho \cup \{e_i \mapsto 1\}$, $\mathsf{ActiveBlocks} \coloneqq \mathsf{ActiveBlocks} \setminus \{i\}$, $W \coloneqq W_{\restriction\rho}$, and go back to step (1).
   b) If e_i does not appear positively in at least $1/3$ of the clauses in W mentioning block i, then set $\rho \coloneqq \rho \cup \{e_i \mapsto 0\}$, $W \coloneqq W_{\restriction\rho}$, $D \coloneqq D \cup \{i\}$, and for every other variable x of block i,
      i) if x appears in W positively more often than negatively, then set $\rho \coloneqq \rho \cup \{x \mapsto 1\}$, and $W \coloneqq W_{\restriction\rho}$;
      ii) if x appears in W negatively more often than positively, then set $\rho \coloneqq \rho \cup \{x \mapsto 0\}$, and $W \coloneqq W_{\restriction\rho}$;
      iii) repeat for every variable of block i.
   c) Once all the variables of block i have been taken care of, go back to step (1).

The procedure terminates once W is either empty or all clauses in W mention only blocks that are no longer in $\mathsf{ActiveBlocks}$. At this point, $|D|$ blocks have been disabled. For the remaining blocks that were not mentioned by any clause in W, enable all of them by setting the corresponding variables $e_i \mapsto 1$. This completes the construction of the restriction ρ, and the algorithm outputs ρ.

The procedure runs for at most s iterations, since each iteration takes care of one of the blocks mentioned by the clauses in the initial W and we never deal with a block twice. Therefore, the algorithm runs in time $\mathrm{poly}(|\pi|, s)$.

As for the correctness of the algorithm, the restriction ρ is d-disabling by construction for $d = |D|$. It is left to argue that for a suitable choice of w, there exists a constant $c > 0$ such that $d \leq c/2(\sqrt{s\log|\pi|})$ and the block-width of $\pi_{\restriction\rho}$ is at most $c \cdot (\sqrt{s\log|\pi|})$.

We want to choose w so that after $\ell \leq s$ iterations, the set W becomes empty. At the first iteration, by an averaging argument, we know that the most frequent block is mentioned

in at least a w/s fraction of $|W|$. More generally, at iteration ℓ, block-width might have decreased up to $w-(\ell-1)$ and up to $\ell-1$ blocks may have become inactive, so the same averaging argument tells us that the most frequent active block is mentioned in at least a $(w-(\ell-1))/(s-(\ell-1))$ fraction of the clauses. Furthermore, observe that if at a given iteration block i is the most frequent active block, we are not promised to kill all the clauses mentioning i, but we are guaranteed to kill at least $1/3$ of them. Indeed, if we enable block i that is because it appeared in at least $1/3$ of all the clauses in W mentioning i; and otherwise we are guaranteed to restrict at least $1/2$ of the remaining at least $2/3$ fraction of the clauses in W mentioning i, which amounts to at least $1/3$ fraction.

Therefore, if ρ_ℓ is the restriction built after ℓ iterations,

$$|W_{\restriction \rho_\ell}| \leq |W| \cdot \left(1-\frac{w}{3s}\right) \cdot \dots \cdot \left(1-\frac{w-(\ell-1)}{3(s-(\ell-1))}\right) \quad (7)$$

$$\leq |W| \cdot \left(1-\frac{w-\ell}{3s}\right)^{\ell} \quad (8)$$

$$\leq |W| \cdot e^{-\ell \cdot \frac{w-\ell}{3s}} . \quad (9)$$

We want to ensure that for some $\ell \leq s$ we achieve $|W_{\restriction \rho_\ell}| < 1$. It suffices to have $|W| \cdot e^{-\ell \cdot \frac{w-\ell}{3s}} < 1$. Taking logarithms on both sides we have

$$\ln |W| < \ell \cdot \frac{w-\ell}{3s} , \quad (10)$$

which holds already for $\ell = \lfloor w/2 \rfloor$, assuming that we have $w > \sqrt{12s \ln |\pi|} \geq \sqrt{12s \ln |W|}$.

Now it suffices to choose a constant c such that $w \coloneqq c \cdot \sqrt{s \log |\pi|} > \sqrt{12s \ln |\pi|}$. In this way we get that after at most $\ell \coloneqq \lfloor w/2 \rfloor$ iterations, $W_{\restriction \rho_\ell} = \emptyset$ and thus the block-width of $\pi_{\restriction \rho_\ell}$ is also at most $c \cdot \sqrt{s \log |\pi|}$. Furthermore, note that $|D| \leq \ell$, since the algorithm only runs for at most ℓ iterations, meaning that ρ_ℓ is d-disabling for $d \leq \ell = \lfloor w/2 \rfloor \leq (c/2)\sqrt{s \log |\pi|}$, as desired. □

B. The block-width analysis algorithm

Using one of the two algorithms above, we can take a refutation π of $\text{REF}_s(\varphi)$ and obtain a new refutation π' of the restricted formula $\text{REF}_s(\varphi)_{\restriction \rho}$ in low block-width. We can now show how to analyze this refutation, inspired by the block-width lower bound, and succeed in finding a satisfying assignment whenever one exists.

We first state the following simple but crucial technical fact used in the proof.

Fact IV.4. *Let φ be CNF formula over n variables. If C is a non-tautological width-n clause over the variables of φ that is not the weakening of any clause of φ, then the unique assignment that falsifies C is a satisfying assignment for φ.*

Now we can present the algorithm and prove its correctness.

Lemma IV.5 (Assignment extraction)**.** *There exists a deterministic algorithm E such that for every $s \in \mathbb{N}$, every π a purported Resolution refutation of $\text{REF}_s(\varphi)$ for a CNF formula $\varphi(x_1, \dots, x_n)$ with m clauses, and every $\rho \in \{0,1,*\}^N$ a d-disabling restriction to the N variables in $\text{REF}_s(\varphi)$, it holds that $E(\varphi, \rho, s, \pi)$ terminates in time $\text{poly}(|\pi|, s, n, m)$ and provides exactly one of the following outputs:*

(a) an incorrect derivation step in π;
(b) a clause $C \in \pi_{\restriction \rho}$ of block-width at least $1/3 \cdot \lfloor (s-d-n)/n \rfloor$;
(c) a satisfying assignment for φ.

Proof. We traverse the refutation π inspired by the Delayer's strategy in the Prover-Delayer game that yields a block-width lower bound for the restricted REF formulas—and the correctness of the algorithm is essentially the proof of this block-width lower bound.

Before the traversal of π starts, we arrange the $s-d$ blocks enabled by ρ in a layered manner, so that there are n layers, each containing $\lfloor (s-d)/n \rfloor$ blocks (with the remainder blocks left from the flooring operations collected all in the last layer), plus one additional layer on top with a single block corresponding to the root. We see the root as laying at layer 0, and intuitively blocks in layer i will be obtained by resolving over variable x_{i+1}. Throughout the traversal, we keep a record $\alpha \in \{0,1,*\}^N$ which we call the *reservation*, in which we "reserve" information needed to continue the traversal. (Intuitively, this record keeps the information that the Delayer has in mind when playing against the Prover.) A reservation is a partial assignment to the variables of $\text{REF}_s(\varphi)$, and hence encodes patches of a potential refutation. In particular, it will encode information about how blocks are connected: say, some block B will be derived from block B', meaning that the reservation encodes that the left pointer of B' is B, etc. Hence, when talking about a reservation, we will often say that a certain block has *information about their parents* or *information about their children*, to mean that such a connection is registered in the reservation.

The algorithm proceeds as follows. First, let $\pi \coloneqq \pi_{\restriction \rho}$, initialize $\alpha \coloneqq \rho$, and collect in a set $D \subseteq [s]$ the d blocks that are disabled by the restriction ρ. Note that ρ is a d-disabling restriction, meaning that it only enables and disables blocks, and sets the value of all the variables in disabled blocks, meaning that the restriction cannot possibly falsify any axiom of $\text{REF}_s(\varphi)$. Note as well that in π, after hitting it with ρ, there are no longer resolution steps over enabling variables nor over variables belonging to a disabled block.

In what follows, capitals letters in roman font, like A, refer to clauses in the refutation π, while capital letters in calligraphic font, like $\mathcal{A}$, refer to clauses encoded in the blocks of $\text{REF}_s(\varphi)$ which are determined by assignments to the variables of $\text{REF}_s(\varphi)$.

Algorithm IV.2: Block-width analysis and assignment extraction

Let C be the root of π and traverse the proof dag following these instructions.

1) If C is obtained by an illegal derivation step, halt

and output C; if C is a leaf of π, halt and output failure. Otherwise, continue to Step 2.

2) If C was obtained from weakening a clause $C' \subseteq C$, then set $C \coloneqq C'$ and move to Step 4.
3) If C was derived from clauses $A \vee v$ and $B \vee \neg v$ by resolving over v, attempt the following reservations according to these rules (with the condition that blocks mentioned in the set D can never be reserved).
   a) If v belongs to a block on layer $0 \leq i < n$, where layer 0 stands for the root block, we have two cases:
      i) if α has no information about this block, update α by reserving two unreserved blocks on layer $i+1$ so that the block of v encodes the clause $\bigvee_{j=1}^{i} x_j$ and it was obtained by resolving the clauses $x_{i+1} \vee \bigvee_{j=1}^{i} x_j$ and $\neg x_{i+1} \vee \bigvee_{j=1}^{i} x_j$, to be encoded in the two reserved blocks on layer $i+1$;
      ii) if the block was reserved in α but it had no children attached, then α already determined the clause $\mathcal{C}$ that is to be encoded in this block. Then, update α by reserving two unreserved blocks on layer $i+1$ and so that that the block of v encodes the clause $\mathcal{C}$ and it was obtained by resolving the clauses $x_{i+1} \vee \mathcal{C}$ and $\neg x_{i+1} \vee \mathcal{C}$, to be encoded in the two reserved blocks on layer $i+1$.
      iii) Otherwise, do nothing.

      If this reservation fails because there are not enough free blocks available, halt and output the clause C.
   b) If v belongs to a block on layer n, we distinguish two cases:
      i) if the block is not mentioned in α, then try to find the first axiom in φ such that $\bigvee_{j=1}^{n} x_j$ is a weakening of it, and reserve it so it encodes $\mathcal{A} \coloneqq \bigvee_{j=1}^{n} x_j$ and its pointers point to this axiom;
      ii) if the block was reserved in α but it was not pointing to any axiom, then there is a clause $\mathcal{A}$ associated to it by the reservation; try to find the first clause in φ such that $\mathcal{A}$ is a weakening of it and point to it in α.
      iii) Otherwise, do nothing.

      If this reservation fails because no axiom could be found, then halt and output the assignment to the variables $x_1, \ldots, x_n$ given by $\neg\mathcal{A}$, the negation of the clause encoded by $\mathcal{A}$.

   If the reservations succeeded, look at $\alpha(v)$, which is now guaranteed to be defined. If $\alpha(v) = 1$, then move to $C \coloneqq B \vee \neg v$, and otherwise move to $C \coloneqq A \vee v$.
4) Clean-up the reservations in α as follows: α should only contain information about (i) blocks disabled by ρ and (ii) blocks mentioned in C or possibly the children of these according to α. Erase all other information from α.
5) Go to Step 1.

Since the algorithm is only traversing a path inside the proof dag of π, the running time of this procedure is never longer than a polynomial in the size of π. As for the correctness of the algorithm, we now show that this behaves exactly as claimed. The central claim is that, at the beginning of each iteration, when looking at clause C, the following invariant is satisfied. Here, $\mathrm{bw}(\alpha)$ stands for the number of blocks mentioned by the variables assigned by α, and $\mathrm{bw}(C)$ is the block-width of a clause C.

Claim (Invariant). The following hold at the beginning of each iteration of the algorithm, when dealing with the clause C in the traversal of π:

(i) the reservation α falsifies C;
(ii) a block $B \notin D$ is only reserved in α if it is either mentioned in C or its parent according to α is mentioned in C, and, in particular, $\mathrm{bw}(\alpha) - d \leq 3\,\mathrm{bw}(C)$;
(iii) if α encodes any information about a block B from layer i, and $B \notin D$, then α also determines that B contains exactly i literals over the variables $x_1, \ldots, x_i$ and no two literals for the same variable;
(iv) the reservation α does not falsify any axiom of $\mathrm{REF}_s(\varphi)$.

Proof sketch. The invariant is readily verified at the initial iteration, when $C = \bot$ and $\alpha = \rho$, the d-disabling restriction given as input.

Now, by straightforward structural induction, it is easy to see that assuming that the invariant holds at the beginning of an iteration and the algorithm correctly proceeds to the next iteration without halting, the invariant holds again at the beginning of the new iteration. □

Note that, if the algorithm reached a clause C that happened to be a leaf of π, then by point (i) of the invariant α would be falsifying C, which would mean falsifying an axiom of $\mathrm{REF}_s(\varphi)$, contradicting point (iv) of the very same invariant. Thus, if π is a correct Resolution refutation, that means that the algorithm always halts before reaching a leaf, and always for one of the following two reasons.

(a) The algorithm attempted the reservation of a block at level $1 \leq i < n$, but there were no free blocks left. This means that α already reserved at least $\lfloor (s-d)/n \rfloor - 1$ blocks on that layer, and so $\mathrm{bw}(\alpha) \geq \lfloor (s-d)/n \rfloor - 1 + d$, since each layer $i < n$ contains exactly $\lfloor (s-d)/n \rfloor$ blocks. By point (ii) of the invariant we have that that $\mathrm{bw}(\alpha) - d \leq 3\,\mathrm{bw}(C)$, so putting this together we have that when outputting C we are outputting a clause of block-width at least $1/3\lfloor (s-d-n)/n \rfloor$, as desired.
(b) The reservation α had a clause $\mathcal{A}$ encoded in a block at layer n, but it failed to find an axiom of φ that $\mathcal{A}$ was a weakening of. By point (iii) of the invariant, since the block is at layer n, it encodes a width-n clause, and by

Fact IV.4, $\neg\mathcal{A}$ encodes a satisfying assignment of φ. In this case the algorithm outputs this assignment, which satisfies the desired behavior.

This completes the proof of correctness of the algorithm. □

C. Putting it together

The following is a formal restatement of Theorem I.1.

Theorem IV.6. *It holds that* $\mathrm{FPAP}_{\mathsf{Res}}[n^3] \in \mathbf{FP}$. *That is, there exists a deterministic polynomial-time algorithm solving the search version of the Proof Analysis problem for Resolution whenever the lower bound parameter satisfies* $s \geq n^3$.

Proof. Let $(\varphi(x_1, \ldots, x_n), \pi, 1^s)$ be an instance of the Proof Analysis Problem with $s \geq n^3$. First, check whether π is indeed a correct Resolution refutation of $\textsc{Ref}_s(\varphi)$. If not, reject. Otherwise, run the algorithm from Lemma IV.3 on π, which outputs a d-disabling restriction ρ, with $d \leq (c/2)\sqrt{s \log |\pi|}$, such that $\pi_{\restriction\rho}$ is a Resolution refutation of $\textsc{Ref}_s(\varphi)_{\restriction\rho}$ of block-width at most $c \cdot \sqrt{s \log |\pi|}$, for some fixed constant c. Then run the extraction algorithm from Lemma IV.5 on π and ρ. Since π is a correct refutation, it must be the case that the extraction algorithm from Lemma IV.5 outputs either a clause of $\pi_{\restriction\rho}$ of block-width at least $(s - d - n)/3n$ or a satisfying assignment of φ. In the latter case, we are done. In the former case, it holds that $1/3\lfloor (s - d - n)/n \rfloor \leq c \cdot \sqrt{s \log |\pi|}$, which implies $|\pi| > 2^{\varepsilon s/n^2} \geq 2^{\varepsilon \cdot n}$ for some small enough ε and sufficiently large n. Since π is so large we can, in time polynomial in the size of the input, go over all 2^n assignments to the variables of φ and output a satisfying assignment of φ if one exists, and otherwise reject. □

If we are interested in inputs where the lower bound is quadratic instead of cubic, then we can still achieve polynomial time at the cost of randomness.

Theorem IV.7. *There exists a zero-error randomized polynomial-time algorithm solving the search version of the Proof Analysis problem for Resolution whenever the lower bound parameter satisfies* $s \geq n^2$. *That is,* $\mathrm{FPAP}_{\mathsf{Res}}[n^2]$ *is in* $\mathbf{FZPP}$.

Proof. We carry out the proof for a fixed constant success probability p, but the argument works for any $p < 1$. The algorithm is essentially the same as before, except we now use the randomized width-reduction procedure in Lemma IV.2 at the beginning, instead of the greedy deterministic one.

Observe that the randomized algorithm in Lemma IV.2 can be used with zero-error, because once a restriction ρ is sampled, we can check if it successfully reduces the block-width to the desired bound, and run it again as many times as needed, which puts us in $\mathbf{FZPP}$.

As for why s can now be allowed to be n^2, observe that the randomized procedure achieves better width reduction, of $O(\log |\pi|)$ whenever p is a constant. Combining this with the $(s - d - n)/3n$ bound of block-width given by Lemma IV.5, which in this case becomes $s/6n - 1/3$ because $d = s/2$, we now have $|\pi| > 2^{\Omega(s/n)}$, meaning that $s \geq n^2$ suffices to obtain an exponential lower bound. □

As discussed in introduction, the deterministic algorithm in Theorem IV.6 gives us a Levin reduction between 3SAT and the Proof Size Problem for Resolution ($\mathrm{PSP}_{\mathsf{Res}}$, see Section II-A3). Here the search version of 3SAT if the one that finds satisfying assignments of satisfiable formulas, while the search version of $\mathrm{PSP}_{\mathsf{Res}}$ consists in finding a Resolution refutation of the right size.

Corollary IV.8. *There is a polynomial-time Levin reduction from the search problem for* 3SAT *to the Proof Size Problem for Resolution.*

Proof. In [13], a 3-CNF formula φ is mapped to the formula $\textsc{Ref}_{n^2}(\varphi)$. If, instead, we map it to $\textsc{Ref}_{n^3}(\varphi)$, then this is still a many-one reduction from 3SAT and hardness of automatability still follows, but this is now a Levin reduction: given a satisfying assignment for φ, we can always come up with a short Resolution refutation of $\textsc{Ref}_{n^3}(\varphi)$ using the standard upper bound; and given a Resolution refutation π of $\textsc{Ref}_{n^3}(\varphi)$ of polynomial-size, we can extract a satisfying assignment for φ using Theorem IV.6. □

V. $\mathrm{PAP}_{\mathsf{EF}}$ IS $\mathbf{NP}$-COMPLETE

In light of the extraction algorithm in Theorem IV.6, it is natural to ask whether stronger proof systems are also analyzable. As shown in Proposition III.6, the existence of a polynomial-time proof analysis algorithm for a proof system S implies that automating S is $\mathbf{NP}$-hard. Could this be the route towards proving that automating systems like Extended Frege is $\mathbf{NP}$-hard?

This turns out to be unlikely. While for Resolution $\mathrm{PAP}_{\mathsf{Res}}[n^2] \in \mathbf{AC}^0$ and $\mathrm{FPAP}_{\mathsf{Res}}[n^3] \in \mathbf{FP}$, already the decision problem for Extended Frege $\mathrm{PAP}_{\mathsf{EF}}$ turns out to be $\mathbf{NP}$-complete. This extends to every proof system S that p-simulates Extended Frege: it is $\mathbf{NP}$-complete to decide whether a formula φ is satisfiable given an S-proof of a Resolution lower bound on φ. In the full version [59], we further build on this $\mathbf{NP}$-hardness result to investigate whether finding polynomial-time analysis algorithms (for weaker systems where they exist) requires proving proof complexity lower bounds first.

The idea of the hardness proof is to show that Extended Frege can prove Resolution lower bounds for certain formulas encoding instances of the VERTEX COVER problem. The lower bound will be "agnostic" in the sense that it will not depend on whether the underlying VERTEX COVER instance is satisfiable or not. This means that an algorithm that distinguishes between "true Resolution lower bounds" (those where the underlying formula is unsatisfiable) from "trivial ones" (those proven for satisfiable formulas) will be able to decide VERTEX COVER and hence all of $\mathbf{NP}$.

The formulas in question come from a convenient encoding of VERTEX COVER in a way that embeds a pigeonhole principle. We define these next.

Definition V.1 (The VERTEX COVER formulas)**.** Let $G = (V, E)$ be a graph on n nodes and let k be a positive integer such that $k \leq n$. The CNF formula $\mathrm{VC}(G, k)$ has variables

$$\{v_1, \ldots, v_n\} \cup \{p_{i,j} \mid i \in [n], j \in [k]\},$$

and clauses

$$\neg v_i \vee \bigvee_{j=1}^{k} p_{i,j} \qquad i \in [n] \qquad \text{(VC-1)}$$

$$v_i \vee v_{i'} \qquad (i, i') \in E \qquad \text{(VC-2)}$$

$$\neg p_{i,j} \vee \neg p_{i,j'} \qquad i \in [n], j, j' \in [k], j \neq j' \qquad \text{(VC-3)}$$

$$\neg v_i \vee \neg v_{i'} \vee \neg p_{i,j} \vee \neg p_{i',j} \quad j \in [k], i, i' \in [n], i \neq i' \qquad \text{(VC-4)}$$

The formula $\mathrm{VC}(G, k)$ is satisfiable if and only if G has a vertex cover of size at most k. The vertices in the cover are given by an assignment to the variables $v_1, \ldots, v_n$ and, to force that there are at most k vertices in the cover, the clauses on the variables $p_{i,j}$ enforce an instance of the pigeonhole principle with one pigeon for each vertex in the cover, and k holes.

Due to the embedded pigeonhole principle these formulas will be hard for Resolution. We show that such a Resolution lower bound is provable already in S^1_2. We need two main ingredients for this. First, the graphs on which we will show hardness will be the graphs obtained from the standard textbook reduction from 3SAT to VERTEX COVER. Second, the proof of the Resolution lower bound in Extended Frege will come from a reduction to Haken's lower bound for the pigeonhole principle. The latter was already formalized by Cook and Pitassi in PV_1.

Theorem V.2 (Cook and Pitassi, 1990 [35])**.** *There exist a positive $\varepsilon_0 \in \mathbb{Q}$ and $n_0 \in \mathbb{N}$ such that*

$$\mathsf{PV}_1 \vdash \forall n \forall \pi (\mathrm{Ref}_{\mathsf{Res}}(\mathrm{PHP}^n_{n-1}, \pi) \wedge n \geq n_0 \rightarrow ||\pi|| > \varepsilon_0 n).$$

Regarding the construction of the graphs, we need to make sure that S^1_2 can prove the correctness of the standard reduction from 3SAT to VERTEX COVER. The proof is the standard textbook construction from 3SAT to CLIQUE and then to VERTEX COVER (as in, for example, [77, §3.1.3]). We state it below but defer the proof to the appendix of the full version of the paper [59].

Lemma V.3 (3SAT $\leq_{\mathrm{p}}$ VERTEX COVER in S^1_2)**.** *There exists a PV function f such that S^1_2 proves the statement that for every 3-CNF formula φ with n variables and m clauses, $f(\varphi)$ outputs a graph $G_\varphi = (V, E)$ with $m \cdot n$ nodes such that the formula φ is satisfiable if and only if G_φ has a vertex cover of size $m \cdot (n-1)$.*

Now, under a suitably crafted restriction, the formula $\mathrm{VC}(G, k)$ for the particular graph $G = G_\varphi$ from Lemma V.3 will become PHP^{n+1}_n, and S^1_2 proves Haken's lower bound, as shown by Cook and Pitassi (Theorem V.2).

Theorem V.4. *For every positive constant $c \in \mathbb{N}$, there exist $n_0 \in \mathbb{N}$ such that, if for every 3-CNF formula φ we write G_φ for the graph obtained in polynomial-time from φ by the reduction f in Lemma V.3, then it holds that*

$$\mathsf{S}^1_2 \vdash \forall \varphi \forall \pi \forall n \leq \varphi \forall m \leq \varphi (n \geq n_0 \wedge \text{3-CNF}(\varphi, n, m) \wedge |\pi| \leq n^c \rightarrow \neg \mathrm{Ref}_{\mathsf{Res}}(\mathrm{VC}(G_\varphi, m(n-1)), \pi)).$$

Proof. Suppose for contradiction that π is indeed a Resolution refutation of $\mathrm{VC}(G, m(n-1))$. Consider the restriction ρ mapping $v_i \mapsto 1$ for all $i \in [mn]$. By inspecting axioms (VC-1) to (VC-4) we get that $\pi_{\restriction_\rho}$ is now a Resolution refutation of $\mathrm{PHP}^{mn}_{m(n-1)}$.

Next, restrict further as follows. Consider the variable substitution ρ' extending ρ as follows: For every $i \in [mn]$ and $j \in [m(n-1)]$, let k, k', r, r' be integers such that $i = kn + r + 1$ and $j = k'(n-1) + r' + 1$ with $0 \leq r < n$ and $0 \leq r' < n - 1$, and set $\rho' : p_{i,j} \mapsto 0$ if $k \neq k'$, and $\rho' : p_{i,j} \mapsto p_{r+1,r'+1}$ if $k = k'$. Let $\pi' \coloneqq \pi_{\restriction_{\rho'}}$. It is now immediate to see that π' is a Resolution refutation of PHP^n_{n-1}. However, by Theorem V.2, $||\pi'|| > \varepsilon_0 n$, implying $||\pi|| > \varepsilon_0 n$. This contradicts the assumption $|\pi| \leq n^c$, when $n \geq n_0$ and n_0 is chosen large enough. □

Corollary V.5. *For every positive constant $c \in \mathbb{N}$, there exists a polynomial-time computable function t such that for every 3-CNF formula φ over a large enough number n of variables and m clauses, $t(\varphi)$ outputs an Extended Frege proof π such that*

$$\pi : \mathsf{EF} \vdash \neg \mathrm{REF}_{n^c}(\mathrm{VC}(G_\varphi, m(n-1))).$$

Proof. The formula in Theorem V.4 is $\forall \Pi^b_1$, so we can apply Cook's translation (Theorem II.2) to get polynomial-size EF proofs of $[\![\neg \mathrm{Ref}_{\mathsf{Res}}(\mathrm{VC}(G_\varphi, m(n-1)), \pi)]\!]$. Extended Frege can then uniformly prove that the REF-like formula obtained from the translation is equisatisfiable to the REF formulas as we have defined them in Definition II.3. We then have that EF proves $\neg \mathrm{REF}_{n^c}(\mathrm{VC}(G_\varphi, m(n-1)),)$ in polynomial size and the proofs can be produced uniformly in polynomial time. □

The previous upper bound in Extended Frege works for every φ, regardless of its satisfiability. This is the key idea behind the final reduction.

Theorem V.6. *For every positive constant $c \in \mathbb{N}$, the language* 3SAT *reduces to* $\mathrm{PAP}_{\mathsf{EF}}[n^c]$ *under polynomial-time many-one reductions.*

Proof. The reduction maps a 3-CNF formula φ over n variables and m clauses to the instance $(\psi, \pi, 1^s)$, where ψ is the VERTEX COVER formula $\psi \coloneqq \mathrm{VC}(G_\varphi, m(n-1))$, together with the Extended Frege proof π given by the map $t(\varphi)$ in Corollary V.5, and the size parameter 1^s is 1^{n^c}.

By Corollary V.5, the proof π is always a correct EF-proof, regardless of the satisfiability of φ. Now, if $\varphi \in$ 3SAT, then G_φ has a vertex cover of size $m(n-1)$ and hence ψ is satisfiable, so $(\psi, \pi, 1^{n^c}) \in \mathrm{PAP}_{\mathsf{EF}}$. On the other hand, if $\varphi \notin$ 3SAT, then G_φ does not have a vertex cover of size $m(n-1)$, so ψ is unsatisfiable. Since π is still a valid EF-proof, we get $(\psi, \pi, 1^{n^c}) \notin \mathrm{PAP}_{\mathsf{EF}}$. This proves that the reduction is correct. □

This yields the following formal restatement of Theorem I.3.

Corollary V.7. *For every propositional proof system Q that p-simulates Extended Frege and every polynomial $s(n)$, the problem $\mathrm{PAP}_Q[s(n)]$ is $\mathbf{NP}$-complete under polynomial-time many-one Levin reductions.*

Proof. Membership in $\mathbf{NP}$ is trivial, since $\mathrm{PAP}_Q \in \mathbf{NP}$ for every Cook-Reckhow system Q. Hardness for $\mathrm{PAP}_{\mathsf{EF}}[s(n)]$ is given by Theorem V.6, and since Q p-simulates EF, this means that an instance $(\psi, \pi, 1^t)$ of $\mathrm{PAP}_{\mathsf{EF}}$ with $t \geq s(n)$ can be turned into an instance $(\psi, \pi', 1^t)$, where π' is the Q-proof obtained by simulating π. □

ACKNOWLEDGMENT

We are indebted to Ján Pich for insightful initial discussions on the problem and particularly for the idea behind the proof of Theorem I.3, originally based on MCSP, as sketched in Section I-B2. We thank Emil Jeřábek [78] for the proof of a lemma in the full version of the paper that helps simulate contraposition in Resolution. We also thank Jonas Conneryd for useful comments and careful proofreading of a draft of this work, and Anupam Das, Stefan Grosser, Antonina Kolokolova, Jan Krajíček, Jakob Nordström, Kilian Risse, and Rahul Santhanam, as well as the anonymous FOCS reviewers, for helpful comments and suggestions.

This collaboration started at the Proof Complexity and Beyond workshop at Mathematisches Forschungsinstitut Oberwolfach, in March 2024. Part of this work was carried out during the Proof Complexity Workshop at the University of Oxford in September of 2024 and during the *Kaleidoscope de la complexité* spring school in April 2025 at the Centre International de Rencontres Mathématiques (CIRM) in Marseille, France. We also gratefully acknowledge that we have benefited greatly from being part of the Basic Algorithms Research Centre (BARC) environment financed by the Villum Investigator grant 54451.

REFERENCES

[1] M. L. Bonet, T. Pitassi, and R. Raz, "On interpolation and automatization for Frege systems," *SIAM J. Comput.*, vol. 29, no. 6, pp. 1939–1967, 2000.

[2] P. Beame and T. Pitassi, "Simplified and improved resolution lower bounds," in *Proceedings of 37th Conference on Foundations of Computer Science*, 1996, pp. 274–282.

[3] J. Krajíček and P. Pudlák, "Some consequences of cryptographical conjectures for S^1_2 and EF," *Information and Computation*, vol. 140, no. 1, pp. 82–94, 1998.

[4] M. L. Bonet, C. Domingo, R. Gavaldà, A. Maciel, and T. Pitassi, "Non-automatizability of bounded-depth Frege proofs," *computational complexity*, vol. 13, pp. 47–68, 2004.

[5] P. Pudlák, "On reducibility and symmetry of disjoint $\mathbf{NP}$ pairs," *Theoretical Computer Science*, vol. 295, no. 1-3, pp. 323–339, 2003.

[6] M. Alekhnovich and A. A. Razborov, "Resolution is not automatizable unless $\mathbf{W}[\mathrm{P}]$ is tractable," *SIAM Journal on Computing*, vol. 38, no. 4, pp. 1347–1363, 2008.

[7] N. Galesi and M. Lauria, "On the automatizability of polynomial calculus," *Theor. Comp. Sys.*, vol. 47, no. 2, p. 491–506, Aug. 2010.

[8] A. Atserias and E. Maneva, "Mean-payoff games and propositional proofs," *Inform. and Comput.*, vol. 209, no. 4, pp. 664–691, 2011.

[9] L. Huang and T. Pitassi, "Automatizability and simple stochastic games," in *Automata, languages and programming. Part I*, ser. Lecture Notes in Comput. Sci. Springer, Heidelberg, 2011, vol. 6755, pp. 605–617.

[10] A. Atserias, "The proof-search problem between bounded-width resolution and bounded-degree semi-algebraic proofs," in *Theory and applications of satisfiability testing—SAT 2013*, ser. Lecture Notes in Comput. Sci., 2013, vol. 7962, pp. 1–17.

[11] A. Beckmann, P. Pudlák, and N. Thapen, "Parity games and propositional proofs," *ACM Trans. Comput. Logic*, vol. 15, no. 2, May 2014.

[12] I. Mertz, T. Pitassi, and Y. Wei, "Short proofs are hard to find," in *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, vol. 132, 2019, pp. 84:1–84:16.

[13] A. Atserias and M. Müller, "Automating resolution is $\mathbf{NP}$-hard," *Journal of the ACM (JACM)*, vol. 67, no. 5, pp. 1–17, 2020.

[14] Z. Bell, "Automating regular or ordered resolution is $\mathbf{NP}$-hard," *Electronic Colloquium on Computational Complexity (ECCC)*, no. TR20-105, 2020. [Online]. Available: https://eccc.weizmann.ac.il/report/2020/105

[15] S. Buss and E. Yolcu, "Regular resolution effectively simulates resolution," *Inform. Process. Lett.*, vol. 186, pp. Paper No. 106 489, 4, 2024.

[16] M. Garlík, "Failure of feasible disjunction property for k-DNF resolution and $\mathbf{NP}$-hardness of automating it," in *39th Computational Complexity Conference, CCC 2024, July 22-25, 2024, Ann Arbor, MI, USA*, vol. 300, 2024, pp. 33:1–33:23.

[17] M. Göös, S. Koroth, I. Mertz, and T. Pitassi, "Automating cutting planes is $\mathbf{NP}$-hard," in *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, 2020, pp. 68–77.

[18] S. F. de Rezende, M. Göös, J. Nordström, T. Pitassi, R. Robere, and D. Sokolov, "Automating algebraic proof systems is $\mathbf{NP}$-hard," in *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, 2021, pp. 209–222.

[19] D. Itsykson and A. Riazanov, "Automating OBDD proofs is $\mathbf{NP}$-hard," in *47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022)*, vol. 241, 2022, pp. 59:1–59:15.

[20] T. Papamakarios, "Depth-d Frege systems are not automatable unless $\mathbf{P} = \mathbf{NP}$," in *39th Computational Complexity Conference (CCC 2024)*, vol. 300, 2024, pp. 22:1–22:17.

[21] P. Pudlák, "Reflection principles, propositional proof systems, and theories," 2020. [Online]. Available: https://arxiv.org/abs/2007.14835

[22] A. Atserias and M. L. Bonet, "On the automatizability of resolution and related propositional proof systems," *Inform. and Comput.*, vol. 189, no. 2, pp. 182–201, 2004.

[23] J. Köbler and J. Messner, "Is the standard proof system for SAT p-optimal?" in *International Conference on Foundations of Software Technology and Theoretical Computer Science*. Springer, 2000, pp. 361–372.

[24] S. A. Fenner, L. Fortnow, A. V. Naik, and J. D. Rogers, "Inverting onto functions," *Inform. and Comput.*, vol. 186, no. 1, pp. 90–103, 2003.

[25] N. Mazor and R. Pass, "Gap MCSP Is Not (Levin) $\mathbf{NP}$-Complete in Obfustopia," in *39th Computational Complexity Conference (CCC 2024)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 300, 2024, pp. 36:1–36:21.

[26] J. Krajíček, "Information in propositional proofs and algorithmic proof search," *J. Symb. Log.*, vol. 87, no. 2, pp. 852–869, 2022.

[27] A. A. Razborov, "Bounded arithmetic and lower bounds in Boolean complexity," in *Feasible mathematics, II (Ithaca, NY, 1992)*, ser. Progr. Comput. Sci. Appl. Logic. Birkhäuser Boston, Boston, MA, 1995, vol. 13, pp. 344–386.

[28] R. Alweiss, S. Lovett, K. Wu, and J. Zhang, "Improved bounds for the sunflower lemma," *Annals of Mathematics*, vol. 194, no. 3, pp. 795–815, 2021.

[29] J. Park and H. T. Pham, "A proof of the Kahn–Kalai conjecture," *J. Amer. Math. Soc.*, vol. 37, pp. 235–243, 2024.

[30] A. Atserias and I. Tzameret, "Feasibly constructive proof of Schwartz-Zippel lemma and the complexity of finding hitting sets," *Electronic Colloquium on Computational Complexity (ECCC)*, no. TR24-174, 2024. [Online]. Available: https://eccc.weizmann.ac.il/report/2024/174/

[31] J. Pich and R. Santhanam, "Learning algorithms versus automatability of Frege systems," in *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, vol. 229, 2022, pp. 101:1–101:20.

[32] K. Iwama, "Complexity of finding short resolution proofs," in *Mathematical foundations of computer science 1997 (Bratislava)*, ser. Lecture Notes in Comput. Sci. Springer, Berlin, 1997, vol. 1295, pp. 309–318.

[33] J. Hå stad, "On small-depth Frege proofs for PHP," in *2023 IEEE 64th Annual Symposium on Foundations of Computer Science—FOCS 2023*. IEEE Computer Soc., Los Alamitos, CA, 2023, pp. 37–49.

[34] J. Håstad and K. Risse, "On bounded depth proofs for Tseitin formulas on the grid; revisited," in *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2022, pp. 1138–1149.

[35] S. Cook and T. Pitassi, "A feasibly constructive lower bound for resolution proofs," *Information Processing Letters*, vol. 34, no. 2, pp. 81–85, 1990.

[36] M. Clegg, J. Edmonds, and R. Impagliazzo, "Using the Groebner basis algorithm to find proofs of unsatisfiability," in *Proceedings of the Twenty-eighth Annual ACM Symposium on the Theory of Computing (Philadelphia, PA, 1996)*. ACM, New York, 1996, pp. 174–183.

[37] E. Ben-Sasson and A. Wigderson, "Short proofs are narrow—resolution made simple," *J. ACM*, vol. 48, no. 2, pp. 149–169, 2001.

[38] A. Atserias and V. Dalmau, "A combinatorial characterization of resolution width," *J. Comput. System Sci.*, vol. 74, no. 3, pp. 323–334, 2008.

[39] A. A. Razborov, "Resolution lower bounds for perfect matching principles," *J. Comput. System Sci.*, vol. 69, no. 1, pp. 3–27, 2004.

[40] N. Thapen, "A tradeoff between length and width in resolution," *Theory Comput.*, vol. 12, pp. Paper No. 5, 14, 2016.

[41] M. Garlík, "Resolution lower bounds for refutation statements," in *44th International Symposium on Mathematical Foundations of Computer Science*, 2019, pp. Art. No. 37, 13.

[42] I. C. Oliveira, "Meta-mathematics of computational complexity theory," *Electronic Colloquium on Computational Complexity (ECCC)*, no. TR25-041, 2025. [Online]. Available: https://eccc.weizmann.ac.il/report/2025/041/

[43] A. Gaysin, "H-coloring dichotomy in proof complexity," *Journal of Logic and Computation*, vol. 31, no. 5, pp. 1206–1225, 04 2021.

[44] M. Müller, "Typical forcings, $\mathbf{NP}$ search problems and an extension of a theorem of Riis," *Annals of Pure and Applied Logic*, vol. 172, no. 4, p. 102930, 2021.

[45] E. Khaniki, "New relations and separations of conjectures about incompleteness in the finite domain," *Journal of Symbolic Logic*, vol. 87, no. 3, pp. 912–937, 2022.

[46] ——, "Nisan-Wigderson Generators in Proof Complexity: New Lower Bounds," in *37th Computational Complexity Conference (CCC 2022)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 234, 2022, pp. 17:1–17:15.

[47] M. Narusevych, "Models of bounded arithmetic and variants of pigeonhole principle," 2022. [Online]. Available: https://arxiv.org/abs/2208.14713

[48] J. Pich and R. Santhanam, "Towards $\mathbf{P} \neq \mathbf{NP}$ from Extended Frege lower bounds," 2023. [Online]. Available: https://arxiv.org/abs/2312.08163

[49] J. Krajíček, "A proof complexity conjecture and the incompleteness theorem," *The Journal of Symbolic Logic*, p. 1–5, 2023.

[50] A. Gaysin, "Proof complexity of CSP," 2023. [Online]. Available: https://arxiv.org/abs/2201.00913

[51] E. Khaniki, "Jump operators, interactive proofs and proof complexity generators," in *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2024, pp. 573–593.

[52] N. Arteche, G. Carenini, and M. Gray, "Quantum automating $\mathbf{TC}^0$-Frege is LWE-hard," in *39th Computational Complexity Conference, CCC 2024, July 22-25, 2024, Ann Arbor, MI, USA*, 2024, pp. 15:1–15:25.

[53] N. Arteche, E. Khaniki, J. Pich, and R. Santhanam, "From proof complexity to circuit complexity via interactive protocols," in *51st International Colloquium on Automata, Languages, and Programming*, 2024, pp. Art. No. 12, 20.

[54] J. Krajíček, "Extended Nullstellensatz proof systems," *Proceedings of the American Mathematical Society*, vol. 152, pp. 4881–4892, 2024.

[55] M. Narusevych, "An independence of the MIN principle from the PHP principle," 2024. [Online]. Available: https://arxiv.org/abs/2406.14930

[56] A. Gaysin, "Proof complexity of universal algebra in a CSP dichotomy proof," 2024. [Online]. Available: https://arxiv.org/abs/2403.06704

[57] R. Santhanam and I. Tzameret, "Iterated lower bound formulas: a diagonalization-based approach to proof complexity," in *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, 2021, pp. 234–247.

[58] J. Li, Y. Li, and H. Ren, "Metamathematics of Resolution lower bounds: A TFNP perspective," *Electronic Colloquium on Computational Complexity (ECCC)*, no. TR24-190, 2024. [Online]. Available: https://eccc.weizmann.ac.il/report/2024/190/

[59] N. Arteche, A. Atserias, S. F. de Rezende, and E. Khaniki, "The proof analysis problem," 2025. [Online]. Available: https://arxiv.org/abs/2506.16956

[60] J. Krajíček, *Proof complexity*. Cambridge University Press, Cambridge, 2019.

[61] P. Hájek and P. Pudlák, *Metamathematics of first-order arithmetic*. Springer-Verlag, Berlin, 1993.

[62] J. Krajíček, *Bounded arithmetic, propositional logic, and complexity theory*. Cambridge University Press, Cambridge, 1995.

[63] S. R. Buss, "Bounded arithmetic and propositional proof complexity," in *Logic of computation (Marktoberdorf, 1995)*, ser. NATO Adv. Sci. Inst. Ser. F: Comput. Systems Sci. Springer, Berlin, 1997, vol. 157, pp. 67–121.

[64] ——, "First-order proof theory of arithmetic," in *Handbook of proof theory*, ser. Stud. Logic Found. Math. North-Holland, Amsterdam, 1998, vol. 137, pp. 79–147.

[65] S. A. Cook and R. A. Reckhow, "The relative efficiency of propositional proof systems," *J. Symbolic Logic*, vol. 44, no. 1, pp. 36–50, 1979.

[66] J. Krajíček, "On the weak pigeonhole principle," *Fundamenta Mathematicae*, vol. 170, no. 1-2, pp. 123–140, 2001.

[67] E. Jeřábek, "Weak pigeonhole principle, and randomized computation," Ph.D. dissertation, Faculty of Mathematics and Physics, Charles University, Prague, 2005.

[68] M. Ajtai, "The complexity of the pigeonhole principle," *Combinatorica*, vol. 14, no. 4, pp. 417–433, 1994.

[69] T. Pitassi, P. Beame, and R. Impagliazzo, "Exponential lower bounds for the pigeonhole principle," *Computational Complexity*, vol. 3, no. 2, pp. 97–140, 1993.

[70] J. Krajíček, P. Pudlák, and A. Woods, "An exponential lower bound to the size of bounded depth Frege proofs of the pigeonhole principle," *Random Structures Algorithms*, vol. 7, no. 1, pp. 15–39, 1995.

[71] S. A. Cook, "Feasibly constructive proofs and the propositional calculus (preliminary version)," in *Seventh Annual ACM Symposium on Theory of Computing (Albuquerque, N.M., 1975)*. Association for Computing Machinery, New York, 1975, pp. 83–97.

[72] J. Krajíček, P. Pudlák, and G. Takeuti, "Bounded arithmetic and the polynomial hierarchy," *Annals of Pure and Applied Logic*, vol. 52, no. 1, pp. 143–153, 1991.

[73] A. Cobham, "The intrinsic computational difficulty of functions," in *Logic, Methodology and Philos. Sci. (Proc. 1964 Internat. Congr.)*. North-Holland, Amsterdam, 1965, pp. 24–30.

[74] S. R. Buss, *Bounded arithmetic*. Bibliopolis, Naples, 1986.

[75] S. Arora and B. Barak, *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[76] E. Allender, "$\mathbf{P}$-uniform vs. $\mathbf{DLOGTIME}$-uniform $\mathbf{AC}^0$," Theoretical Computer Science Stack Exchange, 2025. [Online]. Available: https://cstheory.stackexchange.com/q/55461

[77] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of $\mathbf{NP}$-Completeness*. W. H. Freeman, 1979.

[78] E. Jeřábek, "Modus ponens style inferences in Resolution," Theoretical Computer Science Stack Exchange, 2025. [Online]. Available: https://cstheory.stackexchange.com/q/55062